Maintainability Studies Investigating Aspect Preservation via Automation: Lessons Learned

Aram Hovsepyan[1], Riccardo Scandariato[1], Stefan Van Baelen[1], Yolande Berbers[1], Serge Demeyer[2], Wouter Joosen[1]

[1] IBBT-DistriNet, Katholieke Universiteit Leuven, Belgium
first.last@cs.kuleuven.be
[2] Universiteit Antwerpen, Belgium
first.last@ua.ac.be

Abstract. In the recent past the authors have conducted two experimental studies concerning the benefits of preserving modularity from design to code. In this paper we report on three key lessons learned in designing those investigations.

1 Introduction

The software engineering community emphasizes empirical research methods in order to enable the comparison of different approaches as well as to improve the validity and generalizability of research results. Obviously, it is essential to have a set of systematic guidelines on how to design, conduct, analyze, interpret and report empirical studies. Lessons learned concerning the design of such studies are equally important as these could contribute to the community and provide added value for subsequent empirical experiments.
In this paper we present three key insights that we have acquired through conducting two empirical studies that focus on the semi-automated transformation of modularized design to code in the context of software maintenance. We describe how we have leveraged on these ideas and discuss their added value. Firstly, we discuss the value of the internal attributes as predictors of software quality. Secondly, we outline the importance of using questionnaires as additional means to support the quantitative data. Finally, we propose to extensively screen the subjects and plan for blocking (or pairing), but use this information only if the unpaired statistical tests are inconclusive.
The rest of this paper is structured as follows. In section 2, we shortly describe the two empirical studies that we have conducted. In sections 3, 4 and 5 we present each important lesson learned and discuss its impact on our empirical studies.

2 Short description of the two studies

Typical aspect-oriented modeling (AOM) approaches provide means to specify and execute the composition of the modularized concerns at the modeling level to obtain a combined model. However, it is also possible to keep the concerns modularized at the implementation level by targeting an aspect-oriented platform. Therefore (at least) two alternative development processes could be used used in order to translate modularized design into a final implementation. The *aspect disrupting process* composes the aspect-oriented models and subsequently translates the composed model to object-oriented code. The *aspect preserving process* preserves the modularization and translates each modularized concern into aspect-oriented code artifact. Obviously, the choice of the development process has a direct impact on, e.g., productivity of the developers, quality of the final product, ease of maintenance. In spite of the lack of evidence, the AOM community implicitly advocates the *aspect disrupting process*. In two systematic empirical studies we have investigated the impact on maintainability in two variations of the development process [1,2]. In the first study, we have leveraged on the well known Theme/UML, Java and AspectJ as representative AOM, object-oriented programming and aspect-oriented programming techniques. In the second study, we have focused on a pragmatic AOM approach that allows the combination of domain-specific modeling languages with UML.

3 Lesson 1: Beware of software metrics

Software quality can be measured directly via assessing the external attributes, however, this typically requires a substantial time investment. On the other hand, the internal attributes measure the quality indirectly and are considered to be predictors of quality. There are tools (e.g., aopmetrics) that can automatically analyze the internal attributes via a number of established software metrics (e.g., coupling, cohesion). A considerable amount of research has been spent on investigating the possible relationship between internal attributes and quality. However, there are no universally accepted predictability models that can infer the maintainability based only on software metrics. Nevertheless, the internal attributes are often used within maintainability studies.
Our experience has taught us that, in general, it is a good idea to perform an analysis of the internal quality attributes. This step is remarkably simple given the abundance of tools that can automatically collect various software measures. However, expectations should not be too high. Indeed, the results of this step are unlikely to provide sufficient evidence in support of the test hypothesis. Rather, it is a good practice to use the output of the internal attributes analysis as an input for hypothesis formulation.
In our first study the internal attributes analysis has indicated that the aspect preserving process results in smaller, less complex and more modular implementation. We have leveraged on these results in order to construct the experimental hypothesis that was further investigated using a user study. The study confirmed the analysis of the internal attributes. In our second study, however, the numbers for the internal attributes were nearly identical for both treatments. Nonetheless, the user study has eventually indicated that in three out of four cases the aspect preserving process results in remarkably shorter maintenance cycles than

the aspect disrupting process. In this case the internal attribute analysis did not provide any additional insight into the problem statement. Rather, it could have misled us.

4 Lesson 2: Invest in questionnaires

Any research question is best investigated using not only quantitative, but also qualitative methods. Qualitative methods can provide additional insights and explain the sheer numbers. Questionnaires are one of the available qualitative investigation techniques that could be used. The main advantages of questionnaires is that they are relatively easy to design and administer to the subjects of the experiment. Questionnaires also typically do not require a large investment (in terms of time) neither from the experimenter nor from the subjects.
From our experience we have learned to use versatile follow-up questionnaires with several goals in mind. Firstly, questionnaires could be used to validate certain assumptions regarding the experiment setup (e.g., clarity of the task descriptions). Otherwise one has to include these assumptions in the list of threats to the internal validity of the results. Moreover, before the experiment is conducted the experimenter typically has certain expectations concerning the quantitative results. Questionnaires are useful to support these expectations, i.e., to provide additional evidence demonstrating the causal relation between the treatment and the dependent variable. Finally, open questions could provide some unexpected insights into the causality.
In our first study we have designed a very small questionnaire consisting of four closed and one open questions. The results of the questionnaire have validated two assumptions concerning the experiment setup and have provided additional insight into the usability of the composed model. However, in retrospect, we would like to have asked several additional questions that would explain the results. For instance, according to best practices, it is appropriate to ask a supporting open question for every closed question where subjects express opinions that oppose our expectations. For the second empirical investigation we have invested substantially more time in the design of six questionnaires administered after the tutorials, after each maintenance task and at the end of the experiment. The feedback received via the questionnaires has validated a number of assumptions regarding the experiment setup. Through the open questions we have also obtained valuable insights into how to improve future (replicated) studies. Finally, we have gained additional information regarding issues such as the quality of the solution, traceability, etc.

5 Lesson 3: Prepare for pairs

Subject selection for an empirical study is an important activity as it is closely related to the generalizability of the results. Often, researchers are constrained to the convenience sampling scheme where the most convenient persons are selected as participants. In this case it is essential to screen the subjects in order to make sure that they possess at least to a certain extent the skills that a random subject from a representative population would have. In general, it is a good idea to take the subject screening a bit further and try to form *blocks* or even pair the subjects. Within each block (or pair) the subjects' skills are assumed to be very similar, while across the blocks this is not the case. Blocking increases the power of the statistical tests, which could make a difference if the number of subjects is relatively low. However, the main disadvantage of the blocking technique lies in the fact that it could be difficult to demonstrate that the blocks are adequately designed.

A great lesson that we have learned during our two studies is to always plan for pairs. The data obtained from the experiment operation should be analyzed using unpaired statistical tests. If the unpaired tests provide sufficient results, i.e., statistical significance and acceptable power, the experimenter should drop the information regarding the blocks. Only in case the unpaired tests are inconclusive the experimenter should use the blocking information and run paired statistical tests. The correctness of the pair compositions will then typically be an internal threat to the validity of the results and it should be stated during the presentation and reporting of the experiment results.

We have designed both our studies with subjects pairing where presumably each pair of subjects had comparable skills. We have used self-reported questionnaires in order to collect the relevant subjects' skills. As the number of subjects in our first study was relatively low we had to use the pairing information during the statistical analysis. We were aware that the pairing was subjective and we have reported this in the threats to the validity section. On the other hand, in our second study the number of subjects was higher and given the ample effect size the less powerful statistical tests were sufficient to achieve statistical significance without taking into account the pairing information.

Acknowledgments

This work is part of the EUREKA-ITEA EVOLVE project, and is partially funded by the Flemish government institution IWT (Institute for the Promotion of Innovation by Science and Technology in Flanders), by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund K.U.Leuven.

References

1. Hovsepyan, A., Scandariato, R., Van Baelen, S., Berbers, Y., Joosen, W.: From aspect-oriented models to aspect-oriented code? the maintenance perspective. In: Proceedings of the 9th International Conference on Aspect-Oriented Software Development. ACM (2010)
2. Hovsepyan, A., Scandariato, R., Van Baelen, S., Demeyer, S., Joosen, W.: Preserving aspects via automation: a maintainability study. In: To Appear in Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement. IEEE (2011)