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

Karasneh, B.H.A.

Summary

In this thesis, we address two problems in Software Engineering. The first problem is how to assess the severity of software defects? The second problem we address is that of studying software designs. We explain each of these problems next in a little more detail.

Assigning severity to a software defect is currently done by software developers based on their experience. In practice it turns out developers do not take the task of registering the severity of defects very seriously and often register just a default value offered by a defect tracking tool. Therefore, we study whether we can provide meaningful automated support for assessing the severity of defects. Automated support for assessing the severity of software defects helps human developers to perform this task more efficiently and more accurately. In this thesis, we present a new approach (MAPDESO) for assessing the severity of software defects based on the IEEE Standard Classification for Software Anomalies. The novelty of the approach lies in its use of uses ontologies and ontology-reasoning which links defects to system level quality properties. The approach was validated by studying how it performs on an industrial project. In this validation, the automated prediction method performs well compared to the manual classification performed on an industrial project. We found that our MAPDESO approach based on ontologies outperforms selected machine learning classifiers. At the company where we conducted the study, people appreciate the result of MAPDESO.

Our research can be positioned in the field of Empirical Software Engineering – this is a branch of Software Engineering that aims to create knowledge through the analysis of artefacts (and processes) that are part of software development projects. The large majority of studies in this field focus on program texts ('source code') expressed in some programming language. While software designs area a critical aspect of most software development projects, software designs are hardly studied by the empirical software engineering community.

One of the main reasons why studying software designs is challenging is the lack of their availability. In our research we found that software designs are commonly stored as UML diagrams in image formats and are available from various sources on the internet. Hence we decided to collect a large number of UML models in images

and convert them to real models. Therefore, we developed the following software tools: 1) UMLCrawler, which can download a large number of UML diagrams from the internet, 2) UMLImgClassifier, which can classify UML class diagrams from other diagrams, and 3) Img2UML, which can extract model information from UML models stored in image formats and generate XMI files. The generated XMIs are compatible with different UML tools, where the models can be edited and analysed. Our validation shows that UMLImgClassifier and Img2UML provide high accuracy for classifying and converting UML diagrams to XMI files.

The aforementioned tools are building blocks for constructing a UML Repository. We built an online repository which contains UML class diagrams, XMIs and various design metrics of the class diagrams. In this thesis we present this repository and some of its services of such as sharing, searching, questioning, ranking, defining experiments, uploading, downloading and charting.

This repository is the first of its kind and we believe it will be a useful resource for the empricial software engineering research community. In support of this, we conducted a series of empirical studies using the UML Repository. These empirical studies are a drop in the ocean of empirical studies that can be conducted using this repository.

Our first study shows some examples of interesting relations between class diagrams design metrics. In this way, the corpus of UML designs can be used to find patterns in UML models and hence to figure out good and bad practices. This yields reference data that can be used in quality assurance of UML designs.

Our next study studies the relation between the quality of the UML design and the associated source code. Our most prominent findings in this study are: First, modeled classes (classes that are both in the design and the source code) undergo more changes and contain more faults than classes that exist only in the source code (hence not in the design). Second, anti-patterns can be detected early in the design, and anti-patterns that exist in the design transfer to the source code in the same classes. Third, modeled classes that have anti-patterns in the design have more changes and faults than other classes that do not have anti-patterns in the design. Our study is the first study of the relation between the quality of UML-based software designs and their impact on the quality of the source code.

Subsequently, we studied how the repository can be used in educational settings. Our first study in this direction investigates the difference between students and experts in assessing the quality of UML class diagrams. We assessed quality along six quality attributes and distinguished between students' self-assessment and peer-assessment. We found that there is a significant difference between experts and students in both self-assessment and peer-assessment. We found a high correlation between experts and peer-assessment for assessing understandability. In addition, we found that there is the single quality attribute understandability which is correlated with most other quality attributes in both experts' assessments and students' peer-assessments. Moreover we found that students seem to avoid giving peers low grades. Hence peer

assessment by students should not be used for grading in class settings. However, using qualitative analysis of the feedback given in conjunction with the assessment, we observed that novices provide comments on the quality of UML designs that is similar to that which experts' give. Therefore, we conclude that the feedback provided by students' is valuable, and while peer-grading may not be a good idea, using students for peer-feedback provides useful information for improving their designs.

After that, we conducted a study into using examples for teaching software design. To this end, we offered students the possibility to use the UML repository while performing some design assignment and for improving an initial design they have made without the repository. We conclude that the examples help students in constructing and improving their design. Quantitative analysis showed that experts graded the models produced by students that used examples higher than students in the control group (without repository) for all quality attributes. We conclude that the model repository is a suitable environment for offering model examples – in the sense that its content is useful, but also that model examples can be found in an effective way. Furthermore, the students appreciate the fact that the UML Repository allows them to search for models based on various parts of the model such as class names, attributes, and operations. This type of search is not available anywhere else, also not via generic search engines on the internet. To conclude, this series of empirical study illustrates the importance of the corpus of UML models.