Journal of International Scientific Publication:
Materials, Methods & Technologies, Volume 4, Part 1
ISSN 1313-2539, Published at: http://www.science-journals.eu

QUALITY ATRIBUTES DRIVEN WEB SERVICES DESIGN METHOD

Edzus Zeiris, Maris Ziema
Riga Technical University, Faculty of Computer Science and Information Technology
Meza 1/3, LV-1048, Riga, Latvia
edzus@zzdats.lv, maris@zzdats.lv

Abstract

When developing e-services, it is important to select an e-service system architecture that ensures that the e-service complies with all the quality criteria set for it in order to facilitate broader application of the e-service. This means that it is necessary to use effective design methods to create e-service systems that ensure the creation of an acceptable e-services system architecture. Existing design methods address the quality aspects, but they have a drawback – system architect's subjective opinion that not always might lead to the best solution. Instead, an e-services system design method that is based on formal use of optimization methods can be employed. Initially the e-service algorithm graph is created, and by recursively segmenting its vertices in all possible ways the web service graphs are obtained. The segmentation of the algorithm graph means that all the possible solutions that can affect the quality of the e-service system architecture are dealt with. Using multi-criteria optimisation, a

Pareto optimality set is obtained from all the web service graphs and the web service graphs of the obtained set can serve as the basis for selecting an acceptable e-service system architecture.

Key words: *web services, e-service, architecture, system quality, e-service algorithm graph, web services graph, multi-objective optimization, Pareto compromise set*

1. INTRODUCTION

The rapid development of information technology systems in recent years and the need to optimise resources has also lead to a very rapid development of e–services. Public sector services in all developed countries are being made electronic at a very rapid pace. The European Union has listed those service areas that the Member States shall to deliver electronically and that would facilitate the creation of European scale services.

When developing e–services, it is important to build the architecture of e–services system that makes the e–service compliant with all and any quality criteria specified for it, as this will expand its usability[1,2,3,4]. Furthermore, alongside with delivering electronic services, they will have to be offered through traditional methods too as not all customers are able to access e–services. Consequently, the development costs of the electronic version of a service have to be reduced. This means that efficient design methods should be used in creating e–service systems that would ensure the building of an acceptable e–service system architecture. The existing designing methods not always are suitable for solving this problem. Their major drawbacks are related to the exclusion of the subjective position of the system designer, the compliance with all and any criteria set for the system.

E–service system architecture consists of a number of parts.

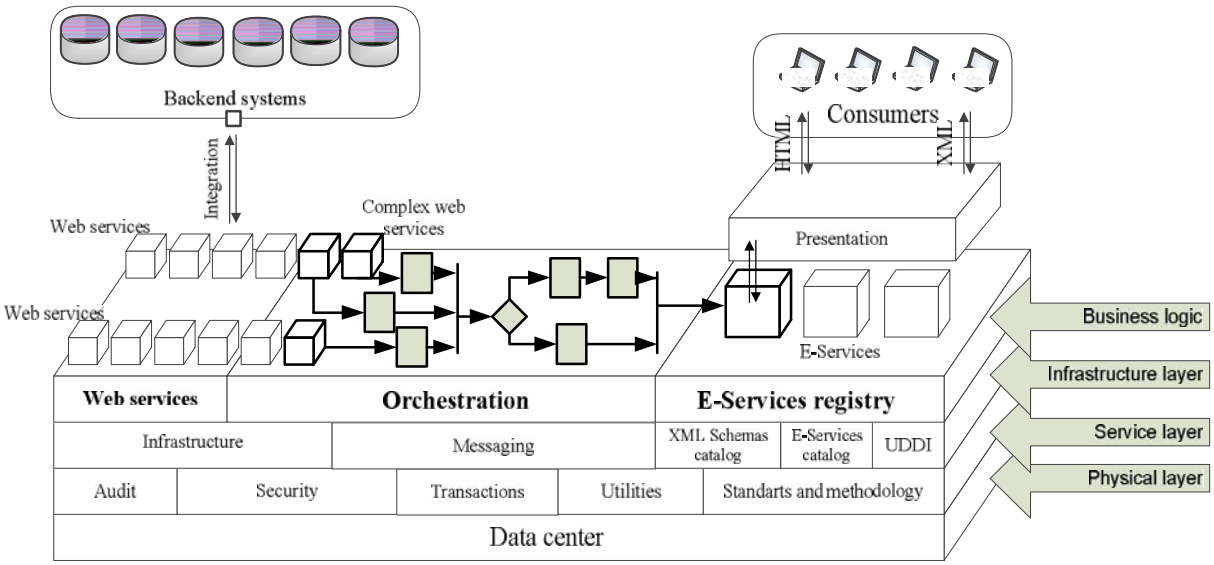

Fig. 1. E – Service system architecture

0. shows how the systems and system components used in the service are combined into unified e–service system architecture. For every data object that is required in the implementation of e–service, an XML Schemas Catalog has to be developed. Data call from the relevant functional system is done by means of the web services. When web service calls are done, also metadata that describe the

request are sent. Also any information that is required for the audit trails is sent together with the metadata. Web services can be distributed into two groups: simple and complex. Complex web services in fact are logical combinations of several simple web services that result from the process integration requirements; a combination may comprise simple services of one or several independent functional systems. Complex web services can be executed by using a BPEL processor. A BPEL processor is used as the orchestration (integration) environment for the e–service's web services. Portals, one–stop agency applications etc ensure the delivery of e–services to the users. E–service entry forms, stop points, information on payments and execution results are transferred through HTML or HML pages that can be used in the portal to implement the service using the XSLT transformation. Web service and e–service holders, i.e. institution specialists and system administrators who are responsible for the maintenance and development of web services and e–services, must have a possibility to intercommunicate on various issues connected with the execution and advancement of web services and e–services. Also, asynchronous e–services have to be executed. Messaging systems are designated for this purpose. The messaging system enables working with text messages and work tasks. The application is inter-integrated with the e–services register, from which it receives data on XML schemas, web services and e–services. On the other hand, the messaging application is a client of the orchestrations as messages on the execution of web services and e–services are received from them. Data on all XML schemas, web services and e–services are entered in a single register called E–services Register. The register keeps all the versions of schemas, services and e–services so that this information is accessible to anyone who is engaged in the development and advancement of e–services.

The main problems in the e–service system architecture are related to the e–service's data standardization, audit, the creation of web service catalog, the execution of asynchronous e–services, security and the development of e–service orchestration that is connected with web service design.

There are several quality criteria that can be specified for the e–service system and that must be complied with in the e–service system architecture: development costs, maintenance costs, integrity, scalability, security etc. Usually, the quality criteria are mutually controversial. To evaluate the quality of system architecture, ISO 9126 Standard for Software Engineering-Product Quality Assessment and its quality metrics and guidelines for the use thereof can be used[5]. To comply with the quality criteria specified for the e–service system, the e–service system architecture must satisfy all the requirements. In such cases, it is not enough to use functionality-based or object-oriented design approach. It is necessary to apply methods that solve architecture development quality problems such as *Attribute-Driven Design method ADD[1,6], Software Architecture Analysis method SAAM[4,7], Jan Bosch software architecture design method[2], Architecture Tradeoff Analysis Method ATAM[4,8], Hazard Analysis of Software Architectural Designs HASARD[4], etc.*

The methods listed above deal with system quality, but they all have drawback. It is the subjective position of the system architect or the person who designs the system architecture and the possibility that the best solution is overlooked or even not dealt with at all. Seemingly the best variant can be chosen, but it may turn out to be the local minimum only, since not all the possible variants are dealt with. Furthermore, it is practically impossible to deal with all the possible variants in the offered methods, since the number of possible systems to be developed could reach rather high levels for the system architect to be able to process them within an acceptable time.

When designing e–services, it is necessary to reduce the resources consumed for the development of e–service and at the same time to provide the appropriate quality of the selected e–service system architecture. It means that it is required to select a system architecture design method that could be automated in apart and that would deal with all and any possible architecture solutions to prevent selecting a local minimum and to take into account all the quality criteria specified for the system

architecture. In creating e–service system architecture, the design of the web services used in the e–service and the orchestration thereof is of great importance in order to select an acceptable e–service system architecture[9,10,11].

2. E-SERVICE ALGORITHM GRAPH

As the start of the e-service design is the definition of the requirements and system analysis. In this step e-service process model is described which is as a base for e-service algorithm graph development and web services design.

The e–service algorithm graph is defined as the oriented graph = (,), where = { , , ,⋯ } is a final set – graph vertexes, which, to their substance, are the executable operation of the e–service algorithm, and ⊂ × . The edge = (,) in the graph means that in the execution of the e–service algorithm the execution of the operation s followed by the execution of the operation . The edges in the e–service algorithm graph show the information flow (Fig. 2.). When creating the e–service algorithm graph that can be used as the basis for developing the e–service system architecture, some of the restrictions related to SOA have to be considered:

1. Every algorithm graph vertex must perform an independent executable activity. It means that the vertex operates in an atomic transaction and is not connected with the algorithms executed in other vertexes. This condition is connected with high cohesion and minimal dependence.
2. Every vertex must contain at least one executable operation = , , ,⋯ . In practice, this is related to the algorithm implementation methods.
3. The operations that repeat during the execution of the algorithm may not be implemented in the graph as various vertexes∀ , , ≠ , ∈ , ∈ , ∩ = ∅. It is required in order to ensure high cohesion and to exclude, from the very beginning, the processing of unnecessary variants.

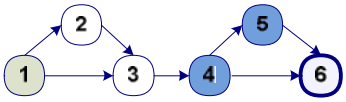

Fig. 2. Example of E – Service Algorithm Graph

The basis for selecting the e–service system architecture is the selection of web services. It means that the e–service algorithm has to be implemented as web services in order to design the e–service system architecture. The algorithm graph can be realised as a web service graph in several ways: from realising every algorithm vertex as an individual web service to realising all the algorithm vertexes as one web service. It means that various e–service algorithm graph segments can be realised as web services, in this way changing the e–service architecture and affecting various e–service execution metrics (Fig. 3.).

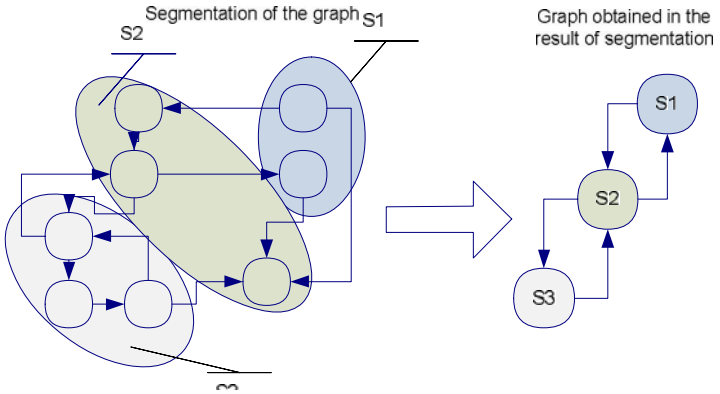

Fig. 3. Segmentation of the Graph

To segment the graph and describe it formally, we will deal with the graph *G* as the depiction Γ in the vertexes set , attributing to every vertex a sub–set(⊂), which actually consists of the vertexes that can be reached from the vertex . Γ is the edges coming out from the vertex and Γ is the edges going into . For such e–service algorithm graphs, there is at least one vertex (∈), which none of the circles Γ = ∅, which we will call the origin of the e–service algorithm, enters into; similarly, there exists at least one vertex (∈) from which none of the circles Γ = ∅, which we will call the result of the e–service algorithm, is going out. To find all the possible web service graphs from which the one to be implemented in the e–service system architecture is selected, initially the given algorithm graph is assumed as the first possible web service graph. All other web service graphs are obtained by segmenting the initial web service graph, which concurs with the algorithm graph. Since several segmentations are possible, the set = { }, that contains all the possible graphs that are recursively derived from the initial web service graph *G* is defined. Graph transformations are done by merging the vertexes. The combination of the vertexes and is as the combination of the incomming and outgoing edges of both vertexes. Γ = Γ ∪ Γ and Γ = Γ ∪ Γ . The set of methods in the new created vertex develops as follows = ∪ . Developing a web service graph in practice, various limitations will be encountered; e.g. functionalities of different systems have to be integrated in one e–service, or it is required to use existing web services and therefore limitations, or in other words, the vertexes' features ( ) can be applied to the assumed initial web service graph, which is the algorithm graph. In the given example (Fig. 2.) vertexes „4" and „5" re created in another system; therefore, they are marked in different colour. In other words ( ) = ( ) = ( ) = ( ) = 0 and ( ) = ( ) = 1. In this case, the combination of vertexes only is possible if ( ) = ( ).

Considering that the algorithm graph that initially is assumed as the web service graph has attributed features, the combination of vertexes only is possible if ( ) = ( ).

Obtaining all the possible algorithm graph segments is an NP – complete problem, where all the possible web service graphs only may be obtained by executing a recursive algorithm. Considering that such segmentation is time-consuming, in this way it is possible to segment in real time an algorithm graph that have restrictions or does not have too many vertexes. If it is necessary to process a larger algorithm graph and to develop all the possible web service graphs by segmenting it, the task must be divided into several smaller tasks. In the algorithm graph, the algorithm graph = (,) segments G1 and 2, that are logically separable have to be found, and the given algorithm graph has to be divided into two or more smaller algorithm graphs to which complete segmentation can then be

applied. In other words, $1 = (1, 1)$, $1 = \cdots$ and $2 = (2, 2)$, $2 = \{\cdots\}$,where $1 \cup 2 =$, $1 \cap 2 = \emptyset$, $\Gamma \subseteq \Gamma$ un $\forall \in$, $\Gamma \cap \forall \in$, $\Gamma' = \emptyset$. The division of a large algorithm graph in smaller graphs is shown in Fig. 4.**Error! Reference source not found.**, where every of the specified segments can be solved as an individual web service design task. The division of graphs is related to computation tasks where it is necessary to divide the graph in two (or more) large parts, minimising the number of edges that cross the split. To solve this problem, it is possible to use standard algorithms from the theory of graphs, for example, flow-based algorithms

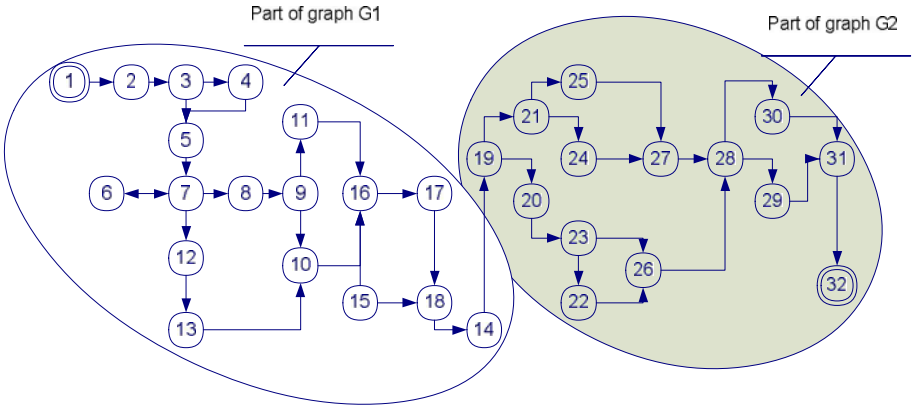

Fig. 4. Initial Segments of Large E – Service Algorithm Graph

If in the result of segmenting the graph it is impossible to find any graph segments that are interconnected with only one edge, then graph segments with minimum edge interconnection are found in the algorithm graph, and two new vertexes are implemented to transform the algorithm graph into a web service graph: one vertex in graph segment G1 as the final position and one vertex for graph segment G2 as the original top[9,10,11].

3. MULTI-OBJECTIVE OPTIMISATION

By segmenting the algorithm graph, web service graphs are obtained that can be used as the basis for the development of the e–service system architecture. There are several criteria that have to be taken into account when selecting an acceptable e–service system architecture, and very often they are contradictive. It means that it is necessary to find a web service graph in the set (which has been formed after segmenting the algorithm graph) which are the best (or not worse) according to the specified criteria and on the basis of which an acceptable e–service system architecture can be built. It is very essential to deal with all the criteria simultaneously, as it is impossible to determine which of them is more important and, also, they are not mutually comparable. The task of finding the web service graph can be reduced to a multi-criteria optimisation task. In a result of multi-criteria optimisation the Pareto optimality in the set is obtained, which can be formally defined as follows:

$$( ) \rightarrow \min_{\in \Omega} \rightarrow \quad (1)$$

where $( ) = \{ ( ), ( ), \cdots ( )\}$ – criteria to be minimised; Ω – X definition area. The solution $\in \Omega$ s will be called a Pareto optimality if and only if $\in \Omega$ does not exist such that

$$( ) \leq ( ) \quad (2)$$

for all $= \{1,2,\cdots\}$, k where at least one is a strong inequality. In other words, for any value $\in$ in the set Ωn no better than the selected value can be found according to every criteria to be optimised. If depicting the Pareto optimality set graphically by two criteria to be optimised, they are all solutions that are nearest to the origin[12,13].

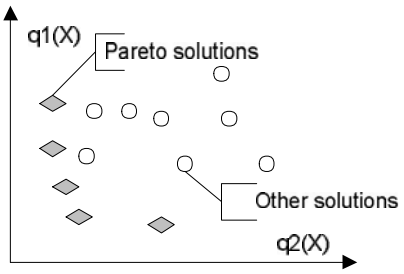

Fig. 5. Pareto compromise set

To be able to mutually compare and view simultaneously various criteria to be minimised, they must be normalised to achieve that the obtained values are similar. To normalize the values of the target functions to be optimized, every calculated value is divided by the maximal calculated non-zero value of the respective criterion. In this case, the value area for every target function is in the area [0,1]. To normalize the values, the following formula will be used:

$$= \frac{}{\max_{\in\Omega} ( )} \quad (3)$$

In the selection of the web service graph, the application of the Pareto optimality set is the following. The e–service algorithm graph is given. Initially it is assumed as web service graph *G*. The task is to find the web service graph set $= \{ ^{*}\}$ that is defined in the area **Ω** (i.e. all and any possible algorithm graph segment combinations) and to minimise the criteria *Q* that are all quality criteria specified for the system. To assess the system quality during its designing stage, one of following four quality assessment methods can be used: scenario-based, simulation-based, mathematical model-based, experienced-based[1, 4, 14, 15].

4. WEB SERVICES DESIGN METHOD

To design web services the Quality Attributes Driven Design Method can be uses that is based on quality criteria and consists of the following steps:

1. Initially it is necessary to create the e-service algorithm graph G that describes the e-service to be designed;
2. In the e-service algorithm graph, it is necessary to determine the possible restrictions for inter-segmentation of vertexes;
3. By recursively segmenting the e-service algorithm graph, the web service graphs $= \{ \}$ are obtained that are then used as the basis for the development of the e-service system architecture; The e-service algorithm graph is assumed as the initial web service graph;

4. The numerical vales of quality metrics of the obtained web service graphs are calculated;
5. From all the web service graphs, the Pareto optimality set = { *} is obtained;
6. Web service graphs of the obtained set P can serve as the basis for selecting an acceptable e-business system architecture. The web service graphs that are contained in the obtained Pareto optimality set can be designed in detail and implemented in the e-business system architecture;
7. In most cases, the Pareto optimality set P contains more than one solution. If the obtained set is sufficiently small, the selection of web service graphs from the set can be left to the system designer. However, if there are many possible solutions, then in order to select a particular web service graph, the criteria can be decreased or combined with another optimisation or design method
8.

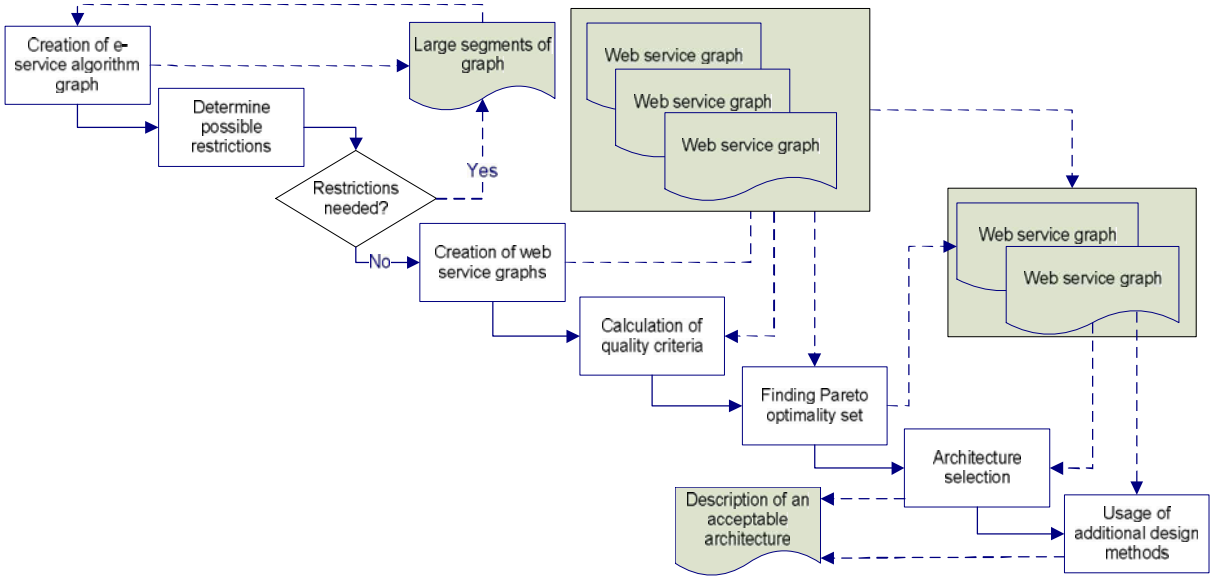

Fig. 6. Web services design method life cycle

As it was mentioned in the section 1, there are methods that are multi-criteria and that make it possible to evaluate the quality characteristics of the system architecture prior to development; however, the accuracy of these methods relies only and solely on the expert's experience and the compliance with the methodology. It means that the possibility that a local minimum is selected is not excluded. Another large drawback of these methods is the fact that the subjective position of the expert is not excluded, as in most cases only one system architect that takes the decision is engaged in the system design. Considering that successful advancement of computer systems relies upon their quality, any methods that are not multi-criteria do not provide the desired result in most solutions. The application of these methods can cause loss both to the client who ordered the computer system and the developer.

The existing multi-criteria design methods are acceptable in designing large and complex computer systems, which cannot be formally described and for which the mathematic multi-criteria optimisation methods cannot be applied. In such cases the ATAM method, which involves several system design experts, can be very efficient. Also other methods looked at herein could be useful in designing large

computer systems if competent experts of the relevant sector who can take decisions and evaluate the quality characteristics of the computer system to be designed are called in.

To design the e–service system architecture, a design method that provides fast and quality result regardless of the qualification of the involved experts is required. Since the e–service algorithm can be described formally in the form of an algorithm graph, it is possible to use the offered web service design method and the selection of an acceptable e–service system architecture that is based on multi-criteria optimisation. In such and similar cases, it is the multi-criteria optimisation method that should be used to prevent that a local minimum is selected.

Considering that in the result of multi-criteria optimisation a Pareto optimality set which contains several results is obtained, then in more complex cases when the system architect is unable to make a selection from the Pareto optimality set, any other design, decision-taking or optimisation method that helps select a solution that is contained in the Pareto optimality set can be applied. In this case, the possibility that a local minimum is selected is excluded.

The comparison of the design methods by the following assessment criteria is offered in Table 1:

1. Multi-Criteria – in selecting the design method, it is important to see to that all the quality criteria specified for the system are dealt with simultaneously;
2. Is the subjective expert position excluded – in many design methods, the selection of solution depends on the system architect, whose subjective opinion influences the system to be designed;
3. Can be estimated in advance – since the development of a system is very time-consuming, it is essential that the quality characteristics can be assessed prior to commencing it;
4. Pareto guaranteed – if the solution is selected from among all the solutions in the Pareto optimality set, it is sure that the selected solution is one of the best possible;
5. Exact solution – it is important to make sure whether the design method provides one or more solutions;
6. Accuracy of the solution: expert – the accuracy of the solution is determined by the system architecture's experience; methodology – the accuracy of the solution is determined by complying with the descriptive methodology; formula – the accuracy of the solution depends on the accuracy of the formulae used.

5. PRACTICAL VALUE

The offered method is designed and evaluated based on doctoral studies and is one of the result of doctoral thesis "E-Services System Architecture" prepared by the author in Riga Technical University. The application of the Quality Attributes Driven Web Services Design Method has been analysed by using a number of study examples; also, e-services system architecture designs obtained by using this design method and classical object-oriented or functionality-based design methods have been compared. E-services developed for Latvian government and local governments in the portals www.latvija.lv, www.riga.lv and www.epakalpojumi.lv have been used to analyse and approbate the method.

ISSN 1313-2539, Published at: http://www.science-journals.eu

Table 1. Comparison of architecture design methods

Method / Comparision Nr.	1.	2.	3.	4.	5.	6.
Functional based	No	No	No	No	Yes	Expert; Methodology
Object-oriented	No	No	No	No	Yes	Expert; Methodology
ADD	Yes	No	Yes	No	Yes	Expert; Methodology
SAAM	Yes	No	Yes	No	Yes	Expert; Methodology
Jan Bosch	Yes	No	Yes	No	Yes	Formulas; Expert; Methodology
ATAM	Yes	Partly, there is more than one expert.	Yes	No	Yes	Experts; Methodology
HASARD	Partly, method is oriented to achieve security requirements.	No	Yes	No	Yes	Expert; Methodology
Multi-Criteria, finding the Pareto compromise set	Yes	Yes	Yes	Yes	No	Formulas
Combined multi-criteria	No	Partly. To select solution from Pareto compromise set there additional analysis is required.	Yes	Yes	Yes	Formulas; Experts; Methodology

Analysing the results obtained from by using the method and comparing them with already implemented e-services, it can be seen that the design method can be used in practice and that it does not have any apparent drawbacks that exclude any essential solutions. The used design methods are different as to the resources that are required for building an acceptable system architecture. It was established that several e-services had been developed iteratively during several years, improving them constantly to achieve the desired quality, and that several service versions that include improvements in the system architecture had been released. To develop an e-service, a number of experienced system architects had been engaged, which means that resources had been considerably consumed. The offered method achieves the same result by engaging one system architect only and in a comparatively shorter period.

6. CONCLUSION

In designing e–service systems, problems related to the quality criteria set for the system have to be solved. Usually, a number of mutually controversial quality criteria will be specified for the e–service system. In such cases it is necessary to find a compromise solution in finding an acceptable e–service system architecture. Currently, there are several design methods that take into account the quality criteria requirements specified for the system. However, the existing methods have several drawbacks. Firstly, they do not deal with all and any possible system architecture variants. It means that not always the best possible solution will be selected. Secondly, the subjective position of the system architect is taken into account in all the methods. This problem can be solved by using a web service design method that is based on multi-criteria optimisation that finds the compromise result and minimises the subjective factor.

The offered methodology for the designing of e–service systems can be used also in other systems that are not complex algorithmically (i.e. algorithm graph can be built) and are based on SOA and web services.

ACKNOWLEDGEMENTS

This work has been supported by the European Social Fund within the project „Support for the implementation of doctoral studies at Riga Technical University".

REFERENCES

1. Bass L., Clements P., Kazman R. Software Architecture in Practice 2nd ed. - Boston: Addison-Wesley , 2003.

2. Bosch J. Design and Use of Software Architectures. - Harlow: Addison-Wesley , 2000.

3. Hofmeister C., Nord R., Dilip S. Applied Software Architecture. - Massachusetts: Addison-Wesley , 2000.

4. Zhu H. Software Design Methodology From Principles to Architectural Styles. - Amsterdam: Elsevier Butterworth Heinemann , 2005.

5. International Organization for Standardization: Software engineering - Product quality / Internets. - http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

6. Bass L., Klein M., Bachmann F. Quality Attribute Design Primitives and the Attribute Driven Design Method// Lecture Notes In Computer Science. Revised Papers from the 4th International Workshop on Software Product-Family Engineering. - London: Springerferlag, 2001. - Nr. 2290 - 169.-186. lpp.46. Kan S. Metrics and Models in Software Quality Engineering 2nd ed. - Boston: Addison-Wesley , 2003.

7. Kazman R., Bass L., Webb M., Abowd G. SAAM: A Method for Analyzing the Properties of Software Architectures// International Conference on Software Engineering. Proceedings of the 16th international conference on Software engineering. - Los Alamitos: IEEE Computer Society Press, 1994. - 81.-90. lpp.

8. Kazman R., Klein M., Clements P. ATAM: Method for Architecture Evaluation. - Pittsburgh: Carnegie Mellon Software Engineering Institute CMU/SEI-2000-TR-004 ESC-TR-2000-004 , 2000.

9. Zeiris E., Ziema M. E-Service Architecture Selection Based on Multi-criteria Optimization// Product-Focused Software Process Improvement 8th International Conference, PROFES 2007, Riga, Latvia, July 2007 Proceedings. - Berlin Heidelberg: Springer-Verlag, 2007. - 345.-35

10. Zeiris E., Ziema M. Criterions for Architecture Estimation and Architecture Selection of E – Services System// Scientific Proceedings of Riga Technical University. Computer Science. Part 5. Volume 27. - Riga: RTU, 2006. – p. 91.-98.

11. Zeiris E., Ziema M. Selection of E – Service Systems Architecture// Scientific Proceedings of Riga Technical University. Computer Science. Part 5. Volume 32. - Riga: RTU, 2007. – p. 99.-107.

12. Miettinen K. Nonlinear Multiobjective Optimization. - Boston: Kluwer Academic Publishers , 1998.

13. Triantaphyllou E. Multi-Criteria Decision Making Methods: A Comparative Study. - Dordrecht: Kluwer academic publishers , 2000.

14. Kan S. Metrics and Models in Software Quality Engineering 2nd ed. - Boston: Addison-Wesley, 2003.

15. Losavio F. Quality Models to Design Software Architecture// Journal of Object Technology. - 2002. - Nr. Vol. 1, No. 4