# A DEFECT PREDICTION MODEL FOR MODEL DRIVEN ENGINEERING

Kariyawasam Siththarage Dinesh Nadun Kumara de Silva

168214E

MSc in Computer Science specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2020

# DECLARATION

I declare that this is my own work and this Post Graduate Degree Project Report does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic, or another medium. I retain the right to use this content in whole or part in future works.

05/29/2020

Kariyawasam Siththarage Dinesh Nadun Kumara de Silva Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this project report is acceptable for evaluation for the MSc Research.

.....
.....

Dr. Dulani Meedeniya Date

## Acknowledgment

My sincere appreciation I need to gift to my family for the continuous support and motivation given to make this thesis a success. I also express my heartfelt gratitude to Dr. Indika Perera and Dr. Dulani Meedeniya, for the supervision and advice given throughout to make this research a success. All my MSc batch mates who contribute with myself throughout the study of this entire duration in my stay in the university for this Post Graduate Study has to be appreciated as well. My parents and my relatives have to be thankful for understanding my busy schedule and the fact that I did not have enough time to spend with them and become a close family member or a relative. Last but not least my newborn child Vihini and the loving wife Lakmini for making me smile throughout this tough period.

# Abstract

Model-Driven Engineering (MDE) is used in the Software Industry which enables level to level transformation until the final system is created. This concept helps to ensure the bridging of gaps between the problem domain and the solution scope of a software system. A software system with a lesser number of software defects or zero defects will be successful software. Earlier the defects are identified it reduces the cost in terms of effort, time, and human resources, rather fixes that defect in a later stage of the software development life cycle. The development of defect prediction models and the efficient usage will prevent unnecessary defect fixing efforts later.

Unified Modelling Language (UML) provides certain notations to create models in different aspects. UML Class diagram is very widely used in identifying and evolving business entities. UML Class diagrams as entities can be mapped with Database Management Systems and propagate the business entities.

Every business must deal with the inevitable truth of change. To survive in a competitive market, business functions, and business directions are under the freedom of change at any moment. Stable business solutions are the compulsory components of successful businesses.

Applying changes to Software makes them fragile when they are not done properly. The phase where adding changes or in the maintenance mode, and the most important stage of the business, must be well away from defects.

This thesis covers a defect predictive approach that can be applied at the UML class diagram models that are created at the beginning of the Software solution, however, the defect prevention applies to the maintenance or in the most vital stage of the business.

This thesis discusses the possible defect prediction models that can be used in MDE to facilitate fast and efficient software development. At the end of the thesis, it will discuss the approach that has taken to introduce a defect prediction strategy to the Model-Driven Engineering its evaluation and the contributions to the research community

**Table of Contents**


| | | |
|---|---|---|
| Declaration | | i |
| Acknowledgement | | iii |
| Abstract | | iv |
| Table of content | | v |
| List of Figures | | viii |
| List of tables | | x |
| List of abbreviations | | xi |
| List of Appendices | | xii |
| Chapter 01 Introduction | | 1 |
| 1.1 | Background | 2 |
| 1.1.1 | What is a Model? | 2 |
| 1.1.2 | Model Driven Engineering | 2 |
| 1.1.3 | MDE Approaches and challenges | 4 |
| 1.1.4 | Defect Prediction | 6 |
| 1.2 | Research Problem | 6 |
| 1.2.1 | Research Problem Statement | 7 |
| 1.3 | Proposed Solution | 7 |
| 1.4 | Research Objective | 7 |
| 1.5 | Research Overview | 8 |
| Chapter 02 Literature Review | | 9 |
| 2.1 | Model Driven Engineering | 10 |
| 2.2 | Defect Prediction | 17 |
| 2.2.1 | Data Mining and Machine Learning | 18 |
| 2.2.2 | Security Defect Prediction | 20 |
| 2.2.3 | Other Defect Prediction approaches | 20 |
| 2.2.4 | Comparing Research Solution with the Literature Review | 33 |

Chapter 03 Proposed Methodology 35

3.1 Solution Architecture 36

3.1.1 Design Principles 37

3.1.2 Importance of Design Principles when absorbing changes into the Design 39

3.2 How the Design Principles can be used as the Defect Preventive methodologies. 48

3.2.1 Importance of Object Constraint Language 48

3.2.2 Possible Solution approaches 48

3.3 Selected Solution to be implemented as the solution for Research Problem 50

Chapter 04 Solution Architecture and Implementation 52

4.1 Solution Introduction

4.2 Characteristics of the external module

4.3 Concepts behind the external module 53

4.6 Detailed Design of the solution 53

4.5 Current implementation of micro service 54

4.6 Further possible extensions of the solution 57

4.7 Further possible extensions of the solution 57

Chapter 05 System Evaluation 58

5.1 Introduction 59

5.2 Evaluating the system concepts 59

5.3 Evaluating case scenarios 61

5.3.1 Evaluate an inheritance-based solution design 61

5.3.2 Evaluate Association based solution design 62

5.3.3 Evaluate Composition based solution design 64

5.3.4 Evaluate Aggregation based solution design 65

5.4 Evaluating system behavior 66

5.4.1 Five concurrent users sending 20 requests each user 67

5.4.2 Ten concurrent users sending 20 requests each user 68

5.4.3 Twenty concurrent users sending 20 requests each user 69

5.5 Evaluation from industry experts 70

Chapter 06 Conclusion 73

Conclusion 74

Reference List 76

Appendix A: Sample Json Payload 80

Appendix B: Json message for a composition relation 82

Appendix C: Json message for an Aggregation relation 83

Appendix D: Main Json Schema 84

Appendix E: Source Code of the solution – server.js 88

Appendix F: Source Code of the solution – Evaluator.js 89

**Table of Figures**


Figure 1.1: Modelling in between reality and the expectation 3

Figure 2.1: Example Model 11

Figure 2.2: Overview UML Profile for this case study 13

Figure 2.3: Design of the model in the case study on UML Profile - Angular JS 15

Figure 2.4: Directive section – case study design model 15

Figure 2.5: With and without Exception handling constructs against class size 22

Figure 2.6: Comparison between overall defects and exception class defects 23

Figure 2.7: A sample program source code with control flow 28

Figure 2.8: Four-step approach for Universal Defect Prediction Model 30

Figure 3.1: Content types created by Content Creators 40

Figure 3.2 Content types of future variations created by Content Creators 41

Figure 3.3 Main features done by the content creators 42

Figure 3.4 A class diagram with needed abstraction layers in place 43

Figure 3.5 A Class diagram with inheritance demonstrated 43

Figure 3.6 UML Composition 44

Figure 3.7 UML Aggregation 44

Figure 3.8 A demonstration of tightly coupled two classes 45

Figure 3.9 A demonstration of loosely coupled two classes 45

Figure 3.10: High-level solution plan 49

Figure 3.11 How modeling UI is linked with the Design Principles evaluating module 50

Figure 4.1 High-level Design of the solution 52

Figure 4.2: The swagger interface of the solution service 54

Figure 4.3: Sample design diagram that is submitted to the solution service 55

Figure 4.4: Response from the solution service for the submitted design 55

Figure 4.5: Flow of the Solution Serice 55

Figure 4.6: AWS Cloud deployment of the Solution Service 56

Figure 5.1: An example of how the abstract layer reduces the coupling 60

Figure 5.2: Sample design diagram that is submitted to the solution service 61

Figure 5.3: Service output for the design 61

Figure 5.4: Suggested solution after recommendations 62
Figure5.5: An Association based solution design 63
Figure5.6: Service output for the design 63
Figure5.7: Suggested solution after recommendations 64
Figure5.8: Composition based design 64
Figure5.9: Service output for the design 64
Figure5.10: Suggested solution after recommendations 65
Figure5.11: Example of an Aggregation based UML 65
Figure5.12: Service output of the design 65
Figure5.13: Extension of the Aggregation based UML after evaluation 66
Figure5.14: Response time against the allocated memory 67
Figure5.15: Response time percentage consuming time 68
Figure5.16: Response time against the allocated memory 68
Figure5.17: Response time percentage consuming time 69
Figure5.18: Response time against the allocated memory 69
Figure5.19: Response time percentage consuming time 70
Figure5.20: Average response time against memory 70
Figure5.21: Sample Class Diagram 71
Figure5.22: Result for the class diagram evaluation 72
Figure5.23: Extended class diagram after the evaluation results 73

## List of Tables

Table 2.1: Shorthands for UML Relationships 10

Table 2.2: Steriotypes for this case study 15

Table 2.3: List of Software Matrices used 34

**List of abbreviations**

| | |
|---|---|
| EDOC | Enterprise Distributed Object Computing |
| EJB | Enterprise Java Beans |
| ILLE-SVM | Improved Local Linear Embedding and Support Vector Machines |
| IOT | Internet of Things |
| LLE-SVM | Local Linear Embedding and Support Vector Machines |
| MDA | Model Driven Architecture |
| MDP | Metrics Data Program |
| MOF | Meta Object Facility |
| NN | Neural Networks |
| OMG | Object Management Group |
| PIM | Platform Independent Models |
| PSM | Platform Specific Models |
| RF | Random Forest |
| SDLC | Software Development Life Cycle |
| SNB | Supervised Naïve Bayes |
| SVR | Support Vector Regression |
| UML | Unified Modelling Language |

**List of Appendices**

Appendix A: Sample Json Payload 81

Appendix B: Json message for a composition relation 83

Appendix C: Json message for an Aggregation relation 84

Appendix D: Main Json Schema 85

Appendix E: Source Code of the solution – server.js 81

Appendix F: Source Code of the solution – Evaluator.js 81