УДК 004.94

В.В. Шкарупило, асп.,
Р.К. Кудерметов, канд. техн. наук, доц.,
Т.А. Паромова, ст. преподаватель
Запорожский национальный технический университет
vadshkar@yandex.ru

Концептуальная модель процесса автоматизированного синтеза композитных веб-сервисов

Предложена концептуальная модель процесса автоматизированного синтеза композитных веб-сервисов. Модель процесса представлена как последовательность этапов концептуализации, специфицирования, верификации и валидации. Модель композитного веб-сервиса представлена как иерархическая система с функциональными и нефункциональными характеристиками.

Ключевые слова: композитный веб-сервис, формальная спецификация, верификация, валидация, Temporal Logic of Actions, Model Checking, DEVS.

Введение

В настоящее время концепция повторного использования, положенная в основу сервис-ориентированной архитектуры (СОА), является одной из определяющих концепций, рассматриваемых при создании распределенных веб-приложений. Обоснованием тому может служить стремление к уменьшению издержек, связанных с процессом разработки. В качестве компонент систем на основе СОА зачастую выступают веб-сервисы. Существенным преимуществом веб-сервисов является их слабая связанность. Это обуславливает повышение качества создаваемой системы с точки зрения интероперабельности – способности компонент системы взаимодействовать между собой. Совокупность веб-сервисов в составе подобной системы принято называть композицией или композитным веб-сервисом (CWS, Composite Web Service), а сами компоненты системы – атомарными веб-сервисами.

Учитывая, что CWS может представлять систему любой сложности, целесообразно рассматривать CWS как стратифицированную систему. Подобный шаг призван упростить процедуры специфицирования, верификации и валидации (V&V) синтезируемого CWS как последовательные этапы автоматизированного процесса синтеза. В [1], при этом, было отмечено, что проработке вопросов V&V уделяется недостаточное внимание. Под синтезом CWS будем понимать набор приемов, направленных на получение CWS с требуемыми функциональными (Ф) и нефункциональными (НФ) характеристиками.

Наличие формальной спецификации, т.е. формального описания последовательности взаимодействия атомарных веб-сервисов в составе CWS, является необходимым условием автоматизации процесса синтеза CWS. Необходимость этого условия обосновывается требованием к однозначности машинной интерпретации спецификации в контексте автоматизации. В качестве формализма спецификации предлагается использовать формализм TLA (Temporal Logic of Actions), предложенный Л. Лампортом [2]. Выбор данного формализма может быть обоснован следующими его отличительными характеристиками: реализация концепции "behavior" дает возможность описывать допустимые сценарии функционирования исследуемой системы, а соответствующий программный инструментарий TLA Toolbox включает реализацию метода верификации Model Checking (TLC, TLA Checker). Использование метода Model Checking дает возможность в автоматизированном режиме проверять допустимые значения параметров спецификации, допустимые состояния системы, а также отслеживать наличие "тупиков" (deadlocks). Преимущество от использования метода Model Checking в качестве метода верификации раскрывается в возможности его полной автоматизации.

Реализация процедуры верификации позволит ответить на следующий вопрос: "Правильно ли создана формальная спецификация CWS?" (проверка корректности спецификации). Подтверждением этому может служить следующее утверждение [3]: "Ключевой проблемой компьютерной науки является поиск методов проверки того, что разработанное аппаратное и программное обеспечение соответствует своей спецификации…".

Под процедурой валидации CWS будем подразумевать попытку ответа на вопрос: "Правильную ли систему с CWS мы создаем?" (проверка достоверности CWS). Проверку пригодности синтезированного CWS предлагается осуществлять для отдельно заданного случая (с заданными требованиями к Ф- и НФ-характеристикам CWS).

1. Постановка задачи

При планировании создания (синтеза) нового CWS необходимо определить его Ф- и НФ-характеристики [4].

Процесс автоматизированного синтеза CWS предлагается организовать в такой последовательности: спроектировать модель CWS, создать формальную спецификацию этой модели, выполнить верификацию спецификации и подтвердить адекватность модели (реализация процедуры валидации). Назовем данный процесс "концептуализация / специфицирование / V&V".

На этапе концептуализации предлагается стратифицировать модель исследуемой системы (CWS), с целью упрощения процедуры реализации последующего этапа специфицирования. Этап специфицирования заключается в создании формального (машинно-интерпретируемого) описания (спецификации) допустимых сценариев функционирования CWS. Под спецификацией, при этом, будем понимать формальную спецификацию на основе формализма TLA.

Этап V&V (верификации и валидации) заключается в проверке того, что разработанная спецификация соответствует разработанной модели, а подтверждение адекватности методом моделирования (процедура валидации) показывает, что Ф- и НФ-характеристики планируемого CWS удовлетворяют заданным требованиям.

Как следствие, в работе ставится задача исследовать предлагаемую последовательность этапов процесса автоматизированного синтеза CWS, а также технологии и инструментарий, используемые для его реализации.

2. Концептуализация CWS

Рассмотрим CWS как стратифицированную систему. Это поможет упростить последующий этап специфицирования. Идея стратификации сложных иерархических систем заключается в следующем [5, 6]: создается концептуальная модель исследуемой системы; каждая страта представляет отдельный уровень иерархии подсистем в рамках модели. Иерархический подход к построению модели исследуемой системы раскрывается в создании моделей компонент системы, которые затем включаются в состав модели системы.

Назовем исследуемую систему "координатор / вычислители" – частный случай CWS. Поведение подобной системы может быть описано посредством шаблона проектирования "Controller" [7]: "Обычно контроллер должен лишь делегировать функции другим объектам и координировать их деятельность, а не выполнять эти действия самостоятельно.". В качестве примеров компонент действующих систем, модель функционирования которых соответствует шаблону "Controller", можно привести компоненты Grid-инфраструктуры (CE/WNs, Computing Element & Working Nodes), где функцию контроллера (координатора) выполняет компонент CE. Предложенные абстракции согласуются с моделью композиции, описанной в стандарте WS-BPEL [8]. Модель носит название "оркестровка" – модель централизованного координирования веб-сервисов в составе композиции (CWS). Функцию координатора процесса синтеза выполняет компонент BPEL Engine, реализованный в составе соответствующих инструментальных средств (Oracle BPEL Process Manager, ActiveBPEL, Eclipse BPEL Designer и т.п.). Обозначим компонент BPEL Engine как *CRD* (Coordinator, Controller).

Для описания формализма специфицирования используем теоретико-множественный подход. Обозначим множество атомарных веб-сервисов как $AWS = \{aws_i \mid i = \overline{1,m}\}$, $m \in N$, где $aws_i \in AWS$ – атомарные веб-сервисы, доступные для использования. Совокупности некоторых aws_i, необходимых для реализации Ф-характеристик CWS представим в виде подмножеств множества AWS: $\{C_j \mid j = \overline{1,n}\}, n \in N$, где $C_j \subseteq AWS$ – подмножество атомарных веб-сервисов, необходимых для реализации j-й Ф-характеристики CWS.

$$C_j = \{aws_k \mid k = \overline{1,p}\}, p \in N,$$

причем $p \le m$.

Пусть каждый aws_i характеризуется парой (af_i, anf_i), где af_i и anf_i – Ф- и НФ- характеристики aws_i, соответственно. Под af_i будем понимать некоторое Ф-преобразование, осуществляемое над набором входных данных vec_i: $af_i = f_i(vec_i)$. Концептуально под aws_i можем понимать некоторую абстрактную сущность, реализующую функцию $f_i(vec_i)$.

Пусть НФ-характеристика anf_i определяется тройкой (r_i, t_i, c_i), где r_i (response) – время отклика aws_i; t_i (throughput) – пропускная способность сетевого канала, образуемого узлом-отправителем запроса и узлом-получателем (на котором развернут некоторый aws_i); c_i (cost) – значение стоимости выполнения функции $f_i(vec_i)$. Будем, при этом, подразумевать, что значение элемента r_i равняется сумме значений времен, затрачиваемых на передачу запроса (от узла-отправителя – к узлу-получателю) и на реализацию Ф-характеристики некоторым aws_i, развернутом на узле-получателе.

Предположим, что в клиентском запросе указываются требования к Ф- (F_req) и НФ-характеристикам (NF_req) CWS. NF_req, при

этом, определяются тройкой (r_req, t_req, c_req), где элементы тройки представляют требования к времени отклика, пропускной способности канала и стоимости CWS, соответственно.

Положительный ответ на вопрос "Удовлетворяют ли некоторая НФ-характеристика CWS требованиям клиентского запроса?" предлагается давать в случае истинности соответствующих неравенств:

$$r\_req \geq \sum_{i=1}^{n} r_i \ . \qquad (1)$$

$$t\_req \leq min(t_i) \ .$$

$$c\_req \geq \sum_{i=1}^{n} c_i \ .$$

Если рассматривать взаимодействия между некоторыми $aws_k, aws_{k+1} \in C_j$ в составе CWS как последовательные взаимодействия между территориально распределенными вычислительными процессами, осуществляющиеся посредством механизма асинхронного обмена структурированными сообщениями (подобный способ рассмотрения можно считать обоснованным, исходя из положений теории взаимодействующих последовательных процессов Ч. Хоара [9] и аспектов механизма организации обмена сообщениями между компонентами распределенных компьютерных систем, описываемого протоколом SOAP – Simple Object Access Protocol – [10]), логично предположить, что естественной формой представления Ф-характеристики CWS может являться формализм, основанный на принципе суперпозиции функций. Подобный способ получения требуемой Ф-характеристики CWS будем называть агрегированием (рис. 1).

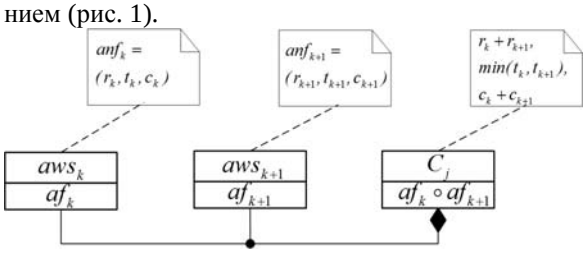

Рисунок 1 – Схема агрегирования j-й Ф-характеристики CWS

Выделим в системе "координатор / вычислители" две страты: St_0 и St_1 (табл. 1).

Таблица 1. Страты системы "координатор / вычислители"

Страты	Назначение компонента	Формальное обозначение
St_0	координатор	CRD
St_1	вычислители	$C_j = \{aws_k\}$

Т.к. функция CRD заключается в координировании атомарных веб-сервисов в составе CWS, под процедурой координирования будем понимать выполнение вызовов некоторых $aws_k \in C_j$ в заданной последовательности. CRD и aws_k будем, при этом, называть элементами соответствующих страт.

Основываясь на введенном формализме, приведем структурную UML-схему стратификации CWS "координатор/вычислители" (рис. 2): под операцией (оператором) "refines" будем понимать реализацию процедуры координирования.

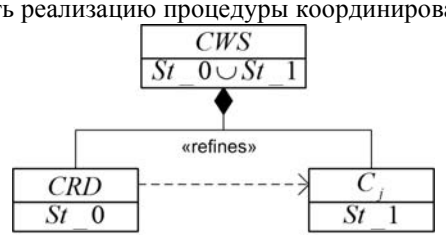

Рисунок 2 – Схема стратификации CWS

Пусть процедура координирования элементов C_j реализуется в рамках некоторого подпроцесса. В теории взаимодействующих последовательных процессов Ч. Хоара предлагается рассматривать всю исследуемую систему как процесс, поведение которого определяется в терминах поведения составляющих его подпроцессов. Поведение подпроцессов, в свою очередь, определяется числом и последовательностью возникновения событий. Пусть изменения состояний рассматриваемой системы происходят в случае наступления событий трех типов: "граничные", "вызов", "результат". Представим эти типы событий в виде соответствующих множеств:

$$REQ = \{req, resp\},$$

где REQ – множество граничных событий, причем req – событие получения координатором CRD запроса с требованиями к Ф- и НФ-характеристикам CWS (будем рассматривать req как начальное событие); $resp$ – конечное событие – отправка результата работы CWS;

$$INVOKE = \{invoke_k\},$$

где $INVOKE$ – множество событий вызова со стороны координатора CRD элементов $aws_k \in C_j$;

$$RES = \{res_k\},$$

где RES – множество событий получения координатором CRD результатов работы элементов $aws_k \in C_j$. Некоторое событие $invoke_k \in INVOKE$ будем рассматривать как побудитель отображения следующего вида:

$$f_k : vec_k \mapsto res_k \ .$$

Будем специфицировать подпроцессы, раскрывающие Ф-характеристики CWS, посредством

соответствующих сценариев. В формализме Ч. Хоара предлагается осуществлять фиксацию событий при помощи протокола – некоторой последовательности обозначений, ассоциированных с событиями. Мы предлагаем вместо понятия "протокол" использовать понятие "сценарий". Это понятие, по нашему мнению, больше согласуется со спецификой рассматриваемой системы, поскольку в модели оркестровки описывается централизованный способ реализации процедуры координирования. Обозначим через S множество сценариев, описывающих динамики (Ф-характеристики) CWS: $S = \left\{ s_j^{CRD} \right\}$.

$$s_j^{CRD} =< invoke_k, ..., res_l >, l = \overline{1, p}. \quad (2)$$

Т.е. каждый s_j^{CRD} описывает способ (сценарий) реализации некоторой Ф-характеристики CWS на основе координирования элементов C_j. Начальная запись сценария соответствует некоторому событию типа "вызов", а конечная – событию типа "результат". Очевидно, что $|S|$ (мощность множества S) равна числу Ф-характеристик CWS.

Согласно теории взаимодействующих последовательных процессов Ч. Хоара, исследуемый объект (система) вначале участвует в некотором событии, а затем ведет себя в точности как процесс (подпроцесс). Формально это предлагается записывать так: $x \to P$, где x, P – некоторые событие и процесс (как последовательность событий), соответственно; '$\to$' – оператор следования; читается как "P за x". Модифицируем этот способ записи, включая в рассмотрение выделенный тип граничных событий. Для этого обозначим через s_{j+1}^{CRD} некоторый альтернативный сценарий, задающий альтернативную Ф-характеристику CWS. Альтернативность будем специфицировать как '|'. Допустимые динамики CWS, при этом можно описать следующим образом:

$$req \to (s_j^{CRD} \mid s_{j+1}^{CRD}) \to resp. \quad (3)$$

Подобный способ специфицирования динамик CWS (3) можно рассматривать как расширенный относительно (2).

Стоит дополнительно отметить, что в [11] предлагается альтернативный способ фиксации (а не задания последовательности) событий. Вместо понятия "протокол" используется понятие "история протекания процесса" (h). Описание h выполняется следующим образом: $e \xrightarrow{h} e'; e, e' \in E$, где E – множество событий, '$\to$' обозначает переходы между событиями, h – упорядоченная совокупность промежуточных событий из E.

Задание последовательности возникновения событий посредством сценариев является, по нашему мнению, более приемлемым решением с точки зрения упрощения процедуры интерпретации сценария в формальную TLA-спецификацию.

3. Пример системы с CWS

Рассмотрим пример частного случая. Пусть предметом рассмотрения является система с CWS, которую можно представить как модификацию системы "координатор / вычислители". В системе с CWS вводится дополнительный актор "Клиент". "Клиент" представлен как генератор граничных событий: формирование запроса – событие req; получение результата работы CWS – событие $resp$.

В качестве сценария предметной области рассмотрим сценарий генерации запросов к системе управления базами данных (СУБД). Типовость данного сценария может быть обоснована распространенностью соответствующих веб-ориентированных программных систем (eBay, newegg и т.п.). Т.к. в качестве СУБД обычно используются решения Oracle или MySQL, пусть набор функций формирования запросов будет представлен в виде следующего множества: $\{select, delete, update\}$, где элементы $select, delete, update$ обозначают функции генерации запросов выбора, удаления и модификации записей таблиц, соответственно. Пусть указанные функции реализуются атомарными веб-сервисами aws_1, aws_2, aws_3, соответственно.

Чтобы модифицировать (удалить) требуемую запись некоторой таблицы, необходимо вначале сформировать запрос $select$ – с целью удостовериться, что выбрана именно требуемая запись; затем, в зависимости от преследуемой конечной цели, выполнить или запрос $delete$, или запрос $update$. Как следствие, видим, что возможным путем автоматизации этой процедуры является синтез CWS, функционирование которого может осуществляться согласно двум сценариям: $req \to (s_1^{CRD} \mid s_2^{CRD}) \to resp$.

Сценарии s_1^{CRD} и s_2^{CRD} раскрывают Ф-характеристики CWS:

$AWS = \{aws_1, aws_2, aws_3\}$;

$C_1 = \{aws_1, aws_2\}$; $C_2 = \{aws_1, aws_3\}$;

$s_1^{CRD} =< invoke\_1, res\_1, invoke\_2, res\_2 >$;

$s_2^{CRD} =< invoke\_1, res\_1, invoke\_3, res\_3 >$.

Представим описанные сценарии в виде UML-диаграммы последовательности взаимодействия (рис. 3).

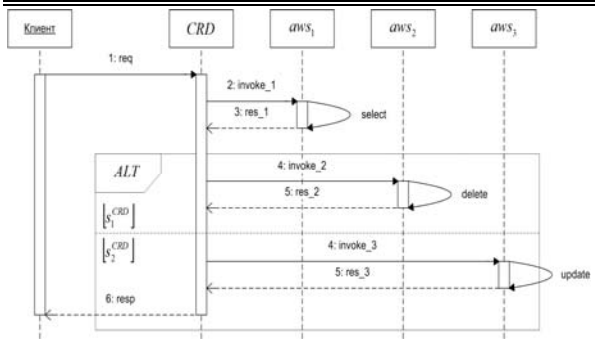

Рисунок 3 – Сценарии функционирования системы с CWS

4. Специфицирование, V&V

Интерпретируем сценарии s_1^{CRD} и s_2^{CRD} в формальную TLA-спецификацию.

Представим записи сценариев в виде последовательности состояний CWS. Для этого введем следующие правила специфицирования совершенности событий:

- инициализация переменных, соответствующих событиям, элементами множества $\{0,1\}$; '0' – событие не наступило; в противном случае – '1';
- используем модификатор (') для обозначения значения переменной, задающего совершенность события в последующий момент времени.

Для специфицирования состояний CWS и их последовательности, введем следующие правила:

- используем оператор конъюнкции (/\) для задания значений переменных, соответствующих событиям, в пределах состояния CWS и для "привязки" текущего состояния – к предыдущему;
- используем оператор дизъюнкции (\/) для задания альтернативности сценариев;
- используем модификатор 'UNCHANGED' для указание, что значение переменной в некотором текущем состоянии не изменилось относительно предыдущего состояния.

TLA-спецификация рассмотренных сценариев (s_1^{CRD} и s_2^{CRD}), созданная с использованием введенных правил интерпретации, приведена в листинге 1.

Листинг 1 – Формальная TLA-спецификация CWS

```
\* оперируем натуральными числами
EXTENDS Naturals
\* переменные, обозначающие события
VARIABLES   req, resp,
       invoke_1, invoke_2, invoke_3,
       res_1, res_2, res_3
\* задание допустимых значений
Def ==  /\ req \in{0,1}
/\ resp \in{0,1}
/\ invoke_1 \in{0,1}
/\ res_1 \in {0,1}
/\ invoke_2 \in{0,1}
/\ res_2 \in {0,1}
/\ invoke_3 \in {0,1}
/\ res_3 \in {0,1}
\* специфицирование состояний CWS:
\* 1 – ни одно из событий не
\* совершилось
Init == /\ req=0  /\ invoke_1=0 /\
res_1=0 /\ resp=0 /\ invoke_2=0 /\
res_2=0 /\ invoke_3=0 /\ res_3=0
\* 2 – поступление запроса со
\* стороны клиента
OnReq == /\ req' = 1 - req
         /\ UNCHANGED <<resp>>
/\ UNCHANGED <<invoke_1, invoke_2,
invoke_3>>
/\ UNCHANGED <<res_1, res_2, res_3>>
\* 3 – вызов координатором aws_1
OnInvoke_1 == /\ OnReq
      /\ invoke_1'=1-invoke_1
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_2, invoke_3>>
/\ UNCHANGED <<res_1, res_2, res_3>>
\* 4 – получение координатором
\* результата вызова aws_1
OnRes_1 == /\ OnInvoke_1
      /\ res_1' = 1 - res_1
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_1, invoke_2,
invoke_3>>
/\ UNCHANGED <<res_2, res_3>>
\* 5 – вызов координатором aws_2
OnInvoke_2 == /\ OnRes_1
      /\ invoke_2' = 1 - invoke_2
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_1, invoke_3>>
/\ UNCHANGED <<res_1, res_2, res_3>>
\* 6 – получение координатором
\* результата вызова aws_2
OnRes_2 == /\ OnInvoke_2
      /\ res_2' = 1 - res_2
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_1, invoke_2,
invoke_3>>
/\ UNCHANGED <<res_1, res_3>>
\* 5 – вызов координатором aws_3
OnInvoke_3 == /\ OnRes_1
      /\ invoke_3' = 1 - invoke_3
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_1, invoke_2>>
/\ UNCHANGED <<res_1, res_2, res_3>>
\* 6 – получение координатором
\* результата вызова aws_3
OnRes_3 == /\ OnInvoke_3
      /\ res_3' = 1 - res_3
      /\ UNCHANGED <<req, resp>>
/\ UNCHANGED <<invoke_1, invoke_2,
invoke_3>>
/\ UNCHANGED <<res_1, res_2>>
\* 7 – отправка клиенту
\* результата работы CWS,
```

```
\* задание альтернативности
\* сценариев
OnResp == (OnRes_2 \/ OnRes_3) /\
resp' = 1 - resp
Spec == Init /\ [OnResp]_<<req,
resp, invoke_1, invoke_2, invoke_3,
res_1, res_2, res_3>>
```

Допустимые состояния CWS специфицированы в листинге 1 под следующими условными обозначениями: Init, OnReq, OnInvoke_1,…, OnRes3, OnResp. Корректность спецификации была проверена посредством использования метода Model Checking (TLC, TLA Checker), интегрированного в среду разработки TLA Toolbox.

Анализируя предложенный способ специфицирования, можно отметить некоторую громоздкость (синтаксическую избыточность) полученной TLA-спецификации CWS. В качестве противоположного (положительного) момента можно указать на наглядность и структурированность TLA-спецификации, полученной посредством использования предложенного набора правил трансляции. Указанными преимуществом и недостатком также может быть охарактеризованы wsdl-описания (Web Services Description Language) атомарных веб-сервисов.

Следующим шагом является реализация процедуры валидации. В нашем случае процедура валидации заключается в проведении дискретно-событийного имитационного моделирования в среде DEVS Suite. Отличительной особенностью DEVS-формализма является концепция "atomic model" [12]. Эта концепция предпочтительна тем, что она позволяет естественным образом представлять иерархические связи (отношения) моделей компонент в составе модели системы с CWS.

Обозначим через $am\_1,...,am\_3$ модели атомарных веб-сервисов $aws_1,...,aws_3$, соответственно. Модель координатора CRD обозначим как am_CRD. Компоненту "Клиент" системы с CWS представим в виде атомарной модели генератора сценариев (или s_1^{CRD}, или s_2^{CRD}), которые затем отсылаются на входные порты модели am_CRD. Обозначим модель компоненты "Клиент" как am_Gen.

Включим модели атомарных компонент в состав модели системы с CWS (cm_CWS). Будем фиксировать моменты появления сообщений на входном и выходном портах (*in* и *out*) модели cm_CWS как моменты наступления граничных событий *req* и *resp*, соответственно.

Перейдем к моделированию исследуемой системы. В качестве НФ-характеристик компонент модели системы с CWS выберем время отклика, мс: $anf_1.r_1 = 30$, $anf_2.r_2 = 40$, $anf_3.r_3 = 35$. Время отклика модели am_Gen установим равное 10 мс, а время, затрачиваемое на реализацию процедуры координирования моделью am_CRD – 50 мс.

Пусть требования к НФ-характеристикам CWS $NF\_req.r\_req = 200$ мс. Наша задача – путем моделирования проверить, удовлетворяет ли модель CWS заданным НФ-требованиям. Удовлетворение требованиям к Ф-характеристикам CWS подтверждается корректностью функционирования модели.

Рассмотрим случай, когда на входной порт модели координатора am_CRD поступил сценарий s_1^{CRD} (рис. 4).

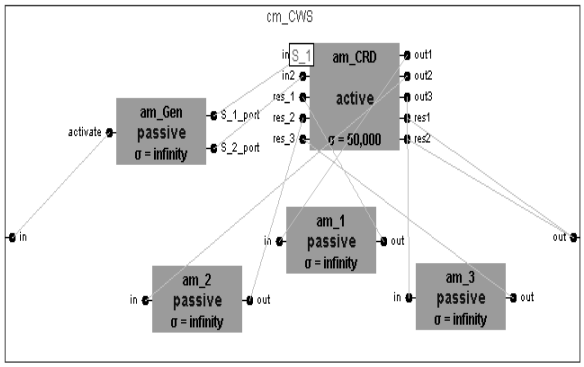

Рисунок 4 – Структурная схема системы с CWS

Фрагмент временных диаграмм процесса моделирования приведен на рис. 5.

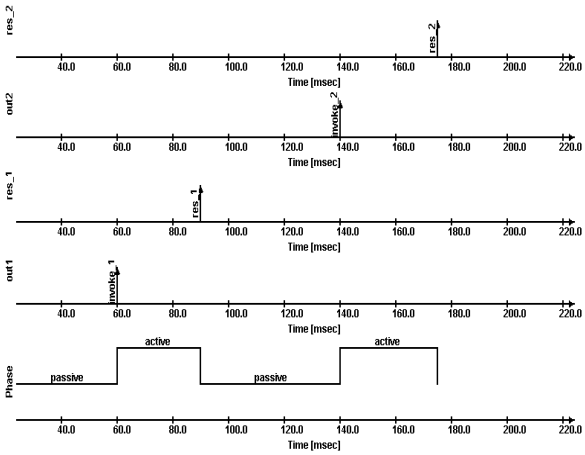

Рисунок 5 – Временные интервалы функционирования am_1, am_2

Для наглядности, приведем схему предложенной концептуальной модели с привязкой к рассмотренному примеру (рис. 6).

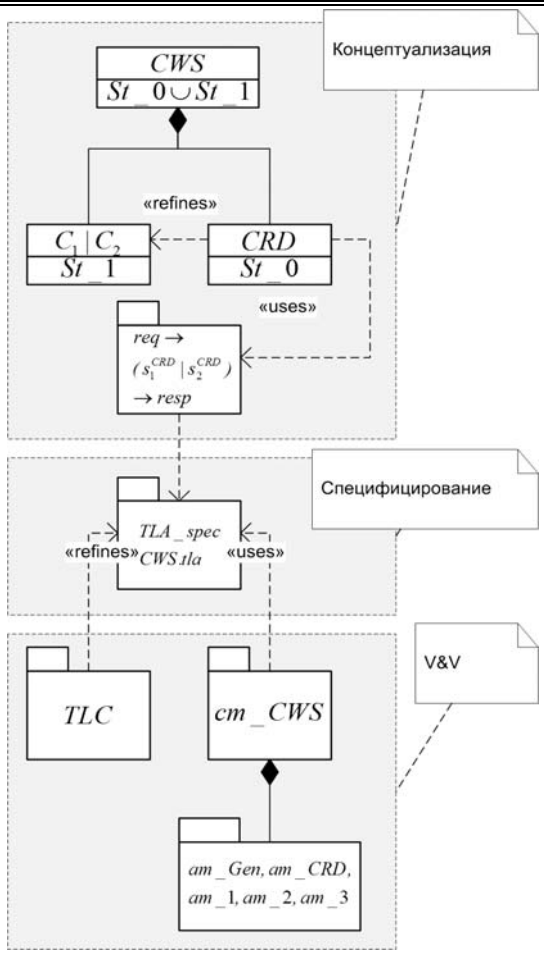

Рисунок 6 – Концептуальная модель процесса автоматизированного синтеза CWS

Выводы

Таким образом, в процессе автоматизированного синтеза CWS выделены для рассмотрения три последовательных этапа: концептуализация, специфицирование, V&V.

На этапе концептуализации проведена стратификация модели композитного веб-сервиса, что, вместе с концептами теории взаимодействующих последовательных процессов Ч. Хоара, позволило сформировать предпосылки для упрощения реализации последующего этапа специфицирования.

На этапе специфицирования предложен набор правил трансляции концептов, введенных на этапе концептуализации, в формальную TLA-спецификацию.

На этапе V&V выполнена проверка корректности TLA-спецификации, создана дискретно-событийная имитационная модель CWS в среде DEVS Suite.

В качестве примера сценария предметной области был рассмотрен сценарий генерации запросов к СУБД.

Список использованной литературы

1. Шкарупило В.В. Комплексный подход к автоматизации композиции веб-сервисов / В.В. Шкарупило, Р.К. Кудерметов // Науковий вісник Чернівецького національного університету ім. Ю. Федьковича. Серія: Комп'ютерні системи та компоненти. – 2011. – Т. 2. – Вип. 1. – С. 113 – 119.

2. Lamport L. Specifying Systems / L. Lamport. – Boston. : Addison-Wesley, 2002. – P. 364.

3. Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем / Ю.Г. Карпов. – СПб.: БХВ-Петербург, 2010. – 560 с.

4. Дерецкий В.А. Подход к композиции веб-сервисов на основе спецификации функциональной семантики / В.А. Дерецкий // Проблеми програмування. – 2009. – № 2. – С. 30 – 39.

5. Месарович М. Теория иерархических многоуровневых систем / М. Месарович, Д. Мако, И. Такахара. – М. : Мир, 1973. – 344 с.

6. Самарский А.А. Математическое моделирование: Идеи. Методы. Примеры / А.А. Самарский, А.П. Михайлов. – 2-е изд. – М.: Физматлит, 2001. – 320 с.

7. Ларман К. Применение UML и шаблонов проектирования / К. Ларман; пер. с англ., под ред. А. Ю. Шелестова. – 2-е изд. – М.: Вильямс, 2004. – 624 с.

8. Web Services Business Process Execution Language Version 2.0 [Электронный ресурс] / OASIS Standard: ad/2007-04-11. – Режим доступа: \www/ URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. – Загл. с экрана.

9. Хоар Ч. Взаимодействующие последовательные процессы / Ч. Хоар; пер. с англ. А. А. Бульонкова. – М. : Мир, 1989. – 264 с.

10. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [Электронный ресурс] / W3C Recommendation: ad/2007-04-27. – Режим доступа: \www/ URL: http://www.w3.org/TR/soap12-part1/. – Загл. с экрана.

11. Топорков В.В. Модели распределенных вычислений / В.В. Топорков. – М.: ФИЗМАТЛИТ, 2004. – 320 с.

12. DEVS-Java Reference Guide [Электронный ресурс]. – Режим доступа: \www/ URL: http://vlab.unm.edu/documents/devsjava-user-ref.pdf. – 12.7.2011. – Загл. с экрана.

Надійшла до редколегії 27.12.2011

В.В. ШКАРУПИЛО, Р.К. КУДЕРМЕТОВ, Т.О. ПАРОМОВА.
Запорізький національний технічний університет

КОНЦЕПТУАЛЬНА МОДЕЛЬ ПРОЦЕСУ АВТОМАТИЗОВАНОГО СИНТЕЗУ КОМПОЗИТНИХ ВЕБ-СЕРВІСІВ

Запропоновано концептуальну модель процесу автоматизованого синтезу композитних веб-сервісів. Модель процесу представлена як послідовність етапів концептуалізації, специфікації, верифікації та валідації. Модель композитного веб-сервіса представлена як ієрархічна система з функціональними та нефункціональними характеристиками.

Ключові слова: композитний веб-сервіс, формальна специфікація, верифікація, валідація, Temporal Logic of Actions, Model Checking, DEVS.

V. SHKARUPYLO, R. KUDERMETOV, T. PAROMOVA
Zaporizhzhya National Technical University

CONCEPTUAL MODEL OF AUTOMATED COMPOSITE WEB SERVICES SYNTHESIS PROCESS

The conceptual model of automated Composite Web Services synthesis process has been proposed. Process model has been represented as the sequence of steps: Conceptualizing, Specification, Verification & Validation. Composite Web Service model has been represented as hierarchical system with functional and non-functional properties.

Keywords: composite Web Service, Formal Specification, Verification, Validation, Temporal Logic of Actions, Model Checking, DEVS.