УДК 681.518

V.A. SHEKHOVTSOV

National Technical University "Kharkiv Polytechnical Institute", Ukraine

TOWARDS NEGOTIATING QOS REQUIREMENTS ORIGINATED FROM STAKEHOLDER ASSESSMENTS OF SIMULATED SERVICE QUALITIES

The paper introduces an approach for development of service-oriented software systems aimed at negotiating quality of service (QoS) requirements originated from interactive 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. Requirements negotiation is aimed at reaching the compromise between stakeholder requirements and the resources available for implementation by adjusting either resources or requirements. To find the desired adjustment values we propose to formulate a multiple criteria optimization problem according to the methodology of systemwise optimization. The solution is implemented as a high-level procedure (negotiation policy) based on low-level procedures (mechanisms) collecting stakeholder opinions on perceived service quality on the level of both particular (standalone) services and business processes representing service usage contexts.

Key words: *quality of service, service performance, service reliability, requirements negotiation, systemwise optimization, quality assessment, business stakeholders.*

Introduction

The value of a requirements negotiation as a process of finding a respective compromise between (possibly conflicting) stakeholder requirements and the capabilities of the organization has been repeatedly shown in the current literature [1]. It allows the stakeholders and the development team to reach some common ground in what they expect from the system. This is especially important for service-oriented systems as formulating QoS requirements for software services require 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 the ISAREAD-S framework (Interactive Simulation-Aided Requirements Engineering and Architectural Design for Services) [2-4]. It is aimed at investigating the ways to support such stakeholder involvement in a form of assessing the perceived quality (exemplified by performance and reliability) of the service-oriented systems in their 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 requirements negotiation 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 find a respective compromise between stakeholder assessments obtained as a result of applying service quality assessment procedures and the capabilities of the organization by correction of either assessments or resource constraints.

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 service-level and process-level procedures (mechanisms) for organizing the interaction with stakeholders; these mechanisms form the foundation for the negotiation solutions proposed in the paper, Section 4 outlines the proposed approach introducing higher-level procedure (policy) for negotiating QoS requirements originated from stakeholder assessments obtained as a result of applying the mechanisms, Section 5 makes conclusions and describes the directions for future research.

1. State of the art and problem statement

Requirements negotiation has received considerable attention in the software engineering literature (a survey of the available techniques is in [5]). One area of research is related to aligning the choice of architectural design decisions to the capabilities of the organization (its available resources). The important technique belonging to this category is Cost-Benefit Analysis Method (CBAM) [6]. It uses cost-benefit analysis based on resource-related considerations to restrict either the space available for architectural decisions or the level of stakeholder expectations.

Another research area is related to establishing the techniques to facilitate the participation of stakeholders in the negotiation process. The most widely known technique of this kind is based on WinWin negotiation process aimed at arriving to mutually satisfactory outcome for a group of interdependent negotiation participants [7]. This method was extended with groupware techniques to form an EasyWinWin approach [8].

These two research directions were combined in WinCBAM approach [1] which adds cost-benefit considerations to WinWin-based negotiation procedures.

Quantitative WinWin [9] extends the above techniques with quantitative considerations by assigning numerical ranks to both stakeholders and the requirements as seen from the perspectives of different stakeholder groups and uses a tradeoff-based process to make resource-based requirement-related decisions.

The main problem of applying the above methods to the problem of negotiating QoS requirements is related to the fact that stakeholders are not able to experience the system before negotiating process starts. As a result, they are forced to be speculative in their opinions e.g. by formulating narrative statements such as "the system should have good performance under any load" etc. As a result, the understanding of the desired QoS of the system resulted from negotiation process becomes biased towards the view of the IT people.

1.1. 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: *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 ISAREAD-S framework [4] 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 high-level policies based on the proposed mechanisms, we address two research problems related to allowing simulations depend on the artefacts of the development process. First problem is related to the idea of making simulations reflect the chosen software architecture; 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 architecture makes it possible to increase performance by reducing the network load.

Another research problem is related to the idea of making simulations reflect the capabilities of the organization to offer services of the particular quality. As these capabilities depend on the resources (financial, human etc) belonging to this organization; we can formulate the research question: *How to make service simulations rely on the data representing available resources?* To answer it, we need to investigate how such resources influence simulation parameters by addressing the question: *What is the dependency between resources possessed by the organization and the factors influencing service qualities?* An example of such dependency could be the situation when available funds make it possible to install hardware with certain capacities.

The knowledge obtained so far allows us to follow the mechanism-policy separation principle by elaborating higher-level requirements engineering 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 organize the process of negotiating the performance and reliability requirements using the mechanism of interactive assessment of simulated service qualities?* To answer this question, it is necessary to establish the set of necessary procedures which define *requirements negotiation policy*. It should rely on both assessment mechanisms and the techniques allowing simulations depend on the artefacts of the development process.

2. Outline of the assessment mechanisms

In [4] we described the proposed approach to establish service-level and process-level assessment mechanisms for the case when the services are directly accessible to stakeholders. In this section, we briefly outline this approach to the degree necessary to understand the proposed negotiation solution.

2.1. Service-level mechanisms

IAS mechanisms (short for Interactive Assessment of Services) aim at an assessment of simulated service qualities at the level of the particular service. According to the model-driven methodology [10, 11] it is necessary to have two mechanisms of this kind: IASC (for model composition) and IASE (for its execution). IASC inputs include the set of qualities of interest to be simulated and assessed and the set of factors influencing the simulation (simulation parameters [4]). 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 for the qualities of interest. The resulting service-level simulation and assessment model becomes the IASC output. It is transferred to IASE for standalone execution.

IASE is responsible for execution of both simulation and assessment interaction submodels of IASM. 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 service user for assessment via interaction processes described by interaction models integrated into IASM. The IASE outputs are this set of simulated qualities and the set of assessment results.

2.2. Process-level mechanisms

IAP mechanisms (short for Interactive Assessment of Processes) aim at interactive assessment of simulated service qualities in context of usage processes at the level of the particular process, in particular: IAPC (for model composition) and IAPE (for model execution). They rely on service-level 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) 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 model IAPM for the process, IASC creates the IASM model for every service of interest; this model later becomes integrated into IAPM. For every interaction activity, a mechanism for constructing the interaction model is invoked and the resulting interaction model is also integrated into IAPM. The resulting model will contain the simulation logic defined by CFM for the process and simulation submodels of different IASM models (for the services of interest); the assessment logic defined by interaction submodels of these IASM models; the interaction logic defined for all interaction activities.

The IAPM is executed by IAPE. Every run is presented to 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. Outline of the proposed approach

Assessment mechanisms can be used as building blocks for high-level policies. Most of them are supposed to be used at early stages of the system development lifecycle, among them is a negotiation policy intended to solve the problem stated in this paper.

3.1. Assessment adapters

Prior to defining a negotiation policy we introduce the notion of assessment adapter mechanism or adapter for short. Such adapters convert external information into the inputs for an assessment mechanism. For ISAREAD-S framework, two such adapters are being elaborated.

1. *The architecture adapter.* Prior to establishing this adapter we investigate the dependency between the description of the software architecture and the factors influencing service qualities (examples are e.g. [12-14]). 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).

2. *The resource adapter.* We base this adapter on the knowledge of the dependencies between the development resources and the factors influencing service qualities (an example of cost-involving dependency can be found in [15]). It derives the parameter values (i.e. the inputs for the run of the assessment mechanism) from the information about the available resources.

3.2. Applying systemwise optimization in a quality assessment space

To address the requirements negotiation problem, we use *systemwise optimization* [16] methodology (also known as system optimization [17]). This methodology proposes, instead of using the traditional optimization approach where some objective is extremized (maximized or minimized) on the set of possible system variants (e.g. trying to find the variant of the system possessing the best qualities, usually in the presence of some constraints) to apply an approach altering the system goals (e.g. adjusting the requirements to the system, probably together with its constraints) to reach the feasible solution in the criteria space.

This methodology does not look for "the best" optimized solution, some "good enough" (but possibly infeasible) solution is instead selected first and then the adjustments of criteria necessary to make this solution feasible are defined.

Suppose we have a set of desired values for the qualities of the system (originated from the assessment mechanism). These values form a desired (good enough) point (denote it as y^*) in the quality assessment (criteria) space; we further call this space a *quality assessment space*. To be simple, reducing the set of criteria to 2 (exemplifying qualities by performance and reliability), it is possible to see such point at the *trade-off diagram* [18] (a scatter diagram with the axes representing the assessment scales of the respective pair of quality characteristics: the x axis representing first characteristic of the pair and the y axis representing the second one), so in general we can consider that this diagram is the representation of the 2-dimensional slice of the quality assessment space.

Now it is possible to build the neighbourhood area D^d of the desired point. This area denotes the degree of flexibility allowed by stakeholders. It can be built using some flexibility (trade-off) ranges for QoS required values. In [16] this area is called the *directive area* in the criteria space (Fig.1 shows this area together with a desired point at a trade-off diagram).

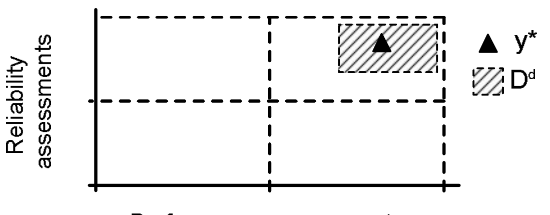

Fig. 1. Trade-off diagram with a desired point and a directive area in the quality assessment space

Further suppose we have resource-related problems that hinder the development process (related to the budget, people, time etc.); these resources sometimes do not allow satisfying all the stakeholder requirements. Suppose particular resource configuration $x \in X$ has been chosen (X is the set of resource alternatives) to reflect the resource capabilities of the development organization. In [16] X is called the set of instrumental variables, we call it the set of resource variants. As a result of applying the resource adapter this particular resource configuration is converted into the set of inputs for assessment mechanisms making these mechanisms produce some set of the simulated software qualities which are, at the end, assessed by business stakeholders. These assessments form the point y_0 in the quality assessment space (the *allowable point*).

Another option to form the allowable point is more radical – it is possible to start from the set of software architecture variants Z and make one particular variant $z \in Z$ converted into the set of inputs for an assessment mechanism using architecture adapter. This way, y_0 corresponds to the chosen software architecture (or the particular set of architectural design decisions [19]).

Now, as in the case of the desired point, it is possible to build a neighbourhood D_0 of the allowable point denoting the degree of flexibility allowed by system developers (usually also in form of the trade-off ranges allowing some other resource configurations in the X set or architectural designs in the Z set to be admissible). We call this neighbourhood the *feasible area* (Fig.2).

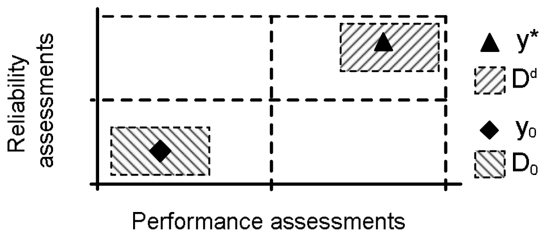

Fig. 2. Adding an allowable point and a feasible area into a trade-off diagram

Several mutual dispositions of D^d and D_0 can be considered. In the most obvious case, these areas intersect in their original form. As a result, it is possible to find some point at their intersection (corresponding to the set of system qualities allowed by the resource variant $x^* \in X$ or architectural solution $z^* \in Z$) which satisfies stakeholder requirements. In this case the allowable point is at the same time the desired point so a feasible (good enough) solution of the problem is available from the very beginning. No optimization problem needs to be formulated in this case.

In most cases, however, these two areas do not intersect. As a result, it is not possible to satisfy the stakeholder requirements with the qualities achieved under current restrictions. Systemwise optimization proposes a methodology to find a solution of this problem (if this solution can be found) at the cost of some trade-offs.

There are several methods to arrive to such a solution. It is possible, for example, to adjust the directive area according to some trade-off from the stakeholder side (Fig.3). This approach differs from the traditional optimization as this trade-off is calculated in the criteria space; one cannot try to arrive to the optimal solution at any cost, and adjusts the goals instead to make desired point allowable at the same time.

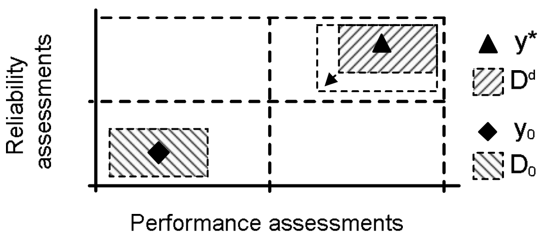

Fig. 3. Adjusting the directive area

The methodology of systemwise optimization [16] states that the adjustment problem has multiple criteria so it is possible to apply such approaches as traditional multicriterial optimization [20], evolutionary algorithms [21], or particle swarm optimization [22] for its solution. The objective of the problem is defined as a function of the adjustment amounts so its results suggest how to alter the area to achieve better compromise, e.g. some quality assessments can be favoured over the others, a Pareto-optimality in relaxing them can be achieved etc. From the practical points of view, after optimal adjustment amounts are found, special negotiations with stakeholders take place to convince them to be less restrictive, the simulator parameters are altered in some way and so on.

It is also possible to select different resource configuration x_1 or architectural variant z_1 which qualities are assessed to form the point y_1 in the quality assessment space that is closer to y^*. This corresponds to a trade-off from the resource side. The best results can be achieved if both these activities are performed simultaneously (Fig.4).

As a result of this process both stakeholder requirements and resource constraints are adjusted in a negotiated way; better understanding of their relation to the success in building the system is achieved.

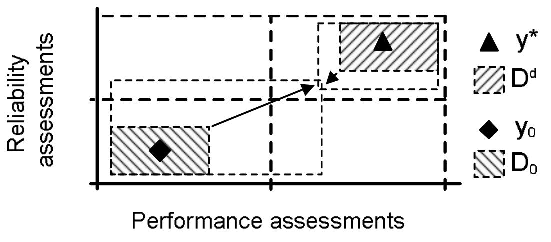

Fig. 4. Adjusting both directive and feasible areas

The process of adjusting the criteria or resource constraints can end in a failure (e.g. when the trade-off ranges are firm and cannot be relaxed further). This is usually the result by itself; it usually makes evident that the whole project cannot be successful.

3.3. Requirements and architecture negotiation policy

The requirements and architecture negotiation policy applies the methodology of systemwise optimization to the negotiation problem as described in the previous subsection (the application of this methodology to software engineering problems is novel). Its BPMN representation is shown on Fig.5. This policy relies to an assessment mechanism extended by an architecture adapter so it is possible to start from an architectural specification (it is also possible to start from the usual input data for assessment mechanism such as the set of services and their qualities). It also relies to a resource-to-parameter conversion made with resource adapter.

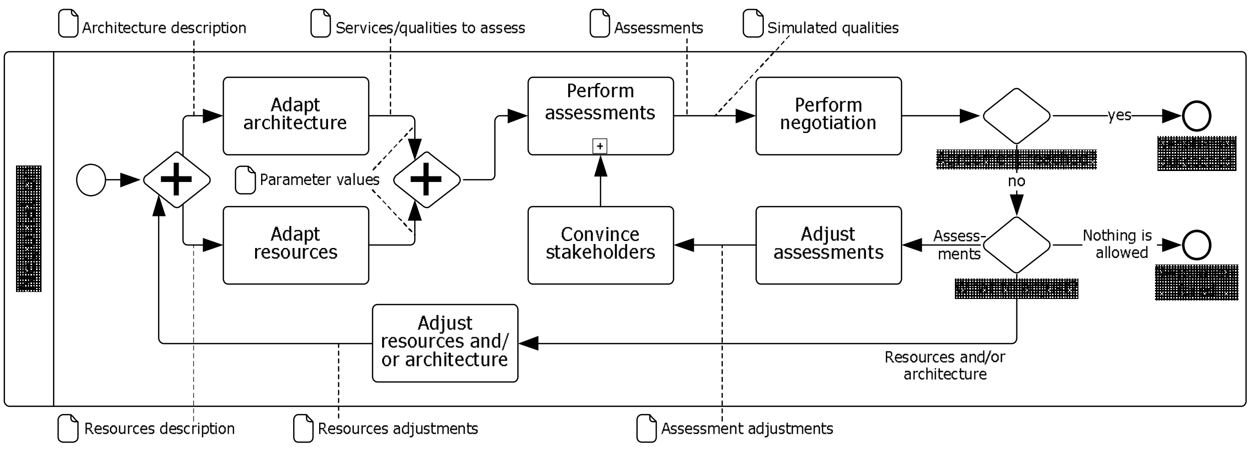

Fig. 5. Requirements and architecture negotiation policy

The negotiation mechanism collects simulated qualities and their assessments from the assessment mechanism and checks if they agree. If the agreement is not reached (e.g. the assessment marks are too low), it is necessary to select the negotiation direction. As defined above, two directions are available: adjusting the capabilities of the organization (altering the feasible area e.g. by adding more resources) and adjusting the opinions of stakeholders (altering the directive area e.g. by lowering the stakeholders' expectancies).

In the first case, the resources after being adjusted are converted by the adapter into the new version of parameter values. As defined above, to find the desired adjustment values a multicriterial optimization problem needs to be formulated and solved.

In the second case, it is necessary to convince stakeholders to change their minds (again, optimization techniques could be used to find the adjustment values to be used in these negotiations) and then run assessment mechanism again with the same values of parameters. It is also possible to pursue both directions convincing stakeholders and adding resources at the same time (this alternative is not shown on Fig.5).

If no more adjustments are possible (e.g. no additional resources are available or the stakeholders are firm in their opinions) but the agreement has yet to be reached, the negotiation process ends in failure. Besides adding resources, other means of adjustment can be utilized, e.g. selecting different software architecture.

Conclusions and future research

In this paper, we defined the principles of new high-level procedures (policies) for negotiating stakeholder requirements obtained from the assessments of simulated software qualities aimed at reaching the compromise between these requirements and resource capabilities of the organization. These policies are based on the methodology of systemwise optimization. Their advantage as compared to known negotiation methods is that stakeholders are able to experience the prospective system before expressing the opinions on its quality.

In future, we plan to elaborate the models underlying resource and architecture adapters, investigate the applicability of different methods for solving the multicriterial problem of finding the optimal adjustments for stakeholder opinions or necessary resources, completely implement the elicitation policy, and establish the validation studies for the proposed solutions.

References

1. *Kazman R. From requirements negotiation to software architecture decisions / R. Kazman, H.P. In, H.-M. Chen // Journal of Information and Software Technology. – 2005. – Vol. 47, N 9. – P. 511-520.*

2. *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. – GI. – 2007. – P. 187-199.*

3. *Kaschek R. Towards simulation-based quality requirements elicitation: a position paper / R. Kaschek, C. Kop, V.A. Shekhovtsov, H.C. Mayr // REFSQ 2008. – LNCS, Vol. 5025. – Springer. – 2008. – P. 135-140.*

4. *Shekhovtsov V.A. Interactive assessment of simulated service qualities by business stakeholders: principles and research issues / V.A. Shekhovtsov // Проблеми програмування. – 2010. – C, 43-48.*

5. *Grunbacher P. Requirements Negotiation / P. Grunbacher, N. Syeff // A. Aurum, C. Wohlin (eds.): Engineering and Managing Software Requirements. – Berlin: Springer. – 2005. – P. 143-158.*

6. *Kazman R. Quantifying the costs and benefits of architectural decisions / R. Kazman, J. Asundi, M. Klein // Proc. ICSE'01. – 2001. – P. 297-306.*

7. *Boehm B. Software requirements negotiation and renegotiation aids: a theory-w based spiral approach / B. Boehm, P. Bose, E. Horowitz, M. Lee // Proc. ICSE'-95. – IEEE. – 1995. – P. 243-253.*

8. *Boehm B. Developing Groupware for Requirements Negotiation: Lessons Learned / B. Boehm, P. Grünbacher, B. Briggs // IEEE Software. – 2001. – Vol. 18, N 3. – P. 46-55.*

9. *Ruhe G. Quantitative WinWin: a new method for decision support in requirements negotiation / G. Ruhe, A. Eberlein, D. Pfahl // Proc. SEKE'02. – ACM. – 2002. – P. 159-166.*

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

11. *Pastor O. Model-driven architecture in practice / O. Pastor, J. Molina. – Springer. – 2007. – 302 p.*

12. *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. – Springer. – 2008. – P. 86-102.*

13. *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.*

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

15. *Mutschler B. Exploring the Dynamic Costs of Process-aware Information Systems through Simulation / B. Mutschler, M. Reichert // EMMSAD'07. – 2007. – P. 163-172.*

16. *Glushkov V. Systemwise optimization / V. Glushkov // Cybernetics and System Analysis. – 1980. – Vol. 16, N 5. – P. 731-733.*

17. *Moiseenko V. System optimization as a generalization of classical optimization / V. Moiseenko,*

V. Yatskevich // Cybernetics and Systems Analysis. – 1997. – Vol. 33, N 3. – P. 416-419.

18. Zhu L. Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process / L. Zhu, A. Aurum, I. Gorton, R. Jeffery // Software Quality Journal. – 2005. – Vol. 13, N 4. – P. 357-375.

19. Jansen A. Software architecture as a set of architectural design decisions / A. Jansen, J. Bosch // WICSA'05. – IEEE CS Press. – 2005. – P. 109-120.

20. Ehrgott M. Multicriteria Optimization / M. Ehrgott. – Springer. – 2005. – 323 p.

21. Coello C. Evolutionary algorithms for solving multi-objective problems / C. Coello, G.B. Lamont, D.A. Van Veldhuizen. – Springer. – 2007. – 800 p.

22. Clerc M. Particle swarm optimization / M. Clerc. – London: ISTE. – 2006. – 243 p.

Поступила в редакцию 10.03.2010

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

УЗГОДЖЕННЯ ВИМОГ ДО ЯКОСТІ ОБСЛУГОВУВАННЯ, ОТРИМАНИХ З КОРИСТУВАЛЬНИЦЬКИХ ОЦІНОК ЗМОДЕЛЬОВАНИХ ХАРАКТЕРИСТИК ЯКОСТІ ПРОГРАМНИХ СЕРВІСІВ

В.А. Шеховцов

У роботі пропонується підхід до розробки сервіс-орієнтованих програмних систем, що припускає узгодження вимог до якості обслуговування, отриманих з оцінок змодельованих характеристик якості програмних сервісів зацікавленими особами. Даний 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. Метою узгодження вимог є досягнення компромісу між вимогами зацікавлених осіб та ресурсами, що доступні для реалізації, шляхом коригування або вимог, або обсягу ресурсів. Для реалізації цього коригування ми пропонуємо сформулювати багатокритеріальну оптимізаційну задачу відповідно до методології системної оптимізації. Запропоноване рішення реалізоване у вигляді процедури верхнього рівня (політики узгодження), що ґрунтується на процедурах нижнього рівня (механізмах), метою яких є збір думок зацікавлених осіб щодо сприйманої якості на рівні як окремих сервісів, так і бізнес-процесів, що представляють контексти їхнього використання.

Ключові слова: якість обслуговування, продуктивність сервісів, надійність сервісів, узгодження вимог, системна оптимізація, оцінювання якості, зацікавлені особи.

СОГЛАСОВАНИЕ ТРЕБОВАНИЙ К КАЧЕСТВУ ОБСЛУЖИВАНИЯ, ПОЛУЧЕННЫХ ИЗ ПОЛЬЗОВАТЕЛЬСКИХ ОЦЕНОК СМОДЕЛИРОВАННЫХ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНЫХ СЕРВИСОВ

В.А. Шеховцов

В работе предлагается подход к разработке сервис-ориентированных программных систем, предполагающий согласование требований к качеству обслуживания, полученных из 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. Целью согласования требований является достижение компромисса между требованиями заинтересованных лиц и ресурсами, доступными для реализации, путем коррекции или требований, или объема ресурсов. Для реализации данной коррекции мы предлагаем сформулировать многокритериальную оптимизационную задачу в соответствии с методологией системной оптимизации. Предложенное решение реализовано в виде процедуры верхнего уровня (политики согласования), основанной на процедурах нижнего уровня (механизмах), целью которых является сбор мнений заинтересованных лиц относительно воспринимаемого качества на уровне как отдельных сервисов, так и бизнес-процессов, представляющих контексты их использования.

Ключевые слова: качество обслуживания, производительность сервисов, надежность сервисов, согласование требований, системная оптимизация, оценивание качества, заинтересованные лица.

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