Towards an Improved Stakeholder Management for Software Reference Architectures

Samuil Angelov[1] and Rich Hilliard[2]

[1] Software Engineering, Fontys University of Applied Sciences, The Netherlands
s.angelov@fontys.nl
[2] Consulting software systems architect, USA
r.hilliard@computer.org

Abstract. A recent survey on software reference architectures (RA) indicates their widespread usage. Among the leading problems when designing and using RA, practitioners point to various aspects of stakeholder management (e.g., stakeholder identification, involvement). In this paper, we identify and analyze issues that lie at the basis of the problems reported in stakeholder management, with a goal to improve the state of the practice.

Keywords: Reference architecture, stakeholder management and identification.

1 Introduction

Software reference architectures have emerged as reusable resources for creating software architectures within a domain of application [1]. Definitions for the term software reference architecture (RA) and software architectures of individual systems (SA) are discussed in [1], [2]. RA are used to lower costs, improve software quality, improve communications, etc. While RA have an indisputable role in the software community, practitioners indicate facing a substantial number of problems when designing and using RA [3]. Among the leading problems reported during design are the identification and involvement of stakeholders and the dissemination of the RA to the stakeholders for usage. Problems reported during usage of RA are poor quality and lack of clear benefits for the stakeholders [3], which indicate that stakeholder management was not properly performed during the RA design. We conclude that stakeholder management in RA is a problem that needs to be investigated.

Literature does not address particular methods for stakeholder management for RA and knowledge from the design of system architectures must be applied to cases of RA. In this paper, we show that RA exhibit a number of specifics that existing work on stakeholder management in system architectures does not address. In Sections 2 and 3, we review the literature on stakeholders and stakeholder management and analyze the results from our literature overview, and build a model that we use for structuring and positioning our research. In Section 4, we analyze RA and identify their specifics compared to SA from the perspective of stakeholder management. In Section 5, we validate our findings and draw final conclusions.

P. Avgeriou and U. Zdun (Eds.): ECSA 2014, LNCS 8627, pp. 90–97, 2014.

2 Literature Review

We start our literature review with an overview of the domain of organizational science, as it sets the fundaments of the stakeholder notion. Next, we discuss the software engineering and software architecting domains.

2.1 Stakeholders in Organization Management

Definitions of "Stakeholder". Freeman [4] defines a stakeholder in an organization as "any group or individual who can affect or is affected by the achievement of the organization's objectives". Mitchell *et al*. summarize definitions found in the organizational literature until 1997 [5] and categorize them as either broad [4] or narrow (based on various "relevance properties" an entity may have for an organization). They point out that the notion of a "stake" is the leading one in defining who can be a stakeholder. Project management methods (e.g., PRINCE2, PMI) naturally focus on the relationship between entities and a project [6]. An overview of the historical development of the stakeholder concept is provided in [7].

Stakeholder Categories and Methods. In [5], a method for stakeholder identification is proposed. The method is based on three attributes: power, legitimacy and urgency. Possession of one or more of these attributes indicates a stakeholder. Classes of stakeholders are defined on the basis of combinations of the three attributes. The method is applied in a number of case studies (e.g., in [8]). Two approaches for stakeholder identification are discussed in [9]. In the first approach, named the *relationship approach*, stakeholders are identified on the basis of their relations with the organization. The types of relations which serve as a basis for the identification of stakeholders are defined to be voluntarism, mutual benefit, and community membership. In the second approach, named the *assignment approach*, the relations are based on moral considerations. The classification scheme in [10] identifies two classes of stakeholders, *actively involved* and *passively involved*. A stakeholders management process model is proposed in [7]. Stakeholder identification, defined as the first step in this process model, is based on the stakeholder categorization scheme proposed in [11], where primary (critical for the organization's survival), secondary (not critical), and public (infrastructure and legislation framework providers) types of stakeholders are defined. Vos and Achterkamp [10] argue that in addition to the stakeholder classification, stakeholder identification should be augmented with procedural guidelines that define how a stakeholder classification scheme should be applied to identify actual stakeholders and that classification schemes should be *context specific*, shifting the focus from the organization at-large to specific types of projects. In [7], [10] references to other stakeholder classification schemes are provided.

2.2 Stakeholders in Software Engineering

Definitions of "Stakeholder". Within software engineering, the focus on stakeholders pertains to requirements definition: "Requirements are the basis for every project, defining what the stakeholders (…) in a potential new system need from it" [12].

The authors of [13] provide references to stakeholder definitions. According to [14], a *project* stakeholder is "someone who gains or loses (…) as a result of that project".

In software architecting, a *software system* stakeholder is an "individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system" [15][1]. Stakeholders and their concerns drive architecture-related decisions, in particular architecture representation [16]. Rozanski and Woods [17], following IEEE Std 1471:2000 [15], define a stakeholder in a *software architecture* as "a person, group, or entity with an interest in or concern about the realization of the architecture".

Stakeholder Categories and Methods. Stakeholder categorization has been seen as the core of stakeholder identification and management in software engineering. Efforts to categorize stakeholders are reported for example in [18], [19], [17], [20]. McManus notes that "stakeholder involvement is generally context-specific; what works in one situation may not be appropriate in another" [21]. He classifies stakeholders into *primary*, *secondary*, *external*, and *extended*. Preis *et al.* propose a stakeholder classification framework based on system science techniques [22]. The stakeholders are divided into two classes: *goal oriented* and *means oriented*. A conceptual summary of classifications schemes in the literature is provided in [23].

A number of efforts defining methods for stakeholder identification exist. Sharp *et al.* [13] propose an approach for identification of stakeholders of a software system based on categorizing the interactions in a project between the stakeholders. The stakeholders are typed as *baseline*, *satellite*, *client* and *supplier* stakeholders. Baseline stakeholders (users, developers, legislators and decision-makers) are the starting point from which stakeholders of the other types are identified. In [14], an approach for the stakeholder identification and managing their involvement is proposed. The stakeholder classification scheme is based on the "onion model", where stakeholders may take different positions depending on how closely they are related to the system (at the center of the onion). MacManus also pays attention to the stakeholders' involvement [21]. He notes that stakeholder involvement is based on the central goal and project objectives. The work in [23] focuses on stakeholder identification in the development projects for inter-organizational systems.

2.3 Stakeholders in Reference Architectures

In [1], the stakeholders are classified based on the number of RA receiving organizations, their role in the RA design process, and the type of organization they represent. In a case study made of five Dutch municipalities [24], Galster *et al.* discuss two stakeholder categories: the customers and software vendors who are applying RA. Martínez-Fernández *et al.* [25] consider RA in a specific context: a software consultancy company defining for their clients RA and define two types of stakeholders: the RA team (software architects and architecture developers) and "concrete software architecture teams" (the application builders). Cloutier *et al.* [26] mention stakeholders of RA to be ranging from engineers to business managers and customers. Notably, one of their conclusions is that further research on the stakeholders of RA is needed.

[1] The stakeholder concept is treated in a broader sense in [8] and [9].

3 An Approach for RA Stakeholder Analysis

From the literature overview, it can be observed that the term "stakes" presents several ambiguities and that authors search for substitute terminology (e.g., "affect", "interest"). We use the definition from [15]: "A *stakeholder* of a *P* is an individual, team, organization, or classes thereof, with an *interest* in *P*". With respect to the issues at stake, in organizational sciences the broad notion of "organization" (or "projects" within it) is seen as the basic issue. In software engineering the issues at stakes lie in the software system [16], [27], or in the project for its creation [14], [18]. The precise issue at stake, however, is often weakly defined. In architecting, the focus lies on identifying *all stakeholders* whose concerns will influence the architecture [16]. Most efforts focus on providing one or a combination of categorization schemes that facilitate stakeholder identification. Specifically for RA, the stakeholder categories are defined only for specific contexts [24], [28] or in an informal manner [1].

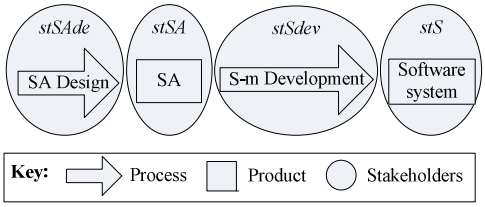

Fig. 1. Areas of stakes in software system design and development

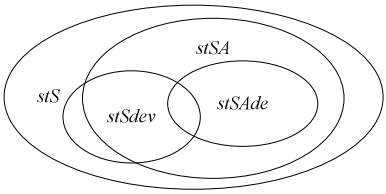

Fig. 2. Relationship between the stakeholder sets [Venn diagram]

Based on our observations from the literature review, we define a model in which we ascribe to each major sub-process and product of the notional software development cycle an "area" with its stakeholders (see Fig. 1). The "stakeholder areas" cover all the entities with an interest in the specific process or product. The "system stakeholders" (*stS*) include stakeholders with developmental, technological, business, etc. influences (as defined in [16]). The "development process stakeholders" (*stSdev*) are the stakeholders of the development process. The "SA design stakeholders" (*stSAde*) are those with concerns about the design process (architects, project leaders, project managers, etc.). Obviously, an entity may be a stakeholder in several of the stakeholder areas (see Fig. 2). The model in Fig. 1 is inspired by the approach of [5] and our observation that the issues at stake need to be well-defined – we focus on the elements of a project, i.e., main processes and products.

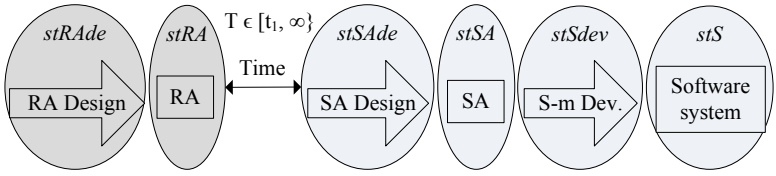

Fig. 3. Areas of stakes in the case of RA, usage decoupled (S_2)

The process of elaborating a reference architecture is conceptually comparable to the process of SA elaboration. Therefore, we extend the initial model with the corresponding RA sub-process and product (see Fig. 3, time gap is explained in Section 4). The stakeholders of the RA design process (*stRAde*) and the RA (*stRA*) are the new elements in the scenario and are the focus of our research. We also investigate if stakeholder management in the other elements changes when a RA is used.

4 Specifics of RA in Stakeholder Management

In this section, we identify the specifics (S_i) of stakeholder management for RA contrasted with SA. The set of specifics was defined by studying and analyzing publications on RA (e.g., [2], [3], [25], [26], [28], [29]).

Looking at the types of purposes and goals of a RA design discussed in [3], [26], [28], (e.g., "decrease development costs", "speed up projects", "standardization"), we observe that they are predominantly stemming from *stSAde* and *stSdev*. In the case of SA, the purposes and goals of architecting are defined by a balance of design, development, and system drivers. These concerns are also relevant for a RA design, but they are not defining the purpose and goals of a RA design.

S_1 (New concerns, New dominant drivers): The purpose of a RA is typically targeted towards *stSAde* and *stSdev*.

Consequences: The incentives for getting involved in a RA design project for requirements elicitation and architecture evaluation may be lower for the *stSA* who are neither *stSAde* nor *stSdev*. At the same time, the *stSAde* and SA-interested part of the *stSdev* become the main beneficiaries of the effort. For the *stSAde*, this implies changes in their roles, as they become *consumers* of the effort, while previously their role was in *producing* a SA. Furthermore, the new purposes require the involvement of new (not typical for SA) *stRA*. For example, standardization purposes can lead to the introduction of *standardization organizations* [29]. Cost reduction and project efficiency concerns may arise from the interests of higher management levels (e.g., *program managers, governance bodies*) and *enterprise architects*.

S_2 (Usage Decoupled): The RA design and its usage may be separated with a substantial period of time, often unknown at design-time (see Fig. 3). In certain scenarios, a RA may be defined without any specific planned usages.

Consequences: The decoupling of usage from design means that concerns that need to be reflected in the RA will come from stakeholders not seeing directly the results of their inputs. This allows us to distinguish *actual* (where RA usage is planned) and *potential stRA* (where the RA usage is not planned). Involvement and retention of the *potential stRA* may be difficult due to insufficient motivation.

S_3 (Multiple Applications): RA have a scope, i.e., they are intended to be applied multiple times in different cases (see Fig. 4, where we indicate the possible multi-organization application scenario with dashed lines).

Consequences: The multiple application contexts (potentially, across multiple organizations) of a RA mean that the stakeholders for the application of a RA can differ per application case. In the situations of very high (or unlimited) number of application contexts, across multiple organizations, not all stakeholders can be involved

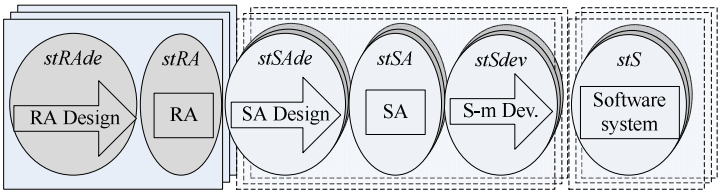

Fig. 4. Multiple applications of RA (S_3) and design organizations (S_4)

and choices need to be made. To indicate the multiple application contexts, we refer to the stakeholders of an eventual system *i* targeted by a RA as stS_i, $stSdev_i$, $stSA_i$, $stSAde_i$.

S_4 (Cross-organizational design): The design of RA can be of cross-organizational nature, i.e., a RA may be designed by several, independent organizations (see Fig. 4).

Consequences: A cross-organizational effort puts the organizations involved in a RA design in a complex communication and management situation. Different organizations may have different policies, rules, and strategies, leading potentially to conflicts with the concerns of other stakeholders and the selection of the design organizations is challenging. To tackle these problems a *coordination (management) body* of some type may be introduced.

S_5 (Long-life): A RA is an investment. It is intended to be applied over a period of time in which the initial investment will pay out. A RA has a life beyond the life of individual systems.

Consequences: The longer life of RA means that the *stRAde* need to remain active after the initial design and ensure RA evolution for the time the RA is intended to be maintained (leading to a more complicated management of the *stRAde* involvement).

S_6 (Abstract): Because RA are to be applied in multiple contexts, they are typically defined at a higher level of abstraction, where specific choices are deferred.

Consequences: RA may be harder to understand, use, and communicate due to this increased abstractness. This may lead to an inability to (properly) apply a RA by its users, frustration, criticism, etc. This may be the cause for stakeholders abandoning or not (fully) engaging in RA application projects.

The specifics identified by us have not been addressed in the software engineering and architecting literature on stakeholder management reviewed in Sections 2.2 and 2.3. Based on the consequences of the specifics, we have made a number of observations on the *stRA* and *stRAde*, which we summarize in Table 1.

Table 1. Observations on the stakeholders of RA

Stakeholders of RA	Source
A *stRAde* is a *stRA*.	definition
The $stSA_i$ and $stSAde_i$ of an eventual system *i* targeted by the RA are *potential stRA*.	S_1, S_2, S_3
For RA with efficiency goals, higher management roles (program managers, enterprise architects) are *stRA*. For RA with standardization goals, standardization bodies are *stRA*.	S_1
For *stRAde* from multiple organizations, a coordination body may be a *stRAde*.	S_4
stRAde may be $stSAde_i$ in order to obtain feedback on the RA and evolve it.	S_5

5 Validation and Conclusions

As initial steps in establishing the validity of the specifics identified, we have studied two well-documented RA: RASDS [30] and ESDS RA [31]. We sought evidence to demonstrate the existence of the specifics. Both the Reference Architecture for Space Data Systems (RASDS) [30] and the Earth Science Data System Reference Architecture ESDS [31] exemplify all specifics, except for S_4 in ESDS due to the single-organization application scope of the RA. Next, we validated the specifics for completeness. Our approach is to study "framework papers" that define the fundaments of RA and which were not used in our initial analysis. We have considered [26] and [32] for this purpose, where [26] focuses on the RA purposes, contexts, and processes and [32] on the RA elements making them complimentary in covering the RA landscape. As a result from this step, we have identified the omission of the "evolution" of RA, which has led to the addition of S_5 to our list of specifics.

Therefore we conclude that the six specifics of RA identified do exist. Their critical role for stakeholder management can be traced back in their relation to the problems reported in [3]. Existing literature on stakeholder management in system architecting does not provide direct solutions for these specifics. We conclude that a dedicated method for stakeholder management for RA is desired. The results presented in this paper are a first step towards such a method.

References

1. Angelov, S., Grefen, P., Greefhorst, D.: A Framework for Analysis and Design of Software Reference Architectures. Inf. and Soft. Technology 54(4), 417–431 (2012)
2. Angelov, S., Trienekens, J.J.M., Grefen, P.: Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case. In: Morrison, R., Balasubramaniam, D., Falkner, K. (eds.) ECSA 2008. LNCS, vol. 5292, pp. 225–240. Springer, Heidelberg (2008)
3. Angelov, S., Trienekens, J., Kusters, R.: Software Reference Architectures - Exploring Their Usage and Design in Practice. In: Drira, K. (ed.) ECSA 2013. LNCS, vol. 7957, pp. 17–24. Springer, Heidelberg (2013)
4. Freeman, R.E.: Strategic Management: A Stakeholder Approach. Pitman, Boston (1984)
5. Mitchell, R., Agle, B., Wood, D.: Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts. Academy of Management Review 22(4), 853–886 (1997)
6. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide), 4th ed. Project Management Institute Inc., Pennsylvania (2008)
7. Preble, J.: Toward a Comprehensive Model of Stakeholder Management. Business and Society Review 10(4), 407–431 (2005)
8. Parent, M., Deephouse, D.: A Case Study of Stakeholder Identification and Prioritization by Managers. Journal of Business Ethics 75, 1–23 (2007)
9. Cappelen, A.: Two Approaches to Stakeholder Identification. Ethics and Economics 2(2), 1–9 (2004)
10. Vos, J., Achterkamp, M.: Stakeholder Identification in Innovation Projects - Going Beyond Classification. European J. of Innovation Management 9(2), 161–177 (2006)

11. Clarkson, M.: A Stakeholder Framework for Analyzing and Evaluating Corporate Social Performance. Academy of Management Review 20, 65–91 (1995)
12. Hull, E., Jackson, K., Dick, J.: Requirements Engineering, 3rd edn. Springer (2011)
13. Sharp, H., Finkelstein, A., Galal, G.: Stakeholder Identification in the Requirements Engineering Process. In: Proceedings of the Tenth International Workshop on Database and Expert Systems Applications. IEEE Computer Society (1999)
14. Alexander, I., Robertson, S.: Understanding Project Sociology by Modeling Stakeholders. IEEE Software 21(1), 23–27 (2004)
15. IEEE: Recommended Practice for Architectural Description of Software-Intensive Systems.Std 1471-2000. IEEE (2000)
16. ISO/IEC/IEEE: Systems and software engineering —Architecture description. 42010, ISO/IEC/IEEE (2011)
17. Rozanski, N., Woods, E.: Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional (2005)
18. Cotterell, M., Hughes, B.: Software Project Management. Int. Thomson Publishing (1995)
19. Newman, W., Lamming, M.: Interactive System Design. Addison-Wesley (1995)
20. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley (2002)
21. McManus, J.: A Stakeholder Perspective within Software Engineering Projects. In: Proceedings of the 2004 IEEE International I Engineering Management Conference, vol. 2, pp. 880–884. IEEE (2004)
22. Preiss, O., Wegmann, A.: Stakeholder Discovery and Classification Based on Systems Science Principles. In: Proceedings of the Second Asia-Pacific Conference on Quality Software, pp. 194–198. IEEE (2001)
23. Ballejos, L., Montagna, J.: Method for Stakeholder Identification in Interorganizational Environments. Requirements Eng. 13, 281–297 (2008)
24. Galster, M., Avgeriou, P., Tofan, D.: Constraints for the Design of Variability-Intensive Service-Oriented Reference Architectures – An Industrial Case Study. Information and Software Technology 55(2), 428–441 (2013)
25. Martínez-Fernández, S., Ameller, D., Ayala, C., Franch, X., Terradellas, X.: Conducting Empirical Studies on Reference Architectures in IT Consulting Firms. UPC (2012)
26. Cloutier, R., et al.: The Concept of Reference Architectures. Systems Engineering 13(1), 14–27 (2010)
27. Conger, S.: The New Software Engineering. International Thomson Publishing (1994)
28. Muller, G.: A Reference Architecture Primer (2008)
29. Angelov, S., Grefen, P., Greefhorst, D.: A Classification of Software Reference Architectures: Analyzing their Success and Effectiveness. In: Joint Working IEEE/IFIP Conference on Software Architecture, 2009 & European Conference on Software Architecture, WICSA/ECSA 2009, September 14-17, pp. 141–150. IEEE, Cambridge (2009)
30. CCSDS: CCSDS Recommended Practice - Reference Architecture for Space Data Systems. CCSDS, NASA (2008)
31. ESDS Reference Architecture Working Group: ESDS Reference Architecture for the Decadal Survey Era. NASA ESDS Reference Architecture v1.0, NASA (2011)
32. Nakagawa, E., Oquendo, F., Becker, M.: RAModel: A Reference Model for Reference Architectures. In: SPLC 2011 Proceedings of the 15th International Software Product Line Conference, vol. 2(28). IEEE Computer Society (2012)