Empirically-grounded Reference Architectures: A Proposal

Matthias Galster
University of Groningen, The Netherlands
m.r.galster@rug.nl

Paris Avgeriou
University of Groningen, The Netherlands
paris@cs.rug.nl

ABSTRACT

A reference architecture describes core elements of the software architecture for systems that stem from the same domain. A reference architecture ensures interoperability of systems through standardization. It also facilitates the instantiation of new concrete architectures. However, we currently lack procedures for systematically designing reference architectures that are empirically-grounded. Being empirically-grounded would increase the validity and reusability of a reference architecture. We therefore present an approach which helps systematically design reference architectures. Our approach consists of six steps performed by the software architect and domain experts. It helps design reference architectures either from scratch, or based on existing architecture artifacts. We also illustrate how our approach could be applied to the design of two existing reference architectures found in literature.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design – *methodologies, representation;* D.2.11 [**Software Engineering**]: Software Architectures – *data abstraction, languages;* K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software development.*

General Terms

Management, Documentation, Design.

Keywords

Software architecture, reference architecture, design process, empirically-grounded.

1. INTRODUCTION

Software architectures play a significant role for software quality. One way to systematically reuse architecture knowledge when developing new software systems is through reference architectures (RA) [1]. RA's capture the essence of the architecture of a collection of similar systems. These systems usually belong to a certain technology domain, application domain, or problem domain. The purpose of a RA is to facilitate the development of concrete architectures for new systems, to help with the evolution of a set of systems that stem from the same RA, or to ensure standardization and interoperability of different systems. Following Bass et al., we differentiate reference model and RA [2]. A reference model is a decomposition of a problem into parts that cooperatively solve the problem. A RA on the other hand is a reference model mapped onto software elements that cooperatively implement the functionality of the reference model.

A RA is usually designed by capturing the essentials of existing architectures of a group of products, and by taking into account future needs and variability. This involves three main problems: First, RA's are usually designed in an unsystematic manner; traceable and repeatable steps to design RA are missing [3]. Second, sometimes a RA needs to be designed from scratch, without the ability to mine existing software architectures. Third, there is usually no solid evidence about the validity of a RA.

To address the aforementioned problems, we attempt to develop an approach to design RA's a) in a systematic manner, either from scratch or based on existing architecture artifacts, and b) so that the resulting RA's are empirically-grounded and thus valid to be applied in a broad range of situations.

By empirically-grounded we mean that evidence for the relevance and applicability of the building blocks (e.g., stakeholders, concerns, models) of a RA must exist. In detail, "empirically-grounded" includes two aspects [4]: 1) "Empirical foundation" means that the RA must be based on sufficient real-life phenomena and proven principles [1]. This means, the RA a) must address real stakeholder interests, b) should be based on concepts proven in practice, and c) the building blocks of the RA must stem from the problem domain. 2) "Empirical validity" means that the RA needs to be evaluated to ensure its applicability and validity.

The advantage of empirically-grounded RA's is that they ensure a solid foundation for instantiated architectures. They would not be specific to a particular organization, and be reusable for a broad range of projects. Thus, the contribution of this paper is an approach that helps architects and domain engineers design empirically-grounded RA's.

In Section 2 of this paper we discuss related work. Section 3 introduces our approach to design empirically-grounded RA's. In Section 4 we discuss how our approach could be applied to the design of two RA's published in literature. In Section 5 we discuss limitations and conclude the paper in Section 6.

2. RELATED WORK

It was only recently that the notion of RA has been brought to attention in the systems engineering community [5]. RA's often appear in organizations or domains where the multiplicity of applications requires life-cycle support in a distributed open world. In software engineering (SE), RA’s occur at different levels of abstraction. For example, generic RA’s for service-oriented architectures, such as IBM’s foundation architecture [6] exist. Also, more specific RA’s can be found, e.g., for e-contracting [7] or web browsers [8]. It has been argued that RA’s are directly used in the product line context [1]. However, RA’s can be categorized so that product line architectures are one type of reference architecture [9]. Moreover, product line architectures are less abstract than RA’, but more abstract than concrete architectures [10]. In general, product line architectures represent a group of systems that are part of a product line, produced by a single organization, whereas RA’s represent the spectrum of systems in a domain [8].

Cloutier et al. presented a high-level model for RA development. This model covers the collection of information from existing systems and the evolution of the RA [5]. High-level guidelines are also provided by Pohl et al. to design RA as part of domain design during product line engineering [1]. Also in the context of software product lines, Bayer et al. proposed PuLSE-DSSA [11]. In PuLSE-DSSA, RA’s are created by capturing knowledge from existing architectures. Angelov et al. recently studied RA’s in SE and developed a classification which could act as a starting point for designing RA’s [12]. However, this “framework for reference architectures” focuses on a classification of RA’s, rather than giving guidelines for designing a RA. ProSA-RA is a process to design RA’s outside the product line domain [13]. However, ProSA-RA focuses on aspect-oriented systems and neglects empirical validity of the RA. Templates (but no guidelines) for RA’s are often provided in industrial organizations.

Despite these efforts, in SE, systematic approaches to define RA’s are missing [3], or guidelines are not documented [14]. Moreover, most proposed RA’s lack an empirical foundation which would make them truly generalizable. In domains outside SE, methods for reference modeling have been introduced to support the creation of reference models, processes, etc. For example, Karow et al. [4] introduced the empirical construction of reference process models in public administrations. Ahlemann and Gastl presented a more general approach for constructing empirically-grounded reference models for business processes [14]. We believe that these approaches can act as a starting point for developing methods that help construct valid RA’s in SE.

3. EMPIRICALLY-GROUNDED RA

Our approach is based on two inputs: 1) Existing RA’s and RA’s in practice, and 2) literature on RA’s. The approach consists of six steps (see following sub-sections):

Step 1: Decision on type of RA	
Step 2: Selection of design strategy	
Step 3: Empirical acquisition of data	Ensure “empirical foundation”
Step 4: Construction of RA	
Step 5: Enabling RA w/ variability	
Step 6: Evaluation of the RA	Ensure “empirical validity”

A feedback from Step 6 allows the refinement of a RA. The RA will go through a maturing process in several iterations. Also, please note that instantiation of a RA and evolution of a RA are out of the scope of our paper.

Throughout this section, we use a project from e-government in the Netherlands to illustrate our approach. The goal of this RA would be to support the implementation of a law about subsidizing health care costs of citizens (WMO law). This RA would be used in local municipalities.

3.1 Step 1: Decision on Type of RA

Deciding on the type for the RA is important as it determines what information needs to be collected for constructing the RA (Step 3). Furthermore, the RA type affects the construction of the RA (Step 4). The selection of the type is primarily driven by the purpose of the RA. Therefore, we group RA types based on two “dimensions”: their usage context (Vogel et al. [9]), and the characterization framework proposed by Angelov et al. [12]. This means, a RA type is a combination of these two “dimensions”.

Based on the usage context, three items are differentiated [9]. *Platform-specific* RA’s (e.g., a service-based RA could be based on a platform that uses web services, WSDL and UDDI, or a platform that uses CORBA and an OMG trader) are specific to a platform. *Industry-specific* RA’s (e.g., AUTOSAR) focus on the needs of organizations in a specific industry. *Industry-cross-cutting* RA’s cover more than one industry (e.g., IBM’s foundation architecture used in e-commerce and e-government).

The characterization framework proposed by Angelov et al. [12], defines five types of RA’s, based on *why* and *when* they are created and *where*. With respect to *when*, “classical” RA’s are created after experience from systems has been accumulated, while “preliminary” RA’s are created before a system fully implements the RA. With respect to *why*, “standardization” RA’s focus on system interoperability in concrete architectures and between systems built based on the same RA. “Facilitation” RA’s aim at providing guidelines for the design of systems. With regard to *where*, RA’s can be created for a single organization, or for multiple organizations. This leads to the following five categories: 1) Classical, standardization architectures to be implemented in multiple organizations; 2) classical, standardization architectures to be implemented in a single organization; 3) classical, facilitation reference architectures for multiple organizations designed by a software organization in cooperation with user organizations; 4) classical, facilitation architectures designed to be implemented in a single organization; 5) preliminary, facilitation architectures designed to be implemented in multiple organizations. Details about these categories can be found in [12].

Example: The goals of a WMO RA would be twofold: a) Support variability between municipalities with the possibility to implement municipality-specific variants; b) at the same time ensure interoperability between municipalities, and municipalities and the national government. Thus, we would construct an industry-specific RA (e-government). Also, the WMO RA is classified as a classical facilitation RA as we aim at providing guidelines for designing instantiated systems in municipalities.

3.2 Step 2: Selection of Design Strategy

In most cases, RA’s are not developed from scratch, but based on previous project experience [15] and existing architectures. However, sometimes, the RA is built without any previous system. Thus, we differentiate two design strategies: a) Design the RA from scratch; b) design RA from existing architecture artifacts. This complies with [10] where RA’s can be “research-driven” or “practice-driven”. “Research-driven” means that the design of the RA is inspired by existing research effort, providing a “futuristic” view of a class of systems (e.g., the e-contracting

RA presented in [7]). “Practice-driven” means that RA’s are defined when sufficient knowledge in a domain exists to propose best-practices [10]. Consequently, building RA’s from scratch has a prescriptive character whereas building RA’s from existing artifacts has a descriptive nature [10].

The decision on the design strategy is affected by the RA type. If a preliminary RA type is chosen, the RA will be designed from scratch. If a classical type is chosen, it is designed based on existing architecture artifacts. The design strategy impacts what data sources are needed for the empirically-grounded RA (Step 3).

Example: The WMO RA is not built from scratch but based on existing architecture artifacts. Various municipalities in the Netherlands already have IT infrastructures that support the implementation of the WMO law in some form.

3.3 Step 3: Empirical Acquisition of Data

3.3.1 Identify data sources

To collect empirically-grounded data, the data sources have to be identified. The RA type and design strategy selected in the previous steps impact the data sources. For example, for classical RA’s, existing documentation can be used, whereas preliminary RA’s would need to draw on other types of documentation and people. Platform-specific RA’s would require detailed platform information (e.g., about CORBA) whereas industry-specific RA’s would require detailed knowledge on business processes in a domain. Nakagawa et al. recommend people (e.g., customers, users, researchers), systems (including documentation and source code), and publications / documents (e.g., technical reports, white papers) as data sources [13].

Example: The consequences of the RA type and the construction strategy on selecting data sources for the WMO RA are as follows: First, to create an industry-specific RA, domain studies help collect sufficient empirical data, including process information within municipalities. Second, the standardization aspect requires a consensus between different organizations. This includes organizations that will use the RA, organizations that will communicate with the RA, as well as organizations that oversee the operations of municipalities. For gathering data required for constructing the WMO RA, we could therefore identify the following data sources. First, the business process descriptions of the WMO law as implemented in municipalities can be searched. This source describes how the law is actually executed in municipalities. In a preliminary study, seven municipalities and their processes of implementing the WMO law were chosen for data collection [16]. Second, the IT infrastructure of municipalities provides insights into commonalities and variations between the technical implementation of the WMO law. Third, consulting with software vendors provides a more technical as well as logical view of what needs to be present in the RA. Fourth, mining standards, regulations and existing RA for local e-government provide help scope the WMO RA.

3.3.2 Record architecture data

The data for empirically-grounded RA’s needs to be collected using empirical data collection techniques, e.g., interviews, questionnaires, or document analysis. The information to collect depends on what information should be included in the RA. This depends on the level of detail of the RA, its intended size, the RA type and the design strategy. When the RA is not built from scratch, existing architectures can be mined and generalized using qualitative analysis techniques, e.g., content analysis. A domain-specific modeling language might be used to model the data. Responsibilities, resources, etc. might be recorded. This step also includes the definition of stakeholders that a) will be using the RA to instantiate concrete architectures, and b) will be using instantiated architectures. Furthermore, architecture concerns (interests of stakeholders in the RA; technical or business) and architecture-significant requirements have to be identified.

Example: For a WMO RA, we could record the data in several ways. Interviews and document analysis can collect business process information and information about the current IT infrastructure, variations and commonalities between municipalities, and potential technical solutions. For the WMO RA, security, privacy and variability are architecture significant requirements. Major stakeholders are IT managers in municipalities, WMO case managers in municipalities, citizens, the national government, and software vendors.

3.4 Step 4: Construction of RA

Based on the empirical data collected in the previous steps, the RA can be constructed. The construction includes the documentation of the RA in a description. Here, we follow ISO/IEC 42010 [17]. This means, the RA is described in terms of architecture views. Also, as our approach utilizes ISO/IEC 42010, viewpoint and view selection depends on concrete concerns. For example, based on the abstraction level of the RA, Beneken suggests functional, logical and technical views [15]. Sometimes, several technical RA views can adhere to one logical view [18]. Components that are defined in the technical view might refine components of the logical view. Similarly, according to Cloutier et al., a RA should address a technical view, business view and the customer context (i.e., requirements), which partially overlap [5]. Moreover, while a functional view is relevant at early stages of development, technical views are used when detailed architectures are needed [15]. Concerns and views would depend on the RA type as well as the data collected in Step 3.

The basic structure of the RA can be derived from collected information (Step 3) and consists of the common building blocks of the architecture (common stakeholders, views, model kinds) according to ISO/IEC 42010. Building blocks identified based on the data collected in Step 3 should be compared to identify common solution parts. The common elements can become parts of the RA. Specific architecture elements can be hidden and annotated with specific data for adaptation during the instantiation of the RA (see Step 5).

Furthermore, special attention should be given to quality attributes (QA). Starting point for this task is the RA type and the collected data. These help identify architecture significant requirements and key drivers. Here, patterns and tactics should be added to the RA in order to achieve QA and key drivers. To find appropriate patterns and tactics, we suggest a search of the patterns and tactics literature, so that designers of the RA do not have to develop new solutions which are not proven.

Example: The description of a WMO RA would describe common modules for assessing the eligibility of a citizen for governmental support. Furthermore, due to existing RA’s for e-government in local municipalities, interfaces with external entities can be described. The RA would have a process view and a technical view. Business processes from the process view are mapped onto implementation structures in the technical view.

3.5 Step 5: Enabling RA with Variability

To allow the creation of instances of the RA, variability has to be enabled. Based on the empirical data collected in previous steps, it is possible to derive variation points in the architecture for instantiation. There are three possible ways to enable variability: 1) Annotation of RA; 2) variability models; 3) variability views. Depending on the case, any of these three approaches or a combination can be used. Annotation would simply mark any elements within any architecture model with information about variability (e.g., using attributes or rules). As the RA description follows ISO/IEC 42010, we also can define specific variability models. This means, a variability model could be used in any RA view. Furthermore, variability views could be used, so that variability-related concerns can be framed by specific viewpoints.

Example: In the WMO RA, annotations of model elements in the process view can be included for differing age requirements to be eligible for government services, or varying method of assessing the physical condition of a citizen. The technical view can include specific variability views to describe how variability impacts QA.

3.6 Step 6: Evaluation of the RA

The quality of the RA can be tested empirically in a specific project situation. Its value can be assessed based on the utility it provides within a project. In detail, for assessing the quality of the RA, we differentiate two aspects: a) Correctness and the utility of the RA itself, and b) the support for efficient adaptation and instantiation. The quality of the RA with regard to correctness and utility depends on whether it can be transformed into a meaningful organization-specific architecture. However, the quality of its adaptation support relies on the usefulness of the annotations with attributes and rules, the variability model, and variability views.

If the RA is not built from scratch, evaluation is less critical as the RA should be using proven architecture concepts anyway. This means, the validation of concepts in RA's is often derived from preceding architectures [5]. In that case, the focus of RA evaluation should be put on evaluating the support for efficient instantiation. In cases where innovative technologies or applications are introduced and in which RA's are built from scratch, it is usually challenging to have sufficient proof for the validity of a RA [5]. However, in our approach, the data used for constructing the RA has been empirically collected to mitigate this problem. In addition, reference implementations, and prototyping and an incremental approach can be used for validation and proof. Moreover, the evaluation of the RA can be done by mapping existing widely used architectures on the RA and see how / if they comply.

For both construction strategies, checklists can be applied to evaluate the quality of a RA. Similar to the evaluation of concrete architectures, a list of questions that guide reviewers can be used. These checklists can be extended with criteria for good RA's, including adaptability, understandability, accessibility within organization, and the inclusion of key issues of specific domains. However, due to the special nature of RA's, specific evaluation methods are needed [10]. A RA evaluation method has been proposed by Angelov et al. [10].

Example: In the case of a WMO RA, evaluation would be done in two ways. First, RA documentation would be sent to domain experts for review. Second, feedback from domain experts in practice would be collected through online discussion forums.

4. USAGE OF APPROACH

In this section we apply our approach to two RA's published in literature and show how our approach could be applied to design these RA's. The questions we are interested in are "How is our process reflected in the design of existing reference architectures?" and "What problems could be avoided in the design of RA?".

4.1 Web Brower RA

Grosskurth and Godfrey proposed a RA for web browsers [8]. We summarize the steps of our process as it would apply to this RA. Step 1: The RA type was a classical standardization RA used in multiple organizations, and industry-cross-cutting. Step 2: The construction strategy was building the RA from existing architecture artifacts, namely two existing mature web browsers. Step 3: The acquisition of data included the identification of data sources (documentation of two web browsers and the software itself; no people). The data collection happened through document analysis. No architecture significant requirements were identified. Steps 4 and 5: The RA was constructed by aggregating common elements between the two analyzed web browsers into one architecture model. After constructing the RA, no annotation took place and no variability model was included. Step 6: The evaluation of the RA was done by checking the reference architecture against the architecture of two other web browsers. This means, the web browser RA was evaluated through mapping existing widely used architectures on the RA. This happened in a straight-forward manner and therefore showed the applicability and completeness of the web browser RA. As this shows, not all steps of our approach were applied when designing the web browser RA. Overall, only the steps of construction (Step 3) and evaluation (Step 5) of our process were explicitly used in the design of the web browser RA, and discussed in [8].

4.2 RA for Visual Mining Applications

Nakagawa et al. presented a RA for the visual mining domain [13]. We chose this RA as it followed a process during design that is similar to ours. Again, we provide a summary of how our steps would have applied to the design of this RA. Step 1: The RA type was a classical facilitation RA, used in multiple organizations, being platform-specific (aspect-oriented RA). Step 2: The construction strategy was building the RA from scratch. Step 3.1: Data sources included people (visual mining researchers, tool developers), visual mining tools, and publications (books, technical reports, scientific papers). Step 3.2: Data types and inputs were identified. Step 4: Based on architecture-relevant requirements, architecture views were created, using patterns, such as MVC. Step 5: No annotations or a variability model were included to facilitate the creation of instances. Step 6: Even though validation of RA is part of the process used to design the RA for the visual mining domain, no evaluation was performed.

4.3 Summary

These two examples show that our approach would have been applicable to the development of existing RA's. However, to fully follow our approach, more activities than the ones performed to develop the RA's would have been necessary. Furthermore, the examples showed that our approach is partially reflected in how these two RA's were designed.

The problems we encounter in the two RA's are the lack of empirical validity, the lack of annotations in the RA, and the lack of support for RA evolution. Thus, our approach would avoid the lack of traceable design and instantiations of concrete architectures.

5. LIMITATIONS

Even though we analyzed RA's in practice, we did not develop a new, fully-fledged RA. On the other hand, we have illustrated our approach using a real project from the e-government domain.

Furthermore, some of the steps in our approach are generic or abstract. For example, we do not prescribe what data collection techniques to use or what sources of data to mine. This is because many aspects depend on the type of RA as well as the construction strategy chosen (see Section 3.1 and Section 3.2).

Finally, designing empirically-grounded RA's might include additional effort that might not be feasible in certain contexts (e.g., for small-scale RA's or very generic RA's).

6. CONCLUSIONS AND FUTURE WORK

We presented an approach to design empirically-grounded RA's in SE. Empirically-grounded RA's are built upon evidence for the relevance and applicability of the different elements of a RA. To develop the approach, we drew on existing RA's and on processes to design reference models and architectures.

Future work includes the application of our approach to the design of a RA in the e-government domain. As shown in some examples throughout the paper, our approach has shown to be useful in this context. Furthermore, we will investigate how to adapt the proposed approach for service-oriented RA's.

7. ACKNOWLEDGMENTS

This research has been partially sponsored by NWO SaS-LeG, contract no. 638.000.000.07N07.

8. REFERENCES

[1] Pohl, K., Boeckle, G. and van der Linden, F. 2005. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer Verlag, Berlin / Heidelberg.

[2] Bass, L., Clements, P. and Kazman, R. 2003. *Software Architecture in Practice*. Addison-Wesley, Boston, MA.

[3] Eklund, U., Askerdal, O., Granholm, J., Alminger, A. and Axelsson, J. 2005. Experience of Introducing Reference Architectures in the Development of Automotive Electronic Systems. *ACM SIGSOFT Software Engineering Notes*, 30, 4, 1-6.

[4] Karow, M., Pfeiffer, D. and Raeckers, M. 2008. Empirical-based Construction of Reference Models in Public Administrations. In *Proceedings of the Multikonferenz Wirtschaftsinformatik* (Munich, Germany). Gito-Verlag, 1613-1624.

[5] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E. and Bone, M. 2010. The Concept of Reference Architectures. *Systems Engineering*, 13, 1, 14-27.

[6] High, R., Kinder, S. and Graham, S. 2005. *IBM's SOA Foundation - An Architectural Introduction and Overview*. Technical Report. IBM.

[7] Angelov, S. and Grefen, P. 2008. An E-contracting Reference Architecture. *Journal of Systems and Software*, 81, 11, 1816-1844.

[8] Grosskurth, A. and Godfrey, M. 2005. A Reference Architecture for Web Browsers. In *Proceedings of the International Conference on Software Maintenance* (Budapest, Hungary). IEEE Computer Society, 661-664.

[9] Vogel, O., Arnold, I., Chughtai, A., Ihler, E., Kehrer, T., Mehlig, U. and Zdun, U. 2009. *Software-Architektur - Grundlagen - Konzepte - Praxis*. Spektrum Akademischer Verlag, Berlin / Heidelberg.

[10] Angelov, S., Trienekens, J. and Grefen, P. 2008. Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case. In *Proceedings of the Second European Conference on Software Architecture* (Paphos, Cyprus). Springer, 225-240.

[11] Bayer, J., Ganesan, D., Girard, J.-F., Knodel, J., Kolb, R. and Schmid, K. 2003. *Definition of Reference Architectures Based on Existing Systems*. Technical Report. Fraunhofer Institute for Experimental Software Engineering.

[12] Angelov, S., Grefen, P. and Greefhorst, D. 2009. A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness. In *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA)* (Cambridge, UK). IEEE Computer Society, 141-150.

[13] Nakagawa, E. Y., Martins, R. M., Felizardo, K. R. and Maldodano, J. C. 2009. Towards a Process to Design Aspect-oriented Reference Architectures. In *Proceedings of the XXXV Latin American Informatics Conference (CLEI´2009)* (Pelotas, Brazil), 1-10.

[14] Ahlemann, F. and Gastl, H. 2007. Process Model for an Empirically Grounded Reference Model Construction. In *Reference Modeling for Business Systems Analysis*, P. Fettke and P. Loos, Eds. IGI Global, Hershey, PA, 77-97.

[15] Beneken, G. 2006. Referenzarchitekturen. In *Handbuch der Softwarearchitektur*, R. Reussner and W. Hasselbring, Eds. dpunkt Verlag, Heidelberg, 357-370.

[16] Bouma, T. D. *Process Analysis and Requirement Specification of Software as Service for WMO Provision Applications at Dutch Municipalities*. Master thesis, University of Groningen, Groningen, The Netherlands, 2010.

[17] ISO/IEC. 2010 *Systems and Software Engineering - Architecture Description*. ISO/IEC 42010, Geneva, Switzerland.

[18] Stricker, V., Lauenroth, K., Corte, P., Gittler, F., Panfilis, S. D. and Pohl, K. 2010. Creating a Reference Architecture for Service-based Systems - A Pattern-based Approach. In *Towards the Future Internet*, G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller and T. Zahariadis, Eds. IOS Press, Amsterdam, 149-160.