Towards Software Architecture Documents Matching Stakeholders' Interests

Matías Nicoletti[1,2], J. Andres Diaz-Pace[1,2], and Silvia N. Schiaffino[1,2]

[1] ISISTAN Research Institute, Facultad de Cs. Exactas, UNICEN University, Campus Universitario, Pje. Arroyo Seco - (7000) Tandil, Buenos Aires, Argentina
[2] CONICET-Argentina
{mnicolet,adiaz,sschia}@exa.unicen.edu.ar

Abstract. Architecture documentation is a crucial activity in any software development project. In practice, architecture documenters face two problems: how to generate relevant documentation contents for the main stakeholders, and how to avoid documenting too much about the architecture. We propose a personalization approach based on stakeholders' interests to tackle these problems. The expected contribution is to facilitate the documenter's tasks, while making the resulting documentation useful to the stakeholders. We specifically describe a user profiling tool that builds stakeholders' profiles, which serve to link the stakeholders to sections of the architectural documents. These links help the documenter to prioritize sections that are potentially relevant to those stakeholders. The tool has been implemented as a semi-automated pipeline based on text mining techniques. The results, although preliminary, show that our proposal is helpful for a stakeholder-centric architecture documentation process.

Keywords: Architecture Documentation, User Profiling, Text Mining, Stakeholders.

1 Introduction

Documentation is a common activity of any software development project, and it is important in early development stages because the decisions made at those stages will shape the rest of the development and the quality of the software product. Documentation is also useful for knowledge sharing and communication among project stakeholders. A relevant documentation artifact is the so-called *Software Architecture Document (SAD)*, which captures the key *design decisions* that enable the system to satisfy its main quality attributes, and therefore, the business goals posed by the system stakeholders [4]. Basically, architectural decisions refer to a number of high-level software structures and patterns (e.g., layers, client-server, etc.), normally depicted via *architectural views*, and their justification in terms of quality-attributes and functional requirements.

Producing good architectural documentation and keeping it up-to-date is challenging, particularly in the context of iterative development processes. On one

F.V. Cipolla-Ficarra et al. (Eds.): ADNTIIC 2011, LNCS 7547, pp. 176–185, 2012.

side, documentation consumers (e.g., stakeholders) want to access to the "right architectural contents" of the SAD, with as less information overloading as possible. Unfortunately, these consumers are often swamped with architectural knowledge that not always satisfies their *information needs*. Recent studies [8, 11] have shown that many individual stakeholder concerns are addressed by a fraction (less than 25%) of the SAD, but for each stakeholder a different SAD subset is needed. On the other hand, the documentation writer (or *documenter*) faces several forces that constraint his/her task, such as: how to generate timely documentation for the main stakeholders, how to keep up with the features being added in development iterations, and how to avoid documenting "too much" about the architecture. In addition, the value of documentation writing is often not clearly perceived by upper levels of management.

In this context, we see the SAD management process as a balancing act between having "good enough" documentation (for the stakeholders) and creating it in a cost-effective manner. By good enough, we mean the degree to which the documentation supports the stakeholders' tasks, while also exposing concerns such as architectural risks or quality-attribute tradeoffs. By cost-effective, we imply that the documentation is delivered incrementally, and its contents are prioritized according to some economic strategy. Over the last years, several architecture documentation approaches have been developed [12] and tool support has become a key asset (e.g., CASE tools, Wikis, collaborative platforms). These efforts have mainly targeted the consumer's perspective of the documentation process. In this article, we focus on the documenter's perspective, that is, how the documenter can produce SAD contents that are actually informed by the stakeholders' information needs.

Certainly, having a SAD with the necessary architectural views to satisfy all the stakeholders would be ideal, but this is seldom the case in real projects, due to economic reasons, schedule pressures, and also conflicting stakeholders' interests, among others. A more practical approach is to select a set of views addressing the concerns of the most relevant stakeholders, and then adjust the view contents and level of details accordingly [2]. In order words, we argue for a *personalization* of the SAD contents. To do so, we propose capturing the main characteristics of each stakeholder by means of *user profiling techniques* [9]. The expected contribution of our approach is to facilitate the documenter's tasks, making the documentation process both cost-effective and more useful to the stakeholders. In this article, we specifically describe a tool approach that builds stakeholders profiles based on topics of interest, which serve to link the stakeholders to sections of the architectural documents. Along this line, we have implemented a semi-automated pipeline based on text mining techniques. The outputs of this pipeline help the documenter to prioritize the sections of a SAD based on their stakeholder relevancy. The results, although preliminary, show that our approach is helpful for the whole architecture documentation process.

The rest of the article is organized around 4 sections. Section 2 discusses background and related work. Section 3 presents the main components of the proposed approach. Section 4 describes the current prototype for the detection of stakeholders' interests and their linkage to architecture documents. This section also includes an experimental evaluation of the prototype. Finally, section 5 gives the conclusions and outlines future work.

2 Related Work

Several approaches have investigated how to codify and make (better) use of architectural knowledge. In fact, the IEEE Standard 1471-2000 about Recommended Practice for Architecture Description of Software-Intensive Systems [6] recognizes the need of supporting the understanding of the SAD. However, few approaches have targeted the personalization of these documents by means of user profiles.

Farenhorst et al. [3] analyzed the requirements for a tool to capture architectural knowledge and share it among a group of architects. Based on these requirements, the authors also created a web portal called JIT AK Portal that includes some personalization functions. These functions are mostly oriented to the documentation reader, and particularly to a single stakeholder type (i.e., the architects). In JIT AK Portal, the documenter is responsible for providing a web interface with structure and contents that match the architects' interests. The approach provides no guidelines for this task. Another related tool is Knowledge Architect [7], developed by Jansen et al. The goals of this approach are to facilitate the access to a base of architectural knowledge and the search for specific issues in the knowledge base. A difference with JIT AK Portal is that Jansen's approach uses a codification strategy that supports the retrieval of knowledge for multiple types of stakeholders, although still concentrated on the reader's side. In addition, Knowledge Architect does not consider personalization techniques regarding the generation of documentation contents.

In [11], the authors propose an automated approach to deal with chunks of architectural information. These chunks are the result of specific exploration paths followed by a stakeholder when reading a SAD. The relevance of a given chunk is determined by factors such as the time spent by a reader on a section, or the access frequency for a section. The idea is to recommend candidate sections to new stakeholders by reusing previous (similar) exploration paths of the SAD. A prototype supporting the approach has been recently published [10]. This approach is interesting in the sense that assists a reader to find relevant information. However, the characteristics of that reader are neither explicitly captured nor used in the assistance.

A limiting aspect of the approaches above is that they are general-purpose in that they do not leverage on available architectural documentation methods. Currently, there are several documentation methods available for software architectures. A few relevant examples include: Views and Beyond (V&B) developed by the Software Engineering Institute, Viewpoints and Perspectives proposed by Rozanski and Woods, and Siemens' 4 Views [4]. These methods basically prescribe the structure of the SAD (i.e., a template), the kind of views to be used, and sometimes the relationships of these views with stakeholder types. Guidelines about the documentation process are usually not part of these methods. One exception is the V&B method, which provides a few rules for combining views or adjusting the level of detail of these views, based on the different stakeholders' roles [2]. A drawback of V&B is that the view templates are general and the stakeholders' roles are static. In practice, the documenter is expected to determine the "right contents" in order to fill in those view templates. V&B is often viewed by practitioners as being bureaucratic (or high-ceremony) with respect to the amount of documentation to be generated.

There have been experiments with Wikis as a mechanism to support architecture documentation. JIT AK Portal and the Knowledge Architect are examples of this kind of tool support. In fact, the V&B method has been also implemented on top of a Wiki. Some lessons learned of this experience are discussed in [1].

3 Proposed Approach

A well-known rule for producing good documentation is to write its contents from the reader's perspective, rather than from writer's perspective [4]. In the architecture documentation process, we interpret this rule as a feedback loop in which the documenter produces incremental versions of the SAD to fulfill the information needs of (most of) the stakeholders. Figure 1 shows a conceptual schema of our approach.

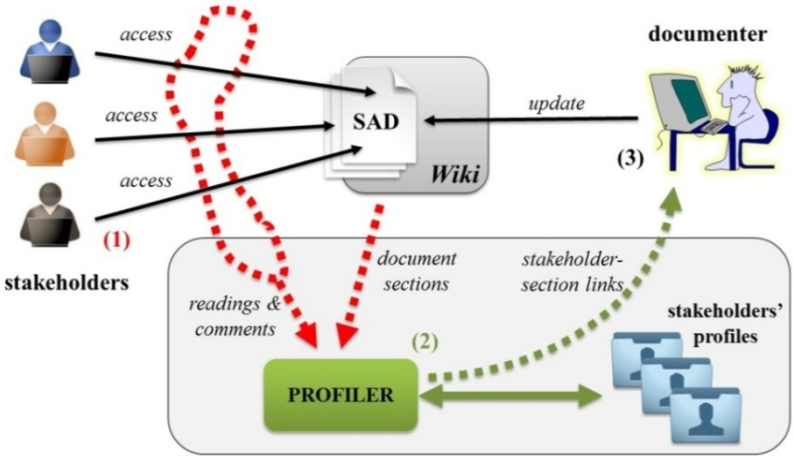

Fig. 1. Actors and components of our approach

We assume a community of stakeholders working on a given project, and interacting with each other through a collaborative platform (including tools such as: chat, forums and shared repositories). For their daily work, these stakeholders have access to a Wiki that contains the architectural documentation (i.e., the SAD) of the project. The documenter periodically delivers new SAD versions in the Wiki. The stakeholders may send feedback about a given SAD version, which will trigger updates in subsequent versions. Along this line, the main goal of the documenter is to decide which documentation tasks should be prioritized (and performed) for the next SAD version, in such a way the most relevant stakeholders are satisfied.

The approach involves three activities. First, the stakeholders access different sections of the current SAD hosted by the Wiki. We assume the SAD is structured around templates for predefined architectural views (e.g., module views, allocation views) and accompanying text. We base our approach on the V&B method, which already provides templates for the SAD. Stakeholders playing different project roles will have different concerns with respect to the SAD. For instance, project managers

are mainly interested in allocation views, whereas developers need extensive information about views of modules and components-and-connectors.

Second, as the stakeholders read different Wiki sections and eventually leave textual comments about the SAD contents, the platform in background collects this feedback and passes on to a profiler. The profiler also receives information about the structure of the SAD. Based on these inputs, the profiler applies text mining techniques in order to identify relevant *topics* in the stakeholders' comments as well as in the documents. The idea here is to infer stakeholders' interests and match them with specific sections of the SAD. Those topics coming from the stakeholders will serve to build (or update) *user profiles* of these stakeholders. Those topics extracted from the SAD will serve to establish *links* between a given stakeholder and a number of SAD sections, meaning that the stakeholder is potentially interested in those sections. These links will be used as recommendations to the documenter.

Third, the documenter takes both the user[1] profiles and recommendations produced by the profiler, in order to update the current SAD version. An update consists of a series of *documentation tasks*, such as: adding new contents to an existing section or architectural view, creating an architectural view, setting the level of detail for a section or view, among others. In this setting, we envision that the documenter keeps a "backlog" of documentation tasks, each task consuming a certain effort. At a given point in time, the documenter will combine several criteria to prioritize the current documentation tasks and select a subset of tasks to perform in the next documentation cycle. One of such criteria is determined by the analysis of the stakeholder profiles and their links to the SAD. This criterion reflects the "stakeholder value" of the documentation being generated. Another criterion is the total effort consumed by the chosen tasks, which is the "cost" of the documentation update. It is up to the documenter to strike a balance between cost and stakeholder value.

A key aspect of our approach is the semi-automated construction of stakeholders profiles (activity 2 in Figure 1), which is actually the focus of this article. In general, a profile can be obtained from various sources, such as: stakeholder roles in the project, access patterns to document sections, interactions with other stakeholders, and analysis of recurring topics in the stakeholder's activities. Information can be collected explicitly, through direct stakeholder intervention, or implicitly through agents that monitor user activity [9]. Initially, we define the stakeholders' profiles based on predefined interests derived from typical project roles. In fact, V&B characterizes several types of stakeholders regarding their use of architectural views. We can think of these characterizations as static profiles. However, these profiles will certainly change over time. For these reasons, we have designed our profiler to augment the static profiles given by V&B with topics of interest. These interests can be derived from the stakeholders' comments, reflecting the particularities of each stakeholder, so as to provide more accurate recommendations to the documenter.

4 Profiling Stakeholders via Text Mining

The Profiler component (see Figure 1) implements two processing stages, namely: profile building, and user-section linking. The first stage executes a mining procedure

[1] The terms 'stakeholder' and 'user' are used interchangeably.

to extract *human-knowledge concepts* from the users' text (see sub-section 4.1). A similar mining procedure is applied on the SAD contents. The outcomes of this stage are: a set of user profiles and a document representation (of the SAD).

Each user profile has a static and a dynamic aspect. The static aspect is taken from the stakeholders' characteristics provided by the V&B method (as the degree of interest of a stakeholder on a given architectural view). They also provide a list of predefined concepts for each stakeholder type (e.g., manager, architect, tester, etc.). The dynamic aspect of the profile contains a list of specific user's interests. In more detail, this list will have concepts and categories that were referenced by each user or that were mentioned in the architecture documents. Categories represent concepts with different levels of abstraction. We preferred the usage of concepts (instead of terms), because concepts describe the context in which terms are used (i.e., semantic knowledge), and are hence more informative regarding stakeholders' interests.

For the second stage, the SAD is divided into related units of text, called *document sections*. These sections are already predetermined by the SAD template structure. Then, the Profiler computes a similarity measure between the profiles and the sections matching those profiles (see sub-section 4.2). The output is a set of links between the users and parts of the original document. We model each link as a weighted relation. Relations can be grouped per user and sorted in descending order by their weights. The resulting ranking of relevant sections per stakeholder is shown to the documenter.

4.1 The Concept-Mining Pipeline

We have designed a technique for extracting concepts from unstructured text. This technique is mainly oriented to address the dynamic aspect of the stakeholder profile. The inputs can be any kind of textual information generated by the target users, such as electronic conversations, opinions on the Web, or comments on Wiki pages. The output is a set of user profiles, each one containing a ranking of the most relevant concepts (for that user) as well as categories for those concepts. Since this profile is based on topics that are regularly mentioned by the user, the information can be considered as an approximation of the user's interests.

The technique is implemented as a semi-automated pipeline with six filters (Figure 2). In step 1, a parsing module identifies the involved users and their messages. In step 2, noisy text is pre-processed to prepare the user messages for future analysis. This processing involves: i) deletion of references to users by their names, ii) filtering of stop-words, and iii) application of a Porter's based stemming algorithm.

In step 3, entities are identified from messages. An entity can be seen as a group of semantically-related terms. To this end, we use two text mining tools provided by the Stanford NLP Group[2]: a named entity recognizer and a POS (Part-Of-Speech) tagger. The recognizer relies on a classification-based approach to detect named entities, like persons, institutions, artifacts or any kind of proper nouns. The POS tagger automatically assigns a grammatical label to every word in a sentence. We are focused on nouns and their modifiers. By grouping the results of both techniques, each user's message is associated to a set of entities.

[2] See `http://nlp.stanford.edu/`

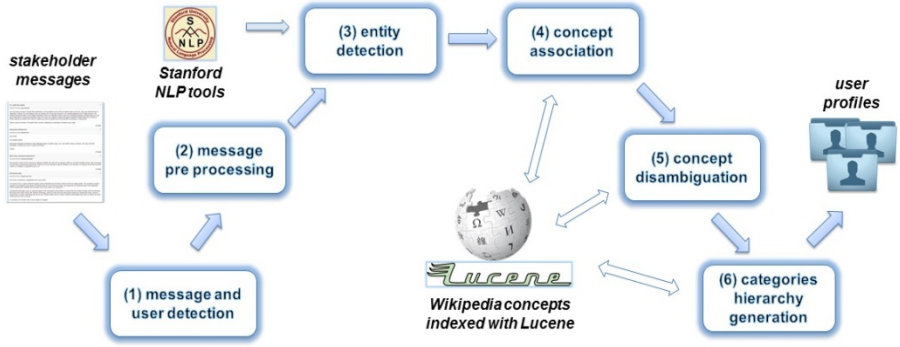

Fig. 2. The concept mining pipeline

In step 4, the concept association is performed. We use a semantic dictionary containing concepts of human knowledge based on Wikipedia. The unstructured information from Wikipedia is indexed with Lucene[3]. Since the domain is limited to Software Architecture concepts, we customized the indexes to work on Software Engineering topics (e.g., categories like: software architecture, software quality, and concepts such as: design pattern, performance, etc.). The entity names are matched against the Wikipedia concepts, using the TF.IDF measure for comparison. For a given entity, the first N concepts returned by the search are mapped to that entity.

In step 5, ambiguous concepts are handled using a disambiguation index. We use an adapted version of Lesk's algorithm, which considers a window of N nearest concepts to the ambiguous one (with N=2). The description of each disambiguation concept is compared to the descriptions of the nearest concepts using the cosine comparison function. The most related concept replaces the ambiguous one.

Finally, in step 6, a category hierarchy is built for each concept. Initially, we associate first-level categories to concepts, and then, we establish relations with higher level categories in order to build the hierarchical structure. Since categories can be quite general (e.g. SOFTWARE) and it takes considerable time to compute them, we decided to limit the hierarchy tree depth to 3 levels.

4.2 Linking Users to Document Sections

The first module of the pipeline was adapted to take a textual document as input in order to produce a document description, which actually resembles a user profile. We refer to such representation as a *section profile*. After the pipeline is executed, the representation will contain the concepts and categories involved in each document section. By doing so, we treat the problem of linking users to sections as the computation of a similarity function between two profiles: a user profile and a section profile. To compute the similarity, we apply the CF.IDF measure. Based on TF.IDF, the CF.IDF measure uses the concept frequency and the inverse document frequency

[3] See `http://lucene.apache.org/`

to determine the relevance of a concept in a collection of documents (or sections, in this case). A variation of this measure is used for dealing with categories.

For each user profile, we compute its similarity with the section profiles. The similarity score between the user profile and one section profile is estimated using Equation 1:

$$sim(up, sp) = \sum_{i \in N} confidf(ucon_i) + \sum_{j \in M} catfidf(ucat_j) \quad (1)$$

with *N:* number of concepts, *M:* number of categories, *confidf(x):* CF.IDF value for concept *x*, and *catfidf(y):* CF.IDF value for category *y*. Then, we sort the associated sections (in descending order) according to the similarity scores. Using a threshold strategy, we finally define the number of sections that are considered relevant for each user. The best value for the threshold was determined through the evaluation procedure (see sub-section 4.3).

4.3 Experimental Evaluation

We empirically evaluated the pipeline above with the goal of assessing how well the user-section links were generated. To simulate architectural documents, we selected 2 articles (called dataset A and dataset B) from InfoQ.com related to software architecture topics. The dataset A was a document that contained 18 sections and 2095 words, and 14 users participated in the discussion. Data set B contained 14 sections, 1126 words, and 17 users participating in the discussion.

Both datasets were tagged in advance. For each user, an expert annotated the potentially relevant sections according to the user's comments in the discussion thread. Afterwards, we ran our pipeline on the two datasets and computed the confusion matrixes for both cases. To assess the results, we worked with standard measures such as: accuracy, precision, recall, and F-measure [5]. We also studied the effects of two parameters: the relevancy threshold and the inclusion of categories for the CF.IDF measure. If the categories are excluded from this measure, the similarity between two profiles is computed by Equation 2 (a simplification of Equation 1).

$$sim(up, sp) = \sum_{i \in N} confidf(ucon_i) \quad (2)$$

On the other hand, we developed a term-based approach in order to have reference results for our concept-based technique. The approach used the TF.IDF measure to compute the similarity between user messages and section contents. We estimated the same measures mentioned above. Table 1 shows the results for the two experiments. The table rows are the approaches grouped by different threshold values. The table columns are the measures per dataset. At first sight, the F-measure reached the highest values using a threshold of 0.2-0.3 in both datasets (see Figure 3). Therefore, by considering the first 20-30% of the linked sections as relevant, we achieved a balance in the tradeoff between precision and recall. In all the cases, the performance of the CF.IDF techniques was better than that of the TF.IDF one, although no major differences were observed. The precision improvement with CF.IDF was around 4-7%, whereas the accuracy was improved by 2-4 %. As regards the F-measure, CF.IDF outperformed TF.IDF by 3.5-6.5%.

Table 1. Experimental results for data sets A and B

		A - Are you a software architect?				B - SOA certification			
		accuracy	precision	recall	fmeasure	accuracy	precision	recall	fmeasure
threshold 0.1	TF-IDF	0.706	0.821	0.252	0.564	0.580	0.706	0.114	0.345
	CF-IDF(-C)	0.722	0.893	0.272	0.611	0.588	0.765	0.124	0.374
	CF-IDF(+C)	0.714	0.857	0.262	0.587	0.580	0.706	0.114	0.345
threshold 0.2	TF-IDF	0.738	0.732	0.445	0.647	0.613	0.647	0.308	0.529
	CF-IDF(-C)	0.770	0.804	0.491	0.711	0.630	0.686	0.334	0.564
	CF-IDF(+C)	0.754	0.768	0.471	0.68	0.588	0.588	0.282	0.481
threshold 0.3	TF-IDF	0.738	0.686	0.522	0.644	0.634	0.647	0.408	0.578
	CF-IDF(-C)	0.754	0.714	0.542	0.67	0.626	0.632	0.407	0.567
	CF-IDF(+C)	0.754	0.714	0.546	0.671	0.576	0.544	0.351	0.488
threshold 0.4	TF-IDF	0.738	0.633	0.677	0.64	0.609	0.569	0.543	0.562
	CF-IDF(-C)	0.762	0.663	0.705	0.67	0.601	0.559	0.536	0.552
	CF-IDF(+C)	0.683	0.561	0.597	0.567	0.567	0.520	0.498	0.513

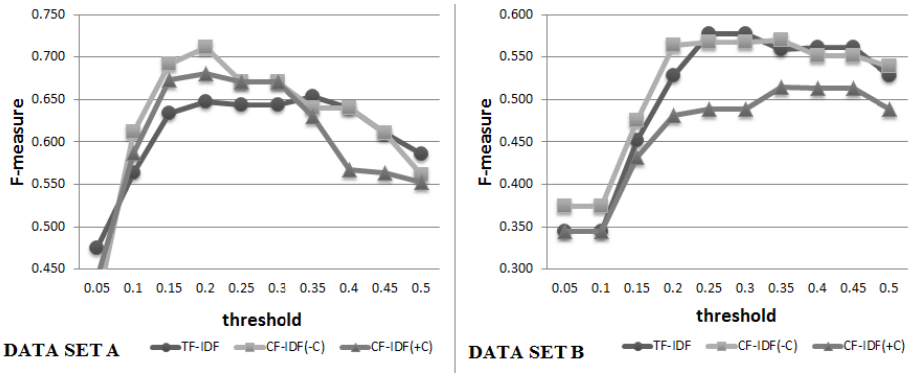

Fig. 3. F-measure comparison

The inclusion of the CF.IDF measure for categories seemed to affect the computations, as evidenced by the slightly lower results with CF-IDF(+C) than with CF-IDF(-C) ones. Despite this difference, having categories in the user-section linking process was supposed to perform better than the standard CF.IDF measure for concepts. We believe this problem might have been caused by the use of categories with a high level of abstraction (currently 3 levels). Overall, the proposed pipeline achieved reasonable results. For a threshold of 0.2, the CF.IDF technique obtained a precision of 68-80%, and a recall of 33%-49%. Thus, we plan to further develop this technique in the context of our general approach (see Figure 1).

5 Conclusions and Future Work

In this article, we proposed a general approach for implementing a stakeholder-centric and agile architecture documentation process. At the core of our approach is the personalization of architectural documents based on the profiles of the stakeholders consuming those documents. These profiles are built on top of existing stakeholders'

characterizations (with respect to architectural knowledge) as well as on text mining techniques. To this end, we have presented a tool that executes a topic-based mining process for constructing user profiles and for ranking the sections (of potential interest to these users) of an architectural document. We argue that such a mining process has advantages in terms of understandability and semantic information for the profiles, when compared to traditional keyword-based mining techniques.

A preliminary tool evaluation using topics has shown promising precision and recall, allowing us to conjecture that the stakeholders' profiles can really assist the documenter in his/her documentation tasks. Nonetheless, our profiling tool is at a prototype stage, and it still needs improvements, mainly on aspects of performance and architecture-related knowledge. We are currently testing the topic-based mining technique with different datasets and types of stakeholders. We are also exploring alternative strategies for handling the concepts and their categories.

As future work, we will integrate the profiling tool with the remaining components of the approach. To feed the profiler, we will investigate techniques and technologies for monitoring stakeholders' activities within a network and collect more data about them. We will also consider the application of Artificial Intelligence planning techniques to support the documenter in selecting the right tasks for the next version of the architecture documentation.

Acknowledgments. This work has been partially supported by ANPCyT (Argentina) through Project PICT Bicentenario 2010 No. 2247.

References

1. Bachmann, F., Merson, P.: Experience using the web-based tool wiki for architecture documentation. Technical Report TN-041, CMU-SEI (2005)
2. Clements, P., et al.: A practical method for documenting software architectures. In: ICSE (2003)
3. Farenhorst, R., Izaks, R., Lago, P., Vliet, H.: A just-in-time architectural knowledge sharing portal. In: Proc. of WICSA 2008, pp. 125–134. IEEE Press, New York (2008)
4. Garlan, D., et al.: Documenting Software Architectures: Views and Beyond. Addison-Wesley, New York (2010)
5. Goossen, F., et al.: News personalization using the CF-IDF semantic recommender. In: Proc. of WIMS 2011, pp. 10–12. ACM Press, New York (2011)
6. IEEE. IEEE recommended practice for architectural description of software-intensive systems. Technical report 1471 (2000)
7. Jansen, A., Avgeriou, P., Van der Ven, J.: Enriching software architecture documentation. Journal of Systems and Software 82(8), 1232–1248 (2009)
8. Koning, H., Vliet, H.: Real-life IT architecture design reports and their relation to IEEE Std 1471 stakeholders and concerns. Automated Software Eng. 13, 201–223 (2006)
9. Schiaffino, S., Amandi, A.: Intelligent User Profiling. In: Artificial Intelligence, pp. 193–216. Springer, Berlin (2009)
10. Su, M.T., Hosking, J., Grundy, J.: Capturing architecture documentation navigation trails for content chunking and sharing. In: The 9th IEEE/IFIP (WICSA), pp. 256–259 (2011)
11. Su, M.T.: Capturing exploration to improve software architecture documentation. In: Proceedings of ECSA 2010, pp. 17–21. ACM Press, New York (2010)
12. Tang, A., et al.: A comparative study of architecture knowledge management tools. Journal of Systems and Software 83, 352–370 (2010)