Markdown Architectural Decision Records: Format and Tool Support

Oliver Kopp[1], Anita Armbruster[1], and Olaf Zimmermann[2]

[1] Institute for Parallel and Distributed Systems, University of Stuttgart
Stuttgart, Germany
{lastname}@ipvs.uni-stuttgart.de
[2] Institute for Software, Hochschule für Technik (HSR FHO)
Rapperswil, Switzerland
olaf.zimmermann@hsr.ch

Abstract. Architectural decision records answer "why" questions about designs and make tacit knowledge explicit. Many architectural decisions are made during development iterations because they have a close connection to the code. It is challenging to come up with task-specific decision capturing practices and supporting tools that are not perceived as time wasters; context switches and media breaks that harm the productivity of coding architects and developers involved in the decision making have to be avoided. To integrate existing architect-centric approaches into the developer toolchain, this paper presents a Markdown-based decision capturing template that we derived from previous work to enhance an existing decision capturing tool for developers. Our early validation results in the context of an open source project suggest that format and tool promise to contribute to an integrated decision capturing practice, with further enhancements being required. Tool and template are available in public GitHub repositories.

1 Introduction

Source code needs to be documented. This typically leads to comments in code and to external documents. Well-written classes and methods, which have expressive names and understandable branches [6], make low-level code comments obsolete. On the other end of the spectrum, however, wide-ranging decisions of high architectural significance are made during development iterations; these decisions are not self-explanatory and not expressed in the code explicitly. An example of such an *architectural decision* is how to keep user session data consistent and current across Web shop instances. Typically, these kind of decisions are recorded in external documentation files, wikis, or tools [18]. The primary tool of developers, however, is their Integrated Development Environment (IDE) with integrated version control system support. Requiring developers to use more tools has a negative impact on productivity, quality, and motivation to capture architecturally significant decisions: opening another tool requires some setup and training effort and leads to undesired, time consuming context switches. Furthermore, model- and document-centric tools typically do not integrate themselves well in the developer's toolchain: The documents are not committed in plain text format into the version control system—if versioned along the code at all. Further, the documents might get out of sync with the