# *Evaluating a Representational State Transfer (REST) Architecture*

**Bruno Costa, Paulo F. Pires**

Department of Computer Science - Federal University of Rio de Janeiro (UFRJ)
68.530 – Rio de Janeiro – RJ – Brazil

{brunocosta.dsn, paulo.f.pires}@gmail.com

***Abstract.*** *The use of Representational State Transfer (REST) as an architectural style for integrating services and applications brings several benefits, but also poses new challenges and risks. Particularly important among those risks are failures to effectively address quality attribute requirements such as security, reliability, and performance. An architecture evaluation can identify and help mitigate those risks. In this work we present guidelines to assist architecture evaluation activities in REST-based systems. These guidelines can be systematically used in conjunction with scenario-based evaluation methods to reason about design considerations and trade-offs. This work also present the results of a survey conducted with industry specialists who have performed architecture evaluations in real world REST-based systems in order to gauge the suitability and utility of the proposed guidelines.*

## 1. Introduction

Architectural decisions determine the ability of the implemented system to satisfy functional and quality attribute requirements. Since these architectural decisions affect several stages of the development process, the impact of architectural mistakes is high. Therefore, it is important to inspect the architecture to identify and mitigate any risks of the software solution not satisfying the quality attribute requirements. This inspection activity is referred to as software architecture evaluation.

Representational State Transfer (REST) is an architecture style formally defined as set constraints that aims to drive the design decisions towards improving quality attributes [Fielding 2000]. Regarded to be used in massively distributed and loosely coupled hypermedia systems, REST has been a *de facto* architecture style for web-based systems [Wilde and Pautasso 2011]. By using REST, some quality attributes of the system, such as interoperability and modifiability, are positively impacted, whereas others, such as performance and reliability can be negatively impacted. Architecture evaluators should identify places and design decisions in the architecture that influence the ability of the system to meet the quality attribute requirements.

The purpose of this work is to provide guidance for architecture evaluation activities in systems that use the REST architectural style. We discuss several design aspects and provide guidelines on important inspection points. We also discuss how the architecture can be probed by the evaluation team and propose the types of questions that should be asked during the evaluation process.

## 2. Scenario-Based Architecture Evaluation

Scenario-based evaluation methods evaluate the software architecture's suitability according to a set of scenarios of interest. A *quality attribute scenario* is a structured description of a quality attribute requirement that is unambiguous and testable [Bass et al. 2012]. Quality attribute requirements can be expressed by two types of scenarios: *(i) general quality attribute scenarios* (hereinafter *general scenarios*), which are generic and can be applied to any software system, and; *(ii) concrete quality attribute scenarios* (or *concrete scenarios*), which are specific to the particular system under consideration.

## 3. Research Protocol

The evaluation guide proposed in this research was developed following Evidence-Based Software Engineering (EBSE) techniques. EBSE aims to provide knowledge about when, how and in what context technologies, processes, methods, or tools are more appropriate for Software Engineering practices [Kitchenham 2004]. In our work, we combined two EBSE techniques: Survey and Systematic Mapping (SM). We conducted two surveys, with different purposes: (i) collect the evaluators' concerns about the architecture evaluation of REST-based solutions; (ii) collect the industry experts' empirical knowledge about REST foundations and design issues. Besides the surveys, we performed a systematic mapping in order to create a classification scheme to be used in the organization of the evaluation guide and collect the available knowledge from literature. The REST Evaluation Guide presented in Sections 4, 5, and 6 was the end result of these activities. We then sent the guide to architecture evaluators in order to assess if it had achieved its goals. The evaluation is presented in Section 7.

Following a systematic methodology [Oishi 2003; Pfleeger and Kitchenham 2001], the first survey consisted of a set of interviews with experts in architecture evaluators to gather an initial set of REST-specific architecture evaluation concerns from a practitioners' point of view. As the result of the interviews, the experts indicated that a REST architecture evaluation guide must address three main issues. In our work, we call these issues *REST Architecture Evaluation Concerns (ECs)*:

**EC1**- "Explain the architectural foundations of REST from an architecture evaluator's point of view";

**EC2**- "Discuss quality attributes and general scenarios impacted by REST principles";

**EC3**- "Discuss (in detail) how REST contributes to the quality attributes and where typical tradeoffs are".

Based on such *ECs* and also in light of the methodology proposed by Oishi (2003), and Pfleeger and Kitchenham (2001), in the next survey we interviewed several experts, practitioners, and researchers who have long been designing and studying REST solutions. Their experience about REST design was used as the first source of knowledge to respond the REST architecture evaluation concerns.

Finally, we conducted a systematic mapping based on the guidelines as included in [Petersen et al. 2008] with two objectives: carry on a comprehensive literature review on REST and to build a classification schema for the empirical knowledge gathered through interviews aligned to the literature review. We started by searching for publications from major research databases of Computer Science, such as ACM Digital Library, IEEE Xplore, SpringerLink, and ScienceDirect using keywords such as *rest*, *representational state transfer*, *quality attributes*, and *design*. The search on the databases retrieved 384 studies that were systematically analysed by their title and abstract. After this step, 42 scientific papers were selected based on a screening criterion. In addition, 8 books and 9 technical reports were selected. The result of the systematic mapping was a schema comprised of five categories: (i) Design of Services; (ii) Representation and Identification; (iii) Documentation and testing; (iv) Behaviour, and; (v) Security. With the classification scheme, the relevant papers were sorted into the scheme. These six categories were later used to classify the design questions that are part of the architecture evaluation guide and are presented in Section 6.

In the next sections we present a summary of the REST Evaluation Guide and in Section 7 we describe the evaluation of the guide made with industry specialists who have performed architecture evaluations in real world REST-based systems by using our proposed guidelines. The full version of the guide including a proof of concept that describes how to use the guidelines in the context of scenario-based evaluation method can be found in the original dissertation [Costa 2014] and [Costa et al. 2014].

## 4. Foundations of REST for Architecture Evaluation (Evaluation Concern EC1)

One of the main goals of an architecture evaluation is to identify risks to address the quality attribute requirements in software architecture. The next sections describe the foundations of REST with special attention to the impact in quality attributes. The foundations are based on interviews and literature review.

### 4.1 REST Constraints

The creator of the REST style, Roy Fielding, has described six constraints that define the REST style, each of which promotes a different set of quality attributes. REST can be described as: *REST = (C-S, S, $, U, L, CoD)*

Client-server (*C-S*) is a frequently found architectural style for network-based applications. In REST, requests are initiated by *user agents* (clients) and ultimately processed by an *origin server* (server), which provides services through a resource hierarchy. Evaluators should inspect the definition of the boundary between client and server according to cohesion and the independent evolution of each one.

The stateless constraint (*S*) describes that all information needed to understand the conversation state data between *origin servers* and *user agents* must be included in the request and response messages. The Stateless constraint enables replication of servers and hence promotes availability, scalability, and reliability; on the other hand, it may decrease performance due to the need for sending the conversational state data embedded in request and response messages.

The cache constraint (*$*) is added in order to improve performance. A cache element acts as a mediator between client and server. Evaluators should identify whether the client design will include a cache mechanism and what data elements (resources) should be cacheable. The degree to which the cache will increase network efficiency and hence performance, depends on the cache strategy. However, the use of a cache may decrease reliability in cases when the client may consume stale data from the cache.

Uniform Interface (*U*) across components is the central feature that distinguishes REST from other network-based styles. Uniform Interface is closely related to resources, identifiers, and representations. The uniform interface constraint positively impacts interoperability and discoverability.

The layered system (*L*) constraint as described by Fielding is a *multi-tier* style. Multi-tier is an architectural style that is a specialization of the client-server style where an intermediary tier acts as server to the previous tier and as client to the subsequent tier. Although simple REST services can be available on the "server-side" of client-server architectures, REST services are often found on "server-side" tiers of multi-tier applications developed using Java EE, .NET, or other implementation platform suitable for multi-tier applications.

Code-on-Demand (CoD) is an optional constraint that enables a dynamic architecture where the user agent logic can be extended by code received from the service. Interoperability may decrease since the downloaded code has to be compatible among all service consumers. Security is also a concern to prevent malicious code to reach the clients.

## 5. REST General Quality Attribute Scenarios (Evaluation Concern EC2)

A quality attribute is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders [Bass et al. 2012]. The REST Evaluation Guide provides a set of general scenarios of ten quality attributes that are important to consider when using the REST style, according to the experts we interviewed (Section 3). These general scenarios are referenced in the REST design questions (Section 6) to illustrate how they are affected by design decisions and are used by architects and evaluators to help, respectively, the specification and evaluation of concrete quality attribute scenarios. In Table 1 we describe three examples of general quality attribute scenarios for interoperability, reliability, and discoverability quality attributes.

**Table 1 REST General Scenarios for Interoperability Quality Attribute**

| Quality Attribute | Scenario |
|---|---|
| Interoperability | **I1**- A service consumer 'A' requests a resource 'R1' and receives the representation of the actual state of 'R1' in response message. |
| Reliability | **R1**- A service consumer 'A' requests a resource 'R1' in a specific version specified directly in the URI and receives the representation of the actual state of 'R1' in response message. |
| Discoverability | **D1**- A service consumer 'A' requests a resource 'R1' and receives the URIs to find resources associated to 'R1' inner the response message |

An example of an interoperability concrete scenario based on the general scenario described above is: "*A service consumer Mobile_Application requests the book resource and receives the representation of the actual state of book in response message*".

## 6. REST Design Questions That Affect Quality Attributes (Evaluation Concern EC3)

In architectures based on the REST style, several design decisions bear quality trade-offs and are driven by the quality attribute requirements. The REST Evaluation Guide provides several design questions that are particularly relevant when designing REST-based systems. Such questions are grouped into topics from the classification scheme that was created in the systematic mapping (Section 3). In each topic we present the design questions that evaluators should ask referencing to the general scenario that is impacted. In Table 2 we present five design questions, the complete list followed by the detailed description of each question and its relation with the affected quality attribute general scenario is presented in the original dissertation  and [Costa et al. 2014].

**Table 2 REST Examples of Design Questions and Quality Attribute Affected**

| Topic | Design Question | Quality Attribute Directly Affected |
|---|---|---|
| Design of Resources | What is the domain model of the application? | Interoperability |
| | Are resource representations standardized within the entire application, department, or enterprise? | Interoperability and Performance |
| Representation and identification | What format is used to represent resources? | Performance and Interoperability |
| Documenting and testing | How resources should be documented? | Testability |
| | How service consumers can perform tests in resources? | Testability |

## 7. Evaluation

In order to analyse if the guidelines achieved their goals, and collect the criticisms and suggestions from experts, the guide was sent to several architecture evaluation teams (from Software Engineering Institute - SEI, Fraunhofer IESE, among others) aiming to be used in real architecture evaluations performed in REST-based systems. After that, based on a Likert Scale, we asked them to rate five statements scaling from 1 (strongly disagree) to 5 (strongly agree), with 3 being neutral. The statements were created based on the ECs, aiming to analyse how much they were addressed in the guide. Statements 1 and 2 were based on EC1 ("Explain the architectural foundations of REST from an architecture evaluator's point of view"); the statement 3 was based on EC2 ("Discuss quality attributes and general scenarios that benefit from REST"), and statements 4 and 5 were created based on EC3 ("Discuss in detail how REST contributes to the quality attributes and where typical tradeoffs are"). The survey indicated that the evaluation concern EC1 was 84% addressed, EC2 was 66% addressed, and; EC3 was 83% addressed.

## 8. Conclusions

The main contribution of our work is the collection of general quality attribute scenarios and design questions that can assist evaluators in REST-based architecture evaluations, improving the mitigation of risks. The proposed guidelines can be used in scenario-based methods such as ATAM to help evaluators to probe architectural approaches in order to identify tradeoffs and risks to addressing quality attribute requirements. In addition, based on the evaluation of the guide (Section 7) we believe that these results indicate that the guide provides real value for software architecture evaluations performed on

REST-based systems. Our work was published [Costa et al. 2014] in the top-tier software architecture conference WICSA 2014 (The Working IEEE/IFIP Conference on Software Architecture). Furthermore, it was selected as a best paper of the conference [PPCA 2014; WICSA 2014] and we were invited to publish an extension of the paper in a special issue of the Journal of Systems and Software (JSS). At the time that this article is being written, such paper is under evaluation.

After the publication in WICSA, we have received several requests for the REST Evaluation Guide, many of them from Brazilian software houses. Thus, we developed a Web Tool in order to provide the entire guide on-line. The tool can be found at http://ubicomp.nce.ufrj.br/restguidance.

## References

Bass, L., Clements, P. and Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley.

Costa, B. (2014). Avaliando uma Arquitetura Baseada no Estilo REST. Universidade Federal do Rio de Janeiro.

Costa, B., Pires, P. F., Delicato, F. C. and Merson, P. (2014). Evaluating a Representational State Transfer (REST) Architecture - What is the impact of REST in my architecture? In *2014 IEEE/IFIP Conference on Software Architecture (WICSA),* Australia, pages 105-114.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Univ. of California, Irvine.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*,

Oishi, S. M. (2003). *How to Conduct In-Person Interviews for Surveys*. SAGE Publications.

Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering (EASE)*, Brazil, pages 68-77.

Pfleeger, S. L. and Kitchenham, B. A. (2001). Principles of survey research. *ACM SIGSOFT Software Engineering Notes*, v. 26, n. 6, page 16-18.

PPCA (2014). Prêmio de melhor artigo no WICSA. http://ppca.unb.br/index.php/97-ultimas-noticias/211-premio-de-melhor-artigo-no-wicsa, [accessed on Apr 7].

WICSA (2014). WICSA 2014 Best Paper Awards. https://web.archive.org/web/20140818151832/http://wicsa2014.org/index.php/wicsa-2014-awards/, [accessed on Apr 7].

Wilde, E. and Pautasso, C. (2011). *REST: From Research to Practice*. Springer Science & Business Media.