Defect Prediction & Prevention In Automotive Software Development

Rakesh Rana

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Göteborg University
Göteborg, Sweden, 2013
To my Mom, Dad & Brother
Abstract

**Background:** Over the last few decades, the importance of software in the automotive domain has been increasing exponentially. Within this domain, some of the functions developed such as active safety features are classed as safety critical which demands very high reliability. Mathematical models referred to as Software Reliability Growth Models (SRGMs) have been used for long time to evaluate and forecast the reliability growth of software systems. While a number of SRGMs have been proposed and used, their evaluation and application within automotive domain still needs to be explored.

**Objective:** This thesis aims to facilitate the adoption of reliability growth modelling within the automotive domain and increase the reliability of software developed within this domain by exploring two directions. First, it aims to establish the applicability of SRGMs in automotive industrial domain for the purpose of evaluating reliability of software and facilitating testing resource allocation. Second, we propose and evaluate new approaches to increase the reliability of software by testing functional models at early stages of development.

**Method:** Empirical evaluations of SRGMs on defect inflow data from automotive domain has been conducted in two research studies. A comparative analysis of two widely used parameter estimation procedures when using SRGMs was also evaluated. Open and semi-structured interviews were used to collect data to familiarize the researchers with the software development process within the automotive domain, the best practices and current challenges. Further a case study was also done to propose and evaluate a new approach for handling natural parameters in dynamic system environment models under fault mode.

**Results:** It was shown that SRGMs can be used in the automotive domain, but for better fit and predictive power certain parameters such as testing effort need to be taken into account. Some reliability growth models were also shown to have good fit and long-term predictive powers which can be used for making asymptote predictions during an on-going project. A framework to combine fault injection and mutation based testing to test models early in the development process is proposed and evaluated. Finally a new approach called fault bypass modelling is also proposed to deal with
a practical problem encountered when running simulations of functional models under fault mode.

**Conclusion:** The findings in this thesis help towards wider adoption of reliability growth modelling as a useful tool for assessment of reliability of software under development within automotive domain. The results also show the SRGMs as a potential tool for effective allocation of testing resources during development. Using the framework and approaches proposed and evaluated in this thesis, we provide a proof-of-concept that new approaches such as combining fault injection and mutation testing can be applied at the functional model level to shift some of the verification and validation efforts from late to early stages in the development process. Early verification and validation of software under development offers high potential with respect of minimizing late defects and developing robust software right from the start.
Acknowledgment

First of all, I would like to thank my supervisor Assoc. Prof. Miroslaw Staron for his invaluable support, guidance, encouragement and feedback which has helped me pursue my PhD study. I am very grateful to Prof. Jörgen Hansson and Martin Nilsson from Volvo Cars who have supported me and this project at all times. Miroslaw, Jörgen and Martin besides your support, without your efforts the project wouldn’t even have existed and I am thankful to you all for giving me the opportunity to work on this project.

Next, I would like to thank my co-supervisors Assit. Prof. Christian Berger and Dr. Fredrik Törner for always giving their advice, guidance and supporting the project whenever needed. Special thanks to my colleagues in the software engineering division for creating a friendly and motivating work environment. Also, I would like to thank Prof. Andrei Sabelfeld for his support and guidance throughout the time he has been my examiner.

I am very grateful to all colleagues at Volvo Cars for helping with the research studies. Also, I would like to thank my colleagues from other companies involved in on-going research studies, especially Wilhelm Meding at Ericsson and Christoffer Höglund at SAAB. I appreciate the time and effort you have put on this project and data provided.

My inexpressible appreciation goes to my family and friends that have always supported me. Special thanks to my parents, brother and sister for their unconditional love and support, you guys have always been and will be the main source of motivation, love and support in my personal and professional life.

The research presented in this thesis has been carried out in a joint research project financed by the Swedish Governmental Agency of Innovation Systems (VINNOVA) and Volvo Car Corporation. It has been supported under the FFI programme (VISEE, Project No: DIARIENR: 2011-04438).
List of Publications

Appended papers

This thesis is based on the following papers:


*Published at 2nd Workshop on Software-Based Methods for Robust Embedded Systems (SOBRES’13), Koblenz, Germany, Sep 2013*

**Other papers**

The following papers are published or in revision but not appended to this thesis, either due to contents overlapping that of appended papers, or contents not related to the thesis.

*Poster, ICRES double workshop “Benchmarking Functional Safety” + “Efficient Systems Development with Functional Safety”.*

[b] E. Nivorozhkin, M. Holmen, and R. Rana “Do antitakeover devices affect the takeover likelihood or the takeover premium?,”
*Published in The European Journal of Finance, 2013.*

[c] M. Holmen, and R. Rana “Causes and consequences of employee representation on corporate boards,”
## Contents

Abstract vii
Acknowledgment viii
List of Publications ix

1 Introduction 1
  1.1 Background and Related Work ................. 3
    1.1.1 Software Development in Automotive Domain 3
    1.1.2 Software Quality .......................... 5
      1.1.2.1 Dependability .......................... 6
    1.1.3 ISO 26262 ................................. 9
  1.2 Software Reliability Growth Models .......... 12
  1.3 Research Focus and Questions ............... 12
  1.4 Research Methodology ........................ 15
    1.4.1 Analytical Research ...................... 16
    1.4.2 Quantitative Research .................... 17
    1.4.3 Case Study ............................... 17
    1.4.4 Interviews ............................... 17
    1.4.5 Research Validity ....................... 18
  1.5 Summary of contributions of each chapter ... 20
    1.5.1 Chapter 2: Evaluation of standard reliability
growth models in the context of automotive software systems 20
    1.5.2 Chapter 3: Comparing between Maximum Likelihood Estimator and Non-Linear Regression estimation procedures for Software Reliability Growth Modelling 20
    1.5.3 Chapter 4: Evaluating long-term predictive power of standard reliability growth models on automotive systems 21
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.4</td>
<td>Chapter 5: Increasing Efficiency of ISO-26262 Verification and Valida</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>tion by Combining Fault Injection and Mutation Testing with Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Based Development</td>
<td></td>
</tr>
<tr>
<td>1.5.5</td>
<td>Chapter 6: Improving Fault Injection in Automotive Model Based</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Development using Fault Bypass Modelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>25</td>
</tr>
</tbody>
</table>

**2 Paper A**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>38</td>
</tr>
<tr>
<td>2.2</td>
<td>Related Work</td>
<td>38</td>
</tr>
<tr>
<td>2.3</td>
<td>Research context and method</td>
<td>40</td>
</tr>
<tr>
<td>2.4</td>
<td>Results and interpretation</td>
<td>41</td>
</tr>
<tr>
<td>2.5</td>
<td>Conclusions</td>
<td>43</td>
</tr>
</tbody>
</table>

**3 Paper B**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>46</td>
</tr>
<tr>
<td>3.2</td>
<td>Background</td>
<td>47</td>
</tr>
<tr>
<td>3.2.1</td>
<td>SRGMs: Software Reliability Growth Models</td>
<td>47</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Model Selection</td>
<td>47</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Comparing between SRGMs</td>
<td>48</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Parameter Estimation</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Research Context and Method</td>
<td>50</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Research Objectives</td>
<td>50</td>
</tr>
<tr>
<td>3.3.2</td>
<td>SRGMs and Data</td>
<td>50</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Data Analysis Techniques</td>
<td>51</td>
</tr>
<tr>
<td>3.4</td>
<td>Results</td>
<td>54</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Parameter estimation using MLE and NLR estimation.</td>
<td>54</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Predictive Accuracy using Predicted Relative Error (PRE) and unbiased</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>PRE (BPRE).</td>
<td></td>
</tr>
<tr>
<td>3.4.3</td>
<td>Which Estimators give better Fit to data and Predicted values.</td>
<td>58</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Working with un-grouped data.</td>
<td>60</td>
</tr>
<tr>
<td>3.5</td>
<td>Conclusions</td>
<td>60</td>
</tr>
</tbody>
</table>

**4 Paper C**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Background</td>
<td>66</td>
</tr>
</tbody>
</table>
CONTENTS

4.2.1 Automotive Software ........................................ 66
4.2.2 Software Reliability Growth Model ...................... 67
4.3 Related Work .................................................. 68
4.4 Models and Data ............................................... 70
4.5 Results: Fit and Long Term Prediction ..................... 73
  4.5.1 Which SRGMs fit best? ................................... 73
  4.5.2 Which SRGMs have the best long term predictive power? ........................................... 78
4.6 Results: Using Past Experience for Reliability Growth Modelling .......................................... 81
  4.6.1 Threats to Validity ......................................... 83
4.7 Conclusion ..................................................... 85

5 Paper D .......................................................... 87
  5.1 Introduction .................................................. 88
  5.2 Background .................................................. 89
    5.2.1 Automotive Software Development & ISO 26262 ........................................................ 90
    5.2.2 ISO 26262 ................................................ 92
    5.2.3 Fault Injection .......................................... 93
    5.2.4 Mutation Testing ....................................... 94
  5.3 Related Work ................................................ 95
  5.4 Road Map for Early Defect Detection ..................... 96
  5.5 Conclusions ................................................ 98

6 Paper E .......................................................... 101
  6.1 Introduction .................................................. 102
  6.2 Fault Injection .............................................. 104
  6.3 Related Work ................................................ 106
  6.4 Problem Analysis .......................................... 108
    6.4.1 CASE: ABS Anti-Lock Braking System .......... 108
    6.4.2 Naïve fault injector ................................ 110
  6.5 FBM: Fault Bypass Modeling .............................. 113
  6.6 Conclusions ................................................ 115

7 Conclusion ...................................................... 121
  7.1 Future Directions ........................................... 121
  7.2 Conclusions ................................................ 122
Chapter 1

Introduction

Software affects almost every part of our life today, from consumer electronics we almost cannot do without to most services we consume on our computing devices and over internet are provided with help of software. Even the products such as cars which contained miniscule amount of software (if any) in 1970s today carry a large amount of software. The role software plays in modern automobiles has been growing exponentially [1]. Some estimates suggest that today’s premium segment cars typically carry about 100 million lines of code distributed over 80 ECUs connected via several buses [2]. With increasing importance of software in automotive domain, a critical issue within the sector is of course system quality [3], attributes related to quality such as reliability, maintainability, reusability etc. are of very high importance. Software quality, mainly the attributes related to dependability are even more important when developing software for systems deemed as safety critical.

In ISO/IEC 9001, quality is referred to as ability of software product to conform to its requirements [4], ISO 9126 [5] defines quality as “the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs.” While ISO 25000 [6] takes the following approach to quality: “capability of software product to satisfy stated and implied needs when used under specified conditions”

The focus of this thesis is on the extended reliability characteristic of software which includes the subcharacteristics of maturity, fault tolerance, recoverability and reliability compliance. Reliability is one of the most important characteristics for software which is part of critical systems. There are three main types of critical
systems [7]:

- Safety critical,
- Mission critical, and
- Business critical.

A system where failure may lead to injury, loss of life or a serious damage to the environment is classed as safety critical. Examples of safety critical systems include medical devices, number of systems within aerospace applications, chemical industry, nuclear power stations and military equipment’s.

A number of systems in the automotive domain, due to their potential impact on health and safety of humans involved are classified as safety critical. Among others these are systems related to braking, steering and active safety systems; a large effort is invested in ensuring the reliability of safety critical software. Software testing is still the main source of ensuring reliability, but given the theoretically infinite scope of software testing, ensuring 100% reliability is either highly difficult, too costly or even impossible for some large and complex systems. Mathematical models based on defect discovery and testing process provide empirical basis for monitoring and assessing reliability of given software artefact.

Software reliability growth models (SRGMs) are the result of applying reliability engineering approaches in the software engineering domain. The early proposals of these models, application and evaluation have come from Japanese corporations and academia applying these models to study reliability growth of software during their testing phases [8]. The early models have been extended to include the test effort [9] and new approaches such as Bayesian models [10] have been applied in this area. Contemporary software reliability models are based on one of following approaches [11]:

- Exponential models
- Non-homogeneous Poisson processes
- Markov processes
- Bayesian reliability models
1.1 Background and Related Work

1.1.1 Software Development in Automotive Domain

As explained in the introduction, automobiles in the last three to four decades have seen an unprecedented rise in use and importance of software. From virtually no software in early 1970s/80s to today where software is an integral and more often critical part of providing basic functionality as well as enables new functions. It is claimed that the cost of electronics and software is about 40% of total production cost of cars [20]. Close to 90% of new innovation in this sector are and will come from software and electronics [21].

Given the rapid surge in extent of software in the automotive domain and the fact that most automotive companies (henceforth referred as OEMs or Original Equipment Manufacturers) had traditionally been using V-shaped product development with high de-
dependence on suppliers for parts supply, much of the structure has also been inherited to the development of software. The development of software in the automotive domain can be represented as in Figure 1.1.

Figure 1.1: The V-model of product development within the automotive domain.

Broy [2] Identifies following areas as the main source of innovation in near future:

- Crash prevention, crash safety
- Advanced driver assistance
- Adaptable man-machine interface
- Programmable car
- Personalization and individualization, and
- Interconnected car networking.

It can be noted that almost all of the upcoming areas which are posed to bring more innovation in automotive domain are heavily dependent on software, thus software is not only an important part of automotive domain in the present time, but its contribution is certain to increase. Also since new frontiers of innovative functions are inclined towards active safety, driver assistance and even autonomous driving, developing software for safety critical systems and ensuring their dependability characteristics will play an important part within automotive software development.

Further software development process within automotive domain is also different from software developments in other domains,
e.g. software for desktop computers, information systems or web-based services. Firstly as explained above the automotive sector uses V-model for software development using sub-suppliers and/or in-house development and secondly the automotive industry including its sub-suppliers extensively uses model-based design with production code generation [22]. These differences in the type of software and its development process must be accounted for in order to successfully use the software quality and reliability models within this domain.

1.1.2 Software Quality

With increasing importance of software within the automotive domain, the aspects of quality with respect to software also have gained high importance. Quality must be defined and measured if it is to be improved.

While quality is one of the very common and well known terms, yet it is ambiguous and also commonly misunderstood. To many people, quality is similar to what a federal judge once said about obscenity “I know it when I see it” [23]. The main reasons for ambiguity and confusion can be attributed to the fact that quality is not a single idea, but a multidimensional concept, where dimensions includes the entity of interest, the viewpoint and the attributes of that entity [23]. Thus, to fully appreciate the complexities related to quality the shift have been from defining quality from a single perspective towards defining and working with quality models. Quality model according to ISO/IEC 25000 [6] is:

“defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality”

The latest and now widely used framework for software quality is given in SQuaRE: Software product Quality Requirements and Evaluation, ISO/IEC 25000 [6] which provides a series of standards on product quality requirements and evaluation. The organization into different families (or divisions) of standards in SQuaRE is illustrated in Figure 1.2.

Quality model division in SQuaRE series (2501n) provides the detailed software quality model that includes characteristics for internal, external and quality in use. Quality characteristics are further decomposed into subcharacteristics and practical guidance for their use is also provided.
ISO 2501n [24] is based on the ISO/IEC 9126-1 [5] where software quality is into sets of characteristics and subcharacteristics, outlined in Table 1.1 & 1.2.

Depending on the product, the way and/or the environment in which it would be used, different quality measures, characteristics and subcharacteristics can be weighted differently to customize the quality model to specific needs. The characteristics that are most important with respect to this thesis are reliability and safety.

In this thesis we evaluate applicability of software reliability growth models (SRGMs) in the automotive domain. SRGMs help to analyse the maturity and reliability compliance (subcharacteristics of reliability, Table 1) of software under development.

Further in this thesis, we also contributes towards assessment of fault tolerance properties and enhancing safety characteristics by proposing and evaluating framework using fault injection and mutation testing approaches at the functional models level. Fault injection and mutation testing provide tools for assessing the effectiveness of fault tolerance properties of system, while safety (from Table 2) and reliability are the focus of safety case argumentation, verification and validation when developing safety critical software in compliance to ISO 26262.

1.1.2.1 Dependability

While reliability and dependability have been used interchangeably in the literature [25], [26], Avizienis et al. provide a clear model of dependability in [27]. The authors also summarise well the threats as well as means to manage these threats to dependability of a system. The dependability tree from the work is summarised in

Figure 1.2: Organization of SQuaRE series of standards [6].
Table 1.1: ISO/IEC 9126-1 internal/external quality model characteristics and subcharacteristics.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Subcharacteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>suitability</td>
</tr>
<tr>
<td></td>
<td>accuracy</td>
</tr>
<tr>
<td></td>
<td>interoperability</td>
</tr>
<tr>
<td></td>
<td>security</td>
</tr>
<tr>
<td></td>
<td>functionality compliance</td>
</tr>
<tr>
<td>Reliability</td>
<td>maturity</td>
</tr>
<tr>
<td></td>
<td>fault tolerance</td>
</tr>
<tr>
<td></td>
<td>recoverability</td>
</tr>
<tr>
<td></td>
<td>reliability compliance</td>
</tr>
<tr>
<td>Usability</td>
<td>understandability</td>
</tr>
<tr>
<td></td>
<td>learnability</td>
</tr>
<tr>
<td></td>
<td>operability</td>
</tr>
<tr>
<td></td>
<td>attractiveness</td>
</tr>
<tr>
<td></td>
<td>usability compliance</td>
</tr>
<tr>
<td>Efficiency</td>
<td>time behaviour</td>
</tr>
<tr>
<td></td>
<td>resource efficiency utilisation</td>
</tr>
<tr>
<td></td>
<td>efficiency compliance</td>
</tr>
<tr>
<td>Maintainability</td>
<td>analysability</td>
</tr>
<tr>
<td></td>
<td>changeability</td>
</tr>
<tr>
<td></td>
<td>stability</td>
</tr>
<tr>
<td></td>
<td>testability</td>
</tr>
<tr>
<td></td>
<td>maintainability compliance</td>
</tr>
<tr>
<td>Portability</td>
<td>adaptability</td>
</tr>
<tr>
<td></td>
<td>installability</td>
</tr>
<tr>
<td></td>
<td>co-existence</td>
</tr>
<tr>
<td></td>
<td>replaceability</td>
</tr>
<tr>
<td></td>
<td>portability compliance</td>
</tr>
</tbody>
</table>

Table 1.2: Quality in Use according to ISO/IES 9126-1.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality in Use</td>
<td>Effectiveness</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
</tr>
<tr>
<td></td>
<td>Safety</td>
</tr>
<tr>
<td></td>
<td>Satisfaction</td>
</tr>
</tbody>
</table>

Figure 1.3.

Faults, errors and failures pose the main threats to dependability. IEEE standard 1044 (Classification for Software Anomalies) provides common vocabulary for these terms that is useful in making communications in this regard effective. According to the
Figure 1.3: Dependability framework as presented in Avizienis et al. [27].

standard [28]:

- **defect**: An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced.

- **error**: A human action that produces an incorrect result.

- **failure**: (A) Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits Or (B) An event in which a system or system component does not perform a required function within specified limits.

- **fault**: A manifestation of an error in software.

- **problem**: (A) Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use Or (B) A negative situation to overcome.

Since defects are imperfections that deviate software/product behaviour from the expected requirements/specifications, some of these defects can result in failures that violate the safety goals of ISO 26262. Therefore discovering defects and assessing reliability
growth is not only to ensure high quality but also to effectively argue the compliance to ISO 26262.

Software testing is defined as the activity of discover errors [29] which are removed (fixed) once discovered. For latent defects/faults that may remain or manifest during use, systems are designed with fault tolerance characteristics. Further fault forecasting and prevention are other means of ensuring dependability of a system. Various models for fault modelling and forecasting can be used to assess the reliability of system and/or for adjusting the testing effort to ensure desired dependability characteristics at the time of release.

### 1.1.3 ISO 26262

With software playing ever increasing role in the automotive domain and the high quality & dependability demands from automotive software, new standards such as functional safety standard ISO 26262 [29] have been introduced in this sector. ISO 26262 is the adaptation of IEC 61508 [30] to the automotive domain. The standard is applicable for safety related systems that include one or more electrical &/or electronics (E/E) systems. ISO 26262 provides the framework and guideline for the development of hardware and software for components that are deemed safety critical.

ISO 26262 is safety standard based on risk; it provides a framework for analysing quantitatively the risk of hazard during operations and defines safety measures to avoid or control systematic failures. Specifically:

- ISO 26262 provides a safety lifecycle for the automotive domain (including development, management, production, operation, service, decommissioning) and supports the need for customizing necessary activities during the safety lifecycle.

- The standard covers aspects related to functional safety on the entire development process including requirements specification, design, implementation, integration, verification, validation and configuration.

- ISO 26262 provides risk classes based on automotive specific risk, Automotive Safety Integrity Levels (ASILs) which are used to specify necessary safety requirements on the item.
- ASILs are adapted levels of SIL (Safety Integrity Level) for the automotive domain; these are named Quality Management (QM) or ASIL A, B, C & D in the increasing order of risk.

- ASIL for given component/item is determined based on three factors which are severity, probability and controllability, where

  - Severity measures the consequences of an even on health which can range from no injuries to life-threatening injuries.

  - Probability measures the likelihood of occurrence of given event, where a particular or combination of failures would result in safety hazard. The probability is ranked on a five-point scale from incredible to highly probable.

  - Controllability measures the probability of given harm can be avoided by the actions of driver or due to external factors. Controllability is ranked in point scale of four levels from controllable in general to uncontrollable.

- ASILs are then determined based on the levels of severity, probability and controllability using ISO 26262 standard (table 4, part 3).

- The standard also provides requirements for verification & validation and measures necessary for achieving sufficient and acceptable level of safety.

The standard consists of nine normative parts and guideline as the tenth part, these are arranged such as to conform to the widely used V shaped development process within the automotive domain. The overview of the core process of ISO 26262 and how it conforms to the V-shaped product development cycle of automotive domain is apparent from Figure 1.4 as given in the standard [29].

The overview of the ISO 26262 framework can be summarised in following steps.-

- Define the Item representing the system or function.

- Carry out a preliminary hazard analysis (PHA) and risk assessment (RA).

- Based on the outcomes of PHA and RA, appropriate ASIL level is assigned to the item (ASIL is short for Automotive Safety Integrated Levels which are adapted versions of SIL)
3. Concept phase

3a. Item definition
3b. Initiation of safety lifecycle
3c. Hazard analysis and preliminary risk assessment
3d. Assign functional safety requirements

4. Product development at system level

4a. Initiation of product development at system level
4b. Specification of technical safety requirements
4c. System design
4d. Item integration and testing
4e. Safety validation
4f. Functional safety assessment
4g. Release for production

5. Product development at hardware level

5a. Initiation
5b. Specification of hardware safety requirements
5c. Hardware architectural metrics
5d. Evaluation of violations of safety goal due to random hardware failures
5e. Hardware integration and testing
5f. Evaluation of violations of safety goal due to random software failures
5g. Verification of software safety requirements

6. Product development at software level

6a. Initiation
6b. Specification of software safety requirements
6c. Software architectural metrics
6d. Software unit design and implementation
6e. Unit testing
6f. Software integration and testing
6g. Verification of software safety requirements

7. Production and operation

7a. Production
7b. Operation, service and decommissioning

Figure 1.4: Overview of core process of ISO 26262 [29].

- Derive the Safety Goals (SG) from the PHA RA, the safety goals inherit the ASIL levels.
- Derive and assign Functional Safety Requirements (FSR) such that SGs are met.
- Technical Safety Requirements (TSR) are described on how FSR shall be implemented.
- Finally TSR are implemented and the process is supported by appropriate documentation.

Compliance to ISO 26262 means that items deemed as safety critical are assigned appropriate ASIL levels and steps a to g are followed over the full safety lifecycle of the item. Using fault injection and mutation based approaches at the functional models level; software components can be partly verified and validated before implementation. And software reliability growth modelling can be used to model defect inflow from the development/testing phase to assess and monitor the reliability growth of software under development.
1.2 Software Reliability Growth Models

Mathematical expressions used to model reliability growth of software during its development/testing are referred as Software Reliability Growth Models (SRGMs). IEEE standard 1633 [31], recommended practice on software reliability uses following definition for software reliability:

(A) The probability that software will not cause the failure of a system for a specified time under specified conditions. (B) The ability of a program to perform a required function under stated conditions for a stated period of time.

Software Reliability Models (SRMs) are white box methods which use source code metrics for assessment and prediction of software reliability and predicting the defect proneness of given software artefact. On the other hand SRGMs are essentially black box techniques that uses defect discovery (defect inflow/failure data) to model the reliability growth of software. Since black-box models do not need access to source code they are easy to adopt and can be applied widely.

There are wide varieties of reliability models that have been proposed and used in the literature over time, [32] provides a good overview and classification of software reliability models according to which development stage they are applied. The proposed classification is shown in Figure 1.5, accordingly the software reliability growth models used in this thesis are Non-homogeneous Poisson Process (NHPP) based models that uses defect inflow data and which are applicable during the software development and testing phase within the software development life cycle, the reasons for this delimitation is discussed in next section.

1.3 Research Focus and Questions

The first major research goal of this thesis is to evaluate methods which can potentially be used to model or increase the reliability of software in the automotive domain. As explained earlier, to model the reliability of software, there are number of different reliability models that have been proposed and applicable at different development stages. In this thesis we limit our evaluation to software reliability growth models based on defect inflow data from devel-
In the automotive domain most OEMs and their suppliers have widely adopted model based development using domain specific languages such as MATLAB Simulink [22]. The code is generally either auto-generated from the functional models or the sub-suppliers supply the software integrated with the hardware, thus access to source code may be an issue.

Defect discovery data is collected in almost every mature organization developing software and is thus available for reliability analysis. Using such data not only means availability of data for modelling and analysis, but also that if reliability models are found to be practically useful their industrial adoption is highly likely.

The second major research focus of this thesis is to develop framework and methods to increase the reliability of automotive
software. As explained above, wide availability of detailed functional/behavioural models of software system offers high potential for using these to shift some effort on verification and validation from late stages of development to early stages in the development cycle. Also given that late defects have been shown a problem in the automotive domain [33], the need to test models early is apparent and makes high practical sense. Early testing and validation using models also contributes towards verification and compliance to functional safety standards such as ISO 26262 for development of safety critical software.

The two main Research Goals (RG) and corresponding research questions of this thesis can be summarised as follow:

**RG1. Evaluating the applicability of software reliability growth models in the context of automotive software development?**

Research Questions (RQ) for research goal 1:

(RQ1.) Do standard software reliability growth models fit defect inflow data from automotive domain? *(Addressed in chapter 2)*

(RQ2.) What are the differences between the widely used parameter estimation methods and how to effectively measure the predictive accuracy using metrics in software reliability modelling? *(Addressed in chapter 3)*

(RQ3.) Which SRGMs have the best long-term predictive power (using automotive domain data)? And can we use growth rates from earlier projects to increase the asymptote predictions accuracy on on-going projects? *(Addressed in chapter 4)*

**RG2. Propose and evaluate methods that can potentially increase the reliability of software in the automotive domain.**

Research questions for research goal 2:

(RQ4.) How fault injection and mutation testing can be used at model level and how it can be applied within the ISO 26262
1.4 Research Methodology

Research methodology describes the systematic process that is undertaken to yield the sought out research results. It outlines the process, steps taken, practices and methods employed to discover answers to the questions one wish to explore. We use research taxonomy provided in Glass [34] and Wohlin [35]. Depending on the questions of interest, research methods can vary from qualitative to quantitative and empirical to analytical [34], the main research methods accordingly can be represented as in Figure 1.6. Work presented in this thesis is result of five separate studies included in chapters 2 to 6. Though most of these are analytical studies using quantitative data, different methods as represented in Table 1.3 have been used in the five studies that were best suited for the specific research questions of the given study.
1.4.1 Analytical Research

According to Glass [34], analytical research methods include proposing or using an existing theory or set of axioms, develop that theory deriving results and where possible comparing the results using empirical observations. Analytical studies can be done using correlation/regression analysis.

Chapters 2 to 5 in this thesis are analytical studies that test the theory of reliability growth modelling in the context of automotive domain, the results obtained are used to assess and analyse the applicability and effectiveness of these models in this context/domain. Chapter 3 tests analytically two of the most commonly used approaches for parameter estimation when applying SRGMs.
1.4.2 Quantitative Research

We use quantitative research methodology as means to address the RG1 in chapters 2 to 5. The chapters use quantitative data (defect inflow data) and mathematical models (SRGMs) for analysis. Quantitative research is defined as [36]: “Explaining phenomena by collecting numerical data that are analysed using mathematical based methods (in particular statistics)”. Although quantitative methods collect data from number of cases and interesting patterns can be reviled using statistical methods, these methods are not appropriate for gaining deeper understanding of underlying reasons. Thus while quantitative methods can be effectively used to evaluate established or proposed theories, but they are not good methods for explaining why questions where qualitative methods such as interviews and case studies are useful.

1.4.3 Case Study

Case study are empirical studies mostly using qualitative data, these are generally used for monitoring projects, activities or assignments [35]. According to Yin [37] case study is an empirical investigation of contemporary phenomenon within a real-life context. Although the level of control is less compared to experiments in a case study, it has a strong focus on empiricism and thus effective for tracking a specific attribute or establishing relationships between different attributes in real-life situations. The strong emphasis on understanding the context while makes case study very suitable for industrial evaluation of software engineering methods and tools [35], the greatest weakness of this method is low power of generalizability.

Case study methodology is used in chapter 5 to understand the current state of best practices with respect to software development within automotive domain as well as the needs in relation to safety critical software development. Case study method is also utilized in chapter 6 to describe the problem and provide the assessment of proposed solution with the help of an example.

1.4.4 Interviews

Interviews are frequently used data collection method employed for collecting qualitative data [38]. Interviews can either be very for-
mal (structured) to quite informal discussion between interviewer and interviewee (open interviews) [39].

A structured interview is the type of interview that is closely structured according to an interview guide which is maintained and reused for all interviews with different interviewees. The interviewer is not allowed to deviate from the questions already planned in the interview guide and follow up questions are also not entertained. Semi-structured interviews do not impose such strict constraints and the interviewer is allowed to ask follow-up questions when necessary in addition to following predetermined questions from the interview guide. The method allows for getting deeper understanding where necessary but also makes it difficult to compare the results. Open interviews are basically an open discussion between the interviewer and interviewee and best to understand the context and getting familiar with an unfamiliar environment.

In this thesis, open interviews were used initially to understand the software development and testing process within automotive sector which provided the context for chapters 2, 4 and 5. Semi-structured interviews were also used as part of initial industrial validation for chapter 6.

1.4.5 Research Validity

For any research, a fundamental question with respect to its results is their validity i.e. how valid the results are. Validity is applicable to both the design of conducted study and the research methods used. Different kinds of factors can affect any research and invalidate the findings [40], thus a primary requirement for doing sound research is controlling all possible factors that can threaten research’s validity. There are different classifications for various types of research validity threats, Cook and Campbell [41] define four types of validity threats that are important with respect to empirical and analytical studies conducted within the discipline of software engineering, these are conclusion, internal, construct and external validity.

**Conclusion validity** measures to what degree the conclusions of given research are reasonable. Conclusion validity only concerns with whether there is a relationship or not and if the relationship concluded is reasonable.

Conclusion validity of the work presented in this thesis is high,
1.4. RESEARCH METHODOLOGY

analytical studies (chapters 2-4) reports evaluation results using sound statistical tests and power.

**Internal validity** is concerned with causality of relationship between dependent and independent variables. The causality can also be checked between the treatment given in an experiment (change in factors/variables) and the outcome measured. The threat of internal validity can be minimized by carefully planning and verifying the causal relationships based on earlier research or using statistical instruments to prove the causality directions/relationships.

Given the nature of individual studies included in this thesis, internal validity does not pose major threats, where applicable the measures and variables used in particular study are grounded in established theory and earlier reported studies.

**Construct validity** is concerned with relationship between theory and observations; It refers to the ability of a measurement tool (e.g., a survey, test, etc) to actually measure the variable of interest. In other words, construct validity deals with question of does it properly measure what it’s supposed to measure for both independent variable (cause) and dependent variable (the effect).

In the work presented here, construct validity is most relevant to question whether number of defects reported actually reflects the measure of software reliability objectively or not. This threat could be minimized by adopting standardised definition of defect (such as in IEEE standard 1044 [28]). Further the relationship between total number of reported defects and software reliability can be carefully examined in earlier published work and by using empirical observations from historical projects of company used in the study.

**External validity** examines the generalizability of results and is very important in the context of studies in software engineering. If studies are done that are dependent on the context such as case studies, using interview or other qualitative data collection methods, care need to be taken when these results are extended to be generalised over wider area than specific to the context of the given study.

In this thesis, threats to external validity are kept in check by limiting our conclusions to a specific domain where necessary. For
examples in chapter 2 and 4, the defect inflow data used comes from automotive domain and thus the results obtained are not claimed for outside this domain. At the same time a threat to external validity remains with regard to results generalizability throughout automotive domain when defect data used was from a single company from this sector.

1.5 Summary of contributions of each chapter

This section summarises the studies included in this thesis. The research goals, questions, context, results and contributions for each study is described very briefly.

1.5.1 Chapter 2: Evaluation of standard reliability growth models in the context of automotive software systems

In this chapter we evaluate eight commonly used software reliability growth models on defect inflow data from a safety critical software system from the automotive domain. Given the scarcity of domain specific evaluation studies of reliability modelling the contribution of this paper was to evaluate these models in the automotive domain and comment about their applicability.

The results showed that while these models can be fitted to the defect inflow data from the automotive domain, for better fit and wider applicability some parameters including the testing effort need to be accounted and manually adjusted.

1.5.2 Chapter 3: Comparing between Maximum Likelihood Estimator and Non-Linear Regression estimation procedures for Software Reliability Growth Modelling

This chapter compares two of the most widely used and recommended parameter estimation procedures/methods while applying software reliability growth models. The Maximum likelihood estimation (MLE) method is highly recommended, but at the same
time it is difficult to apply and lacks extensive tools support that limits its use in the industrial domain. Non-Linear Regression (NLR) is based on least square methods and offers an easy alternative estimation method.

The chapter contributes by analytically comparing the parameter estimates obtained from these two methods for same data set and also comparing them to results obtained via empirical equations and those reported in earlier study. Comparative studies are important for making an informed choice between the two methods by helping enhancing the knowledge regarding characteristics, advantages and limitations of each method. The chapter also proposes a new metric to measure the asymptote accuracy that is balanced for over and under prediction, which was not the case for metrics often used and recommended in earlier studies.

1.5.3 Chapter 4: Evaluating long-term predictive power of standard reliability growth models on automotive systems

Chapter 4 builds on the results from chapter 2 and increases the external validity of the same, while making further distinct contributions by evaluating long-term predictability of standard reliability growth models in context of automotive domain. Using data from four large automotive software projects, applicability of commonly used software reliability growth models is assessed.

This chapter makes two distinct contributions towards wider adoption of reliability modelling within automotive domain. Firstly the applicability of SRGMs for test effort allocation is assessed by evaluating the long-term predictive power of reliability models within this domain. Secondly another important practical question with regard to possibility of using historical information (in form of growth rate) from earlier project to improve the predictive accuracy of model on current projects is also tested in this chapter.
1.5.4 Chapter 5: Increasing Efficiency of ISO-26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model Based Development

This research presented in this chapter is a result of stimulations from observations of current practices and identifying major needs within the automotive domain. There is high prevalence of model based development in this domain and need for compliance to functional safety standard, ISO 26262 for development of safety critical systems.

The chapter contributes with a framework which combines methods of fault injection and mutation testing approaches to be used at the model level that can be used for increasing the efficiency of ISO 26262 compliance. The compliance to ISO 26262 standard can be achieved by shifting some of the verification and validation from late stages to early development stages by using functional models which are detailed enough for such analysis.

The framework is encouraging addition to the automotive domain, where earlier studies [33] have shown that late defects is still a problem that affects the development of software in this domain. Shifting some of the verification and validation to early stages (at model level) will thus not only aid in argumenting effectively the compliance to ISO 26262, but will also address the more common problem of late defects that puts high pressure on development teams close to release.

1.5.5 Chapter 6: Improving Fault Injection in Automotive Model Based Development using Fault Bypass Modelling

Chapter 6 introduces and evaluate a new approach Fault Bypass Modelling. When testing software systems in form of their model equivalents (higher level or functional/behavioural models), we generally need environment models to simulate the surrounding environment. It is noted that system environment models generally work well under normal operating conditions, but under the fault conditions unrealistic feedback loops may cause the system environment model to either break down or produce incorrect results.
The chapter provides a proof-of-concept for fault bypass modelling which offers a simple but effective way of building environment models that will not suffer from such problems. Given one of the main interest of building functional/behavioural models is to be able to analyse systems response under faulty conditions and to test what-if scenarios; thus fault bypass principle provides a possible solution for problem of high practical relevance. The framework is evaluated using a simple model level and initial validation has been done with our industrial partner.
Bibliography


[73] V. Rupanov, C. Buckl, L. Fiege, M. Armbruster, A. Knoll, and G. Spiegelberg, “Early safety evaluation of design decisions in e/e architecture according to iso 26262,” in *Proceed-


Chapter 2

Paper A

Evaluation of standard reliability growth models in the context of automotive software systems


Abstract

Reliability and dependability of software in modern cars is of utmost importance. Predicting these properties for software under development is therefore important for modern car OEMs, and using reliability growth models (e.g. Rayleigh, Goel-Okumoto) is one approach. In this paper we evaluate a number of standard reliability growth models on a real software system from automotive industry. The results of the evaluation show that models can be fitted well with defect inflow data but certain parameters need to be adjusted manually in order to predict reliability more precisely in late test phases. In this paper we provide recommendations for how to adjust the models and how the adjustments should be used in the development process of software in the automotive domain by investigating data from an industrial project.


2.1 Introduction

Software plays a significant role in modern cars. In past few decades the amount and importance of software in cars has increased exponentially [1], to the extent that today’s premium cars carry more than 70 ECUs and software of the order of over ten million lines of code (SLOC) [2]. Software is not only replacing traditional models of control systems but today it is at the heart of providing new functionality and driving innovation. With the rapid growth in significance of software in automotive industry there are a number of challenges the industry faces in developing and maintaining good software for modern cars [2] [42].

Automotive software differs from software in other sectors due to stringent demands for rapid development, need for cost effective development, and high demand for innovation and need of high quality and reliability, especially for applications, which are deemed safety critical. To ensure that cars are safe for drivers, occupants and other road users as well as to maintain the consumer confidence, the quality and reliability demand for safety critical software is very high. Functional safety standards such as ISO 26262 [29] provide strict guidelines for the development of software for safety critical applications with significant emphasis on ensuring reliability.

Software reliability growth models (SRGMs) have been used to assess the maturity of software for number of years. Efficient estimation of latent defects in software is valuable information, test managers can use this information to make important decisions not only to ensure optimal resource allocation but also to decide when the given software is ready for release [43]. Applying SRGMs for estimating reliability in industrial applications needs careful consideration to the applied model assumptions, data availability and predictive power, but proper use of SRGMs provides several benefits for developing high quality and reliable software.

2.2 Related Work

Over the years, a number of SRGMs has been presented [44], although similar extent is lacking in the comprehensive evaluation of these models on industrial domain specific applications. This is especially true for the automotive sector. Different industrial
domains have very different demands for its software and the development process also varies to a large extent, not all SRGMs would be suited for every sector. Woods [19] applied eight SRGMs on software products from industry and showed that defects predicted based on cumulative defects matches well with after release defects. Staron & Meding [45] evaluated SRGMs on large software projects in the Telecom sector and proposed a new model based on historic trends data. In this paper we apply common SRGMs on a large project from the automotive sector and evaluate it on simplest fit measure. The applications of SRGMs in automotive software projects are very scarce and with increasing dominance of software in the automotive industry, the need and importance of such studies is very apparent.

In [17], authors present a review of common Non-Homogeneous Poison Process (NHPP) based software reliability models and compare their performance on real time control system. We evaluate SRGMs with only two and maximum three parameters, which are easy to implement and intuitive to understand, this also means that these models can be easily adopted in the industry.

The automotive domain in itself is quite unique, firstly the industry due to various reasons including the historic factors is driven by the V development model with high dependence on suppliers, this has also become true to a large extent for the development of software within this domain. Secondly automotive unlike some other industries and like many other similar sectors have widely adopted the model based development approach. Additionally within the Original Equipment Manufacturers (OEMs) there exist numbers of different departments/teams (for example Power-train, Central Electric Module, Infotainment etc.) which develops quite different type of software products and works in quite different working environments. Currently there is also significant trend in automotive domain towards being more agile in their software development process. All these factors affect the defect inflow profiles and the use of SRGMs needs to take these factors into consideration for successful application. In this paper we give a way forward for effective implementation of SRGMs in the automotive sector, what needs to be emphasized and what would lead to optimal software reliability modelling in this domain.
Table 2.1: Software Reliability Growth Models used in the study.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Type</th>
<th>Mean Value Function</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel-Okumoto (GO)</td>
<td>Concave</td>
<td>$m(t) = a(1 - \exp(-bt))$</td>
<td>[13]</td>
</tr>
<tr>
<td>Delayed S-shaped model</td>
<td>S-shaped</td>
<td>$m(t) = a(1 - (1 + bt) \exp(-bt))$</td>
<td>[46]</td>
</tr>
<tr>
<td>Rayleigh model</td>
<td>Concave</td>
<td>$m(t) = a \exp(-\frac{b}{t})^2$</td>
<td>[23]</td>
</tr>
<tr>
<td>Inflection S-shaped model</td>
<td>S-shaped</td>
<td>$m(t) = \frac{a(1-\exp(-bt))}{1-\beta \exp(-bt)}$</td>
<td>[17]</td>
</tr>
<tr>
<td>Yamada exponential imperfect debug-</td>
<td>S-shaped</td>
<td>$m(t) = \frac{ab}{\alpha+b}(\exp(\alpha t) - \exp(-bt))$</td>
<td>[47]</td>
</tr>
<tr>
<td>Yamada linear imperfect debugging</td>
<td>S-shaped</td>
<td>$m(t) = a(1 - \exp(-bt))(1 - \frac{\alpha}{b}) + \alpha at$</td>
<td>[47]</td>
</tr>
<tr>
<td>Logistic population model</td>
<td>S-shaped</td>
<td>$m(t) = \frac{a}{1-\exp(-b(t-c))}$</td>
<td>[48]</td>
</tr>
<tr>
<td>Gompertz model</td>
<td>S-shaped</td>
<td>$m(t) = a \exp(-b \exp(-ct))$</td>
<td>[49]</td>
</tr>
</tbody>
</table>

2.3 Research context and method

We use data from a large project within the development of an active safety function from our industrial partner, Volvo Car Corporation (VCC) from the automotive sector. Department of Active Safety within VCC develops functions/features such as driver alert control, collision warning, lane departure warning etc. The defect data has been used earlier in [33] in a study that introduced a new lightweight defect classification scheme LiDeC. We use dynamic software reliability growth models that have been reported in many earlier studies and are summarized in Table 2.1.

To fit the models to our data we used non-linear regression (NLR) routine of the commercially available statistical software package, IBM SPSS. The starting values we used are same for all models and iterations are done until the reduction between successive residuals errors is less than $1.0 \times 10^{-08}$. Models with two and three parameters were used in fitting of the curves as these param-
eters could be interpreted empirically (for instance with respect to the testing effort or maximum number of defects). The models were built based on the data set from all the development phases of the system - starting at requirement analysis and ending with vehicle production testing (i.e. excluding the post-release defects).

2.4 Results and interpretation

The fitting of different SRGMs (two and three parameter models) on actual data is presented in Figure 2.1 and Figure 2.2, due to confidentiality reasons the Y-axis scale is not presented and time scale is trimmed at beginning and end representing only partial data for illustrating the fit of the used models. For fitting the model, however, the full data set was used.

![2-Parameter models](image)

Figure 2.1: Two parameter software reliability growth models applied to data set from automotive software project.

Although (as shown in Fig 2.1) the models fit the data, they have a tendency of growing exponentially. The exponential growth gives unrealistically high values of asymptotes (maximum predicted defects), such growth is not possible in practice the number of defects discovered late in the projects decreases over time, thus giving the well-known S-shape of the cumulative defect inflow profile. This shortcoming can be overcome by using three parameter
models which include the $\alpha(t)$ parameter. The additional parameter is meant to describe the function of test progress over time, and therefore provide more accurate results with logical empirical explanations. Figure 2.2 presents these models.

![Figure 2.2: Three parameter software reliability growth models applied to data set from automotive software project.](image)

The analysis of the models and their fit, as shown in Figure 2.2, suggests that the $\alpha(t)$ parameter is promising and will be used in our further analyses. Using the Mean Square Error (MSE) measure to analyse the goodness-of-fit of the models (shown in Fig 2.3) we observed that the most accurate model was the InflectionS model and the logistic model (used to model population growths in general [50] [51]).

MSE presented in Figure 2.3 for the simplest and one of the earliest Goel-Okumoto (GO) model was approximately 10 times larger than the rest of the models thus we excluded it from the chart to rescale it and focus on the remaining models. As expected, the three parameter models generally fit better than two parameter models, but we observed one exception - the DelayedS model fits better than the Yamada exponential imperfect debugging model (Y-ExpI) and Yamada linear imperfect debugging model (Y-LinI), both of which attempts to account for the testing effort using a third parameter. This means that our initial results should be complemented with more accurate model of the testing effort.
Another significant observation is with respect to the three parameter general logistic model, which performs best among models used in this study with respect to minimum MSE criteria, despite this model not being widely used for software reliability modelling. The three parameters general logistic model is used in many applications and domains but not as widely in the software reliability modelling although it yields relatively accurate results. Our observation suggests that traditional three parameter models such as logistic and Gompertz model provides superior fit to our data from automotive domain software project. InflectionS model also does very well in MSE fit criteria with MSE only higher than logistic and lower than that using Gompertz model.

2.5 Conclusions

A number of SRGMs have been proposed and evaluated over time. It is noted here that despite software being dominant in modern automotive industry there is a gap in studies evaluating the application of SRGMs in this domain. In this paper we take a step in direction of addressing this gap by applying eight common SRGMs on defect data from a large automotive software project and evaluating their fit using MSE criteria. We, furthermore, provide a way forward for effective application of SRGMs in automotive software
reliability modelling which are as follows:

- It was observed that simple two parameters models can provide good fit (with exception of the GO model), but the asymptotes obtained might be unrealistic;

- Logistic and inflectionS models had the best fit to our data among the different models tried;

- Since one of the important factors for successful use of SRGMs is to use appropriate time scale, we identify that modelling the change of testing effort over time (generally done using parameter $\alpha(t)$) will be critical in applying SRGMs within automotive sector;

- Using parameter estimates from two parameter models and based on historic values one could also model/predict the testing effort i.e. $\alpha(t)$ for the current project which would give useful insight to project managers for optimizing the resource allocation going forward.

Realistic accounting of testing effort will help us to fit the SRGMs to actual defect inflow data. Finding the models, which provide the best fit, have superior predictive power, and use the data in its available form will significantly enhance the adoption of software reliability modelling in industries where software is starting to play a critical role. And customizing the SRGMs to conform to given industrial domains such as automotive sector will provide a powerful tool to test and quality managers within these industries to use them for optimal resource management, increasing the quality and reliability, and ensuring timely delivery of high quality software.
Chapter 3

Paper B

Comparing between Maximum Likelihood Estimator and Non-Linear Regression estimation procedures for Software Reliability Growth Modelling


Abstract

Software Reliability Growth Models (SRGMs) have been used by engineers and managers for tracking and managing the reliability change of software to ensure required standard of quality is achieved before the software is released to the customer. SRGMs can be used during the project to help make testing resource allocation decisions and/or it can be used after the testing phase to determine the latent faults prediction to assess the maturity of software artefact. A number of SRGMs have been proposed and to apply a given reliability model, defect inflow data is fitted to model equations. Two of the widely known and recommended techniques for parameter estimation are maximum likelihood and method of least squares. In this paper we compare between the two estimation procedures for their usability and applicability in context of SRGMs. We also highlight a couple of practical considerations, reliability practitioners must be aware of when applying SRGMs.
3.1 Introduction

Software is plying an ever increasing role in our day today life. Most of the products and services we consume are now based on software or uses software in certain ways [52]. Over the years the complexity of software artefacts has been growing rapidly, while at the same time the demands for dependability of software systems have also increased. The link between complexity and software faults have been suggested for long, studies as early as 1980s such as [53] suggest that software complexity often affects its reliability. Thus while it is important to keep the complexity of software under check, it is also important to tack and monitor their reliability growth.

Software testing is still the main source of ensuring reliability and quality of software systems. Testing in the area of software products is highly resource intensive exercise, some of the estimates put it around 50% of overall development cost [54]. But testing resource consumptions can be much more resource/cost efficient, if project managers are able to plan testing activities well [55]. Software reliability growth models have been used to estimate the reliability change in software products and use the reliability growth predictions for making testing resource allocation decisions. Since the software can rarely be made fully error free, project managers need to balance costs associated with software testing to cost of fixing bugs after release [56].

Software reliability can be modelled using reliability models which can be based on Non-Homogeneous Poisson Process (NHPP), Markov process or Bayesian models. One of the major difficulty faced when using Markov and NHPP models is with their parameter estimation [11].

A number of difficulties that may be encountered when applying SRGMs to defect data; in this paper we explore practical considerations when using two types of estimators Non-Linear Regression and Maximum Likelihood Estimator. We compare between the two and introduce a measure for assessing the predictive power of reliability models. The data used for this study is time-domain failure data for a real-time control system provided in [57] and used in many earlier studies including [17], [58]. In the data 136 faults have been reported with their time between failures (TBF). In the next section we describe the basics of SRGMs and list related work, section 3 outlines the research questions and methodology
while the following section (4) is used to present the results. The paper is summarized in section 5 with conclusions and directions for future work.

3.2 Background

3.2.1 SRGMs: Software Reliability Growth Models

Software reliability engineering tends to focus on using engineering techniques for assessing and improving the reliability of software systems during development and post development. A roadmap on the software reliability engineering is presented in [44]. Application of empirical reliability engineering techniques have led to two basic categories, the first class of models called software reliability models (SRMs) are static models that uses attributes of software source code to assess or predict its reliability, while the software reliability growth models (called SRGMs) or the dynamic models generally uses statistical distributions of the defect inflow patterns to estimate/predict the end-product reliability [23]. The SRMs and SRGMs could also be differentiated based on their access to source code which former being a white box models while the latter being black box modelling of software reliability. We focus on SRGMs in this study.

3.2.2 Model Selection

Since the start of reliability modelling within software domain in early 1960s [59], a number of SRGMs have been proposed and evaluated [57]. With so many models which generally differ from one another on their assumptions about underlying software development and testing process, model selection has been a critical challenge. Studies such as by Goel [15] and Musa [60] have shown that different models/families of models are better suited than others for certain applications. A number of studies have also looked into the questions of model selection and suggested various solutions. Sharma [32] recommends that different models should be first compared and evaluated before making a selection. Stringfellow and Andrews [43] presented an empirical method for selecting the SRGM using a proposed criteria and iteratively applying dif-
ferent models, while Khoshgoftaar and Woodcock [61] supports using Akaike Information Criteria (AIC) which is based on the log-likelihood function as a tool to select the best model for given application/data.

### 3.2.3 Comparing between SRGMs

One common way to understand the differences between different models and their ability to fit and predict given defect data is to do comparative studies. A number of NHPP based SRGMs have been reviewed and compared on their fit and predictive power by Pham [17]. Ullah et al. [18] also present a study comparing eight SRGMs onto large dataset consisting of fifty defect data from industrial and open source projects. Other studies have also evaluated and compared different SRGMs on industrial data, Wood [19] made comparison of eight SRGMs on defect inflow data and found it correlated with post release defects. Staron and Mending [45] evaluated different SRGMs using large software projects from telecom sector, while in [62] seven SRGMs have been evaluated for their applicability within automotive software projects and long-term predictive power. SRGMs comparison and use in practice for embedded software in consumer electronics is also presented in [63]. Although a number of studies have compared and evaluated different SRGMs within different context, we are still far from making a consensus on how to select SRGMs for given purpose and which models are best for given process characteristics. The situation with different SRGMs comparison is very well summarised by Stephan Kan as: “Some models sometimes give good results, some are almost universally awful, and none can be trusted to be accurate at all times.” [23].

### 3.2.4 Parameter Estimation

Two practical and important challenges faced when applying SRGMs in practice/industry are the process to be followed and how to estimate the parameters. IEEE standard 1633: recommended practice on software reliability [31] provides a 13-steps procedure on assessing and predicting the software reliability. The standard also lists three methods commonly used for parameter estimation when using SRGMs as: method of moments, least squares and the maximum likelihood estimation. Maximum likelihood estimation is the
recommended approach by the standard and by the various studies introducing new SRGMs [15], [13], [46].

Parameter estimation using Maximum likelihood estimation requires solving sets of simultaneous equations to maximize the likelihood of defect data coming from given function (model equation) to find the parameters. Although MLE fulfils number of important statistical properties of optimal estimator and thus considered the best estimator for large data, unfortunately the set of equations used to find parameters using MLE are very complex and usually need to be solved numerically [60], [19], [64]. This is a practical issue that limits the use of MLE by industrial practitioners who may not be trained to use sophisticated statistical modelling required to use MLE for different SRGMs. The problem of using MLE widely for parameter estimation is further compounded either due to SRGM models with complex log-likelihood functions and cases where MLE does not converge to give unique estimation of unknown parameters. Meyfroyt [65] provides necessary and sufficient conditions for ensuring unique, positive and finite estimation of parameters using MLE for Goel-Okumoto, Yamada S-shaped and Inflection S-shaped models. Use of MLE in industry is further restricted due to lack of commercial tools which can provide reliable MLE parameter estimation for different SRGMs. On the other hand the least square estimation uses curve fitting to the observed data for making estimation of unknown parameters. Parameters values are estimated for curve that gives minimum sum of square of errors, i.e. curve that fits best (with respect to sum of squared errors). Given the nature of common SRGMs the least square estimation usually leads to using non-linear regression (NLR) for estimating the unknown parameters. Contrary to MLE, least square estimation is easy to apply, and NLR is often available as standard routine in most commercially available statistical packages.

Wood [19] applied both MLE and least square estimation and found least square predictions to be more stable and better correlated to field data although MLE results were more reasonable. He also noted major difference between the confidence intervals where least square estimates were unsatisfactory, and while MLE confidence interval estimates were realistic they were too wide to make practical conclusions.

It can be safely assumed that statistically MLE is much better parameter prediction procedure than least square, but the least
square is much easier and provide consistent results in wider data sets and thus a preferred method of choice by industrial practitioners. Also in certain cases where MLE cannot provide the parameter estimations, least square approach is the natural alternative. Thus the least square estimator/NLR is also used more often than MLE for studies evaluating different SRGMs over large datasets [18], [62]. Given the differences between the two estimators the need to understand the applicability and performance differences of these two estimators is quite apparent.

3.3 Research Context and Method

3.3.1 Research Objectives

In this study we take a look at some of the practical considerations and questions faced by software reliability practitioners. The objective is mainly to document these aspects and mark their importance. Mainly we look at:

(i) Comparing MLE verses NLR procedure for estimation of unknown SRGM model parameters.

(ii) Assessing predictive accuracy using predicted relative error metric.

(iii) Working with un-grouped data.

We also comment on reproducibility of earlier studies from literature and provide directions for further research.

3.3.2 SRGMs and Data

In this study we use three of the very early and widely used software reliability models, the SRGMs used and their mean value functions are listed below in Table 4.1. The main reason for their selection is their wide familiarity and availability of MLE simultaneous equations. The mean value functions have parameters a, which refers to total number of predicted defects and b, which is generally the shape parameter or growth rate parameter. Parameter $\beta$ in Inflection S-shaped model is assumed to be 1.2 following the earlier studies [17].
### Table 3.1: Summary of SRGMs used in this study.

<table>
<thead>
<tr>
<th>No</th>
<th>Model Name</th>
<th>Mean Value Function</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Goel-Okumoto (GO)</td>
<td>( m(t) = a(1 - e^{-bt}) )</td>
<td>[13]</td>
</tr>
<tr>
<td>2</td>
<td>Delayed S-shaped model</td>
<td>( m(t) = a(1 - (1 + bt)e^{-bt}) )</td>
<td>[46]</td>
</tr>
<tr>
<td>3</td>
<td>Inflection S-shaped model</td>
<td>( m(t) = \frac{a(1-e^{-bt})}{(1+\beta e^{-bt})} )</td>
<td>[66]</td>
</tr>
</tbody>
</table>

The data used for this study is time-domain failure data for a real-time control system provided in [57] and used in many studies including [17], [58]. In the data 136 faults have been reported with their time between failures (TBF). For practical reasons we also assume 136 to be the real asymptote of given data, i.e. actual total number of defects. Cumulative time obtained by successively adding TBF is used for fitting the cumulative distribution functions to different SRGMs. 122 failures are used for fitting the data and making parameter estimates, while the rest are used to evaluate the predictive power.

#### 3.3.3 Data Analysis Techniques

To ensure high reproducibility we list all the data analysis techniques and equations used for analysis in this study with their references.

1) For parameter estimation using least squares we use Non-Linear Regression routine available in statistical package IBM SPSS, the starting values provided were \((a = 120 \text{ and } b = 0.0001)\) and iterations were done until successive residual errors difference was less than \(1.0E-08\) (default value in SPSS).

2) For parameter estimation using MLE, we use package maxLik, a package for statistical environment R [67]. The optimization method used was Nelder-Mead (NM) and the starting values provided were same as those used for NLR routine.

3) We also compare the parameter estimations obtained by above methods (NLR and MLE using maxLik) with earlier study by Pham [17] using the same data.
4) To make the two estimators comparison even more robust, we further use the non-linear simultaneous equations for getting the analytical solution using MLE. The equations are available for Goel-Okumoto model and Delayed S-shaped model described in [68] and reproduced below:

For GO model:

\[
\frac{n}{a} = 1 - e^{-b s_n}
\]

\[
\frac{n}{b} = \sum_{i=1}^{n} s_i + a s_n e^{-b s_n}
\]

For Delayed S-shaped model

\[
\frac{n}{a} = 1 - (1 + b s_n) e^{-b s_n}
\]

\[
\frac{2n}{b} = \sum_{i=1}^{n} s_i + a b s_n^2 e^{-b s_n}
\]

Where \( n \) represents number of failures reported; time between failures is represented as \( \{t_k; k = 1, 2, ..., n\} \) and where time to \( k^{th} \) failure is given by \( s_k = \sum_{i=1}^{n} t_k \); for details refer to [68]. Equations (1) & (2) or (3) & (4) can be solved simultaneously to obtain the point estimates of parameters \( a \) and \( b \). We used Matlab fsolve to solve system of non-linear equations given above.

5) To make comparison of asymptote prediction accuracy, we use the metric Predicted Relative Error (PRE), which is described in the IEEE standard 1633 and also used in earlier studies as measure of prediction accuracy [18]. PRE is defined as ratio between predicted error (predicted minus the actual asymptote) to the predicted number of failures.

\[
PRE = \frac{(Predicted - Actual)}{Predicted}
\]

PRE resolves a common problem with using relative error for comparing between different models prediction, the relative error is the ratio of prediction error over actual value and thus if the predicted value is much larger (in multiples) than the
actual value, relative error can be greater than 100%. PRE provides a comparative scale between [-1 1] or [-100% 100%], where value close to zero means better predictive accuracy and closer to +/-100% is as worse prediction as it can get. Although we identify one major problem with PRE, which is: It provides asymmetric value based on over or under prediction. The problem can be easily understood using a simple example. Let us assume actual value be a and case1: the predicted value is 20% higher than actual i.e. 1.2a; for case2: the predicted value is 20% lower than actual (i.e. -20% of actual or 0.8a).

Now applying PRE to case1 and case2, gives PRE values:-

PRE (case1): \( \frac{(1.2a-a)}{1.2a} = \frac{0.2}{1} = 0.16667 \) or 16.667%, while for

PRE (case2): \( \frac{(0.8a-a)}{0.8a} = -\frac{0.2}{0.8} = -0.25 \) or -25.00%

To make PRE symmetric and thus give consistent value for over and under estimation we define BPRE, referring to Balanced Predicted Relative Error, as follows:

\[
BPRE = \frac{\text{Predicted} - \text{Actual}}{\eta \times \text{Predicted} + (1 - \eta)(2 \times \text{Actual} - \text{Predicted})},
\]

where \( \eta = \begin{cases} 
1 & \text{if Predicted > Actual} \\
0 & \text{if Predicted < Actual} 
\end{cases} \)

Now applying above defined BPRE to same case1 and case2, gives BPRE values:-

BPRE (case1): \( \frac{(1.2a-a)}{1.2a+0} = \frac{0.2}{1.2} = 0.16667 \) or 16.667%, and

BPRE (case2): \( \frac{(0.8a-a)}{0+1*(2a-0.8a)} = \frac{-0.2}{1.2} = -0.16667 \) or -16.67%

Miyazaki et al. [69] defined a balanced relative error metric \( R_i \), also referred as Balanced Relative Error, BREbias defined as:
\[ R_i \text{ or BRebias} = \frac{(Actual - Predicted)}{(\min(Actual, Predicted))} \]

Our metric BPRE is similar to \( R_i \), but different in the sense that while \( R_i \) is unbounded on both sides, BPRE is bounded \([-0.5, 1)\) which is useful to make comparisons when deviations are particularly large compared to actual values.

6) To compare the model fitting to data for both fit and predicted values, we use another widely used metric, Mean Square Error (MSE). Mean square error measures the average deviations between the predicted and actual values [70], thus a measure of fit, it is given by:

\[
MSE = \frac{1}{k-q} \sum_{1}^{k} (a_i - p_i) \tag{7}
\]

Where \( a_i \) is actual values, \( p_i \) predicted values for data set of size \( k \) and \( q \) is the number of parameters.

### 3.4 Results

#### 3.4.1 Parameter estimation using MLE and NLR estimation.

Parameter estimation using maximum likelihood and non-liner regression procedure are summarised in TABLE 4.2. The table also provides comparison of parameters values obtained in study using same data by Pham [17] and also by solving MLE simultaneous equations provided in [70].

<table>
<thead>
<tr>
<th>Asymptote</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geel-Okumoto</td>
<td>132</td>
<td>114.05</td>
<td>125</td>
<td>139.37</td>
</tr>
<tr>
<td>DelayedS</td>
<td>132</td>
<td>103.33</td>
<td>140</td>
<td>125.16</td>
</tr>
<tr>
<td>InflectionS</td>
<td>132</td>
<td>107.60</td>
<td>133.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Growth Rate</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geel-Okumoto</td>
<td>3.80E-05</td>
<td>6.07E-05</td>
<td>6.00E-05</td>
<td>3.65E-05</td>
</tr>
<tr>
<td>DelayedS</td>
<td>9.73E-05</td>
<td>1.66E-04</td>
<td>7.00E-05</td>
<td>9.76E-05</td>
</tr>
<tr>
<td>InflectionS</td>
<td>5.79E-05</td>
<td>1.07E-04</td>
<td>7.00E-05</td>
<td></td>
</tr>
</tbody>
</table>
Form TABLE 4.2 it can be observed that the asymptote (a, total number of predicted defects/failures) predictions obtained in this study using maximum likelihood estimator utilizing package maxLik gives very consistent results for all three models. While the asymptote predictions using nonlinear regression routine (NLR) varies much more with minimum prediction being 103 for Delayed S-shaped model and 114 for GO model. It is further interesting to note that significant differences are also observed between our predictions using (MLE) and values obtained by earlier study by Pham, although in both case the estimator used is the same (MLE). The difference observed here may be attributed to difference in tools used or the starting values predicted. Given that the tool used and starting values details are not available for earlier study, it is difficult to verify the source of this observed difference.

Predictions for growth rate parameter (b) with different estimators are also listed in TABLE 4.2. While there are variations between different models growth rates obtained in this study using MLE and NLR. The growth rate is predicted to have highest value for Delayed S-shaped model and lowest for GO model using both (MLE & NLR) estimators in our study. The growth rates predicted in Pham study are closer to each other. It can also be noted that for both asymptote and growth rate, our estimates using MLE are very close to the parameters estimates obtained using MLE simultaneous equations described earlier. The fitting of predicted models using different estimators to actual data is also represented in Fig 4.2, Fig 4.3 and Fig 4.4.

### 3.4.2 Predictive Accuracy using Predicted Relative Error (PRE) and unbiased PRE (BPRE)

We now compare the predictive accuracy of asymptote values obtained using MLE estimator to NLR estimators.

It is interesting to note from TABLE 4.3 that all but one estimate under predicts for given dataset. Using PRE and BPRE values for same parameter predictions we can also see that BPRE gives better and more accurate representation of undervalued asymptote prediction as described in the section 3. The BPRE values for asymptote predictions using MLE and NLR are also presented below in Fig 4.5. We also add two more models using NLR procedure to make further check.
Figure 3.1: Goel-Okumoto model fitting to data with different estimators.

Figure 3.2: Delayed S-shaped model fitting to data with different estimators.

Figure 4.5 shows that in our study although MLE estimators also under predict asymptote values the prediction is consistent for
3.4. RESULTS

Figure 3.3: Inflection S-shaped model fitting to data with different estimators.

Table 3.3: PRE and BPRE for different estimators and models.

<table>
<thead>
<tr>
<th>Asymptote, PRE</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel-Okumoto</td>
<td>-3.0%</td>
<td>-19.2%</td>
<td>-8.8%</td>
</tr>
<tr>
<td>DelayedS</td>
<td>-3.0%</td>
<td>-31.6%</td>
<td>2.9%</td>
</tr>
<tr>
<td>InflectionS</td>
<td>-3.0%</td>
<td>-26.4%</td>
<td>-0.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Asymptote, BPRE</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel-Okumoto</td>
<td>-2.9%</td>
<td>-13.9%</td>
<td>-7.5%</td>
</tr>
<tr>
<td>DelayedS</td>
<td>-2.9%</td>
<td>-19.4%</td>
<td>2.9%</td>
</tr>
<tr>
<td>InflectionS</td>
<td>-2.9%</td>
<td>-17.3%</td>
<td>-0.4%</td>
</tr>
</tbody>
</table>

From TABLE 4.4 and Fig 4.6 we can observe that MSE fit values using NLR are much better compared to values obtained using MLE. This is not surprising given that least square procedure actually minimizes the sum of square of errors between the observed data and used model. On comparing MSE values using MLE obtained in this study to earlier study by Pham and by using equations, we can see that in all but one case MSE values obtained in this study are much smaller than those presented in
earlier study and they are also closer to values obtained using MLE simultaneous equations.

Further the interesting point to note from the comparison is that despite NLR giving very good fit values, it does comparatively worse for the MSE values for the predicted values. Mean square error using MLE are significantly smaller to ones obtained using NLR which confirms that MLE is a better estimator for prediction purposes.

Although as described earlier that SSE (Sum of Squared Errors)/MSE is not a fair comparison parameter between MLE and NLR for the fitted data points, but since MSE for the predicted data is not optimized for both estimators (MLE & NLR), it serves the purpose of comparing between the two estimators on evaluating fit of given model to observed data and goodness-of-fit to predicted data.

### 3.4.3 Which Estimators give better Fit to data and Predicted values.

Another widely used parameter to compare different models and their estimators for their performance is their ability to fit the observed defect/failure data and to the predicted the data. Mean
3.4 RESULTS

Square Error (MSE) is often used to compare the fit of observed and predicted values. MSE is described in section 3 and values obtained for MLE and NLR estimators are provided in TABLE 4.4. The MSE values using MLE and NLR estimation using additional Logistic and Gompertz model (for NLR estimator) is also presented in Fig 4.6.

Table 3.4: Comparing MSE fit and predict values for different estimators and models.

<table>
<thead>
<tr>
<th>MSE fit</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
<th>Using Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel-Okumoto</td>
<td>67.0</td>
<td>20.8</td>
<td>62.7</td>
<td>65.4</td>
</tr>
<tr>
<td>Delayed S-shaped</td>
<td>246.6</td>
<td>89.2</td>
<td>420.4</td>
<td>223.8</td>
</tr>
<tr>
<td>Inflection S-shaped</td>
<td>155.7</td>
<td>42.3</td>
<td>132.1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MSE predict</th>
<th>MLE</th>
<th>NLR</th>
<th>Pham</th>
<th>Using Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goel-Okumoto</td>
<td>42.7</td>
<td>301.6</td>
<td>50.4</td>
<td>1.6</td>
</tr>
<tr>
<td>Delayed S-shaped</td>
<td>12.8</td>
<td>702.0</td>
<td>22.5</td>
<td>40.9</td>
</tr>
<tr>
<td>Inflection S-shaped</td>
<td>9.3</td>
<td>501.6</td>
<td>23.0</td>
<td></td>
</tr>
</tbody>
</table>
3.4.4 Working with un-grouped data.

A further practical consideration that needs to be accounted when working with un-grouped data is as follows: in some cases the time between failures is zero for example in the data set used in this study it occurs at fault numbered 33, 61 and 104, highlighted in table 5. When using MLE estimators with log-likelihood function for NHPP process as given in [17] and using MLE packages such as MaxLik the failures where mean time between failures (MTBF) is zero need to be grouped, else the package can returns NaN errors. And when using the MLE simultaneous equations for GO and Delayed S-shaped model as given in [68], the data used should be un-grouped including the failures with MTBF values equal to zero, else its equivalent to not considering those failures in the analysis which is also not correct.

3.5 Conclusions

In this study using data from literature we have compared between two of the most widely recommended and used methodology for estimating parameters for the purpose of applying SRGMs to defect/failure data. It is noted in the study that while MLE is the recommended estimator with superior statistical properties, its usability and applicability in all situations is questionable. Further MLE is difficult to apply which limits its use in industry, especially due to lack of tools support.

Although external validity of work presented here may be considered low due to use of only single dataset, the study provides important results that point towards different results obtained using different estimation procedures. The study provides useful and practical insights for industry practitioners and early researchers applying reliability modelling to defect/failure data.

We further provide an improvised metric (BPRE) for comparing the predictive accuracy which is symmetric for over and under prediction addressing the problem identified in this study with widely used metric PRE (predicted relative error).

With results in this study suggesting that the fit, predict and predictive accuracy obtained using MLE and NLR estimators may be much different from one estimator to another, more research in this direction is needed to establish these differences in different
3.5. CONCLUSIONS

contexts and thus helping to resolve the dilemma faced by reliability practitioners of which estimator to use and in which conditions a given estimator is better than other. Initial results presented here and properties of MLE and NLR estimators suggest that while NLR is a good estimator for fitting the data to observed failure data, MLE is better estimator for making reliable predictions.

Figure 3.6: Data used in this study, provided in [57] and used in earlier studies including [17], [58].
Chapter 4

Paper C

Evaluating long-term predictive power of standard reliability growth models on automotive systems


Abstract

Software is today an integral part of providing improved functionality and innovative features in the automotive industry. Safety and reliability are important requirements for automotive software and software testing is still the main source of ensuring dependability of the software artefacts. Software Reliability Growth Models (SRGMs) have been long used to assess the reliability of software systems; they are also used for predicting the defect inflow in order to allocate maintenance resources. Although a number of models have been proposed and evaluated, much of the assessment of their predictive ability is studied for short term (e.g. last 10% of data). But in practice (in industry) the usefulness of SRGMs with respect to optimal resource allocation depends heavily on the long term predictive power of SRGMs i.e. much before the project is close to completion. The ability to reasonably predict the expected defect inflow provides important insight that can help project and quality managers to take necessary actions related to testing resource allocation on time to ensure high quality software at the release. In this paper we evaluate the long-term predictive power of commonly used SRGMs on four software projects from the automotive sector. The results indicate that Gompertz and Logistic model performs best among the tested models on all fit criteria’s as well as on predictive power, although these models are not reliable for long-term prediction with partial data.
4.1 Introduction

Today’s modern automobiles contain significant amounts of software; premium segment cars today typically have up to 80 ECUs and software of the order of 100 million lines of code [71]. Software now not only plays a central role in providing and enhancing basic functionality, but also provides most innovations and advanced features. Software is rapidly becoming one of the dominant parts of modern automotive product development. Software systems have been the key to enable functions such as ABS (Anti-lock braking systems), ESC (Electronic Stability Control) - now regarded as standard and essential features. New functions such as Lane Assist, active safety functions, e.g. Volvos city safety [72] and performance enhancement systems within power management, engine-management are also essentially software controlled.

Requirements of automotive software are different from many other types of modern software applications such as personal computers, mobile, web and telecommunications domain. The main difference is that functional safety, reliability, real-time behaviour are of critical importance. While some intermittent failures may be tolerable to some extent in software environment such as personal computing, this is not the case for most applications that have safety critical functionality. Software in automotive applications are highly reliable with failure rates of the order of 1 ppm (parts per million) a year [2] and the development of safety critical applications are expected to comply with state-of-the-art standards like ISO-26262 (Road Vehicles - Functional Safety) [29]. The standard addresses the needs of developing safety critical applications for vehicles by standardizing the safety life-cycle process [73].

Software reliability growth models (SRGMs) have been used for modelling the defect inflow profiles and reliability of software systems. A number of SRGM models have been proposed and evaluated on various industrial and open source software projects. But most of the assessment of predictive power of SRGMs has been limited to short term predictions, typically for predicting the last 10% of defects in a project. Although evaluating models on their fit to data ex-post and predictive power over last few data points are interesting questions from the academic viewpoint, its practical use is limited. For industrial software development, the use of SRGMs as tools for predicting/forecasting defect inflow demands reasonably accurate prediction on long-term i.e. when the project is not
close to completion but when such information can be used to take practical decisions, for example about resource allocations. Being able to predict precisely the expected defect inflow when project is halfway (50%) through gives good time for engineers/managers to make active and positive interventions. In this paper we focus on distinct and practical research questions as described below:

(i) Which SRGMs fit best to the defect data from automotive software projects?

(ii) Which SRGMs have the best long term predictive power?

(iii) Which models growth rates are consistent between projects over time?

Using four software projects from the automotive industry, we essentially ask which models can be applied to defects data without the need to modify the data in the first research question. With the second research question we evaluate which of these models gives reliable prediction of total number of defects using partial data i.e. early in the project, when it’s 50% and 70% complete. Another important aspect for successful applying SRGMs to predict defect inflow precisely is the possibility to use the growth rate based on earlier experiences (growth rate from earlier projects), in this paper we also assess this aspect through the third research question, by comparing the growth rate between these projects which gives further insight into the usability of different models.

Our results show that on average the Logistic and Gompertz model fits better than others to the defect data. They also have superior long term predictive power compared to other models, but cannot be applied reliably in all cases with partial data. It is also observed that using growth rates from previous projects significantly improves the predictive power of every reliability model used in this study. The remaining of paper is organised as follows: section 2 describes the background of software in automotive domain and software reliability growth models. Current work is placed in perspective of earlier studies in section 3 and the research methodology in described in section 4. Results from the study are presented in section 5 and 6, followed by discussion and conclusion in section 7.
4.2 Background

4.2.1 Automotive Software

Software in automotive domain is highly diverse and complex. A typical modern car today has more than 800 functions realized by software distributed over large number of ECUs connected by number of system buses [74]. New features being developed demand further interaction between what used to be standalone applications and connectivity with external sources/devices. It is claimed that about 90% of innovations today in automotive sector are driven by electronics and software and ECUs related development costs contribute to about 50-70% of overall vehicle development costs [74]. With sharply increasing complexity of software systems, demands for high reliability and cost constraints - calls for high control over the process of software development activities within the industry. And as per standard notion that testing of software constitutes about 50% of the development cost/resources, the need to carefully monitoring and optimizing the process is very apparent. Recognizing the increasing complexity of automotive software functionality and integration issues in E/E architecture level, standards such as Automotive Open System Architecture (AUTOSAR) [75] is being developed. The goals of AUTOSAR are multiple: It attempts to define an open architecture; standardization the basic software functionality of automotive ECUs to support the development of highly dependable systems. It also attempts to make the transferability of software possible and help OEMs and tier-1 suppliers to keep their costs in check by promoting reuse. Embedded software is currently driving the major innovations in the automotive sector. Broy [2] Identifies following areas where software is the main source of innovation:-

- Crash prevention, crash safety;
- Advanced energy management;
- Advanced driver assistance;
- Adaptable man-machine interface;
- Programmable car;
- Personalization and individualization; and
• Interconnected car networking.

Given that most functionality within automobiles relates directly or indirectly to driver assistance, engine performance, passive or active safety, it is not surprising to note that there exists stringent safety requirements for development of these functionalities. Functional safety standard ISO-26262 provides a framework that has to be applied from safety perspective. A brief overview of safety case argumentation with respect to ISO-26262 can be found in [76].

4.2.2 Software Reliability Growth Model

IEEE standard 1633 [31] (Recommended practice on software reliability) defines software reliability (SR) as:

(A) The probability that software will not cause the failure of a system for a specified time under specified conditions. (B) The ability of a program to perform a required function under stated conditions for a stated period of time.

The standard refers to Software Reliability Model (SRM) as mathematical expression used to specify the software failure process in a general form as a function of factors such as fault introduction, fault removal and the operational environment. IEEE standard 1633 also specifies the recommended procedure for software reliability assessment & prediction and provides overview of basic concepts and commonly applied SRMs. Software reliability can be assessed and predicted using the white box approach which takes into account the characteristics of the source code. Factors such as software architecture, length, complexity, logical complexity, operands, control loops etc. are used to assess and predict the software reliability.

Software reliability growth models SRGMs, on the other hand are essentially black-box reliability models as they only use failure data for reliability modelling without giving consideration to the source code characteristics. A number of SRGMs have been proposed over last several decades and also used for practical applications. According to Lyu [44]; to assess and predict the reliability of any software systems, its failure data needs proper measurement and collection over its development and deployment/operation. The underlying software failure process also needs to be understood for reliable prediction. The failure data acts as input to SRGMs
and failure process understanding for the selection of appropriate reliability model for making accurate mapping and prediction.

SRGMs are usually based on data from the formal testing phase which is focused on customer testing. The idea is that defect arrival rate/pattern in such testing is a good indicators of software reliability (post development testing) [23]. Such testing after the development is complete, the failure/defects are detected and fixed which makes the software artefact more stable and reliable thus giving the characteristic name reliability growth model to mathematical expressions used for describing such process. The SRGMs can be classified based on the model assumptions into Concave and S-shaped. Concave models are essentially exponential models which assume that failure intensity decreases exponentially after the initial peak and thus their cumulative distribution function (CDF) is concave shaped. While S-shaped models underline the assumption that testers are initially unfamiliar with the product thus the testing is inefficient at the beginning, but as testing progresses the testers skill improves which help increasing the defect detection (failure intensity) rate which then reaches a peak before it starts to decrease. This gives a characteristic S-shape to the CDF with a slow start.

Application of SRGMs can be primarily used for two purposes, the first is to assess the release readiness and secondly to make optimal test resource allocation. In case of using SRGMs for optimal resource allocations such as amount of testing time, test case allocation or testing resource allocation, SRGMs are applied during the testing process on the partial defect inflow data. Then the predicted/estimated defect inflow information is used to allocate the testing resources optimally such that the product is ready for release by the release date. Given the nature of applications of SRGMs it is apparent that models that provide superior fit to full defect data are better suited for applying SRGMs for release readiness, while models that have better long-term predictive power are more useful for resource allocation applications.

4.3 Related Work

Since the introduction of reliability engineering approach for studying and predicting reliability of software products a number of SRGMs have been proposed [13], [14], [49], [9], [77]; much of these
4.3. RELATED WORK

models differ in their assumptions of the underlying defect discovery and defect removal process [57]. SRGMs have been built for calendar time and execution time, execution time models have been shown to be superior to calendar time models in many cases [78], [79], but given the fact that calendar time is much more meaningful for engineers and managers most models have been developed on the calendar time component. One of the earliest models in this field was proposed by Musa [80] which was an execution time model, it has since been applied with good results. Goel and Okumoto proposed a Non-Homogeneous Poisson process (NHPP) based model in 1979, now referred as GO model [13], a number of SRGMs since then have been models on NHPP with improved assumptions.

As the number of SRGMs have been proposed and used, there have also been studies that have reviewed and compared among these models. Goel [15] in 1985 reviewed number of analytical SRGMs providing critical analysis and their underlying assumptions, a step-by-step process was also proposed for applying these models for the analysis of failure data using examples. Hoang Pham presented a review and performance comparison of commonly used NHPP based SRGMs in [17] and found that NHPP models that also take into consideration the environmental factors are more precise although needs more data and effort for their modelling. Cai et al. [81] take a highly critical view on probabilistic SRGMs basing their argument that each software product and process is unique and thus Fuzzy SRGMs must be developed and used for evaluating reliability in case of software development.

Another significant challenge to use SRGMs in industrial settings is the uncertainty about the predictive power of different SRGMs; most reliability models have been proposed and evaluated on failure data ex-post, which is applicable for release readiness evaluation, but not appropriate for the application of SRGMs for predicting and optimizing the testing effort during the testing phase. Most studies that propose, evaluate or compare between different models also evaluate the predictive power of these models on short-term (typically using last 10% data, for example [82], [17]). It has been indicated in our earlier study [12] that evaluation of SRGMs on industry specific domains is not well researched area which is specifically true for the automotive sector where software is increasingly playing the dominant role in providing functionality
and new features. There are some studies with the focus on applying or evaluating SRGMs on industrial applications, Woods [19] applied and evaluated eight SRGMs on the data from industry and concluded that defect predictions using the cumulative defect inflow data correlated well with that of after release data. In [45], the authors evaluated different SRGMs on their predictive power during the development of a large software project within telecom domain and proposed new model based on moving averages using historic data. In our earlier study [12] number of SRGMs are evaluated on defect data from an automotive software project and found Logistic, Inflection and Gompertz model as good performers. Ullah et al. [18] studied the fitting and prediction capability of eight SRGMs on fifty data sets from industrial projects and open source software projects, they found Musa-Okumoto and Inflection models doing best on industrial datasets, while Gomperts and Inflection on the open source projects.

This study follows much in structure to the Ullah et al. [18] study with important distinctions. Firstly in this paper we focus on questions important with respect to application of SRGMs for optimal resource allocation, i.e. we evaluate the models fit to partial data and assess their predictive power on long-term basis. Secondly we use defect inflow data from four different projects exclusively from the automotive domain, thus our purpose is two folds; firstly by evaluating seven common SRGMs on multiple projects from the automotive industry we attempt to answer which models fit the data best for this specific industrial domain. Secondly we evaluate these SRGMs for their predictive capacity on long-term basis which provides important insight into applying SRGMs for optimal resource allocation independent of the industrial domain. We also analyse an important practical issue about portability of growth rate between the projects which is critical when applying SRGMs during an ongoing project with the objective of good long term prediction and using the predictions for resource allocation decisions.

4.4 Models and Data

In this study we use seven SRGMs; these are selected based on their use in literature for the evaluation of reliability of software systems. Table 4.1 summarizes them with their mean value func-
4.4. MODELS AND DATA

The data used for this research is obtained from our industrial partner, Volvo Car Corporation (VCC) from the automotive sector. The data is from four different projects which are similar in scope, but differ widely in their attributes such as number of reported defects, time scale, vehicle platform/car models they applied to and others. The projects come from the E/E (Electrical and Electronics) integration department within the VCC which deals with the integration of various software functionalities and responsible for the final assessment of full EE hardware and software systems. The projects used in this study have been completed during the last decade. All the projects consist of different modules developed by different teams and tested within the development team (unit testing), while further integration and acceptance testing is done by dedicated teams in the integration department. All defects detected during all testing phases are reported in the central bug database used by the company which was also the source of data compilation for this study. Since the defects data used in this study span for most part of development and testing phase, the time domain used for defect data analysis is calendar time and not CPU/test execution time which is more appropriate for analysis using data only from formal testing phase. Although due to confidentiality reasons the data cannot be shared, Fig 4.1 provides the cumulative defect profiles for these four projects, helpful for understanding the nature of data and visualize the variations across the projects. (The Y-axis shows the normalized total number of defects reported, while time axis is trimmed to show only the partial data in the figure)

Fitting of SRGMs to data for parameter prediction is usually
done using maximum likelihood procedure or Linear/Non-Linear Regression (NLR). Similar to the study by Ullah et al. [18] we use NLR to fit the given SRGM onto the data. In order to have better comparability between full and partial data, the starting values given were same for full and partial data sets as well as very similar across the models. Non-Linear Regression routine within the commercially available statistical software package, IBM SPSS is used and iterations are run until the residual errors between the successive iterations is less than 1.0*E-08. The $\beta$ parameter value for the Inflection model is assumed to take a value of 1.2 based on the underlying process and as per parameter estimation procedure given in [66]. The data set used in this study includes defect data comes mainly from the development and testing phase, but this is

Figure 4.1: Cumulative defect inflow profile for projects under study.
entirely pre-release data (i.e. excluding the post-release defects).

To evaluate the different SRGMs fit and predictive power on long-term basis we apply these models to full data (100%) and partial data-sets which consisted of 90% of data, 70% of data and 50% of data points. The applied data was used to fit the model and estimate the parameters, while the rest of data was used to evaluate the fit measure for predicted values and evaluate the models prediction capabilities. Below we summarize briefly the different models convergence to the partial data sets.

It can be noted from table 4.2 that most models were able to converge to full as well as partial data (shown by symbol Y), only the Logistic model did not converge to the actual data for the Proj-D using 90% of data points, Proj-B with 70% data points and with all projects with 50% data points (marked by “N”). Three cases in the predicted section are also highlighted in red in the table and marked *, these are the cases where given model do converge well to the partial data (giving high $R^2$ fit value) thus giving good parameter estimate values, but on evaluating the predict criteria provides negative values of $R^2$ fit to overall data. Negative value of $R^2$ can be simply interpreted as the case when the mean of data provides a better fit (overall) than the fitted and predicted values, one of these cases is shown in Figure 4.2 below. We delete these values when taking the average values for the purpose of comparison between the SRGMs. These three cases also highlight the limitation of using NLR methodology to fit reliability models to partial data, i.e. using NLR for an on-going project. NLR is based on least square which minimizes the sum of squared errors between actual observations and model predicted values. Method of least square applied to partial data thus may favour model parameters that give good fit to the observed partial data but may not be optimal from prediction perspective.

4.5 Results: Fit and Long Term Prediction

4.5.1 Which SRGMs fit best?

Mean Square Error (MSE): is the average of squared error, where error is defined as the difference between actual and predicted value. MSE measures the variance for an unbiased estima-
Table 4.2: Models convergence to partial data.

<table>
<thead>
<tr>
<th>Model</th>
<th>100% data</th>
<th>90% data</th>
<th>70% data</th>
<th>50% data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proj-A</td>
<td>Proj-B</td>
<td>Proj-C</td>
<td>Proj-D</td>
</tr>
<tr>
<td>Musa-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Gompertz</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Logistic</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>100% data</th>
<th>90% data</th>
<th>70% data</th>
<th>50% data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proj-A</td>
<td>Proj-B</td>
<td>Proj-C</td>
<td>Proj-D</td>
</tr>
<tr>
<td>Musa-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Logistic</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>100% data</th>
<th>90% data</th>
<th>70% data</th>
<th>50% data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proj-A</td>
<td>Proj-B</td>
<td>Proj-C</td>
<td>Proj-D</td>
</tr>
<tr>
<td>Musa-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Gompertz</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Logistic</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

The MSE value can be used as an ordinal scale to compare between different estimators to assess which one is better than another, lower the MSE, the better the model/estimator. Mathematically it is defined as:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2
\]  

(4.1)

**Coefficient of determination/ R-Square \((R^2)\):** measures the goodness of fit. It is widely used parameter to measure how well the modelled/predicted values match to the actual data. \(R^2\) takes value between 0 and 1, the closer the value to 1, the better the estimator. Since its not appropriate to measure \(R^2\) for only the
4.5. RESULTS: FIT AND LONG TERM PREDICTION

Figure 4.2: Case with negative R², Gompertz model, project D using 70% data.

predicted value, we compare $R^2$ value for fitted data and $R^2$ value for full data to make the comparison of prediction goodness-to-fit. Mathematically $R^2$ is defined as:

$$R^2 = 1 - \frac{\sum_{1}^{n} (Y_i - \hat{Y}_i)^2}{\sum_{1}^{n} (Y_i - \bar{Y}_i)^2}$$  \hspace{1cm} (4.2)

Theil’s Statistics (TS): measures the average deviation percentage over all data points. TS value closer to 0 means a better fit and thus desired, it can be expressed as:

$$TS = \sqrt{\frac{\sum_{1}^{n} (Y_i - \hat{Y}_i)^2}{\sum_{1}^{n} (Y_i)^2}} * 100\%$$  \hspace{1cm} (4.3)

*In (1) to (3), $Y_i$ = Actual value; $\hat{Y}_i$ = predicted value; $\bar{Y}_i$ = mean value and $n$ is the number of data points used.

The average of above three measures of goodness-of-fit to fitted data and the predicted data across all four projects for each of SRGM evaluated in this study is represented in figure 4.3 to figure 4.5. In fig 4.3 - 4.5, line plots represent goodness of fit for the observed data while the bars show the goodness of fit for the predicted values using 90%, 70% and 50% data. The fit and predict measures are scaled on primary and secondary vertical axis respectively. It can be observed from all three measures that fit is superior for the fit part (observed/used data) which is not surprising when using
NLR which minimizes the square of errors. Secondly one could also observe that models where less data is used (e.g., 50% data compared to 70%, 90% or 100% data) the fit to observed data is better than when using more data simply because its relatively easy to configure parameters of any given model to fit less data points, while fitting the same model to more data points leads to more errors and thus comparatively inferior fit. At the same time using more data gives parameter estimates that are more precise for prediction making the predict fit measure better as more data is used (represented by the bars). Based on the three model-fit metrics described above (MSE, $R^2$ and TS), we rank each model.
Figure 4.5: Average TS for studied SRGMs over full & partial data.

According to their better fit characteristics, summarized in Table 4.3. The overall rank for the model is calculated based on the sum of ranks for each fit criteria including the goodness-to-fit metrics for the data used to fit the model (fit) and data predicted using the model (predict).

Table 4.3: Models goodness-of-fit rank matrix.

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE (fit)</th>
<th>$R^2$ (fit)</th>
<th>TS (fit)</th>
<th>MSE (predict)</th>
<th>$R^2$ (predict)</th>
<th>TS (predict)</th>
<th>Sum ranks</th>
<th>Overall rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Gompertz</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Logistic</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

From the Table 4.3, it is clear that the Logistic model is best among all the models in terms of fitting the full as well as partial data, which is followed by Gompertz, Rayleigh and Delayed-S models. It is also interesting to note that MO model fit and predict better than only the Inflection-S model in our data, while it was one of the best fitted models to industrial datasets in the study by Ullah et al. [18].
4.5.2 Which SRGMs have the best long term predictive power?

To evaluate which model has the best long-term predictive power, we use the Predicted Relative Error (PRE), which is a measure of the asymptote correctness. PRE is defined as the ratio of error in predicted asymptote over the predicted number of defects. \( \text{PRE} = \frac{\text{Predicted} - \text{Actual No of defects}}{\text{Predicted}} \); PRE is specified as one of approach to measure model predictive validity in the standard IEEE 1633 [31] and used in the Ullah et al. [18]. In [83], it is observed that PRE does not give consistent results for positive and negative deviations, thus we use balanced PRE or BPRE [83], which is defined as:

\[
\text{BPRE} = \frac{\text{Predicted} - \text{Actual}}{\eta \times \text{Predicted} + (1 - \eta)(2 \times \text{Actual} - \text{Predicted})},
\]

where \( \eta = \begin{cases} 1 & \text{if Predicted} > \text{Actual} \\ 0 & \text{if Predicted} < \text{Actual} \end{cases} \) (4.4)

BPRE for the four projects in this study are reported in Table 4.4, PRE (and thus BPRE) values of +/-10% are usually considered good, certain models fit well, but not all models give best prediction for all projects data. It can be observed from the table that Logistic, Gompertz and Rayleigh models fit well in most cases; taking the average of BPRE ignoring the sign of predictive relative error we get \( \text{Avg}^+ \). Figure 4.6 shows the \( \text{Avg}^+ \) value for each model for full (100%) and partial data 90%, 70% and 50%. It is further noted from the figure 4.6 and table 4.4 that on average for all four projects, as expected BPRE is better when full data is used, followed by 90% and 70% data than using 50% of the data points.

With respect to different SRGMs, the concave models, MO and GO perform poorly with respect to the prediction accuracy for our projects data. S-shaped models such as Inflection and Delayed perform better, while the top three models with best prediction performance on average across all projects are Logistic, Gompertz and Rayleigh models respectively. The results are consistent with

\footnote{In order to evaluate the mean of predictive power for a given model, normal average will be affected by the sign convention for over and under predictions. Since we are only interested in the precision accuracy, \( \text{AVG}^+ \) only considers the magnitude of BPRE for calculating the mean value, ignoring the sign.}
our earlier study [12], but different from those reported in Ullah et al. study; further indicating towards the need for industry/domain specific evaluation of SRGMs.

Table 4.4: Models BPRE values for full and partial datasets.

<table>
<thead>
<tr>
<th>BPRE-100%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>40.8%</td>
<td>99.9%</td>
<td>99.6%</td>
<td>99.9%</td>
<td>85.1%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>50.4%</td>
<td>99.9%</td>
<td>99.9%</td>
<td>99.9%</td>
<td>87.4%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>22.6%</td>
<td>89.1%</td>
<td>86.5%</td>
<td>89.5%</td>
<td>71.9%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>13.3%</td>
<td>89.1%</td>
<td>35.5%</td>
<td>72.6%</td>
<td>52.6%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>1.4%</td>
<td>62.7%</td>
<td>11.5%</td>
<td>38.0%</td>
<td>28.4%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>2.0%</td>
<td>68.4%</td>
<td>15.1%</td>
<td>10.8%</td>
<td>24.1%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-1.6%</td>
<td>31.1%</td>
<td>4.5%</td>
<td>4.2%</td>
<td>10.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-90%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>70.4%</td>
<td>99.8%</td>
<td>99.8%</td>
<td>99.9%</td>
<td>92.5%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>71.0%</td>
<td>99.8%</td>
<td>99.8%</td>
<td>99.9%</td>
<td>92.6%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>33.4%</td>
<td>87.2%</td>
<td>89.2%</td>
<td>89.2%</td>
<td>74.8%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>17.6%</td>
<td>89.7%</td>
<td>47.6%</td>
<td>96.9%</td>
<td>62.9%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>2.0%</td>
<td>63.1%</td>
<td>17.6%</td>
<td>73.2%</td>
<td>28.7%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>2.4%</td>
<td>78.4%</td>
<td>23.1%</td>
<td>10.8%</td>
<td>10.6%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-2.4%</td>
<td>33.4%</td>
<td>6.7%</td>
<td>10.6%</td>
<td>10.6%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-70%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>99.4%</td>
<td>99.7%</td>
<td>99.8%</td>
<td>99.9%</td>
<td>99.7%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>99.4%</td>
<td>99.7%</td>
<td>99.8%</td>
<td>99.9%</td>
<td>99.7%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>87.2%</td>
<td>80.0%</td>
<td>86.9%</td>
<td>84.6%</td>
<td>84.7%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>36.6%</td>
<td>50.1%</td>
<td>65.2%</td>
<td>100.0%</td>
<td>63.0%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>7.4%</td>
<td>-7.6%</td>
<td>24.8%</td>
<td>100.0%</td>
<td>34.9%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>5.8%</td>
<td>42.7%</td>
<td>37.2%</td>
<td>24.3%</td>
<td>27.5%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-4.6%</td>
<td>3.5%</td>
<td>99.5%</td>
<td>26.9%</td>
<td>26.9%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-50%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>99.6%</td>
<td>99.5%</td>
<td>99.7%</td>
<td>99.5%</td>
<td>99.6%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>99.7%</td>
<td>99.5%</td>
<td>99.7%</td>
<td>99.4%</td>
<td>99.6%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>85.4%</td>
<td>66.0%</td>
<td>77.9%</td>
<td>65.4%</td>
<td>73.7%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>79.2%</td>
<td>-25.6%</td>
<td>44.3%</td>
<td>-28.8%</td>
<td>44.5%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>36.9%</td>
<td>-37.9%</td>
<td>-16.7%</td>
<td>-38.4%</td>
<td>32.5%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>22.1%</td>
<td>-22.1%</td>
<td>-16.5%</td>
<td>100.0%</td>
<td>40.2%</td>
</tr>
</tbody>
</table>

Figure 4.6: Average BPRE for studied SRGMs for partial data.
A further interesting observation is made from figure 4.6 and table 4.4: it can be noted from the figure that for some models, the prediction accuracy on average with partial data is better than with average prediction accuracy when full or 90% data is used for the same model. This observation is clear from Delayed S-model where average accuracy using 50% data is significantly better than even using full data. The reason for such anomalous observation is explainable based on the defect inflow data and characteristics of models. As it can be seen the inconsistency is significant mainly for Delayed and Inflection S-shaped models - this occurs for a reason; it has been demonstrated in earlier studies [33] that defect inflow profile in automotive software projects often is characterized by presence of late defects. Late defects are not a standard characteristic of normal S-shaped curve which is based on inverted bell shaped inflow profile. When full data is used for reliability modelling, the late defects are also accounted in model parameter estimations pushing the asymptote prediction higher, while when partial data (such as 50% or 70%) is used the late defects are not accounted to the same extent resulting in prediction with higher accuracy for S-shaped models.

Further one can note from table 4.4 that there can be variations and inconsistencies that can be observed with respect to predictive power of given models on different projects. Most of these inconsistencies can be explained based on different nature (defect inflow profile) and characteristic of least square fitting methodology. Let us take a few examples: we can observe that MO model gives very high BPRE values for all but Project-A with 100% data, going back to the cumulative defect profile for all projects (in fig 4.1) one could see that project-A cumulative defect profile is the most concave and thus the result does make practical sense and for same reason the other concave model, GO model also gives good BPRE estimates for project-A. Another case is poor BPRE prediction by Gompertz model with 50% data (while it gave good estimates with 100%, 90% & 70% data), the sharp decline in prediction accuracy in this case can also be understood by looking at the cumulative defect profile for Project-C. It is observed that project-C has long phase in early part of project with slower than expected cumulative defect growth thus if using only partial data from early part of project, it may not provide enough data points for S-shaped models to predict the asymptote precisely. These observations highlight a high need for
exhaustive defect inflow analysis using exploratory techniques as the first step before selecting and applying any SRGM to the data, we raise the lack of exploratory data analysis in SRGM literature in our next study and provide further evidence that such exploration can be very useful in successfully applying reliability engineering in practice.

4.6 Results: Using Past Experience for Reliability Growth Modelling

In this section we evaluate the third research question: *Which models growth rates are consistent between projects over time?* When applying SRGMs for making better predictions to be used for optimal resource allocation during the project, an important question is if one can use the past experience for making the predictions for future. More precisely how far growth rate from previous projects can be used for making defect inflow prediction for current projects and how good these estimates can be expected. This possibility to use previous projects data for prediction at early project phases constitutes critical information for applying SRGMs for resource allocations in industry. To answer this question, we take the growth rate parameter prediction for Proj-A (using full data points) for each SRGM and use it to make asymptote predictions for rest of

![Figure 4.7: Average BPRE for studied SRGMs with growth rate from Project-A (100% data).](image-url)
the projects (B, C & D). The growth rates are also used to predict
the total number of defects for each project using partial datasets
(90%, 70% and 50% data points). To measure the correctness of
asymptote predictions BPRE is calculated and listed in Table 4.5.
The Average+ value for BPRE of each model is presented in figure
5.5, which can be compared to figure 4.6.

Table 4.5: BPRE for studied models using growth rate from Proj-
A.

<table>
<thead>
<tr>
<th>BPRE-100%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>40.8%</td>
<td>11.3%</td>
<td>22.2%</td>
<td>33.9%</td>
<td>27.1%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>50.4%</td>
<td>25.6%</td>
<td>35.3%</td>
<td>44.3%</td>
<td>38.9%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>22.6%</td>
<td>-12.5%</td>
<td>0.4%</td>
<td>12.1%</td>
<td>11.9%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>13.3%</td>
<td>-19.1%</td>
<td>-8.7%</td>
<td>1.3%</td>
<td>10.6%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>1.4%</td>
<td>-24.9%</td>
<td>-16.2%</td>
<td>-10.5%</td>
<td>13.3%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>2.0%</td>
<td>-7.8%</td>
<td>-4.9%</td>
<td>12.7%</td>
<td>6.9%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-1.6%</td>
<td>-9.4%</td>
<td>-6.6%</td>
<td>6.9%</td>
<td>6.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-90%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>42.5%</td>
<td>2.1%</td>
<td>21.3%</td>
<td>31.8%</td>
<td>24.4%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>51.6%</td>
<td>17.5%</td>
<td>34.1%</td>
<td>42.3%</td>
<td>36.4%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>23.8%</td>
<td>-18.8%</td>
<td>-2.5%</td>
<td>8.5%</td>
<td>13.4%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>14.3%</td>
<td>-24.1%</td>
<td>-11.4%</td>
<td>-2.4%</td>
<td>13.0%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>1.6%</td>
<td>-28.9%</td>
<td>-18.8%</td>
<td>-13.3%</td>
<td>15.7%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>2.2%</td>
<td>-17.8%</td>
<td>-7.4%</td>
<td>16.8%</td>
<td>11.0%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-2.1%</td>
<td>-18.8%</td>
<td>-9.2%</td>
<td>10.5%</td>
<td>10.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-70%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>43.9%</td>
<td>-16.0%</td>
<td>11.5%</td>
<td>10.8%</td>
<td>20.6%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>52.5%</td>
<td>-9.3%</td>
<td>25.3%</td>
<td>24.2%</td>
<td>26.6%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>24.6%</td>
<td>-28.5%</td>
<td>-13.4%</td>
<td>-14.3%</td>
<td>20.2%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>15.7%</td>
<td>-31.7%</td>
<td>-19.9%</td>
<td>-19.5%</td>
<td>21.7%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>2.5%</td>
<td>-34.9%</td>
<td>-25.9%</td>
<td>-25.1%</td>
<td>22.1%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>3.1%</td>
<td>-32.4%</td>
<td>-18.8%</td>
<td>21.2%</td>
<td>18.9%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-2.9%</td>
<td>-33.0%</td>
<td>-20.4%</td>
<td>28.9%</td>
<td>21.3%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BPRE-50%</th>
<th>Proj-A</th>
<th>Proj-B</th>
<th>Proj-C</th>
<th>Proj-D</th>
<th>Avg+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musa-Okumoto</td>
<td>37.3%</td>
<td>-26.7%</td>
<td>-8.9%</td>
<td>-27.3%</td>
<td>25.1%</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>46.7%</td>
<td>-20.1%</td>
<td>5.8%</td>
<td>-21.0%</td>
<td>23.4%</td>
</tr>
<tr>
<td>Inflection-S</td>
<td>16.1%</td>
<td>-34.5%</td>
<td>-24.8%</td>
<td>-34.7%</td>
<td>27.5%</td>
</tr>
<tr>
<td>Delayed-S</td>
<td>10.6%</td>
<td>-36.0%</td>
<td>-28.1%</td>
<td>-35.8%</td>
<td>27.6%</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>0.4%</td>
<td>-37.9%</td>
<td>-31.9%</td>
<td>-37.3%</td>
<td>26.9%</td>
</tr>
<tr>
<td>Gompertz</td>
<td>1.4%</td>
<td>-39.7%</td>
<td>-30.7%</td>
<td>-40.9%</td>
<td>28.2%</td>
</tr>
<tr>
<td>Logistic</td>
<td>-0.9%</td>
<td>-40.1%</td>
<td>-31.9%</td>
<td>-40.4%</td>
<td>28.3%</td>
</tr>
</tbody>
</table>

Comparing the two figures and Table 4.4 & 4.5, following in-
teresting observations can be made:

- Using growth rates from earlier projects improves significantly
the long-term prediction accuracy of all the models. Com-
pared to parameter estimation using each projects full & par-
tial data alone, the models asymptote predictions obtained
using growth rates from project-A gives much better and con-
sistent results for all the models. With an exception of GO
model, all models have BPRE value within 30% range for all
partial datasets. Except of concave models (MO & GO) all models BPRE values are very close to +/-10% mark for full and 90% data, which is considered good prediction range;

- Also, irrespective of SRGM used, using growth rate from earlier project in later projects (using 70% data and 50% data) gives BPRE values close to +/-20% and +/-30% range respectively, which is again much better than using these models with only the current projects data in isolation for making asymptote predictions;

- Although Logistic and Gompertz models predictive power is better than rest of models towards end of project (using 100% and 90% data), Rayleigh, Delayed and Inflection S-shaped model outperform Logistic and Gompertz for long term predictive power using 70% and 50% data points;

Most importantly the observation that all models perform better across the project timeline using growth rate from earlier projects is significant result with high practical importance for applying SRGMs for making resource allocation decisions during the project.

4.6.1 Threats to Validity

It is important to be aware of potential threats to validity when conducting an empirical study; we evaluate our study on the basic validity tests used to evaluate threats to validity in empirical research in software engineering [84], [85]:

Construct validity refers to construction of right dependent and independent variables and testing the right hypothesis. SRGMs and their application to defect inflow prediction is well established, we construct specific research questions of interest and although we do not test for hypothesis, we use widely used and recommended metrics to answer these questions, threat of construct validity is thus kept under check. The statistical methods used in our study are also well established which minimizes the risk of method bias.

Having used an improvised metric BPRE for assessing the predictive power we also address the threat of conclusion validity. BPRE defined in details in [83] is used to ensure that there is consistency (symmetry) between the metric value for over and under predictions. Also using only the magnitudes while taking average across projects for given models (Avg+) address the threat that a
model with wide over and under predictions is averaged out and presented as better predictor than another with consistent lower under or over predictions. Thus BPRE and Avg+ in fact help us increase the conclusion validity of the given study.

Internal validity protects against external factors not accounted in the study affecting the dependent variable. In this study we evaluate the goodness of fit and predictive power of different SRGMs on the defect inflow data. While there are many factors that can influence the defect inflow process; the measures of fit and predictive power to a large extent depends on the data used and model applied. To further minimize the threat from external factors, the starting values provided for NLR for each model were same when applied to full and partial data; they were also very similar across the different models. One factor that can be considered a reasonable threat is the use of NLR procedure to fit the data, while the widely recommended method for parameter estimation is the maximum likelihood estimator (MLE). Given that NLR is easy to adopt in industrial setting and the fact that the main objective of this study is to compare between the performance of different models, the effect of estimator is not a serious threat, but it is recognized that if the purpose is to evaluate the precision of each model individually than difference between these estimators should be accounted.

External validity deals with the generalizability of results in settings outside of the given study. In this study we explicitly used projects from one industrial domain (automotive); we evaluate SRGMs on automotive domain which is less researched area. With respect to research questions 2 and 3 which have wider applicability in using SRGMs for predicting total number of defects during an on-going project for optimal resource allocation; the threat is reduced by the fact that we use four different projects with very different characteristics. The selection of SRGMs and fitting procedure was based in literature and evaluation criterias used were also from recommended practices and standards. Although this study provides very useful insight into the long-term predictive power of different SRGMs and using growth rate from earlier projects, more research is needed and recommended to validate the observations further in different contexts, using data from other industrial domains and also from open source software projects.
4.7 Conclusion

A wide range of software reliability growth models (SRGMs) have been reported and evaluated in the literature. It was also noted that although some studies exist that focus on the application and evaluation of these models in industrial context, evaluation of SRGMs in specific industrial domains is less researched area and significant gap exist within the evaluation of SRGMs applicability in the automotive industry.

In this paper we evaluated seven widely used SRGMs onto four software projects from the automotive domain. We set out to evaluate:

- Which SRGMs fit best to the defect data from automotive software projects?

With respect to goodness-to-fit criteria to the data used for parameter estimation (fit) and predicted data (predict), it is observed and reported that Logistic model outperformed all other models, although since it could not converge using 50% data points for all projects and a couple of other cases, the reliability that it can be applied in all conditions and early phases is relatively low. Gompertz model was next with high goodness-of-fit for both fit and predict values and also high reliability of applicability with partial datasets. Rayleigh and Delayed S-shaped models also fitted well to data and predicted values and surprisingly Inflection model was outperformed by MO & GO (concave models) in the data-fit criteria which are quite different from results obtained by Ullah et al. study of SRGMs on industrial datasets. This highlights that not all models perform best under all cases and certain models are better suited for different conditions and processes. Our results were consistent with our earlier case study on a defect dataset from automotive domain from different department and development phase [12].

Next we investigated important questions in reliability modelling with wider context:

- Which SRGMs have the best long-term predictive power?

Logistic, Gompertz and Rayleigh models were the top three models respectively to predict long-term defect inflow with respect to accuracy. Comparing different SRGMs on Balanced Predicted Relative
Error (BPRE) provided evidence contrary to some earlier studies. Concave models (MO & GO) were found to be highly inappropriate for making long or even short term predictions of asymptote in our dataset from automotive domain. S-shaped models (Inflection, Delayed) followed next with better prediction accuracy.

Another important question evaluated was regarding usability of growth rates from earlier projects for making long term asymptote predictions was also studied.

- Which models growth rates are consistent between projects over time?

Using growth rate form earlier project provided much better and consistent results for asymptote prediction on full as well as partial datasets than trying to model these datasets in isolation. Despite the fact that these projects have many differences between them (for example with respect to total number of defects), the asymptote predictions obtained were much better when using growth rate from earlier projects compared to using only the current projects defect data for reliability modelling. It was also observed that using growth rate from earlier projects, although the Logistic and Gompertz model predictive power was superior to other models for 100% and 90% data; with respect to long-term predictive power using 70% and 50% data Delayed and Inflection S-shaped models performed slightly better than even the Logistic and Gompertz models.

The results obtained here are significantly important as they fill an important gap in the reliability models evaluation in the domain specific industrial applications. It also provides critical information to the engineers and managers in automotive industry who currently use or wish to use software reliability modelling for making resource allocation decisions. Further studies in this direction will ensure that reliability engineering is adopted more widely in the industry and provides useful information to industrial practitioners.
Chapter 5

Paper D

Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model Based Development


Published at 8th International Joint Conference on Software Technologies - ICSOFT-EA, Reykjavík, Iceland, July 2013
Abstract

The rapid growth of software intensive active safety functions in modern cars resulted in adoption of new safety development standards like ISO 26262 by the automotive industry. Hazard analysis, safety assessment and adequate verification and validation methods for software and car electronics require effort but in the long run save lives. We argue that in the face of complex software development set-up with distributed functionality, Model-Based Development (MBD) and safety-criticality of software embedded in modern cars, there is a need for evolving existing methods of MBD and complementing them with methods already used in the development of other systems (Fault Injection and Mutation Testing). Our position is that significant effectiveness and efficiency improvements can be made by applying fault injection techniques combined with mutation testing approach for verification and validation of automotive software at the model level. The improvements include such aspects as identification of safety related defects early in the development process thus providing enough time to remove the defects. The argument is based on our industrial case studies, the studies of ISO 26262 standard and academic experiments with new verification and validation methods applied to models.
5.1 Introduction

Nowadays, a typical premium car has up to 70 ECUs, which are connected by several system buses to realize over 2,000 functions [2]. As around 90% of all innovations today are driven by electronics and software the complexity of cars embedded software is expected to grow. The growth is fuelled by cars beginning to act more proactively and more assistive to its drivers, which requires software to interact with hardware more efficiently and making more decisions automatically (e.g. collision avoidance by braking, brake-by-wire or similar functions). In total with about 100 million lines of code (SLOC) [20], premium segment vehicles carry more software code than in modern fighter jets and airliners [20]. Software for custom functionality in modern cars is usually developed by multiple suppliers although it is designed by a single OEM (Original Equipment Manufacturer) like Volvo Cars. The distributed development and use of standards like AUTOSAR aims to facilitate reuse of software and hardware components between different vehicle platforms, OEMs and suppliers [86]. However, testing of such systems is more complex and today testing of software generally accounts for almost 50% of overall development costs [87].

ISO-26262 in automotive domain poses stringent requirements for development of safety critical applications and in particular on the testing processes for this software. These requirements are intended to increase the safety of modern cars, although they also increase the cost of modern cars with complex software functions influencing safety or car passengers.

The position for which we argue in this paper is that efficient verification and validation of safety functions requires combining Model Based Development (MBD) with fault injection into models with mutation testing. This position is based on the studies of the ISO 26262 standard (mainly chapter 6 that describes requirements on software development but also chapter 4, which poses requirements on product development [29]). It is also based on previous case studies of the impact of late defects on the software development practices in the automotive section [33].

The requirements from the ISO 26262 standard on using fault injection techniques is challenging since it relates to the development of complete functions rather than components of sub-components of software. The current situation in the automotive sector is that fault injection is used, but it is used at the level of one electronic
component (ECU) or one software system, rarely at the function level [88] [89].

The current state of art testing is not enough for detecting safety defects early in the automotive software development process since fault injection is done late in the development (when ECUs are being developed), which usually makes the detection of specification-related defects difficult and costly [33]. This detection should be done in the model level when the ECUs functionality is still under design and thus, it is relatively cheap to redesign. The evidence from literature on successful use of fault injection shows that the technique indeed is efficient in finding dependability problems of hardware and software systems when applied to compute systems [90]. Finally, to be able to increase the effectiveness of the fault injection strategies and identify whether the faults should be injected at the model, software or ECU level mutation testing should be applied to verify the adequacy of test cases and finally how the combination of these approaches when applied at the model level will enhance the detection of safety defects right at the design stage.

In this paper, we provide a roadmap, which shows how to introduce fault injection and mutation testing to modelling of automotive software in order to avoid costly defects and increase the safety of modern and future cars.

The remaining of the paper is structured as follows: In the next section (2) we provide an overview of software development in automotive domain and associated concepts. This is followed by brief discussion on related work in section 3 and our position is presented and discussed in section 4. Section 5 concludes our work.

5.2 Background

In this section we take a brief overview on the current state of automotive software development process and environment, how safety is important in safety critical applications and overview of theoretical background on fault injection techniques and mutation testing.
5.2.1 Automotive Software Development & ISO 26262

Various software functions/applications developed within the automotive industry today are classed as safety critical for example Volvos City Safety consists of components that are safety critical.

Broy [2] gives examples of functions/areas within automotive domain of recent development which includes crash prevention, crash safety, advanced energy management, adaptable man-machine interface, advanced driver assistance, programmable car, car networking etc., much of these fall within the safety critical functionality and demands high quality and reliability. Also a number of on-going projects are directed towards the goal of self-driving cars.

Software development in automotive sector in general follows the V process, where OEMs take the responsibility of requirement specification, system design, and integration/acceptance test. This is followed by the supplier, which develops the actual code that runs on ECUs. Although the code is tested at the supplier level (mainly unit testing), the OEMs are responsible for the final integration, system and acceptance testing to ensure that the given implementation of a software (SW) meets its intended functional and safety goals/demands.
In this model of software/product development (see Figure 5.2) testing is usually concentrated in the late stages of development, which also implies that most of the defects are discovered late in the development process. In a recent study using real defect data from an automotive software project from the industry [33] showed that late detection of defects is still a relevant problem and challenge yet to overcome. The defect inflow profile presented in this study is presented in Figure 5.3 for reference, which exhibits a clear peak in number of open defects in the late stages of function development/testing.

Testing the software is an important tool of ensuring correct functionality and reliability of systems but it is also a very resource intensive activity accounting for up to 50% of total software development costs [91] and even more for safety/mission critical software systems. Thus having a good testing strategy is critical for any industry with high software development costs. It has also been shown that most of the defects detected during testing do not depend on actual implementation of code, about 50% of defects detected during testing in the study by Megen and Meyerhoff [92] were found during the test preparation, an activity independent of the executable code. And since automotive sector has already widely adopted MBD for the software development of embedded systems, a high potential exists for using the behavioural modes developed at the early stages of software development for performing some of the effort spent on V&V (Verification & Validation). Early V&V by helping to detect defects early will potentially save significant amount of cost for the projects.
Figure 5.3: Defect inflow profile for automotive software project, as given in [33].

5.2.2 ISO 26262

ISO/IEC 26262 is a standard describing safety requirements. It is applied to safety-related systems that include one or more electrical and/or electronic (E/E) systems. The overview of safety case and argumentation is represented in Figure 5.4.

Written specifically for automotive, the ISO-26262 standard is adapted for the V-model of product development corresponding to the current practice in the industry. The guidelines are laid out for system design, hardware and software design and development and integration of components to realize the full product. ISO-26262 includes specifications for MBD and provides recommendations for using fault injection techniques for hardware integration and testing, software unit testing, software integration testing, hardware-software integration testing, system integration testing and vehicle integration testing. Although the functional safety standard specifies clearly the recommendations for using fault injection during various stages of testing but does not recommend anything with respect to using mutation testing. This also reflects the current
5.2. BACKGROUND

- **Item**: The item representing a system or a function is defined.
- **PHA**: A Preliminary Hazard Analysis & Risk Assessment is done to assign an appropriate ASIL level.
- **SG**: Safety Goals are derived from the Hazard Analysis and they inherit the assigned ASIL level.
- **FSR**: Functional Safety Requirements are drawn such that the set Safety Goals are met.
- **TSR**: The Technical Safety Requirements are formulated describing how to implement FSR.
- **Doc**: Further development includes implementation, integration and documentation of safety cases.

Figure 5.4: Overview of ISO-26262 safety case & argumentation process.

standard practice within the automotive industry where mutation testing is not widely adopted yet.

### 5.2.3 Fault Injection

Fault injection techniques are widely used for experimental dependability evaluation. Although these techniques have been used more widely for assessing the hardware/prototypes, the techniques are now about to be applied at behavioural models of software systems [93], thus enabling early verification of intended functionality as well as enhancing communication between different stakeholders. Fault injection techniques applied at models level offer distinct advantages especially in an industry using MBD, but use of these techniques at model level in automotive industry is currently at its infancy. Figure 5.5 shows a mind map of classification of fault injection techniques based on how the technique is implemented; some of the tools which are developed based on given approach are also listed for reference. For a good overview of fault injection techniques readers are referred to [90] [94].
Figure 5.5: Common classification of fault injection techniques and implementation tools, description available in [90] [94]

5.2.4 Mutation Testing

Mutation testing is technique for assessing the adequacy of given test suite/set of test cases. Mutation testing includes injection
of systematic, repeatable seeding of faults in large number thus generating number of copies of original software artefacts with artificial fault infestation (called a mutant). And on the basis of what percentage of these mutations are detected by the given test cases/suite gives a metrics (called mutation adequacy score [95]) which can be used for measuring the effectiveness of given test suite. Faults for mutation testing approach can be either hand written or auto-generated variants of original code. The effectiveness of this approach in mimicking the real faults has also been established [96] i.e. mutants do reflect characteristics of real faults. Mutation theory is based on two fundamental hypotheses namely Competent Programmer Hypothesis (CPH) and the Coupling Effect, both introduced by DeMillo et al. [97]. CPH at its core reflects the assumption that programmers are competent in their job and thus would develop programme close to correct version while coupling effect hypothesis according to Offutt is Complex mutants are coupled to simple mutants in such a way that a test data set that detects all simple faults in a program will detect a high percentage of the complex defects [98].

5.3 Related Work

A number of European Union sponsored projects have within the area of embedded software development and safety critical systems have looked at and developed techniques to effectively use fault injection for safe and reliable software development. The examples include the ESACS [99] (Enhanced Safety Assessment for Complex Systems), the ISAAC [100] (Improvement of Safety Activities on Aeronautical Complex systems). These projects have used the SCADE (Safety-Critical Application Development Environment) modelling environment to simulate hardware failure scenarios to identify fault combinations that lead to safety case violations.

A model-implemented fault injection plug-in to SCADE called FISCADE is introduced in [101] which utilizes approach similar to mutation based testing and replaces the original model operators by equivalent fault injection nodes. The derived models are then used to inject the fault during execution and log the results which are analysed later. Dependability evaluation of automotive functions using model based software implemented fault injection techniques have also been studied in [102].
A generic tool capable of injecting various types of faults on the behavioural or functional Simulink models is also developed and introduced [93]. The tool called MODIFI (or MODel-Implemented Fault Injection tool) can be sued to inject single or multiple point faults on behavioural models, which can be used to study the effectiveness/properties of fault tolerant system and identify the faults leading to failure by studying the fault propagation properties of the models.

Another work [103] with its root in the European CESAR (Cost-efficient methods and processes for safety relevant embedded systems) project provides a good theoretical overview of how fault and mutation based test coverage can be used for automated test case generation for Simulink models. We provide a practical framework on how fault injection combined with mutation testing within an MBD environment can be used in the industry. And how will this practice enhance the verification and validation of software under development, its functional validation that would generates statistics for the effective argumentation of ISO 26262 compliance.

5.4 Road Map for Early Defect Detection

We contend that fault injection can be effectively used at the model level to verify and validate the attainment or violation of safety goals. By applying mutation testing approach at the model level enough statistical evidence will be provided for the coverage needed for argumentation of fulfilment of safety goals as per the ISO 26262 safety standard requirements.

A major challenge in successful argumentation of ISO-26262 compliance is to provide statistical evidence that SGs would not be violated during operation and doing this within reasonable testing efforts.

If we are able to differentiate early between defects that will or not cause the violation of SGs, the amount of testing required will be manageable. With MBD the testing for functionality under these defect conditions could be modelled using fault injection techniques, while the possibility of implementation bugs in the actual code can be checked using the mutation testing approach. The
framework on how this could be achieved in practice is as follows:

![Figure 5.6: MBD based representation of a general system with inputs, outputs and dependencies.](image)

As illustrated in Figure 5.6, a given system/function generally has following common features (in context of model based development): firstly it will have \( x \) inputs \((i_{1,2,\ldots,x})\); it would have dependencies to other \( y \) components/ functions \((d_{1,2,\ldots,y})\); it will have \( z \) outputs \((o_{1,2,\ldots,z})\); and it will have a number of sub-units/modules within it that implements the intended functionality, let us assume that this part contains \( n \) basic blocks in the modelling environment corresponding to \( n \) statements for a hand written code. To verify and validate the correct functionality and ISO-26262 compliance of this generic function using fault and mutation testing approach we can follow the steps as:

- Assign or define the Functional Safety Requirements (FSRs) and Technical Safety Requirements (TSRs) for the \( z \) outputs of the given system/function in accordance to ISO 26262.

- Use fault injection technique to inject common occurring defects and other theoretically possible fault conditions at the \( x \) inputs.

- By studying the fault propagation of different injected faults at inputs and their effect on outputs, the individual faults and combinations of it that violate the FSRs for given system can be noted.
• Steps (b) & (c) should also be done to test and validate the given system/function dependencies on other functions/components.

• Mutation approach is then used to inject faults (or cause mutations) to the n basic blocks of given functional model and assess the detection effectiveness of test suite/cases for possible implementation bugs.

• The mutants which are not killed by given set of test cases/suits are examined for their effect on given functions FSRs, if the given mutation violates the SGs/FSRs then a suitable test case will be created to detect/kill such mutants i.e. detect such bugs in actual code.

Thus by following the above mentioned steps we not only ensure that the given function works as intended, does not violate the SGs and TSRs under faulty inputs and/or due to dependencies on other functions, but we can also identify possible implementation defects using the mutation approach and ensure that we have test cases ready to catch such faults that can potentially violate the SGs/TSRs even before the code is implemented/generated.

Further to make this framework/approach more effective in industrial practice we identify some best practices that will have positive impact on detecting defects early in the development process and thus have effective V&V of ISO 26262.

• Model evolution corresponding to different levels of software/product development.

• Specification and testing for SGs, FSRs and TSRs on the behavioural models.

• Identification of different types of defects/types of faults and at what stage they could be modelled/injected at models to ensure that models are build robust right from the start instead of adding fault tolerance in later stages of development.

5.5 Conclusions

In this paper we have examined the growing importance of software in automotive domain. The development of software in automotive
and other similar industries has widely adopted the paradigm of model based development and by the nature of application much of the functionality developed and implemented in these sectors is safety critical. Safety critical software/application development requires observation of stringent quality assessment and adherence to functional safety standards such as ISO 26262 in automotive and DO173 in aerospace industry.

Development of behavioural models in MBD offers significant opportunity to do functional testing early in the development process. Fault injection and mutation testing approach in combination can be used to effectively verify and validate the functional properties of a software system/function. The approach will also provide the required statistics for the argumentation of safety standards compliance. In this paper the need for such validation and a framework on how this could be achieved in practice is discussed. More research and tools are needed to bring this approach into wider industrial adoption.

By detecting defects early and being able to do much of verification and validation of intended functionality, robustness and compliance to safety standards on the models the quality and reliability of software in automotive domain will be significantly enhanced. More effective approaches and tools support will also reduce the V&V costs and lead to shorter development times. High quality, reliable and dependable software in automobiles brings innovative functionality sooner, keeps product costs lower and most importantly ensures that automobiles are safer than ever before.
Chapter 6

Paper E

Improving Fault Injection in Automotive Model Based Development using Fault Bypass Modeling


Published at 2nd Workshop on Software-Based Methods for Robust Embedded Systems (SOBRES’13), Koblenz, Germany, Sep 2013
Abstract

Fault injection is widely used for validating dependability of computer systems. These techniques have been traditionally used for testing dependability of the both hardware and software systems. With widespread use of model based development in automotive software development more sophisticated needs arise for using fault injection techniques at the model level, which can yield significant benefits in combination with model-based testing or model mutation. In this paper, we address challenges with injecting faults into behavioural models in terms of analysis of results and propose a framework for distinguishing between correct and incorrect simulation results. The focus is laid on an important challenge encountered when injecting faults in continuous models, i.e. managing system-environment interdependencies. We analyse the problem in details and outline an effective approach to deal with this problem.
6.1 Introduction

Dependability is a very important feature of any computer system [104]. A crash or system failure causes discomfort for consumers who then associate the product with low quality. For certain safety critical applications, such as within aerospace and automotive, these failures may result in life-threatening situations for the occupants. Therefore dependability validation is strictly monitored and regulated in development of safety critical applications - for example by following standards like ISO 26262 [29] in automotive and DOC-178C for safety-critical software development within the aerospace industry.

Fault injection is an important and widely used technique, it has been extensively used as technique for experimental dependability evaluation of computer systems and for evaluation of fault-tolerance mechanisms. Fault injection has traditionally been used for emulating hardware and software faults on the prototypes. The prototype-based fault injection has following advantages as given in [104].

- identify dependability bottlenecks,
- study system behaviour in the presence of faults,
- determine the coverage of error detection and recovery mechanisms, and
- evaluate the effectiveness of fault tolerance mechanisms (such as reconfiguration schemes) and performance loss.

Alongside the evolution of requirements for dependability, software product development methodologies have significantly changed over the last decade. Model Based Development (MBD) has been embraced widely as preferred tool for developing software systems. Many sectors such as aerospace and automotive domain today widely use MBD for new software/function development [22]. As a consequence of this widespread use of MBD, code generation has also gained momentum in these industries and Model Based Testing (MBT) alongside with it. Traditional software development within automotive and many other domains can be approximated to the established V-model. In this model of software/product development, testing is heavily focused on the right side, which leads
to most defects being discovered late in the development process; however MBD has the potential of shifting some of the effort spent on Verification & Validation (V&V) to the left arm of V-model, i.e. by allowing testing of models at early development phases some of the defects can be detected much sooner, this shift could save significant amount of resources/cost for the projects \[104\] and also reduce development time.

Megen and Meyerhoff in their study \[92\] showed that about 50% of defects detected during testing are found in test preparation, an activity which does not depend on the executable code. Still previous case studies within the automotive domain, peak in open defects is observed during the late stages of software development just before the release dates \[33\]. These observations suggest that by using MBD and through testing models early in the development phase, a possibility exists for shifting the open defect peak from the right-hand side (close to release) towards earlier phases of the project. This shifting of defect detection to the left (early detection) can provide better opportunity for the teams to react early upon problems and thus avoid late defects. MBD offers an important advantage with respect to early testability - the executable models can be simulated to verify and validate the functionality at a very early stage of development process. And by using fault injection in these behavioural models, dependability measurements can also be done very early on; robust, high fault-tolerant models thus can be identified and developed. The functional and logical errors can also be identified early in the development process, thereby saving development time and costs significantly.

For a majority of software functions developed within automotive and many other domains - real time behaviour is an important factor and often test cases depend on the system behaviour \[105\]. One way to test the complex functional behaviour of such systems in real time is to use closed loop testing of models with continuous signals, but testing systems with continuous signals (at least in automotive domain) is poorly supported by existing test methods which are generally data-driven \[105\]. Two major obstacles for testing behavioural/functional models of software systems in a closed loop configuration with continuous signals using fault injection techniques are:

- realistic environment modelling, and
- obtaining realistic system response in presence of injected
faults to correctly differentiate between normal system behaviour from system failure.

With regard to the first obstacle, most industrial domains currently using MBD on large extent are either using, developed or are developing virtual environmental models, which are capable of simulating full system and environment conditions that are very close to real ones. These models are intended not only to aid and accelerate new conceptual/functional development but also to test developed functional models in MIL (Model in Loop) testing for possible logical errors [101]. Development of these extensive environmental models addresses the first challenge.

In this paper we take a closer look at the second obstacle (2) and propose a framework to overcome it. The rest of this paper is structured as follows. In the next part (section 2) we provide an overview of fault injection techniques, which is followed by a discussion on related work in section 3. In section 4, we describe and analyse the problem with injecting faults on behavioural models using an example. Section 5 introduces the proposed solution to the described problem followed by conclusions in section 6.

6.2 Fault Injection

A system may not always perform as intended or expected. The causes and consequences of system performance deviations from the expected function are referred as factors to dependability. Dependability evaluation involves study of failures and errors. Important aspects of dependability are defined comprehensively in work by Avizienis et al. [27]. Dependability is particularly important when system/software being developed is safety critical, where failure can cause serious hazard or loss of life. Functional safety standards such as IEC-61508 [93] mandate use of fault injection technique for system development, while it is recommended or highly recommended in automotive safety standard ISO 26262 [106].

Based on injection of faults to the actual hardware/prototype or on its model the fault injection techniques can be classified as physical or simulation-based. While based on the implementation of fault injection mechanisms, the techniques can be classified as hardware-implemented fault injection (HIFI) or software-implemented fault injection (SWIFI) techniques. A brief overview
Table 6.1: ISO-26262 Recommendation for Using Fault Injection Techniques

<table>
<thead>
<tr>
<th>ISO 26262 Chapter</th>
<th>Reference to recommendation</th>
</tr>
</thead>
</table>
| 4 Hardware-software integration and testing | • Table 5 - Correct implementation of technical safety requirements at the hardware-software level.  
• Table 8 - Effectiveness of a safety mechanisms diagnostic coverage at the hardware-software level. |
| System-integration and testing | • Table 10a - Correct implementation of functional safety and technical safety requirements at the system level.  
• Table 13b - Effectiveness of a safety mechanism’s failure coverage at the system level. |
| Vehicle-integration and testing | • Table 15 - Correct implementation of the functional safety requirements at the vehicle level.  
• Table 18 - Effectiveness of a safety mechanism’s failure coverage at the vehicle level. |
| 5 Hardware-integration and testing | • Table 11 - Hardware integration tests to verify the completeness and correctness of the safety mechanisms implementation with respect to the hardware safety requirements. |
| 6 Software-unit testing | • Table 10 - Methods for software unit testing. |
| Software-integration and testing | • Table 13 - Methods for software integration testing. |
of fault injection techniques is provided here, whereas a more detailed description on different fault injection techniques, their advantages, drawbacks and important tools is presented in [90], [94].

In hardware-based fault injection, faults are injected at the physical level by controlling the environment parameters. Faults may be injected by injecting voltage sags, disturb the power supply, heavy ion radiation, electromagnetic interference etc., while software-based fault injection refers to techniques that inject faults by implementing it in the software, different types of faults can be injected for example register and memory faults, error conditions and flags, irregular timings, missing messages, replays, corrupted memory etc. for using fault injection techniques where target is a software application, fault injector may be inserted within the application or it can also be inserted between the target and operating system, while in case where the target is operating system, fault injector used have to be embedded within itself [104]. Software-based fault injection techniques can be classified into compile-time faults or run-time faults based on when the faults are injected. Software implemented fault injection methods can be adapted to inject faults on various trigger mechanisms such as exception, traps, time-out, code-modification etc.

On the other hand, simulation-based fault injection [94] involves constructing a simulation model of given hardware using hardware description languages such as VHDL and faults are injected into these models during simulation. Two important approaches to inject faults within simulation-based techniques are:

- Those that need modification to VHDL code, and
- Those that use modified simulation tools using built-in commands of VHDL simulators.

Simulation-based fault injection techniques have been quite powerful and widely used for hardware models, though their application on behavioural models for software artefacts has been limited.

6.3 Related Work

While MBD has been widely adopted as the development methodology in automotive domain, but quality assurance for software
that takes full advantage of MBD is still not well supported. Bringmann and Kramer [105] suggest that in practice only a few automotive domain-specific model based testing procedures are available, the applied test methods and tools are often proprietary, ineffective and requires significant resource input in form of effort and money. They highlight the main requirements for successful automotive model based testing, some of which are

- Reactive testing/ closed loop testing
- Real-time issues and continuous signals
- Testing with continuous signals

The authors also introduced TPT (Time Partition Testing) as a model based testing approach, its test cases can be used on different test platforms and can be executed in real time. Lamberg et al. [107] also describes a systematic way of testing embedded software for automotive electronics, the process referred as MTest allows model-based testing in early function and software development, but it does not use or suggest using fault injection for dependability evaluation. In this paper we introduce the fault bypass modelling (FBM) framework which addresses same requirements to allow closed loop testing with continuous signals but at model level under fault injection conditions.

Techniques for injecting faults into system models have been developed and evaluated in ESACS [99] and ISAAC [100] projects using SCADE (Safety-Critical Application Development Environment) modelling language to simulate hardware failure scenarios, these techniques were applied to identify fault combinations that lead to safety case violations. In [101] a model-implemented fault injection plug-in to SCADE called FISCADE is introduced which can replace original operators by fault injection nodes. During execution FISCADE controls the SCADE simulator to execute the model, inject the fault and log the results. Model based software implemented fault injection techniques have also been used for dependability evaluation of automotive functions such as in [108]; our work complements these earlier works by presenting a framework to manage system-environment interdependencies under fault injection conditions.

A further attempt to use fault injection techniques for dependability evaluation of software functional model is done with development of MODIFI tool [93]. MODIFI (or MOdel-Implemented
Fault Injection tool) extends the fault injection methodology to behaviour models in Simulink. The tool allows for introducing single or multiple points faults on behavioural models, the fault tolerant system properties are studied by analysing faults leading to failure. But injecting faults in behavioural models is not same as injecting faults in physical hardware prototypes or their models; the interdependencies between the system and their environment may lead to inconsistent simulation results under fault injection modes. FBM offers the solution in form of fault bypass principle to the single factor causing incorrect system behaviour under fault injection mode for behavioural models.

6.4 Problem Analysis

Fault injection techniques are very useful to test the robustness and dependability of systems. The methodology is straightforward; however in practice for behavioural models, systems are not passively related to their environment. The relationship between system and its environment is active, which means that there are feedback loop(s) between the system and the environment. In such cases a fault injected into the system does not necessarily gives the output which is purely the property of that system, for system-environment models, any change in input of the system not only affects the system itself, but can also influence its environment which is again fed back to the system through single or multiple feedback loops. In continuous models it often results in unrealistic triggers/control values from system to its environment and in turn wrong/unnatural values for environment parameters to the system. Thus the outputs in these cases are unreliable making it difficult to distinguish between a correct system behaviour and system failure. We take a closer look on this problem with an example from automotive domain, the ABS.

6.4.1 CASE: ABS Anti-Lock Braking System

ABS system takes a given vehicle speed and the wheel speeds from different sensors on board the vehicle (its environment), the system uses these inputs to calculate relative slip value at each wheel. Based on this relative slip value a control signal is generated which controls the activation/deactivation of brake pressure valve in ac-
Figure 6.1: Simple ABS model representation in Simulink based on [109]. In this model wheel speed and vehicle speed is used to calculate the relative slip value, which is compared against the desired set value of relative slip (for maximum traction). The ABS controller (here embedded within the Wheel Speed block) activates/releases brake pressure according to anti-lock principle when relative slip value is different from desired slip value; rest of the blocks in the model are used to simulate the dynamics of a moving vehicle.

cordance to Anti-Lock Braking principle. A simplistic example of single wheel ABS system model in Simulink is presented in figure 6.1. (for detailed component description and working of model, refer to [109]). The same system can be represented in a simplified form (figure 6.2), which separates the elements of ABS system and functions/parameters that are needed to simulate the system (collectively called its environment). Executing the system-

Figure 6.2: ABS model represented in System-Environment configuration.

environment model to verify ABS functionality gives the following
outputs describing vehicle speed, wheel speed which matches to expected output of ABS functionality (refer to figure 6.3). Solid lines represent Vehicle speed, while dashed lines are used for wheel speed. Vehicle and wheel speed without an ABS system shows wheel locking at about t=1.2 sec and vehicle taking about 14.5sec to come to a standstill from the point of brakes application, while with the use of ABS system the wheel locking is prevented due to ABS system response and vehicle is stopped approximately at t=11sec.

Also one can verify the reduction in stopping distance obtained by using the ABS system compared to one without it, indicated here in figure 6.4.

6.4.2 Naïve fault injector

In figure 6.5 we show an additional block - the fault injector - capable of generating different faults and injecting them into the input
or the output signals of our ABS system. The fault injector can be used to inject faults both at the input signal and the output signals of system to evaluate fault tolerant capacity of given system implementation and also to study the system behaviour/characteristics under these conditions. To simulate a situation where a wheel speed sensor is faulty, we inject a fault at wheel speed signal, for example injecting a fault by adding a high pulse starting at t=6sec - this high constant pulse injection simulates a real condition approximation where the sensor relays permanently high value as its output signal due to a fault/sensor failure. The resulting vehicle and wheel speed of our system-environment model simulation under given fault condition is presented in figure 6.6. Under the given state/scenario, model simulation shows vehicle speed increasing exponentially starting at t=6sec (the instant of fault injection) which is unrealistic. It is understandable that in a real system even if the wheel speed sensor would provide a faulty signal to ABS system due to malfunctioning, the vehicle would stop under normal braking conditions. But due to feedback loop between the system and environment we get wrong results, making it difficult to evaluate/study the correct system behaviour in presence of faults - at
Figure 6.5: ABS system-environment model representation in Simulink with fault injector setup.

Figure 6.6: Vehicle and wheel speed with ABS system with fault injection.

least in an automated manner.

One solution under such a situation, which is also widely used in the industry, is to use open loop discrete models instead of closed loop continuous models. The open loop testing is represented in
6.5. FBM: FAULT BYPASS MODELING

Figure 6.7: ABS system in open loop discrete model configuration.

the output is saved as data file and compared to reference/expected output. The major limitation with such testing is that its limited by the availability of recorded sensors data as well as need to have the correct output for reference purposes. Thus to test systems under conditions where the input and output data is already not available or if a new functionality is developed or existing system configuration changed, the input/output data may not be available and thus this type of testing infeasible. Closed loop continuous models do not suffer from these limitations.

6.5 FBM: Fault Bypass Modeling

Having described the problem in simple terms we analyze why we observe an unrealistic system response in the ABS system; this will also help us understand the proposed solution framework, which we call “Fault Bypass Modelling (FBM)”.

To test the ABS function/system under MIL, we have to simulate the environment. The environment simulates the vehicle dynamics and gives wheel and vehicle speed as input to the ABS system. The ABS generates the control signal, which controls the brake pressure like in a real vehicle. But since one of the environment parameters (in this case $\mu$ or coefficient of friction between the wheel and the road) is dependent on the value of relative slip, real time $\mu$ value can only be calculated using corresponding relative slip value calculated within the system and provided to environment. The fact that a natural parameter is calculated based on a property of the system introduces a “superficial” loop between system and its environment. This “superficial” loop is only present in the model, as in the real vehicle the value of $\mu$ would be naturally determined based on actual relative slip value at each wheel-road
surface. This superficial loop is necessary for the system model to be executable as realistic real time $\mu$ values during the simulations are obtained by its virtue, but under fault injection modes this loop becomes the source of unrealistic feedback mechanisms by providing un-natural values of $\mu$.

Thus in such cases, in order to obtain a realistic/corrected system behaviour under fault injection conditions, the FBM principle is described as following:

“If a signal injected with faults or its derivative is used to calculate/control any natural environment parameter(s)$^1$, the part of signal or its derivative which is used to calculate/control the environment parameter(s) should be made fault free to break the unrealistic feedback loop.”

Figure 6.8: ABS system-environment model representation in Simulink with fault injector setup using FBM.

The FBM setup although very similar to control system setup, the FBM principle is fundamentally different from that of control theory. In control system the controller senses the output of system, compares it to the expected/desired behaviour and computes the corrective action based on model of systems response [110],

$^1$Natural Environment Parameter here refers to such a parameter, which is not a property of system but needs correct value from system to define its correct state/value.
whereas FBM principle is used while designing new systems to get the correct system response under fault injection conditions. The desired behaviour of system is mostly unknown in case of FBM, while it is a must for using a control system using control theory.

Use of FBM principle in our case of ABS system is as follows: since the value of $\mu$ in on-road conditions is not affected by the calculated value of wheel speed sensors but by the real wheel speed, the effect of fault injection on this signal should be bypassed to calculate the real value of relative slip and provided to the environment model for correct value prediction of $\mu$ according to real natural scenario. This is achieved in our model by using a second instance of our system model (ABS Reference System-1 in figure 6.8) which takes the system inputs which by pass the fault injection and control module is used to provide real relative slip value to the environment model, while the ABS system control value (ctrl_sig) that is affected by the injected fault is not bypassed and used to determine the actual system behaviour under fault condition.

Using the proposed FBM, the system can be simulated with various types of faults to simulate different faulty conditions that can affect the ABS system. Figure 6.9A shows the system output that matches the tester’s expectation on encountering a faulty wheel speed sensor, the wheel gets locked soon after the fault is injected and vehicle stops under normal braking conditions without ABS system functionality.

Figure 6.9B indicates an expected increase in stopping distance by about 6meters and stopping time by about 3sec, if wheel speed sensor is failed in the described conditions, such analysis and information availability at design or early development phase is highly valuable to validate as well as to improve the system design and implementation.

### 6.6 Conclusions

Model-based development is already widely adopted in automotive and other sectors, software components in these domains are no longer hand written but usually modelled with MATLAB/Simulink, Statemate or similar tools. These behavioural models offer significant opportunity to verify and validate intended functionality and assess their dependability at an early development stages. Fault injection techniques can be used for dependability evaluation at
Figure 6.9: Using FBM shows for the given case the wheel gets locked at $t=6.3$ sec, soon after the time of fault injection and the effect on the stopping distance which increase by approx. 6 m under given fault conditions.
model levels as they have been used for hardware artefacts. However, interdependencies between the system and its environment at model level may cause unrealistic system behaviour under fault injection conditions. In this paper we observed this with the help of ABS system example in Simulink. The problem limits the use of fault injection techniques at early stages of development which implies difficulty in testing behavioural models for real life scenarios such as sensor failures, disrupted signals, impact of noise etc.

A framework referred to as Fault Bypass Modelling (FBM) is introduced and evaluated using a case example from automotive domain. Using FBM framework helps obtain realistic system behaviour under fault injection modes. This ensures that correct system behaviour/response under real life situations can be studied and analysed very early in the development cycle. A number of test cases can be designed based on known failure modes and simulated in virtual environment to test models allowing the selection of models with best fault tolerant properties for further development.

Allowing system behaviour analysis early in the development cycle will help in reducing the late defects; improve the quality of function/software under development and also reduce the development time. Further early system behaviour analysis for real life scenarios also imply better communication and understanding between the multidisciplinary development teams. These advantages are especially useful while developing safety critical functions where quality and reliability are of prime importance. Today, embedded software functions development in automotive as well as other domains such as aerospace is increasingly designed using model-based methods.

MBD has many advantages most of which are well documented. One of the major advantages of MBD in safety critical systems development is availability of behavioural models for testing of functional decomposition before the logical system structure is set. The testing process can include various techniques, but one of the most efficient ones is injecting faults and testing the robustness of the algorithms/functions. Historically, the use of fault injection techniques for closed loop model testing has been limited due to difficulties in separating the system failure due to injected fault or system failure due to un-natural feedback as result of system-environment interdependencies. Using FBM resolves this specific problem thus making it possible to use fault injection techniques
correctly to test behavioural models for dependability evaluation, robustness and correct functionality. Applying FBM means that models can be tested under closed loop configuration with continuous or discrete signals. Close loop testing allows test engineers to build/model, number of test cases based on real scenarios - thus quickly expanding the test space without exponential growth of testing effort. Combining fault injection on the model level with continuous testing, nightly testing or similar techniques, provides possibilities of continuous quality assurance of functionality and shorter feedback loops. Storing test cases with injected faults in libraries and adding new test cases over time decreases the probability of defects slipping to the customers which is unacceptable for safety critical software.

By allowing more efficient testing of models, FBM helps in early defect detection which not only saves significant cost but also reduced development time. Using fault injection techniques at models level for dependability evaluation implies that dependability evaluation of given system can be done already at the function development level (compared to system development or integration level as it is done today), more robust models can thus be identified and developed, effectiveness of fault-tolerance mechanism can also be evaluated and shortcomings identified/removed early. Preliminary evaluation of FBM applicability in industrial setting was done at Volvo Cars by conducting two semi-structured interviews with designers/developers with minimum experience of 15 years each in model development and testing. The feedback obtained was positive, indicating that it could prove to be useful to evaluate correctly what-if scenarios for complex functions very early in the development process. However it was also pointed out in the initial evaluation that FBM if used should be integral part of environment models as the system developers and testers expects the environment models to be free from superficial loops/system dependencies.

Future work in this area is expected to use and validate the framework with further case studies on industrial scale models and incorporate the initial feedback by making FBM integral part of the environment model, this would mean that system developers and testers do not have to take into account the FBM, but the development/modelling of environment models for the given system, FBM should be implemented to allow correct model testing using
fault injection techniques.
Chapter 7

Conclusion

7.1 Future Directions

Exploratory analysis of defect inflow data distribution from industrial software projects: Exploratory data analysis can provide helpful insights to visualize and understand the defect inflow distribution and with the aid of simple statistical analysis better decisions can be made with regard to reliability model selection. There has been less attention given to the exploratory analysis of defect inflow data and thus a possible work for future.

Evaluating SRGMs and their long-term predictive power for embedded software projects from industry: The focus of reliability modelling research has been on proposing and finding models that fit the defect inflow data best with less emphasis on their long term predictive power. To increase the external validity of results presented in chapter 4 and to be able to generalize results over other industrial domains, long-term predictive power of SRGMs should be evaluated on wider industrial domains such as embedded software development projects.

Explore Machine Learning to Predicts Defects and Analysing Risks in Large Software Development Projects: Recently data mining and Machine Learning (ML) techniques have been applied in many software engineering problems with acclaimed success including defect prediction. These new techniques looks promising and are expected to grow which also present high opportunity and need for research in this direction.
SRGMs based on functional/behavioural models of software: A number of industrial domains including automotive and aerospace and widely adopting model based development paradigm to develop software. Given these functional models are available for reliability analysis at different abstractions at early stages of software development, reliability growth models based on such models can be very useful in these industries. Thus this presents an area of high impact future research direction.

Measuring impact of software quality and/or software reliability assessment on software development projects: A number of quality models and reliability growth models are available and also used to different degrees in industrial domains, but where, what and how much these measures/evaluation impacts the actual software or software development is not well understood. Research in this direction is needed to understand and justify how much effort should be spent on these evaluations.

7.2 Conclusions

Software is relatively a recent addition to the domain of automotive product development, but in a very short span of time the contributions of software in this domain has gained high importance. Also given the nature of domain some of the software developed is classed as safety critical which demands high reliability. Mathematical models may be used to assess and predict the reliability of software. New approaches are further needed to increase the dependability of software to keep up with the pace of increasing importance and complexity of software in this domain. In this thesis we set out with two distinctive research goals, first to evaluate the applicability of SRGMs in the context of automotive software development and to propose and evaluate methods with the potential of increasing reliability of automotive software.

The first research goal was addressed by analytical research done by applying commonly used SRGMs on the defect inflow data from automotive domain. The applied models were assessed on their ability to fit and predict. The evaluation was also done on different models ability to make long-term predictions which is important from the practical perspective. The results obtained showed
that SRGMs if applied correctly can be applied in the context of automotive software and some models do have high potential to aid in the assessment of software reliability growth in automotive software. Thus models may also be useful for adjusting the testing effort during on-going projects to achieve desired dependability characteristics by the set time line. The additional methodological question to support the first research goal confirmed the superior characteristics of Maximum Likelihood Estimation (MLE) with respect to parameter estimation but also highlighted its weaknesses. It also suggest that while Non-Linear Regression (NLR) is used widely and is easy to apply, care needs to be taken when applying NLR specifically to partial data sets and where possible MLE should be the preferred method for parameter estimation when applying software reliability growth models.

Research goal two to propose and evaluate methods to increase the reliability of software particularly in automotive domain was addressed in chapters 5 and 6. A framework was proposed to combine fault injection and mutation testing approaches which when applied at the models level can be used to increase the efficiency of ISO-26262 verification and validation. The framework also addresses the problem of late defect discovery and thus has a potential to improve the overall quality of software in the automotive domain. Further a new approach called fault bypass modelling was also proposed and evaluated with high potential to make environment modelling more robust. Fault bypass principle offers a simple solution to a problem of high practical relevance which makes it effective for using models for early verification and validation. The use of models for analysing the system response under faulty conditions and to evaluate correctly the system behaviour in what-if scenarios will lead to wider use of models for early validation which helps in building robust and quality software right from the start.

Thus this thesis has shown how SRGMs can be applied for modelling and predicting reliability growth of software in automotive domain. The thesis also proposed methods and framework to increase the reliability of software in this domain and in general. While addressing the set out research goals the thesis suggests future research directions that will make the defect detection and prevention in automotive domain more effective in future.