A Novel SLA-Aware Approach for Web Service Composition

Dmytro Pukhkaiev[#1], Tetiana Kot[#2], Larysa Globa[#3], Alexander Schill [*4]

[#] *Information Telecommunication Department, National Technical University of Ukraine "Kyiv Polytechnic Institute"*
37, Peremoga Ave., 03056 Kyiv, Ukraine

[1] dpukhkaiev@its.kpi.ua

[2] tkot@its.kpi.ua

[3] lgloba@its.kpi.ua

[*] *Faculty of Computer Science, Dresden University of Technology*
46, Nöthnitzer Str., 01187 Dresden, Germany

[4] alexander.schill@tu-dresden.de

Abstract—**Changeable requirements for modern services, provided in global environment, and their constant re-engineering demand dynamic composition of services which is able to estimate changes of various parameters, both functional and non-functional ones. Web Services Agreement is a convenient way to contain QoS parameters, but state-of-the-art SLA-aware methods are not able to support all classes of non-functional parameters and provide run-time support and dynamic reconfiguration at the same time. This paper provides a novel SLA-aware dynamic Web Services Composition Approach that allows performing such composition.**

Keywords: Web Service Composition, Web Services Agreement, QoS requirements

I. Introduction

Nowadays the way how software applications are designed, architected, delivered and consumed has significantly changed. Service-Oriented Computing (SOC) [1] is the computing paradigm that utilizes services which are considered as 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. SOC can be considered as framework for service publishing, discovery, binding and composition. SOC 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 the messaging up to the 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 create an implementation of 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.

One of WSC tasks requiring realization is necessity to consider QoS parameters both on Process enactment and Diagnosis stages. The most convenient way to contain and monitor their values is usage of Service Level Agreement (SLA) [8].

Besides, another WSC issue regarding QoS parameters is that despite the existence of various dynamic composition approaches there is no approach which combines 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. The proposal for SLA-aware WSC approach is presented in Section 4. Section 5 concludes the work with a summary and outlook on future work.

II. Background

A. Workflow design and enactment

Graphical standards provide formalization of 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].

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 interaction of the process 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.

B. Web Services Composition

WSC is a method to connect different web services that are 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 which are effectively combined and reused in order 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 the 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, it is replaced with another one.

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 the predicates and requisites match in each step of the workflow lifecycle. Although the functional Web service composition has been widely researched in literature, the assessment criteria related to non-functional attributes in particular, significantly increases the number of requirements for composition. QoS parameters can be used for ranking of 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, this 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].

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.

C. WS-Agreement

In order to be successful, WS providers have to guarantee declared capabilities related to the 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 the functional and non-functional parameters of a service oriented application. It consists of two components that are the agreement Context and the agreement Terms [17].

D. 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 the 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 (see Section IV-B).

III. QoS-Based Evaluation of WSC Approaches

The following section provides an evaluation of existing WSC approaches in terms of QoS requirements.

A. WSC approaches

An overview of several approaches for modeling web service quality composition is presented in this subsection.

Continuous-time Markov chains solution. The approach to estimate workflow execution time and cost, based on continuous-time Markov chains, is proposed in [20]. Simple

QoS model is provided via assigning an execution cost to each activity, whose start and end is signalled by a service.

Quality Vector solution. Description of elementary service quality as a quality vector is proposed in [21], a similar solution can be found in [22]. The authors propose to calculate quality criteria, which are components of the vector, for composite services by using special aggregation functions [22].

Global planning approach [23] is used to select optimal component services of a composite service. Additionally, service selection can be formulated as an optimization problem which is solved using linear programming [22].

Agent-oriented solution. There is a proposal in [24] to use the agent-oriented methodology Tropos to model a web services properties. To provide modelling of a quality composition, the interacting services are represented as a multi-agent system.

Ontology-based solution. An ontology-based framework for dynamic web services selection is presented in [15]. It is based on two-layered model: first layer is service ontology, second layer is QoS ontology. Services to QoS correspondence is defined in the services ontology.

Ontology-based solution relies on ontologies only as knowledge base and for decision making operations. In fact it utilizes agent-based architecture. Nevertheless, it is important to separate this approach from other agent-oriented approaches because ontologies can focus not only on objective, but also subjective QoS parameters. But it still lacks runtime support and QoS assignments. It is also important that this method can be used as WSC approach. The only difference from dynamic selection approach based on QoS is that services are selected not for user directly but for an application, creating a sequence of chosen services.

B. Methodology for evaluation

Presented WSC approaches are compared with respect to QoS characteristics, which are objective and subjective QoS, run-time support, QoS assignment to composite service and quality requirement [25], [26].

Run-time support. WSC approaches should provide run-time support in order to be able to monitor QoS parameters and respond to their changes.

QoS assignment to composite service. When quality constraints and preferences are assigned to composite services it is easy to reuse the last ones, i.e., use it in another quality driven composition.

QoS WS Requirements include performance, reliability, interoperability, exception handling, robustness, scalability, capacity, accessibility, accuracy, integrity, availability, security and network-related QoS requirements.

C. Evaluation without SLA

The evaluation of WSC approaches presented in Section III-A regarding requirements provided in Section III-B is summarized in Table I.

TABLE I
QOS-BASED COMPARISON OF WSC APPROACHES

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

Markov chains- and agents-based approaches provide run-time support while quality vector and ontology-based approaches does not provide such support.

Table I shows that quality vector-based and agent-oriented solutions allow to assign QoS to composite service.

Not all of the QoS WS requirements are met in approaches presented above at the same time. Specifically, execution price and time are estimated by Markov chains solution. While Quality Vector solution considers execution price, execution duration, reliability, availability and reputation, and similar to its solution focuses on four quality dimensions, i.e., execution cost, execution time, reliability, and availability. Ontology- and agent-based solutions theoretically deal with most of QoS requirements.

Thus, current state-of-the-art QoS WSC models and solutions are far from required one. None of the presented approaches is able to meet all QoS characteristics. The most crucial characteristic is the ability to support all types of QoS parameters. Markov chains and Quality Vector solutions are not applicable in this regard while Ontology-and agent-based solutions meet some QoS characteristics although there are still some types of QoS parameters i.e. non-measurable parameters [27] that should be taken into account.

D. Evaluation with SLA

SLA enables a convenient way to perform WSC and monitoring of composed service. Applying SLA to methods mentioned in Section III-A provide much more acceptable results than presented in Table I. For instance, run-time support is now available for all existing methods, using SLA. Table II summarizes the results.

TABLE II
QOS-BASED SLA-AWARE COMPARISON OF WSC APPROACHES

Requirements	**WSC Approaches**			
	SLA-Aware Markov Chains	**SLA-Aware Quality Vector**	**SLA-Aware Agent-oriented**	**SLA-Aware Ontology-based**
Objective QoS	+	+	+	+

Subjective QoS	-	-	-	+
Run-time support	+	+	+	+
QoS assignment	+	+	+	+
QoS WS Requirements	Low	Average	High	High

Table II shows that applying SLA to WSC methods provides Run-time support and QoS assignment (assuming that WS-Agreement will be generated for composite service using WS-Agreements of single services) for all approaches.

Thus, Ontology-based WSC approach combined with SLA-awareness should be able to perform the most reliable WSC of all the approaches presented above.

IV. Proposal for SLA-Aware WSC Approach

A. General description

The workflow on the stage of enactment lacks an 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 stage of enactment is described by a model presented in the Section IV-B.

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 service consumer.

When consumer application, built with WSAF, requests to use a service, agent is called for communication with service. An agent is created for the 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 the values of objective QoS-attributes (reliability, availability and execution time, for instance) and get user feedback for subjective attributes (which is an indicator of appropriateness for requested QoS parameters in comparison to 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:

- upon initialization, WSAF sets up all configured agencies;
- 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;
- the consumer application creates a local proxy object for the service agent; the consumer invokes the proxy with its WS-Agreement;
- 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;
- the agent selects a service implementation based on agency data, and then dynamically creates a proxy object for each selected service;
- 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. After this it applies WS-Agreements on the available quality data, providing service implementations ranking.

Comparing to general WSAF, appliance of SLA provides:

- substitution of consumer's and provider's policies by WS-Agreements, providing the composition with the 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.

Classic Ontology-based approach and the approach which emerged from combination of Ontology-based approach and SLA-awareness are presented on Fig.2.

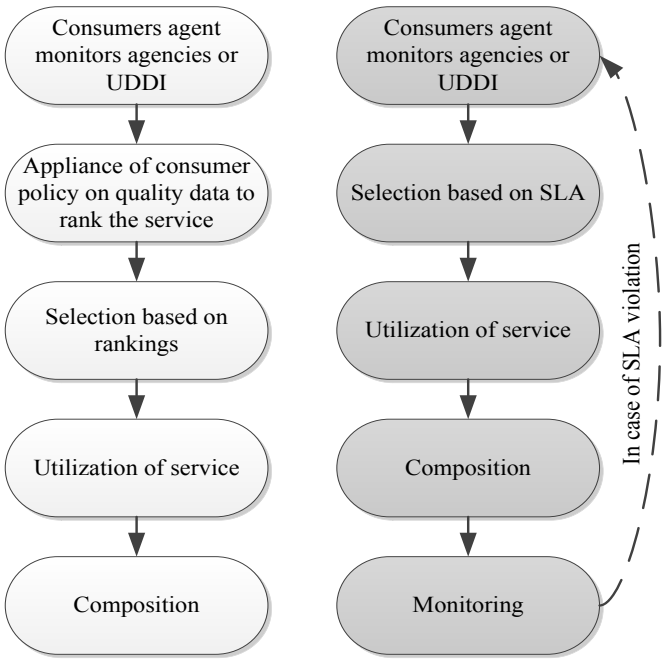

Fig. 1 a) Classic Ontology-based approach, b) SLA-aware WSC approach

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 classic Ontology-based approach is enabling of run-time monitoring of the composite service.

B. 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 of can be represented as:

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

where E_{Id} – set of workflow's identification objects, I - set of informational input objects, Q – set of quality parameters which were specified on the design stage.

E_{Id} is a set of four parameters $\{E_{Id\ j,\ j=1,4}\}$ and consists of: $E_1 = N_{EP}$ – workflow name; $E_2 = O$ – set of tasks; $E_3 = Pl$ – set of partner links with tasks executors; $E_4 = Ex$ – workflow executor/executors.

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

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

where Ex_{ij} – j's performer of i's task, N_{EPij} – name of j's performer of i's task, O_{ij} – i's task which is executed by j's performer.

Informational objects have the same representation as on the stage of 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 = performance, Q_2 = reliability, Q_3 = robustness, Q_4 = accessibility, Q_5 = availability, Q_6 = cost, Q_7 = additional QoS parameters (scalability, capacity, accuracy).

Utilization of quality parameters allows to:

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

In order to create workflow implementation, an 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 the 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 set of informational input objects, O - set of informational output objects;

F is indicator for the 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 integral indicator of web service quality compliance. It can be represented mathematically as:

$$Nf = (P, Rl, Rb, Ac, Av, C, A) \quad (3)$$

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 the 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 lesser influence on composite web service than previously mentioned ones, however they should be taken into account if possible, and include:

- scalability - represents 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 error rate generated by the web service.

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

Unlike other approaches, the last one covers wider range of QoS parameters. In order to compose a best-choice web service, satisfying required parameters, the last ones 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 III 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 IIII
QOS COMPUTATIONS AND RANKINGS FOR COMPOSITE SERVICES

QoS Parameter	R	WS Composition Patterns			
		Sequence	Parallel	Switch	Loop
Performance	2	$\sum_{i=1}^{m} P_i$	$\max(P_i)$	$\sum_{i=1}^{m} p_j P_i$	R_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	$\prod_{i=1}^{m} Ac_i$	$\prod_{i=1}^{m} Ac_i$	$\prod_{i=1}^{m} p_j Ac_i$	Ac_i^k

C. Practical implementation

Main functions of the software environment for SLA-aware WSC and its architecture are presented in this section.

Architecture of software environment for SLA-aware WSC on the stage of workflow execution is presented on Fig.2.

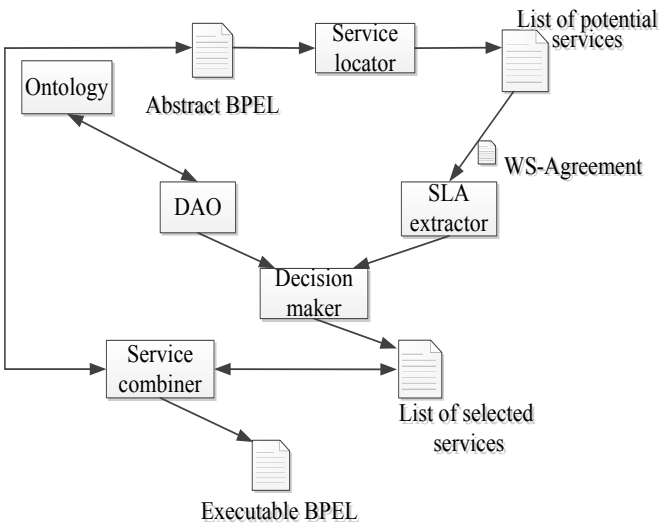

Fig. 2 Architecture of software environment for SLA-aware WSC

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

Service locator is a module that extracts the information about functional parameters from abstract BPEL-file. Is also searches the appropriate services in UDDI or service brokers (considering functional parameters). After that it puts all found services into the list of potential services.

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

Decision maker utilizes data received from SLA extractor and calculates rankings for services in accordance to rules, located in Ontology module. The, this module makes a decision which service suits the composite service in the best way.

Services that were not chosen for the composition are placed into the list of 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.

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

- compares current QoS parameters with the required values and with values of QoS parameters in the list of QoS-aware services;
- monitors the value of indicator 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 previous one, the previous WS is replaced by “better” one, thus WS re-composition is done.

V. CONCLUSION AND FUTURE WORK

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

Analysis of current state-of-the-art web service composition approaches has been made and it has showed that none of them is able to meet all required QoS chacteristics. A novel SLA-aware approach for Web Service Composition which satisfies required QoS chacteristics is presented in this paper. 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.

Future work will focus on the 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 on real world scenarios. Quantitative results on QoS and service development and re-engineering time will be measure to prove proposed approach efficiency.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications*. Springer, 2004.

[2] M. Papazoglou, V. D'Andrea, D. Plexousakis, P. Grefen, J. Yang, M. Mecella, and P. Plebani. *SOC: Service-Oriented Computing Manifesto*, 2003.

[3] M. Papazoglou and D. Georgakopoulos. "Service-Oriented Computing," *Communications of the ACM*, vol. 46, pp. 25-28, Oct. 2003.

[4] van der Aalst, W.M.P., ter Hofstede, A.H.M. and Weske, M., "Business process management: a survey," in *Proc. BPM'03,* 2003, p.1-12.

[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, pp.164-166, Apr. 2012.

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

[9] OMG (2011) Business Process Model and Notation (BPMN) [Online]. 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 modelling APCCM '06, vol. 53, pp. 95-104., 2006

[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, pp. 370-390, Oct. 2009.

[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, pp. 84-93, Sep./Oct. 2004.

[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 Services 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] 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.

[21] 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.

[22] 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.

[23] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J.t Kalagnanam, H. Chang "QoS-aware Middleware for Web Services Composition," IEEE Transactions on Software Engineering, vol. 30, pp. 311-327, 2004.

[24] M. Aiello and P. Giorgini. "Applying the Tropos methodology for analysing web services requirements and reasoning about Qualities of Services. CEPIS Upgrade," The European journal of the informatics professional, vol.5, pp. 20-26, 2004.

[25] K.-C. Lee et al. (2003) QoS for web services: Requirements and possible approaches, November 2003. W3C Working Group Note, [Online]. Available: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[26] A. Mani and A. Nagarajan. (2002) Understanding quality of service for web services [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-quality/index.html

[27] 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.