Capturing Tacit Architectural Knowledge Using the Repertory Grid Technique (NIER Track)

Dan Tofan
University of Groningen
d.c.tofan@rug.nl

Matthias Galster
University of Groningen
m.r.galster@rug.nl

Paris Avgeriou
University of Groningen
paris@cs.rug.nl

ABSTRACT
Knowledge about the architecture of a software-intensive system tends to vaporize easily. This leads to increased maintenance costs. We explore a new idea: utilizing the repertory grid technique to capture tacit architectural knowledge. Particularly, we investigate the elicitation of design decision alternatives and their characteristics. To study the applicability of this idea, we performed an exploratory study. Seven independent subjects applied the repertory grid technique to document a design decision they had to take in previous projects. Then, we interviewed each subject to understand their perception about the technique. We identified advantages and disadvantages of using the technique. The main advantage is the reasoning support it provides; the main disadvantage is the additional effort it requires. Also, applying the technique depends on the context of the project. Using the repertory grid technique is a promising approach for fighting architectural knowledge vaporization.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**]: Software Architectures - *Languages*; D.2.10 [**Software Engineering**]: Design - *Representation*

General Terms
Design, Documentation, Human Factors

Keywords
Software architecture, repertory grid, tacit architecture knowledge

1. INTRODUCTION
The software architecture of a system is the result of a set of architectural design decisions [3]. Such decisions and their rationale are an important part of the architectural knowledge (AK) of a system [1]. In practice, such knowledge is often lost, or vaporized [3], resulting in problems like increased maintenance costs and design erosion [3].

Many regard tacit knowledge as knowledge that is difficult for a person to express [1], e.g., how to ride a bike. Busch [4] offers a comprehensive list of definitions of tacit knowledge, collected from more than 100 sources, across many disciplines. In the software architecture field, examples of tacit AK include unrecorded design decisions made by an architect based on his or her experience, the design rationale, or assumptions [1]. Examples for explicit AK are documented patterns, standards, views and reference architectures [7]. Explicit AK is easily communicated through documents. However, tacit AK is more difficult to record, thus also more likely to become lost. Furthermore, AK vaporizes when tacit AK never becomes explicit (i.e., is never recorded).

Several approaches have been proposed to convert tacit AK to explicit or documented AK to prevent its loss. For example, the ISO/IEC 42010 standard [11] provides recommendations for architectural descriptions. Clements et al. [5] describe how to create architecture documentation using views. Kruchten et al. [13] argue for a decision view to document decisions. A summary of approaches to manage AK can be found in [1].

All current approaches to manage AK are based on a particular notion about knowledge within the architecting community. De Boer and Farenhorst [2] provide an overview of different notions of AK, which all refer to the dominant idea that AK is 'Design Decisions + Design'. For example, one notion is that AK is 'the integrated representation of the software architecture […] along with architectural decisions and their rationale external influence and the development environment' [2]. However, current work does not consider the complexity of tacit AK, caused by the impact of human factors on tacit AK (i.e., experience, values, subjectivity, etc.). Therefore, we argue that the architecture community should look beyond the traditional way of understanding knowledge in software architecture. Thus, we propose to employ theories from the knowledge engineering domain to provide approaches for preserving tacit AK.

Specifically, we introduce a new perspective on AK originating from the personal construct psychological theory [12]. Based on this theory, the repertory grid technique (RGT) has already been used in knowledge engineering to capture experts' knowledge [9]. Therefore, we suggest using RGT to capture architecture design decisions. In the next section we give a brief overview of the RGT. In section 3 we describe an empirical study for our idea. We discuss conclusions and future work in section 4.

2. THE REPERTORY GRID TECHNIQUE
According to the personal construct psychology theory, humans create representations of their experiences in their minds. These representations use contrasting poles or 'constructs' [12], which characterize 'elements'. For example, let us consider someone who needs to decide at which restaurant to have dinner. Each

ICSE'11, May 21–28, 2011, Waikiki, Honolulu, HI, USA

considered restaurant (i.e., alternative) is an element. The person may use constructs like 'quiet vs. noisy' or 'good vs. bad food' to characterize restaurants. Of course, a different person may have a completely different set of elements (i.e., alternatives) or constructs (i.e., criteria to characterize alternatives).

The RGT provides a systematic approach to capture such constructs from an individual. Here, the first step is to set up a topic (i.e., decide on a restaurant). Next, a set of elements (i.e., alternatives) is elicited from that person (i.e., restaurants names). The subject must be knowledgeable about all elements. Afterwards, the constructs (i.e., decision criteria) are elicited, usually by applying the triadic approach: selecting three elements, and asking in what way two of them are alike (e.g., restaurants A and B are 'quiet'), but different from the third one (e.g., C is 'noisy'). Next, the subject rates each element against each construct, on a predefined scale. Based on these ratings, tools like WebGrid [10] provide means to analyze the grids. For a more detailed description of RGT, and its applications in software engineering, please refer to [8] and [6].

3. EXPLORATORY STUDY

3.1 Study Design

To understand the applicability of using the RGT for capturing architectural knowledge, we performed an exploratory empirical study. We wanted to understand the feasibility of using RGT to capture tacit AK, and generate hypotheses for future experiments. Therefore, our research question is: *What are the advantages and disadvantages of the Repertory Grid Technique for capturing architectural knowledge?*

We used seven subjects with diverse backgrounds and levels of technical experiences. All were undergraduate or graduate students at the University of Groningen in Netherlands. Only one subject had significant industry experience. The first author of this paper interviewed each of them individually, using the following study structure for each session.

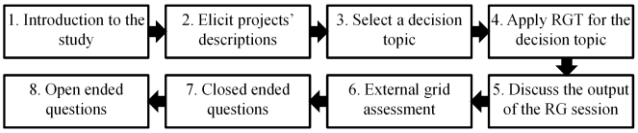

Figure 1. We performed the above steps with each subject.

In the introduction of the study (step 1) we presented each subject with the plan for the session and ethical considerations. We asked about recent projects in which the subject had to take architectural decisions (step 2). Afterwards, we selected the decision topic (step 3) for which the person was both knowledgeable and able to provide alternatives. As part of the RGT session (step 4), we elicited elements, constructs and ratings for the decision topic.

Next, we discussed the output of the grid session with each subject (step 5). Figure 2 shows an example of such output. Then we presented each subject with a grid output created by a different person and a different decision topic (step 6). We asked the subject to indicate the most and least likely alternatives to be selected by the author of the grid. We did that to verify if a person can understand the rationale of a decision based on a grid created by another person. Next, each subject filled in a questionnaire (step 7), providing background information and feedback on the RGT session. In the final step we used a set of open questions to discuss the benefits and drawbacks of the grid session. Each session was audio and screen recorded, with the consent of the participants, using an off the shelf tool.

For analyzing the final step of our study, we transcribed all discussions. Next, two of the authors applied open coding to the transcripts. Several topics emerged during the analysis, which were used as codes. After coding, we used frequency analysis to identify how often codes appeared. Afterwards, we grouped codes into categories. Then we discussed the outcomes of the codings from both researchers to resolve disagreements. We mapped the agreed categories to advantages, disadvantages and the application context of the technique.

For analyzing the output of the RG exercise, we used the WebGrid [10] web application. We asked subjects to write each decision alternative on a paper card, which could be easily shuffled on the table during the elicitation of the characteristics. Meanwhile, we used WebGrid ourselves to enter the data, while the subject could see the tool's user interface on an additional monitor. The raw data from the data collection is available at [14].

3.2 Study Results

Table 1 contains an overview of the outcome of the first three steps of our study. The first column has the ID of each participant. The second column shows their current degree program. In the fourth column, 'academic' means that the project was part of a larger research effort in which the participants were involved. The 'course' type refers to projects that were assignments within a Software Architecture course, that ID 5 and 6 took shortly before our study. 'Industrial' refers to a commercial project.

Table 1. Decision topics examined with the RGT.

ID	Study	Project description	Proj. type	Decision Topic
1	Ph.D.	Network analysis metrics for power grids	Academic	Graph theory library
2	B.Sc.	Visualization of architectural design decisions	Academic	Technology for data visualization
3	Ph.D.	Support medical disease treating	Academic	Features for the first release
4	B.Sc.	Same as ID 2	Academic	Database selection
5	M.Sc.	Smart Grid application	Course	Main hardware
6	M.Sc.	Same as ID 5	Course	Main hardware
7	Ph.D.	E-banking reporting system	Industrial	Reporting engine

The RGT sessions (step 4) took on average 57 minutes, with a standard deviation of 16 minutes. As an example, Figure 2 shows the output of such a session for ID1, as analyzed by WebGrid, using hierarchical clustering analysis. It groups similar elements (e.g., Prefuse and Infovis), based on the similarity of their ratings given to the constructs. The same can be seen for other constructs, e.g., '*more complex vs. simple to use*' is grouped with '*used only for visualization vs. used for graph analysis*'. The ratings of the two constructs differ only for the Leda element. If a person regards two constructs as very similar, then these constructs cannot be used to differentiate elements. Repeating such grouping produces the dendrograms for the elements and for the constructs. The scale above each tree shows the similarity score. For example, ID1 perceives Prefuse and Infovis as around 95% similar, with

regard to the ratings of the criteria (i.e., constructs) used to evaluate each option. They form a cluster which is similar around 85% to Leda. If a person regards two elements as very similar, then s/he cannot distinguish them, thus potentially selecting any of the two, when considering a decision topic.

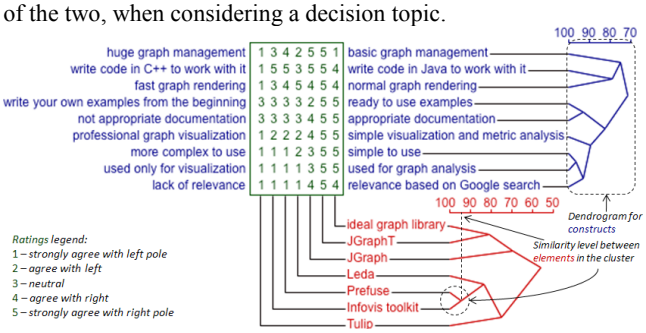

Figure 2. The grid from ID1 shows the elements, constructs, ratings and dendrograms for his decision topic.

The constructs and the ratings represent the subjective perspectives of the participants on the elements. It is interesting to check whether there is a link between the alternative predicted by the grids, and the actual decision made in the project by the subject. For example, in Figure 2, JGraphT is the closest to the ideal graph library, so JGraphT is the alternative to be chosen, as predicted by the grid. The subject confirmed that JGraphT was also the actual decision he made.

Overall, in five out of seven decision topics, the real decision matched the one predicted by the grid. The two mismatches were ranked as third and forth alternative predicted by the RGT. While discussing the output of the RGT session (step 5), the subjects were pleasantly surprised by such matches, as they got a confirmation of making the right choice. For one of the mismatches, the subject noticed that the equal weights of the constructs may be the reason for that. For us, the high number of matches indicates that the elicited content is indeed AK, as it includes the rationale for the architectural decision.

For the external grid assessment step (step 6), we showed each subject an output like the one in Figure 2, created previously by a different person, for the decision topic of choosing a programming language. We asked them to indicate the most and least likely alternative to be selected by that person. Each subject identified the expected answer in a few seconds, by looking at the similarities between the ideal element and all the other elements. This suggests that the output is useful for communicating alternatives of an architectural decision in a concise manner. Additionally, numbers can be associated to the alternatives. For the example in Figure 2, WebGrid indicates the similarities of the alternatives with the ideal graph library as: 81% with JGraphT, 61% with JGraph, 44% with Leda, 39% with Prefuse and Infovis and 31% with Tulip. This can be used to rank the alternatives.

For step 7, each subject rated some statements from 1 (strongly disagree), 3 (neutral) to 5 (strongly agree). Table 2 summarizes the results, which suggest the applicability of the RGT for explaining, documenting and making architectural decisions. Additionally, we asked each subject to indicate the most difficult part of the RGT exercise or session. Out of the seven persons, five indicated constructs elicitation through the triadic approach. The other two pointed to the rating step.

Table 2. The values indicate the average and standard deviation of the subjects' ratings for each statement

Statement	Values
The output of the exercise helps explain to a colleague why I took a certain decision.	4.14 1.07
The exercise provides a structured way of explaining architectural decisions.	4.14 0.38
The exercise provides a structured way of documenting architectural decisions.	3.57 0.53
The exercise provides a structured way of making architectural decisions.	4.00 1.29
When trying to understand why another architect took a certain decision, I could use such an output.	4.29 0.49
I found the exercise too tiring for the output.	2.43 1.13
Overall, I enjoyed the exercise.	4.42 0.53

In the following subsections, we discuss the results from analyzing the interview transcripts using open coding. Also, we apply consistency checks on the extracted themes with other steps of the study, to increase its validity.

3.2.1 Advantages

By applying the open coding procedure and its supporting steps as described in the study design, we identified reasoning support, as the main category. It comprises the following codes: *systematic* (indicated by 2 out of 7 persons), *new insights* (3), *reflective* (4) and *decision support* (5). As examples, for the *systematic* code, one subject characterized the approach as 'very professional', while the other one as 'a formal way to document what you have in mind'. For *new insights*, a person said: 'I also like that you could say these concerns or these elements are very alike. Hey! This is true. I didn't think at it that way'. For *reflective*, we quote: 'it forces you to think about why you chose something'. These perceptions are in line with the averages of the ratings for statements 2, 3 and 4, from Table 2.

The other category is readability, containing the *picture output* (5 mentions) and *conciseness* (2) codes. For the picture output, one said: 'I like that my choices […] have been documented […] in a graph that justifies my choice'. For the other code, a person indicated that 'it provides much information with less text'. These perceptions resonate with the statements 1 and 5.

3.2.2 Disadvantages

The main category concerns effort. It comprises four codes: *learning curve* (1 out of 7), *straining* (2), *tool support* (3) and *time consuming* (4). For the *learning curve*, the person commented that 'you need to tell the people how to read the results'. For *straining*, a subject noted that the RG session proved 'difficult, because you have to think so much'. The comments on *tool support* asked for a friendlier interface and adding weight to the constructs. However, for *time consuming,* the subjects observed that 'I disliked it took too long' and that 'it takes some time that I wouldn't spend'.

3.2.3 Application Context

Additionally, we identified when and for whom RGT might be useful, based on three codes: *size* (3), *time* (2) and *role* (6). For

project *size*, a subject noted that 'for my small project I won't go for [RGT], for a big project I would'. For *time*, one considered RGT useful 'at the very beginning of the project'. For *role*, the subjects had mixed opinions whether developers, testers or maintainers can use the output of the RGT. A subject commented: 'not sure about the typical developer using such output […] testers and maintainers not at all […] yes, for a developer promising to become an architect'. During the sessions, we noticed more limitations of the RG technique. First, minimum five alternatives are needed for a decision topic. Next, the decision topic under focus is poorly related to others, which is often not the case in practice. We will address these issues in future work.

3.3 Study Limitations

External validity: Our results are not generalizable due to the small sample size and the background of participants. Additionally, experienced practitioners may perceive differently the technique and its output.

Internal validity: Due to its exploratory nature, the risk of internal validity was low. However, to ensure that our results emerged from the collected data, we checked the consistency and plausibility of the outcomes from the different steps of the study.

Reliability: To ensure reliability of our study results, we piloted the data collection instruments and reviewed the study protocol. Also, parts of the data analysis were performed by more researchers, followed by cross-checking the results.

Construct validity: Our subjects have never worked as real architects. This might have influenced the criteria used for deciding between decision alternatives. Additionally, real architects may have different perceptions on the technique.

To further validate our study, we used two criteria recommended by Edwards et al. [6]: design decisions for data collection, and data gathering process. For the first criterion, we chose to elicit elements, constructs, and ratings from each subject, instead of providing them. We believe that our approach is better suited for exploratory research [6], as the subjects provided all the content of the grids. Unlike other RGT studies, for ratings we used a 5-point scale which provided enough discrimination among the elements and constructs.

For the second criterion, we chose to conduct long, individual sessions with each participant, instead of short ones. This allowed each subject to thoroughly reflect on perceptions and decisions, necessary for eliciting tacit AK. One researcher performed data collection during the sessions with each participant, introducing bias risks, but also facilitating the elicitation. Moreover, the usual limitations of exploratory studies also apply to ours, e.g., no hypothesis, or lack of triangulation.

4. CONCLUSIONS AND FUTURE WORK

We propose an inter-disciplinary approach for capturing AK, by applying a theory from outside of the software architecture domain. The personal construct psychology offers the repertory grid technique, which we used in an exploratory study to capture tacit AK. Furthermore, we identified some advantages and disadvantages of the technique. Its reasoning support and readability advantages recommend it as a complementing approach to existing ones. Its effort drawback is also specific to other approaches. Therefore, the technique shows a high potential for capturing tacit AK, thus reducing AK vaporization.

As future work, we plan to use the repertory grid technique in an industrial context to understand its applicability in a real-world setting, and improve tool support for it. We have a preliminary hypothesis that the output of the grid requires less time for understanding the rationale of a decision, compared to existing approaches, like the decision view or textual decision templates.

5. ACKNOWLEDGMENTS

This research has been partially sponsored by NWO SaS-LeG, contract no. 638.000.000.07N07. We thank Tim Menzies and the study participants for their help.

6. REFERENCES

[1] Babar, M.A., Dingsøyr, T., Lago, P., and Van Vliet, H. *Software Architecture Knowledge Management: Theory and Practice*. Springer-Verlag New York Inc, 2009.

[2] de Boer, R.C. and Farenhorst, R. In Search of 'Architectural Knowledge.' In *Proceedings of the Third SHARK Workshop*, (2008), 71–78.

[3] Bosch, J. Software architecture: The next step. In *Proceedings of First European Workshop on Software Architecture (EWSA 2004),* (2004), 194–199.

[4] Busch, P.A. 2004. *Knowledge Management Implications of Articulable Tacit Knowledge: Case Studies on its Diffusion*. Doctoral Thesis. Macquarie University.

[5] Clements, P., Garlan, D., Bass, L., et al. *Documenting software architectures: views and beyond*. Pearson Education, 2002.

[6] Edwards, H.M., McDonald, S., and Young, S.M. The repertory grid technique: Its place in empirical software engineering research. *Information and Software Technology* 51, 4 (2009), 785-798.

[7] Farenhorst, R. and de Boer, R.C. 2009. *Architectural Knowledge Management: Supporting Architects and Auditors*. Doctoral Thesis. VU University, Amsterdam.

[8] Fransella, F., Bell, R., and Bannister, D. *A Manual for Repertory Grid Technique*. Wiley, 2004.

[9] Gaines, B. and Shaw, M. Knowledge acquisition tools based on personal construct psychology. *The Knowledge Engineering Review*, (1993).

[10] http://gigi.cpsc.ucalgary.ca:2000/, accessed on November 2010.

[11] ISO/IEC. ISO/IEC CD1 42010 IEEE. *Systems and software engineering — Architecture description*, 2010.

[12] Jankowicz, D. Why does subjectivity make us nervous? Making the tacit explicit. *Journal of Intellectual Capital,* 2, 1 (2001), 61–73.

[13] Kruchten, P., Capilla, R., and Dueñas, J.C. The decision view's role in software architecture practice. *IEEE Software*, 26, 2 (2009), 36–42.

[14] http://www.cs.rug.nl/~dan/NIER-ICSE2011/, accessed on December 2010.