Embedded Design Rationale in Software Architecture

Rafael Capilla
Universidad Rey Juan Carlos
Madrid, Spain
rafael.capilla@urjc.es

Abstract

The increasing interest to consider design decisions and its rationale as an inherent part of the software architecture development process has led to a number of research works that promote the capturing and use of the architecturally significant decisions. Hence, the stakeholders can keep track of the reasons of changes. This paper explores a variety of initiatives from previous works and advocates for an "embedded use of design rationale" in software architecting activities with tool support.

1. Introduction

The shift perceived by software architects since 2004 that attempts to include architectural design decisions as part of the architecting activity is becoming more and more important for researchers and professional software architects. As Bosch stated in 2004 [3], a software architecture should be seen "*as the result of a set of design decisions rather than a set of components and connectors*". In this context, software architects need to assume new roles acting as decision makers rather than mere software modelers. Because the reasoning process that guides the design phase often remains implicit in the architect's mind, the design decisions and its underlying rationale are often relegated and never recorded. Thus, this tacit process and the design decisions made are overlooked and never made explicit neither documented. Also, the importance of rationale management for software engineering has been stated in [9], and software maintenance can benefit from the decisions captured, in particular in those cases where an understanding of a system is required due to, for instance, an unavailability of the experts or system's creators (e.g.: staff turnover). As mentioned in [16] "*long-term benefits and reduced maintenance costs should motivate users to capture the design rationale, particularly in successive iterations of the system as it evolves*" and such benefits should be supported by empirical evidence [10]. In addition, knowledge sharing becomes more important to communicate, not only the final architecture, but also the reasons that led to a particular design solution, as well as the design alternatives considered and evaluated. Therefore, the communication between the stakeholders can be enhanced if we are able to explain the reasons of the decisions made. This paper summarizes current state of the art related to design decisions and promotes the use of "*embedded design rationale*" in order to incorporate design decisions as a usual asset in architecting.

2. Background

2.1 Representing Design Knowledge

Researchers in the software architecture field have proposed templates of attributes for capturing the most relevant information about key design decisions [7, 15, 20], including the semantics of the links that connect decisions to requirements and architectures as well as the dependencies between decisions. Also, the use of ontologies for organizing and visualizing different types of decisions is stated in [1, 15, 18]; as well as models able to describe the relationships between the traditional software artifacts and the design decisions [7, 21]. As a result, the inclusion of design decisions and design rationale explicitly linked to architectures is currently being discussed in the upcoming ISO/IEC 42010 standard (Draft WD4, Jan 2009) for architectural description of software-intensive systems.

2.2 Processes

A set of complementary processes enrich now the traditional modeling task. Making decisions, evaluating design alternatives, searching for similar solutions, or sharing decisions to others, are examples that belong to the reasoning activity [8, 17, 21].

2.3 Prototype Research Tools

Some of the approaches mentioned before have been implemented in a variety of prototype research tools (e.g.: Archium [13], AREL [19], PAKME [2], ADDSS [6], The Knowledge Architect [14], SEURAT [5], EAGLE [11], The Architectural Knowledge Wiki [22]) that highlight the importance of design rationale. These tools provide capturing and documentation facilities of design decisions and the definition of relationships between artifacts of the design phase, while supporting the evolution of decisions and architectures or knowledge sharing capabilities are implemented only in some of them.

2.4 Design Rationale in OSS and Commercial Tools

From the analysis of six architectural modeling tools described in [12], only Compendium has a first class notion of design decision, in addition to Archium [13], which is specifically designed for this goal. Today, some tools have been modernized as well as new ones have appeared in the arena. In general, open source and commercial tools lack of adequate support for design decisions. For instance, the ArchStudio 4 is an open source tool that defines architecture as "*the set of principal design decisions about a system*", which is intentionally a broad definition that means that every project has a set of design decisions that are considered principal by its stakeholders. Current ArchStudio capabilities could only document principal design decision using XML schemas and embodied in xADL descriptions. Rationale Software Architect (RSA) and Rational Software Modeler (RSN) tools from IBM enable designers to produce UML 2.0 architectural models, but specific support for design decisions explaining the architecture products is missing. The OMG SysML plug-in only provides a stereotyped UML comment (<<rationale>>) to describe the justification for decisions and the requirements, design, and other decisions. Hence, there is a need to strengthen the facilities designers demand to incorporate design decisions connected to modeling diagrams.

3. Embedded Design Rationale

This section claims the use of an "**embedded design rationale**" approach in software architecture as an integrated but complementary architecture view or feature, similar to the one stated in [16]. At present we foresee two main ways to implement the idea that design decisions can be captured concurrently with modeling activities. From the analysis of the prototype tools mentioned in Section 2.3, we observed two principal ways to address the same problem.

1. **Decisions are captured just AFTER modeling (DCaM)**: Some insight from the creators of two research tools show the following. What it worked in Archium [14] was to define first a component and connector (C&C) view of the architecture and after to specify, rank, and capture the decisions. AREL [19] does not mandate the sequence of events for capturing decisions and these are captured when the design is relatively stable. This approach is closer to what architects currently do using modeling tools, as they start modeling the architectures prior to capturing the decisions. Also, the relationships between architectures and decisions are defined as the architecture evolves.
2. **Decisions are captured just BEFORE modeling (DCbM)**: This approach is used in tools like PAKME [2], ADDSS [7], and The Knowledge Architect [15], which is more close to the reasoning process in which decisions are captures and documented explicitly just before the architecture is modeled, often with an external modeling tool. The links between decisions are established when decisions are captured and links to architectures are established after the architecture is upload into the tool repository. Also, the links between requirements and decisions are defined before an architecture product is modeled.

In the **DCaM** approach, the links defined between parts of the architecture might serve also to define the links between decisions, while in the **DCbM** approach, the links connecting decisions define the relationships between the architectural artifacts that are modeled afterwards. The former seems to make decisions quicker and unconsciously and embedding the design rationale inside the design choices selected, while the latter seems to follow a more elaborated and slower process aimed to simulate explicitly the reasoning activity that happens in the architect's mind. In both cases, the design rationale is linked to architectures and the relationships between decisions are entangled or connected both to requirements and architectures. DCaM tools can also capture the decisions following a DCbM approach, but the process is still not fully clear. Tools like EAGLE [11] and The Architectural Knowledge Wiki [22] are more collaborative in the sense that they focus more on knowledge sharing, and design decisions are captured in text templates and organized in a decision tree which can be filtered out according to project specific needs. Some of these tools capture knowledge in documents while others try to inject decisions into design models, code, or deployment artifacts. SEURAT [5] offers sharing and collaborative decision-making using an Eclipse plug-in rather than a web interface, and captures the rationale as structured arguments that can be linked to system requirements and desired qualities. The knowledge capturing process of these three tools can be classified as DCbM.

3.1 Advantages and Drawbacks

At present, the competition between both strategies shows no clear winner, and the following lessons can be extracted.

3.1.1 Embedded design rationale: One of the main advantages of the approaches and tools mentioned is that we can consider the design decisions and its rationale almost integrated within the architecting process. Decisions are now explicitly documented and fully integrated with other software artifacts.

3.1.2 Replication of the reasoning activity: The difficulty to replicate the reasoning activity is still the major barrier that impedes a wider use of these tools in software architecture. Architects need to perceive the advantages of capturing and using such architectural knowledge (e.g.: explain the reasoning activity). Other reasons such as: organizational, political or even fear may block the introduction of these complementary tasks in addition to modeling.

3.1.3 Iterative in nature: The architecting and the decision-making processes are iterative in nature, but this fact is not in contradiction with the classification made as both approaches follow an iterative process.

3.1.4 Intrusiveness: The intrusiveness of some tools is bigger than others, as some of them change the usual way architects do architecting. Archium and AREL seem to be less intrusive than other tools as designers start modelling as usual, but the reasoning activity of tools like ADDSS seems more logical from the DCbM perspective.

3.1.5 Maintenance: Maintaining design decisions clearly benefits further maintenance operations on behalf of the links that connect decisions to requirements and architectures as well as between decisions themselves. The relationships between decisions and other software artefacts are used to bridge the traditional gap between requirements and designs or even code.

3.1.6 Evolution: Tools like AREL and ADDSS offer some support for evolution of decisions but also ADDSS shows nicely the evolution of architectures, while commercial modeling tools cannot shown in the same layout the history of the architecture. Tools that follow the DCbM approach seem to support better the evolution of architectures, but this lack resides more on the graphical capabilities rather than on the underlying model supporting the tool. When capturing decisions, the evolution of the architecture should be always aligned with the evolution of decisions.

3.2 Research Topics to Support Embedded Design Rationale

From the analysis of DCaM and DCbM strategies, we have identified some issues that the tools examined could incorporate into more mature versions.

- **(a) Strategies for capturing rationale:** Even that capturing decisions before modeling seems more logical from the reasoning point of view, we believe that architects still prefer to start modeling their diagrams. To support this strategy we advocate improving the capturing process of current DCaM tools. For instance, once a design pattern is selected, the tool could automatically fill the description of the decision to: (i) accelerate the knowledge capturing task, (ii) reduce the intrusiveness of capturing such knowledge. Tools belonging to DCbM approach should provide better integration with modeling tasks to avoid having two different tools: one for modeling and another for capturing decisions. Finally, a combination of codification and personalization strategies [7] is useful to delimit the amount of information about decisions we want to capture and make the process more agile and flexible.
- **(b) Rationale in the process model:** Describing the rationale about a process model (e.g.: the design process) and annotate such decisions in the process model [4] seems useful to explain a process (e.g.: RUP) to designers who are learning from the decisions made.
- **(c) Evolution and maintainability:** Because the design process is iterative in nature, it is necessary that DCaM tools provide a way to show the evolution of the architecture. Capturing the decisions that evolve offer the architect the possibility to revert from past decisions. Also, maintaining the links of fine-grained decisions (e.g.: a variation point in the architecture) should be enhanced as the dependency network between decisions should be manageable.
- **(d) Visualization of design decisions:** Architects easily see the resultant architecture, but visualizing the relationships of the knowledge [18] embodied with architectures is not so easy and often not well implemented in the tools. Therefore, it would be useful to visualize graphically the decisions and their relationships.
- **(e) Integration with code:** Almost all the tools examined do not provide a way to embed the

design rationale inside code descriptions, and they limit this knowledge to the architecture level.

4. Summary

The adoption of an "*embedded design rationale*" approach in current research and commercial tools for capturing design decisions results key. Capturing AK inside typical architecting activities is not easy, as software architects perceive that they need to change the way in which architectures are built. The frontier between the DCaM and DCbM is diffuse, just because decisions are always reasoned and made before modeling. From our experience with ADDSS [7] and from the analysis of other tools, we suggest reducing the intrusiveness of the tools capturing decisions. Hence, we need a research agenda to decide about the best knowledge capturing strategies as well as the steps of the decision-making process and how this can be integrated with current architecting activities.

Acknowledgements

I would like to thank Klaas-Jan Stol for his review and useful suggestions.

References

[1] Akerman, A., Tyree, J. Using ontology to support development of software architectures. IBM System Journal 45(4), 813-826, 2006.

[2] Babar, M.A., Gorton, I. and Kitchenham, B. A Framework for Supporting Architecture Knowledge and Rationale Management. in: A.H. Dutoit, R. McCall, I. Mistrik, and B. Paech, (Eds.), Rationale Management in Software Engineering, Springer, pp. 237-254, 2006.

[3] Bosch, J. Software Architecture: The Next Step, Proceedings of the 1st European Workshop on Software Architecture (EWSA 2004), Springer-Verlag, LNCS 3047, pp. 194-199, 2004.

[4] Burge, J. Brown, D. C., "Integrating Design Rationale with a Process Model", Workshop on Design Process Modelling, Artificial Intelligence in Design '02, Cambridge, UK, 2002.

[5] Burge, J. E. Software Engineering Using Rationale, http://www.users.muohio.edu/burgeje/SEURAT/Downloads.html, 2008.

[6] Capilla, R., Nava, F., Pérez, S., Dueñas, J.C. A Web-based Tool for Managing Architectural Design Decisions (SHARK'066), ACM SIGDOFT Software Engineering Notes 31(5), 2006.

[7] Capilla, R., Nava, F., Dueñas, J.C. Modeling and Documenting the Evolution of Architectural Design Decision (SHARK/ADI'07), ICSE Workshops, IEEE CS, 2007.

[8] Capilla, R., Nava, F. Extending Software Architecting Processes with Decision-Making Activities. CEE-SET, Springer-Verlag LNCS 5082, 182-195, 2007.

[9] Dutoit, A. H., McCall, R., Mistrík, I., Paech B. (Eds.) Rationale Management in Software Engineering, Springer-Verlag, 2006.

[10] Falessi, D., Becker, M. Cantone, G. Documenting Design Decision Rationale to Improve Individual and Team Design Decision Making: An Experimental Evaluation, International Symposium on Empirical Software Enginering (ISESE'06), 134-143, 2006.

[11] Farenhorst, R., Izaks, R., Lago, P., van Vliet, H. A Just-In-Time Architectural Knowledge Sharing Portal, WICSA'08, IEEE CS, 125-134, 2008.

[12] Jansen, A., Bosch, J. Evaluation of Tool Support for Architectural Evolution. Automated Software Engineering Conference (ASE), IEEE CS, 375-378, 2004.

[13] Jansen, A. and Bosch, J. 2005. Software Architecture as a Set of Architectural Design Decisions. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05) - Volume 00 (November 06 - 10, 2005). IEEE Computer Society, Washington, DC, 2005.

[14] Jansen, A., de Vries, T., Avgeriou, P. and Veelen, M. v. Sharing the Architectural Knowledge of Quantitative Analysis, Proceedings of the Quality of Software-Architectures (QoSA), 220-234, 2008.

[15] Kruchten P., Lago P., van Vliet H, Building up and reasoning about architectural knowledge. In: Hofmeister, C. (Ed.), Proceedings of Second International Conference on the Quality of Software Architectures (QoSA 2006), Springer LNCS 4214, 2006.

[16] Kruchten, P., Capilla, R., Dueñas, J.C. The Decision's View Role in Software Architecture Practice. IEEE Software vol 26(2), 36-42, 2009.

[17] Lago, P., Avgeriou, P. 1st Workshop on Sharing and Reusing Architectural Knowledge. ACM SIGFOFT Software Engineering Notes 31(5), 32-36, 2006.

[18] Lee, L., Kruchten, P. Visualizing Software Architectural Design Decisions, ECSA'08, Springer-verlag LNCS 5292, 359-362, 2008.

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

[20] Tyree, J., Akerman, A. Architecture Decisions: Demystifying Architecture, IEEE Software, 22(2), 2005.

[21] Zimmermann, O., Gschwind, T., Küster, J. M., Leymann, F. Schuster, N. Reusable Architectural Decision Models for Enterprise Application Development. (QoSA'07), Springer-Verlag LNCS 4880, 15-32, 2007.

[22] Zimmermann, O., and Schuster, N. Architectural Decision Knowledge Wiki. Available at: http://www.alphaworks.ibm.com/tech/adkwik, 2008.