УДК 681.518

V.A. SHEKHOVTSOV

National Technical University "Kharkiv Polytechnical Institute", Ukraine

TOWARDS EVALUATING SERVICE-ORIENTED ARCHITECTURES BASED ON STAKEHOLDER ASSESSMENTS OF SIMULATED SERVICE QUALITIES

The paper introduces an approach for development of service-oriented software systems aimed at evaluating service-oriented architectures based on assessments of simulated service qualities by business stakeholders. This approach is a part of ISAREAD-S framework aimed at involving business stakeholders in a software process in a form of assessing the perceived quality of the service-oriented system (exemplified by service performance and reliability) in its usage context. We investigate evaluation of both particular architecture as a whole (snapshot evaluation) and particular architectural decisions (incremental evaluation). We propose three-stage evaluation technique of converting the description of either architecture or the particular decision into the set of values for the factors influencing simulation of the prospective system, obtaining stakeholder assessments of simulated qualities, and deriving the evaluation marks from these assessments. The solution is implemented as a high-level procedure (architecture evaluation policy) based on low-level procedures (mechanisms) collecting stakeholder opinions on perceived service quality.

Key words: *quality of service, service performance, service reliability, software architecture, architectural decision, architecture evaluation, business stakeholders.*

1. Introduction

The value of software architecture evaluation as a process of assessing either the way of organizing the software system from its components or the particular decision related to such organization (architectural decision) has been widely shown in the current literature [1]. Alongside evaluation performed by IT specialists, an important research challenge is to establish an evaluation process performed by business stakeholders (not necessary having any IT background). It should allow the stakeholders and the development team (first of all, software architects) to reach common ground in what they expect from the system. This is especially important for service-oriented systems as assessing service-oriented architectures (SOA) requires knowledge of their possible uses which is difficult to obtain without the involvement of their prospective users.

Our research is put into context of a broader problem of stakeholder involvement into the development of service-oriented systems. To address this problem, we proposed ISAREAD-S framework (Interactive Simulation-Aided Requirements Engineering and Architectural Design for Services) [2, 3]. It is aimed at investigating ways of supporting stakeholder involvement in the software process by allowing business stakeholders to assess the perceived qualities of the prospective service-oriented system (exemplified by performance and reliability) in its usage context.

To implement such support we plan to elaborate a set of simulation-based methods aimed at making QoS (quality of service) assessment mechanisms (according to mechanism-policy separation principle we use the term *mechanism* to refer to low-level procedures which are not aware of their possible uses) accessible to the business stakeholders and using their assessments as a driving force for software process activities related to requirements engineering and architectural design.

This paper is devoted to establishing architecture evaluation policies (we use the term *policy* to refer to high-level procedures based on specific mechanisms) to be integrated into this framework. Their purpose is to perform an evaluation of either architecture as a whole (the consequences of applying the particular architectural style) or the particular architectural decision as an integrated assessment of its simulated qualities.

The structure of the paper is as follows. Section 2 describes the state of the art and formulates the problem statement, Section 3 shows the principles of the existing procedures (mechanisms) for organizing the interaction with stakeholders; these mechanisms form the foundation for the architecture evaluation solutions proposed in the paper, Section 4 outlines the proposed approach introducing higher-level procedure (policy) for evaluating service-oriented architectures based on stakeholder assessments obtained as a result of applying the mechanisms, Section 5 makes conclusions and describes the directions for future research.

2. State of the art and problem statement

Software architecture evaluation has received considerable attention in the software engineering literature (a survey of the available techniques is in [4, 5]). Such techniques could be classified according to [1] into scenario-based techniques, prototype-based techniques, and simulation-based techniques. Some of these techniques are listed in Table 1.

Table 1

State-of-the-art architecture evaluation techniques

Category	Examples	Desription	Problems
Scenario-based evaluation	Software Engineering Institute-based methods (SAAM [6], ATAM [7]), SALUTA [8], ASAAM [9], DoSAM [10]	Rely on scenarios of the system behavior which reflect the chosen variant of the system architecture or the particular architectural decision. Users participate in these scenarios and assess their experience.	Manual scenarios cannot be made realistic; Cannot be used to evaluate quantified qualities such as reliability.
Prototype-based evaluation	Architectural prototyping [11], architecture performance evaluation [12], general prototype-based techniques [1]	Use either horizontal or vertical prototypes as a basis for the architecture evaluation. Users interact with prototype reflecting the chosen architecture and assess their experience.	Using prototype to obtain realistic performance or reliability of the system requires access to the target system which can be not feasible.
Model-based evaluation	Analytical models: reliability [13, 14], performance [15, 16], Simulation models: reliability [17, 18], performance [19, 20]	Simulating or analytically modeling the chosen software architecture (or the quality characteristic values reflecting this architecture) and use the results to obtain assessments.	Simulations and mathematical models are usually non-interactive; the results are collected after the run. Could be cost-ineffective.

The common problem of applying the above methods to the problem of evaluating software architectures is related to the fact that they, as a rule, are not intended to be used by business stakeholders, targeting IT specialists instead. As a result, the understanding of the desired system architecture resulted from evaluation process becomes biased towards the view of the IT people: a problem known as "the inmates are running the asylum".

2.1. Problem statement

After analyzing the state of the art we can formulate both general and specific research questions which determine the problem statement.

The general question is [21]: *How to involve business stakeholders into the development process for service-oriented software systems as a means of control for the performance and reliability of the produced artefacts?* We address this question by introducing IS-AREAD-S framework [2] offering mechanisms for interactive assessment of simulated service performance and reliability; we present an outline of this framework's assessment mechanisms in the next section.

Prior to introducing an architecture evaluation policy based on the proposed mechanisms, we address the research problem of making simulations reflect the chosen software architecture or partial architectural decision; it leads to the research question: *How to make service quality simulations depend on the software architecture?* To answer, we need to investigate how the architecture affects simulation parameters by addressing the question: *What is the dependency between the software architecture and the factors influencing service qualities?* An example of such dependency could be the situation when the chosen architectural decision (e.g. introducing a caching solution) makes it possible to increase performance by reducing the network load.

The knowledge obtained so far allows us to follow the mechanism-policy separation principle by elaborating higher-level policies based on the proposed assessment mechanisms. As a result, we can formulate the specific research question related to the topic of this paper: *How to use the mechanism of interactive assessment of simulated service qualities to perform an evaluation of the software architectures or particular architectural decisions?* To answer this question, it is necessary to establish the set of necessary procedures which define *architecture evaluation policy*. It should rely on both assessment mechanisms and the techniques allowing simulations depend on the artefacts of the development process.

The main benefit of establishing the evaluation policy is that business stakeholders become directly involved in diverse quality-related software engineering activities alongside software development lifecycle. We finally plan to establish a lifecycle simulation support by asking the stakeholders to make assessments of simulated qualities reflecting the current state of the SUD as the development progresses with a purpose to make these assessments drive the development; this leads to the last research question: *How to organize software development lifecycle to benefit from the stakeholder assessment of simulated service performance and reliability?*

3. Outline of the assessment mechanisms

In [2] we described the proposed approach to establish service-level and process-level assessment mechanisms which allow business stakeholders to experience simulated qualities of the prospective SUD in its usage contexts and formulate their opinions on these qualities. In this section, following [21], we outline this approach to the degree necessary to understand the proposed architecture evaluation solution.

3.1. Service-level mechanisms

Service-level IAS mechanisms organize interactive assessment of simulated service qualities at the level of the particular service. According to the model-driven methodology [22, 23] we define two mechanisms of this kind: IASC (model composition mechanism) and IASE (model execution mechanism). IASC inputs include the set of qualities of interest (to be simulated and assessed) and the set of factors influencing the simulation (simulation parameters [2, 3]). To get the integrated quality simulation model, we compose simulation modules corresponding to the qualities of interest and the necessary parameters together with the base simulation structure. Also, we integrate into this model the set of user interaction models depending on the qualities of interest, types of the expected stakeholders, and the project categories. The resulting service-level simulation and assessment model IASM becomes the IASC output transferred to IASE for standalone execution.

The input for every IASE run is the set of parameter values corresponding to the parameters used to build IASM. As a result of the run, the set of simulated values for the qualities of interest is obtained and presented to the business stakeholder for assessment via interaction processes described by user interaction models integrated into IASM. The IASE outputs are this set of simulated qualities and the set of assessment results.

3.2. Process-level mechanisms

Process-level IAP mechanisms organize interactive assessment of simulated service qualities in context of usage processes at the level of the particular process. There are two such mechanisms: IAPC (model composition mechanism) and IAPE (model execution mechanism). They rely on service-level IAS mechanisms dealing with individual services.

IAPC forms the simulation model of the usage process making it ready for interactive assessment of service qualities. It combines the control flow model (CFM) for the usage process (conforming to the network BPM notation such as BPMN or Petri Nets) with the role model for the usage process. The role model includes the set of roles defined for process participants (clerk, manager etc), the sets of interaction activities for different roles (they make participants affect the state of the process simulation), the sets of assessment activities for different roles (they correspond to the services of interest to be simulated and assessed by stakeholders) and the sets of qualities of interest and necessary parameters defined for every service of interest.

While composing the integrated IAPM model for the process, IASC creates the IASM model for every service of interest; these models later become integrated into IAPM. For every interaction activity, a mechanism for constructing the interaction model is invoked and the resulting models are also integrated into IAPM. The integrated IAPM model will contain the simulation logic defined by CFM for the usage process, the simulation submodels of different IASM models (for the services of interest), the assessment logic defined by interaction submodels of these IASM models, and the interaction logic defined for all interaction activities.

Every IAPE run is supposed to be driven by the stakeholder belonging to the particular role. During the run, the basic simulation flow is managed by the model derived from the CFM of the usage process; when the logic of the run requires invoking an activity representing the service of interest, the simulation of its qualities and the assessment interaction logic are handled by IASE invoked for its IASM. IASE inputs are parameter values for all the slots of this service; when this logic requires interacting with the simulation, the logic of this interaction is handled by the corresponding interaction mechanism. The outputs for IAPE run include the set of all simulated quality values for all the services of interest and the set of corresponding assessment results.

3.3. Iterative interactive assessment

Process-level mechanisms allow performing interactive assessments for the particular usage process. To guarantee obtaining all the necessary assessments for the particular SUD, we proposed to establish specific "iterator-like" Iterative Interactive Assessment (IIA) mechanism which externally controls the simulation and allows iterating over usage processes, roles, and runs.

4. Outline of the proposed approach

Outlined low-level assessment mechanisms serve as building blocks for high-level procedures (policies). Most of these policies are supposed to be used at early stages of the software development lifecycle such as requirements elicitation [24] or validation [25]. The proposed architecture evaluation policy is aimed at the architectural design stage.

4.1. Architecture adapter

Prior to defining an architecture evaluation policy we introduce the notion of assessment adapter mechanism. Such adapters convert external information into the inputs for an assessment mechanism.

For the purpose of this paper, we focus at one of the possible adapters of this kind: *the architecture adapter*.

Prior to establishing this adapter we investigate the dependency between the description of the software architecture or the particular architectural decision and the factors influencing service qualities (examples are e.g. [19, 26, 27]). This adapter is based on the knowledge of this dependency; it converts the description of the software architecture into the inputs for the assessment mechanisms: the set of services, the corresponding QAPM-S and the set of parameter values (assuming it is possible to establish the rules connecting particular architectural decisions to simulation parameters). It should allow experimenting with an influence of different architectural design decisions on the quality of the prospective SUD.

4.2. Architecture evaluation policy

In this section, we propose software architecture evaluation policy aimed at calculating integrated assessments for the software architecture or particular architectural decisions consisting of the three steps (Fig. 1):

1. adapting the architecture (*snapshot evaluation*) or the particular decision (*incremental evaluation*);
2. assessing the simulated qualities reflecting the adapted architecture or the particular decision;
3. calculating the integrated assessment value(s).

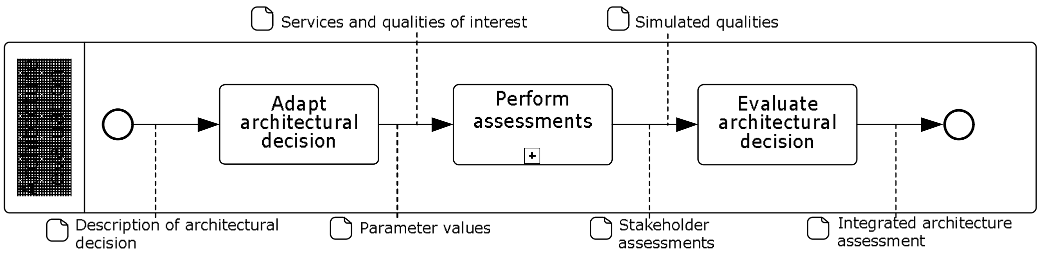
Services and qualities of interest
Simulated qualities
Adapt architectural decision
Perform assessments
Evaluate architectural decision
Description of architectural decision
Parameter values
Stakeholder assessments
Integrated architecture assessment

Fig.1. Architecture evaluation policy (incremental evaluation)

The first step is supported with an architecture adapter. On the second step, we propose to execute IIA mechanism to obtain the set of simulated qualities $qsim_{sq}^{mrt}$ and the set of stakeholder assessments $qest_{sq}^{mrt}$, where

$s \in S$ - a particular service of interest belonging to the set *S* of all services comprising the SUD;

$q \in Q^s$ - a quality of interest ($Q^s \subseteq Q$ is the set of all qualities of interest for the particular service s, Q is the set of all defined qualities of interest);

$m \in M^S$ - a usage process ($M^S \subseteq M$ is the set of all usage processes defined for SUD S, M is a set of all defined usage processes);

$r \in R^m$ - a role ($R^m \subseteq R$ is a set of all available roles defined for the usage process m, R is the set of all defined roles);

$t \in T^r$ - a simulation run driven by some stakeholder ($T^r \subseteq T$ is a set of all available simulation runs for the role r, T is the set of all available stakeholder sessions).

Based on these values, on the third step we can obtain integrated assessments for the software architecture or the particular architectural decisions. We define two categories of such assessments:

1. *Integrated (architecture-wide) assessment.* It corresponds to obtaining a scalar value characterizing the architecture as a whole or the particular architectural design decision in integral fashion. We denote such assessment for the architectural decision d_i as $qass^{\Sigma}(d_i)$.

2. *Quality-related assessments.* They are calculated for particular qualities (we can have this way e.g. integrated assessments for the performance or reliability). We denote these assessments for the architectural decision d_i as $\left\{qass_q^{\Sigma}(d_i), q \in Q\right\}$

It is also possible to group assessments by other dimensions such as the role or stakeholder type.

To obtain such assessment the simplest approach is to use weighting technique. We illustrate this approach with the simplified example where assessments are only indexed by stakeholder run so we have $qsim_{sq}^{t}$ and $qest_{sq}^{t}$. In this case the weights $w^t, t \in T^S$, $\sum_{t \in T^S} w^t = 1$ are assigned to stakeholder runs reflecting their relative importance for the project. We can also assign the weights to the particular services $s \in S$ reflecting their influence on the integral SUD quality; in this paper we assume that all the services are of equal importance for the quality of the prospective SUD.

In this case quality-related assessments for the architectural decision d_i could be calculated as follows:

$$qass_q^{\Sigma}(d_i) = \sum_{s \in S} \sum_{t \in T^S} w^t qest_{sq}^t(d_i),\ q \in Q$$

The weights w^t and other evaluation-related information is planned to be encapsulated in an instance of the *evaluation influence model* EIM. Such model is supposed to be an input for the evaluation policy; its goal is to facilitate reuse of the evaluation approaches.

We also plan to enhance the proposed relative weights approach with more advanced assessment calculation methods originated from different evaluation techniques [1, 7, 28].

4.3. Lifecycle support for architectural design

To establish lifecycle simulation support we plan to follow the idea of model calibration [29] when the system model is supposed to evolve as the development progresses. For this process, in this paper we limit ourselves with an architectural design phase of the software lifecycle. The schema of the quality-driven development process for this stage is given on Fig. 2.

To achieve quality awareness for the architectural design process, we plan to make new architectural design decisions reflected in the simulation model (through the architecture adapter) and ask the stakeholders to make assessments of the impact of these decisions during the development (we call this process *incremental architectural assessment*). If some design decision leads to the drop in the assessment marks this can be the indication of the problems with this decision.

This way, it will be possible to check the opinions of stakeholders before real implementation of the decision, thereby decreasing the risk of accepting the decisions that could cause problems during the development. On Fig. 2, it can be seen how every decision is reflected in the simulation model and evaluated. Only after that the final *meta-decision* related to the feasibility of its implementation is supposed to be made.

This technique could also be applied to other steps of the software process. In general, we plan to establish the unified *Quality-Driven Software Process* (QUP) [30] which will be supposed to take into account opinions of business stakeholders on the quality of the prospective SUD as its development progresses over time.

6. Conclusions and future research

In this paper, we defined the principles of new high-level procedures (policies) for evaluating software architecture or particular architectural decisions based on stakeholder assessments of simulated software qualities. Their advantage as compared to known evaluation methods is that stakeholders are able to experience the prospective system before expressing the opinions on the quality of its architecture as a whole or the chosen architectural decision.

In future, we plan to elaborate the models underlying the architecture adapter, investigate the applicability of different methods for solving the problem of obtaining evaluation marks from stakeholder assessments, completely implement the evaluation policy, and establish the validation studies for the proposed technique.

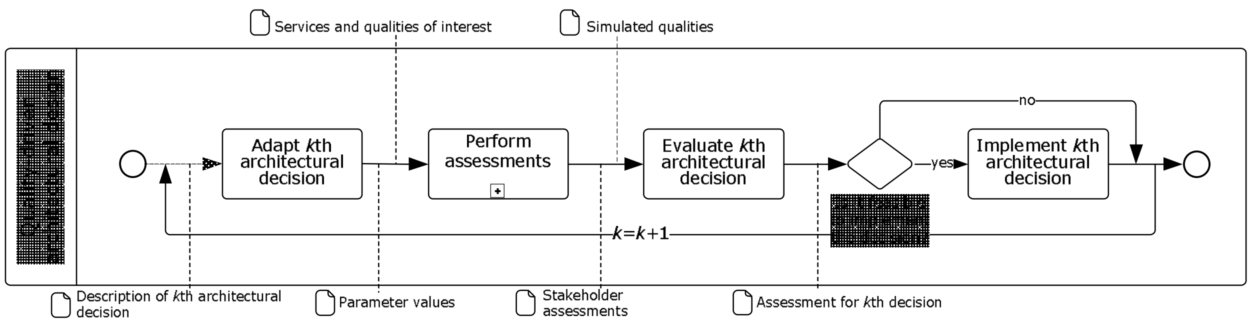
Services and qualities of interest
Simulated qualities
Adapt kth architectural decision
Perform assessments
Evaluate kth architectural decision
no
yes
Implement kth architectural decision
k=k+1
Description of kth architectural decision
Parameter values
Stakeholder assessments
Assessment for kth decision

Fig.2. A proposal for a quality-driven architectural design

In addition, we see this particular solution as a particular case for more generic framework. In this framework, architectural decisions could be seen as a particular instantiation of more generic concept of *changing the state of the project*. This change could be any event that could affect this state: adding resources, choosing particular implementation strategy (as opposed to design strategy) etc; an example of a connection between the decisions considered during the implementation stage (selecting code checking rules for static code analysis) and the design decisions is shown in [31]. As a result, we plan to elaborate a generalized universal *state change adapter* which could be instantiated as various adapters including the one described in this paper.

References

1. *Bosch J. Design and Use of Software Architectures / J. Bosch. – Reading: Addison-Wesley. – 2000. – 370 p.*

2. *Shekhovtsov V.A. Interactive assessment of simulated service qualities by business stakeholders: principles and research issues / V.A. Shekhovtsov // Проблеми програмування. – 2010. – № 2-3. – С. 288-298.*

3. *Shekhovtsov V.A. Constructing POSE: a Tool for Eliciting Quality Requirements / V.A. Shekhovtsov, R. Kaschek, S. Zlatkin // Proc. ISTA 2007. – LNI, Vol. P-107. – Bonn: GI. – 2007. – P. 187-199.*

4. *Babar M. Comparison of Scenario-Based Software Architecture Evaluation Methods / M.A. Babar, I. Gorton // APSEC'04. – IEEE. – 2004. – P. 600-607.*

5. *Roy B. Methods for Evaluating Software Architecture: A Survey. Technical Report No. 2008-545 / B. Roy, T.C.N. Graham. – Queen's University at Kingston. – 2008. – 82 p.*

6. *Kazman R. SAAM: A Method for Analyzing the Properties of Software Architectures. / R. Kazman, L. Bass, G. Abowd, M. Webb // Proc. ICSE'94. – ACM. – 1994. – P. 81-90.*

7. *Kazman R. Experience with Performing Architecture Tradeoff Analysis / R. Kazman, M. Barbacci, M. Klein, S.J. Carriere // Proc. ICSE'99. – ACM. – 1999. – P. 54-63.*

8. *Folmer E. Software Architecture Analysis of Usability / E. Folmer, J. Gurp, J. Bosch // Proc. EHCI-DSVIS'04. – LNCS, Vol. 3425. – Berlin-Heidelberg: Springer. – 2004. – P. 321-339.*

9. *Tekinerdogan B. ASAAM: aspectual software architecture analysis method / B. Tekinerdogan // Proc WICSA'04. – 2004. – P. 5-14.*

10. *Bergner K. DoSAM - Domain-Specific Software Architecture Comparison Model / K. Bergner, A. Rausch, M. Sihling, T. Ternit // Proc. QoSA'05. – LNCS, Vol. 3712. – Berlin-Heidelberg: Springer. – 2005. – P. 4-20.*

11. *Bardram J.E. Architectural Prototyping: An Approach for Grounding Architectural Design and Learning / J.E. Bardram, H.B. Christensen, K.M. Hansen // Proc. WICSA'04. – 2004. – P. 15-24.*

12. *Martensson F. An Approach for Performance Evaluation of Software Architectures using Prototyping / F. Martensson, H. Grahn, M. Mattsson // Proc. SEA'03. – 2003. – P. 605-612.*

13. *Krishnamurthy S. On the estimation of reliability of a software system using reliabilities of its components / S. Krishnamurthy, A.P. Mathur // Proc. SRE'97. – 1997. – P. 146-155.*

14. *Gokhale S. S. Architecture-Based Software Reliability Analysis: Overview and Limitations / S.S. Gokhale // IEEE Transactions on Dependable and Secure Computing. – 2007. – Vol. 4. – P. 32-40.*

15. *Williams L.G. PASA: A Method for the Performance Assessment of Software Architecture / L.G. Williams, C.U. Smith // Proc. 3rd Workshop on Software Performance. – 2002. – P. 179-189.*

16. *Смит К.У. Эффективные решения: практическое руководство по созданию гибкого и масштабируемого программного обеспечения / К.У. Смит, Л. Уильямс. – К.: Диалектика, Вильямс. – 2003. – 448 с.*

17. *Gokhale S. S. Reliability Simulation of Component-Based Software Systems / S.S. Gokhale, M.R. Lyu, K.S. Trivedi // Proc. ISSRE'98. – 1998. – P. 192-201.*

18. *Cortellessa V. Reliability Modeling and Analysis of Service-Oriented Architectures / V. Cortellessa, V. Grassi // L. Baresi, E.D. Nitto (eds.): Test and Analysis of Web Services. – Berlin-Heidelberg: Springer. – 2007. – P. 339-362.*

19. *Cortellessa V. MOSES: MOdeling Software and platform architEcture in UML 2 for Simulation-based performance analysis / V. Cortellessa, P. Pierini, R. Spalazzese, A. Vianale // QoSA 2008. – LNCS, Vol. 5281. – Berlin-Heidelberg: Springer. – 2008. – P. 86-102.*

20. *Hennig A. Performance Prototyping - Generating and Simulating a Distributed IT-System from UML Models / A. Hennig, A. Hentschel, J. Tyack // Proc. ESM'2003. – IEEE. – 2003. – P. 502-508.*

21. *Shekhovtsov V.A. Towards negotiating QoS requirements originated from stakeholder assessments of simulated service qualities / V.A. Shekhovtsov // Радіоелектронні і комп'ютерні системи. – 2010. – № 1(42). – С. 108-114.*

22. *Mellor S. MDA Distilled: Principles of Model-Driven Architecture / S. Mellor, K. Scott, A. Uhl, D. Weise. – Reading: Addison-Wesley. – 2004. – 176 p.*

23. *Pastor O. Model-Driven Architecture in Practice / O. Pastor, J. Molina. – Berlin-Heidelberg: Springer. – 2007. – 302 p.*

24. *Shekhovtsov V.A. Towards eliciting QoS requirements from stakeholder assessments of simulated service qualities / V.A. Shekhovtsov // N.D. Pankratova (ed.) System Analysis and Information Technologies. – Kyiv: NTUU "KPI". – 2010. – P. 398.*

25. *Shekhovtsov V.A. Towards validating QoS requirements using stakeholder assessments of simulated service qualities / V.A. Shekhovtsov // Восточно-Европейский журнал передовых технологий. – 2010. – № 2/9 (44). – С. 4-8.*

26. *Grassi V. Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach / V. Grassi, R. Mirandola, A. Sabetta // The Journal of Systems and Software. – 2007. – Vol. 80. – P. 528–558.*

27. *Marzolla M. UML-PSI: the UML Performance SImulator / M. Marzolla, S. Balsamo // Proc. QEST'04. – IEEE. – 2004. – P. 340-341.*

28. *Babar M.A. Eliciting Better Quality Architecture Evaluation Scenarios: A Controlled Experiment on Top-Down vs. Bottom-Up / M.A. Babar, S. Biffl // Proc. ISESE'06. – ACM. – 2006. – P. 307-315.*

29. *Ardagna D. Rethinking the Use of Models in Software Architecture / D. Ardagna, C. Ghezzi,*

R. Mirandola // Proc. QoSA'2008. – LNCS, Vol. 5281. – Berlin-Heidelberg:Springer. – 2008. – P. 1-27.

30. Шеховцов В.А. Использование качества в роли фактора, управляющего процессом разработки программного обеспечения / В.А. Шеховцов, О.В. Горченок, А.Г. Долгарев и др. // Системний аналіз та інформаційні технології. Матеріали X міжнародної науково-технічної конференції. – К.: НТУУ "КПІ". – 2008. – С. 423.

31. Shekhovtsov V. Facilitating reuse of code checking rules in static code analysis / V. Shekhovtsov, Y. Tomilko, M. Godlevskiy // Proc.UNISCON 2009. – LNBIP, Vol. 20. – Berlin-Heidelberg: Springer. – 2009. – P. 91-102.

Поступила в редакцию 3.06.2010

Рецензент: д-р техн. наук, проф., зав. каф. АСУ М.Д. Годлевский, Национальный технический университет «Харьковский политехнический институт», Харьков.

ІНТЕГРАЛЬНЕ ОЦІНЮВАННЯ СЕРВИС-ОРІЄНТОВАНИХ АРХІТЕКТУР НА ОСНОВІ КОРИСТУВАЛЬНИЦЬКИХ ОЦІНОК ЗМОДЕЛЬОВАНИХ ХАРАКТЕРИСТИК ЯКОСТІ ПРОГРАМНИХ СЕРВІСІВ

В.А. Шеховцов

У роботі пропонується підхід до проектування сервіс-орієнтованих програмних систем, що передбачає інтегральне оцінювання сервіс-орієнтованих архітектур на основі оцінок змодельованих характеристик якості програмних сервісів зацікавленими особами. Цей підхід є частиною комплексу рішень ISAREAD-S, метою якого є підключення зацікавлених осіб до процесу розробки програмного забезпечення через оцінювання сприйманої якості системи (на прикладі її продуктивності й надійності) у контексті її використання. Ми досліджуємо інтегральне оцінювання конкретних варіантів реалізації архітектури в цілому (оцінювання за принципом "миттєвих знімків") і оцінювання окремих архітектурних рішень (інкрементне оцінювання). Ми пропонуємо підхід до інтегрального оцінювання, що складає із трьох етапів: перетворення опису варіанта архітектури або архітектурного рішення в набір значень факторів, що впливають на виконання імітаційної моделі системи, збору користувальницьких оцінок змодельованих характеристик якості й виведення інтегральних оцінок із цих користувальницьких оцінок. Запропоноване рішення реалізоване у вигляді процедури верхнього рівня (політики оцінювання архітектури), заснованої на процедурах нижнього рівня (механізмах), метою яких є збір думок зацікавлених осіб щодо сприйманої якості.

Ключові слова: якість обслуговування, продуктивність сервісів, надійність сервісів, програмна архітектура, архітектурне рішення, оцінювання архітектури, зацікавлені особи.

ИНТЕГРАЛЬНОЕ ОЦЕНИВАНИЕ СЕРВИС-ОРИЕНТИРОВАННЫХ АРХИТЕКТУР НА ОСНОВЕ ПОЛЬЗОВАТЕЛЬСКИХ ОЦЕНОК СМОДЕЛИРОВАННЫХ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНЫХ СЕРВИСОВ

В.А. Шеховцов

В работе предлагается подход к проектированию сервис-ориентированных программных систем, предполагающий интегральное оценивание сервис-ориентированных архитектур на основе оценок смоделированных характеристик качества программных сервисов заинтересованными лицами. Данный подход является частью комплекса решений ISAREAD-S, целью которого является подключение заинтересованных лиц к процессу разработки программного обеспечения через оценивание воспринимаемого качества системы (на примере ее производительности и надежности) в контексте ее использования. Мы исследуем интегральное оценивание конкретных вариантов реализации архитектуры в целом (оценивание по принципу «мгновенных снимков») и оценивание отдельных архитектурных решений (инкрементное оценивание). Мы предлагаем подход к интегральному оцениванию, состоящий из трех этапов: преобразования описания варианта архитектуры или архитектурного решения в набор значений факторов, влияющих на выполнение имитационной модели системы, сбора пользовательских оценок смоделированных характеристик качества и вывод интегральных оценок из этих пользовательских оценок. Предложенное решение реализовано в виде процедуры верхнего уровня (политики оценивания архитектуры), основанной на процедурах нижнего уровня (механизмах), целью которых является сбор мнений заинтересованных лиц относительно воспринимаемого качества.

Ключевые слова: качество обслуживания, производительность сервисов, надежность сервисов, программная архитектура, архитектурное решение, оценивание архитектуры, заинтересованные лица.

Шеховцов Владимир Анатольевич – канд. техн. наук, докторант, Национальный технический университет «Харьковский политехнический институт», Харьков, Украина, e-mail: shekvl@yahoo.com.