Generalized Markdown Architectural Decision Records: Capturing the Essence of Decisions

Oliver Kopp and Anita Armbruster

IAAS, University of Stuttgart
lastname@informatik.uni-stuttgart.de

Abstract. Documenting design decisions in a project fosters understanding while developing and maintaining the software. For that, Markdown Architectural Decision Records (MADR) have been proposed. When setting up a new project, certain decisions have to be done again. With the current MADR approach, how to reuse the knowledge of already taken decisions. This paper proposes Generalized Architectural Decision Records, where knowledge of concrete decisions is stored in a general format to capture "templates" for new decisions fostering reusability.

1 Introduction

Software needs to documented. In previous work [2] we introduced the Markdown Architectural Decision Records (MADR) as developer-friendly format to capture architectural decisions: Architectural decision records answer "why" questions about designs and make tacit knowledge explicit. In MADR, the decisions are written down using Markdown next to the code where the decisions were taken. MADR was derived from the Y-Statements [4], which in turn were compared with other formats by Zimmermann et al. [5] and turned out to be most promising. MADR requires an explicit justification of the decision and recommends enlisting the pros and cons of each option considered. A concrete decision might be to use Maven as build tool and comparing it to other build tools such as Gradle or Ant. In this form, the decision is not transferable to other projects. However, reusing rationales can increase development efficiency [3].

In this paper we present GADR as an approach to capture the generic aspects of architectural decisions and to make them reusable in other projects. Thereby, MADR fields related to the concrete decision outcome (such as justification, positive and negative consequences) are dropped as they are tightly related to the concrete setting and not general enough.

Generalizing and reusing architectural decision knowledge is not new [6]. ADMentor [5] is one tailored tool for handling that knowledge. This tool, however is not capable to store the knowledge in Markdown For that, Büchler [1] developed a storage solution. However, that solution uses a format not compatible to MADR. In this paper, we want to extend MADR's capabilities and not introduce another solution. This paper presents a first step towards a complete new development and capturing process for decision records.

S. Kolb, C. Sturm (Eds.): 11th ZEUS Workshop, ZEUS 2019, Bayreuth, Germany, 14-15 February 2019, published at http://ceur-ws.org/Vol-2339

The remainder of the paper starts with a presentation of GADR (Sect. 2). It is followed by a conclusion and an outlook on future work (Sect. 3).

2 GADR

Figure 1 presents the class model of MADR and the GADR. MADR inherits from GADR, since MADR specializes GADR. Both have a title, a context, and decision drivers. Both enumerate possible options having a description, pros, and cons. Both offer linking other MADRs or GADRs to, for example, enable linking to updated versions of a decision.

MADR adds the status of a concrete decision, the deciders on a concrete decision, and a date of the decision taken. This is not required at generalized decisions, because they are not decided, but get decided when forming a MADR.

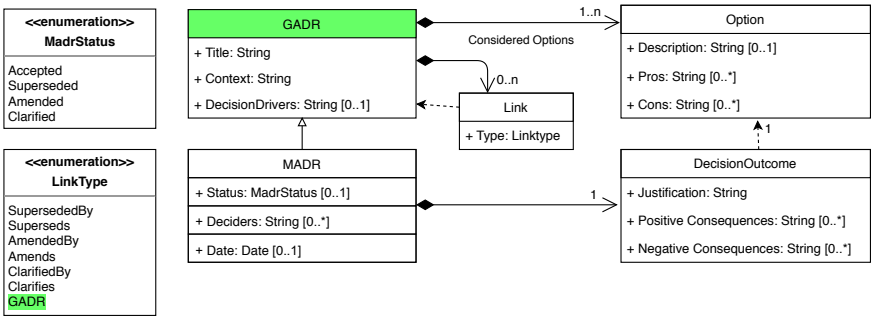

Fig. 1. MADR and GADR modeled using UML. Green: New in GADR.

The initial concept is to store the generalized ADRs in a central repository. When a concrete architectural decision has to be taken in a project, that repository is searched for. When a generalized ADR exists, it is copied into the repository of the project. Then, it is extended with pros and cons being valid for the concrete project. When a MADR has to be updated due to new facts or other reasons, a new MADR has to be created, the status of the old one set to "superseded", and a link to the new one added. History of GADRs and MADRs needs to be tracked with version control. There is no intention to add history to MADR or GADR itself.

We created an initial repository for GADRs in the context of Java on GitHub at `https://github.com/adr/gadr-java`. We used the GADR for java build tools in JabRef[1] and Eclipse Winery[2]. When creating each ADR, we did not

[1] `https://github.com/JabRef/jabref/blob/master/docs/adr/0003-use-gradle-as-build-tool.md`

[2] `https://github.com/eclipse/winery/blob/master/docs/adr/0023-use-maven-as-build-tool.md`

need to add new pros and cons of each option. We weighted the pros and cons differently leading to a different decision outcome (Gradle for JabRef, Maven for Eclipse Winery) and the respective justification. A main reason for not modifying the pros and cons is the fact that the decision record was created after the build system was chosen. Nevertheless, this usage of GADR shows that MADRs can be created based on GADRs in real-life projects.

3 Conclusion and Outlook

This paper presented GADR as first idea to store the general aspects of concrete architectural decisions using Markdown. The format was extracted from MADR to enable simple reuse of collected pros and cons. We described one usage of GADR in the context of JabRef and Eclipse Winery. The next step is to come up with a well-described development workflow comparable to the work by Zimmermann et al. [5] and Thurimella et al. [3]. In parallel, we aim for collecting more generalized architectural decision records to ease knowledge transfer.

It is an open topic to structure the collection of generalized architectural decision records. Our current approach is to offer an index to the GADR repositories at `https://adr.github.io/gadr/`. Possibly, an updated index of all available GADRs will be added there to ease search for GADRs. Developing a graphical tool presenting all MADRs an their relation to GADRs and thus enabling governance is foreseen.

References

1. Büchler, A.O.: Software Engineering Repository (SE-Repo) Gesamtkonzept, Umsetzung mit Git und Validierung (2017), Master Thesis, HSR Hochschule für Technik Rapperswil
2. Kopp, O., Armbruster, A., Zimmermann, O.: Markdown Architectural Decision Records: Format and Tool Support. In: ZEUS. CEUR Workshop Proceedings, vol. 2072. CEUR-WS.org (2018)
3. Thurimella, A.K., Schubanz, M., Pleuss, A., Botterweck, G.: Guidelines for Managing Requirements Rationales. IEEE Software 34(1), 82–90 (Jan 2017)
4. Zdun, U., Capilla, R., Tran, H., Zimmermann, O.: Sustainable Architectural Design Decisions. IEEE Software 30(6), 46–53 (Nov 2013)
5. Zimmermann, O., Wegmann, L., Koziolek, H., Goldschmidt, T.: Architectural Decision Guidance Across Projects – Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge. In: Working IEEE/IFIP Conference on Software Architecture (2015)
6. Zimmermann, O., Miksovic, C.: Decisions required vs. decisions made. In: Aligning Enterprise, System, and Software Architectures. IGI Global (2013)