REST ARCHITECTURE STATE OF PRACTICE IN MACEDONIAN IT COMPANIES

Agon MEMETI
Department of Informatics, Faculty of Math and Natural Sciences, University of Tetovo, Macedonia
Florinda IMERI
Department of Informatics, Faculty of Math and Natural Sciences, University of Tetovo, Macedonia
Betim ÇIÇO
Faculty of Contemporary Siences and Technologies, South East European University

ABSTRACT

Rebuilding applications by reusing the existing ones is of primary importance for the IT developers. Every system uses resources such as Web pages, business information, etc. that can be represented in a computer-based system because the main purpose of IT department is making these resources accessible to its clients. Service architects and developers are interested in creating services that are implementable, maintainable, extensible and scalable. It is the RESTful design that promises this and a lot more. It has been more than a decade since the introduction of *Representational State Transfer* (REST), and it seems to become one of the most important technologies for Web applications. Every major development language now includes frameworks for building RESTful Web services. Considering the importance of REST while reusing services, we have conducted a survey in IT companies in Macedonia, which relies upon collection of empirical data. The survey aimed at exploring developers' experience on REST Architecture in IT Companies and the data are here reported and evaluated.
Keywords: REST Architecture, service reuse, resources, empirical data.

1. INTRODUCTION

The products or services offered by businesses today rely on information products developed by enterprise IT department. Time to market of a product is very significant for enterprises in order to be competitive in the market. As a result, minimizing the production time by IT departments for a given product is essential for the enterprises (Algermissen, 2011).

The primary attraction for developers is to rebuild applications by reusing the existing ones. Reusing existing objects/components or making use of class libraries, functions, data, etc is as old as retelling a story (Imeri and Antovski, 2012). To avoid duplicate designs of similar processes, abstract business processes were used to support for reusability.

Every system uses resources, such as Web pages, business information, or anything that can be represented in a computer-based system. The purpose of a service is to provide to its clients an easy access to these resources. Also service architects and developers are interested in creating services that are easy to implement, maintain and are extensible, and scalable. It is the RESTful design that promises this and a lot more.

It has been more than a decade since REST introduction, and it seem to become one of the most important technologies for Web applications. Its importance seems to grow very quickly since all technologies are moving towards Application Programming Interface (API) orientation. Every major development language now includes frameworks for building RESTful Web services. As such, it is important for Web developers and architects to have a clear understanding of *Representational State Transfer* (REST) and RESTful services. Many books and tools have been created but there is still a general lack of understanding its fundamentals as an architecture style. The reason perhaps could be found in the fact that REST was presented in a doctoral dissertation, with relatively high entry barriers for its understanding, or because the description used models that were more oriented towards documentation than to working practitioners (Wilde and Pautasso, 2011).

Seeing the importance of REST while reusing services, we have conducted a survey in IT companies in Macedonia, which relies upon collection of empirical data. The survey aimed to explore developers' experience on REST Architecture, its benefits and challenges from their viewpoint and see the possible contributors towards successful reuse, which could help increase of the knowledge and understanding of REST. We have prepared a questionnaire and sent it electronically to developers in 30 companies in Macedonia and other Balkan countries, but only the companies in Macedonia replayed to the questionnaire. Consequently, comparing the results obtained from IT companies in Macedonia and the region failed.

The forthcoming part of the paper reports on the state of the art of the REST Architecture, the questionnaire and research questions and empirical results. In the end, the conclusions are drawn.

2. REST ARCHITECTURE

Restful Services as architectural style lighter than SOAP-based Web Services, due to its simplicity, heterogeneity and web-based format, entities/resources are identified by unique URLs, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages (Memeti *et. al.*, 2015).

REST originated at the intersection of academia and software development, among the architects of the World Wide Web (Adamczyk *et al.*, 2011). Unfortunately, researchers have only recently started to work on RESTful services. No paper about RESTful Web services has been reported up to 2007. Information about RESTful services was reported only in 2010.

The REST architectural style has been chosen for the resource centric approach because it could be easily applied and naturally mapped to it. The HTTP methods can be mapped to the Create (POST), Retrieve (GET), Update (PUT) and Delete (DELETE) operations and the XML documents can be used to provide a uniform resource representation (Szepielak, 2006).

Many frameworks and tools for building RESTful Web services are available today. They are written in different programming languages and range from simple to quite sophisticated in their support of HTTP and other Web technologies. As they continue to improve, misunderstandings and violations present in today's Web services will likely lessen (Wilde and Puatasso, 2011).

When multiple providers offer the same service, a client has a choice and can select the most suitable one. Often, this choice comes down to the Quality of Service (QoS) parameters (Wilde, 2011). RESTful Web services today ignore QoS requirements; their only concern is providing functional interfaces. To add QoS parameters to RESTful services, a language for describing the parameters and a mechanism to incorporate the description in the HTTP payload is needed. Defining a standard QoS description language might benefit from the work in Semantic Web. Semantic Web ontologies define standard ways of interpreting information, such as QoS parameters, enabling all clients to interpret them the same way (Wilde, 2011).

Building an integrated software environment in an enterprise often requires developing large amounts of Web services. The integration efforts can be greatly reduced by using a specialized framework for their development. Providing such tools that simplify software development in integration projects is essential for optimizing their efficiency and cost. One of the most important choices to make when building an integration solution is to select an appropriate integration approach and suitable technologies for its realization (Wilde, 2011).

According to Fielding, (2000) the four principles of REST (called constraints) are: i) identification of resources, ii) manipulation of resources through

representations, iii) self-descriptive messages and, iv) hypermedia as the engine of application state.

These principles describe the architecture of systems and interactions that make up the Web. The building blocks of the Web are called resources. A resource is anything that can be named as a target of hypertext (e.g., a file, a script, a collection of resources). In response to a request for a resource, the client receives a representation of that resource, which may have a different format than the resource owned by the server. Resources are manipulated via messages that have standard meanings; on the Web, these messages are the HTTP methods (Fielding, 2000).

Activities that companies are supporting such as knowledge management and communication or decision making, the development approach such as object -oriented or component-based approach which is characterized by reusability (elements for re-used in other workflows), substitutability (alternative implementations easily inserted with precisely specified interfaces, and with the ability to verify and validate substitutions), extensibility and scalability (the ability to extend system component and to scale it, by increasing capabilities of individual components, with new functionalities), customizability (ability to add new features based on the needs of a particular domain), and composability (easy construction of complex solutions using basic components and the ability to estimate the efforts of system integration and compose) are of great importance for the developers.

According to (Choi and Kim, 2008) the advantages of REST architecture are: i) *availability:* utilization of service which increases the reusability service, ii) g*enerality:* a more generic service, since a reusable service could be implementable for unknown future requirements, iii) *understandability:* service and its interface are understandable. The more understandable it is, the more it is used, iv) *functionality:* to have a highest degree of guaranteed reuse, the service should pack with a complete range of functionality, v) *reliability:* since the service is to be used by many consumers, it should be reliable, be able to operate without any failure continuously and, vi) *portability:* the portable services which are to be consumed through any kind of environment without being modified increases the reusability service.

3. EMPIRICAL ANALYSIS

The present paper reports on usability of the REST architecture, its advantages and challenges. Consequently, a questionnaire with predefined answers and some open questions for general reflection was compiled. The research involved both qualitative and quantitative studies. In order to have clear understanding, at some points, some of the data were transformed from qualitative into

quantitative presentations.

As regarding the sampling frame and size of our survey, the data presented in this paper are answers taken only from 10 software developing companies in Macedonia. The questionnaire was electronically sent to 20 companies in the region, but 10 companies in Macedonia replayed to the questions, only.

Reported in Table 1, the questionnaire consists of 13 questions which have not been made before. Below are presented research questions of our survey.

1. *RQ1. Do companies use REST Architecture?*
2. *RQ2. Does developers' experience with REST influence to the percentage of service reuse?*
3. *RQ3. Which factors influence toward use of REST Architecture?*
4. *RQ4. Which challenges and benefits are reported from companies?*

Table 1. The questionnaire

Q1.Which Architectural Styles and Patterns are used by your organization?
Q2.What is your experience with REST Architecture?
Q3.Which activities are supporting the architectural styles used by your company?
Q4.Which development approach is used by your organization?
Q5.When a development team set up a project, would they start to go through old services or just start to write the new program application?
Q6.What is the percentage of service reuse?
Q7.How is the software production process carried out in your organization?
Q8.While enhancing performances of existing applications, do you have any problems with permissions to use same data?
Q9.If yes, how do you resolve the problem?
Q10.Does the quality and flexibility of applications/services increased using REST (based on your experience)?
Q11.Does the productivity of applications/services increases using REST (based on your experience)?
Q12.Which advantages of REST Architecture are more important from your point of view?
Q13.Which challenges do you face while using REST Architecture?

3.1 Results

The survey involved software development companies, with more than 3 years working experience, whose application domain is information systems such as administrative, commerce and educational software and the data of the survey are in forthcoming part of the paper provided.

A) RQ1. Do companies use REST Architecture?

Answers to questions Q1 gave us information about the way companies develop software, i.e. which architectural style are using in their company. From the answers we could see that companies use different styles. Some have chosen one and some have chosen more than one style. The results were summed up and presented as percentage data to find the mostly used architecture. Depicted in the Figure 1, the results reported that 60% of them use REST architecture; 30% component-based applications; 20% data-centric; 10% event/driven; 40% service-oriented and 60% client/server which gives the same percentage like REST usage.

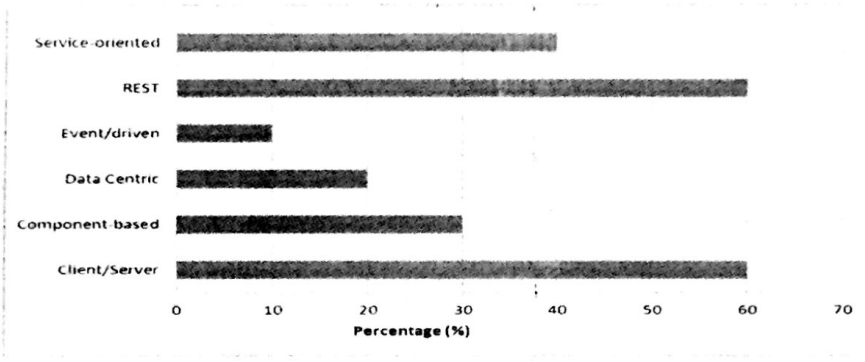

Fig. 1: Percentage Architectural Styles used by Software Companies.

B) RQ2. Does developers' experience with REST influence to the percentage of service reuse?

Since REST is relatively a new approach, our second research questions tries to see the relations among developers experience with REST and the percentage of service reuse. Based on surveys on software reuse (Imeri and Antovski, 2011) which is closely related to component based software reuse, we hypothetically concluded that the more experienced developers are, the higher the reuse is. The results obtained from the question Q2 reported that 80% of the developers had high experience with REST. 20% of the developers had low experience. The results on reuse service obtained from question Q6 reported different values (Figure 2). So, the developers' experience with REST merely has a considerable impact on service reuse.

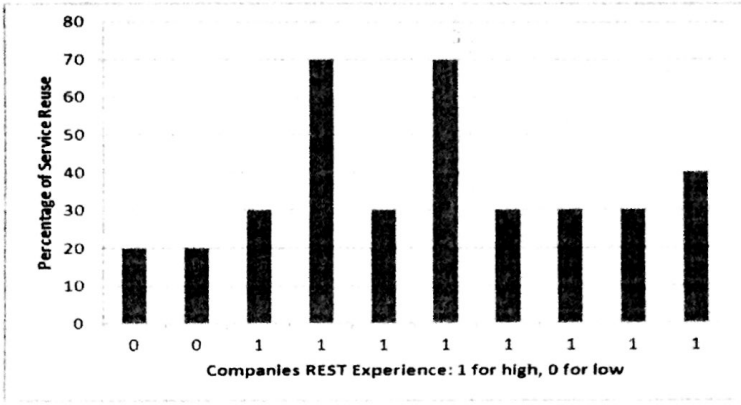

Fig. 2: REST experience vs. service reuse

C) RQ3. Which factors influence toward use of REST Architecture?

Information about the factors influencing the use of REST Architecture is from the questions Q3, Q4, Q5 and Q7 provided information.

80% of companies are using Knowledge Management and Communication (KMC), and 20% of them Design Reasoning and Decision Making (DRDM).

Regarding the development approach, 70% of companies have chosen object oriented (OO) paradigm and 30% of companies have chosen component-oriented (CO) paradigm. When a development team set up a new project, 60% of companies responded that while they develop new applications they go through old services (OS) whereas 40% start to write new program (NP).

Finally, questions regarding the software production process, 50% of responders answered that the organization produce specific software (isolated products - IO) whereas 50% product families (PF), meaning that organization develops product that evolves over time, being somehow adopted for each new project (in-house products).

Contingency tables (Table 2) were used to investigate the influence/dependence of the aforementioned factors by displaying the contingency distribution of variables. As we had 2x2 tables for the Chi-squared test, the Yates' continuity correction was applied.

We have applying the Yates' Chi Squared Test (as one of statistical test applied to sets of categorical data), since the effect of Yates' correction is to prevent overestimation of statistical significance for small data (https://en.wikipedia.org/wiki/Yates%27s_correction_for_continuity, 2015), based on the fact that the number of our responders is small.

Table 2 reports the Contingency tables of observed and expected frequencies of different variables which influenced the use of REST Architecture such as: i) the interrelation between the uses of REST Architecture vs. Companies development approach, ii) the use of REST Architecture vs. Activities supporting the architectural style, iii) the use of REST Architecture vs. Software production process and, iv) the use of REST Architecture vs. Developing process. Lastly, the significance of the difference between the two variables is assessed with a variety of statistical tests including Yates' chi-squared test.

The critical value of $S\alpha^2$ at a 5% significance level for 1 degree of freedom is seen to be 3,841 (https://en.wikipedia.org/wiki/Chi-squared_distribution). Below, are presented the calculated values of $S\alpha^2$. Almost in all our cases taken as factors that influence the use of REST Architecture, the calculated $S\alpha^2$ value were smaller than the critical value of $S\alpha^2$, i.e., independent variables except the software production process found near the critical value of $S\alpha^2$, and can be taken as a dependent variable, such a factor that influence the use of REST Architecture.

Table2. Contingency tables of factors influencing the use of REST.

		CO	OO	Total
	REST	2	4	6
Observed	Others	1	3	4
	Total	3	7	10
		CO	OO	Total
Expected	REST	1,8	4,2	6
	Others	1,2	2,8	4
	Total	3	7	10
χ^2 Yates'	0,607143			

		KMC	DRDM	Total
	REST	5	1	6
Observed	Others	2	2	4
	Total	7	3	10
		KMC	DRDM	Total
Expected	REST	4,2	1,8	6
	Others	2,8	1,2	4
	Total	7	3	10
χ^2 Yates'	1,638889			

		OS	NP	Total
	REST	3	3	6
Observed	Others	3	1	4
	Total	6	4	10
		OS	NP	Total
Expected	REST	3,6	2,4	6
	Others	2,4	1,6	4
	Total	6	4	10
χ^2 Yates'	1,100694			

		IO	PF	Total
	REST	4	2	6
Observed	Others	1	3	4
	Total	5	5	10
		IO	PF	Total
Expected	REST	3	3	6
	Others	2	2	4
	Total	5	5	10
χ^2 Yates'	3,5			

D) RQ4. Which challenges and benefits are reported from companies?

The challenges of REST consist of data permissions. Results reported that 40% of developers have had problems with data permission due to their sensibility. 60 % of developers have not had problems with permission. Results are in Figure 3 depicted.

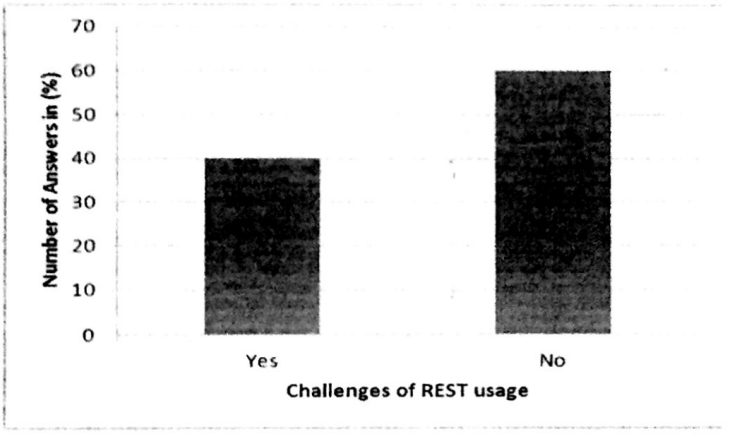

Fig. 3: Data permissions problems.

Figure 4 depicts some of the characteristics of the REST architecture. 80% of the companies reported that they use the Rest architecture because of its flexibility. 60% of the companies reported that they use the Rest architecture because of its scalability.

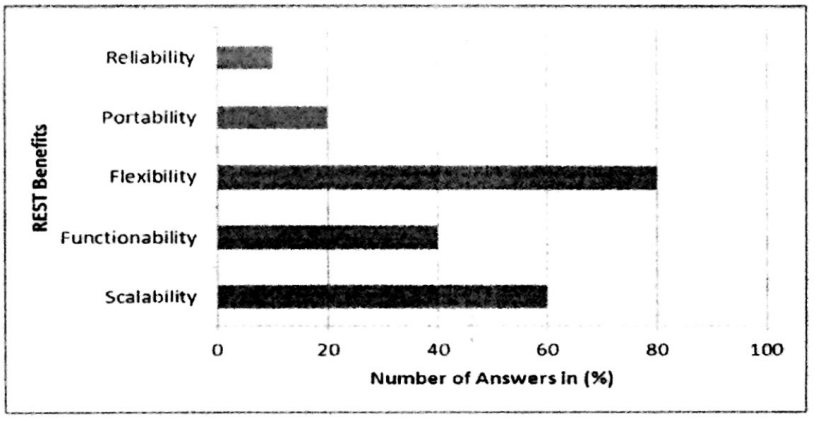

Fig.4: REST Architecture benefits.

4. CONCLUSIONS

The presented investigation addresses the companies' efficiency when meeting the customers' requests and demands for new products. In general the empirical data collected represent an overview of the current situation among companies using the REST architecture. The most evident patterns that could be identified

from our analysis, such factors that influence toward the use of REST Architecture are presented in Table 2, where variable dependencies are calculated.

The data statically (empirically) reported that the use of REST Architecture in Macedonian companies is relatively high, about 60%. Developers' experience merely has any impact on service reuse.

The factors that influence the use of REST architecture are Table 2 reported, showing that low probabilities of the tested variables dependence among factors we have chosen. A Yetes' value ($S«^2$= 1, 638889) is the calculated value among the use of REST architecture and activities supporting the architectural styles. As far as the development approach was concern, the calculated variables show that ($S«^2$= 0, 607143) is the Yetes' value among the development approach and the use of REST Architecture. In addition, a value ($S«^2$= 1, 100694) is reported as a Yetes' value while calculating the development approach such working through old services or write new program application as a activity supported by investigated companies, and lastly a Yetes' value ($S«^2$= 3, 5) based on the software production process. Consequently, all chosen variables are independent from the use of REST architecture. The process of software production taken as a variable in our case influences the use of REST because of the low probability of dependence, considering that the calculated $S«^2$ value is near the critical value of $S«^2$.

The advantages of REST consist of flexibility and productivity. Results showed that about 80% of applications that involved REST architecture showed positive impact, which is in line with the principles of Fielding (2000).

ACKNOWLEDGEMENTS

This work is done as a part of a PhD research. Its focus is to analyze the use of REST architecture and the challenges that developers face while using it. The authors owe a debt of a special gratitude to the companies for proving the data.

REFERENCES

Adamczyk P, Smith PH, Johnson RE, Hafiz M. 2011. REST and Web Services/ : In Theory and in Practice. REST: From Research to Practice. Spreinger. 35-57.

Algermissen J. 2011. Quantifying Integration Architectures. REST: From Research to Practice. Springer. 137–160.

Chi-squared distribution. Available online: https://en.wikipedia.org/wiki/Chi-squared_distribution.

Choi SW, Kim SD. 2008. A quality model for evaluating reusability of services in soa. E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing. E-Commerce and E-Services. ISBN: 978-0-7695-3340-7. 293-298.

Fielding R. 2000. Architectural Styles and the Design of Network-based Software Architectures. PhD Dissertation. 1-180.

Imeri F, Antovski L. 2012. An Analytical View on the Software Reuse. ICT Innovations 2012. Web Proceedings ISSN 1857-7288, 213.

Imeri F, Antovski L. 2013. An Empirical Study on Software Reuse in Small IT Companies in the Balkan Region. International Journal of Computers and Technologies. vol. 9, no. 2.1040–1048.

Memeti A, Selimi B, Besimi A, Cico B. 2015. A Framework for Flexible REST Services: Decoupling Authorization for Reduced Service Dependency. 4th Mediterranean Conference on Embeded Computing. IEEE Press. ISBN:978-9-9409-4364-6. 51-55.

Szepielak D. 2006. REST-Based Service Oriented Architecture for Dynamically Integrated Information Systems. PhD Symposium at ICSOC. 8–12.

Wilde E, Pautasso C. 2011. The Essence of REST Architectural Style. REST: From Research to Practice. Springer. 21–33.

Yates's correction for continuity. Available online: https://en.wikipedia.org/wiki/Yates%27s_correction_for_continuity.