An online corpus of UML Design Models : construction and empirical studies

Karasneh, B.H.A.

Chapter 1

Introduction

In this chapter, we discuss the context of the dissertation, the study motivation, and objectives. Also, the research approach and research methods are outlined.

Traditional engineering disciplines such as electrical and mechanical engineering are guided by physical laws. They constrain the engineering of solutions by enforcing regularity and limiting the complexity. Software engineering is not constrained by physical laws. Engineers often create complex software, which is difficult to understand, test and maintain. A large part of research in Software Engineering focuses on the software implementation because it is an inevitable part of software development. However, the implementation is a late activity of the software development process. Implementation is preceded by requirement and design. Software design is the backbone of the implementation. The quality of the design strongly influences the quality of the implementation. Solving problems during the design saves time, effort and cost of maintenance compared to solving the problems that grow out of design flaws only in the implementation. Software models are ways of expressing software designs. The Unified Modeling Language (UML) is used to develop and express software design. The benefit of designing software using a modeling language is that one discovers problems early. Also they provide a high-level overview of systems. Assessing the quality of software design models is an interesting research area. Design models should capture the requirements and also serve as a basis for the implementation. If a software model does not meet the requirements, it may lead to deliver a wrong product. The same applies when a software model

is difficult to understand, complex or difficult to implement, it is likely to lead to producing wrong software. In practice, the quality of software models is important for software architects, developers, testers and maintainers. Software developers, for example, need comprehensible models for producing implementation codes. The designs give developers a view of the software at a high abstraction, which highlights important classes and relationships between classes. Empirical studies are needed for defining, measuring, and testing of modeling quality. Empirical studies have become an important part of software engineering research and practice.

1.1 Problem Statement

Various views exist on what constitutes quality of software. One approach defines quality as the absence of defects. Conversely, one can see the amount and severity of defects as an indicator of the lack of quality in a software system. The assessing of severity of defects is currently mostly done manually. This is quite labor intensive, moreover it suffers from lack of agreement between different developers that perform the rating of defects. Because of the large effort involved in filling out defect reports, developers routinely fill out default values for the severity levels. We can summarize the problem as follows: software defects must be prioritized by assessing their severity. In this thesis, we explore how to automate severity assessment of software defects. Our interest in this thesis is in assessing the quality of software designs – esp. when represented using UML models. Ideally, we would study how UML models affect the quality of the final implementation in industrial projects. However, currently it is difficult to study UML models in an empirical manner because they are difficult to find and collect. Companies are extremely conservative in sharing their software designs because they see designs as part of their competitive advantage. However, we believe that a corpus of UML designs could be very beneficial to the SE research community. We foresee that a repository of examples of software design models can be useful to help to create better designs.

1.2 Objective of the Study

For the software defect severity, our aim is to research how this severity prediction can be achieved through reasoning about the requirements and the design of a system. For this purpose, we want to use ontologies to link reasoning about defects to knowledge about the requirements and design of the system. To enable more empirical studies about UML models, our aim is to build a repository that contains a huge number of UML models. We aim to collect these from the internet, literature and collaboration with companies and universities.

To guide our objectives, we formulate the following research questions:

- **RQ1**: How we can use knowledge about the requirements and design of a system to enable the automatic classification of defects into severity levels?
- **RQ2**: How we can establish a repository of UML models?
- **RQ3**: How can a repository of UML models be used for empirical studies?
- **RQ4**: Does the aid of a repository of software design models improve the quality of software designs created by novice designers?
- **RQ5**: What kind of design flaws can be detected in UML design models?

1.3 Research Methodology

Empirical research is a way of obtaining knowledge using direct and indirect observation or experience. Empirical studies attempt to compare theories with reality and improve the theories as a result. Empirical studies have become an important part of software engineering research and practice. 20 years ago, it was rare to see a conference or journal article about a software development tool or process that had empirical data to back up the claim. Nowadays, it is becoming more common that software engineering conferences and journals stimulate articles that describe a study or evaluation. Researchers in software engineering have been increasingly interested in empirical studies because it allows them to evaluate and improve techniques, methods and tools in developing software. Several methods are used to accomplish our research, including: case studies, experiments, and surveys. For the automated measures of defect severity, we use an industrial case studies that follow the same approach: data collection, data analysis and conversion, and data classification. More details about the methodology and the approach are in chapter 3. Collecting UML models from the internet and via collaboration with other universities is a type of field-study, and we use our collection. We ran experiments using the model repository to answer the research questions. We use a survey method for getting feedback about using our repository of models. Lastly, in our research journey we performed experiments for detecting design flows in UML class diagrams. We focus on design anti-patterns, where an anti-pattern is a literary form that describes a bad solution to recurring design problems that lead to negative effects on code quality.

1.4 Contributions

Our research aims to contribute to the area of software quality. More specifically, our contributions are in three areas: First, the area of automatic prediction of software defect severity. Secondly, the creation of UML Repository. Thirdly, the topic of quality

of UML-based software designs. The contributions in the first area of defect severity are:

1. A method that shows how knowledge about the software requirements and design can be captured and used to predict the severity levels of defects. Including knowledge of the requirements for establishing the severity of defects helps to reflect what is important according to the users of the system, not only according to its developers. This method leads to a better classification of severity of defects than is currently produced by professionals in industrial projects.

2. The use of ontologies and ontology reasoning (AI techniques) to capture and link knowledge about requirements and design. To this end, we propose a set of general rules for reasoning that is synthesized based on industrial projects.

3. The explicit representation of knowledge about classifying defects enhances the understandability of the knowledge that is often implicit and thereby enables the transfer and reuse of domain knowledge.

The contributions in the second area of quality of creation of UML repository are:

1. Creation of a repository with many UML software designs. This repository can be used as:

   (a) A source for experimental material for empirical studies of UML diagrams.

   (b) A basis for corpus studies related to UML modeling (e.g. benchmarking).

   (c) A source of examples of UML that can be used for learning UML by examples.

   (d) A start of an open community on UML modeling.

The contributions in the third area of quality of UML-based software designs are:

1. Provide empirical evidence about the benefits of creating and improving UML models with the aid of examples.

2. Provide empirical evidence of the difference between experts' and novices' assessment of the quality of UML models.

3. Provide a solution for detecting code smells and design anti-patterns in UML class diagrams. This contrasts with the current research which focusses on detecting anti-patterns in the source code implementation.

4. We found that the anti-patterns that occur in the design models, if not tended to, percolate to the source code where they decrease software quality.

1.5 Dissertation Outline

The outline of this work is as follows:

- **Chapter 2: Background.** Defines the foundation on which this dissertation is built and introduces the used terminology. This chapter discusses: (i) software defects and standards of software reports defects. (ii) Ontologies and ontology reasoning. (iii) Introduce UML as modeling language and elaboration of some UML diagram types that are commonly used in practice. (iv) Challenges in modeling using UML especially for novices, and challenges for applying empirical study on UML.

- **Chapter 3: A Method for Automated Prediction of Defect Severity Using Ontologies.** In this chapter we present our approach for creating and using ontologies for predicting the severity of software defects. In this study we use two case studies with 80 defects in total with different severity levels. We show how the data was collected from the company, how we analyze it and convert it to IEEE standard software anomalies report. We use five severity levels only. We show the results and a comparison between our results and using machine learning for predicting severity of the software defects. We also show the feedback from a software architect, software developer, software engineer and the project service coordinator, who emphasized that our approach yields very promising results.

- **Chapter 4: Establishing an Infrastructure for Empirical Research on UML Diagrams.** In this chapter we introduce: i) a crawler for collecting UML models from the internet, ii) a classifier for UML class diagram, iii) a tool (called Img2UML) that converts UML class diagram, sequence diagram and use case diagrams from image formats into XMI format. The Img2UML tool also extracts models information from images and stores this in a database. We show results of the tool, its limitation and some statistics related to the class diagrams collected from the internet.

- **Chapter 5: Models-db.com: An Online Repository UML Models.** In this chapter we present an online repository for UML models, especially with UML class diagrams. The repository contains images, XMI, and design metrics for a large number of class diagrams. The repository is searchable, and the user can search based on names of classes, attributes and operations. Furthermore, the repository allows users to share their models, to define experiments and to generate reports (including statistics and graphs). This repository will be useful for research as the first corpus of UML models. This repository will provide a good place for researchers and students to study and analyze UML models. This improves the possibilities for empirical studies in UML. Also, it supports the application of UML in education and industry.

- **Chapter 6: UML repository as benchmark for Quality Analysis.** In this chapter, we present a statistical analysis of models in the repository, and show some interesting patterns.
- **Chapter 7: Quality assessment of UML Class Diagrams.** We present the experiment we have conducted for comparing how experts and students assess different quality properties of class diagrams. Six quality attributes were addressed. The results reveal that the assessments of students and experts differ in all quality attributes, and that the experts are harsher in their evaluation.
- **Chapter 8: Using Examples for Teaching Software Design.** The goal of this research is to study the effects of using a collection of examples for creating a software design. We performed a controlled experiment for evaluating the use of a broad collection of examples for creating software designs by software engineering students. In this study, we focus on software designs as represented through UML class diagrams. The treatment is the use of the collection of examples. These examples are offered via our searchable repository.
- **Chapter 9: Conclusion and Future Work.** In this chapter we draw conclusions and discuss future work.

1.6 Publications

This is a chronological list of publications that were (co-)authored during this doctoral research:

1. M. Iliev, B. Karasneh, M. R. V. Chaudron, and E. Essenius, "Automated prediction of defect severity based on codifying design knowledge using ontologies," in *1st International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE 2012)*, pp. 7–11, June 2012 (Chapter 3)
2. B. Karasneh and M. R. V. Chaudron, "Extracting uml models from images," in *5th International Conference on Computer Science and Information Technology (CSIT2013)*, pp. 169–178, IEEE, 2013 (Chapter 4)
3. B. Karasneh and M. R. V. Chaudron, "Img2uml: A system for extracting uml models from images," in *39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2013)*, pp. 134–137, IEEE, 2013 (Chapters 4 and 5)
4. B. Karasneh and M. R. V. Chaudron, "Online img2uml repository: An online repository for uml models.," in *Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling (EESSMOD@MoDELS 2013)*, pp. 61–66, 2013 (Chapters 5, 6, 7 and 8)

5. T. Ho Quang, M. R. V. Chaudron, I. Samúelsson, J. Hjaltason, B. Karasneh, and H. Osman, "Automatic classification of uml class diagrams from images," in *Proceedings 21st Asia-Pacific Software Engineering Conference (APSEC 2014)*, 2014 (Chapters 4 and 5)

6. B. Karasneh, D. Stikkolorum, E. Larios, and M. R. V. Chaudron, "Quality assessment of uml class diagrams: A study comparing experts and students," in *MoDELS*, 2015 (Chapter 7)

7. D. Stikkolorum, T. Ho Ho Quang, B. Karasneh, and M. R. V. Chaudron, "Uncovering students' common difficulties and strategies during a class diagram design process: an online experiment," in *MoDELS*, 2015

8. B. Karasneh, R. Jolak, and M. R. V. Chaudron, "Using examples for teaching software design," in *Proceedings of the 22st Asia-Pacific Software Engineering Conference (APSEC2015)*, 2015 (Chapter 8)

9. B. Karasneh, M. R. V. Chaudron, F. Khomh, and Y.-G. Guéhéneuc, "Studying the relation between anti-patterns in models and in source code," in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016)*, 2016 (Chapter 6)