







IRSTI 81.93.29

DOI: <https://doi.org/10.26577/JMMCS2024-122-02-b10>

V.V. Shkarupilo¹ , V.A. Lakhno¹ , N.B. Konyrbaev^{2*} ,
Zh.D. Baishemirov^{3,4} , A.B. Adranova² , A.G. Derbessal² 

¹National University of Life and Natural Environmental Sciences of Ukraine, Ukraine, Kyiv
²Korkyt Ata Kyzylorda University, Kazakhstan, Kyzylorda
³Abai Kazakh National Pedagogical University, Kazakhstan, Almaty
⁴Kazakh-British Technical University, Kazakhstan, Almaty
*e-mail: n.konyrbaev@mail.ru

HIERARCHICAL MODEL FOR BUILDING COMPOSITE WEB SERVICES

The current evolution of disseminated computer program frameworks is characterized by a growing adherence to the principles of service-oriented engineering (SOA). Simultaneously, these frameworks are becoming more intricate, with an increasing number of components and more complex data connections between them. This situation underscores the importance of employing mechanisms to unify artifacts in the process of developing composite web services, which govern, among other aspects, the architectural plane of the frameworks under construction. A model for constructing composite web services is suggested as a suitable tool, implemented in accordance with a hierarchical approach, intended for use in designing distributed systems. Model is constructed over an assumption that coordination of the components of a composite web service is carried out in a centralized manner ? in accordance with the orchestration model. To implement formalization and obtain, based on analytical representations, the corresponding software implementations, it has been decided to use the DEVS mathematical apparatus. The aspect of software implementation is considered pivotal in determining the feasibility of automating the acquisition of composite web services that operate within the orchestration model. Obtained research results has been interpreted as a confirmation of the effectiveness of this approach on the basis of the scenario of querying the database. Resulting artifacts have been represented with UML notation. The relationship between analytical representations and corresponding software implementations has also been demonstrated. Usage of the DEVS Suite tools has made it possible to visualize the process of simulation - to obtain estimated values of the indexes of the resulting solutions.

Key words: distributed system, composite web service, DEVS, UML.

В.В. Шкарупило¹, В.А. Лахно¹, Н.Б. Коңырбаев^{2*}, Ж.Д. Байшемиров^{3,4},
А.Б. Адранова², А.Г. Дербесал²

¹Украина биоресурстар және табиғатты пайдалану ұлттық университеті, Украина, Киев қ.

²Қорқыт ата атындағы Қызылорда университеті, Қазақстан, Қызылорда қ.

³Абай атындағы Қазақ ұлттық педагогикалық университеті, Қазақстан, Алматы қ.

⁴Қазақстан-Британ техникалық университеті, Қазақстан, Алматы қ.

*e-mail: n.konyrbaev@mail.ru

Композиттік веб-қызметтерді құрудың иерархиялық моделі

Бөлінген бағдарламалық жүйелерді дамытудың қазіргі деңгейін сервистік-бағдарланған архитектура (СБА) ережелерін ұстану барған сайын кең таралған тәжірибеге айналатын деңгей ретінде сипаттауға болады. Сонымен қатар, мұндай жүйелердің күрделілік деңгейі қатысатын құрамдас бөліктердің саны бойынша, осы құрамдас бөліктер арасында орнатылған ақпараттық байланыстардың күрделілігі бойынша да арта түсуде. Бұл жағдай, өз кезегінде, құрылатын жүйелердің архитектуралық құрамдас бөлігін реттейтін құрама веб-қызметтерді әзірлеу процесінде артефактілерді біріктіру тетіктерін пайдаланудың маңыздылығын анықтайды. Тиісті құрал ретінде иерархиялық тәсілге сәйкес жүзеге асырылатын композиттік веб-қызметтерді құру моделі ұсынылады.

Модель бөлінген жүйені жобалау кезеңінде пайдалануға арналған. Модель композиттік веб-қызметтің құрамдас бөліктерін үйлестіру орталықтандырылған - оркестрлік модельге сәйкес жүзеге асырылады деген болжамға негізделген. Ресімдеуді жүзеге асыру және аналитикалық ұсыну негізінде сәйкес бағдарламалық қамтамасыз етуді енгізу үшін DEVS математикалық аппаратын пайдалану туралы шешім қабылданды. Бағдарламалық қамтамасыз етуді енгізу, өз кезегінде, оркестрлік модель бойынша жұмыс істейтін композиттік веб-қызметтерді алу процесін автоматтандыру мүмкіндігін анықтайтын фактор ретінде қарастырылады. Зерттеу нәтижелері дерекқорға сұраныстарды орындау сценарийінің мысалын пайдалана отырып, бұл тәсілдің тиімділігін растады. Алынған артефактілер UML экспрессивті құралдары арқылы ұсынылды. Сондай-ақ аналитикалық көріністермен сәйкес бағдарламалық қамтамасыз етуді енгізу арасындағы байланыс көрсетілді. DEVS Suite құралдарының мүмкіндіктерін пайдалану, басқалармен қатар, модельдеу процесін визуализациялауға – жасалатын шешімдердің көрсеткіштерінің болжалды мәндерін алуға мүмкіндік берді.

Түйін сөздер: үлестірілген жүйе, композиттік веб-сервис, DEVS, UML

В.В. Шкарупило¹, В.А. Лахно¹, Н.Б. Конырбаев^{2*}, Ж.Д. Байшемиров^{3,4},
А.Б. Адранова², А.Г. Дербесал²

¹Национальный университет биоресурсов и природопользования Украины, Украина, г. Киев

²Кызылординский университет имени Коркыт Ата, Казахстан, г. Кызылорда

³Казахский национальный педагогический университет имени Абая, Казахстан, г. Алматы

⁴Казахстанско-Британский технический университет, Казахстан, г. Алматы

*e-mail: n.konyrbaev@mail.ru

Иерархическая модель построения составных веб-сервисов

Текущий уровень развития распределенных программных систем можно охарактеризовать как такой, при котором следование положениям сервис-ориентированной архитектуры (СОА) становится все более повседневной практикой. Вместе с тем уровень сложности таких систем продолжает возрастать – как с позиции количества задействованных компонентов, так и с позиции комплексности информационных связей, устанавливаемых между данными компонентами. Такое положение вещей, в свою очередь, обуславливает важность использования в процессе разработки составных веб-сервисов механизмов унификации артефактов, регламентирующих, в том числе, архитектурную составляющую создаваемых систем. В качестве соответствующего инструмента предлагается модель построения составных веб-сервисов, реализованная согласно иерархическому подходу. Модель предназначена к использованию на этапе проектирования распределенной системы. Модель построена на допущении, что координирование компонентов составного веб-сервиса осуществляется централизованно – согласно модели оркестровки. Для проведения формализации и получения на основе аналитических представлений соответствующих программных реализаций принято решение задействовать математический аппарат DEVS. Программная реализация, в свою очередь, адресована в качестве фактора, обуславливающего возможность автоматизации процесса получения составных веб-сервисов, функционирующих согласно модели оркестровки. Результаты проведенных исследований подтвердили действенность такого подхода на примере сценария выполнения запросов к базе данных. Получаемые при этом артефакты были представлены с использованием выразительных средств UML. Также была продемонстрирована связь между аналитическими представлениями и соответствующими программными реализациями. Использование возможностей инструментария DEVS Suite позволило, в том числе, визуализировать процесс имитационного моделирования – для получения оценочных значений показателей создаваемых решений.

Ключевые слова: распределенная система, композитный веб-сервис, DEVS, UML.

1 Introduction

Today, the concept of reuse, which underlies service-oriented architecture (SOA), is one of the defining concepts considered when creating distributed web applications. The reason for this could be to save money on the development process. Web services are parts of SOA-based systems. One big advantage of web services is that they are not tightly connected. This makes the system better at working together and communicating with its different parts. The group of online services in a system is often called a composite web service, and the individual parts of the system are called atomic web services.

Given that composite web services (CWS) can embody systems of varying complexity, it is prudent to conceptualize a CWS as a stratified system. This approach facilitates the process of specification, verification, and validation, ensuring that the synthesized CWS operates correctly through a series of automated steps. Research [1] highlights a deficiency in addressing verification and validation (V&V) issues during development. Synthesizing a CWS involves employing diverse methods to create a CWS with the necessary functional (F) and non-functional (NF) characteristics.

Having a detailed plan in writing, like a set of rules or instructions. A detailed plan for how atomic web services interact with each other is needed to automatically create CWS. This condition is needed because machines need to clearly understand the specification when they are doing automated tasks. Proposed to use TLA (Temporal Logic of Actions) formalism by L. Lamport [2] is a way to explain how something works. The selection of this formalism is substantiated by several distinguishing characteristics: firstly, the utilization of the Model Checking verification method (TLC, TLA Checker) is seamlessly integrated into the corresponding TLA Toolbox software. Secondly, the incorporation of the "behavior" concept enables the description of acceptable scenarios for the operation of the system under examination. Through the Model Checking approach, it becomes feasible to automatically verify permissible system states, acceptable parameter values for specifications, and to detect potential "deadlocks". One notable advantage of adopting the Model Checking approach for verification lies in its potential for complete automation.

The verification process is tasked with determining whether the formal specification of the Composite Web Service (CWS) has been accurately constructed. This entails confirming the correctness of the specification. A supporting statement from [3] reinforces this notion: "Finding methods to ensure that the developed hardware and software meet their specifications is a core challenge in computer science." Conversely, the validation procedure for CWS seeks to answer the question, "Are we developing the appropriate system with CWS?" This process focuses on validating the suitability of the CWS, ensuring its compliance with predefined standards for both functional (F) and non-functional (NF) attributes. It is recommended to evaluate the practicality of the synthesized CWS for a specific instance, with predetermined criteria for its F and NF characteristics.

2 Formulation of the problem

It is imperative to identify the functional (F) and non-functional (NF) properties of a newly conceptualized Composite Web Service (CWS) during the planning phase of its creation [4]. The automated synthesis of CWS is proposed to proceed systematically through

the stages of conceptualization, specification, and verification and validation (V&V). This sequence, referred to as "conceptualization/specification/V&V," entails developing a formal model specification, designing a CWS model, validating the model, and assessing its adequacy. To streamline the specification stage, it is recommended to structure the model of the CWS during the conceptualization phase. This involves creating a formal, machine-interpretable definition of acceptable scenarios for the operation of the CWS. The verification and validation (V&V) stage ensures alignment between the developed model and specification, while also confirming the adequacy of the model by verifying that the planned CWS's F and NF characteristics meet the required specifications. As a result, this work undertakes the investigation of the proposed sequence of steps in the automated synthesis process of CWS, along with the technologies and tools utilized in its implementation.

3 Conceptualization of CWS

Consider the Composite Web Service (CWS) as a hierarchical structure, which simplifies the subsequent specification process. Complex hierarchical systems can be organized as follows [5, 6]: initially, a conceptual model of the system is crafted, with each layer representing a different level of the subsystem hierarchy. The hierarchical modeling approach is demonstrated by creating models of system components that are then integrated into the overall system model.

The system under investigation is denoted as "coordinator/computers," representing a specific instance of CWS. The "Controller" design pattern [7] can aptly describe the behavior of such a system. According to this pattern, "a controller should typically delegate tasks to other objects and manage their activities rather than executing tasks themselves." Examples of existing system components aligning with the "Controller" pattern include elements of the Grid infrastructure (CE/WNs, Computing Element & Working Nodes), where the CE component assumes the role of the controller (coordinator). These proposed abstractions resonate with the composition model outlined in the WS-BPEL standard [8], known as "orchestration a model for centrally coordinating web services within a composition (CWS). The function of the synthesis process coordinator is performed by the BPEL Engine component, implemented as part of the corresponding tools (Oracle BPEL Process Manager, ActiveBPEL, Eclipse BPEL Designer, etc.). Let's denote the BPEL Engine component as CRD (Coordinator, Controller).

To describe the specification formalism, we use a set-theoretic approach. Let us denote the set of atomic web services as $AWS = \{aws_i \mid i = \overline{1, m}\}$, $m \in N$, where $aws_i \in AWS$ – atomic web services available for use. Sets of some necessary for the implementation of the F-characteristics of CWS will be represented as subsets of the set AWS : $\{C_j \mid j = \overline{1, n}\}$, $n \in N$, where $C_j \subseteq AWS$ – subset of atomic web services required for implementation j -th F-characteristics of CWS. $C_j = \{aws_k \mid k = \overline{1, p}\}$, $p \in N$, and $p \leq m$.

Let everyone aws_i characterized by a pair (af_i, anf_i) , where af_i and anf_i – Φ - and $H\Phi$ -characteristics aws_i , respectively. By af_i will understand some F-transformation, performed on a set of input data vec_i : $af_i = f_i(vec_i)$. Conceptually under aws_i we can understand some abstract entity that implements a function $f_i(vec_i)$.

Let the NF characteristic anf_i determined by three (r_i, t_i, c_i) , where r_i (response) – response time aws_i ; t_i (throughput) – the capacity of the network channel formed by the

sender node of the request and the recipient node (on which some aws_i); $c_i(\text{cost})$ – function execution cost value $f_i(\text{vec}_i)$. In this case, we will assume that the value of the element r_i equals the sum of the times spent on transmitting the request (from the sending node to the receiving node) and on implementing the F-characteristic to some aws_i , deployed on the recipient node.

Let's assume that the client request specifies requirements for F- (F_req) and NF characteristics (NF_req) CWS. NF_req , wherein, are determined by three (r_req, t_req, c_req), where the elements of the triple represent the response time, link capacity, and cost requirements of CWS, respectively.

A positive answer to the question "Does some NF characteristic of CWS satisfy the requirements of the client request?" it is proposed to give if the corresponding inequalities are true:

$$r_req \geq \sum_{i=1}^n r_i. \quad (1)$$

$$t_req \leq \min(t_i).$$

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

If we view the interactions among certain $aws_k, aws_{k+1} \in C_j$ entities through the lens of CWS as sequential exchanges between computing processes dispersed geographically, facilitated by asynchronous exchange of structured messages, it seems plausible to consider a formalism grounded in the principle of function superposition as an apt means to depict the functional characteristic of CWS. This approach can be justified by Charles Hoare's theory of interacting sequential processes [9] and specific aspects of message exchange mechanisms among distributed computer system components outlined in the SOAP protocol [10]. This technique serves as a natural method for attaining the requisite functional characteristic of CWS aggregation, as illustrated in (Fig. 1).

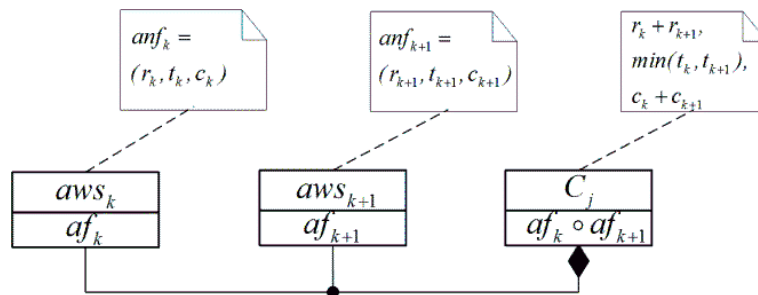


Figure 1: j-th F-characteristic of CWS aggregation scheme

Let's separate the "coordinator/computers" system into two strata: St_0 and St_1 (Table 3).

Because the function is to coordinate atomic web services as part of CWS; by the coordination procedure we mean the execution of calls to some $aws_k \in C_j$ in a given sequence. CRD and aws_k . In this case, we will call them elements of the corresponding strata.

Losses	Purpose of the component	Formal notation
St_0	coordinator	CRD
St_1	calculators	$C_j = \{aws_k\}$

Utilizing the introduced formalism, we offer a structural UML diagram depicting the stratification of the CWS "coordinator/computers"(fig. 2). Here, the term "refines," denoted by the operation (operator), signifies the execution of the coordination procedure.

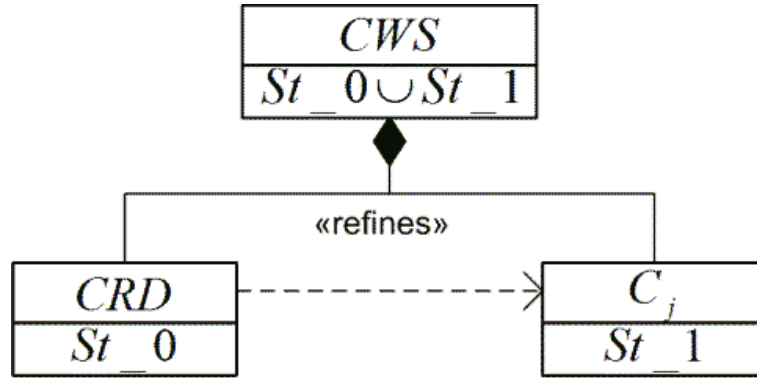


Figure 2: CWS stratification scheme

Let the procedure for coordinating elements C_j is implemented within a certain subroutine. In the theory of interacting sequential processes proposed by Charles Hoare, it is recommended to regard the entire system under examination as a process. Here, the behavior of this process is delineated by the behaviors of its constituent subprocesses. These subprocesses' behavior, in turn, is contingent upon the frequency and order of events. Consequently, alterations in the states of the system in question transpire upon the incidence of events of three distinct types: "boundary", "challenge", "result". Let us represent these types of events in the form of corresponding sets:

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

where REQ – many boundary events, and req – coordinator receiving event CRD request with requirements for F and NF characteristics of CWS (we will consider req as initial event); $resp$ – final event – sending the result of the CWS work;

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

where $INVOKE$ – many call events from the coordinator CRD elements $aws_k \in C_j$;

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

where RES – set of receiving events by coordinator CRD results of element's operation $aws_k \in C_j$.

Some event $invoke_k \in INVOKE$ we will consider as a stimulus the following type of display:

$$f_k : vec_k \mapsto res_k.$$

We aim to delineate subprocesses that unveil the functional characteristics of CWS via suitable scenarios. Drawing from Charles Hoare's formalism, we propose documenting events using a protocol—a predefined sequence of notations linked to events. We advocate substituting the term "protocol" with the notion of "scenario." This adjustment aligns better with the intricacies of the system under consideration, as the orchestration model delineates a centralized approach to orchestrating the coordination process. Let us denote by a set of scenarios describing the dynamics CWS's (F-characteristics):

$$S = \{s_j^{CRD}\}.$$

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

Those every s_j^{CRD} describes a method (scenario) for implementing some CWS F-characteristic based on coordination of elements C_j . The initial entry of the script corresponds to some event of the "call" type, and the final entry corresponds to an event of the "result" type. It's obvious that $|S|$ (cardinality of the set S) equal to the number of F-characteristics of CWS.

The item under investigation (system) first takes part in an event, and then it behaves exactly like a process (subprocess), according to C. Hoare's theory of interacting sequential processes. Formally, it is proposed to write it like this: $x \rightarrow P$, where x , P – some event and process (as a sequence of events), respectively; ' ' – follow operator; reads like " P for x ". Let's modify this recording method by including a selected type of boundary events into consideration. To do this, let us denote by s_{j+1}^{CRD} some alternative scenario specifying an alternative CWS's F-characteristic. The alternative will be designated as ". The following characteristics apply to acceptable CWS speakers:

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

One could think about this method of defining the CWS dynamics (3) as an expansion of (2).

It is also important to note that [11] suggests an alternative method of documenting events (instead of laying out a timeline). The concept of "process history" (h) is utilized in place of "protocol." Synopsis h is carried out in the manner: $e \xrightarrow{h} e'$; $e, e' \in E$, where E – several incidents, '→' indicates the changes between occurrences, h – an arrangement of transitional events from E .

From the perspective of streamlining the process for interpreting a script into a formal TLA specification, we believe that setting the sequence of events using scripts is a more acceptable method.

4 A system with CWS example

Now, let's delve into a specific scenario. Let's suppose we're examining a system equipped with Composite Web Services (CWS). This system can be perceived as a modified version of

the "coordinator/computers" system. We introduce an additional actor named "Client" into this system, denoting a source of boundary events: request generation is represented by event *req*; while receiving the CWS output is depicted by event *resp*.

As an example domain scenario, let's consider the process of generating queries to a Database Management System (DBMS). The significance of this scenario is underscored by the prevalence of corresponding web-based software systems (eBay, newegg, etc.). Because Oracle or MySQL solutions are usually used as a DBMS; let the set of query generation functions be presented as the following set: $\{select, delete, update\}$, where the elements denote the functions for generating queries for selecting, deleting and modifying table records, respectively. Let the specified functions be implemented by atomic web services aws_1, aws_2, aws_3 , respectively.

To modify (delete) the required record of a table, you must first generate a query to make sure that exactly the required record is selected; then, depending on the end goal being pursued, execute either the request *delete*, or request *update*. As a consequence, we see that a possible way to automate this procedure is the synthesis of CWS, the functioning of which can be carried out according to two scenarios:

$$req \rightarrow (s_1^{CRD} | s_2^{CRD}) \rightarrow resp.$$

Scenarios s_1^{CRD} and s_2^{CRD} reveal the F-characteristics of CWS:

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

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

$$s_1^{CRD} = \langle invoke_1, res_1, invoke_2, res_2 \rangle;$$

$$s_2^{CRD} = \langle invoke_1, res_1, invoke_3, res_3 \rangle.$$

Let us present the described scenarios in the form of a UML interaction sequence diagram (fig. 3).

5 Specification, V & V

Interpreting scenarios s_1^{CRD} and s_2^{CRD} into a formal TLA specification involves envisioning scenario records as sequences of Composite Web Service (CWS) states. To achieve this, we establish rules for specifying event occurrences: initialization of variables corresponding to events involves assigning elements of the set; '0' denotes the event did not occur, while any other value represents occurrence; a modifier (") indicates the value of a variable specifying event occurrence at a subsequent point in time.

To define CWS states and their sequence: utilize the conjunction operator (\wedge) to connect the current state to the previous one and set variable values based on events within the CWS state; employ the disjunction operator (\vee) to indicate alternation in scenarios; the 'UNCHANGED' modifier denotes that a variable's value in the current state remains unchanged from the previous state.

The TLA specification for the given cases (s_1^{CRD} and s_2^{CRD}), is then created using the interpretation rules outlined above. Listing for formal CWS TLA demonstrated (Fig. 4).

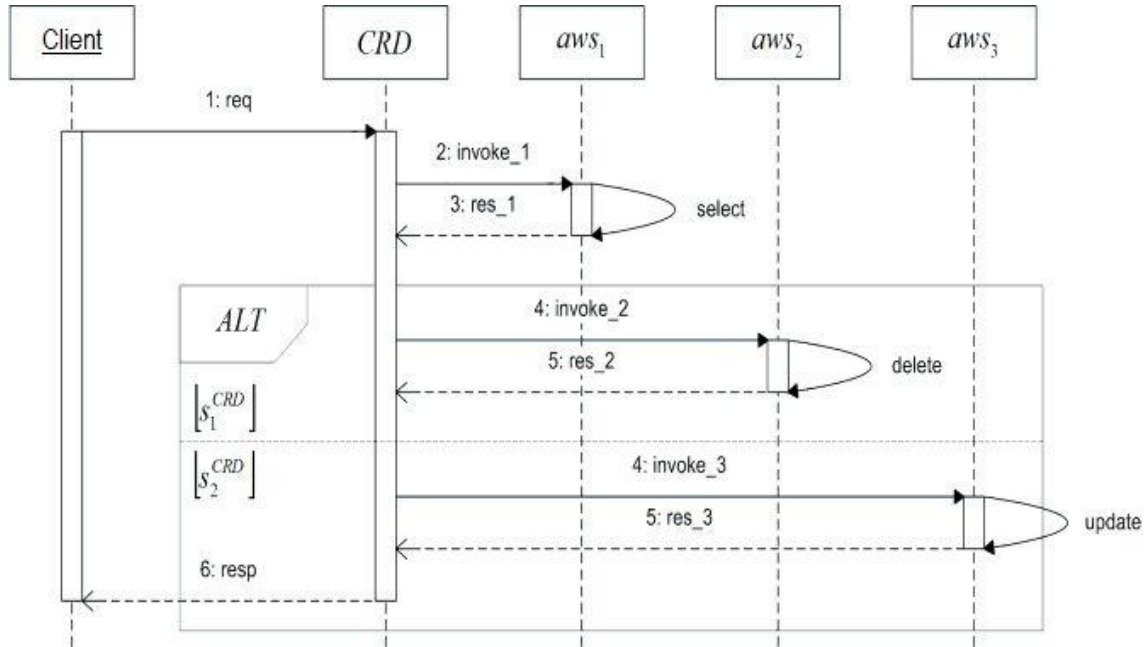


Figure 3: Scenarios for the operation of a system with CWS

Listing 1 defines valid CWS states according to the following conventions: Init, OnReq, OnInvoke_1,..., OnRes3, OnResp. The correctness of the specification was checked using the Model Checking method (TLC, TLA Checker), integrated into the TLA Toolbox development environment.

Analyzing the proposed specification method, one can note some cumbersomeness (syntactic redundancy) of the resulting CWS TLA specification. As an opposite (positive) point, we can point out the clarity and structure of the TLA specification obtained through the use of the proposed set of translation rules. The indicated advantages and disadvantages can also characterize wsdl (Web Services Description Language) descriptions of atomic web services.

The next step is to implement the validation procedure. In our case, the validation procedure consists of conducting discrete-event simulation modeling in the DEVS Suite environment. A distinctive feature of the DEVS formalism is the concept of “atomic model” [12]. This concept is preferable in that it allows one to naturally represent the hierarchical connections (relationships) of component models within a system model with a CWS.

Let us denote by am_1, \dots, am_3 atomic web service models aws_1, \dots, aws_3 , respectively. Coordinator Model CRD let's denote it as am_CRD . We will represent the “Client” component of a system with CWS in the form of an atomic model of a scenario generator (s_1^{CRD} , or s_2^{CRD}), which are then sent to the model's input ports am_CRD . Let us denote the model of the “Client” component as am_Gen .

Let's include models of atomic components as part of the system model with CWS (cm_CWS). We will record the moments when messages appear on the input and output ports (in and out) models cm_CWS as moments of the onset of boundary events req and $resp$, respectively.

```

\* operate with natural numbers
EXTENDS Naturals
\* variables denoting events
VARIABLES req, resp,
           invoke_1, invoke_2, invoke_3,
           res_1, res_2, res_3
\* setting acceptable values
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}
\* CWS state specification:
\* 1 - none of the events happened
\*
Init == /\ req=0 /\ invoke_1=0 /\ res_1=0 /\ resp=0 /\ invoke_2=0 /\ res_2=0 /\ invoke_3=0 /\ res_3=0
\* 2 - receipt of a request
\* from the client
OnReq == /\ req' = 1 - req
/\ UNCHANGED<<resp>>
/\ UNCHANGED<<invoke_1, invoke_2, invoke_3>>
/\ UNCHANGED<<res_1, res_2, res_3>>
\* 3 - call by coordinator 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 - receipt by coordinator
\* call result 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 - call by coordinator 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 - receipt by coordinator
\* call result 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 - call by coordinator 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 - receipt by coordinator
\* call result 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 - sending to client
\* work result CWS,
\* task of alternativeness
\* scenarios
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>>

```

Figure 4: Fragment of specification

Now let's model the system we are studying. As NF features of the system model's constituent parts with CWS, we select the response time, $ms:anf_1.r_1 = 30$, $anf_2.r_2 = 40$, $anf_3.r_3 = 35$. Model response time am_Gen set equal to 10 ms, and the time spent on implementing the coordination procedure by the model $am_CRD - 50$ ms.

Let the requirements for SF characteristics $CWS NF_req.r_req = 200$ ms. Our task is to check through simulation whether the CWS model satisfies the given NF requirements. Satisfaction of the requirements for the CWS F-characteristics is confirmed by the correct functioning of the model.

Consider the case when the input port of the coordinator model am_CRD script arrived s_1^{CRD} (fig.5).

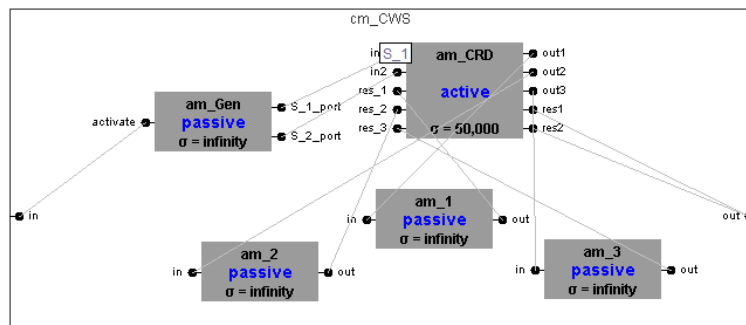


Figure 5: Block diagram of a system with CWS

A fragment of time diagrams of the modeling process is shown in fig. 6.

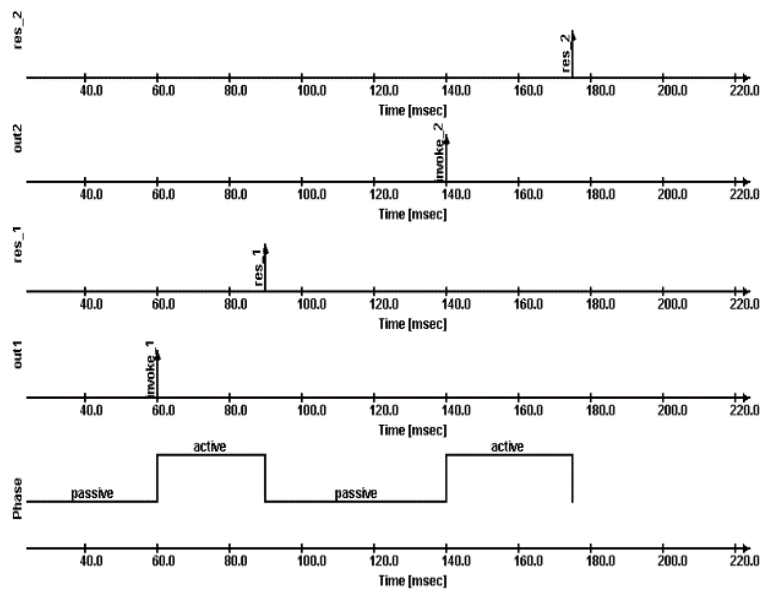


Figure 6: Operation time intervals am_1 , am_2

The results of the simulation show that the total value of the component's NF characteristics cm_CWS sums to 225 milliseconds, which is insufficient to meet inequality

(1). For clarity purposes, we have incorporated an illustration of the proposed conceptual model in Figure 7, specifically pertaining to the scenario under examination.

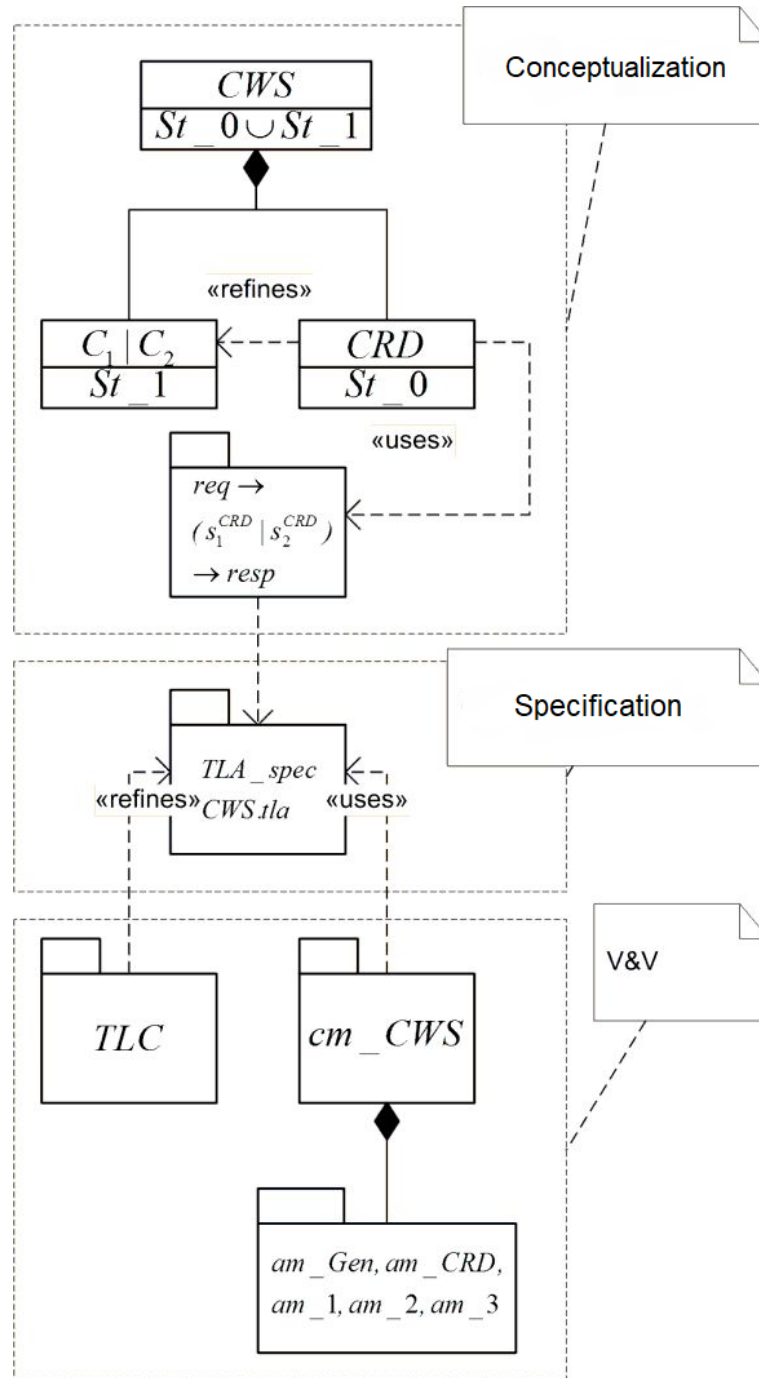


Figure 7: CWS automated synthesis process conceptual model

6 Conclusion

Henceforth, the automated synthesis of CWS entails three consecutive stages: conceptualization, specification, and V&V.

In the conceptualization phase, the composite web service model was stratified, drawing upon Charles Hoare's theory of interacting sequential processes to establish conditions facilitating the subsequent specification stage.

Guidelines for formalizing ideas from the conceptualization stage into a TLA specification are proposed during the specification step.

During the V&V phase, a discrete-event simulation model of CWS was developed within the DEVS Suite environment, and the accuracy of the TLA specification was validated. A domain scenario instance was examined, involving the formulation of queries for a database management system.

References

- [1] Shkarupilo V.V., An integrated approach to automating the composition of web services, Scientific Bulletin of the Chernivets National University, Series: Computer systems and components, 2(1) (2011), 113 – 119.
- [2] Lamport L., Specifying Systems, Boston:Addison-Wesley (2002). <https://lamport.azurewebsites.net/tla/book-02-08-08.pdf>
- [3] Pakonen A., Model-checking I&C logics — insights from over a decade of projects in Finland, 12th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies (2021), 792–801. <https://dx.doi.org/10.13182/T124-34322>
- [4] Deretsky V.A., An approach to the composition of web services based on the specification of functional semantics, Problems of programming, 2, (2009), 30–39. <https://core.ac.uk/download/pdf/38330531.pdf>
- [5] Mesarovic M.D., Macko D., Takahara Y., Theory of hierarchical multi-level systems, Elsevier Science, (1970).https://esploro.lib.uga.edu/permalink/01GALI_UGA/182omg4/alma99201813902959
- [6] Samarsky A.A., Mathematical Modeling: Ideas. Methods. Examples, Moscow:Fizmatlit, (2001).
- [7] Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development 3rd. Addison Wesley Professional(2004). <https://bsituos.weebly.com/uploads/2/5/2/5/25253721/applying-uml-and-patterns-3rd.pdf>
- [8] Web Services Business Process Execution Language Version 2.0, OASIS Standard: ad/2007-04-11 (2007). <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
- [9] Hoare C.A.R., Communicating Sequential Processes, Prentice Hall International(2022). <http://www.usingcsp.com/cspbook.pdf>
- [10] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [Electronic resource], W3C Recommendation: ad/2007-04-27, (2007). <http://www.w3.org/TR/soap12-part1/>
- [11] Toporkov V.V., Modeli raspredelennykh vychisleniy, Moscow:FIZMATLIT,(2004). https://rusneb.ru/catalog/000199_000009_002557693/
- [12] Tendeloo Y.V., Vangheluwe H., An evaluation of DEVS simulation tools. Simulation, 93(2),(2017), 103–121. <https://dx.doi.org/10.1177/0037549716678330>

Information about authors:

Shkarupylo Vadim – Associate Professor of the Department of "Computer Systems, Networks and Cybersecurity National University of Life and Natural Environmental Sciences of Ukraine (Kyiv, Ukraine, email: shkarupylo.vadym@nubip.edu.ua);

Lakhno Valerii – Doctor of Science, Professor of the Department of "Computer Systems, Networks and Cybersecurity National University of Life and Natural Environmental Sciences of Ukraine (Kyiv, Ukraine, email: lva964@nubip.edu.ua);

Konyrbaev Nurbek (corresponding author) – PhD, Associate professor, head of the Department of Computer Sciences, Kyzylorda University named after Korkyt Ata (Kyzylorda, Kazakhstan, email: n.konyrbaev@mail.ru);

Bayshemirov Zharasbek – PhD, 1) Abai Kazakh National Pedagogical University, Acting Professor of Mathematics and Mathematical Modeling, Postdoctoral Student (Almaty, Kazakhstan); 2) Associate Professor of Kazakhstan-British Technical University (Almaty, Kazakhstan, email: zbai.kz@gmail.com);

Adranova Asselkhan – PhD, senior lecturer of Computer Sciences Department of Kyzylorda University named after Korkyt Ata (Kyzylorda, Kazakhstan, email: aselhan.adranova@mail.ru);

Derbesal Assem – master of technical sciences, teacher of Computer Science Department of Kyzylorda University named after Korkyt Ata (Kyzylorda, Kazakhstan, email: asem.galymzhankyzy@gmail.com).

Авторлар туралы мәлімет:

Шкарупило Вадим Викторович – ф.-м.ғ.к., доцент, Украина биоресурстар және табиғатты пайдалану ұлттық университеті "Компьютерлік жүйелер, желілер және киберқауіпсіздік" кафедрасының доценті (Киев, Украина, электрондық пошта: shkarupilo.vadym@nubip.edu.ua);

Лакно Валерий Анатольевич – техника ғылымдарының докторы, Украинаның биоресурстар және табиғатты пайдалану ұлттық университеті "Компьютерлік жүйелер, желілер және киберқауіпсіздік" кафедрасының профессоры (Киев, Украина, электрондық пошта: lva964@nubip.edu.ua);

Қоңырбаев Нұрбек Беркінбайұлы – PhD, қауымдастырылған профессоры, Қорқыт Ата атындағы Қызылорда университетінің "Компьютерлік ғылымдар" БББ жетекшісі, қауымдастырылған профессоры (Қызылорда, Қазақстан, электрондық пошта: n.konyrbaev@mail.ru);

Байшеміров Жарасбек Дүйсембекұлы – PhD, 1) Абай атындағы Қазақ ұлттық педагогикалық университеті, математика және математикалық модельдеу кафедрасының профессор м.а., постдокторант (Алматы, Қазақстан); 2) Қазақстан-Британ техникалық университетінің қауым. профессоры (Алматы, Қазақстан, электрондық пошта: zbai.kz@gmail.com);

Адранова Әселхан Бағдатқызы – PhD, Қорқыт Ата атындағы Қызылорда университетінің "Компьютерлік ғылымдар" БББ аға оқытушысы (Қызылорда, Қазақстан, электрондық пошта: assel.adranova@gmail.com);

Дербесал Әсем Ғалымжанқызы – техника ғылымдарының магистрі, Қорқыт Ата атындағы Қызылорда университетінің "Компьютерлік ғылымдар" БББ оқытушысы (Қызылорда, Қазақстан, электрондық пошта: asem.galymzhankyzy@gmail.com).

Received: March 26, 2024

Accepted: June 20, 2024