Understanding and Supporting Software Architectural Decisions
Tofan, Dan

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2015

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 8

Tool Support for REGAIN and GADGET


In this chapter, we present tool support for the REGAIN (see Chapter 6) and GADGET (see Chapter 7) approaches. The tool helps architects and other stakeholders capture and analyze decisions. In addition, the tool helps architects make group decisions. The tool is based on the theoretical and conceptual foundations created and evaluated in Chapters 6 and 7. We developed the tool as a research tool in an academic environment, and we used the tool with industrial practitioners, who offered us feedback and influenced the development of the tool. The tool is web-based and available as an open source project. In this chapter, we highlight the motivation, features, and development aspects of the tool.
8.1 Introduction

In Chapter 6, we presented and validated the REGAIN approach for making and capturing architectural decisions, based on the Repertory Grid technique. In Chapter 7, we presented and validated the GADGET approach for making and capturing group architectural decisions. To make the results of Chapters 6 and 7 more applicable, to support architects in practice, and to facilitate the transfer of our research results to industry, we developed a web-based tool to support capturing and making architectural decisions.

This tool provides several benefits. Mainly, the tool helps architects make better decisions by providing means for using the REGAIN and GADGET approaches, which improve representing, analyzing, and communicating decisions throughout the software development and life cycle. In turn, by using REGAIN and GADGET with the tool, vaporization of architectural knowledge is reduced.

Other tools for architectural decisions have been proposed, as reported in the systematic mapping study in Chapter 4. For example, Zhu et al. support trade-off analyses between different decisions (Zhu et al., 2005). PAKME is a web-based tool for capturing architectural knowledge (Babar et al., 2008). Also, Ali Babar et al. investigated web-based evaluation of architectural decisions (Babar et al., 2006). Tang et al. compared tools for architectural knowledge management (Tang et al., 2010), and reported the need for better support to facilitate knowledge sharing and collaborative work. More recent tools include an Enterprise Architect plugin for capturing architectural decisions (Manteuffel et al., 2014) and a web-based tool for group architectural decisions (Nowak and Pautasso, 2013b). We further motivate the development of a new tool in the next section.

8.2 Motivation for a new tool

We implemented a new tool, rather than reusing or extending existing tools, for the following seven reasons:

1. In Chapters 6 and 7 practitioners indicated the need for tool support for REGAIN and GADGET.
2. Lack of user-friendly, dedicated tool support for REGAIN and GADGET. We used other tools in previous studies (e.g. WebGrid (Gaines and Shaw, 2007), Idiogrid (Grice, 2002)). However, these tools were not user-friendly for users not familiar with the concepts of the Repertory Grid Technique, and needed modifications to be used for REGAIN and GADGET.

3. By developing a customized tool, we had the flexibility to implement concepts and validate ideas from our research (such as prioritization approaches identified in Chapter 6, and specific steps for GADGET in Chapter 7).

4. We needed tool support during some of our studies. An earlier version of the tool has been used in a previous study with industrial practitioners in Chapter 7.

5. We aimed at long term benefits of our tool. This required evolution and adaptation of the tool over time. Thus, our initial design decision was to offer this tool as an open source project, with source code freely available to the software architecture community, including industrial organizations.

6. Most existing architecture tools are desktop applications. Based on early feedback from practitioners, we developed a web-based tool to allow a wider distribution and easier use (i.e. practitioners do not need to install anything, to respect the corporate rules on what can be installed on their workstations).

7. We wanted to facilitate the transfer of research results (i.e. REGAIN, GADGET approaches) to the industry.

8.3 Features

The features of the tool focus on capturing decisions, decision analysis, and group decision making.
8.3.1 REGAIN Support

The screenshots below illustrate how the tool supports REGAIN, using an architectural decision that we had to make for the development of the tool itself.

In the upper part of Figure 8.1, we notice the main menu of the application. Clicking the Profile link takes the user to a page that allows the user to add some basic optional personal details like first name, last name, and phone number. The Grids link allows the user to manage the usage of the REGAIN approach. The Sessions link allows the user to manage the usage of the GADGET approach. The main menu is visible from all pages of the tool.

After clicking on the Grids link, the user can start a new REGAIN session, like in Figure 8.1, in which we can see the topic of the decision: choose the programming language for the tool support. The optional description contains more details on the topic.

Figure 8.1. Indicate topic for the decision and an optional description.

In Figure 8.2, step 2 of REGAIN is illustrated: alternatives for the decision can be added or removed.
8.3. Features

Figure 8.2. Indicate alternatives for the decision.

Figure 8.3 illustrates the third step of REGAIN, in which concerns are indicated. Concerns can be added directly. In addition, the triadic elicitation of concerns can be used (for details, see Chapter 6 and (Jankowicz, 2001)): the user drags and drops two similar alternatives on the left panel, and a contrasting alternative on the right panel. Next, the user indicates how the two alternatives are similar and different from the alternative on the right.

Figure 8.3. Indicate concerns.
To prioritize concerns, the hundred-dollar approach (i.e. assign a number between 1 and 100 to each priority to reflect the importance of the concern, while ensuring that the sum of priorities is 100) is implemented, following the study in Chapter 6. The priorities can be indicated directly, as visible in Figure 8.4.

Figure 8.4. Prioritize concerns.

Figure 8.5 shows how the user rates each alternative against each concern (i.e. step 4 in REGAIN).

Figure 8.5. Each alternative is rated against each concern.

Figure 8.6 shows a summary of the architectural decisions: topic, alternatives, concerns, ratings, and priorities. In addition, the output of the hierarchical cluster analysis shows that Python is closest to the ideal alternative. Python was the output of this decision. In addition, we notice two other types of
8.3. Features

analysis: similarity analysis (i.e. similarity levels among each pair of alternative and each pair of concerns), and principal component analysis (a feature which is not fully implemented yet).

Figure 8.6. The alternatives, concerns and ratings are summarized in the upper part of the figure. In the lower part of the figure, the hierarchical cluster analysis shows similarities among alternatives and concerns.

8.3.1.1. Concerns Prioritization

Figure 8.4 shows that priorities of concerns can be indicated directly by the user, using the hundred-dollar approach. However, feedback from practitioners indicated that indicating directly priorities is cumbersome. Therefore, we implemented a feature that makes it easy for practitioners to prioritize concerns with the hundred-dollar approach.

Users can adjust the priorities by dragging the sliders for concerns. If the slider for a concern is dragged to the right, the tool automatically increases the priority of that concern, and decreases proportionally the priorities of all other concerns, so that their total sum remains 100. The screenshots in Figure 8.7
illustrate how increasing the priority of the low cost concern decreased the other priorities proportionally with their initial priorities.

![Grid Management](image)

**Figure 8.7.** User-friendly prioritization feature: increasing priority of low cost concern (left) decreases priorities of other concerns (right).

### 8.3.2 GADGET Support

As found in Chapter 3, and in the work of other researchers (e.g. (Nowak and Pautasso, 2013a; Smrithi R.V. and H., 2014)), many architectural decisions are made by groups of architects, rather than individuals. To support this, the tool implements support for GADGET. With the help of the tool, users can create decisions, share decisions, or collaboratively identify decision alternatives.

Similar to decisions made by individuals, groups of decision makers can prioritize concerns and rate alternative-concern pairs as well as indicate how they would prioritize and rate from the perspective of other stakeholders. Decision makers can see the priorities and ratings of other decision makers and thus engage in discussions and work towards consensus in several iterations, as part of using the GADGET approach.

Figure 8.8 shows a summary of using the tool for a group decision with four participants, who indicated various alternatives, concerns, ratings, and priorities for the decision in several iterations. We notice that the results from previous iterations are captured, thus offering traceability into how the group decision was made.
Figure 8.8. Summary of a group decision session on which Python web framework to use.

Figure 8.9 shows a summary of metrics for the group decision in Figure 8.8, during the iteration. The Range metric shows the difference between the highest and lowest value among decision makers. Mean and standard deviation are also calculated using the values indicated by group decision makers.

Differences among decision makers are highlighted with shades of red. Stronger shades are used for larger differences. This allows decision makers to identify immediately the differences among them and focus on increasing consensus, by focusing on their differences.

Clicking on any of the red cells offers details about the concrete values that led to differences. The lower part of Figure 8.9 is displayed, after clicking on the web2py/’semi-rapid application development’ cell. This cell has a range of three. In the lower part of Figure 8.9, decision makers can see that the participant with color green has a different perspective.
8. Tool Support for REGAIN and GADGET

8.4 Tool Development and Deployment

We implemented an open-source, web-based tool for REGAIN (presented in Chapter 6) and GADGET (presented in Chapter 7). The tool is deployed at www.repertorygridtool.com on a dedicated virtual server. It requires users to create an account, using an email address and a securely stored password. The tool allows users to access their own content and decisions related to them. A wizard guides users through capturing decision topics, alternatives, concerns, prioritization, ratings, and analysis of decisions. As of March 2015, the tool has more than 800 registered users.

The tool is open-source and uses a permissive, industry-friendly, open source software license (i.e. MIT license). The development of the tool is ongoing and we add more features to the tool, based on the feedback we received during the interview study and from the tool users.
The tool is developed mostly in Python and JavaScript, and it has around 20,000 lines of code. Most development took place between early 2012 and mid-2014. The tool uses the Django web framework, an open source, Python-based framework which also provides security features. The tool uses SQLite, a lightweight SQL database, but the tool can use almost any SQL database. Furthermore, the tool uses jQuery and jQuery UI libraries for the user-interface design and cross-browser compatibility. The source code for this tool is available on GitHub and Ohloh (http://www.ohloh.net/p/rgt-tool). Thus, other developers have access to the source code and can develop it further or create their own branch of the software.

Compatibility, usability, and security were key drivers for the architecture of the REGAIN tool, which is compatible with the recent versions of all major browsers: Internet Explorer, Chrome, and Firefox. Also, the tool uses the jQuery and jQuery UI libraries, for a friendly user-interface and cross-browser compatibility. The Django framework offers multiple security features, such as protection against SQL injections.

8.5 Conclusions

In this chapter, we presented an open source, web-based tool to support architects in individual and group architectural decision making. A video clip demoing the tool can be found at http://youtu.be/GAHxrlfOit70. An earlier version of the tool was featured on InfoQ (http://www.infoq.com/news/2012/07/rugrgt-tool). We expect the tool to evolve in the future based on our ongoing research efforts. Also, ongoing development focuses on increasing the usability of the tool. The issue tracker on GitHub (https://github.com/danrg/RGT-tool/issues?state=open) shows the open tickets that will be addressed in the future.

8.6 Acknowledgments

We thank the students who contributed to the tool development