UDC 621. 93

AN APPROACH TO DYNAMIC WEB SERVICE COMPOSITION

Dmytro S. Pukhkaiev, Tetiana M. Kot, Larysa S. Globa

National Technical University of Ukraine "Kyiv Polytechnic Institute", Kyiv, Ukraine

Today, changeable requirements to modern web-oriented services demand their fast development and constant re-engineering. This is realized via dynamic composition of services, allowing to estimate changes of both functional and non-functional service parameters. The last ones are considered using Web Services Agreement technique. Nevertheless, state-of-the-art SLA-aware methods are not able to consider all classes of non-functional parameters. They also don't provide service run-time support and dynamic reconfiguration. The novel approach to dynamic Web Services Composition, extending SLA with QoS ontology, is described in the paper. It includes service selection agents that use the QoS ontology and WS-Agreements, allowing agents to choose the most appropriate service based on quality preferences exposed by service consumer. The proposed approach allows performing dynamic WS composition based on SLA, providing required values of QoS parameters, improving general QoS and decreasing service development and re-engineering time.

Introduction

Nowadays the way software applications are designed, architected, delivered and consumed has significantly changed. Service-Oriented Computing [1] is the computing paradigm that uses services considered to be fundamental elements to support the development of rapid, low-cost and easy composition of distributed applications. Services are introduced as autonomous platform-independent computational elements that can be described, published, discovered, orchestrated and programmed for the purpose of developing massively distributed interoperable applications. Service oriented computing can be considered the framework for service publishing, discovery, binding and composition. It relies on the Service-Oriented Architecture (SOA) [2], which is a way of reorganizing software applications and infrastructure into a set of interacting services.

Web services [3] are a case in which XML standards are utilized and there is a technology that goes from messaging up to coordination of loosely coupled elements.

One of the most popular approaches for design and implementation of web-oriented applications is Business Process Management (BPM), which "supports business processes using methods, techniques and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information" [4]. BPM life cycle consists of four stages: Process design, System configuration, Process enactment and Diagnosis.

One of the most considerable advantages of web services (WS) technology utilization is the possibility of connecting them together in order to implement high level business-process [5]. Web Service Composition (WSC) [6] is a method that allows performing such connections.

Fast changes of requirements demand dynamic composition and reconfiguration of services, considering required values of functional [7] and non-functional [7] parameters (which represent Quality of Service (QoS) characteristics, i.e. reliability, availability, response time, cost, performance, etc.).

Hence, WSC unites stages of Process enactment and Diagnosis of BPM life cycle.

To consider QoS parameters both on Process enactment and Diagnosis stages is one of WSC tasks to be performed. The most convenient way to contain and monitor their values is using Service Level Agreement (SLA) [8].

In addition, another WSC issue regarding QoS parameters is that despite the existence of various dynamic composition approaches, there is no approach combining evaluation of different classes of QoS parameters during performing composition and none of them is able to meet all QoS requirements, described further.

This paper presents a novel SLA-aware approach for WSC, allowing taking into consideration required QoS parameters.

The paper is structured as follows: Section 2 provides a background on different stages of BPM life cycle and its impact on WSC. Section 3 contains state-of-the-art analysis of WSC approaches. Section 4 presents the proposal for SLA-aware WSC approach. Section 5 summarizes the work and provides perspectives for future research.

Background

Workflow design and enactment

Graphical standards formalize computational independent workflows, their possible flows and transitions in a diagrammatic way. Computational independent workflows are designed using graphical notations such as BPMN 2.0[9], UML AD[10], USDL[11].

The workflow enactment stage consists of several steps. First step provides an orchestration [12] for web services when execution standard such as WS-BPEL [13] is applied. It is used to define the sequence of invocations of existing web services and the kind of the process interaction with external participants.

WS-BPEL specification singles out two different types of language abstract and executable. WS-BPEL Abstract Processes are introduced to hide the information project owner wishes to conceal. WS-BPEL Executable Processes are introduced to be fully described.

Next step is WSC itself.

Web Services Composition

WSC is a method to connect different web services used for creating high level business architecture by compiling of atomic web services in order to provide functionalities that are not available during design. Consequently, there is a possibility to develop a new functionality by simply reusing of components that are already available, but unable to complete a task successfully on their own.

Various authors classify WSC approaches. In [6], WSC approaches are grouped as follows:

– Static and Dynamic Composition;
– Model Driven Service Composition;
– Declarative Service Composition;
– Automated and Manual Composition;
– Context-based Service Discovery and Composition.

A composite service is a set of individual services effectively combined and reused to achieve a desired effect. Automatic WSC consists of four phases: Planning, Discovery, Selection, and Execution [14]. The first phase involves creating a plan, i.e., sequence of services in desired composition. The second phase embodies service discovery due to the plan. After discovery of suitable services, the selection phase starts. It embodies a selection of the optimal composition from available combinations of individual web services considering non-functional parameters like QoS properties. The final phase involves services execution due to the plan. If some service is not available, they substitute one another.

First of all, WSC tools must ensure that functional parameters have adequate values, this means that consumers input information lead to consumers required output. Concurrently, the workflow management system must ensure that predicates and requisites match in each step of the workflow lifecycle. Although, the functional Web service composition has been widely studied in literature, the assessment criteria related to non-functional attributes in particular, significantly increases the number of composition requirements. QoS parameters can be used for ranking the composite service or for further prune of results.

Storing QoS parameters can be done directly in BPEL-file. This method provides very low ability to implement reconfiguration of any QoS parameter of a composed service. Furthermore, it is not trivial to monitor QoS parameters during service execution.

In agent-based solutions, agencies gather QoS data from agents, store, aggregate, and present it to agents [15].

The approach which solves these problems is introduced in [16]. Distinctive feature of this approach is utilization of SLA via WS-Agreement [17] during both workflow enactment and workflow analysis stages.

WS-Agreement

To succeed WS providers have to guarantee declared capabilities related to services they develop. A guarantee highly depends on resource usage which means that the service consumer must request situational guarantees from the service provider. Moreover, in order to avoid losses associated with guarantees violation service consumer should be notified during run-time. Associated guarantees are specified in an agreement between a service consumer and a service. This agreement can be formally specified using the WS-Agreement Specification [18].

WS-Agreement is an XML-based document containing descriptions of functional and non-functional parameters of a service oriented application. It consists of two components that are the agreement Context and Terms and Conditions of the agreement [17].

Workflow analysis

Considering workflow analysis methods and tools, two types of analysis, both considering computational workflow, can be specified [19]:

– Design time analysis (simulation and verification);
– Runtime analysis (i.e., process mining based on execution logs).

QoS parameters are transforming relatively frequently, either because of internal changes in web services or because of changes in their environment (i.e., system load has changed). During composite service execution, some component services may change values of their QoS properties on-the-fly; some of them may become unavailable, while others may emerge. As a result, approaches where Web services are statically composed are inappropriate. Runtime changes should be taken into account.

Dynamic composition, taking into account runtime QoS transformations, should be applied. In the proposed approach runtime changes of QoS parameters such as execution time, reliability etc. are taken into account (presented in next sections).

QoS-Based SLA-Aware Comparison of WSC Approaches

An overview of several approaches for modeling web service quality composition, presented in details in [20] has shown that state-of-the-art QoS WSC models and solutions are far from required one. None of the presented approaches can meet all QoS characteristics. The most crucial characteristic is the ability to support all types of QoS parameters. Markov chains [21] and Quality Vector solutions [22, 23] are not applicable in this regard while Ontology-and agent-based solutions [15] meet some QoS characteristics although there are still some types of QoS parameters i.e. non-measurable parameters [24] that should be taken into account.

Table 1 shows SLA enables a convenient way to performing WSC and monitoring of composed service. Applying SLA to methods considered in [20].

Table 1. QoS-based SLA-aware comparison of WSC approaches

Requirements	WSC Approaches			
	Markov Chains	Quality Vector	Agent-oriented	Ontology-based
Objective QoS	+	+	+	+
Subjective QoS	-	-	-	+
Run-time support	+	+	+	+
QoS assignment	+	+	+	+
Requirements considering level	Low	Average	High	High

Table 1 shows that Ontology-based WSC approach combined with SLA-awareness should be able to perform the most reliable WSC of all approaches presented in [20].

An approach to dynamic web-service composition

General description

The workflow on the enactment stage lacks implementation. WSC is intended to solve this problem by finding appropriate services, based on workflow description and composing them into single application – composite web service. The workflow on the enactment stage is described by a model presented in the section below.

SLA-aware WSC approach partially implements agent-based architecture and is realized in the Web Services Agent Framework (WSAF) [15]. Its description is provided below.

An approach includes service selection agents that use the QoS ontology and WS-Agreements allowing agents to choose the most appropriate service based on quality preferences exposed by the service consumer.

When consumer application, built with WSAF, requests to use a service, the agent is called for communication with service. An agent is created for each service to expose the service interface, enlarged with functionality, to capture the consumer's QoS preferences and functional requirements and provide agencies or other agents query for a suitable match. The service agent can determine values of objective QoS-attributes (reliability, availability and execution time, for instance) and get the user feedback for subjective attributes (which is an indicator of appropriateness for requested QoS parameters compared with values of these parameters, specified by consumer on the stage of workflow design). Afterwards, values of these QoS are transferred to the appropriate agencies.

A typical consumer-to-agent interaction and control flow is described below:

1) upon initialization, WSAF sets up all configured agencies;

2) providers register service implementations with WSAF by configuring each service in terms of WSDL URIs, service domains, and the service's advertised QoS requirements. Each configured service interface has an agent;

3) the consumer application creates a local proxy object for the service agent; the consumer invokes the proxy with its WS-Agreement;

4) the agent uses business process file and WS-Agreement to load and run its script. The script typically consults the QoS and service ontologies to complete its configuration;

5) the agent selects the service implementation based on agency data, and then dynamically creates a proxy object for each selected service;

6) the consumer invokes the agent's service operations. Each invocation is forwarded to the service proxy, while being monitored by the agent; when the service responds, the agent inserts appropriate data to the relevant agencies.

The service agent finds services matching the given interface, using UDDI. Then, it applies WS-Agreements on the available quality data, providing service implementations ranking.

Compared to general WSAF, appliance of SLA provides:

– substitution of consumer and provider's policies by WS-Agreements, providing the composition with a set of functions that makes the policies usage redundant.

– adding the monitoring stage: to evaluate QoS parameters during composed WS execution and its reconfiguration in case of QoS parameters violation.

SLA usage in the proposed approach provides generalized way of storing QoS parameters for service providers. It helps performing WSC on the stage of workflow enactment as well as on the stage of workflow analysis. Another advantage compared to existing Ontology-based approach is enabling of run-time monitoring of the composite service.

Workflow and WSC models

Workflow and WSC models, providing proposed approach realization are presented in this section.

The workflow model on design stage is presented in [19].

A workflow model on enactment stage is developed and can be characterized by:

– name;
– executor;
– relative and interactive activities – workflow components;
– workflow parallel tasks, united into a single stage of execution;
– set of informational input objects;
– set of informational output objects;
– executor/executors;
– execution time;
– execution resources.

Mathematically, a description of the workflow on the enactment stage can be represented as:

$$EP = (E_{id}, I, Q) \quad (1)$$

where E_{id} is a set of workflow's identification objects, I is a set of informational input objects, Q is a set of quality parameters specified on the design stage.

E_{id} is a set of four parameters $\{E_{id,id=1,4}\}$ and consists of: $E_1 = N_{EP}$ – workflow name; $\mathrm{E}_2 = \mathrm{O}$ – a set of tasks; $\mathrm{E}_3 = \mathrm{Pl}$ – a set of partner links with tasks executors; $\mathrm{E}_4 = \mathrm{Ex}$ – workflow executor/executors.

Workflow executor is software. Human interaction is minimized on this stage. The executor model can be represented as:

$$Ex_{ij} = (N_{EP}, O) \quad (2)$$

where Ex_{ij} is a performer j of the task i; $N_{EP} = \{N_{EPij}\}$, N_{EPij} is a name of performer j of the task i; $O = \{O_{ij}\}$, O_{ij} is a task i, executed by a performer j.

Informational objects have the same representation as on the stage of the workflow design [19].

Q is a set of seven parameters $\{Q_{i,i=1,7}\}$ and consists of quality components specified for resulting application: Q_1 is performance, Q_2 is reliability, Q_3 is robustness, Q_4 is accessibility, Q_5 is availability, Q_6 is cost, Q_7 is additional QoS parameters (scalability, capacity, accuracy).

Considering of quality parameters allows to:

– control quality of an application – workflow realization;
– substitute tasks executors onto more appropriate ones during the application execution;
– increase the application quality.

In order to create the workflow implementation, application for web services dynamic composition is required. It can be characterised by:

– name;
– set of informational input objects;
– set of informational output objects;
– indicator for appropriateness of composite web service to the requested functionality;
– integral indicator of web service quality compliance.

Mathematically, the process of WSC can be represented as:

$$C = (I, O, F, Nf) \quad (3)$$

where I is a set of informational input objects;

O is a set of informational output objects;

F is an indicator for appropriateness of composite web service to the requested functionality. F has a Boolean type, it either has a state of appropriateness to the requested functionality or a state of inappropriateness;

Nf is an integral indicator of web service quality compliance. It can be represented mathematically as:

$$Nf = (P, Rl, Rb, Ac, Av, C, A) \quad (4)$$

where P is performance. It represents how fast a service request can be completed. It can be measured in terms of throughput, response time, latency, execution time, and transaction time. It is assumed that performance is measured in terms of execution time.

Rl is reliability. It represents the ability of a web service to perform its required functions under stated conditions for a specified time interval.

Rb is robustness. It represents the web service ability to function correctly even in the presence of invalid, incomplete or conflicting inputs. Web services should still work even if incomplete parameters are provided to the service request invocation.

Ac is accessibility. It represents whether the web service is capable of serving the client's requests.

Av is availability. It is the probability that the system is up.

C is cost of a web service.

A represents additional QoS parameters. They have less influence on the composite web service than previously mentioned ones, however they should be taken into account if possible, and include:

– scalability, representing the capability of increasing the computing capacity of service provider's computer system and system ability to process more users' requests, operations or transactions in a given time interval;

– capacity is the limit of simultaneous requests number which should be provided with guaranteed performance;

– accuracy is an error rate generated by the web service.

SLA-aware approach for WSC takes into account all previously mentioned parameters, both functional and non-functional.

Unlike other approaches, the latter covers a wider range of QoS parameters. In order to compose the best web service, satisfying required parameters, the latter from every web service, being evaluated for composition compliance, must be integrated into a single parameter called integral indicator of web service quality compliance.

The idea of composite WS development is based on ranking the influence of QoS parameters on WS quality and integration of each WS QoS parameters of the same type.

Table 2 presents how various QoS Parameter are integrated into a single one, depending on which WS composition pattern is applied, and ranking (R) of the influence of QoS parameters on WS quality.

Table 2. QoS-based SLA-aware comparison of WSC approaches

QoS Parameter	R	WS Composition Patterns			
		Sequence	Parallel	Switch	Loop
Performance	2	$\sum_{i=1}^{m} P_i$	$\max(P_i)$	$\sum_{i=1}^{m} p_j P_i$	P_i^k
Reliability	1	$\prod_{i=1}^{m} Rl_i$	$\prod_{i=1}^{m} Rl_i$	$\prod_{i=1}^{m} p_j Rl_i$	Rl_i^k
Robustness	5	$\prod_{i=1}^{m} Rb_i$	$\prod_{i=1}^{m} Rb_i$	$\prod_{i=1}^{m} p_j Rb_i$	Rb_i^k
Accessibility	4	$\prod_{i=1}^{m} Ac_i$	$\prod_{i=1}^{m} Ac_i$	$\prod_{i=1}^{m} p_j Ac_i$	Ac_i^k
Availability	3	$\prod_{i=1}^{m} Av_i$	$\prod_{i=1}^{m} Av_i$	$\prod_{i=1}^{m} p_j Av_i$	Av_i^k
Cost	*	$\sum_{i=1}^{m} c_i$	$\sum_{i=1}^{m} c_i$	$\sum_{i=1}^{m} p_j c_i$	$C_i k$
Scalability	7	$\max(Sc_i)$	$\sum_{i=1}^{m} Sc_i$	$\sum_{i=1}^{m} p_j Sc_i$	Sc_i
Capacity	7	$\max(Ca_i)$	$\sum_{i=1}^{m} Ca_i$	$\sum_{i=1}^{m} p_j Ca_i$	Ca_i
Accuracy	6	$\prod_{i=1}^{m} Acr_i$	$\prod_{i=1}^{m} Acr_i$	$\prod_{i=1}^{m} p_j Acr_i$	Acr_i^k

Practical implementation

The concept of web-service composition module was presented in [20]. This section describes main functions of the software environment for SLA-aware WSC, based on its architecture (Fig.1.), presented in previous work.

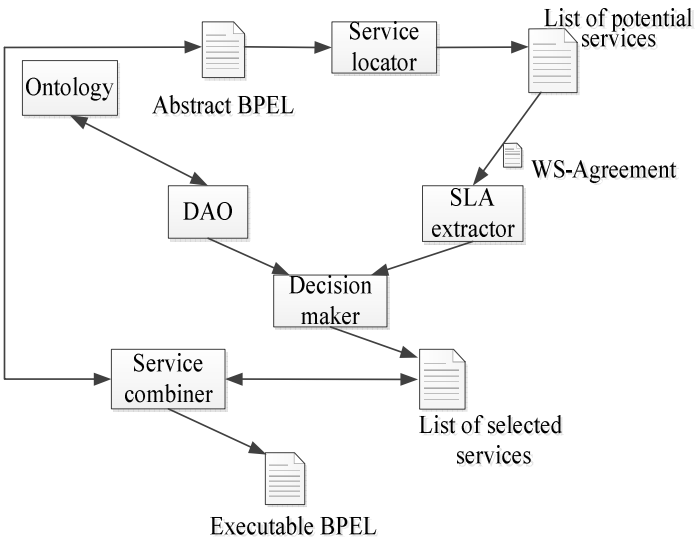

Fig. 1 Architecture of software environment for SLA-aware WSC

Software environment for SLA-aware WSC consists of Service locator, SLA extractor, Decision maker, Ontology module and Service Combiner.

Service locator is a module that extracts the information on functional parameters from abstract BPEL-file. It also searches the appropriate services in UDDI or service brokers (considering functional parameters). After that all found services are listed in potential services.

SLA extractor is a module, extracting SLA information from every service meeting functional parameters through WS-Agreement. It provides Decision maker module with extracted information.

Decision maker uses data received from SLA extractor and calculates rankings for services in accordance with rules, located in Ontology module. Then this module decides which service better suits the composite service.

Services chosen for composition are listed in QoS-aware services (if they meet QoS requirements). Otherwise they are excluded from the list.

Service Combiner performs WSC by importing chosen services into workflow file thereby enriching abstract BPEL-file with concrete services and making it executable.

On the workflow analysis stage the software environment for SLA-aware WSC performs the following functions:

– compares current QoS parameters with required values and values of QoS parameters in the list of QoS-aware services;

– monitors the indicator value for the appropriateness of composite web service to the requested functionality;

– in case of QoS violation, identifies a defective web service and changes it with the best-chosen web service from the list of QoS-aware services;

– in case of functional parameter violation, recomposes the web service;

– in case of determining that a WS from the list is better in terms of QoS parameters than one in previous WSC, puts "better" WS in the waiting list and recalculates the composition at the end of WS billing period. If the recalculated WS composition is "better" than the previous one, the previous WS is replaced by "better" one, thus WS re-composition is done.

Real world scenario

The concept of web-service composition module was presented in [20].

This section provides an example of a possible scenario for using the presented approach and clarification of software tool based on this example.

Let us assume a person who uses the tool for dynamic WSC (provider) aims to develop and provide service which helps its consumers to book a fully customizable vacation having a hotel, flight, taxi and cultural events pre-booked. He is not able to program such service or there is no much time for the development. Thus, using existing web services which can partially provide necessary functionality is a convenient option for fast application development.

The provider has various constraints regarding his tool, e.g. response time, cost etc. After successful registration in WSC System, the provider can either create a new Project or edit an existing one. After choosing an appropriate option, the provider can upload BPMN or BPEL file into the system for analysis. At this stage QoS constraints should be specified. Otherwise Service Locator will search for any services, satisfying functional parameters, extracted from uploaded BPEL (BPMN) file. WS-Agreement for the composite service is generated if QoS restrictions have been specified by the provider.

On fig. 2 BPMN diagram for Vacation Service is presented. Concrete workflows are omitted for simplicity.

Service locator extracts the information about functional parameters from BPEL-file which was either uploaded or generated from BPMN, and searches the appropriate services in UDDI or service brokers.

Now provider has a list with sets of web services capable to reach the provider's goal (in the given example different combinations of services, providing flight, hotel, taxi and cultural events booking, are presented). They are sorted by the integral indicator of web service quality compliance by default. The sorting is done transparently by Decision Maker module. The provider can define whether this stage needs human interaction (i.e. choosing a web service) or it should be done automatically. The last case means that the best service regarding the integral indicator of web service quality compliance would be applied as default. Then chosen services are purchased. And the provider has a functioning composite web service.

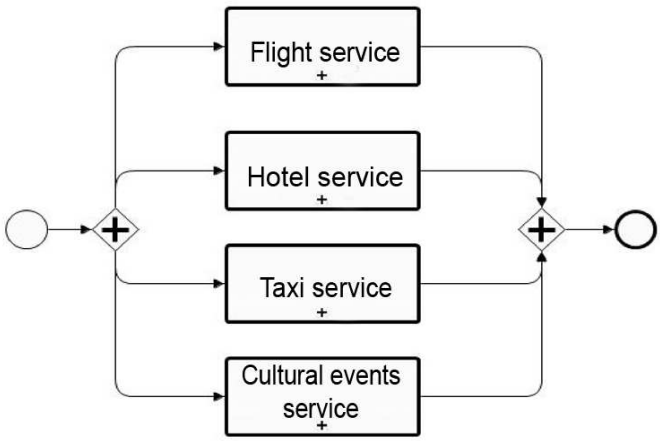

Fig. 2 Simplified BPMN diagram of provider's application

The suggested WSC Tool contains built-in monitoring and dynamic reconfiguration module, providing faster and more reliable service reconfiguration due to already composed a list of possible services.

QoS parameters of composite web service are under constant observation by WSC System. Let us assume that one of these parameters, for instance response time, is not appropriate for some period of time. QoS parameters of all services within the composite are re-evaluated. The inappropriate one is identified. After that the provider receives a notification with options to reconfigure struggling web service i.e. to choose a web service which satisfies non-functional parameters better than the previous one. Reconfiguration is performed on flight automatically if the provider has chosen the option of automatic reconfiguration in system settings. In case of violating functional parameters, composite web service is recomposed from scratch – violating functional parameters cannot be allowed, this means that application is not running properly.

In the case when web service for booking a hotel has less availability than specified by the provider, the overall composite service does not satisfy QoS constraints.

WSC System checks availability parameters of all four individual web services and identifies that hotel service is struggling. WSC System still contains a list of potential services for provider's task. These services can be compared to a failed web service in terms of QoS parameters. Manually or automatically unsatisfying hotel web service is replaced by the most appropriate one from the list. Then WSC System re-assembles composite web service. The provider again has a service which is fully functional and satisfies all requirements.

End user is provided with the web interface which gathers information from various web services thus being able to compose a customized vacation.

Conclusion and future work

QoS parameters are extremely important to evaluate. These parameters violation can cause substantial losses in financial or time expenses. The workflow management system must ensure that the predicates and requisites match in each workflow step.

Current state-of-the-art web service composition approaches have been analyzed. It has showed that none of them can meet all required QoS characteristics. This paper presents a novel SLA-aware approach for Web Service Composition which satisfies required QoS characteristics is. The proposed approach allows performing dynamic WS composition based on SLA, providing required values of QoS parameters, improving general QoS and decreasing service development and re-engineering time.

The future research will focus on implementation software WSC tool using Java technology, its testing and verification when WS development and reconfiguration. Availability and efficiency of the developed models, analysis methods and composition tool will be experimentally tested in the real world scenarios. Quantitative results on QoS and service development and re-engineering time will be measured to prove the proposed approach efficiency.

References

1. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web Services. Concepts, Architectures and Applications. – Springer, 2004. – 354 p.

2. M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann: "Service-Oriented Computing: State of the Art and Research Challenges"; IEEE Computer, 40 (November 2007 (2007), 11; P. 38 – 45.

3. M. Papazoglou and D. Georgakopoulos. "Service-Oriented Computing," *Communications of the ACM*, vol. 46, 2003. – P. 25-28.

4. W.M.P. van der Aalst, Benatallah B., Casati F. , Curbera F. , and Verbeek H.M.W.. Business Process Management: Where Business Processes and Web Services Meet. // *Data and Knowledge Engineering*. -2007. – 61(1). – P. 1-5.

5. Y. Jadeja, K. Modi, and A. Goswami. Context Based Dynamic Web Services Composition Approaches: a Comparative Study // International Journal of Information and Education Technology, vol. 2, Apr. 2012. – P.164-166.

6. S. Dustdar, and W. Schreiner, "A survey on web services composition," in Int. J. Web and Grid Services, vol. 1, No. 1, 2005. – P.1–30.

7. S.Bansal, A. Bansal, and M.B. Blake, "Trust based Dynamic Web Service Composition using Social Network Analysis," IEEE International Workshop on Business Applications for Social Network Analysis (BASNA 2010), August 2010. – P. 1-8.

8. C. Molina-Jimenez, J. Pruyne, and A. van Moorsel. The Role of Agreements in IT Management Software. Architecting Dependable Systems III, LNCS 3549. Springer Verlag, Volume 3549, 2005. – P. 36-58.

9. OMG Business Process Model and Notation (BPMN) [Online]. – 2011. – Available: http://www.omg.org/spec/BPMN/2.0/PDF

10. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, P. Wohed "On the suitability of UML 2.0 activity diagrams for business process modeling," Proceedings of the 3rd Asia-Pacific conference on Conceptual modeling APCCM '06, vol. 53, 2006. – P. 95-104.

11. Oberle, D., Bhatti, N., Brockmans, S., Niemann, M., Janiesch, C., "Countering Service Information Challenges in the Internet of Services," Journal of Business & Information System Engineering, vol.1, 2009. – P. 370-390.

12. H. Foster, S. Uchitel, J. Magee, J. Kramer, M. Hu "Using a rigorous approach for engineering Web service compositions: a case study," in Proceedings of the 2005 IEEE International Conference on Services Computing (SCC '05), 2005. – P.217-224.

13. OASIS (2007) Web Services Business Process Execution Language (WSBPEL) [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf

14. J. Cardoso, and A. Sheth, Semantic Web Services, Processes and Applications, Springer, 2006.

15. E.M. Maximilien and M.P. Singh., "A framework and ontology for dynamic web services selection," IEEE Internet Computing, vol. 8, Sep./Oct. 2004. – P. 84-93.

16. M. B. Blake, and D. J. Cummings, "Workflow Composition of Service-Level Agreements," in Proceedings of IEEE International Conference on Services Computing (SCC), July 2007,. – P.138-145.

17. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. (2007) Web Ser-vices Agreement Specification (WS-Agreement) Online]. Available: http://www.ogf.org/documents/GFD.107.pdf.

18. H. Ludwig. "Web services QoS: External SLAs and Internal Policies Or How do we deliver what we promise?," In Proceedings of the First Web Services QualityWorkshop at WISE, 2003. – P. 115-120.

19. Kot T., Reverchuk A., Globa L., Schill A. A novel approach to increase efficiency of OSS/BSS workflow planning and design. – Springer: Lecture Notes in Business Information Processing. – 2012. – Vol. 117. – P. 142-152.

20. Tetiana M. Kot, Andrey V. Reverchuk, Larysa S. Globa, Alexander Schill. Complex approach to service development // ISSN 2219-9454, Telecommunication Sciences, 2012, Volume 3, Number 2. – P.19-29.

21. J. Klingemann and K. Wasch J., Aberer. "Deriving service models in cross-organizational workflows," in Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, March 1999. – P. 100-107.

22. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng. "Quality driven web services composition," in Proceedings of the 12th International conference on World Wide Web, May 2003. – P. 411-421.

23. R. Aggarwal, K. Verma, J. Miller, and W. Milnor. "Constraint driven web service composition in METEOR-S," in Proceedings of the 2004 IEEE International Conference on Services Computing, September 2004. – P. 23-30.

24. M. Lin, J. Xie, H. Guo, H. Wang "Solving Qos-driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction", in Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005. – P. 9-14.

Received in final form October 9, 2013