Knowledge from Document Annotations as By-Product in Distributed Software Engineering

Anna Averbakh∗, Kai Niklas‡, Kurt Schneider∗

∗ Software Engineering Group, Leibniz Universität Hannover, Germany
{anna.averbakh, kurt.schneider}@inf.uni-hannover.de
† Talanx Systeme AG, Germany
kai.niklas@talanx.com

Abstract—Knowledge management can play a major role in the success of a distributed software engineering project promising huge increases in efficiency and effectiveness. However, it often suffers from a lack of participation. Major problems are that sharing knowledge is time consuming and bears additional effort for the knowledge worker.

In the course of development projects, software engineers create, read and annotate (particularly during reviews) a lot of documents. These annotations can contain valuable knowledge which should be made persistent and sharable with project partners. To lower the sharing effort for project participants, we present a light-weight approach to collect annotations as by-product from project (-related) documents. The annotations are extracted from documents and interlinked with other experience and knowledge artifacts in a shared Wiki-based experience infrastructure for global software engineering. As immediate benefit for knowledge workers, making annotations and context searchable saves information retrieval time. In the long term, annotations combined with other experiences are engineered into reusable recommendations.

In a preliminary evaluation in the software engineering research field and industry confirmed that our annotation sharing concept was perceived as helpful and time saving.

Index Terms—annotation management; knowledge management, distributed software engineering

I. INTRODUCTION

Team members in a software development project do not have to be distributed to show the symptoms of a distributed project. Already a distance of 30 meters is enough to diminish communication and awareness like in a globally distributed project [1].

Many distributed collaborative software development projects follow structured and formal processes like Waterfall, V-Model or RUP. In such a process the team’s work mostly revolves around (distributed) documents. Project participants often make annotations in these documents for different purposes, e.g. review or quick retrieval. Distributed projects have less direct communication and thus a lower awareness about project-related knowledge at a partner’s location [1]. Participants from the partner company may not know that these annotations exist or in which documents to look. Document annotations are a valuable knowledge source. Making these annotations available and efficiently searchable can save money and time for the subcontractor, which would cost him, if he had to ask around or look through each document in the shared project repository. Besides, annotated documents are often stored in the personal or project-wide workspace and do not become available for further projects.

As a motivating use case consider a situation, where a subcontractor team in a distributed collaboration has to create a design document. The subcontractor team needs to look up recommendations and examples on how the partner company creates UML diagrams. Such knowledge can be found e.g. in annotations inside reviewed documents from older projects containing UML diagrams. Beside communication and awareness problems, the perceived effort is still one of the most severe impediments to information and knowledge sharing (e.g. [2]). If the effort to share knowledge exceeds the reward or benefit, people will not do it [3].

Our contribution in this paper approaches these problems by presenting a light-weight concept to lower the effort and time to share project document annotations. The pieces of information and knowledge contained in annotations are made accessible and searchable for other teams, projects and collaboration partners. We present a methodology including a tool and environment, where extracted annotations can enrich the project memory and together with other experience and knowledge artifacts engineered into reusable guidelines.

In the next chapter we present the concepts of annotations as by-product and how to link them with other experiences. Then we present three preliminary evaluations and discuss them followed by a conclusion and topics for future research.

II. RELATED WORK

In this section we present related research about annotation sharing and supporting the user by extracting knowledge as by-product.

A. Annotation Extraction and Sharing

There are a lot of annotation systems and tools for collaborative work on web documents (e.g. [4], [5]). Alani et al. automatically extract knowledge from web documents and populates a knowledge base [6]. Bischofberger et al. use annotations as tags to connect software artifacts for cooperative tasks [7]. These annotations are rather formal. In our case, we extract free text annotations from distributed offline software engineering project(-related) documents.
Tools like Mendeley\(^1\) or Adobe Reader\(^2\) allow sharing annotations. However, Mendeley only allows to view them in its infrastructure and Adobe Reader only offers a non-searchable annotation list without visual context. Martine et al. gather and link discussions from e-mail attachments, shared folders and meetings to documents they relate to in a Wiki [8]. These discussions can have a similar content to annotations. However, information about a document extracted from annotations has a more tight connection to parts of the document than discussion points from e-mails or meetings. Liang et al. provide Excel and Word plug-ins that allow highlighting text passages, add meta data annotations and relate them to others [9]. Our approach is more light-weight and is less intrusive in the every-day work process. It does (almost) not require additional effort from the software engineer and does not limit the way how to annotate.

B. Knowledge as By-Product

Meyer et al. followed a by-product approach using (Microsoft Word) meeting protocols [10] to create added value for project participants. They automatically extracted decisions, tasks and definitions and displayed them in a corporate Wiki, thus making them searchable. The tasks were automatically uploaded into an issue tracker (Jira). Our approach also strives to use documents that are created anyway without added effort for the project members. In our case, we use annotated documents. Schneider describes a light-weight method to store software engineering decision rationale as by-product [11]. Schneider also presents a strategy how to record knowledge emerging as by-product from prototype development [12]. We apply this approach to annotations.

III. Knowledge from Annotations as By-Product

In this section we present the concepts of this contribution. With a use case we illustrate why annotations are a valuable knowledge source and can save time as well as money in distributed software development. We present how they can be made persistent and shared for organizational learning.

---

1https://www.mendeley.com/
2http://get.adobe.com/de/reader/

---

A document creation process, e.g. writing a specification, usually contains review loops. In a co-located project setting, the feedback will likely be in oral form: an architect can personally tell the analyst what to revise in his specification draft. In a distributed development setting, however, direct communication is more difficult and asynchronous [13]. Thus, feedback will likely be shared in solid form like e-mail or as annotations inside the document. Figure 1\(^3\) illustrates information and knowledge flow in a part of a (simplified) distributed development process. The process starts with an analyst from project A, who has the task to write a specification. Thereby, he consults other documents like guidelines, rules and norms (1). The output of the writing process is a specification draft. This draft is then reviewed by the customer and architect. The former reads the specification and makes annotations on functional issues like missing or wrong requirements, improvement suggestions or opaque passages (3). The architect reads the draft in order to check if all requirements can be implemented and if there are mistakes in diagrams (4). To strengthen his argument, he may reference to a rule or guideline. He also annotates passages that are critical or costly and should be renegotiated with the customer. After receiving the annotated drafts from the reviewers, the analyst from project A writes a final version of the specification. There can be, of course, more than one review cycle. The tester uses the specification to create test cases. He marks passages in the specification that are hard to or cannot be tested. He also may annotate requirements with the test case number to quickly retrieve the relation (5). To other analysts from project B or project A at a partner location, the annotated documents become inaccessible after project A ends (6). Besides, permissions and documents are too unstructured to be valuable for people from project B.

Our approach is to store (potential) knowledge from annotations (Fig. 2) in a knowledge base, where they can be immediately searched and commented and provide an infrastructure to engineer them into recommendations in the long term. An analyst from a different project B (or from a sub-project

\(^3\)In Figs. 1 and 2 we use the FLOW notation to illustrate information and experience flows between people and documents. For the full syntax description refer to Stapel et al. [14].

---

Fig. 1. Part of a (simplified) development process in a distributed setting without document annotation sharing.
of project A) can learn and benefit from the annotations. He can save time and money by reusing recommendations engineered from annotations in the previous project. He can avoid phrasing pitfalls or learn e.g. about company specific characteristics like UML diagram specifics. He can more easily renegotiate and clarify untestable or very costly requirements. New annotations created during project B are a further input to the annotation database. Besides review annotations and annotated guidelines, other annotated cross-project documents like publications can also be collected. For a process manager annotations could reveal possible flaws and hidden places of unclarity in the specification template. We assume that repeated annotations denoting unclarity or problems about a similar topic or at the same place in the document could indicate that either the template is not sufficiently descriptive or the employees lack expertise and need training.

Considering the participation and motivation problems in knowledge management initiatives (e.g. [3]), we chose a lightweight approach similar to Schneider [11] to elicit the annotations. To be as little intrusive as possible for the annotation author, we extract the annotations automatically into the shared knowledge base. The author only needs one click to start the annotation extraction process. The command can be triggered from the context menu of the document.

IV. INTERLINKING AND ENGINEERING ANNOTATIONS

To create more benefit for the project participants, we related the extracted annotations to other knowledge and experience artifacts (s. Fig. 3). We integrated the annotations into a shared experience base for distributed software engineering (GloSE Base) [15]. This experience base is a Wiki containing experiences of different maturity. It has a rights management structure to handle sensible experiences, which is also applied to annotations. Subjective (raw) experiences are engineered into reusable recommendations. The extracted annotations in Fig. 3 can be viewed and searched in the shared knowledge base. A copy of the annotated document is attached to the page and can be downloaded. Each annotation is contextualized in the annotation context: a screenshot of the annotation surroundings in the document. Annotations and other experiences can be engineered into a recommendation. In turn, a recommendation can link to the annotations that support it. Each annotation can be commented or linked to other documents that are related to the annotation content, e.g. a source to prove a statement. These comments and documents can also be engineered into best practices. Additionally, project participants can still access the document in its original context of the original shared folder structure. We are implementing this interlinking concept with a huge publication knowledge base structured according to SPICE (ISO/IEC 15504) on global software engineering [16].

V. PRELIMINARY EVALUATION

We evaluated the usefulness of the annotation extraction and sharing concepts in the software engineering domain. We identified the following research questions:

RQ1: Is persisting and sharing annotations useful for software engineering tasks?

RQ2: What kind of annotations are used in the software engineering field?

RQ3: Is our annotation extraction and interlinking approach helpful? In particular, does our approach save time reading a document to find relevant information to solve a problem?

RQ4: Can annotations be used to improve software engineering processes? Specifically, can review annotations denoting frequent mistakes of a kind give evidence to flaws and unclarity in document templates?

A. Methodology

We conducted three evaluations. To answer RQ1 and RQ2, we distributed a survey to software engineers in three middle sized and big software companies. The software engineers were given questions about their annotation behavior and whether our approach could be helpful for their work.

To answer RQ3, we conducted a cross validation. Our participants were software engineering research associates. The evaluation was conducted at their workplace. We implemented an annotation extractor for PDF documents that automatically uploads annotations into our Wiki-based experience base as described in section IV. The participants were given two publications and had to answer several questions once with and once without the help of extracted and shared annotations. The participants from the control group had the same procedure but with switched publications. The publications were chosen to be tedious to read without the possibility to quickly scan the content. They were long (23 and 31 pages), relatively unstructured with long textual blocks and few figures. Their topic was chosen to be related to publications the participants would normally read but not from to their direct research field. The questions were not directly searchable in the documents.

To answer RQ4, we analyzed 32 annotated requirement specifications from several reviews in a student software

---

4 We have observed that researchers more often read and annotate PDF than MS Office documents.
To answer RQ4, we found out that most annotations (74) denoting uncertainty were on pages where the students had to write use cases and draw use case diagrams. This finding is supported by the fact that the most frequent term in all annotations of uncertainty was “use case”. This analysis indicates that these template chapters lack a good example or explanation or that the students lack expertise about how to create good use cases. After project end, the students had a post-mortem session where they could share their feedback and experiences during the project. Many groups complained about their problems with use cases and lack of a good use case example in the template. This feedback also supports the results of the annotation analysis.

VI. DISCUSSION

Our study indicates that extracting, sharing and interlinking project (related) document annotations is useful and helpful. Certain limitations should be considered when interpreting our results: Both the academic and industrial studies did not produce statistically significant results due to a low number of participants ($n = 9$ and $n = 6$). A statistical significance could be achieved with a larger group.

5http://lucene.apache.org/
We limit our evaluation to “perfect” annotations in a fixed content (PDF publications). We consider an annotation “perfect” when it marks the answer in the text. We wanted to see whether such annotations can be helpful for the task of finding an answer in a research environment. Thus, our results seem unlikely to be completely representative in the software engineering field. An extensive study could give more reliable insights but our evaluation provides a tendency.

The rather low inclination (2 of 6 answers) for sharing review annotations probably reflects study participants’ opinion that these kind of annotations have a too specific context to be helpful in other projects. This may apply for direct reuse. However, our analysis results of review annotations indicate they can be helpful after engineering. Annotation analysis can be an indirect way to communicate development process flaws, which can be suitable especially in a distributed setting with impaired direct communication. Improving document templates, for example, could be helpful in the next project, as templates are usually reused. In a collaborative project, a good template is even more important if a team has to use a partner’s template [17].

As a limitation to our evaluation, we only analyzed a specific type of documents (requirement specifications) in a co-located undergraduate student software project and not in an industrial collaborative setting.

For the annotation analysis some of our assumptions, like the heuristic for finding annotations of unclarity, may be simplistic. However, it provided a problem indication that was later confirmed by students participating the software project. In addition, we plan a more sophisticated analysis to uncover more specific problems.

VII. CONCLUSION AND OUTLOOK

In this contribution, we have presented a concept for extracting project document annotations as by-product, make them persistent and searchable as part of the organizational memory and interlinking them with other knowledge and experience artifacts in a shared experience base. Sharing document annotations can save information retrieval time. Engineering them into reusable recommendations can help to avoid mistakes that can become costly. Preliminary evaluations indicate that document annotations in the software engineering domain are a valuable knowledge asset and should be shared and engineered. We also showed, that our concept is perceived helpful by software engineers.

Our analysis of annotations has shown that they can be an indicator to flaws (in templates or in expertise) and give a notion on what can be improved in a software engineering process. This indirect way of uncovering problems can be helpful in a distributed setting when direct communication is impaired.

For further research we will add the possibility to monitor folders and automatically upload annotated documents to unburden document authors even more. We will also consider an automatic and manual support for annotation categorization and tagging. This will provide further annotation context like the topic or problem it refers to. It will make the search for annotations easier, since a document can have annotations related to different problems and topics.

In the case of a changeable document, a synchronization and tracing of annotation context in the Wiki and the original document can be helpful. Although we made the observation that helpful annotations can lower the barrier to read academic publications, we will evaluate to which extent this insight can also be applied for the distributed software engineering domain.

Our evaluation also showed that software engineers create a lot of review annotations in transient content, but only few participants consider them worth sharing. The analysis of reviewed student software project requirements documents has shown that analyzing annotations of unclarity can give a hint about flaws in templates or team members’ expertise. We will also evaluate other ways for annotation analysis to support software engineering process improvement. In particular, we will investigate if review annotations of a specification can indicate communication flaws between customer and requirements engineer or provide input for a glossary.

REFERENCES