An Integrated Tool for Trade-off Analysis of Quality-of-Service Attributes

Leo Hatvani
Mälardalen Real-Time Research Centre,
Mälardalen University
Västerås, Sweden
leo.hatvani@mdh.se

Anton Jansen
ABB Corporate Research
Västerås, Sweden
anton.jansen@se.abb.com

Cristina Seceleanu
Mälardalen Real-Time Research Centre,
Mälardalen University
Västerås, Sweden
cristina.seceleanu@mdh.se

Paul Pettersson
Mälardalen Real-Time Research Centre,
Mälardalen University
Västerås, Sweden
paul.pettersson@mdh.se

ABSTRACT
In this paper, we present a tool for performing trade-off analysis of Quality-of-Service attributes of design solutions resulted from architectural, behavioral, or deployment changes in service-oriented systems. The tool allows for comparing the performance, reliability, and maintainability of such solutions, in an attempt to compute the optimal one with respect to the weighted sum of the considered quality attributes. Our tool uses the Analytic Hierarchy Process for computing trade-offs, and is integrated into the Quality Impact Prediction for Evolving Service-Oriented Software IDE.

Keywords
Quality-of-Service attributes, trade-off analysis, AHP

1. INTRODUCTION
Service-oriented systems (SOS) have recently emerged as context independent component-based systems for which the offered Quality-of-Service (QoS) is an essential factor that the user takes into consideration when choosing a service out of functionally similar ones. Since any non-trivial modification made to the service architecture model inevitably influences several quality aspects, a central problem is to identify the effect of the possible changes. By performing a systematic analysis of the possible trade-offs between QoS attributes, the risk of selecting a design solution with negative impact on important quality attributes is reduced.

In this paper, we present a tool for the systematic trade-off analysis of QoS of SOS design solutions, which facilitates the comparison of the latter’s performance, reliability, and maintainability. Our tool uses the Analytic Hierarchy Process (AHP) [2] for computing trade-offs, and is integrated into the Q-ImPRESS IDE, a Quality Impact Prediction for Evolving Service-Oriented Software framework. The novelty of our approach, as compared to the work of Zhu et al. [3], is integrating AHP with different automated analysis methods of the system’s architecture and behavior.

Assuming a small set of design solutions (less than 10), the quality of a design is equated with the weighted sum over the QoS attributes. Our tool determines the corresponding weight of each attribute metric, and outputs the best design solution, while also involving the designer’s experience and preference.

2. AHP
The Analytical Hierarchy Process (AHP) [2] is a generic approach for multi-criteria decision making. Computing architectural trade-offs can be seen as a special case of such decision making, in which the different criteria are represented by the functional and quality requirements of the system. The decision making targets the selection of the architectural solution that suits best these criteria.

The AHP method consists of 4 steps: (1) create a decision hierarchy, (2) determine relative criteria, (3) assess alternatives, and (4) interpret results. The first step identifies the criteria and associated analysis results to be used in the decision making. In addition, the alternatives to be considered should be decided upon. The second step consists of pair-wise comparisons of the selected criteria, to establish the importance of each criterion. In this pair-wise comparison, two criteria are put on a weight scale and the architect has to indicate how much one of the criteria is preferred over the other, using a predefined scale. In step three, the alternatives are pair-wise compared for each criterion. For each comparison, the relevant analysis results are presented alongside a weight scale similar to the one used in the previous step. In the fourth and last step, the architect examines the resulting scores of the alternatives with each other. To improve understanding, we show the contribution of each criterion to the end result.

3. IMPLEMENTATION OF AHP METHOD WITHIN Q-IMPRESS IDE
The Q-ImPRESS IDE [2] can be used to create multiple design alternatives for SOS[1]. These alternatives are then evaluated, by

[This work was supported by the European Union under the ICT priority of the 7th Research Framework Programme in the context of the Q-ImPRESS research project.]
carrying out quality predictions for each of several QoS attributes. Our tool implements the Analytic Hierarchy Process method for evaluating trade-offs between these quality prediction results.

The Q-ImPrESS IDE operates on a limited number of QoS attributes: Response Time/Throughput, Reliability, Utilization, and Cost/Effort. The AHP method is implemented using the wizard paradigm, so our tool is named “AHP Wizard”. The AHP Wizard automatically pulls the required data from the Q-ImPrESS IDE, and asks the user to enter only the pairwise comparisons between criteria, and pairwise evaluations of values for these criteria, respectively. To simplify assessing the alternatives, we have made significant customization of the AHP method for Q-ImPrESS IDE. In this way, the tool is usable for users without AHP knowledge.

The main customization is that the AHP decision hierarchy is predefined in the tool. We have found that, with a limited number of qualities, a one level decision hierarchy is the most suitable for the tool’s purpose. Data that is used by the user is automatically pulled from the results repository of the Q-ImPrESS IDE. The comparisons that are integral to AHP Wizard are realized by a series of option buttons. Each of the option buttons is assigned to one of the following values: (i) Extremely preferred, (ii) Very strongly preferred, (iii) Strongly preferred, (iv) Moderately preferred, (v) Equally preferred. The options are displayed twice to make the display of comparisons more compact, and to logically disable inputting conflicting data.

### 3.1 AHP Wizard Architecture

In this section, we describe the architecture of the AHP Wizard, including the packages that have been used, and the link between the AHP Wizard and the rest of the Q-ImPrESS IDE.

The underlying framework of the Q-ImPrESS IDE (not including calls to external tools) is the Eclipse Rich Client Platform (Eclipse RCP). Therefore, our tool is constructed using the SWT package of Eclipse RCP, in order to align the tool with the rest of the IDE tools, and take advantage of all the benefits of Eclipse RCP.

The AHP Wizard has a fairly simple architecture of the graphical interface. Its graphical interface is based on the Eclipse JFace package (org.eclipse.jface.wizard) wizard implementation. This package contains the implementation of the WizardPage class, which
we have extended with appropriate controls, in order to provide the classes that describe each of the wizard dialogues that are used during the trade-off analysis procedure.

Besides the standard controls, we have also used the JFreeChart (http://www.jfree.org/jfreechart/) package, to enable the stacked bar graph visualization of the results, and the Velocity Engine (http://velocity.apache.org/) to provide facilities for HTML output of the trade-off results.

The Q-ImPrESS IDE is connected to the AHP Wizard by invocation only; this is done through the Result Viewer, that is, the tool for viewing the QoS prediction results, which is presented in section 4.

On the other hand, the AHP Wizard is connected to the Q-ImPrESS IDE by using the facilities of the Q-ImPrESS IDE backbone. The backbone offers API calls that enable any part of the Q-ImPrESS IDE to read and write results in the, so called, Result Repository, where all of the quality prediction results reside.

The AHP Wizard is also connected to the file system of the Eclipse Environment, to be able to ensure the data persistency over the currently selected choices. This persistency is done through a custom class developed for this purpose, and by serializing this class with a binary file. In essence, this class acts like a simple database.

4. THE USAGE OF AHP WIZARD

In order to access the AHP Wizard, the user has to mark the alternative designs that are going to be considered for the trade-off analysis. In order to make this selection as easy as possible, and to maintain a consistent work-flow, we have integrated the invocation of the AHP Wizard into the Result Viewer, which is the tool used to display the results of the already run quality predictions. This tool and its interface can be seen in Figure 1.

The selection is done by choosing the corresponding check boxes placed near the design alternatives, respectively. In the following, we describe the steps 4.1 to 4.3, which are part of using the AHP Wizard.

4.1 Comparison of Qualities

After the user has selected the design alternatives that are to be compared using the AHP method, he/she is presented with a window displayed in Figure 2. This dialogue asks for the user's preferences on the QoS attributes. There are six pair-wise comparisons that need to be made.

4.2 Comparison of Data

In the second step (as displayed in Figure 3), the AHP Wizard asks for the preferences of each comparable pair of data. It is expected that the user makes comparisons between individual values of a certain quality. For example, the user has to choose whether a response time of 8ms is better, and how much better than the one of 10ms. If the user needs more data on one specific value, such data is presented via a tool-tip that is displayed when hovering over that value.

4.3 Displaying the Trade-off Results

Upon completion of the above steps, the results are displayed, as shown in Figure 4. The trade-off analysis results are shown as stacked bar graphs. Each bar is separated into portions that corre-
Figure 4: Displaying results.

spond to zones of the total score attributed to a particular QoS attribute, multiplied by the weight of the attribute, respectively. The total height of each bar corresponds to the score that has been computed for that design alternative. After completing all of the wizard’s steps, the user can choose to return and make adjustments.

When the user is satisfied with the results, the wizard will export all of the entered data, and the analysis results, to an HTML file. This enables further processing of the results using any spreadsheet or word processing application. If the user wants to redo the analysis on the same alternative designs, the previously made choices, which have already been saved, are automatically loaded, such that they are available for further refinement. In this way, the user can adjust only the choices that have been changed, while preserving all the previously entered ones.

5. DESIGN DECISIONS

Our trade-off project’s main requirement was to create a tool, within the Q-ImPrESS IDE, which should use the Analytic Hierarchy Process to help the system designer compare various design solutions, via the trade-off analysis between different quality prediction values. In order to meet such requirement, we have made several major design decision, which we list below, as they were made, to portray the design and development cycle of the tool. The constraints under which this project has been developed are: limited number of person hours, and major changes of the underlying API during tool development.

5.1 Wizard Interface Paradigm

Considering the structure of the AHP method, one can easily notice that it consists of several steps. In addition, the method requires the user to input a large amount of data, which triggers the conclusion that a single window representation is inadequate for the job. Therefore, the wizard paradigm has proved itself to be the most convenient way to present multiple questions spread over multiple dialogues. We have also relied on a two-way flow of the wizard paradigm (that is, we rely on the wizard’s possibility of returning to previous steps).

Our favorite paradigm is similarly used in many polling applications.

5.2 Fixed AHP Tree

The first step in applying the AHP method is to create a hierarchy tree of different parameters that are included in the trade-off process (this is explained in more detail in [3]). We have chosen to implement a fixed one-level hierarchy tree, which cannot be changed by the user. This decision has been deemed appropriate as there is always a fixed set of inputs to the AHP method (quality prediction results), and they are completely independent of each other. This, in turn, has drastically reduced the number of person hours required for completing the trade-off analysis project, entailing its feasibility.
5.3 Result Representation Using Stacked Bar Graph

After creating a prototype application, we noticed the need for the visual representation of the trade-off analysis results. The simplest and most straightforward way to represent the ratio between several values is either the pie chart or the bar graph. In our case, we have also wanted to represent the data that the values result from. The quality of a design alternative is calculated as a weighted sum over performance, reliability and maintainability values, hence we have chosen to represent the trade-off data via stacked bar graphs, which show almost all the data involved in the AHP method.

5.4 Storing the results in HTML format

HTML is an open standard that can be read on almost any platform. Motivated by this argument, we have searched and discovered that the Velocity Engine by Apache is an appropriate tool for generating the needed HTML, from a template. It also allows us to easily export the image of the generated graph to a portable and open format.

5.5 Persistence of the trade-off results

While developing the tool, we have noticed that there are many situations when the user has to go out of the tool, make a fix in the design alternative, rerun quality predictions, and then re-evaluate the trade-off between the new prediction values. To simplify this route, we have implemented an entirely transparent system of saving the user’s choices, and of loading them automatically when the user reopens the tool, assuming the same design alternatives. Unfortunately, there exists the pitfall of the tool not recognizing changes of prediction values within a design alternative. We will address this deficit in future extensions of the tool, to enable alerting the user on changes.

6. LESSONS LEARNED

Prior to developing the final version of the tool, we have developed a simple prototype with full AHP method functionality, yet weak on the usability side. However, this prototype has been very useful in determining many of the design decisions that we have just described, for the current, integrated version of the tool.

The tool development has been done mostly by iterative prototyping, which has met the time and person hour constraints of the project.

We have heavily relied on already existing packages, and thus were able to integrate mature and bug-proofed code, in our tool, which has also contributed to the timely completion of the tool.

7. CONCLUSION

In this paper, we have presented a tool that provides support for the AHP-based trade-off analysis of quality prediction values for multiple design alternatives of SOS. We have shown the main issues faced during the tool implementation, and their remedies, as well as explained the integration of our AHP tool within the Q-ImPRESS IDE. We have also presented a comprehensive list of major decisions taken during the development process, together with the lessons learned from this project.

8. REFERENCES

