Systematic Architectural Decision Management

A process-based Approach

Ana Dragomir, Horst Lichter, Tiberiu Budau
RWTH Aachen University, Research Group Software Construction
Aachen, Germany
{adragomir, lichter}@swc.rwth-aachen.de
tiberiu.budau@rwth-aachen.de

***Abstract*— The documentation of architecture and design decisions lies at the backbone of building a comprehensive architectural knowledge basis within a company. As a consequence, a plethora of supporting frameworks has been lately proposed by the research community. The existing frameworks focus on capturing the rationale that lies behind a certain decision, but less on sustaining the collaborative process that architects employ when making decisions. In this paper, we propose an innovative architectural decision making process that sustains the collaboration of architects, the timely notification of involved stakeholders, the inclusion of feedback cycles to improve the overall quality of the architecting process and a tag-based traceability system that leverages informal learning. The analysis of the current state of the practice in the industry has been conducted within various workshops and interviews with our industry cooperation partner – the software provider of one of the biggest insurance trusts worldwide. Based on these results, we have identified various improvement potentials that are still not addressed by existing research in the field. (*Abstract*)**

Keywords—architecture decisions; process-based; feedback cycles; collaboration; architectural decisions making process;

I. INTRODUCTION

According to the generally accepted Lehman's Laws of Increasing Complexity and Decreasing Quality, unless proper measures are taken, over time the complexity of a system increases and its quality tends to diminish [1]. To ensure a sustainable evolution of software systems and avoid increasing complexity and declining quality, important decisions need to be taken at an architectural level. The set of architectural-level decisions taken during a software system's lifecycle represent the rationale of its architecture. As early as the 1990s it was already acknowledged that, along with elements and their form, rationale is a crucial facet of software architecture [2]. Rationale has since gained ever more importance, architecture being often described as the set of architectural decisions that were made through the lifecycle of a software product ([3], [4]).

Although many approaches to document architectural decisions have been proposed, we have recognized based on interviews and workshops undergone with architects and process managers of Generali Deutschland Informatik Services (GDIS) (the IT provider of one the biggest insurance service groups worldwide) that important practice-relevant requirements are still not covered by the existing state of the art. We have identified four important improvement potentials that are not thoroughly addressed by the current state of the art and concretized them in a new, innovative concept.

The remainder of this paper is organized as follows: in Section II we give an overview of the related work. Section III presents our research goals. Section IV summarizes the current state of the practice in the industry, as opposed to the state of the art in academia (covered in Section II) and highlights the identified improvement potentials. Section V presents our solution concept. Section VI concludes and gives an outlook of our work.

II. RELATED WORK

Kruchten has emphasized the importance of documenting decisions, rationale being present in all the views of the well-known "4+1 Model" [5]. Later on, he has developed an ontology of architectural decisions [6], being the first that described them as first class entities. In [6], a preliminary, yet still actual, analysis of the typology of architectural decisions, their attributes, and relations to one another and to other external artifacts is presented.

The notion of rationale has been later refined in the well-known ISO/IEC/IEEE 42010 [7] standard, to which we also adhere: "architecture rationale records explanation, justification or reasoning about architecture decisions that have been made". The rationale for a decision can include the basis for a decision, alternatives and trade-offs considered, potential consequences of the decision and citations to sources of additional information" [7].

While Kruchten posed the question "*what* should be considered when documenting architecture decisions?" later research focused more on the "*how*" aspect. As remarked in [8], there are currently three well-established approaches for documenting decisions: decision templates, architectural annotations and decision models. Our concept focuses on a life-cycle process for architectural decisions that should be built on top of a well-defined decision model. In comparison to a template- or annotations-based approach, this may add increased overhead, however, the benefits outweigh the costs, as automation and sound tool support can pave the way to several advantages and to enforcement of quality attributes – at the architectural decisions level ([21]).

A plethora of related approaches have been developed or are being developed by the scientific community in this direction. In [9] a "rationale-based architecture model" has been introduced. Central to this approach is the traceability of

Software Architecture (WICSA), 2014 IEEE/IFIP Conference on , vol., no., pp.255,258, 7-11 April 2014
doi: 10.1109/WICSA.2014.39

the architecture rationale – embodied in decisions – to architecture elements such as requirements or software components. Traceability plays also in [10] a central role: the decisions are documented using a template intertwined with elements extracted from a self-developed requirements model (traceability to requirements) and from ACME [11] component descriptions. Other approaches further focus on creating traceability links between decisions, the static architecture view (described using ADLs [12] or reconstructed from the source code [13]) and eventually the corresponding source code. However, we have observed that in practice traceability is often not considered and, when asked, architects always prefer very flexible solutions that do not assume complicated integrations of various heterogeneous repositories (code repositories, requirements repositories, etc.). As a consequence, in contrast to the above mentioned approaches, our solution offers a flexible tag-based mechanism that allows the easy definition of new traceability types whenever necessary.

In [14], three inter-decision relationship types are proposed: "restricts", "associate/implies", and "refines". However, an elaborate industry evaluation is missing. According to our industry case study, the architects could not always envision the usability of the inter-decision relationships and militated also here for an extension of the tag-based approach, which we have developed for the traceability feature.

One of the most generic, yet comprehensive and complete, architectural decision models is defined in [8] and basically includes three sections: general information, alternatives and their argumentation and traceability to other artifacts and AD-relationships. Our solution goes one step further by including more advanced collaboration and feedback features as well as tag-based traceability and inter-decision relationships. In [8] a decision process including an innovative decision refinement cycle ("approve/challenged") is also proposed. However, this cycle does not include feedback from other stakeholders, nor does it document feedback explicitly. Furthermore, the authors do not specifically state whether this refinement is done collaboratively or in isolation, whether the architects are peers in a network or they answer to a higher authority.

Tool-based approaches (e.g., [12], [15] and [16]) focus also on the pure documentation or reuse of decisions, offering minimal or no support for the collaboration of the involved architects. The ArchiTech tool described in [17] aims to offer more support for the architects, by giving them recommendations that best suit their quality requirements and constraints. The "facts" on which the recommendations are based need to be modeled by a domain engineer, this leading to a rather centralistic approach. By contrast, we plan to offer support by enhancing the collaboration of architects, offer hints based on past experience and include feedback cycles.

Last but not least, to the best of our knowledge, the explicit notification of interested stakeholders regarding the decision's change of status has not been mentioned in any research paper yet published.

III. GOALS

Considering the problems stated in the first section and keeping into account that the current state of the art fails to address them, our main goal is to develop a concept that efficiently sustains the decision making process of software architects. To achieve this, we have pursued the following sub-goals:

- G1: Analyze the existing decision making process that architects are following in practice
- G2: Identify improvements of the previously identified process
- G3: Develop an enhanced process that addresses the previously identified improvements

IV. CURRENT STATE OF THE PRACTICE

The state of the practice described in this section is based on our experiences with GDIS. Being CMMI Level 3 certified, GDIS implements the "Decision Analysis and Resolution" [18] process area and documents the wide majority of decisions taken within various projects.

We have recorded the state of the practice in three main steps: (1) an initial meeting with three GDIS employees to get an overview of the decision making process in the company, (2) a thorough analysis of the decision artifacts of two large-scale projects (45000 and 9000 IT person days respectively), and (3) 12 follow-up discussions with further employees where the results of step 2 were presented and discussed upon – thus gaining new insights.

GDIS uses a template to document architectural decisions. The template includes many of the attributes proposed in the literature (e.g. [6], [8], [15], [19], [22], etc.), such as: title, problem description and motivation, complete list of identified alternatives, selected alternative, invoked stakeholders, derived requirements and related decisions. Unlike most of the state of the art (exception making very few, e.g., [20]), GDIS went one step further to prioritize the criteria considered for a given decision and then quantify the selected alternatives based on these. However, within the context of the same project, the *quantification of identified criteria occurred differently*: using either numbers from 1 to 10, or "--", "-","0","+","++", etc. Even more, the analyzed projects were using different *project-specific locations* (CVS repositories, Wiki pages, etc.) to store the decisions and *often apply the template very differently or even change it*.

Traceability links to other artifacts are not documented, but architects considered that including these might be useful.

Next, we have analyzed through elaborate discussions with managers and architects how the current decision making process occurs. The main six steps that constitute it are: S1: A **problem/issue at the architectural level is identified**, and a decision needs to be made how to solve it. S2: The **architects identify and discuss the various alternatives** that could be considered. S3: The **architects quantify all alternatives**, using a set of relevant criteria. S4: The **architects determine the best alternative** based on the quantification performed in

Software Architecture (WICSA), 2014 IEEE/IFIP Conference on , vol., no., pp.255,258, 7-11 April 2014
doi: 10.1109/WICSA.2014.39

the previous step. S5: The **architects discuss the implications of the chosen alternative**. S6: Only now, that is after the decision has been analyzed and made, **one of the previously involved architects is designated to document it in the project's template instantiation.** When asked, the architects mentioned that *they rarely analyze past decisions for their impact nor do they use decisions taken in other projects to learn from* and accused the bad decisions retrievability as the main factor that leads to this. *Feedback cycles are rarely present* in the decision making process. Furthermore, they mentioned that the lack of proper *notification mechanisms* leads to the fact that employees often oversee the outcome of some decisions that they might have been previously interested in.

At any time during the steps S2 to S5, if the issue proves itself to be beyond the expertise, knowledge or authority of the involved architects the decision can be escalated to other, higher-level, decision forums defined within the company.

Based on the performed analysis we have identified the following improvement potentials: (**I1**) The decisions should be documented in a central location based on a well-defined domain model imposed across all projects; the decision making process should be tool-supported and include various, optional feedback loops; (**I2**) Decisions should be easily retrievable; (**I3**) It should be possible to flexibly add traceability links to other artefacts; (**I4**) Notification mechanisms to communicate the outcome of decisions should be employed.

V. Proposed Approach

In our opinion, the collaborative process ([23]) that architects employ when making decisions should be sustained by proper tool-support based on a common data repository and used across projects. To achieve this, the decision making process should be first defined and described. We consider, that this is possible by defining and describing the lifecycle of a decision, which corresponds to the automaton shown in Figure 1. We have chosen an automaton over a flow-chart based model, in order to abstract away from the actors. This is because we deliberately wanted to create a process generic enough to be implemented as a workflow in companies with varying guidelines and policies (I1).

The automaton contains six states that we have grouped according to the phases of the well-known Deming cycle in order to underline that continuous improvement should be one of the major goals of the decision making process.

During the **"initializing" state**, the architect initiating the decision specifies the decision's title and formulates its problem description. Once this is completed the available alternatives need to be identified, and so the decision moves in the **"in progress" state**. In this state, the initiating architect can involve other architects by notifying them and request their collaboration. The invited architects can then collaboratively work on the set of alternatives by deleting, editing or adding new ones. To avoid long waiting times, the refinement is time-framed, the time until which the refinement is possible being communicated to the architects during the notification. Once the time for collaborative refinement expires, the notified architects are no longer allowed to add new alternatives, unless a new collaborative refinement session is initiated and they are again invited to contribute. The reason for this is to encourage architects to propose their alternatives in due time and thus to encourage collaboration. After the decision has been collaboratively refined, it moves to the **"in debate" state**. In this state the involved architects can select the relevant criteria to be considered and they can initially quantify the alternatives according to them. Proactive feedback types, such as criteria estimation hints based on the past decision making history can be offered here in order to ease the quantification effort. Furthermore, if additional feedback from other architects is needed, or if the decision needs to be escalated, then it can be transferred for a given period of time in the **"in review" state,** during which the newly notified collaborators can make amendments, i.e., by adding or deleting criteria or modifying the scores of existing ones. If the reviewers consider that the proposed alternatives need changes, the decision can return in the "in progress" state. Otherwise, when the set time expires, the decision returns automatically in the **"in debate" state**, from where it can be subject to another review/escalation phase (i.e. returns in the "in review" state) or finally closed, ending in the "**decided" state.** In this state, the decision, although made and thus immutable, can still be commented upon, by anyone who wishes to do so. These comments will not change, in any way, the decision's final state, their sole purpose being to allow the retrospective knowledge sharing.

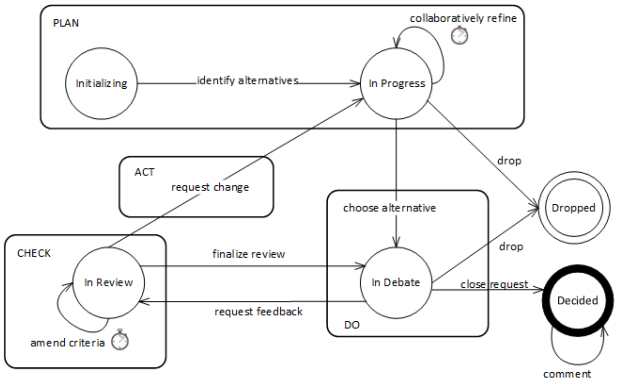

Figure 1. Lifecycle of a decision

Finally, while in the "in progress" or in the "in debate" state, the decision can be dropped, i.e., it is decided that the decision must not be further considered nor implemented anymore (e.g. because the upper management decided so). In this case, the decision ends in the **"dropped" state** where it is saved in its current form together with the reason why it was discarded.

Regardless of its state, traceability links to other artifacts (such as risks, requirements, etc.) can be defined or the decision can be involved in relationships with other decisions. Also, at any time during the process, the architects can search for related decisions that could help them improve the current one, by discovering new alternatives, understanding past rationale, etc. To make decisions easily retrievable but also to easily add new traceability types and inter-decisions relations,

Software Architecture (WICSA), 2014 IEEE/IFIP Conference on , vol., no., pp.255,258, 7-11 April 2014
doi: 10.1109/WICSA.2014.39

we propose the use of semantic tags, ordered by semantic tag types (I2, I3). The main advantage of our tag-based approach is that new types of traceability (risk traceability, software components traceability, etc.) and inter-decision relations types (implication, refinement, etc.) can be added very easily in the system, by simply defining new semantic tag types. These tags and tag-types can then be easily used to define complex search queries that can potentially improve the decisions retrievability to a great extent. Thus, complex queries of the type: "search all decisions of the PROJECT *PRJ11* that affect the COMPONENT *CRM* and violate the COMPANY-GUIDELINE *CG10*" or "retrieve all the alternatives that expose the RISK of *running over-budget*" are now becoming possible. Note that words in capital letters are tag-types while words in italics are concrete tags.

Last but not least, once the decision has been decided upon or dropped, notification mechanisms should be employed in order to communicate its outcome to interested stakeholders. At the decision level one should be able to specify what stakeholders or groups of stakeholders should be notified when a change occurs. Furthermore, stakeholders should be able to also subscribe themselves to various topics, using the same tag mechanism specified above (e.g.: "notify me when decisions in the context of PROJECT *PRJ11* are made"). Again, note that words in capital letters are tag- or inter-decision relationship types while words in italics are concrete tags (I4).

VI. Conclusions and Future Work

In this paper we propose a new concept that capitalizes on the current state of the and further focuses on enhancing the collaboration of involved architects and boosting informal learning by including feedback cycles and tag-based traceability and inter-decision relationships. Throughout our work, we have closely collaborated with our industry partner. We claim that this has conferred our concept a very pronounced "industry touch".

In our future work, we plan to evaluate our concept in the industry. Furthermore, we will analyze what metrics could be used to determine the quality of the decision making process. According to discussions with our industry partner, metrics in this field are highly desirable.

Acknowledgment

The authors want to thank our partners from Generali Deutschland Informatik Services for their continuous support throughout our work.

References

[1] M. M. Lehman, "Programs, life cycles, and laws of software evolution", IEEE, vol. 68(9), pp. 1060–1076, 1980.

[2] D.E. Perry, A.L. Wolf, "Foundations for the study of software architecture", ACM SIGSOFT Software Engineering Notes, vol. 17(4), pp. 40–52, 1992.

[3] S. Zörner, "Softwarearchitecture documentation and communication: Effectively and Reproducible Document Designs, Decisions and Solutions" (in German: "Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten"), Hanser, München, 2012.

[4] J. Bosch, "Software architecture : the next step", European Workshop on Software Architecture, vol. 3047, pp. 194-199, 2004

[5] P. Kruchten, "Architectural blueprints - the 4+1 view model of software architecture", IEEE Software, pp. 42–50, November 1995

[6] P. Kruchten, "An ontology of architectural design decisions in software-intensive systems", the 2nd Groningen Workshop on Software Variability Management, pp. 1–8, 2004

[7] ISO, Systems and Software Engineering – Architecture Description. ISO/IEC/IEEE 42010, pp. 1–46, May 2011

[8] U. van Heesch, P. Avgeriou, and R. Hilliard, "A documentation framework for architecture decisions", Journal of Systems and Software, 85(4), pp. 795–820, April 2012

[9] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning", Journal of Systems and Software, vol. 80(6), pp. 918–934, June 2007

[10] D. Dermeval, J. Pimentel, C. Silva, J. Castro, E. Santos, G. Guedes, M. Lucena, and A. Finkelstein, "STREAM-ADD - Supporting the documentation of architectural design decisions in an architecture derivation process", IEEE 36th Annual Computer Software and Applications Conference, pp. 602–611, July 2012

[11] D. Garlan, R.T. Monroe, and D. Wile. Acme: Architectural Description of Component-Based Systems. Foundations of Component-Based Systems, Gary T. Leavens and Murali Sitaraman (eds), Cambridge University Press, 2000

[12] A. Jansen, J. van der Ven, P. Avgeriou and D. K. Hammer, "Tool support for architectural decisions", the Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA'07), pp. 44-54, 2007

[13] G. Buchgeher and R. Weinreich, "Automatic tracing of decisions to architecture and implementation", the Ninth Working IEEE/IFIP Conference on Software Architecture, IEEE Computer Society, pp. 46-55, 2011

[14] B. Michalik, J. Nawrocki, "Towards decision centric repository of architectural knowledge", 4th Central and East European Conference on Software Engineering Techniques, CEE-SET 2009, pp. 3–15, 2009

[15] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas, "A web-based tool for managing architectural design decisions", ACM SIGSOFT Software Engineering Notes, vol. 31(5), 2006

[16] M. A. Babar, I. Gorton, "A tool for managing software architecture knowledge", the second ICSE Workshop on Sharing and Reusing Architectural Knowledge, Rationale, and Design Intent, pp. 11-18, 2007

[17] D. Ameller, C. P. Ayala, J. Cabot and X. Franch, "Non-functional requirements in architectural decision making", IEEE Software 30(2), pp. 61-67, 2013

[18] CMMI Overview: http://www.sei.cmu.edu/searchresults.cfm

[19] J. Tyree, A. Akerman, "Architecture decisions: demystifying architecture", IEEE Software, vol. 22(2), pp. 19–27, 2005

[20] A. Herrmann, B. Paech, "Learning from documented decisions" (in German: "Lernen aus dokumentierten Architektur-Entscheidungen"), Softwaretechnik-Trends, vol. 26(4), 2006

[21] R. C. de Boer, H. van Vliet, "Experiences with Semantic Wikis for Architectural Knowledge Management", 9th Working IEEE/IFIP Conference on Software Architecture, pp. 32-41, 2011

[22] M. Shahin, P. Liang, and M.R. Khayyambashi, "Architectural Design Decision: existing Models and Tools", Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 293-296, 2009

[23] R. Farenhorst, P. Lago, and H. van Vliet, "Effective Tool Support for Architectural Knowledge Sharing", Proceedings of 5th European Conference on Software Architecture, pp. 123-138, 2007

Software Architecture (WICSA), 2014 IEEE/IFIP Conference on , vol., no., pp.255,258, 7-11 April 2014
doi: 10.1109/WICSA.2014.39