Proposing and empirically validating change impact analysis metrics

Arvanitou, Elvira Maria

Chapter 8 – Conclusions and Future Work

This chapter elaborates on the conclusions of the PhD thesis. In particular in Chapter 8.1, the answers to research questions are recapped, and the main contributions are highlighted. In Chapter 8.2, we present future work.

8.1 Answers to Research Questions and Contributions

The problem statement that this thesis aims to address is stated in Chapter 1.3.1 as follows: "*Current change impact analysis practices that are based on instability and change proneness, are not supported: (a) by metrics for requirements and architecture development phases, (b) by metrics that consider both change proneness and instability, and (c) by automated tools that quantify change proneness and instability*". To address this problem we have asked and answered 6 research questions, as presented in Chapters 2 to 7. Table 8.1.a repeats the research questions, maps them to the Chapters in which they are addressed, and highlights the achieved contributions and the primary outcome per chapter beyond the state of the art.

Table 8.1.a: Thesis Contributions

Research Question	Chapter	Contributions
RQ_1: Which are the most important design-time quality attributes, and how can they be measured?	Chapter 2	An overview of the most studied (in literature) design-time quality attributes and associated metrics. *Primary Outcome*: Instability and change proneness are among the mostly studied design-time quality attributes but they lack quantification in the requirements and architecture development phases.
RQ_2: Are maintainability prediction source-code metrics applicable at the architecture phase?	Chapter 3	An overview of maintainability predictors at the source-code level, which are able to capture coarse-grained change, and are therefore applicable at the architecture phase. *Primary Outcome*: Novel change proneness and instability metrics are required for the architecture phase.

Research Question	Chapter	Contributions
$RQ_{3.a.i}$: How to assess Instability at the source-code level?	Chapter 4	A method and a tool that quantifies instability at source-code level. *Primary Outcome*: The Ripple Effect Measure (REM) has been proposed and validated both in an empirical and a theoretical manner. The comparison to existing metrics suggested that REM is the optimal assessor of instability.
$RQ_{3.a.ii}$: How to assess Change Proneness at the source-code level?	Chapter 5	A method and a tool that quantifies change proneness at source-code level. *Primary Outcome*: The Change Proneness Measure (CPM) has been proposed and empirically validated. The comparison to existing metrics suggested that CPM is the optimal assessor of change proneness.
$RQ_{3.b}$: How to assess Change Proneness at the architecture level?	Chapter 6	A method and a tool that quantifies change proneness at the package level, which is the first level of architectural decomposition in object-oriented systems. *Primary Outcome*: The Module Change Proneness Measure (MCPM) has been proposed and empirically validated. The comparison to existing package metrics suggested that MCPM is the optimal assessor of change proneness. To calculate MCPM, we have tailored the REM metric so as to fit the package level.
$RQ_{3.c}$: How to assess Change Proneness at the requirements level?	Chapter 7	A method and a tool that quantifies instability and change proneness at the requirements level. *Primary Outcome*: The Requirements Ripple Effect Metric (R2EM) has been proposed and empirically validated in an industrial setting. Similarly as before, R2EM is used as a parameter for quantifying requirements change proneness.

Next, the response to each research question is briefly discussed:

RQ_1: Which are the most important design-time quality attributes, and how can they be measured?

The main motivation for setting this research question was to familiarize with the state-of-the-art on design-time quality attributes and metrics, and

at the same time assess the importance of the selected quality attributes (namely instability and change proneness). Furthermore, we aimed at identifying existing metrics for quantifying instability and change proneness. To answer this research question, we have performed a mapping study, in which we reviewed more than 150 primary studies. The outcome of this process affirmed our intuition and background knowledge: (a) on the importance of change proneness and instability, in the sense that they are regularly studied in the literature, and (b) on the lack of instability and change proneness metrics on certain development phases, i.e., requirements and architecture. In particular, change proneness and instability have been ranked as the most studied quality attributes after maintainability (18 and 15 studies respectively). Regarding metrics, the study has revealed the existence of 8 metrics at the source-code level, 6 at the detailed-design level, but none at the architecture and requirements level.

RQ_2: Are maintainability prediction source-code metrics applicable at the architecture phase?

Given the lack of instability and change proneness metrics at the level of architecture and the plethora of available metrics at the source-code level, in this research question we aimed at assessing the applicability of source-code maintainability predictors to be applied at the architecture level. Driven by this need, we proposed a novel metric property, namely Software Metrics Fluctuation (SMF), which is able to assess the ability of a metric to capture small-scale changes (pertinent only for the source-code level) or large-scale changes (pertinent only for the architecture level). More specifically, by studying 21 metrics we concluded that only one coupling metric (most of the metrics reported as related to instability are coupling metrics), namely Response for a Class—RFC, is able to capture large-scale changes in the underlying design: thus, it is suitable for the architecture level. However, since RFC is not a purely coupling metric (since it captures a size dimension of a class, as well), we believe that a purely coupling metric might be more suitable for assessing change proneness and instability.

$RQ_{3.a.i}$: How to assess Instability at the source-code level?

After setting the scene for this PhD thesis in Chapters 2 and 3, we proceeded to achieving the main contributions of this research effort, i.e., to propose and validate change proneness and instability metrics at the implementation, architecture, and requirements level. As a starting point, we proposed

a novel metric, namely Ripple Effect Measure (REM), which is able to assess class instability at the source-code level. The proposed metric has been validated both in an empirical and a theoretical manner. The validation suggested that REM is the optimal assessor of class instability. In particular, REM has exhibited 38% stronger correlation to class instability, compared to the most optimal metric (i.e., coupling between object—CBO).

$RQ_{3.a.ii}$: How to assess Change Proneness at the source-code level?

In order to proceed to class change proneness assessment, we proposed to combine the previously validated class instability metric with historical density of class changes. To achieve this goal, we proposed and empirically validated the Change Proneness Measure (CPM). The outcome of the validation suggested that CPM is the optimal assessor of change proneness, compared to existing metrics. More specifically, CPM has proven to be 48% more correlated to class change proneness, compared to the most optimal coupling metric (i.e., message passing coupling—MPC). Additionally, we confirmed our intuition that the combination of using structural (i.e., instability metric) and historical (i.e., change density metric) data could lead to improved results, compared to when only one of the aforementioned parameters are used. Therefore, at this stage, we have defined how change proneness of an artifact can be calculated based on its instability and evolution data.

$RQ_{3.b}$: How to assess Change Proneness at the architecture level?

Driven by the outcome of RQ_2 we have set a separate research question on change proneness and instability of architectural modules. By taking into account that the scope of this PhD thesis is Object-Oriented technologies, we considered as module the package level, which is the first level of architectural decomposition of system, after the class level. To answer this research question, we first tailored REM to fit the architecture level, and then fed it to the calculation of a new metric, called Module Change Proneness Measure (MCPM), following the same method defined in $RQ_{3.a}$. To evaluate the proposed metric, we compared its assessing power with existing package metrics, and successfully validated it as the optimal change proneness assessor. Based on the performed evaluation, MCPM is on average 23% more accurate predictor of module change proneness compared to existing coupling metrics at the package level.

$RQ_{3.c}$: How to assess Change Proneness at the requirements level?

Finally, in this research question we focus on the third targeted development phases, i.e., requirements. Based on the previously acquired knowledge on how to calculate change proneness from instability, as an answer to this research question we focused on how to quantify the ripple effects between requirements. To this end, we proposed the Requirements Ripple Effect Metric (R2EM), which we successfully validated in an industrial setting. In particular, we affirmed that R2EM is able to accurately rank the strength of the relationships between different requirements. In particular, R2EM has proven to be strongly correlated (i.e., 0.6) to the experts' opinion.

8.2 Ongoing and Future Work

As future work opportunities that have emerged from this PhD thesis, we have identified four main directions: (a) improving the accuracy of the proposed methods, (b) strengthening the industrial applicability of the methods, (c) extending the scope of the proposed metrics, and (d) reusing the proposed metrics in the definition of metrics for other quality attributes.

8.2.1 Improvement of Metrics Accuracy

Although all proposed metrics are validated as the optimal predictors compared to existing ones, there is still a need to improve their effect size (a statistical measure that considers both the size of the sample and the correlation coefficient). For each metric, specific improvement strategies have been already planned:

- ***Instability Metrics***: We encourage researchers to investigate the possibility of assigning weights to the various axes through which a class can receive changes (i.e., protected attributes, called methods, and overridden methods) by an empirical study on class change history, and observe if such a change can increase the validity of REM. Additionally, we encourage researchers to use combinations of coupling metrics to explore the possibility of increasing the power of metric for predicting the ripple effect.
- ***Change Proneness Metrics:*** We encourage researchers to investigate if the use of a larger portion of software history, will increase the ability of the proposed metrics (i.e., CPM, MCPM, and R2EM) to more accurately assess software artifacts' change proneness. Furthermore, an

additional extension in this respect would be to differently weight more recent change history compared to older ones, in the sense that they are more probable to be representative of future changes.

8.2.2 Industrial Applicability

By acknowledging that from all the proposed metrics, only R2EM has been validated in industry, we plan to investigate the usefulness of metrics targeting other development phases with practitioners. In particular, we would like to investigate how the instability and change proneness metrics can inform the reasoning and decision making processes of software engineering, as well as their willingness to apply them. To boost the applicability of the methods, we plan to also update the tool-chain developed in this PhD thesis and transform it either to a web-service or to a one-click quality assessment process.

8.2.3 Extending the Scope of the Proposed Metrics

All metrics have by now been evaluated with Java code, therefore the results cannot be generalized to other programming languages, and of course development paradigms other than object-oriented programming. To this end, we plan to: (a) transform the tool-chain to enable the parsing of other programming languages, like C++, C#, php, etc., (b) perform evaluations on additional and larger systems, and (c) tailor the proposed metrics to fit other programming paradigms. Additionally, since SMF has until now only be applied in existing metrics, we plan to evaluate the expected applicability of the proposed metrics, in the development phase that we have introduced them.

8.2.4 Propose Metrics for Other Quality Attributes

One of the most important parameters taken into account when building the proposed methods was our intention to have uniform metric calculation models for various development phases. In the current literature this is not the common case, in the sense that new metrics do not use existing ones as a starting point, but are usually developed from scratch. In this thesis, we have tried to reuse already defined metrics, in future metric definitions. For example, between metrics of different phases: REM has been used in MCPM; in metrics of the same development phase: REM has been used in the calculation of CPM. Since the proposed metric definition reuse proved to be successful, we currently work on the quantification of software modularity and intend to propose met-

rics on the development phases that we have explored in this thesis: i.e., implementation, architecture, and requirements.