Toward Collecting and Delivering Knowledge for Software Design at the Whiteboard

Adriana Meza Soria
University of California, Irvine
amezasor@uci.edu

André van der Hoek
University of California, Irvine
andre@ics.uci.edu

ABSTRACT

Knowledge – of all kinds – is essential to software design. It is well known, however, that most knowledge resides in the developers' heads. Especially when designing at the whiteboard, tool support is minimal, most often non-existent. In this brief research note, we sketch our ongoing research into supporting designers at the whiteboard with proactive knowledge that is collected, earlier, in some (semi-)automatic fashion.

CCS CONCEPTS

• Software and its engineering → software notations and tools

KEYWORDS

Design knowledge, whiteboard design, sketching tools

ACM reference format:

Adriana Meza Soria and André van der Hoek. 2018. Toward Collecting and Delivering Knowledge for Software Design at the Whiteboard. In *Proceedings of 11th International Workshop on Cooperative and Human Aspects of Software Engineering, Gothenburg, Sweden, May 2018 (CHASE'18),* 2 pages.
https://doi.org/10.1145/3195836.3195859

1 INTRODUCTION

Consider the following two abbreviated scenarios, both taken from real-world situations that actually occurred:

- Two developers had to update a meal service software system to accept IOUs. The application was built using a web service, and they found the right component to insert the new functionality. Their design was flawless, but sadly wrong since, once implemented, they discovered the web service was also used by other applications that now fail.
- Two architects were tasked with redesigning some communication software between two locations that are connected using a dedicated link. The old software uses UDP. While designing at the whiteboard, they decide to upgrade to TCP/IP in order to increase reliability. A colleague walks by who overhears the conversation, returns to their office, and proclaims 'the communications link is too slow to support TCP/IP roundtrip'.

The second scenario is preferred to the first, in that the missing knowledge was identified early, but in both cases, the question arises as to how the developers or architects could have known what they needed to know to avoid the problem they initiated.

Simply counting on experience or expertise is insufficient; even the most senior software designers do not have the ability to know 'everything' that pertains to their design work. Therefore, tool support is essential. Unfortunately, nearly all support to date exists for software designers working at a desktop. Less research aims to support them where they are found most often when faced with a complex design task: at the whiteboard.

2 KNOWLEDGE

In the domain of software engineering, knowledge has been defined in many different ways, including the raw material of software design teams [12] and an important factor to enhance the completeness, correctness, and understandability of requirements [9]. These definitions highlight that knowledge is critical, but they do not provide a precise idea of what knowledge is. Garzas et al. attempt to be more precise, defining knowledge as the set of information chunks that software engineers need and find materialized in standards, methodologies, methods, metrics, and concepts [4]. This definition, however, focuses on knowledge of software engineering, not on knowledge that relates to the software under consideration. Tang et al. introduce a taxonomy that separates: (1) *contextual knowledge*, capturing knowledge about the situation, (2) *reasoning knowledge*, focusing on understanding the decisions that have gone into the software to date, and (3) *technical knowledge*, in the sense of Garzas [10]. Tang et al. include a fourth category orthogonal to the first three, *design knowledge*, by which they indicate knowledge of the former three that is specified in a model and thus readily available. In our work, we are concerned with contextual and reasoning knowledge: facts about the state of the world and state of the system that might influence the decisions that designers at the whiteboard need to make.

CHASE'18, May 27, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5725-8/18/05…$15.00
https://doi.org/10.1145/3195836.3195859

3 EXISTING TOOLS

Several tools have been developed to address both capturing knowledge and delivering knowledge. Examples include AREL [11], Archium [6], ADDSS [3], Knowledge Architect [5], SEURAT [2], and Decision Architect [8]. The support these tools provide differs. For instance, ADDSS and AREL support developers with contextual knowledge they usually need during the early stages of projects. As another example, Archium provides design patterns and architectural styles to reuse (i.e., technical knowledge). Overall, the tools vary significantly in terms of the kinds of artifacts they support (e.g., business processes, architecture, lower-level design), methods for capturing knowledge (e.g., automatic extraction, manual specification, tool-supported specification), and methods for accessing it (e.g., automatic notifications, search tools).

With respect to our research agenda, proactively providing relevant knowledge to designers when they work at the whiteboard, none of these existing tools addresses this goal. Only one offers knowledge proactively (SEURAT) and none are designed for working at the whiteboard. A significant challenge is that, at the whiteboard, design takes place using informal sketches [1], making it difficult for tools to understand what knowledge may be relevant. Another, perpetual challenge is that collecting relevant knowledge often is a manual process.

4 OUR RESEARCH

Our goal is to support designers at the whiteboard with proactive knowledge delivered to them right when they need it (and, complementary, collect such knowledge automatically during earlier phases of the work). We will be developing our project in three stages along three research questions.

First, we seek to understand what knowledge is useful when designers work at the whiteboard. Our hypothesis is that not all contextual and reasoning knowledge is as relevant and suspect that, particularly for contextual knowledge, a few key pieces of knowledge might govern the majority of situations. With such knowledge typically resident in the heads of the lead designers, and not communicated to others as much as it should, we plan to engage in field studies, interviews, and subsequent surveys to get a sense of what kind of knowledge actually matters to designers, and when.

Second, we want to explore if it is possible to (semi-)automatically collect knowledge in previous phases of the project. Depending on what we find for research question 1, this question may or may not become more important. With a smaller set of knowledge that is 'truly important', it might be possible to simply preload it in a tool for further consumption (see below). But if more knowledge is required, the question becomes how to collect it. We will explore how modern technology could assist. For instance, assuming that discussions during early design sessions identify important knowledge, it might be possible to develop a solution as simple as a button on the electronic whiteboard that a developer presses to signal key knowledge was just generated, with audio and sketch content captured for later processing. It might also be that we adopt and augment a sketch tool like Calico [7] and include the opportunity to create reusable knowledge cards by labeling a scrap as such, which I turn enables the developer to add a few brief notes before the card becomes part of a library. We also believe that voice recognition today is becoming sufficiently sophisticated that it may become possible for developers to instruct a tool with certain keywords to record key knowledge. Our plan is to explore these and other such solutions through tool prototyping.

Finally, we want to explore whether it is possible to create tools that proactively deliver relevant knowledge to software designers working at the whiteboard at just the right time. We again will start with a simpler approach: letting developers select relevant items beforehand. That is, to understand if having relevant knowledge at hand actually can make a difference, we envision presenting developers before a meeting with a limited set of cards (mimicking Calico scraps, as well as design method cards), ask them to identify relevant ones, and engage in the design task with the cards at hand. From there, we progressively want to work toward approaches that perform sketch recognition, hand-writing recognition, and listen to the ongoing discussion of the developers to identify key design elements that are matched in some way against the library of knowledge.

REFERENCES

[1] Baltes, S. and Diehl, S. 2014. Sketches and Diagrams in Practice. *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (New York, NY, USA, 2014), 530–541.

[2] Burge, J. and Brown, D. 2008. SEURAT. *2008 ACM/IEEE 30th International Conference on Software Engineering* (May 2008), 835–838.

[3] Capilla, R., Nava, F., Pérez, S. and Dueñas, J.C. 2006. A Web-based Tool for Managing Architectural Design Decisions. *SIGSOFT Softw. Eng. Notes.* 31, 5 (Sep. 2006). DOI:https://doi.org/10.1145/1163514.1178644.

[4] Garzas, J. and Piattini, M. 2005. An ontology for microarchitectural design knowledge. *IEEE Software.* 22, 2 (Mar. 2005), 28–33. DOI:https://doi.org/10.1109/MS.2005.26.

[5] Jansen, A., Avgeriou, P. and van der Ven, J.S. 2009. Enriching software architecture documentation. *Journal of Systems and Software.* 82, 8 (Aug. 2009), 1232–1248. DOI:https://doi.org/10.1016/j.jss.2009.04.052.

[6] Jansen, A. and Bosch, J. 2005. Software Architecture as a Set of Architectural Design Decisions. *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)* (2005), 109–120.

[7] Mangano, N., Baker, A., Dempsey, M., Navarro, E. and van der Hoek, A. 2010. Software Design Sketching with Calico. *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering* (New York, NY, USA, 2010), 23–32.

[8] Manteuffel, C., Tofan, D., Koziolek, H., Goldschmidt, T. and Avgeriou, P. 2014. Industrial Implementation of a Documentation Framework for Architectural Decisions. *2014 IEEE/IFIP Conference on Software Architecture* (Apr. 2014), 225–234.

[9] Taheri, L., Pa, N.C., Abdullah, R., Abdullah, S. and Shafazand, M.Y. 2014. Identifying knowledge components in software requirement elicitation. *2014 IEEE International Conference on Industrial Engineering and Engineering Management* (Dec. 2014), 286–291.

[10] Tang, A., Avgeriou, P., Jansen, A., Capilla, R. and Ali Babar, M. 2010. A comparative study of architecture knowledge management tools. *Journal of Systems and Software.* 83, 3 (Mar. 2010), 352–370. DOI:https://doi.org/10.1016/j.jss.2009.08.032.

[11] Tang, A., Jin, Y. and Han, J. 2007. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software.* 80, 6 (Jun. 2007), 918–934. DOI:https://doi.org/10.1016/j.jss.2006.08.040.

[12] Walz, D.B., Elam, J.J. and Curtis, B. 1993. Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration. *Commun. ACM.* 36, 10 (Oct. 1993), 63–77. DOI:https://doi.org/10.1145/163430.163447.