УДК 004.89
DOI: 10.15587/2313-8416.2015.38838

FORMALIZING OF WEB-SERVICES SPECIFICATIONS USING TEMPORAL LOGICS

© M. K. H. Al-Dulaimi. A. M. K. Al-Dulaimi

Issues related to improving the design of multiservice SOA networks are discussed. Issue of creation of complex services, their orchestration and choreography are considered. Problem arising when creating an integrated service is discussed. It is shown that the formal description of the service using temporal logics will perform rigorous verification service and eliminate design errors
Keywords: *Web-service, SOAP, WSDL, OASIS, verification, temporal logics, Computational Tree Logic, contradictions*

Обговорюються питання, пов'язані з удосконаленням конструкції мультисервісної SOA мережи. Розглядуються питання про створення комплексних послуг, їх інструментування та хореографії. Обговорюється проблема, що виникає при створенні інтегрованих послуг. Показано, що формальний опис послуги за допомогою тимчасових логік буде виконувати сувору послугу перевірки та усунення помилок проектування
Ключові слова: *веб-сервіс, SOAP, WSDL, OASIS, перевірка, тимчасові логіки, обчислювальна Дерево Логіка, протиріччя*

1. Introduction

Web-services are defined as self-describing, self-contained, modular applications which published, placed and interactive online in Internet. Currently, a large number of business applications based on the use of web-services. At the same time widely used approach with combining various Web-services for the creation new services. This approach takes into account SLA and provides required business logic.

Current technology, implementing web-services based primarily on SOA and its components, SOAP, WSDL, Registry (UDDI), Business Process Languages Layer (WSBPEL), Choreography Description Language, WS-CDL. Reference model of SOA systems is represented by organization OASIS.

Formation of complex service is split into two phases. First phase is development of the static model (orchestration). Second phase is description of the service interaction during their performance (choreography). Formal description of static models is performed using WSBPEL, dynamic models using WS-CDL.

Design goals of the static and dynamic models are the integration of the requirements of each component of an integrated service into the overall system. These requirements include format of message exchange, method for transmitting messages, team management, exception handling, etc.

Development of such mechanisms requires their strictly check thorough testing and verification. Presentation of specifications using languages WS BPEL and WS-CDL allows only formalizes requirements for integrated service. However, these tools do not allow checking correctness (validation) of the developed service. Temporal logic is one of the most powerful tools for analyzing interaction of components in an environment. Thus, purpose of this paper is developing the method of converting parts of the complex service specifications by means of temporal logics.

2. Literature review

Questions of verification and validation real-time service using temporal logic, such as web-services, touched upon in the researched works of D. Drusinsky (runtime monitoring and Verification of Systems with Hidden Information, Innovations in Systems and Software Engineering), Yde Venema (expressiveness and completeness of an interval tense logic.), Z. Manna and A. Pnuel, D M Gabbay and other. Questions of requirements of specification can be represented discover in [1, 2, 3]. Formal definition of Linear Temporal Logic present in [3].

3. Formalizing specifications of web-services

Formalizing of web-service specifications by means of temporal logics is the sequence of state changes using predicates and temporal operators. A distinctive feature of temporal logics is that they allow formalizing web-service behavior, given parallelism, as well as the temporal order of state changes.

Formalizing process consists of converting the set of claims submitted by the web-services specification of a set of atomic propositional (AP) of temporal logics (set P). Requirements of specification can be represented by two kinds of formulas [1–3]:

– path formula $f \models f(a_1 ... a_n)$ (statements are true for the all lifetime of the process);

– state formula $p \models P(a_1)$ (statements are true to a certain state of web-service).

The example of using path formula is statement "In the channel cannot be more than k messages" [4]:

$$f \models G(\neg(M_{k+1} : send) U (M_k : reseive)). \quad (1)$$

The example of using state formula is statement "Serverwaits for an incoming request" [4]:

$$p \models S : ready. \quad (2)$$

Fig. 1 shows fragment of client initialization and local search in SIP [4]. SDL diagrams formalizing by means of temporal logics is accomplished by defining the true statements for each state [1]:

1. Initial stage of registration of a new client opening a communication channel

$$Created \models C : connectionopen();$$

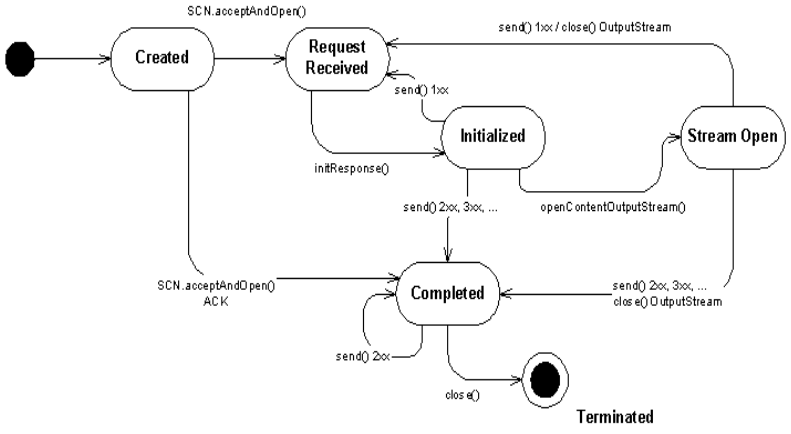

Fig. 1. The diagram of SIP server transaction of stateful processing

2. SIP server handles requests from the client, creates outbound communication channel or broadcast information responses:

$$\text{Re}\,quest\,\text{Re}\,ceive \models initresponse() \vee send()1xx;$$

3. After receiving request *initresponse*() protocol will look for the calling user and the formation response. According to specification responses can be final and information:

$$Initialized \models S : send(2xx, 3xx, 4xx....) \vee$$
$$\vee send(1xx) \vee connectOutputStream;$$

4. When connection was open, established a direct connection without SIP server $StreamOpen \models openConnectionOutputStream$;

5. Completion stage connection determined by the state

$$Compliate \models C : receive(2xx, 3xx, 4xx, 5xx, 6xx),$$

characterized by receiving a confirmation from the server, information about the status of the called subscriber;

6. State of Completion connection corresponds to the state $Termenate \models close$. The reason for connection is closed and is not defined in this case.

Accounting sequence of states is taken into account the chronological order of the events in the operation of web-service. There exist two different ways of taking into account the sequence of offensive states web-service by temporal logics tools [2, 3]:

– Linear Temporal Logic (LTL) allows describing the unique sequence of state changes in web-service;

– Computational Tree Logic (CTL) allows considering several sequences of state changes in web-service simultaneously, which depend on different parameters.

For linear temporal logic generates its own formula for each path scenario of web-service behavior (Fig. 2). The time is linear: at each moment is considered only one chain of events. Linear temporal logic separates each scenario of protocols behavior. On Figure 2 it is presented four scenarios of existence session for SIP web-service:

a) Server opens a new connection when a new user was detected. Server receives a user request and generates a response to user. For this scenario is characterized by a response of the final class (*2xx, 3xx, 4xx, 5xx, 6xx*). Request is considered to be completed after receiving a response.

b) Server opens a new connection when a new user was detected. Server user receives a request and generates a response to it, for this scenario is characterized by the use of the redirect server «Open Stream», which participates in the formation of user response. The state «Completed» indicates that the final response generated class and request processing is completed.

c) Server opens a new connection when a new user was detected. Server receives a user request and generates a response to user, for this scenario is characterized by the absence of the redirect server and the response of the Information (*1xx*). The state «Complited» indicates that the final response generated class and request processing is completed.

d) Server opens a new connection when a new user was detected. Server receives a user request and generates a response to user, for this scenario is characterized by the use of the redirect server that participates in the formation of user response information.

Computational Tree Logic allows describing behavior of several different scenarios of web-service in a single model. In this case, the path formula allows considering several scenarios of web-service behavior (Fig. 3).

To formalize the specification of web-services appropriate to use Linear Temporal Logic, this allows unambiguously formalizing every possible scenario of web-service behavior.

Application of Linear Temporal Logic as a specification formalism will also solve the problem of test specification for the presence of inconsistencies. Building LTL formulas is performed in accordance with the syntactic and semantic rules. Subsequent analysis and transformation of LTL performed on the basis of equivalent transformations of LTL.

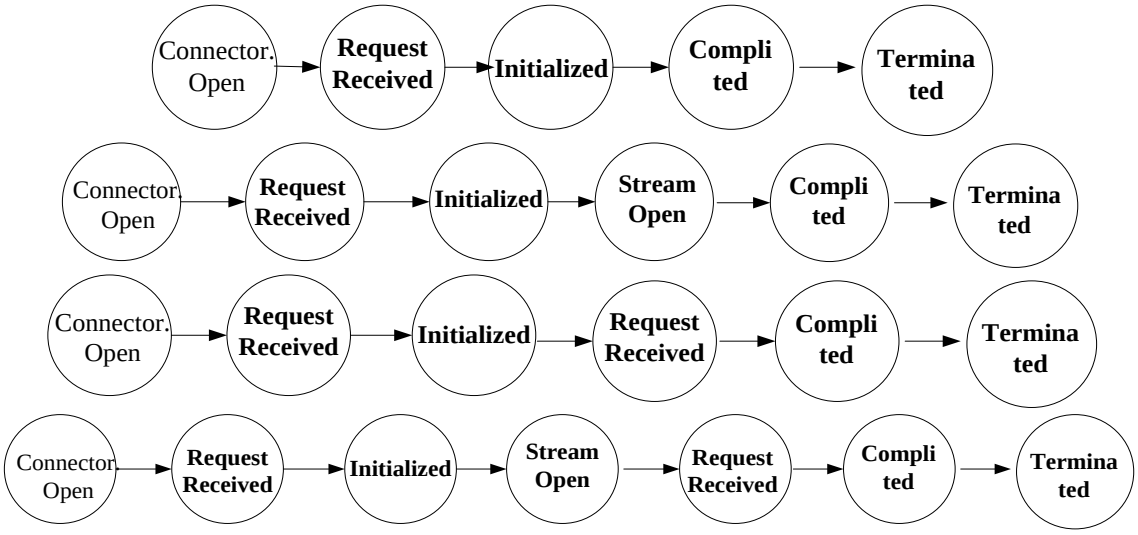

Fig. 2. Formalizing behavior of SIP web-service using LTL for formula $G(connector.open \rightarrow F(ter\mu inate))$

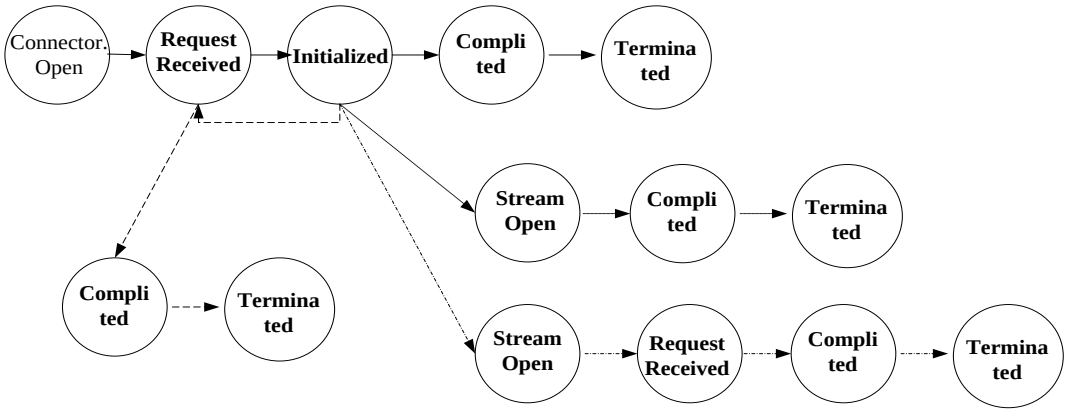

Fig. 3. Formalizing behavior of SIP web-service using CTL for formula $AG(connector.open \rightarrow AF(terminate))$

The following definitions are established the set of base formulas that can be expressed in a Linear Temporal Logic [3].

Formally, temporal logic is defined as:

$$TL = < A, O, C >, \quad (3)$$

where A – temporal logic alphabet; O – temporal operators set; C – logical connectives.

Alphabet is the basic concept of Linear Temporal Logic. Atomic statements form elements of the alphabet. Atomic propositional are assertion is true in the specification of web-service. Set of atomic propositional includes the basic elements of web-service, such as a server (S), client (C), message (M) and statements about them, such as server availability $(S:ready)$, client activity $(C:active)$, message sending $(M:send)$ etc. They form the alphabet of temporal logic P, ($AP \in P$).

Temporal operators are [1, 2, 5]:

– operators X («next moment»): condition must be executed in the next step. For example, the statement «after sending a message SYN should be generated ACK» can be represented as follows: $(C:SYN) \rightarrow X(S:ACK)$.

– operators F («for the future»): condition must be executed in one of the futures steps. For example, the statement «if the channel is open, it will be obviously closed» can be represented as follows $EG((H:Open) \rightarrow F(H:Close))$.

– operators G («always», «globally»): condition must be executed at each step. For example, the statement « TCP server port must be open» [6] can be represented as follows: $G(D:Enable)$.

– operators U («until»): condition ϕ must be performed continuously, or until the first condition ψ is met. For example, the statement «message will be discarded until the buffer is full» [2] can be represented as follows:

$$G(M:Drop)U(B.Buffersize \geq \max \mid \max = r).$$

– operators R («release»): the condition ψ should be performed at a certain way until the condition ϕ. For example, the statement «connection (H) will be in state *connection* until not come message *Bue*» can be represented as follows: $(M:Bue)R(H:connection)$

Logical operators are conjunction, disjunction, implication, negation and compliance operators.

Logical link between linear temporal logic formulas are defined by the following set of rules [2, 3]:

1. p is a formula for all $p \in AP$.

2. If ϕ – formula, then $\neg\phi$ – formulas;

3. if ϕ and ψ – formula, то $\phi \vee \psi$ and $\phi \wedge \psi$ – formulas;

4. If ϕ – formula, then $X\phi$, $F\phi$, $G\phi$ – formulas,

5. If ϕ и ψ – formulas, then $\phi U \psi$, $\phi R \psi$ – formulas. (4)

However, the state of web-services can be defined not only binary conditions, but also the natural numbers (for example package time to live, TTL, round retransmit time, retry count, number of source synchronization and other) [2, 6, 7]. A number of extensions introduced to account such conditions of temporal logic. The extension has the following notations: $(p : meaning = \geq, \leq x \mid x = number)$ – atomic propositional (p) takes a value higher or less than proposed in specification (*number*). This extension is not contrary to the basic axioms and theorems of temporal logics.

4. Construction of temporal logic for web-services.

Using a set of rules, described in the above, could be formulating the following sequence of steps for constructing LTL, describing the behavior of web-service.

1. Formed temporal logic alphabet:

– Select of all elements of web-service (for example, the message sender, receiver, etc.);

– Select of the main state (message: sent, session: set, etc.).

2. Defined connections between the elements of alphabet.

It is noted that the temporal operators have higher precedence than the logical connectives, thus for formalizing requirement of «sending a message INVITE affords confirmation *confirmation*» formulated statement: $G(send \rightarrow F(confirmation))$.

In order to improve the formalization of specifications based on temporal logics are encouraged to apply commonly used formalisms as templates. For example, the lack of blocking («reachable from each state there exists the possibility of functioning web-service») represented as follows: $GFtrue$.

For secure communication a channel following assertion is true:

$G(send \rightarrow X(\neg send \cup receive))$ – if executed send, then the next state in the future will *receive*, and *send* until this moment will not run.

The following formula actual for mutual exclusion:

$G(\neg(S_1 : crit \wedge S_2 : crit))$ – two processes cannot be in the critical section simultaneously.

6. Semantics definition of linear temporal logic

The main task of temporal logic is an algebraic description of the web-service behavior. Behavior represented by the set of temporal logic formulas describing the sequence of state changes. Linear temporal logic formulas should be formed in accordance with the semantic rules [2, 3, 5].

To solve the problem of constructing a semantically correct structures of linear temporal logic formula to be determined in terms of the model:

$$M = (S, R, Label), \quad (5)$$

where S – non-empty finite set of states (atomic propositional); $R : S \rightarrow S$ takes an element $s \in S$ only the one next $R(S)$; *Label* associates with each state $s \in S$ of atomic sentences $Label(s)$ true in s.

State s specified vertices, and function $R(S)$ is determining connection between vertices. Visual representation of the model is represented of state graphs.

For example, temporal logic formula.

$$X(G(STAT) \rightarrow (C : RETR) \rightarrow \\ \rightarrow (S : messageDelete \mid x) U (C : DELE)), \quad (6)$$

This formula describes a scenario of deleting messages in accordance with the specification of web-mail service POP3 [8] presented in the following graph states (Fig. 4).

Constructing semantically true formulas of linear temporal logic for the specification defined on a subset of natural language are performed as follows:

1. Within the specification released temporal logic alphabet.

2. Specifies the logical link between the elements of temporal logic alphabet.

3. Determined by the presence of parallel execution of multiple logical connectives.

4. Create chronological sequence between logical connectives.

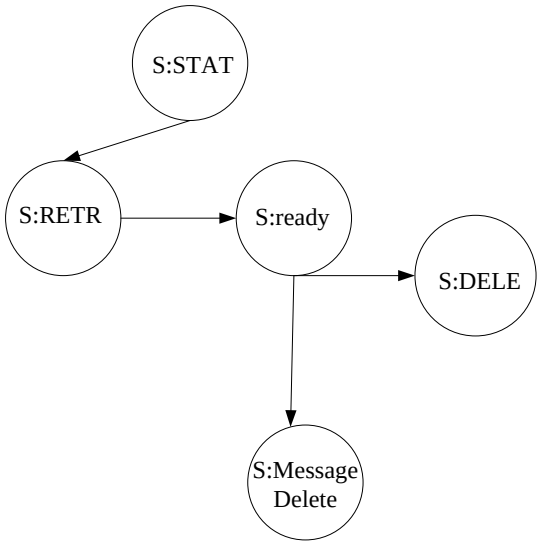

Fig. 4. Graphical interpretation of the formula 6

In constructing formulas of temporal logic for the specification of protocols specified as SDL (UML) diagrams are the following steps:

1. Alphabet of temporal logic, each element of which corresponds to the SDL (UML) state, is generated.

2. Features of forward and reverse the incidence diagram are considered. The logical links between the elements of the alphabet of temporal logic are determined.

3. Presence of parallel processes is determined.

4. Chronological sequence between logical connectives is built.

For example, the statement «the request must be confirmed» can be visually represented as follows (Fig. 5):

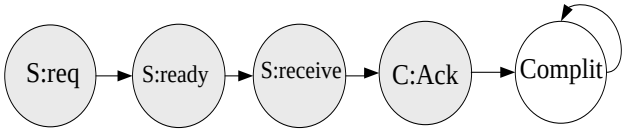

Fig. 5. Interpretation of formula $G(M:req \to F(M:Ack))$

The sequence of events corresponding to the statement « the connection is established after receiving confirmation » is presented at Fig. 6:

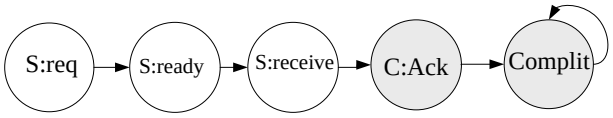

Fig. 6. Interpretation of formula $M:Ack \to X(Complit))$

7. A consistency check

As shows above, the development and modification of protocols possible emergence like a series of defects in contradiction specification is requirements [9].

Specifications must be presented a set of every temporal logic formula in terms of satisfaction relation ($|=$) for solving the problem of testing the consistency of [2].

Definition 1. The formula is given between a state *s* and an element of the formula ϕ, which must be true in this state ($s|=\phi$) [2, 10]:

$$\begin{aligned}
&s|=p \Leftrightarrow p\in Label(s);\\
&s|=\neg\phi \Leftrightarrow \neg(s|=\phi);\\
&s|=(\phi\vee\psi) \Leftrightarrow (s|=\phi)\vee(s|=\psi);\\
&s|=(\phi\wedge\psi) \Leftrightarrow (s|=\phi)\wedge(s|=\psi);\\
&s|=X\phi \Leftrightarrow R(s)|=\phi;\\
&s|=G\phi \Leftrightarrow \phi|=true;\\
&s|=(\phi U\psi) \Leftrightarrow \exists j\geq 0: R^j(s)|=\\
&\quad =\psi\wedge(\forall 0\leq k<\mathrm{j}{:}\mathrm{R}^k(s)|=\phi).
\end{aligned} \qquad (7)$$

s' a direct descendant of s, $R(s)=s'$, if $R^n(s)=s'$ for $n\geq 1$, then state s' called descendant.

The next step allocated state and formulas corresponding to each state and is determined by the type of formula (path formula or states formula). Consistency check executed after this [3, 10].

Definition 2. Two formulas of temporal logic are consistent if there are no such states for which formulas are mutually exclusive.

For example, the two statements: «reserved channel cannot be blocked» $H.reserv|=Gactive$ and «channel is blocked after timeout reserved» $H.reserv|=time-out\to\neg(active)$.

Consistency relations are verified path formulas and state formulas. Before checking the consistency holds the equivalent transformation formulas for the purpose of simplification. Equivalent transformation rules are divided into three categories [3].

Duality:

$$\begin{aligned}
&(\phi\to O\psi)\Leftrightarrow\phi\neq\psi;\\
&\phi\wedge\psi=\neg(\neg\phi\vee\neg\psi);\\
&\phi\vee\psi=\neg(\neg\phi\wedge\neg\psi)\\
&\phi R\psi=\neg(\neg\phi U\neg\psi);\\
&\neg G\phi=F\neg\phi;\\
&\neg F\phi=G\neg\phi;\\
&\neg X\phi=X\neg\phi.
\end{aligned} \qquad (8)$$

Absorption:

$$FGF\phi=GF\phi. \qquad (9)$$

Commutation and expansion:

$$\begin{aligned}
&\phi U\psi=\phi\wedge(X\psi);\\
&X(\phi U\psi)=(X\phi)U(X\psi);\\
&G\phi=\phi\wedge XG\phi;\\
&F\phi=\phi\vee XF\phi.
\end{aligned} \qquad (10)$$

Example

For example, it is considered a communication channel between two users: sender (*S*) and receiver (*R*). The equipment equipped an output buffer of the sender (*out*), and receiver – input (*in*). Both buffers have unlimited capacity.

If *S* sends a message *m* to *R*, then it puts the message in your output buffer *out*. The receiver receives the message by removing them from their input buffer *in* [3]. Thus, the alphabet consists of a temporal logic, atomic propositions set: message is in the buffer in the message is *in* the buffer *out*: $AP=\{m...m', m...m'\in out, m...m'\in in\}$, where m - message (Fig. 7).

Fig.7. Example of interaction between two network elements (sender and receiver), unidirectional communication channel

The following requirements for a communication channel using formulas of temporal logic are formalized [11, 12]:

«Message cannot be in both buffers simultaneously»:

$$G(m:out \wedge \neg m:in)\,. \quad (11)$$

«Channel does not lose messages»:

$$G(m:out \rightarrow F(m:in))\,. \quad (12)$$

«After receiving confirmation that the message has been delivered to the recipient, is its removal from the buffer .out »:

$$G((m:out \wedge m:in)U\neg(m:out))\,. \quad (13)$$

«Channel preserves the order»:

$$G(m:out \wedge \neg m':out \wedge F(m':out) \rightarrow \\ \rightarrow F(m:in \wedge \neg m':in) \wedge F(m':out))). \quad (14)$$

For consistency checks in accordance with the formulas 7 were made following the rules in the process of true web-service:

$$S.out \models G(m:out \wedge \neg m:in) \Leftrightarrow \\ \Leftrightarrow S.out \models m:out \wedge S.out \models \neg m:in; \quad (15)$$

$$S.hannel \models G(m:out \rightarrow F(m:in)) \Leftrightarrow \\ \Leftrightarrow m:out \neq (m:in); \quad (16)$$

$$S.out \models G((m:out \wedge m:in) \rightarrow F\neg(m:out)) \Leftrightarrow \\ \Leftrightarrow S.out \models (m:out \wedge m:in) \wedge S.out \not\models \neg(m:out); \quad (17)$$

$$S.hannel \models G(m:out \wedge \neg m':out \wedge F(m':out) \\ \rightarrow F(m:in \wedge \neg m':in) \wedge F(m':out))) \Leftrightarrow \\ \Leftrightarrow m:out \wedge \neg m':out \wedge F(m':out) \neq \\ \neq (m:in \wedge \neg m':in) \wedge F(m':out). \quad (18)$$

According to the definition of execution formula 15 and 17 contain a contradiction:
$S.out \models G(m:out \wedge \neg m:in)$ and
$S.out \models (m:out \wedge m:in)$, and $S.out \not\models \neg(m:out)$ and
$S.out \models m:out$.

Thus, the application of temporal logics allows detecting the contradictions in the original specification.

8. Conclusion

An algorithm formalization of web-services is constructed on the basis of temporal logics. The algorithm is based on the theory of formal grammars that allows you to formally introduce the execution of processes in the system model, to establish a cause-and-effect relationships, as well as to evaluate the correctness of the system as a whole and its individual components. Proposed method has different computational power. It allows using a particular type of language depending on the objectives of the study.

References

1. Stirling, C. P. Modal and temporal logics [Text] / C. P. Stirling. – Handbook of Logic in Computer Science. – Oxford University Press. – 1992. – Vol. 2. – P. 477–563.

2. Lafuente, A. Directed Search for the Verification of communication protocols [Text] : Doctorial thesis/ A. Lafuente. – Institute of computer science, University of Freiburg, 2003. – 157 p.

3. Rosenberg, J. SIP: Session Initiation Protocol. RFC 3261 [Text] / J. Rosenberg, H. Schulzrinne, U. Columbia, G. Camarillo. – East Hanover, 2002. – 269 p.

4. Rosenberg, J. An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing RFC 3581 [Text] / J. Rosenberg, H. Schulzrinne. – Parsippany, 2003. – 13 p.

5. Stirling C. P. Modal and temporal logics for processes [Text] / C. P. Stirling. – LNCS 1043, 1996. –P. 149–237. doi: 10.1007/3-540-60915-6_5

6. Kaivola, R. Using compositional preorders in the verification of sliding window protocol [Text] / R. Kaivola. – Lecture Notes in Computer Science, 1997. – P. 48–59. doi: 10.1007/3-540-63166-6_8

7. Postel, J. Transmission control protocol. RFC 793 [Text] / J. Postel. – California, 1981. – 85 p.

8. Myers, J. Post Office Protocol. Version 3. RFC 1939 [Text] / J. Myers, C. Mellon, M. Rose. – Pittsburgh, 1996. – 23 p.

9. Lawrence, S. Software Engineering: Theory and Practice (2nd ed.) [Text] / S. Lawrence. – Upper Saddle River, NJ: Prentice Hall, 2001. – 630 p.

10. Clarke, E. M. Automatic verification of finite-state concurrent systems using temporal logic specifications [Text] / E. M. Clarke, E. A. Emerson, A. P. Sistla // ACM Transactions on Programming Languages and Systems. – 1986. – Vol. 8, Issue 2. – P. 244–263. doi: 10.1145/5397.5399

11. ITU-T Recommendation H.248. Series H: Audiovisual and Multimedia Systems: Infrastructure of audiovisual services – Communication procedures: Gateway control protocol [Text] / Geneva: International Telecommunications Union, 2000.

12. Greene, N. Megaco Protocol Version 1.0. RFC 3015 [Text] / N. Greene, A. Rayhan. – Ottawa, November, 2000. – 179 p.

References

1. Stirling, C. P. (1992). Modal and temporal logics. Handbook of Logic in Computer Science. Oxford University Press, 2, 477–563.

2. Lafuente, A. (2003). Directed Search for the Verification of communication protocols. Institute of computer science, University of Freiburg, 157.

3. Rosenberg, J., Schulzrinne, H., Columbia, U., Camarillo, G. (2002). SIP: Session Initiation Protocol. RFC 3261. East Hanover, 269.

4. Rosenberg J., Schulzrinne, H. (2003). An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing RFC 3581. Parsippany, 13.

5. Stirling, C. P. (1996). Modal and temporal logics for processes. LNCS, 149–237. doi: 10.1007/3-540-60915-6_5

6. Kaivola, R. (1997). Using compositional preorders in the verification of sliding window protocol. Lecture Notes in Computer Science, 48–59. doi: 10.1007/3-540-63166-6_8

7. Postel, J. (1981). Transmission control protocol. RFC 793. California, 85.

8. Myers, J., Mellon, C., Rose, M. (1996). Post Office Protocol. Version 3. RFC 1939, Pittsburgh, 23.

9. Lawrence, S., Prentice, H. (2001). Software Engineering: Theory and Practice (2nd ed.) Upper Saddle River, 630.

10. Clarke, E. M., Emerson, E. A., Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, 8 (2), 244–263. doi: 10.1145/5397.5399

11. ITU-T Recommendation H.248. Series H, Geneva (2000). Audiovisual and Multimedia Systems. Infrastructure of audiovisual services Communication procedures. Gateway control protocol. International Telecommunications Union.

12. Greene, N., Rayhan, A. (2000). Megaco Protocol Version 1.0. RFC 3015. Ottawa, 179.

Рекомендовано до публікації д-р техн. наук Дуравкін Є. В.
Дата надходження рукопису 27.02.2015

Mohammed Khodayer Hassan Al-Dulaimi, PhD, Department of Software Engineering, Instructor at Al-Rafidain University College, Iraq, Bagdad, Hay Al Mustansurya
E-mail: mohamdhassan2004@yahoo.com

Aymen Mohammed Khodayer Al-Dulaimi, graduate student, Department of Telecommunication Systems, Odessa National Academy of Telecommunications name after A. S. Popov, st. Kovalska, 1, Odessa, Ukraine, 65000
Email: aymenaldulaimi@yahoo.com

УДК 678.057
DOI: 10.15587/2313-8416.2015.38997

ЧЕРВ'ЯЧНО-ШЕСТЕРЕННИЙ ЕКСТРУДЕР ПРИ ПЕРЕРОБЦІ ПОЛІМЕРНИХ МАТЕРІАЛІВ

© Д. А. Степанюк, М. П. Швед, Д. М. Швед

У статті представлено аналіз основних схем екструзійних установок, та наведено основні параметри, які впливають на пульсацію продуктивності в черв'ячному екструдері без шестеренного насосу.
Розглянуті аналіз та експериментальні дослідження продуктивності та її коливання в класичному черв'ячному екструдері та в тому ж екструдері з шестеренним насосом
Ключові слова: *екструдер, шестеренний насос, гомогенізація, дозування, пульсація, полімер, градієнт тиску, черв'як, зазори, формуючий інструмент*

The paper presents the analysis of the main schemes of extrusion plants, and the main parameters that affect the performance pulsation in worm extruder without gear pumps.
It is considered analysis and experimental research of productivity and its fluctuations in the classic worm extruder and in the same extruder with gear pump
Keywords: *extruder, gear pumps, homogenization, dosage, pulsation, polymer, pressure gradient, worm, gaps, shaping tool*

1. Вступ

Зростаючі обсяги виробництва та переробки пластичних мас вимагають від галузі полімерного машинобудування оптимізації процесів та ширшого використання ресурсоенергозберігаючих технологій.

Найбільш поширеною залишається одночерв'ячна екструзія. При цьому одночасно виконуються наступні операції: живлення, стискання, плавлення твердого полімеру, змішування, створення тиску та дозування розплаву. Всі названі процеси тісно пов'язані між собою і виконуються в черв'ячному екстрадері одним робочим органом – черв'яком, що ускладнює оптимізацію процесів [1]. Така конструкція екструдера має суттєвий недолік, так як присутні коливання продуктивності, які призводять до перевитрати сировини і енергії.

2. Постановка проблеми

Метою роботи є визначення, аналіз та експеремнтальні дослідження продуктивності та її коливання в класичному черв'ячному екструдері та в тому ж екструдері з шестеренним насосом.

3. Літературний огляд

Каскадні установки в порівнянні з традиційними черв'ячними екструдерами характеризуються кращими питомими показниками та широкою номенклатурою перероблюваних матеріалів [2].

Використання каскадних установок дозволяє встановлювати раціональні режими роботи виділених операцій при якісному веденні усього технологічного процесу. При створенні таких екструдерів необхідно вирішувати ряд наступних завдань [3]:

– виділення з технологічного процесу переробки основних операцій;

– визначення можливості їх інтенсифікації;

– вибір відповідних агрегатів та вузлів, які забезпечують проведення і керування цими операціями;

– визначення можливостей погодження їх сумісної роботи.

На першій стадії каскадних агрегатів зазвичай використовують одночерв'ячний, дисковий, або двочерв'ячний екструдери. Як правило, друга стадія представлена одночерв'ячним екструдером. З метою ефективної переробки полімерів в якості першого каскаду використовують дискові, або двочерв'ячні екструдери, які забезпечують більш якісне змішування і гомогенізацію розплаву, ніж одночерв'ячний екструдер.

Технологія багатостадійної екструзії використовується фірмами Battenfeld, Barmag (Німеччина), Buss (Швейцарія), Mitsubishi Petrochemical (Японія) та іншими.

Використання каскадних установок має ряд переваг. По-перше, розмежування операцій дає змогу автономно і більш точно корегувати технологічні