UNDERSTANDING AND IMPROVING REGRESSION TESTING PRACTICE

Nasir Mehmood Minhas

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2022:07

Department of Software Engineering

Understanding and improving regression testing practice

Nasir Mehmood Minhas

Blekinge Institute of Technology Doctoral Dissertation Series
No 2022:07

Understanding and improving regression testing practice

Nasir Mehmood Minhas

Doctoral Dissertation in
Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SWEDEN

2022 Nasir Mehmood Minhas
Department of Software Engineering
Publisher: Blekinge Institute of Technology,
SE-371 79 Karlskrona, Sweden
Printed by Exakta Group, Swede, 2022
ISBN: 978-91-7295-444-1
ISSN: 1653-2090
urn:nbn:se:bth-23634

Abstract

Background Regression testing is a complex and challenging activity and consumes a significant portion of software maintenance costs. Researchers are proposing various techniques to deal with the cost and complexity of regression testing. Yet, practitioners face various challenges when planning and executing regression testing. One of the main reasons is the disparity between research and practice perspectives on the goals and challenges of regression testing. In addition, it is difficult for practitioners to find techniques relevant to their context, needs, and goals because most proposed techniques lack contextual information.

Objective This work aims to understand the challenges to regression testing practice and find ways to improve it. To fulfil this aim, we have the following objectives: 1) understanding the current state of regression testing practice, goals, and challenges, 2) finding ways to utilize regression testing research in practice, and 3) providing support in structuring and improving regression testing practice.

Method We have utilized various research methods, including literature reviews, workshops, focus groups, case studies, surveys, and experiments, to conduct the studies for this thesis.

Results Research and practice stress different goals, and both follow their priorities. Researchers propose new regression testing techniques to increase the test suite's fault detection rate and maximise coverage. The practitioners consider test suite maintenance, controlled fault slippage, and confidence their priority goals. The practitioners rely on expert judgment instead of a well-defined regression testing process. They face various challenges in regression testing, such as time to test, test suite maintenance, lack of communication, lack of strategy, lack of assessment, and issues in test case selection and prioritization.

We have proposed a GQM model representing research and practice perspectives on regression testing goals. The proposed model can help reduce disparities in research and practice perspectives and cope with the lack of assessment. We have created regression testing taxonomies to guide practitioners in finding techniques suitable to their product context, goals, and needs. Further, based on the experiences of replicating a regression testing technique, we have provided guidelines for future replications and adoption of regression testing techniques. Finally, we have designed regression testing checklists to support practitioners in decision-making while planning and performing regression testing. Practitioners who evaluated the checklists reported that the checklists covered essential aspects of regression testing and were useful and customizable to their context.

Conclusions The thesis points out the gap in research and practice perspectives of regression testing. The regression testing challenges identified in this thesis are the evidence that either research does not consider these challenges or practitioners are unaware of how to replicate the regression testing research into their context. The GQM model presented in this thesis is a step toward reducing the research and practice gap in regression testing. Furthermore, the taxonomies and the replication experiment provide a way forward to adopting regression testing research. Finally, the checklists proposed in this thesis could help improve communication and regression test strategy. Moreover, the checklists will provide a basis for structuring and improving regression testing practice.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisors, Professor Jürgen Börstler and Professor Kai Petersen, for their support, guidance and feedback on my work. I have learned a lot from their feedback and critique of my work. I acknowledge their continuous support in the completion of my PhD thesis. Besides their participation and guidance for the thesis, they have been inspirational mentors.

I am thankful to all the practitioners who participated in my studies. Without their collaboration, I would not have been able to complete this thesis. In particular, I am thankful to Ola Söder from Axis Communications and Peter Visuri from Extrapreneur (former Sony mobile communications), who remained committed from my first till my last study. Special thanks to Dr. Mohsin Irshad from Ericsson, who provided delightful times during and after work besides contributing to one of my crucial studies.

I thank my colleagues at SERL Sweden for providing a healthy and constructive environment. In particular, I would like to thank Dr. Mohammad Usman, Dr. Nauman Ali, Dr. Ahmad Nauman Ghazi, Dr. Usman Nasir, and Umar Iftikhar for their pleasant company inside and outside the campus. I always learned something from discussions with Dr. Muhammad Usman and Dr. Nauman Ali. I am grateful to all those who contributed to my research as co-authors.

I feel humbled while saying thanks to my parents, as my achievements are because of their prayers, support, and sacrifices. I acknowledge the sacrifices and unconditional support of my wife (Nayla) and my children. Without Nayla's and my children's support and love, I would not have completed this thesis. I must thank my brothers and sisters, who always appreciated my achievements and encouraged me to do more.

Karlskrona, September 18, 2022

Publications

Papers included in this thesis

Chapter 2 (Study 1) Nasir Mehmood Minhas, Kai Petersen, Nauman Bin Ali, and Krzysztof Wnuk. “Regression testing goals – view of practitioners and researchers”, In 24th Asia-Pacific Software Engineering Conference Workshops (APSECW) EAST-17, pp. 25–32. IEEE, 2017. https://doi.org/10.1109/APSECW.2017.23

Chapter 3 (Study 2) Nasir Mehmood Minhas, Thejendar Reddy Koppul, Kai Petersen, and Jürgen Börstler. “Using goal–question–metric to compare research and practice perspectives on regression testing”, J Softw Evol Proc 2022; e2506.
https://doi.org/10.1002/smr.2506

Chapter 4 (Study 3) Nasir Mehmood Minhas, Kai Petersen, Jürgen Börstler, and Krzysztof Wnuk. “Regression testing for large-scale embedded software development – Exploring the state of practice”, Information and Software Technology 120 (2020): 106254. https://doi.org/10.1016/j.infsof.2019.106254

Chapter 5 (Study 4) Nauman bin Ali, Emelie Engström, Masoumeh Taromirad, Mohammad Reza Mousavi, Nasir Mehmood Minhas, Daniel Helgesson, Sebastian Kunze, and Mahsa Varshosaz. “On the search for industry-relevant regression testing research”, Empirical Software Engineering, 24.4 (2019): 2020-2055.
https://doi.org/10.1007/s10664-018-9670-1

Chapter 6 (Study 5) Nasir Mehmood Minhas, Mohsin Irshad, Kai Petersen, and Jürgen Börstler. “Lessons learned from replicating a study on information-retrieval based test case prioritization”, Software Quality Journal 2022 (Submitted).

Chapter 7 (Study 6) Nasir Mehmood Minhas, Jürgen Börstler, and Kai Petersen. “Checklists to support decision making in regression testing”, Journal of Systems and Software 2022 (Submitted).

Papers that are related to but not included in this thesis

Paper 1 Nasir Mehmood Minhas, Sohaib Maqsood, Kai Petersen, and Aamer Nadeem. “A Systematic mapping of test Case generation techniques using UML interaction diagram”, Journal of Software: Evolution and Process 32.6 (2020): e2235.
https://doi.org/10.1002/smr.2235

Paper 2. Hasan Javed, Nasir Mehmood Minhas, Ansar Abbas, and Farhad Muhammad Riaz "Model Based Testing for Web Applications: A Literature Survey Presented", Journal of Software (JSW) 11.4 (2016):347–361.

Paper 3. Asad Masood Qazi, Adnan Rauf, and Nasir Mehmood Minhas "A Systematic Review of Use Cases based Software Testing Techniques", International Journal of Software Engineering and Its Applications 10.11 (2016):337–360.

Other Papers not included in this thesis

Paper 1 Nasir Mehmood Minhas. "Authorship ethics: an overview of research on the state of practice", In 2021 IEEE/ACM 2nd International Workshop on Ethics in Software Engineering Research and Practice (SEthics). IEEE, 2021.

Paper 2 Nasir Mehmood Minhas, Asif Majeed, Jürgen Börstler, and Tony Gorschek. "SWVP-A Requirements Prioritization Technique for Global Software Development", In 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 1-9. IEEE, 2019.

Paper 3 Nayla Nasir, Nasir Mehmood Minhas. "Implementing Value Stream Mapping in a Scrum-based project-An Experience Report", QuASoQ 2018 (2018): 48.

Paper 4. Jefferson Seide Molléri, Nauman bin Ali, Kai Petersen, Nasir Mehmood Minhas, and Panagiota Chatzipetrou "Teaching students critical appraisal of scientific literature using checklists", In Proceedings of the 3rd European Conference of Software Engineering Education, Pages 08–17, 2018.

Paper 5. Javed Iqbal, Muzafar Khan, and Nasir Mehmood Minhas "Are project managers informally following capability maturity model integration practices for project management?", Global Journal of Information Technology: Emerging Technologies 8.3 (2018): 86-94.

Paper 6. Ricardo Britto, Muhammad Usman, and Nasir Mehmood Minhas "A quasi-experiment to evaluate the impact of mental fatigue on study selection process", In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, Pages 264–269, 2017.

Contribution statement

Nasir Mehmood Minhas is the lead author of studies 1, 2, 3, 5, and 6. He led the design, execution, and reporting of these studies. The following paragraphs describe the authors' contribution to the studies included in the thesis.

Study 1. **Nasir Mehmood Minhas:** Conceptualization and design of the study, acquisition, analysis and interpretation of data, and drafting and revising of the manuscript.
Kai Petersen: Conceptualization of the study, reviewing the design, acquisition of data, reviewing the manuscript critically and editing it.
Nauman bin Ali: Conceptualization of the study, reviewing the design, acquisition of data, reviewing the manuscript critically and editing parts of it.
Krzysztof Wnuk: Reviewing and editing the parts of the manuscript.

Study 2. **Nasir Mehmood Minhas:** Conceptualization and design of the study, acquisition, analysis and interpretation of data, drafting of the manuscript, and revising it.
Thjendar Reddy Koppula: Acquisition and interpretation of data, and writing the parts of the initial draft of the manuscript.
Kai Petersen: Conceptualization of the study, reviewing the design, and reviewing the manuscript critically and editing parts of it.
Jürgen Börstler: Reviewing the design, reviewing the manuscript critically and editing parts of it.

Study 3. **Nasir Mehmood Minhas:** Conceptualization and design of the study, acquisition of data, analysis and interpretation of data, and drafting of the manuscript, and revising it.
Kai Petersen: Conceptualization of the study, reviewing the design, acquisition of data, and reviewing the manuscript critically and editing parts of it.
Jürgen Börstler: Reviewing the design, reviewing the manuscript critically and editing parts of it.
Krzysztof Wnuk: Reviewing and editing the parts of the manuscript.

Study 4. In study 4, **Nauman bin Ali** and **Emelie Engström** conceived, organized, and led the study. The **co-authors** mainly contributed to the study execution process.

Nasir Mehmood Minhas contributed to the following phases of research: Study design, data collection, data analysis, drafting the manuscript, and reviewing and revising the manuscript.

Study 5. **Nasir Mehmood Minhas:** Conceptualization, design, and implementation of the study; analysis and interpretation of the findings, drafting of the manuscript, and revising it.
Mohsin Irshad: Implementation of the study, interpretation of the findings, reviewing and editing manuscript.
Kai Petersen: Conceptualization of the study, reviewing the experiment design, reviewing the results, reviewing the manuscript critically and editing parts of it.
Jürgen Börstler: Conceptualization of the study, reviewing the experiment design, reviewing the results, reviewing the manuscript critically and editing parts of it.

Study 6. **Nasir Mehmood Minhas:** Conceptualization and design of the study, acquisition of data, analysis and interpretation of data, and drafting of the manuscript.
Jürgen Börstler: Conceptualization of the study, reviewing the design, reviewing the manuscript critically and editing parts of it.
Kai Petersen: Conceptualization of the study, reviewing the design, reviewing the manuscript critically and editing parts of it.

Contents

1 Introduction **1**

1.1 Overview 1

1.2 Background 3

1.2.1 Regression testing 4

1.2.2 Regression testing in practice 5

1.2.3 Goal question metric (GQM) approach 5

1.2.4 Replication 6

1.2.5 Checklists 7

1.2.6 Industry academia collaboration 8

1.3 Research gaps and contributions 8

1.4 Research questions 10

1.5 Research methods 12

1.5.1 Empirical methods 12

1.5.2 Systematic literature review 17

1.6 Threats to validity 18

1.7 Summary of studies included in the thesis 20

1.7.1 Regression testing goals - view of practitioners and researchers 20

1.7.2 Using goal-question-metric (GQM) to compare research and practice perspectives on regression testing 21

1.7.3 Regression testing for large-scale embedded software development 21

1.7.4 On the search for industry-relevant regression testing research 22

1.7.5 Lessons learned from replicating a study on information-retrieval based test case prioritization 23

1.7.6 Checklists to support decision making in regression testing . . 24

1.8 Discussion 24

1.8.1 Regression testing state of practice 25

1.8.2 Supporting the adoption of regression testing research in practice 26
1.8.3 Providing support to improve regression testing practice . . . 27
1.9 Conclusions and future work 28
1.10 References 29

2 Regression testing goals - views of practitioners and researchers 35
2.1 Introduction 35
2.2 Related work 37
2.3 Methodology 37
2.3.1 Planning the research. 38
2.3.2 Designing the focus groups. 38
2.3.3 Conducting the focus group session. 39
2.3.4 Analyzing the data and reporting the results. 40
2.4 Threats to validity 41
2.5 Results and analysis 42
2.5.1 Defining regression testing. 42
2.5.2 GQM activity. 44
2.6 Conclusions 52
2.7 References 53

3 Using goal-question-metric to compare research and practice perspectives on regression testing 57
3.1 Introduction 57
3.2 Related work 60
3.3 Methodology 64
3.3.1 Literature review 65
3.3.2 Survey 68
3.3.3 Threats to validity 72
3.4 Results and analysis 74
3.4.1 Literature review 74
3.4.2 Survey 78
3.4.3 Using GQM to integrate research and practice perspectives . . 91
3.5 Discussion 93
3.6 Summary and conclusions 97
3.7 References 99

Appendix A Interview guide 107

4 Regression testing for large-scale embedded software development – exploring the state of practice 111

4.1 Introduction . . . 111
4.2 Related work . . . 112
4.3 Methodology . . . 120
4.3.1 Research questions . . . 120
4.3.2 Case companies . . . 120
4.3.3 Data collection . . . 122
4.3.4 Interpreting, analyzing and validating interview transcripts . . 124
4.4 Threats to validity . . . 126
4.5 Results and discussion . . . 127
4.5.1 The Practitioners' perceptions of regression testing (RQ1) . . 127
4.5.2 Regression testing practices (RQ2) . . . 128
4.5.3 Suggested improvements for regression testing (RQ3) 136
4.5.4 Goals and criteria for successful regression testing (RQ4) . . . 138
4.6 Summary and conclusions . . . 140
4.7 References . . . 143

Appendix B Interview guide 147

5 On the search for industry-relevant regression testing research 151

5.1 Introduction . . . 151
5.2 Related work . . . 153
5.2.1 Evaluation of the industry relevance of research . . . 153
5.2.2 Reviews of regression testing research . . . 154
5.3 Research questions . . . 156
5.4 Method . . . 156
5.4.1 Practitioners' involvement . . . 157
5.4.2 Need for a literature review . . . 159
5.4.3 Pilot study . . . 159
5.4.4 Search strategy . . . 160
5.4.5 Selection of papers to include in the review . . . 161
5.4.6 Taxonomy extension . . . 163
5.4.7 Taxonomy evaluation . . . 164
5.4.8 Mapping of techniques to taxonomy . . . 165
5.5 Limitations . . . 165
5.5.1 Coverage of regression testing techniques: . . . 165
5.5.2 Confidence in taxonomy building process and outcome 166
5.5.3 Accuracy of the mapping of techniques and challenges 167

5.6 RQ1 – Regression testing problem description 167
5.6.1 Investigated context factors 168
5.6.2 Desired effects 171
5.7 RQ2 – Regression testing solution description in terms of utilised information sources 173
5.7.1 Requirements 175
5.7.2 Design artefacts 175
5.7.3 Source code 176
5.7.4 Intermediate and binary code 176
5.7.5 Issues 177
5.7.6 Test cases 177
5.7.7 Test executions 177
5.7.8 Test reports 178
5.8 RQ3 – Mapping of current research 178
5.8.1 Addressed context factors 178
5.8.2 Desired effects 180
5.8.3 Information sources 180
5.9 Suggestions for practitioners 180
5.10 Recommendations for researchers 181
5.11 Conclusion 183
5.12 References 185

6 Lessons learned from replicating a study on information-retrieval based test case prioritization 195
6.1 Introduction 195
6.2 Background 198
6.2.1 Regression testing 198
6.2.2 Replication 199
6.2.3 Information retrieval 199
6.3 Related work 200
6.4 Methodology 202
6.4.1 Research questions 202
6.4.2 Information about the original study 203
6.4.3 Information about the replication 206
6.4.4 Analysis of the replication results 211
6.4.5 Automation of replication 211
6.4.6 Threats to validity 213
6.5 Results 214
6.5.1 RQ1. Degree to which the replication is feasible to implement. 214

6.5.2 RQ2. Comparison of the results to the original study. 218
6.6 Discussion 220
6.6.1 Lessons learned of replicating artefact-based studies in software testing 220
6.6.2 General lessons learned for artefact-based replications 224
6.7 Conclusions 226
6.8 References 227

7 Checklists to support decision making in regression testing 233
7.1 Introduction 233
7.2 Background and related work 234
7.2.1 Significance of checklists 234
7.2.2 Use of checklists in software engineering 235
7.2.3 Summary 237
7.3 Methodology 237
7.3.1 Research approach 238
7.3.2 Selection of participants 238
7.3.3 Study participants 239
7.3.4 Data collection 239
7.3.5 Data analysis 242
7.4 Validity threats 245
7.5 Results 246
7.5.1 State of regression testing practice in companies 246
7.5.2 Regression testing activities **(RQ1)** 247
7.5.3 Practitioners' opinion on prospective regression testing checklists **(RQ2)** 249
7.5.4 Checklists evaluation **(RQ3)** 255
7.6 Discussion 255
7.6.1 Regression testing activities 256
7.6.2 Checklists creation and evolution 257
7.6.3 Checklists evaluation 258
7.6.4 Implications 258
7.7 Conclusion 260
7.8 References 261

Appendix C 265
C.1 State of regression testing practice in the case companies 265
C.2 Introductory workshop 270
C.3 Interview guide 272

C.4 Checklist evolution 273
C.5 Checklist evaluation 274

List of Figures

1.1 Thesis overview: research questions, chapters, and contributions (C1 – C6) 14

2.1 GQM representation 44
2.2 Goal-question-measure mapping 49

3.1 Role and testing experience of the survey respondents 87
3.2 Software development domains on which survey respondents are working 88
3.3 Regression testing goals – questionnaire results 88
3.4 A GQM based model of regression testing goals, information needs, and metrics. 92

4.1 Mind-map used for data management and classification. 123
4.2 Regression testing process in the case companies 129
4.3 Relationship between RT challenges. 135

5.1 A description of the flow of activities including alternately reviewing the literature and interacting with practitioners from the research project EASE. 158

6.1 Steps followed to replicate the original study 207
6.2 Steps to automate the replication of IRCOV 212
6.3 APFD boxplots for IRCOV Original vs IRCOV replication IRCBO= IRCOV branch coverage original, IRCBR= IRCOV branch coverage replication IRCLO= IRCOV line coverage original, IRCLR= IRCOV line coverage replication IRCMO= IRCOV method coverage original, IRCMR=IRCOV method coverage replication 219

7.1 Overview of the approach used to design and evolve the regression testing (RT) checklists. 238
7.2 Data analysis steps. 243
7.3 Mapping regression testing activities to checklist items 245
7.4 Evaluation feedback from the participants on the final version of regression testing checklists. 256

List of Tables

1.1 Research questions and thesis chapters 11
1.2 Research methods and thesis chapters 13
1.3 Mapping of regression testing techniques for embedded systems in taxonomies 27

2.1 Focus group participants 39
2.2 GQM-Template for evaluation of regression testing 40
2.3 Defining regression testing 42
2.4 Regression testing goals 46
2.5 Allocated priorities to the goals 47
2.6 G5. Questions (Information needs) 48
2.7 Measures 49
2.8 Measures found in literature 51

3.1 A summary of how our current study extends our previous work [4] . 59
3.2 Research questions 64
3.3 Snowball iterations 66
3.4 Inclusion and exclusion criteria 67
3.5 Data extraction form 67
3.6 GQM mapping of regression testing goals, information needs, and metrics – Literature 75
3.7 Interview participants 79
3.8 Regression testing goals – interviews results I-1 to I-11 are the practitioners' IDs and (✓) means that the goal was defined by the respective practitioners. 81
3.9 Literature metrics of regression testing, recognized by the interview participants 86

3.10 GQM mapping of goals, information needs, and metrics – Survey . . 90

4.1 Summary of related work. The first column indicates the subsection in Section 4.2 (GTP: General Testing Practices, TMT: Testing Methods and Tools, AT: Automated Testing, RT: Regression Testing). 117
4.2 Literature findings on RT state of practice 119
4.3 Overview of interviewees. 121
4.4 Analysis procedure adopted for step 2 and step 3. 125
4.5 The practitioners' definitions of regression testing, as response to interview question "What is regression testing for you?". 128
4.6 Regression testing practices. 130
4.7 Test selection and prioritization criteria. 132
4.8 Information sources utilized for test selection and prioritization. . . . 132
4.9 Regression testing challenges. 134
4.10 Identified improvements. 137
4.11 Regression testing success goals. 138
4.12 Regression testing success criteria. 139

5.1 Systematic literature studies used as start-set for snowball sampling . 160
5.2 The list of papers included in this study 163
5.3 Data extraction form 164
5.4 A taxonomy of context, effect and information factors addressed in the included papers and considered relevant by our industry partners . . . 169
5.5 Mapping of techniques to the taxonomy 179

6.1 Software programs used in replication 208
6.2 Simulation parameters for Commons-CLI. (MC = Method coverage, LC = Line coverage, & BC = Branch coverage) 215
6.3 APFD values for all ten folds of each coverage type 218
6.4 Statistical results of replication compared to the original study for Commons-CLI. 219
6.5 Comparison of mutant generators 222
6.6 Recommendations drawn from the challenges/lessons learned 224

7.1 List of practitioners who participated in checklist design and evolution. 240
7.2 Contextual information of the companies represented by the participants (size classification: *"small < 50, large > 250"* [39]). 241
7.3 Structure of transcription sheet – RT (regression testing). 244

7.4 Activities considered essential for regression testing (RT) in the companies. 248
7.5 Types of regression testing (RT) checklists suggested by participants. . 250
7.6 Checklist to assess the readiness of testers to be filled by test team members ($CL_1 - V_{1.0}$). 251
7.7 Checklist to determine the team 's readiness to be filled by test manger ($CL_2 - V_{1.0}$). 252
7.8 Checklist to determine exit criteria of regression testing to be filled by test manager together with team members ($CL_3 - V_{1.0}$). 252
7.9 Evolution of checklist to know the readiness of testers to be filled by test team members ($CL_1 - V_{1.1}$). 253
7.10 Evolution of checklist to know the readiness of test team to be filled by test manager ($CL_2 - V_{1.1}$). 254
7.11 Evolution of checklist to determine exit criteria of regression testing (RT) to be filled by test manager together with team members ($CL_3 - V_{1.1}$). 254
7.12 Regression testing activities identified from selected studies 257

C.1 State of regression testing (RT) in the case companies. 267
C.2 Regression testing activities considered essential in the case companies. 269

Chapter 1

Introduction

1.1 Overview

Software testing is a complex and costly activity and consumes up to 50% of the total software development cost [1, 3, 4, 7]. The most challenging activity for large-scale software development with continuous integration and delivery is regression testing, which consumes a significant share of the total testing cost [1, 17, 28, 31]. Regression testing is performed after changes in the system or between release cycles. The goal is to provide confidence that changes have not impacted the unchanged parts of the system negatively [2, 32]. Various approaches are used to perform regression testing, including running all test cases (retest-all), running a selected set of test cases (test case selection), and running test cases in order of priority (test case prioritization) [2]. It is possible to adopt a retest-all approach for smaller projects, where test suites have fewer test cases. Contrarily, in large-scale software development with continuous integration and delivery, it is costly to opt for the retest-all approach because test suites are large and can consist of thousands of test cases [7]. Test case selection and prioritization techniques attempt to cope with the larger test suites. The primary purpose of these techniques is to reduce the execution cost of regression testing [55]. However, selecting suitable test cases or choosing an appropriate order is a complex decision-making problem.

Many regression testing techniques have been proposed in research, but a few have been adopted in industry [2, 17]. The lack of evaluation of the proposed techniques in industrial context is one of the reasons [5]. Researchers' primary challenge is proposing regression testing techniques that can overcome challenges, fulfil practitioners' goals, and fit in an industry context [1]. The primary step in this regard is to understand the regression testing practice and identify the regression testing challenges that practitioners face. We need to introduce regression testing research to industry practitioners. In this regard, we can work on possible alternatives, such as identifying regression testing techniques capable of coping with practitioners' challenges and relevant to an industry context, finding ways to evaluate and adopt regression testing techniques in the industry and supporting practitioners in planning to perform regression testing.

We initiated this work as a part of an industry-academia collaboration (IAC) project EASE[1] aiming to reduce the industry-academia gap in software testing. Together with testing practitioners from the platform of EASE, we identified software testing challenges in the large-scale development companies operating in communication, surveillance, and embedded software systems. The identified challenges were mainly related to regression testing. We have conducted two studies to understand regression testing practice, challenges, and goals. The results of the first study are based on a focus group of practitioners and researchers, and the second study presents the results of a multi-case study with two large companies. Later, we compared the research and practice perspectives of regression testing using the results of a literature review and a survey. We introduced a GQM (goal-question-metric) model by integrating the literature and survey findings to map the regression testing goals, information needs, and measures. We also have created a three-facet taxonomy using the results of a systematic literature review to facilitate practitioners finding regression testing research relevant to their needs. The goal was to enable the practitioners to determine which regression testing techniques could be applicable in a given industrial context and to learn the cost implication of adopting these techniques. To further support the adoption of regression testing research, we have conducted a replication experiment to replicate a test case prioritization technique. In this thesis, we have reported the lessons learned during the replication experiment, which have implications for research and practice. Practitioners can use these lessons to replicate regression testing techniques in their product contexts. Researchers can use them to report required aspects when publishing new or replicated regression testing techniques.

Lastly, with the input of 25 senior testing practitioners from 12 companies, we developed regression testing checklists to help streamline regression testing activities and improve regression testing processes. The participating practitioners evaluated these checklists and reported that the checklists cover essential aspects of regression testing. Furthermore, they reported that checklists are useful and customizable according to their product context. The contributions of the studies included in this thesis are:

[1]EASE – the Industrial Excellence Centre for Embedded Applications Software Engineering `http://ease.cs.lth.se/about/`

- Identification of regression testing goals.
- Mapping research and practice perspectives of regression testing using a goal-question-metric model.
- Understanding regression testing practice and identifying challenges.
- Taxonomies to facilitate practitioners finding regression testing techniques relevant to the industrial context.
- Based on the results of a replication experiment, a step-wise guideline to support future replication and adoption of regression testing techniques.
- Regression testing checklists to support practitioners in decision making and streamlining regression testing practice.

The rest of this chapter is organized as follows: Section 1.2 presents a brief description of the concepts used in the thesis. Research gaps and contributions of this work are discussed in Section 1.3, whereas Section 1.4 describes the research questions that have been investigated in this work. Section 1.5 provides a brief discussion of the methods used in the studies of this thesis. Threats to the validity of studies included in this thesis are discussed in Section 1.6, and summaries of the included studies are presented in Section 1.7. Section 1.8 provides a brief discussion on the findings of this thesis, and Section 1.9 concludes the chapter and discusses future work.

While the remainder of the thesis is organized as follows: Chapter 2 presents the results of a focus-group-based study on regression testing goals. Based on the literature and survey findings, Chapter 3 provides a goal-question-metric mapping of regression testing goals to compare the regression testing research and practice. As a result of a multi-case study, Chapter 4 presents the state of regression testing practice and its challenges. Chapter 5 proposes a three-facet taxonomy of regression testing to support industry adoption of regression testing research. Chapter 6 presents the results of a replication experiment conducted to replicate a test case prioritization technique. Finally, Chapter 7 proposes regression testing checklists to structure and improve regression testing practice.

1.2 Background

This section presents a brief description of the basic concepts that have been used in this thesis.

1.2.1 Regression testing

Regression testing is applied to a system under test after any change, including a defect fix or adding a new feature. The IEEE Systems and Software Engineering Vocabulary defines regression testing as:

"The selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements" [6].

The goals of regression testing are to find defects and to obtain confidence about the quality of the systems under test. Running all test cases in the regression suite (retest all) or running a subset of test cases in the regression suite (selective regression testing) are the two alternatives for regression testing. Selective regression testing (SRT) is used instead of re-test all when test suites are significantly large. SRT refers to running regression testing with a smaller scope, where the critical concern in this regard is the size of the test suite [7]. Practitioners prefer to run regression tests with a selected scope. The primary challenge for a testing practitioner is to determine the scope of regression testing (i.e. which tests to include in the regression test suite) [1, 7]. Techniques used for SRT are test case selection, prioritization, and test suite minimization [2, 8].

Test case selection Researchers use the terms test case selection or regression test selection interchangeably. The concept refers to selecting a subset of test cases from existing test suites to test the modified parts of the system under test (SUT). Test case selection techniques are modification aware, as only those test cases are selected which are relevant to the modified parts of the SUT [2, 20].

Test case prioritization Test case prioritization refers to the reordering of test cases in an existing test suite, to ensure to run the most critical test cases at priority. The goal of test case prioritization is to maximize the fault detection rate. Test case prioritization does not involve any procedure to select a test case; it only reorders the test cases on the basis of some predefined criteria [2, 27].

Test case minimization For large-scale systems, test suites can be very large, and grow further when the software undergoes changes. New test cases can be added for new or updated features. Continuously growing test suites can contain redundant and obsolete test cases. A test case becomes obsolete, if the requirement associated to it is removed or updated during the changes in the software. Adding of new test cases can cause some existing test cases to become redundant given that new test cases satisfy the same testing requirements. Removal of redundant test cases does not effect the testing coverage [30]. The presence of redundant and obsolete test cases in a test suite increases the costs for test execution. Such test cases need to be identified and eliminated. Test case minimization techniques are meant to reduce the size of the test suites by eliminating redundant and/or obsolete test cases [2, 19].

1.2.2 Regression testing in practice

There is a gap between research and practice on regression testing. Researchers are focusing on proposing new regression testing techniques. In contrast, the practitioners are not using these techniques. Instead, they rely on their expertise and experience [1]. In order to tailor regression testing research to industry needs, it is important to know the industry practices related to regression testing. A few authors have investigated the state of regression testing in practice. However, the investigations are limited to a specific aspect of regression testing. For instance, Parsons et al. [58] conducted a survey-based study to investigate the adoption of regression testing strategies in agile development. Similarly, Juergens et al. [59] carried out an industry case study highlighting the challenges concerning test case selection techniques when applied in manual system tests. Engström and Runeson [1] investigated regression testing practices and challenges and pointed out that a majority of the challenges are related to testability issues, and good practices are related to test automation. The authors suggested that researchers must better understand the practitioners' needs and practices to tailor regression testing research to industry needs.

In this thesis, we conducted a study (study 3) focusing on understanding the industry's regression testing practice. We have conducted this study with a broader scope, and besides understanding regression testing practices, we have investigated the challenges that practitioners face during regression testing. We have also discussed the potential improvements in regression testing practice.

1.2.3 Goal question metric (GQM) approach

A goal question metric (GQM) approach advocates a goal-oriented measurement framework [56]. A GQM model has three levels 1) conceptual level (Goals), 2) operational level (Questions), and quantitative level (Metrics) [57]. It represents a hierarchical structure, and the elements of a GQM are defined in a top-down manner. First, the goals are defined, and then questions are formulated whose answers guide towards achieving goals. Finally, the metrics provide actual measurement and determine whether the goals are achieved [56, 57]. While making a GQM plan, it is essential to make clear what information needs are available, as the operational level stems from informational needs that provide the basis for quantification [57].

We incorporated the GQM approach in our two studies (Study 1 & 2). Study 1 was conducted using a focus group, and we structured the focus group discussion using the GQM approach. Later, we created a mapping of a chosen regression testing goal with information needs and metrics. In Study 2, we created a GQM model to map the regression testing goals identified from research and practice with the corresponding questions and metrics.

1.2.4 Replication

Replication is a means to validate experimental results and examine if the results are reproducible [34]. Replications are essential for solidifying knowledge and validating if the original experiment results are authentic and generalizable. In principle, replication provides a way forward to create, evolve, break, and replace theoretical paradigms [35, 36]. During the previous four decades, software engineering researchers have built new knowledge and proposed new solutions. Many of these lack consolidation [35]. Replication studies can help in establishing the solutions and expanding the knowledge. Software engineering researchers have been working on replication studies since the 1990s. Still, the number of replicated studies is small, and a more neglected area is the replication of software testing experiments [35, 38, 40, 41]. Most software engineering replication studies are conducted for experiments involving human participants; few replications exist for artefact-based experiments [38].

Replication could be of two types 1) internal replication –a replication study carried out by the authors of the original study themselves, 2) external replication –a replication study carried out by researchers other than the authors of the original study [35, 37].

In software engineering research, the number of internal replications is much higher than external replications [38, 39]. The results of 82% of the internal replications are confirmatory, and the results of 26% of external replications conform to the original studies [38]. From the empirical software engineering perspective, Shull et al. [36] classify replications as exact and conceptual replication. In an exact replication, the replicators closely follow the procedures of the original experiment, whereas, in a conceptual replication, the research questions of the original study are evaluated using a different experimental setup. If the replicators keep the conditions in the replication experiment the same as the actual experiment, it would be categorized as exact dependent replication. If replicators deliberately change the underlying conditions of the original experiment, it would be referred to as exact independent replication. Exact dependent and exact independent replications could be mapped to strict and differentiated replications, respectively. A strict replication compels the researchers to replicate a prior study as precisely as possible. In contrast, in a differentiated replication, researchers could intentionally alter the aspects of a previous study to test the limits of the study's conclusions.

In this thesis, we conducted an exact independent (external) replication (Study 5) of a test case prioritization technique to explore the possibility of adopting regression testing techniques.

1.2.5 Checklists

A checklist is a standardized tool that enlists the required process criteria for the practitioners performing a specific activity. It provides support in recording the presence or absence of the essential process tasks [42]. Two popular uses of checklists are, using checklists as mnemonic systems or as evaluation tools. The first is used as a reminder system to help practitioners avoid omitting any essential task. It also assures that practitioners follow the organizational framework and utilize best practices. Such checklists help minimize human error and improve the overall performance. In contrast, the evaluative checklists can aid in the standardization of evaluation by providing assessment guidelines and ultimately improving the evaluation process's credibility [42].

Using a checklist to aid any process is not a new concept. For example, in the aviation industry since the 1930s, it has been a standard operating procedure for the pilots and other aviators to use checklists [43]. Pilots are using the checklists before, during, and after the flight [44]. In medicine, checklists are used as a decision aid to identify a medical condition and decide on an appropriate course of treatment. In comparison, surgical checklists are recommended as a safety measure to reduce the margin of human error and any adverse effects during surgery [45].

Social and behavioral scientists are using self-reporting questionnaires as an assessment mechanism. Usually, such questionnaires include checklist items that enable the goal-based assessment of a phenomenon [46].

Software engineers are using checklists in various tasks, including the audit of requirement/design specifications and code inspection [47]. Checklists can help make a process repeatable, and the practitioners can use various checklists in the software development life cycle. For instance, they can use a release checklist to assure that no essential step is missed out. At the start, a checklist does not have to be exhaustive. If some items are missing, still, checklists are helpful, and we can add the missing or new items later. Improvising a checklist is always helpful in adding future goals [48].

The final study of this thesis (Study 6) presents a regression testing checklist, with the aim of assisting practitioners in decision making.

1.2.6 Industry academia collaboration

Industry-academia collaboration (IAC) has been a widely discussed topic in the software engineering community since the early days of software engineering. Practitioners can benefit from IAC by improving various practices, and researchers can benefit from it by producing industry-relevant research. Researchers need to work on different mechanisms to communicate their research to practitioners. To improve the communication between software testing researchers and practitioners, Engström et al. [15] proposed a four-faceted taxonomy (SERP test). It can help identify an intervention with the desired outcome in a specialized testing area for a given context. The taxonomy is helpful as a framework for direct communication (industry-academia collaboration) and indirect communication (reporting research results to the industry). Success stories regarding IAC in the field of software engineering are limited [26]. Garousi and Felderer [25] argue that differences in objectives, less scientific value in industry-related problems and limited applicability of methods proposed in academia are hindrances to industry-academia collaboration. The communication gap between researchers and practitioners is one of the critical challenges that can lower the motivation for collaboration.

The studies included in this thesis are the outcome of industry-academia collaboration. We attempted to address some of the mentioned issues of IAC. For example, regression testing taxonomies presented in Chapter 5 attempt to reduce the communication gap between research and practice of regression testing and improve the applicability of regression testing research. The goal-question-metric model presented in Chapter 3 is a step forward to aligning research and practice goals of regression testing.

1.3 Research gaps and contributions

Regression testing is a well-researched area, however, the adoption of the proposed techniques in industry is limited [33]. The possible reasons for this are 1) lack of empirical evaluation of the techniques in industry [5, 33], 2) the differences in goals, and 3) cognitive gap between research and practice [8, 15, 16].

This thesis aims at reducing the industry-academia gap in regression testing. The focus is 1) to understand practitioners' needs, challenges, and goals, 2) to map the solutions proposed in research to industry context and goals, and 3) to provide support to improve the regression testing process in industry.

In the following, along with the identified gaps, we present the contributions of this thesis.

Gap1 There is a disparity in regression testing research and practice, and software engineering researchers need to understand the industry needs and practices of regression testing [1, 16]. While working on Study 1, we identified a disparity in perceptions of regression testing goals between practitioners and researchers. Study 2 also points to differences in research and practice perspectives of regression testing.

Contribution 1 Study 1 highlights differences in the perspectives of researchers and practitioners regarding the goals of regression testing. It also provides a set of jointly prioritised regression testing goals by researchers and practitioners.

Contribution 2 Using the GQM model, Study 2 maps regression testing goals, information needs, and metrics to compare research and practice perspectives of regression testing and reduce the differences between the two perspectives.

Gap2 To our knowledge, not many studies identify the state of regression testing practice, the practitioners' goals, and the challenges they face during the regression testing activity. Only a few authors have investigated the state of regression testing in practice [1, 58, 59]. However, the investigations are limited to a specific aspect of regression testing. We identified this gap while searching for industry-related regression testing research for our systematic literature review (Study 4).

Contribution 3 In the context of large-scale software development, study 3 presents the state of regression testing practice, the practitioners' goals, and the challenges they faced.

Gap3 Despite the extensive body of research literature, research results have shown to be hard to adopt for the practitioners [2, 17]. It is hard for them to know what to search from the literature, as studies presenting techniques do not include the context information. We identified this gap while interacting with the practitioners for our studies and while analyzing the industry-relevant research for study 4.

Contribution 4 Study 4 presents a taxonomy to facilitate the industrial adoption of regression testing research by addressing the attributes of concern from the practitioners' perspective.

Contribution 5 Based on the lessons learned by replicating a test case prioritization technique, Study 5 presents guidelines to support the replication and industry adoption of regression testing techniques.

Gap 4 Regression testing practice lacks a well-defined structure and mainly relies on expert judgments [1]. The various challenges identified in Study 3 stem from the lack of structure in regression testing practice. We also observed this fact while working on studies 1 and 2.

Contribution 6 To support practitioners in decision-making and improve regression testing practice, study 6 presents regression testing checklists designed, evolved, and evaluated by involving senior testing practitioners.

1.4 Research questions

This thesis aims to understand the industry needs for regression testing, find the relevant solutions in the literature, and provide support for regression testing practice. The work can be distributed into three segments:

1. Understanding the state of regression testing practice with a focus on goals and challenges.
2. Exploring the state of research with a focus on applicability in practice.
3. Providing support to the practitioners in structuring and improving the regression testing practice.

Table 1.1 summarizes the mapping of chapters and research questions. The focus of RQ1 is understanding the state of regression testing practice, which is further divided into RQ1.1 and RQ1.2. RQ2 explores the possibilities for the adoption of regression testing research, and this question is divided further into RQ2.1 and RQ2.2. The purpose of RQ3 is to investigate the ways to provide support in regression testing practice. The following provides the detail of RQs.

RQ1 *What is the state of regression testing practice in industry?*

Table 1.1: Research questions and thesis chapters

	Chapters					
RQs	2	3	4	5	6	7
	RQ1.1					
		RQ1.1				
			RQ1.2			
		RQ2.1		RQ2.1		
					RQ2.2	
						RQ3

RQ1.1 *What are the goals considered essential for the success of regression testing?*
While implementing any strategy for regression testing, it is important to assess whether the chosen strategy is a good choice. A good strategy can be aligned with the success goals. The purpose here is to understand the regression testing goals and learn which information should be utilized to measure these goals. Based on the findings, the aim is to provide actionable guidelines for practitioners and researchers to align their goals. This can help in reducing industry-academia gap. This aspect is the focus of Chapters 2 and 3.

RQ1.2 *How is regression testing performed in large-scale software development companies?*
Researchers argue that there is a gap between regression testing research and practice. Researchers propose their techniques by utilizing different criteria and well-managed sources of information. Whereas in industry, the most crucial criteria is the experience of the practitioners involved, and the best source of information is their knowledge about the system under test. The goal here is to understand the practices that practitioners utilize during regression testing, how they select and/or prioritize the tests, what challenges practitioners face while running regression tests, how they address the challenges and how they measure the success of regression testing. These aspects are the consideration of Chapter 4.

RQ2 *How can regression testing research be utilized in practice?*

RQ2.1 *How to map regression testing research with industry context?*
There is a large body of research on regression testing, and researchers focus on three main areas: 1) test case selection, 2) test case prioritization,

and 3) test suite minimization. Only a few studies have been empirically evaluated on large-scale systems. The purpose of Chapter 5 is to identify and synthesize the industry-relevant research on regression testing to provide recommendations for future research and guidelines for practitioners regarding the choice of regression testing methods.

RQ2.2 *To what extent regression testing techniques proposed in the literature are replicable?*
Most regression testing techniques proposed in the research do not get the attention of practitioners. Practitioners do not know much about these techniques since most regression testing techniques have not been tested in the industry. These techniques need to be re-examined in the context of industry goals to ensure that regression testing techniques proposed in research are applicable in industry. Replication can be helpful in this regard. In Chapter 6, we experimented with replicating a test case prioritization technique and discussed the lessons learned.

RQ3 *How to support practitioners improve decision-making in the regression testing process?*
Study three lists the challenges practitioners face during regression testing activities. These challenges are of two kinds, one that requires technical support and the other where the process needs to be improved. In Chapter 7, together with practitioners, we have developed checklists to improve the regression testing process. These checklists aim to improve decision-making by streamlining the regression testing process and aligning it with practitioners' goals.

1.5 Research methods

Systematic literature reviews and various empirical methods have been utilized in this thesis. The empirical methods utilized in this thesis are 1) focus groups, 2) surveys, 3) case studies, and 4) experiments. Table 1.2 summarizes the methods used in different studies (chapters) It also shows the involvement of companies in the studies. Figure 1.1 describes an overall view of the thesis regarding a relationship between research questions, thesis chapters, research methods, and contributions.

1.5.1 Empirical methods

Empirical methods have been used in software engineering research since the 1960s.These methods have become powerful means to create scientific evidence on software devel-

Table 1.2: Research methods and thesis chapters

	Chapters						
Research method	2	3	4	5	6	7	Partner Companies
Focus Group	✓						Two
Case Study			✓				Two
Survey		✓				✓	Multiple
Experiment					✓		
Literature Review		✓					
SLR				✓			Two

opment, operation, and maintenance through various iterations. Besides researchers, software practitioners benefit from these methods in their decision making and learning [52]. In empirical methods, evidence is collected using qualitative and quantitative methods. Qualitative methods comprise interviews and participant observation, and quantitative methods include surveys, archival data, experimentation, and simulations [51]. Data collected through qualitative methods are subject to qualitative analysis (e.g., grounded theory, thematic analysis, narrative analysis, etc.). Quantitative data are analyzed using statistical methods and mathematical modelling based approaches [10, 51]. The following subsections provide a summary of empirical methods used in different studies of this thesis.

Focus group

A focus group is a convenient method for data collection from a group of individuals when it is required to obtain participants' viewpoints on a topic of shared interest. It allows people to sit together and have an open discussion about a specified topic [49]. The role of the moderator is crucial for the success of a focus group. A focus group should be conducted with at least three participants, and the suggested upper limit for a focus group is twelve participants [9].

Our first study (Chapter2) focused on understanding the perspectives of academics and practitioners on regression testing goals. The study's first objective was to identify the goals for the success of regression testing, and the second objective was to investigate whether there are any disparities in thoughts between researchers and practitioners. Senior researchers and practitioners participated in the study. The most suitable way was to allow them to sit together and voice their opinion on the topic. To achieve this objective, we utilized the focus group method.

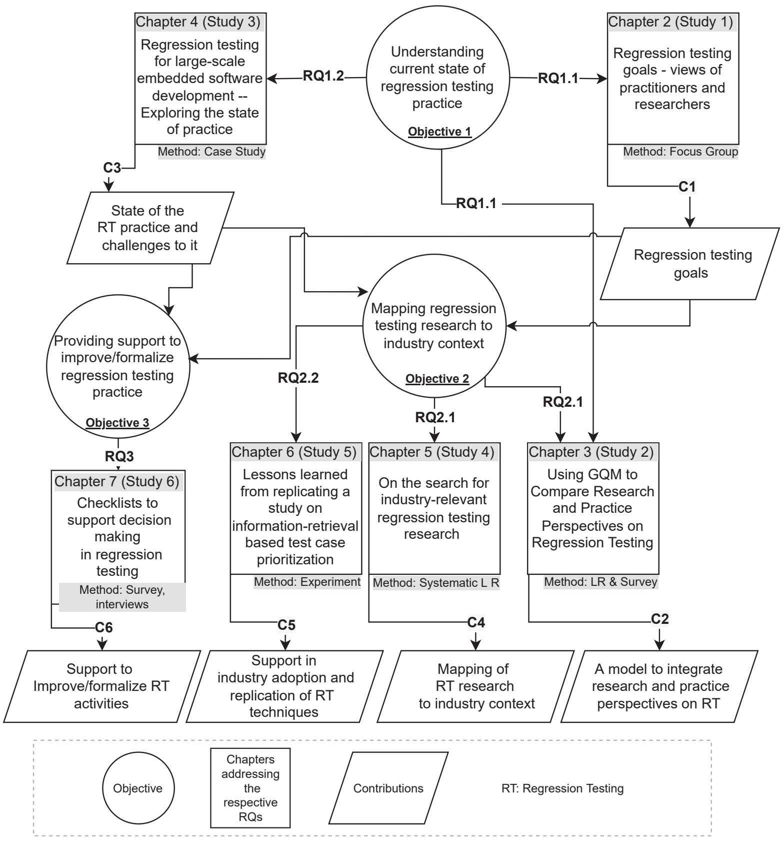
Chapter 4 (Study 3)
Regression testing for large-scale embedded software development -- Exploring the state of practice
Method: Case Study
Understanding current state of regression testing practice
Objective 1
Chapter 2 (Study 1)
Regression testing goals - views of practitioners and researchers
Method: Focus Group
RQ1.2
RQ1.1
C3
C1
RQ1.1
State of the RT practice and challenges to it
Regression testing goals
Mapping regression testing research to industry context
Objective 2
Providing support to improve/formalize regression testing practice
Objective 3
RQ2.2
RQ2.1
RQ2.1
RQ3
Chapter 7 (Study 6)
Checklists to support decision making in regression testing
Method: Survey, interviews
Chapter 6 (Study 5)
Lessons learned from replicating a study on information-retrieval based test case prioritization
Method: Experiment
Chapter 5 (Study 4)
On the search for industry-relevant regression testing research
Method: Systematic L R
Chapter 3 (Study 2)
Using GQM to Compare Research and Practice Perspectives on Regression Testing
Method: LR & Survey
C6
C5
C4
C2
Support to Improve/formalize RT activities
Support in industry adoption and replication of RT techniques
Mapping of RT research to industry context
A model to integrate research and practice perspectives on RT
Objective
Chapters addressing the respective RQs
Contributions
RT: Regression Testing

Figure 1.1: Thesis overview: research questions, chapters, and contributions (C1 – C6)

Survey

A survey helps to identify the characteristics of a larger population, and it is a suitable method where the aim is to collect the opinions of a large sample [49]. In Chapter 3 (Study 2), we were interested in knowing as many practitioners' perceptions of regression testing goals as possible. Simultaneously, we were keen to have some insight into the regression testing goals and other associated practices. Therefore, we decided to survey by opting for two data collection methods interviews and an online questionnaire. Interviews allowed us to interact with the practitioners and understand their perceptions directly. The online questionnaire helped us reach the broader population and collect the information about regression testing goals from a larger sample. We used the survey method in Chapter 7 (Study 6) to collect practitioners' feedback on the regression testing checklists.

Online Questionnaire: A questionnaire is a preferred method for data collection in a survey. It needs to be designed carefully and should be rooted in related literature. Before preparing a questionnaire, the researcher should be clear about the intended objectives of the survey. The questions need to be framed clearly and concisely, as unclear questions can irritate the respondents [50]. A questionnaire can be composed of open or close-ended questions. Both have their merits and demerits. Open-ended questions allow the respondents to provide their opinion according to their perceptions. However, with open-ended questions, there is always a chance of misinterpretation. Close-ended questions are easy to answer as the respondent has to select one option. Close-ended questions are the preferred option for an online questionnaire [50]. Besides the ordering of questions, the wording choice and provision of options have vital role to obtain the correct results [49]. In the open-ended questions, along with agree/disagree or yes/no, there should always be an option to facilitate the respondents to choose if they are undecided or have some other opinion [50].

The online questionnaire was used as one of the data collection methods for Study 2 (Chapter 3). We also used online questionnaires while evolving and verifying regression testing checklists in Study 6 (Chapter 7).

Case study

To address the second research question a multi-case study (Study 3) was conducted. Case studies enable the researcher to get an in-depth understanding of the phenomenon under investigation [10]. A case study could be exploratory, descriptive, or explanatory [11]. Mostly case studies are based on qualitative data and are meant to provide a deeper insight into some aspects of the study [10]. Yin [11] argues that a case study would be a suitable choice when: 1) research questions are based on how or why, 2) the researcher has no or little control over the behavioral events, and 3) the focus of the study is on contemporary events.

Regarding the data collection for case studies, Runeson and Höst [10] suggest to use multiple sources of data collection for a case study, since it will improve the validity of the results. The data collection methods adopted in the studies of this thesis are focus groups, interviews, and archival data.

Interviews: Interviews are vital tools for data collection in software engineering empirical studies. They allow the researcher to have direct interaction with the subjects and get a deeper insight into the phenomenon under study. Interviews can be classified into three classes: 1) unstructured, 2) semi-structured, and 3) fully structured. Conventionally, a semi-structured interview method is used in case studies [10] The researcher formulates the interview questions in semi-structured interviews and prepares a guide/plan to conduct the interviews. The order of asking questions is flexible in semi-structured interviews. The purpose is to be able to adjust the order according to the expertise/interest of the subject [10]. Interviews were the primary data collection method for Chapter 4 (Study 3), and these were part of the data collection methods for Chapter 3 (Study 2) and Chapter 7 (Study 6). We conducted semi-structured interviews for all the mentioned studies. The objective of these studies was to understand the current state of regression testing practice, regression testing goals, and factors considered significant for regression testing practice.

Archival data: Archival data refers to the procedural documents, historical data/metrics, and documentation related to management activities [10]. This type of data enables the researcher to understand the working environment of the case company, their processes, and previous history regarding the execution of the methods. Using archival data, a researcher can validate information obtained through interviews. Archival data was the secondary source of information for Study 3 (Chapter 4). The primary purpose was to validate some of the data points collected through interviews.

Experiments

Researchers use experiments to examine the causal relationship between different variables that characterize a phenomenon [53]. An experiment is used when we want to test the existing theories, test the accuracy of models, and validate measures. It can investigate the correctness of claims and can test the use of specific standards, methods, and tools for a given context. An experiment allows the experimenter to have certain control over the situation and systematically manipulate behaviour [51]. The experimenter can formulate and test a hypothesis about the phenomenon under study using an experiment. Based on the experiment results, the experimenter can decide to accept or reject the hypothesis [51, 52]. However, to have correct results of an experiment, it is required to prepare, analyze, and conduct an experiment correctly [51]. The starting point for the experiment is when we have an idea of a causal relationship. The process further proceeds with experiment scoping, planning, operation, analysis and interpretation, presentation, and concludes with reporting. A well planned, designed, and reported experiment provides a basis for replication [51].

Replication experiments: The replication of an experiment is meant to reconstruct the experiment design under similar conditions and potentially with new objects of study. The purpose of replication is to validate that the results produced in the original experiment are repeatable. The replication will be regarded as successful if the replicators can replicate both the design and results. However, in general, the results of most replications differ from the original study to some extent [51]. Replications are categorized as internal replication and external replication [37]. Internal replications are carried out by the authors of the original experiment, and external replications are carried out by the authors other than of original study [37]. We conducted an experiment to replicate a test case prioritization technique in Study 6 (Chapter 7).

1.5.2 Systematic literature review

A systematic literature review (SLR) provides a way to collect the research evidence in a systematic way [12]. This method was adopted in software engineering from medical research. With the help of this method, a researcher can collect, evaluate and interpret the available research relevant to a topic or phenomenon. The guidelines for conducting SLRs suggested by Kitchenham [13] propose three phases of a systematic review (i.e., planning, conducting, and 3 reporting the review).

Planning

The crucial phase in systematic reviews is the planning phase. As a part of planning, the researcher needs to establish the need for an SLR. In the planning phase, the researcher has to prepare a review protocol that includes research questions, search strategy, criteria for selecting primary studies, quality assessment criteria, data extraction, and synthesis strategy.

Conducting

After planning, the next step is to conduct the review. In this phase, a researcher follows the strategies defined in the planning phase to execute the review process. This phase consists of identifying relevant research from various research databases, selection of primary studies in the light of inclusion and exclusion criteria, quality assessment of the studies, data extraction, and data synthesis.

Reporting

Careful documentation of the results collected in the previous phases is an integral part of an SLR. Finally, these results are published in a report or an article.

To investigate RQ3, we conducted a systematic literature review of empirically evaluated regression testing techniques. The findings regarding RQ3 are presented in Chapter 5 (Study 3). Regarding the study selection process, a snowball sampling technique was followed [14].

1.6 Threats to validity

We have used various research methods to conduct studies in this thesis. Every study has its limitations. Therefore, every individual study could have unique threats to its validity. We have discussed validity threats for all our studies in the respective chapters (i.e., Chapter 2 – Chapter 7). Here we summarise how we mitigated the threats to validity in various studies.

Construct Validity This aspect of validity is regarding the underlying operational measures, concepts, and terms of the study. Four of our studies, Study 1, 2, 3, and 6, presented in Chapters 2, 3, 4, and 7, are exploratory and involve humans as study subjects. In these studies, the potential threats to construct validity could be the selection of relevant participants and the appropriate design of data collection instruments. For selecting participants, we followed non-random (snowball) sampling techniques. While designing the data collection instruments, we followed the guidelines by Runeson and Höst [10] to design interview instruments and Kitchenham and Pfleeger [50] to design the survey questionnaires. To avoid inconsistency and bias, we performed pretests and involved independent reviewers to review the instruments. Study 4, presented in Chapter 5, and part of Study 2, presented in Chapter 3, are based on literature reviews. Threats to construct validity in these studies could be the selection of relevant literature. We followed a snowball sampling search strategy [14]. Snowball sampling has been effectively used to conduct [60] and extend [61] systematic literature reviews. Study 5, presented in Chapter 6, is a replication experiment. We followed the philosophy of exact replication [36]. Therefore, if the original study suffers from any aspects of construct validity, the replication may do so. However, we took various measures to avoid researcher bias in this study's constructs, such as using automated tools and random generators for mutant generation and selection.

Internal Validity Internal validity threats mainly deal with the credibility (data collection, sample selection) of the study, i.e., whether the obtained results are valid or not. It refers to the factors that affect the outcome of the research. In the case of our exploratory studies (i.e., Chapters 2, 3,4, and 7), the critical aspect could be if we get responses from relevant people. Especially for questionnaire-based surveys, this aspect is critical, where the researcher has no control over the respondents. To mitigate this threat, we provided a good introduction and added experience-related constraints in the survey questionnaires. Another aspect that can impact the outcome is the interpretation and analysis of data collected in surveys and interviews. We involved multiple researchers for data interpretation and analysis. For data collected during the interviews, we validated our interpretations from the participants. Concerning the literature-based studies (Chapter 5 and part of Chapter 3), we took several steps to limit the chances of missing relevant literature, such as following a systematic study selection process, performing pilot selection on a randomly selected subset of papers, involving multiple researchers in the study selection process, and peer-reviewing of excluded papers. In the replication experiment presented in Chapter 6, we took various measures to mitigate threats to internal validity. For instance, to reduce the researcher's control over the variables that could impact the experiment's outcome, we avoided human interaction and used automated tools for data (mutants) generation and selection.

External Validity This aspect of validity refers to the generalization of findings. Concerning the studies presented in Chapters 2 and 4, the findings are not generalizable since these represent specific contexts. However, we have provided the detail of participants of Study 1 (Chapter 2) and teams that participated in the multi-case study (Chapter 4) to facilitate analytical generalization of results. Concerning the survey-based studies presented in Chapter 3 and Chapter 7, we tried to involve participants with diverse backgrounds working in different environments. However, we can not claim that findings apply to all development contexts. To overcome this limitation, we have provided the contextual detail of the participants, their teams and companies. Although we attempted to include the maximum possible relevant studies in our literature-based studies (Chapter 5 and part of Chapter 2), we cannot claim the exhaustive searches. We have provided the research questions and search mechanisms in both studies that could help the researchers to extend the findings of these studies. Concerning the replication experiment presented in Chapter 6, the software programs used are small and medium-sized Java programs. Therefore, we can not claim the generalizability of results to large-scale industrial projects.

Reliability This aspect concerns the extent to which the data and analysis depend on the specific researchers. The results are reliable if they are free of biases, and independent researchers can reproduce them using similar methods. We attempted to minimize its impact by involving multiple researchers during the planning, execution, analysis, and reporting of the studies included in this thesis. We have explained all aspects of data collection, analysis, and reporting in the respective studies. Furthermore, data collection instruments are made available in the studies. In all exploratory studies (i.e. Chapters 2, 3, 4, and 7), we ensured triangulation by involving multiple authors in interpreting the results, and the participants reviewed and validated all the results. In SLR (Chapter 5), we tested the data extraction form on a sample of papers. This helped to develop a shared understanding of the form. Furthermore, to increase the reliability of the study, the actual data extraction (from all selected papers) and the formulation of facets in the taxonomies were reviewed by two additional reviewers (authors of the paper).

1.7 Summary of studies included in the thesis

The studies conducted in this thesis have a certain level of interdependence on each other. For instance, study 1 motivated us to expand its findings. As a result, we conducted study 2. Similarly, the results of studies 1 and 2 provided the reasons to conduct study 3 to understand regression testing practice in the industry. The challenges identified in study 3 motivated us to conduct all subsequent studies (study 4, study 5, and study6). The following subsections briefly summarise each study included in this thesis.

1.7.1 Regression testing goals - view of practitioners and researchers

In this study, we aimed to elicit the views of industry and academia testing experts. We conducted a focus group study with seven experts who were the representatives of two large companies and two universities. We structured the focus group discussion using the GQM approach and divided it into three phases. We elicited goals, then mapped the goals to questions, and then worked on the measures to evaluate goals. We identified a prioritized list regression testing goals, *“Confidence”* was marked as the highest priority regression testing goal. Other goals identified in this study are, *“High precision”*, *“Fault slippage to the customer”*, *“Efficiency”*, and *“Inclusiveness”*. We also identified the information needs to be required to evaluate the success in regression testing in terms of “*Confidence*”. Finally, we elicited the measures corresponding to the identified information needs. We observed a few similarities in the views of the practitioners and researchers about the definition of regression testing goals. However, both perspectives differ in the priorities of these goals. From the results of this study, we can conclude that there is a gap in the industry and academic perceptions of regression testing. We conducted this study with a limited scope. Therefore, we cannot say that these findings apply to different software development contexts.

1.7.2 Using goal-question-metric (GQM) to compare research and practice perspectives on regression testing

In the second study, we intended to extend the findings of Study 1 by adding the perception of more practitioners. Therefore, we conducted a survey by using interviews and questionnaires. A total of 56 practitioners from multiple companies representing different software development domains participated in the study. We conducted a literature review of 44 relevant research papers to know the research perspective on regression testing goals.

We identified that industry and research highlight different regression testing goals. The literature emphasizes increasing the fault detection rates of test suite, whereas the focus in practice is on test suite maintenance, controlled fault slippage, and awareness of changes. Similarly, the literature suggests maintaining information needs from test case execution histories to evaluate regression testing techniques based on various metrics, whereas, at large, the practitioners do not use the metrics suggested in the literature.

The outcome of this study is GQM-based mapping of regression testing goals, information needs, and measures, representing both perspectives (industry and research). The GQM model can serve as a tool to bridge the gap between industry and academia. It can guide researchers in proposing new techniques closer to industry contexts by considering the practitioners' goals for regression testing. Practitioners can benefit from information needs and metrics presented in the literature and can use GQM as a tool to follow their regression testing goals.

1.7.3 Regression testing for large-scale embedded software development

Most of the regression testing techniques proposed in literature do not get the attention of the practitioners. Among the reasons could be that researchers do not understand practitioners needs and perspectives on regression testing [5, 29]. Researchers need to understand the essential aspects of how regression testing is performed in industry and the challenges practitioners face during the regression testing activity. This could help researchers propose regression testing techniques closer to industry context and ultimately improve the adoption rate of proposed techniques in industry.

This study aims at exploring the regression testing state of practice in the large-scale embedded software development. The study has two objectives, 1) to highlight the potential challenges in practice, and 2) to identify the industry-relevant research areas regarding regression testing. We conducted a qualitative study in two large-scale embedded software development companies, where we carried out semi-structured interviews with representatives from five software testing teams. We did conduct the detailed review of the process documentation of the companies to complement/validate the findings of the interviews.

We found that mostly, the practitioners run regression testing with a selected scope, the selection of scope depends upon the size, complexity, and location of the change. Test cases are prioritized on the basis of risk and critical functionality. The practitioners rely on their knowledge and experience for the decision making regarding selection and prioritization of test cases. The companies are using both automated and manual regression testing, and mainly they rely on in-house developed tools for test automation. The challenges identified in the companies are: time to test, information management, test suite maintenance, lack of communication, test selection/prioritization, lack of strategy, lack of assessment, etc. Majority challenges identified in the study are management related, and there is a dependency among the identified challenges. The proposed improvements are in line with the identified challenges. Regression testing goals identified in this study are customer satisfaction, critical defect detection, confidence, effectiveness, efficiency, and controlled slip through of faults.

Considering the current state of practice and identified challenges, we conclude that there is a need to reconsider the regression test strategy in the companies as most of the identified challenges are either management related or have a dependency to test strategy. We further suggest that researchers need to analyze the industry perspective while proposing new regression testing techniques. The industry-academia collaboration projects would be a good platform in this regard.

1.7.4 On the search for industry-relevant regression testing research

Regression testing is a means to assure that a change in the software, or its execution environment, does not introduce new defects. It involves the expensive undertaking of re-running test cases. Several techniques have been proposed to reduce the number of test cases to execute in regression testing, however, there is no research on how to assess industrial relevance and applicability of such techniques. In Study 4, we conducted a systematic literature review with the following two goals: firstly, to enable researchers to design and present regression testing research with a focus on industrial relevance and applicability and secondly, to facilitate the industrial adoption of such research by addressing the attributes of concern from the practitioners' perspective. Using a reference-based search approach, we identified 1068 papers on regression testing. We then reduced the scope to only include papers with explicit discussions about relevance and applicability (i.e. mainly studies involving industrial stakeholders). Uniquely in this literature review, practitioners were consulted at several steps to increase the likelihood of achieving our aim of identifying factors important for relevance and applicability. We have summarized the results of these consultations and an analysis of the literature in three taxonomies, which capture aspects of industrial-relevance regarding the regression testing techniques. Based on these taxonomies, we mapped 38 papers reporting the evaluation of 26 regression testing techniques in industrial settings.

1.7.5 Lessons learned from replicating a study on information-retrieval based test case prioritization

Researchers have been proposing techniques to support regression testing practice, and a few of them are evaluating their techniques in the industry context. However, most regression testing techniques proposed in research have not been evaluated in industry. Adopting these techniques in practice is challenging because the results are inaccessible to the practitioners, and they do not know the context these techniques can fit. Replications of existing solutions for regression testing can be helpful in this regard, provided the availability of data and other contextual information for these replications. Replication studies help solidify and extend knowledge by evaluating previous studies' findings. However, too few replications are conducted in the field of software testing focusing on artefact-based experiments.

In Study 5, we have replicated an artefact-based study on software regression testing. The original study presents a test case prioritization technique. We attempted to replicate the original study using six software programs, four from the original study and two additional software programs. The replication study was implemented using Python (Jupyter notebook) to support future replications.

As an outcome of the replication experiment, various general factors facilitating replications are identified, such as: (1) the importance of documentation; (2) the need of assistance from the original authors; (3) issues in the maintenance of open-source repositories (e.g., concerning needed software dependencies); (4) availability of scripts. We also raised several observations specific to the study and its context, such as insights from using different mutation tools and strategies for mutant generation. The conclusion of our replication experiment is that it is not easy to externally replicate an experiment when context information and relevant data are not available in a complete manner, and without the support of original authors it becomes an uphill task.

1.7.6 Checklists to support decision making in regression testing

From our interactions with practitioners of various companies for our studies (1, 2, 3, & 6), we observed that regression testing practice lacks a well defined structure, which is among the primary reasons for various regression testing challenges.

In Study 6, with the input from 25 experienced testing professionals of 12 companies, we identified factors that practitioners think are significant to consider while planning, performing, and analyzing regression testing. Based on these factors, we designed regression testing checklists to help practitioners make regression testing decisions. We sent these checklists to the participating practitioners and collected their feedback. We made improvements in the regression testing checklists based on the suggestions received from the participants. Finally, the practitioners provided feedback on the proposed checklists concerning their usefulness in their environment, and most participants provided us with positive feedback.

The first part of the proposed checklists can help test managers to gauge the readiness of their team/team members and decide on the start of regression testing activity. These checklists can also help practitioners track all necessary steps while planning and performing regression testing. The second part of the checklists can help the test team for post regression testing activity. The participants from two companies showed their intention to use these checklists. In future, based on the feedback from the companies using the checklists, we plan to improve these checklists. We are also looking to automate these checklists and use machine learning techniques to make an intelligent decision support system for regression testing.

1.8 Discussion

Working on strategies to reduce the industry-academia gap and improving regression testing practice is the primary goal of this thesis. To achieve this goal, we opted to work on three research questions. **RQ1** aimed at understanding the current state of regression testing practice in the companies, identifying the challenges the practitioners face during regression testing, and finding the essential goals for regression testing. In **RQ2**, we worked on strategies to identify regression testing research that could be mapped to industry context and find ways to adopt regression testing research in practice. Finally, in **RQ3**, the aim was to help practitioners improve regression testing practice without altering how they work.

In this quest, we conducted six studies, four of which involved testing practitioners. While working on strategies to tailor regression testing research to the industry context, we engaged practitioners partially. We conducted an experiment-based study where a practitioner was involved in all steps of the study. In the following paragraphs, we present a brief discussion of the findings of this thesis.

1.8.1 Regression testing state of practice

Concerning the state of regression testing practice, the companies consider regression testing an essential activity for their products, and in most cases, they cannot ignore it. Some times exploratory testing is used to complement regression testing. The frequency of regression testing depends on the domain and criticality of the product/module under test. There is no concept of a separate regression test plan, but it is considered part of the overall test plan. Sometimes, an informal regression test plan is part of the sprint planning meeting. The practitioners set regression testing goals, most do it informally, and some have a formal mechanism for setting and assessing the goals. Expert judgment based on experience and domain knowledge is the primary driver for decision-making. The companies are transitioning from manual to automated regression testing. The changes and their impact are the basis for selecting a regression test scope. Based on knowledge of requirements specifications, the practitioners decide on the impacted modules to include in the regression test scope. During the regular changes, regression testing is performed with a selected set of test cases. However, before the release, companies with automated regression tests prefer to run the complete regression suite.

While working on the studies to understand the regression testing practice, we identified disparities in research and practice perspectives of regression testing. Hence we endorse the findings concerning the gap in research and practice of regression testing [8, 23, 25, 26]. The challenges identified by the practitioners were divided into two categories (i.e., i) challenges that need process support and ii) challenges that need technique level support). The regression testing challenges identified in this thesis are not new, as some of the challenges were discussed a decade ago [1]. There could be two reasons for recurring challenges: 1) regression testing techniques proposed in the literature do not fit the companies' context, and 2) companies are not aware of the availability of the techniques that could be suitable for their context. Both facts entail the need to work on the strategies of mapping the regression testing research to the industry context.

1.8.2 Supporting the adoption of regression testing research in practice

Test case selection and prioritization are the challenges identified in our multi-case study presented in Chapter 4. To support the practitioners in mitigating these technique-level challenges, we identified 26 regression testing techniques evaluated in an industry context. We created a taxonomy to map the regression testing techniques to industry context, goals, and needs. The identified techniques apply to web-based, real-time, embedded, database, and component-based systems. The challenges listed in this thesis represent the perspectives of the companies working on large-scale embedded systems. The technical areas where these companies seek improvements are related to test case selection and prioritization. In chapter 5, we have identified five studies, which are addressing the test case selection and prioritization issues for regression testing in embedded systems [18, 21–24]. Some common characteristics of these studies are that the systems under test consist of several million lines of code and development processes are iterative in their nature. Four studies [18, 21, 23, 24] have been carried out in the telecommunication domain, while the study presented in [22], was conducted in the automotive domain. The technique presented in [24] works for test case selection and prioritization, while other mentioned techniques work for test case selection. The detail about the mentioned techniques with respect to scope, context, effect, and information is presented in Table 1.3. Desired effects addressed in these techniques could be mapped to the regression testing goals identified in Chapter 2 (Study 1) and Chapter 3 (Study 2).

To support the adoption of regression testing research in the industry, we experimented with replicating a test case prioritization technique. During the experiment, we observed that various limitations could hinder the successful replication/adoption of regression testing techniques. Based on the lessons learned in the replication experiment, we have devised a set of guidelines to facilitate future replications and the adoption of regression testing techniques. This thesis also presents a goal-question-metric model to compare research and practice perspectives on regression testing. The GQM model provides a detailed mapping of regression testing goals, information needs, and measures. The model is flexible and could be tailored to a specific industry context by adding or removing goals from it. Practitioners can use this model to follow and evaluate the regression testing goals according to their context. Researchers can use the GQM model for proposing new techniques to fulfil practitioners' goals and needs.

Table 1.3: Mapping of regression testing techniques for embedded systems in taxonomies

SID	Scope	Context	Effect	Information
[18]	Selection	Heterogeneous embedded	Effectiveness & Efficiency, improved precision	Source code & Test reports
[21]	Selection	Product-line embedded	Effectiveness & Efficiency	Source code & Issues
[22]	Selection	embedded	Effectiveness & Efficiency	Design Artifacts
[23]	Selection	Product-line embedded	Test suite reduction, improved precision	Source code & Issues
[24]	Selection / prioritization	Product-line embedded	Effectiveness & Efficiency	Test reports

1.8.3 Providing support to improve regression testing practice

While interacting with the practitioners for our various studies, we observed that practitioners do not follow a defined regression testing process, and their decision-making is based on expert judgement. The lack of testing process in the industry projects has been reported by Kasoju et al. [54]. It is common to use expert judgment for various software development activities. However, making decisions without defined procedures can impact the results negatively. Experienced practitioners may overlook essential aspects when making reviews and decisions [47]. To improve the regression testing process and provide support in practitioners' decisions, we have designed regression testing checklists with the input of 25 senior testing practitioners. The proposed checklists can help practitioners structure their regression testing activities and ultimately improve the practitioners' decision-making on regression testing. We did not alter the way how practitioners carry out their activities. Instead, we documented the steps they think are essential to consider before, during, and after the regression testing.

The GQM model and regression testing checklists presented in this thesis are a step towards goal-based regression testing process improvement.

1.9 Conclusions and future work

Researchers argue that there is a gap between industry and academia [15, 25, 26]. In this thesis, we aimed to understand and reduce the industry-academia gap concerning regression testing. We identified a disparity in the research and practice perspectives on regression testing goals. The practitioners emphasize the goals like confidence, test suite maintenance, controlled fault slippage, and awareness of changes. In contrast, the focus of researchers while proposing regression testing techniques is on increasing test suite's rate of fault detection and maximizing coverage. We learned that the practitioners rely on expert judgment for test case selection, prioritization, and other decisions related to regression testing. There are several challenges that practitioners face during regression testing activities. They need research support on test case selection and prioritization issues, improving communication, streamlining regression testing activities, and methods for estimating success goals.

We have proposed a GQM model that maps the practitioners' and researchers' goal along with the information needs and measures. This model can help the practitioners to follow and assess the success goals, and researchers propose regression testing techniques relevant to practitioners' goals and needs.

To find test case selection and prioritization techniques suitable to industrial context, we have proposed regression testing taxonomies and demonstrated the use of these taxonomies by mapping 26 regression testing techniques according to industry context, desired effects (goals), and information needs. The proposed taxonomies can help communicate the regression testing techniques to practitioners. Further, we have replicated a test case prioritization technique to demonstrate the adoption of regression testing research. Based on the lessons learned, we have reported various guidelines that can help the future replication and adoption of regression testing techniques. These guidelines need to be considered by the researchers while conducting and reporting original experiments.

Lastly, we have proposed checklists to support the regression testing practitioners to improve communication among the test teams and streamline the regression testing activities. The proposed checklists will serve as a tool to remind the practitioners not to miss any essential regression testing activities. These checklists will also support the managers in various decisions about regression testing. Practitioners' feedback on the usefulness, comprehensiveness, and customizability of checklists was encouraging. We have delivered the final version of the checklists to the participating practitioners, and some of them are committed to sharing the usage data with us.

In future, we aim to test and enhance the GQM model and frame a comprehensive mechanism for the assessment of regression testing. We plan to improve and automate the regression testing checklists based on the companies’ usage data.

1.10 References

[1] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *Proceedings of the International Conference on Product Focused Software Process Improvement*. Springer, 2010, pp. 3–16.

[2] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[3] P. K. Chittimalli and M. J. Harrold, "Recomputing coverage information to assist regression testing," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 452–469, 2009.

[4] S. Banitaan, M. Alenezi, K. Nygard, and K. Magel, "Towards test focus selection for integration testing using method level software metrics," in *Tenth International Conference on Information Technology: New Generations (ITNG), 2013*. IEEE, 2013, pp. 343–348.

[5] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proceedings of the Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2011, pp. 99–102.

[6] I. ISO, "Ieee, systems and software engineering–vocabulary," *ISO/IEC/IEEE 24765: 2010 (E)) Piscataway, NJ: IEEE computer society, Tech. Rep.*, 2010.

[7] P. Ammann and J. Offutt, *Introduction to software testing*. Cambridge University Press, 2016.

[8] X. Lin, "Regression testing in research and practice," *Computer Science and Engineering Department University of Nebraska, Lincoln*, pp. 1–402, 2007.

[9] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 93–116.

[10] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[11] R. K. Yin, "Case study research: Design and methods (applied social research methods)," *London and Singapore: Sage*, 2009.

[12] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[13] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[14] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 38.

[15] E. Engström, K. Petersen, N. bin Ali, and E. Bjarnason, "Serp-test: a taxonomy for supporting industry–academia communication," *Software Quality Journal*, pp. 1–37, 2016.

[16] V. Garousi, K. Petersen, and B. Ozkan, "Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review," *Information and Software Technology*, vol. 79, pp. 106–127, 2016.

[17] R. H. Rosero, O. S. Gómez, and G. Rodríguez, "15 years of software regression testing techniques – a survey," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 05, pp. 675–689, 2016.

[18] E. D. Ekelund and E. Engström, "Efficient regression testing based on test history: An industrial evaluation," in *Proceedings of IEEE International Conference on Software Maintenance and Evolution, ICSME*, 2015, pp. 449–457.

[19] G. Rothermel, M. J. Harrold, J. Von Ronne, and C. Hong, "Empirical studies of test-suite reduction," *Software Testing, Verification and Reliability*, vol. 12, no. 4, pp. 219–249, 2002.

[20] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Transactions on software engineering*, vol. 22, no. 8, pp. 529–551, 1996.

[21] G. Wikstrand, R. Feldt, J. K. Gorantla, W. Zhe, and C. White, "Dynamic regression test selection based on a file cache an industrial evaluation," in *Proceedings of the International Conference on Software Testing Verification and Validation, ICST*. IEEE, 2009, pp. 299–302.

[22] S. Vöst and S. Wagner, "Trace-based test selection to support continuous integration in the automotive industry," in *Proceedings of the International Workshop on Continuous Software Evolution and Delivery, CSED*, 2016, pp. 34–40.

[23] E. Engström, P. Runeson, and G. Wikstrand, "An empirical evaluation of regression testing based on fix-cache recommendations," in *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation, ICST*, 2010, pp. 75–78.

[24] E. Engström, P. Runeson, and A. Ljung, "Improving regression testing transparency and efficiency with history-based prioritization - an industrial case study," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation, ICST*, 2011, pp. 367–376.

[25] V. Garousi and M. Felderer, "Worlds apart: industrial and academic focus areas in software testing," *IEEE Software*, vol. 34, no. 5, pp. 38–45, 2017.

[26] V. Garousi, M. M. Eskandar, and K. Herkiloğlu, "Industry–academia collaborations in software testing: experience and success stories from canada and turkey," *Software Quality Journal*, vol. 25, no. 4, pp. 1091–1143, 2017.

[27] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE transactions on software engineering*, vol. 28, no. 2, pp. 159–182, 2002.

[28] A. K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment," *Communications of the ACM*, vol. 41, no. 5, pp. 81–86, 1998.

[29] M. J. Harrold and A. Orso, "Retesting software during development and maintenance," in *Proceedings of Frontiers of Software Maintenance FoSM*. IEEE, 2008, pp. 99–108.

[30] M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 2, no. 3, pp. 270–285, 1993.

[31] J. Chi, Y. Qu, Q. Zheng, Z. Yang, W. Jin, D. Cui, and T. Liu, "Relation-based test case prioritization for regression testing," *Journal of Systems and Software*, vol. 163, p. 110539, 2020.

[32] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, “Test case selection and prioritization using machine learning: a systematic literature review,” *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–43, 2022.

[33] N. bin Ali, E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, S. Kunze, and M. Varshosaz, “On the search for industry-relevant regression testing research,” *Empirical Software Engineering*, pp. 1–36, 2019.

[34] N. Juristo and O. S. Gómez, *Replication of Software Engineering Experiments*. Springer Berlin Heidelberg, 2012, pp. 60–88. [Online]. Available: https://doi.org/10.1007/978-3-642-25231-0_2

[35] J. L. Krein and C. D. Knutson, “A case for replication: synthesizing research methodologies in software engineering,” in *RESER2010: proceedings of the 1st international workshop on replication in empirical software engineering research*. Citeseer, 2010, pp. 1–10.

[36] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, “The role of replications in empirical software engineering,” *Empirical software engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[37] M. Shepperd, N. Ajienka, and S. Counsell, “The role and value of replication in empirical software engineering results,” *Information and Software Technology*, vol. 99, pp. 120–132, 2018.

[38] F. Q. Da Silva, M. Suassuna, A. C. C. França, A. M. Grubb, T. B. Gouveia, C. V. Monteiro, and I. E. dos Santos, “Replication of empirical studies in software engineering research: a systematic mapping study,” *Empirical Software Engineering*, vol. 19, no. 3, pp. 501–557, 2014.

[39] R. M. Bezerra, F. Q. da Silva, A. M. Santana, C. V. Magalhaes, and R. E. Santos, “Replication of empirical studies in software engineering: An update of a systematic mapping study,” in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2015, pp. 1–4.

[40] A. Santos, S. Vegas, M. Oivo, and N. Juristo, “Comparing the results of replications in software engineering,” *Empirical Software Engineering*, vol. 26, no. 2, pp. 1–41, 2021.

[41] M. Cruz, B. Bernárdez, A. Durán, J. A. Galindo, and A. Ruiz-Cortés, “Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018,” *IEEE Access*, vol. 8, pp. 26 773–26 791, 2019.

[42] B. Hales, M. Terblanche, R. Fowler, and W. Sibbald, “Development of medical checklists for improved quality of patient care,” *International Journal for Quality in Health Care*, vol. 20, no. 1, pp. 22–30, 2008.

[43] W. Y. Higgins and D. J. Boorman, “An analysis of the effectiveness of checklists when combined with other processes, methods and tools to reduce risk in high hazard activities,” *Boeing Technical Journal*, 2016.

[44] B. M. Hales and P. J. Pronovost, “The checklist - a tool for error management and performance improvement,” *Journal of critical care*, vol. 21, no. 3, pp. 231–235, 2006.

[45] A. Chaparro, J. R. Keebler, E. H. Lazzara, and A. Diamond, “Checklists: A review of their origins, benefits, and current uses as a cognitive aid in medicine,” *ergonomics in design*, vol. 27, no. 2, pp. 21–26, 2019.

[46] R. Van de Schoot, P. Lugtig, and J. Hox, “A checklist for testing measurement invariance,” *European Journal of Developmental Psychology*, vol. 9, no. 4, pp. 486–492, 2012.

[47] M. Usman, K. Petersen, J. Börstler, and P. S. Neto, “Developing and using checklists to improve software effort estimation: A multi-case study,” *Journal of Systems and Software*, vol. 146, pp. 286–309, 2018.

[48] M. A. Heroux and J. M. Willenbring, “Barely sufficient software engineering: 10 practices to improve your cse software,” in *2009 ICSE workshop on software engineering for computational science and engineering*. IEEE, 2009, pp. 15–21.

[49] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for software engineering research,” in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[50] B. A. Kitchenham and S. L. Pfleeger, “Principles of survey research: part 3: constructing a survey instrument,” *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 2, pp. 20–24, 2002.

[51] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[52] M. Felderer and G. H. Travassos, "The evolution of empirical methods in software engineering," in *Contemporary Empirical Methods in Software Engineering*. Springer, 2020, pp. 1–24.

[53] Y.-G. Guéhéneuc and F. Khomh, "Empirical software engineering," in *Handbook of Software Engineering*. Springer, 2019, pp. 285–320.

[54] A. Kasoju, K. Petersen, and M. V. Mäntylä, "Analyzing an automotive testing process with evidence-based software engineering," *Information and Software Technology*, vol. 55, no. 7, pp. 1237–1259, 2013.

[55] R. Kazmi, D. N. Jawawi, R. Mohamad, and I. Ghani, "Effective regression test case selection: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–32, 2017.

[56] H. Koziolek, "Goal, question, metric," in *Dependability metrics*. Springer, 2008, pp. 39–42.

[57] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[58] D. Parsons, T. Susnjak, and M. Lange, "Influences on regression testing strategies in agile software development environments," *Software Quality Journal*, vol. 22, no. 4, pp. 717–739, 2014.

[59] E. Juergens, B. Hummel, F. Deissenboeck, M. Feilkas, C. Schlogel, and A. Wubbeke, "Regression test selection of manual system tests in practice," in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR)*, 2011, pp. 309–312.

[60] D. Badampudi, C. Wohlin, and K. Petersen, "Experiences from using snowballing and database searches in systematic literature studies," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE*, 2015, pp. 17:1–17:10.

[61] K. R. Felizardo, E. Mendes, M. Kalinowski, E. F. d. Souza, and N. L. Vijaykumar, "Using forward snowballing to update systematic reviews in software engineering," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2016, pp. 53:1–53:6.

Chapter 2

Regression testing goals - views of practitioners and researchers

2.1 Introduction

Regression testing is a well-researched area. However, the majority regression testing techniques proposed by the researchers are not getting the attention of the practitioners [5]. Communication gaps between industry and academia, and disparity in the regression testing goals are the main reasons. Close collaboration can help in bridging the communication gaps and resolving the disparities. A close collaboration between industry and academia is important to both sides, and this collaboration should be based on similar views of the studied problems and their importance [12]. Setting common goals and achieving a shared understanding is important for successful industry-academia collaboration. Having consensus on goals for collaborative research is a real challenge [21]. For a successful regression testing, it is essential to be able to manage the constraints. The key constraint of regression testing is the maintenance of the test suite (adding new test cases or updating or deleting obsolete test cases) [8, 11]. Test suite maintenance is not an easy task and if not done in a correct manner, utility of the test suite will be decreased and associated risks will be amplified [10]. To measure the success of regression testing, we need to define the regression testing goals. Chernak [17] emphasizes that test suite evaluation is the basis for the improvement of the overall testing process.

In earlier work Engström et al. [5] investigated regression testing practices and challenges using the focus group meeting and an online questionnaire with the industry practitioners. We complement these findings by exploring the value for practitioners and researchers alike. The objective is to reflect on how to evaluate regression testing. By choosing the right measures for the goals of a successful regression testing. From the EASE [1] platform, together with the testing practitioners, we identified seven software testing challenges in 3 companies. These companies operate in mobile-communications, surveillance, and embedded software systems. To identify the testing challenges at the companies, we utilized the SERP-test taxonomy. The SERP-test is designed to support the industry-academia collaboration [6]. The identified challenges were related to test planning, test design, and test execution. Out of these challenges, three were related to regression test selection, regression test prioritization, and test suite minimization. With the consultation of companies' representatives, we find that companies were more interested to cope with the regression testing challenges. This study is a step forward in the identified direction, with a focus on understanding the regression testing goals. The broad objective of the study is to obtain the answer to the following question:

RQ : *What are the views of academics and practitioners about regression testing?*

The study aims at exploring the views of academics and practitioners about the goals of regression testing. The purpose is to investigate the commonalities and differences in their viewpoints and defining some common goals for the success of regression testing. We conducted a focus group study with industry and academic participants. Seven experts participated in the study. Among the participants, 4 were representatives of testing practitioners from 2 large companies, and 3 were senior researchers from 2 universities. The contributions of this study could be listed as, a) regression testing definition, b) success goals, c) information needed (questions) to evaluate the success and d) measures to answer the questions.

The reminder of this paper is structured as follows: Section 2.2 presents the related work, Section 2.3 presents the detail about the methodology (i.e. planning, design, and conduct of the focus group). Threats to validity have been discussed in Section 2.4. Study results have been discussed in Section 2.5, and conclusions on key findings have been presented in Section 2.6.

[1]EASE- the Industrial Excellence Centre for Embedded Applications Software Engineering `http://ease.cs.lth.se/about/`

2.2 Related work

Researchers believe that industry-academia collaboration in software engineering is very low [19–21]. Garousi et al. [19] emphasize the importance of collaboration between industry and academia to support improvement and innovation in the industry. Ramler et al. [18], suggest the collaboration between industry and academia for the improvement and innovation of software testing in the industry. This collaboration could be the basis for transferable and empirically evaluated results. To facilitate the collaboration between industry and academia, Engström et al. [26] proposed a taxonomy of testing. The taxonomy can assist to improve communication between practitioners and researchers. It can work for both types of communication (i.e. direct communication and indirect communication).

Kapfhammer [7] pointed out the limited adoption of regression testing techniques, the reason identified is the lack of empirical evaluations. Chernak [17] stresses the importance of test suite evaluation as a basis for improving the test process. Chernak emphasizes that objective measures should be defined and built into the testing process to improve the overall quality of testing. Rothermel & Harrold [1, 2], proposed a 5 step framework to evaluate the regression testing techniques.

Engström et al. [6] suggested that more empirical evaluations conducted in industrial settings are required to facilitate the adoption of regression testing research in practice. The authors concluded that in order to enable practitioners to utilize the outcomes of research on testing, these outcomes must be evaluated in the actual environment. Through a focus group and an online questionnaire, Engström & Runeson [5] conducted a survey on regression testing practices, authors investigated what is considered regression testing by practitioners i.e. the definition, purpose and scope of it. They further investigated the challenges practitioners face with respect to regression testing. Our work complements the results of [5], as our subjects are the representatives of both sides (i.e. industry and academia). It is comparatively more focused, as purpose was to identify the regression testing goals.

We conducted an exploratory study to systematically elicit the goals, information needs and measures. We are focusing on industry-academia collaboration within regression testing challenges. The current focus is on regression test suite evaluation, as the first step in this study we tried to establish the regression testing goals.

2.3 Methodology

Focus groups are used to acquire the viewpoints of a group on some defined topic, which is a common area of interest for all group members. The key role in the focus groups is the moderator, who is responsible for guiding, facilitating and making sure that the discussion stays focused. Different guidelines are available for focus groups, Kontio et al. [4], [3] have deduced software engineering specific guidelines for conducting focus groups. Our approach to conducting the focus group was aligned with [4], a brief description about each step is given in the following sub sections.

2.3.1 Planning the research.

It is essential to make sure, that the focus group is suitable for the planned work [4]. Considering the research question presented in Section 2.1, our intention was to know the viewpoints of academics and practitioners about regression testing. Focus group was selected as it facilitates discussion, immediate reflection and it helps find the depth of the problem and some potential ideas for future research. As part of planning, we acquired the informed consent of the participants. We did also inform all participants about the purpose of the activity.

2.3.2 Designing the focus groups.

Focus group can comprise 3 to 12 participants, but a suitable number is between 4 and 8 [4]. We invited 7 participants from 2 Sweden based companies and 2 Swedish universities. Among the invited participants, 4 were testing practitioners from the companies (2 from each). 3 participants were senior academics from 2 universities. It is important to mention that all 3 academics are actively involved in software testing research. A brief description of the participants is shown in Table 2.1.

We followed the GQM approach for the focus group. GQM is an established method for planning and executing software engineering research and capturing software engineering related phenomena [25]. We phrased the questions using the interview guide formulated by Petersen et al. [9]. Table 2.2 shows the GQM template for the evaluation of regression testing, the template is divided into 5 activities (i.e. A1. A2, A3, A4, & A5). The purpose of A1 and A2 was to identify and prioritize the regression testing goals respectively, whereas A3 was to elicit the information needs (questions) corresponding to the identified goals. A4 was to capture the measures that could be used to answer the questions of related goal(s), while the objective of A5 was to know about the measures that the industry experts are actually using for the evaluation of test suites.

Table 2.1: Focus group participants

P.Id.	Organization	Participant's Expertise
P1.	Sony Mobile Communications	Testing/ Development
P2.	Sony Mobile Communications	Testing/Development
P3.	Axis Communications	Testing/Development
P4.	Axis Communications	Testing
P5.	Blekinge Institute of Techology	SE & Testing Research
P6.	Lund University	RE & Testing Research
P7.	Lund University	Regression Testing Research

2.3.3 Conducting the focus group session.

A focus group may last for 2 to 3 hours and it should have a predefined schedule. Within one session, the number of issues to be focused should be limited so that participants can have sufficient time to give their opinion on every aspect of the topic [4]. We allocated 2 hours for the activity, 30 minutes were assigned for setting up the focus group environment and introducing the basic purpose to the participants, although the overall objective was already communicated. We used the following schedule in the focus group:

1. Introduction: Short introduction to the goals of the workshop.
2. Round-the-table: What is regression testing in your view? Describe in one to 2 sentences.
3. Summarizing, presenting and verifying.
4. GQM activity (Table 2.2).
5. Summarizing, presenting and verifying (after every GQM, i.e. A1....A5).
6. Closing (Any other reflection or discussion points? Next steps).

We used color stickers (green and yellow) for data collection, green stickers were used by the practitioners and yellow stickers were used by the researchers. Discussion points were recorded by 2 of the authors. We took several breaks in between to collect the answers (to gather the sticky notes), cluster similar answers, put logical labels on clusters. Reflect on the names of the clusters and also whether individual sticky notes belong in it. Finally, we presented the initial results and asked the participants to verify the labels according to their given options.

Table 2.2: GQM-Template for evaluation of regression testing

Activity	Question	Rational
A1	Regression Testing is successful when a), b), c)... Complete the sentence (e.g. Regression testing is successful if it is a) efficient.)	Capture the goals
A2	Which success factors/goals are most important to you? Prioritize.	Prioritize success factors and goals and hence determine which measures should really be collected and whether this matches to what is collected today.
A3	What information is needed to evaluate success? Formulate as a question (e.g. How complex are our test cases, How many test cases are we running in a given test period?)	Capture the information needs (questions)
A4	What measures do we need to collect to answer the questions? (e.g. #test-steps for complexity)	Capture the measures that allow to quantify (and hence automate) the analysis of results
A5	What are you collecting today (measurements) of what has been identified in #4	Learn about input we already have available for evaluating test suites

2.3.4 Analyzing the data and reporting the results.

We followed the inductive approach for data analysis. It is a systematic approach for analyzing qualitative data [22, 23].

According to Thomas [22],*"inductive analysis refers to approaches that primarily use detailed readings of raw data to derive concepts, themes, or a model through interpretations made from the raw data by an evaluator or researcher"*.

The inductive approach allows the findings to emerge from the raw data without imposing any restrictions, the approach revolves around 3 steps: 1) data reduction, 2) data visualization and 3) conclusions and verifications .

We collected the sticky notes from the participants and made the groups of the responses along with the labels (reduction). We displayed the results to the participants and asked them to verify the labels with reference to their options. For example, we received the 43 options for regression testing goals, we reduced the options to 10 by making the clusters of the options on the basis of similarities. After the clustering of the data, results were displayed and the participants were invited to verify the labels according to their given options. In the second phase, together with the authors, results were reviewed by all participants in a separate review meeting, resultantly identified anomalies were fixed in the results.

The inductive approach provided us with the required flexibility to understand the viewpoints of the experts. The outcomes of focus group study are presented in Section 2.5.

2.4 Threats to validity

This study presents the viewpoints of academics and practitioners about the goals, information needs and measures of regression testing. The results presented here are of an exploratory nature. We addressed the threats to validity according to guidelines of Runeson and Host [24].

Construct validity: This aspect of validity is regarding the underlying operational measures, concepts and terms of the study. One potential threat to construct validity for our study is the subjects of the study representing 2 different working environments (i.e. academics and industry). Potentially they can have different understanding of concepts and terms. To mitigate the threats to this aspect of validity, we started with the exploration of the perception of participants about regression testing. To ensure the common understanding about the concept and terms during the entire focus group meeting.

Internal validity: This aspect of validity threat is important if causal relations are examined. Generally, we can state that studying causal relationships was not in the scope of this study. It is a descriptive/interpretive study, as it presents the viewpoints of the participants. We created a mapping between information needs and measures, that is the only causal relationship presented in the study. The mapping created between information needs and measures requires empirical evaluation to determine the validity of relationships between information needs and measures.

External validity: This aspect of the validity refers to the generalization of findings. We selected subjects of the study from academics and industry, to ensure the acceptability of results for both communities (i.e. practitioners and researchers). But as the practitioners were representing only 2 companies, so acceptability of results cannot be ensured in all companies working in the field of telecommunication. Further analytical generalization of results is possible, to support this we have reported the information of the participants in Table 2.1.

Reliability: To ensure the reliability of the study, we triangulated the results, as we presented and verified the initial results to the participants during the focus group meeting. Later after the complete analysis, results were presented to all participants in a review meeting. For detail please refer to the Section 2.3.4. Goals and measures identified in this study have not been verified through actual implementations.

2.5 Results and analysis

2.5.1 Defining regression testing.

As a kick-off activity, we asked the experts to give their opinion about, *[What is regression testing in your view?].* The purpose was to elicit the definition of regression testing with respect to participants' perception/experience. 5 out of 7 people came up with their definitions, presented in Table 2.3. Here an interesting fact that can be drawn from the individual definitions is the agreement between the views of academics and practitioners. We find that, the definitions presented at S.No. 1, 2 and 5 are almost the same and could be grouped together. Similarly, definitions at 3 and 4 are on same lines and we can group these 2 as well. After collecting the 5 definitions, we presented the definitions to the participants. Participants were agreed with our grouping scheme i.e. to group 1,2, & 5 and 3 & 4 in the form of the following 2 definitions:

1. *Regression testing is an activity which gives us confidence in what we have done and a trust that we have not broken anything.*
2. *Regression testing is an activity which makes sure that everything is working correctly after the changes in the system and it is a guarantee to continue in future.*

Table 2.3: Defining regression testing

S.No.	Perspective	Definition
1.	Academic	Make sure that we have not broken anything.
2.	Academic	Trust on what you have done
3.	Industry	To make sure that everything else work correctly
4.	Industry	To make future work possible, it is a guarantee to continue in future
5.	Industry	Trust on what you have done and make sure that we have not broken anything

Regression testing definitions presented in the literature

We selected 3 definitions from the literature to compare with the definitions presented by our experts. First definition was selected from a study presented by Engström and

Runeson [5]. We selected this definition as it represents the practitioners' perspective and it could be regarded closer to our concept. Second and third are the standard definitions taken from IEEE software Engineering terminology [13], and IEEE, Systems and software engineering – vocabulary [14] respectively.

1. *“Regression testing involves repetitive tests and aims to verify that previously working software still works after changes to other parts. Regression testing shall ensure that nothing has been affected or destroyed” [5].*

2. *“Regression testing is defined as retesting a system or component to confirm that changes cannot introduce any new bugs or causes other code errors, and the system or component can still follow the prescribed requirement specification” [13].*

3. *“Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements” [14].*

We observed that the definitions finalized in our study are closer to the definition presented by Engström and Runeson [5]. The distinctive factor of the definition proposed in our study is that it presents the viewpoints of both practitioners and researchers, while Engström's definition presents the viewpoints of practitioners only. On the other hand IEEE standard definitions is about that after the modification modified system or component still conforms to the specified requirements. That is system or component still works correctly after the changes. Our second definition conforms with the standard definitions. If we look at the individuals' definitions presented in Table 2.3, three words (*make sure, guarantee, and trust*) are prominent. This indicates that through regression testing our experts are seeking some assurance about the system's correctness, a guarantee that future work is possible and a trust on what they have done. Moreover, the results indicate that practitioners and researchers have similar viewpoints on the definition of regression testing that addresses one of the challenges highlighted in Section 2.2.

Regression testing definition adopted

During the second phase of the study (i.e. presentation of results and obtaining feedback from the participants), it was decided to adopt a single definition for the study.

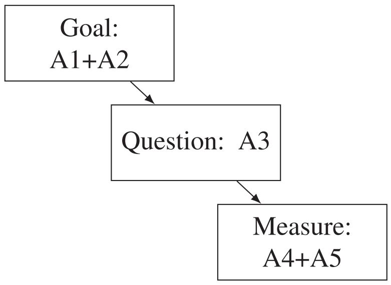

Figure 2.1: GQM representation

The agreed upon opinion was to merge the two definitions into a single definition in a way that it should represent the viewpoint of all participants. Later on, we combined the both definitions and created the following definition:

> *Regression testing is an activity which makes sure that everything is working correctly after the changes to the system. It builds the trust, that nothing has broken in the system and it guarantees to continue work in the future.*

2.5.2 GQM activity.

To execute the GQM (Goal, Question, Metric) theme, we used the GQM template, the template of questions used here are the inspiration from [9]. We divided this activity as A1, A2, ..., A5 as listed in Table 2.2. The purpose of A1 was to elicit the goals and A2 was to prioritize the goals. A3 was for the elicitation of information needs (questions) to achieve the regression testing goals. With the A4 we intended to collect measures for answering the questions related to information needs. Finally, with the A5 intention was to know about the measures that are currently used by the practitioners. The concept followed in the study is represented in Figure 1.

A1–Goals identification

To identify the goals, participants were asked to complete the following sentence, *[Regression Testing is successful when a), b), c) ?]*. Here a), b), c) indicate that the participants can state more than one goal regarding the success of regression testing. A total of 43 different options for goals were identified by the experts, we find that majority of the options were similar. For example we had the following options for G1 (Fault slippage to the customer):

1. No fault slippage through.
2. The customer/user finds no further bugs/flaws.
3. No issue leakage.
4. Ensure that the system fulfills the desired properties and no stopping bug slipped away.
5. We have no issue leakage on release.

With the consent of participants, we decided to group the identified goals on the basis of similarities and assign an appropriate label for each group. Hence the identified goals were restricted into 10 regression testing goals. The final list of the goals along with the description about each goal is shown in Table 2.4.

There are some goals identified by the participants, which are either irrelevant or too generic in scope. For example, visual overview could be taken as irrelevant or too broad in scope. Similarly, automation could be subsumed in efficiency. Achieving desired pass fail rate has been highlighted by 4 participants, if we see the goal description it can be subsumed by the effectiveness goal. It is important to highlight that visual overview and automation were identified by only one participant.

Confidence, Efficiency, and Effectiveness are the goals identified by the majority of participants. Here it is important to mention that a goal identified by more participants does not refer to its importance, rather it only shows how may subjects have pointed out a particular goal. G5 (i.e. confidence) was identified by all 7 participants, but with varying descriptions. For example, some of the perceptions can be summarized as, *"Stakeholders are confident with the reached quality and/or we can ensure that nothing is broken."* To measure the desired quality or to determine that nothing is broken, requires multiple testing metrics.

A2–Goals prioritization

Next task was to assign the priority order to the elicited goals. The question asked to the participants was, *[Which success factors/goals are most important to you? Prioritize]*. The participants were asked to assign priorities against every goal, each participant was given 10 points to prioritize. We used colored markers for priority assignment, red for researchers and Black for practitioners. As the distribution of experts was not equal on both sides (i.e. 3 researchers and 4 practitioners), we decided to normalize the priority of both sides. For normalization we devised an equation presented at (2.1).

$$NP = AP/N * 4 \tag{2.1}$$

Table 2.4: Regression testing goals

G.Id.	Options	Goal	Goal description
G1.	5	Fault slippage to customer	The customer/user finds no further bugs/flaws
G2.	3	High precision	Non affected test cases excluded
G3.	3	Inclusiveness	Have run the most effective tests first
G4.	5	Achieved desired coverage	All desired modules have been executed when a regression test suite runs
G5.	7	Confidence	Stakeholders are confident with the reached quality and/or We can ensure that nothing is broken
G6.	7	Efficiency	Finished retesting with the limited time and low cost
G7.	7	Effectiveness	Costly faults detected early and/or finding new defects in old code
G8.	1	Visual overview	Visualization of complete software is displayed
G9.	1	Automation	Being able to automate
G10.	4	Achieving desired pass fail rate	When the expected tests pass and/or fail

Here NP = Normalized Priority, AP = Actual Priority and N = No. of Experts.

The normalized priorities along with the total points are shown in Table 2.5. G5 *(i.e. Confidence)* was marked with 21 total points, G2 *(i.e. High precision)* was given 17 points while G1 *(i.e. Fault slippage to customer)* was third in the list with 14 points. It was observed that in most cases there was a sort of agreement between researchers and practitioners. But there was a complete disagreement regarding the priority of some goals. We can see that for researchers G1 & G5 are the highest priority goals with equal priority, whereas for Practitioners G5 is the highest priority. Similarly, G8 and G9 are somewhat important for practitioners but researchers assigned *zero* to both the goals. An interesting fact, that we think is important to mention here is that the participants on both sides marked *zero* priority for G7 (i.e. effectiveness). Although this goal was identified by all 7 participants. And it is among the goals which have been

Table 2.5: Allocated priorities to the goals

G. Id	A Priority	I Priority	Total Priority
G1.	9	5	14
G2.	8	9	17
G3.	4	3	7
G4.	3	0	3
G5.	9	12	21
G6.	7	3	10
G7.	0	0	0
G8.	0	5	5
G9.	0	3	3
G10.	0	0	0

cited in the literature by different authors [16, 17]. We found similarity in views of both sides, regarding the top 3 goals (i.e G5, G2, & G1 respectively). As G5 **(Confidence)** was ranked as the highest priority goal by the participants, and considering its generic nature we decided to elicit the information needs for G5, in the next phase of the focus group (i.e. A3). During the final review meeting participants were agreed to consider G1, G2, G3, G5, & G6 as the final list of goals.

A3–Questions (Information needs elicitation)

To elicit questions (information needs), participants were asked to answer the question, *[What information is needed to evaluate the success?].* We decided to elicit information needs only for G5 (i.e. confidence), we took the decision because of the following reasons:

1. Because of the generic nature of the goal.
2. It was ranked as the highest priority goal by the participants.
3. It was highlighted that, to achieve this goal multiple metrics need to be evaluated.

47 questions (information needs) were identified by the participants. During analysis, we find that a majority questions are similar. On the basis of identified similarity, we grouped these 47 questions (information needs) into 10. The final list of information needs questions is shown in Table 2.6. The questions with most options were, *Have critical parts been covered? (16 options)*, and *What are the test outcomes? (10 options)*. A majority of information needs listed in Table 2.6 are quantifiable, but some

Table 2.6: G5. Questions (Information needs)

Q.Id.	Question	Extracted from
Q1.	What are the changes to the system?	5 similar options
Q2.	What is the experience of development and testing?	3 similar options
Q3.	Have critical parts been covered?	16 similar options
Q4.	Have modifications been tested?	5 similar options
Q5.	What are the test outcomes?	10 similar options
Q6.	What is the perception of team about confidence?	3 similar options
Q7.	What is the nature of defects in the product?	2 similar options
Q8.	What is the balance between pass fail?	1 option
Q9.	What is the complexity of the product under test?	1 option
Q10.	What has been blocked by the tests?	1 option

information needs are relatively generic in nature. Information need listed at Q2 (Team Experience) cannot be taken as a direct testing metric, but it is important with regard to confidence. Similarly, Q6 (Confidence perception) is not a specific metric, still it can affect the measure of other factors. Product characteristics listed as Q9 can determine the complexity of the product, this can also affect confidence perception. We can draw a correlation between Q2, Q6, and Q9. After finishing with the clustering, the final list of grouped information needs was presented to the participants for the verification of the clusters. Later in the results review meeting, all the stakeholders were agreed to consider Q1, Q3, Q4, Q5, and Q7 as the final list of information needs to achieve the confidence goal. Participants were agreed about the subjective importance of Q2 and Q7 with respect to the underlying goal of confidence.

A4–Measures identification

The aim here was to identify the suitable measures to collect the information needs and ultimately achieve the defined goal (i.e. Confidence). We asked, *[What measures do we need to collect to answer the questions?].* Our experts identified 5 measures presented in the Table 2.7. Later together with the experts we started a brainstorming activity to find the possible mapping between the questions and measures. We carried the activity in a step wise manner. That is, for every single goal we asked the experts to map it with possible measure(s). 4 measures (i.e. M1,M2,M3, & M4) were mapped to 7 questions (i.e. Q1, Q3, Q4, Q5, Q7, Q8, and Q10). The finalized GQM (goal-question-measure) mapping is shown in Figure 2.

Table 2.7: Measures

MID	Measure
M1	#LOC changed
M2	#Defect fixes from test
M3	#Changed LOC covered
M4	#defect history/change
M5	#Affected Non-changed LOC/Modules

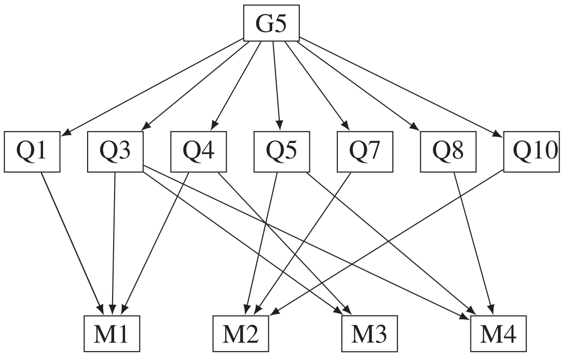

Figure 2.2: Goal-question-measure mapping

A5–Available measures

The last activity was to know about the actual measures that are being used in the companies. We asked the question, *[What are you collecting today? (measurements) of what has been identified in A4]*. This question was to know about the actual state of the implementation of measurement program regarding the evaluation of regression testing in the industry. Therefore we asked practitioners to answer this question. We requested the researchers to sit as observers and provide their feedback on the activity. Practitioners expressed, that they do not use any explicit measures. There is no predefined measurement mechanism regarding the evaluation of regression testing that could be used. Instead, they rely on their experience, to evaluate the success. Their agreed-upon statement about the measurement was, it is a gut feeling, that we have tested enough and we are successful. To further continue and to come up with some substantial outcome, we added another question.

Do, we actually need to evaluate the success of regression testing?

We asked the participants to provide their opinion about the need for measuring the regression testing. There was a consensus among the participants about the importance of measuring the success of regression testing. It was emphasized that suitable measures need to be defined and used in the companies. It was also highlighted, that participating companies are interested to implement an evaluation mechanism/framework to measure the success.

Related measures presented in the literature

To make a way towards the identification/implementation of evaluation framework, we decided to identify the measures from literature and test the identified measures in the partner companies. As a starting point we identified some measures from literature to further strengthen our findings.

Rothermel and Harrold [1, 2] presented a complete framework for the evaluation of the regression testing techniques. They suggested *inclusiveness, precision, efficiency, generality, & accountability* as measures to evaluate the regression testing. Horváth et al. [10] used *code coverage & partition metrics* for measuring fault detection capability and fault localization. They defined coverage metric (Cov) as a ratio of the number of procedures in a code group P that are covered by test group T. Whereas they defined partition metric (Part) to express the average ratio of procedures that can be distinguished from any other procedures in terms of coverage. *output uniqueness* is defined by Alshahwan and Harman [15], who define the output uniqueness as if the 2 test cases yield different kinds of output. The authors believe that this metric can help in effective fault detection capability, it also works for fault finding consistency.

Vidacs et al. [11] uses the code coverage, efficiency & uniqueness for Assessing the Test suites of large system. The authors argue that better coverage or partitioning can be achieved using more test cases, provided test cases are different. But, in case if such test cases are added to the test suite, which covers the same code, they will increase the test suite size possibly with little additional benefit. They suggested measuring the efficiency, that (refer to the relative number of test cases in test suite), to measure efficiency, they defined *coverage efficiency* (EFFCOV) and *partitioning efficiency* (EFFPART). Coverage efficiency refers to the average number of procedures covered by a test case, while partitioning efficiency shows that on average, how much a single test contributes to the partitioning capability of whole functional unit. To measure uniqueness authors used 2 metrics (*specialization* metric SPEC and *uniqueness* metric UNIQ). SPEC shows how specialized a test group is to a code group, while the UNIQ

Table 2.8: Measures found in literature

S.No.	Metric	Measure	Reference
1.	Effectiveness	Defects, TestCaseEscaps, AssertionDensity, Directness	[16, 17]
2.	Fault Detection Capability	CodeCoverage, OutputUniqaueness	[10, 11, 15, 16]
3.	Fault localiztion	Partition	[10]
4.	Effeciency	EffCov, EffPart	[11]
5.	Uniqueness	UNIQ, SPEC	[11]

metric measures what portion of the covered elements is covered only by a particular test group.

To measure the effectiveness, Chernak [17] named his measure as *defect*, which is the ratio between the number of defects covered by a test suite to the number of defects missed by the test suite. Athanasiou et al. [16] argued that test code quality has 3 dimensions completeness, effectiveness, and maintainability. They defined *assertion density* as a measure of calculating the effectiveness of test code to detect the defects. For the effectiveness of test code authors also suggested *directness* as measure, they defined directness as it measures the extent to which the production code is covered directly. Test suite evaluation metrics and corresponding measures selected from literature are presented in Table 2.8.

Mapping between focus group and literature findings

As we mentioned already, due to the time constraint, we investigated only one goal (G5) in the subsequent steps of the focus group session. Therefore we decided, to create a mapping between the goals presented in the Table 2.4, and metrics/measures we find in the literature. Majority goals listed in the Table 2.4 are specific and measurable. Measures presented in the literature can be mapped to identified goals. For instance, G1 can be mapped to the metric "Fault detection capability" , related measures have been discussed in the following studies [10, 15, 16]. G2 & G3 can be mapped to the metrics "precision " and "inclusiveness" defined in [1, 2]. Similarly, G6 can be linked to the metric "Efficiency" presented in [1, 2, 11]. Finally, G7 can be mapped to "effectiveness" metric discussed in [16, 17].

The measures identified from literature can also be mapped to some of the questions listed in Table 2.6. For example, Q5 could be mapped to *No. of Defects, TestCaseEscaps, & OutputUniqueness,* similarly Q7 can be mapped with *No. of Defects & CodeCoverage,* Q3 can be mapped with *AssertionDensity & Directness*.

2.6 Conclusions

In this paper, we presented the results of our exploratory study. The study presents the views of practitioners and researchers about the regression testing definition, goals of regression testing, information needed (questions) to evaluate the success, and the measures to reveal the required information. For the elicitation of information, we used the focus group method, a total of seven experts (representatives of industry and academia) participated in the study. We have identified the five priority goals for the success of regression testing including fault slippage to the customer, high precision, inclusiveness, confidence, and efficiency. The top priority and common goal among all participants was ‘confidence’ (about the system integrity). We identified information (seven questions) needed to achieve this goal. We also elicited the measures corresponding to these information needs. Finally, a mapping among goal, questions, and measures was created.

While defining the concepts and goals of regression testing, we did not observe any significant difference of opinion between researchers and practitioners. However, there were visible differences in the priority of goals. We believe that such platforms where industry and academia can sit together can be beneficial for both. Resultantly, researchers can work on actual industry problems and practitioners could be able to cope with the real challenges by using the relevant research.

2.7 References

[1] G. Rothermel and M. J. Harrold, "A framework for evaluating regression test selection techniques," in *Proceedings of the 16th International Conference on Software Engineering, ICSE-16., 1994.* IEEE, 1994, pp. 201–210.

[2] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Transactions on software engineering*, vol. 22, no. 8, pp. 529–551, 1996.

[3] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: obtaining practitioner and user experiences," in *Proceedings of the International Symposium on Empirical Software Engineering, ISESE'04. 2004.* IEEE, 2004, pp. 271–280.

[4] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 93–116.

[5] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *Proceedings of the International Conference on Product Focused Software Process Improvement*. Springer, 2010, pp. 3–16.

[6] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[7] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2011.* IEEE, 2011, pp. 99–102.

[8] L. S. Pinto, S. Sinha, and A. Orso, "Understanding myths and realities of test-suite evolution," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012, p. 33.

[9] K. Petersen, C. Gencel, N. Asghari, and S. Betz, "An elicitation instrument for operationalising gqm+ strategies (gqm+ s-ei)," *Empirical Software Engineering*, vol. 20, no. 4, pp. 968–1005, 2015.

[10] F. Horváth, B. Vancsics, L. Vidács, Á. Beszédes, D. Tengeri, T. Gergely, and T. Gyimóthy, "Test suite evaluation using code coverage based metrics," pp. 46–60, 2015.

[11] L. Vidács, F. Horváth, D. Tengeri, and Á. Beszédes, "Assessing the test suite of a large system based on code coverage, efficiency and uniqueness," in *Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016*, vol. 2. IEEE, 2016, pp. 13–16.

[12] S. Masuda, "Software testing in industry and academia: A view of both sides in japan," in *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2017*. IEEE, 2017, pp. 40–41.

[13] I. S. C. Committee *et al.*, "Ieee standard glossary of software engineering terminology (ieee std 610.12-1990). los alamitos," *CA: IEEE Computer Society*, 1990.

[14] I. ISO, "Ieee, systems and software engineering–vocabulary," *ISO/IEC/IEEE 24765: 2010 (E)) Piscataway, NJ: IEEE computer society, Tech. Rep.*, 2010.

[15] N. Alshahwan and M. Harman, "Coverage and fault detection of the output-uniqueness test selection criteria," in *Proceedings of the International Symposium on Software Testing and Analysis 2014*. ACM, 2014, pp. 181–192.

[16] D. Athanasiou, A. Nugroho, J. Visser, and A. Zaidman, "Test code quality and its relation to issue handling performance," *IEEE Transactions on Software Engineering*, vol. 40, no. 11, pp. 1100–1125, 2014.

[17] Y. Chernak, "Validating and improving test-case effectiveness," *IEEE software*, vol. 18, no. 1, pp. 81–86, 2001.

[18] R. Ramler, M. Felderer, T. Kitamura, and D. Marinov, "Industry-academia collaboration in software testing: An overview of taic part 2016," in *Proceedings of the IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2016*. IEEE, 2016, pp. 238–239.

[19] V. Garousi, M. M. Eskandar, and K. Herkiloğlu, "Industry–academia collaborations in software testing: experience and success stories from canada and turkey," *Software Quality Journal*, pp. 1–53, 2016.

[20] V. Garousi and K. Herkiloglu, "Selecting the right topics for industry-academia collaborations in software testing: an experience report," in *Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST), 2016*. IEEE, 2016, pp. 213–222.

[21] V. Garousi, K. Petersen, and B. Ozkan, "Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review," *Information and Software Technology*, vol. 79, pp. 106–127, 2016.

[22] D. R. Thomas, "A general inductive approach for analyzing qualitative evaluation data," *American journal of evaluation*, vol. 27, no. 2, pp. 237–246, 2006.

[23] L. Liu, "Using generic inductive approach in qualitative educational research: A case study analysis," *Journal of Education and Learning*, vol. 5, no. 2, p. 129, 2016.

[24] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[25] V. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, vol. 2, no. 1994, pp. 528–532, 1994.

[26] E. Engström, K. Petersen, N. bin Ali, and E. Bjarnason, "Serp-test: a taxonomy for supporting industry–academia communication," *Software Quality Journal*, pp. 1–37, 2016.

Chapter 3

Using goal-question-metric to compare research and practice perspectives on regression testing

3.1 Introduction

Regression testing is carried out subsequent to any change in the system to verify that the change did not impact the unchanged parts of the system [37, 66, 71]. It is a complex and costly activity, especially for large scale systems with continuous integration and delivery, and can consume up to 80% of testing and 50% of maintenance cost [7, 43, 65, 66]. The research proposes test case selection and prioritization to deal with the cost and complexity of regression testing [6, 9, 67, 71]. Test case selection refers to selecting a subset of test cases from the regression test suite to test the effects of changes. In contrast, test case prioritization guides an optimal ordering of test cases that can help achieve the desired goals [6, 9]. If the selected suite is large, test case prioritization can be applied as a subsequent process. However, test case selection and prioritization can also be applied independently. The primary goal of regression test case selection and prioritization techniques is to detect faults as early as possible [71].

One of the challenging aspects of software testing is to decide when to stop the testing. How much to test, is an essential question as it affects the overall budget of the project. Especially in large scale software development with continuous integration, it is imperative for the practitioners to decide how long they should be testing the software before releasing it [57–59, 81]. Various authors have defined prediction models for stopping criteria while taking into consideration the testing time, effort, cost, reliability, and coverage [58, 60, 61]. The industry practitioners set regression testing goals and evaluate the achievement of these goals using their experience and product knowledge. From the practitioners' perspective, regression testing goals provide an opportunity to decide to stop running more tests, as the achievement of the defined goals gives them confidence about the attained quality, and they can decide to release the product [4, 29].

A goal is an intended outcome of a process that a practitioner plans to achieve, and it should be realistic and measurable. Therefore a goal should be associated with the metrics which can be used to evaluate it. Regression testing goals could correspond to the pre-defined objectives that a practitioner wants to achieve by applying a regression testing process or technique. The achievement of these goals should be assessed using metrics (see, e.g., [42]). Furthermore, test case selection and prioritization should be based on regression testing goals. These goals may vary from organization to organization based on their priorities [7]. The goal of most existing regression testing techniques is effectiveness (increasing test suite's rate of fault detection). Some techniques encompass efficiency (i.e., execution time and cost) as a goal. Test coverage is also among the goals of various techniques as the assumption is that test cases with a higher coverage will detect more defects [66]. Coverage-based techniques aim to cover maximum code with fewer test cases [34]. Regression testing techniques utilize multiple sources of information, including coverage information, requirement information, and test execution history. Furthermore, to evaluate the outcomes, these techniques use metrics including APFD (average percentage of fault detected) and its variants, coverage-based metrics, and metrics related to execution time [71, 76, 79].

Various regression test selection and prioritization techniques have been proposed in the literature [9]. However, the adoption rate of these techniques in industry is not encouraging, and only a few techniques have been evaluated in the industry context. It is a clear indication of the gap between research and practice [7, 39–41, 43]. Among the other factors, one important aspect is the disparity in the regression testing goals of practitioners and researchers [4]. There are few studies that have an explicit focus on regression testing goals, especially in an industry context [2, 4, 29].

This research aims to get a better understanding of the regression testing goals from the literature and practitioners' perspective. We, therefore, have reviewed the literature and conducted a survey with industry practitioners. For the survey, we opted for interviews and an online questionnaire as data collection methods. We incorporated the GQM (goal-question-metric) approach [52] to map regression testing goals with related information needs and metrics.

In an earlier study [4], we investigated regression testing goals, in a more limited scope. We conducted a focus group-based study with the practitioners and researchers. The participating practitioners represented large-scale embedded software development companies, and the researchers are actively working on testing research. This study aimed to know the industry-academia perspective on regression testing goals. The present study is the continuity of the earlier study and extends it by adding further data and insights concerning regression testing goals, information needs, and metrics. The earlier study used a focus group-based workshop with seven industry and academic participants. In contrast, the present study comprises findings from 44 research papers and perspectives of 56 industry practitioners (representing nine development domains). Table 3.1 presents a summary of how our current study extends the earlier study.

Table 3.1: A summary of how our current study extends our previous work [4]

Factors	Minhas et al. [4]	Current study
Focus of the study	Regression testing goals	Regression testing goals
Perspectives	Practitioners and researchers	Practitioners and literature
Literature review	Reviewing 7 articles to find measures	Reviewing 44 articles to find goals, information needs, and measures
Method used	Focus group	Interviews and online questionnaire
Participants	Academics and practitioners	Practitioners
No of participants	4+3 (7)	11+45 (56)
Development domains	2	9

The contributions of this study are as follows:

- Identification of some new regression testing goals from the practitioners' perspective.
- Mapping of regression testing goals to information needs, and metrics.
- Identification of differences in research and practice concerning regression testing goal preferences, use of information needs and metrics.
- Formulation of a GQM model to present an integrated view of the perspectives from the literature and from practice, that can be used as a guide to reduce the industry-academia gap.

The remainder of this paper is organized as follows: Section 3.2 presents a review of related work, Section 3.3 describes the methodology. Section 3.4 presents the results of the literature review and the survey. Section 3.5 discusses the implications of this study for researchers and practitioners, and Section 3.6 concludes the paper.

3.2 Related work

This section discusses related work on regression testing goals, information needs, and metrics. Included studies are recent systematic literature reviews, literature survey, and some empirical studies on regression testing.

We looked at 11 systematic reviews on regression testing published during the last six years (i.e., 2017 to 2022) [43, 71–80] to learn the recent trends in regression testing techniques and see which goals, information needs, and measures are considered. Six of the 11 studies reviewed the techniques regardless of application domain or techniques type. Four of the 11 studies are conducted with a specialized focus. For example, Pan et al. [71] reviewed machine learning-based regression testing techniques, Abdul Manan et al. [73] reviewed regression test prioritization in combinatorial testing, Husnain et al. [74] reviewed regression test case prioritization techniques for web services, and Lima and Vergilio [78] reviewed regression testing techniques in continuous integration environment. The most-reported goal in these studies is increasing the fault detection capability, whereas the evaluation metrics reported are APFD (average percentage of faults detected) and its variants. Pan et al. [71] conducted a systematic review of machine learning-based regression test case selection and prioritization techniques. The authors motivate their work from the context of continuous integration. They argue that with the adoption of continuous integration in software development, the frequency of regression testing is increased, and running all tests can be time-consuming and resource-intensive. This problem could only be resolved by introducing regression test case selection and prioritization. Pan et al. further revealed that machine learning-based techniques rely on multiple sources of information, including coverage information, test execution history, and domain-specific information. The evaluation metrics used in these techniques are variants of APFD and some general metrics like precision and recall. The goals of the machine learning-based techniques are early identification of critical faults and increasing the test suite's rate of fault detection.

Rahmani et al. [79] conducted a systematic review of regression test case prioritization (TCP) techniques proposed from 2017 to 2020. The authors classified the techniques based on TCP approaches (e.g., risk-based, history-based, etc.). The authors also investigated the metrics and source of information utilized with these techniques. The metrics reported in this study are the variants of APFD, execution time, code coverage, requirement coverage, and severity measure. The information used for these techniques is requirement information (e.g., requirement coverage, requirement dependency). The most-reported goal for the techniques proposed during 2017–2020 is increasing the test suite's rate of fault detection. In another review [80] the authors classified the regression test prioritization techniques based on the approaches and metrics used. The criteria used in the prioritization techniques are cost, code coverage, and fault detection ability. The goal highlighted by the authors is effectiveness, and to measure the effectiveness, they mentioned the use of precision and recall.

Rehan et al. [72] conducted a systematic analysis of multi-criteria based regression test selection. The authors analyzed the techniques based on the selection criteria and metrics used for evaluation. They reported that the efficiency of test selection depends on execution cost, coverage, fault detection ability, and code changes. Early detection of critical faults and increasing the test suite's rate of fault detection are the goals discussed in the study. The information sources utilized in the techniques are coverage information, the number of faults detected, execution history, degree of severity, and execution time. The metrics used in the reviewed techniques are coverage, fault detection rate, code modifications, and severity measure. Abdul Manan et al. [73] analyzed regression test case prioritization in combinatorial testing. The goals of these techniques are to increase the test suite's rate of fault detection and early identification of critical faults. The metrics used with these techniques is APFD, APFDc (cost-cognizant weighted average percentage of faults detected), and APSC (average percentage of statement coverage).

Lima and Vergilio [78] presented a mapping of regression test prioritization techniques in continuous integration environment. The authors revealed that the trend of proposing test case prioritization techniques during the last four years had been increased, and 80% of the proposed techniques are history-based. The goal of the majority of the techniques is to increase test suite's rate of fault detection. The evaluation metrics used in these techniques are APFD, APFDc, fault detection rate, and time. The sources of information utilized in these techniques are test execution and fault history, coverage information, code changes, user priority. Hasnain et al. [74] analyzed the regression test case prioritization techniques for web services. The authors reported that most regression testing techniques for web services are criteria-based. These techniques use coverage information, test adequacy, fault severity, and fault dependency as sources of information. Evaluation metrics used for these techniques are APFD, APFDc, APFDD (average percentage of fault dependency detected), fault detection rate, and severity measure. The goals of these techniques are increasing test suite's rate of fault detection and early identification of critical faults.

bin Ali et al. [43] reviewed the empirically evaluated regression testing techniques. The aim was to map the existing regression testing techniques to the aspects of the industrial context. Along with mapping the existing solutions on regression testing to the industry context, the authors mapped these solutions regarding desired effects (goals) and information needs. According to the authors, existing techniques are considering the goals of increasing test suite's rate of fault detection, identification of sever/critical faults, and coverage. The information sources mentioned in this study are requirement changes, code changes, execution history (test reports), and fault severity. The measurement metrics mentioned in this study are cost, coverage, severity measure, and fault detection rate.

In a systematic literature review, Khatibsyarbini et al. [76] classified the regression test prioritization techniques as fault-based, history-based, search-based, and coverage based. The authors revealed that APFD is the most utilized metric, followed by APFDc and coverage metrics. The goal of these techniques is effectiveness (i.e., increasing test suite's rate of fault detection). Test execution history and code coverage information are mentioned as information sources in this study. de S. Campos Junior et al [77] conducted a review of empirical studies and systematic literature reviews on test case prioritization. They revealed that most reported prioritization techniques are coverage based and history-based. The information sources utilized in these techniques are test execution and fault history, coverage information, change analysis, and requirement information. The metrics used for evaluation are APFD, APFDc, and APSC.

Along with the recent systematic literature reviews, we selected a literature survey published in 2012 by Yoo and Harman [9] to see the trends of regression testing goals from the older studies. The reason to select this study is that it presents regression testing research spanned over three decades. Yoo et al. surveyed the literature on regression testing published between 1977 – 2009 to explore different regression testing techniques. The authors analysed various approaches to regression testing and provided the list of trends, issues, and goals of regression testing techniques. The goals listed in this work are increasing test suite's rate of fault detection, early identification of critical/severe faults, detection of faults related to changes, and coverage. Confidence is stated as the overall goal of regression testing. The authors stated, *"the purpose of regression testing is to provide confidence that the newly introduced changes do not obstruct the behaviours of the existing, unchanged part of the software."* Information needs mentioned in this study are test execution and fault detection history, code coverage information, and requirement information. Whereas the metrics highlighted in this survey are APFD, APFDc, fault severity, code coverage metrics, and metrics related to changes.

Besides considering the systematic reviews of regression testing, we also reviewed some primary studies to see the trends of regression testing concerning the goals, information needs, and metrics. These studies either propose a regression testing technique or present practitioners' perspectives.

Jafrin et al. [2] proposed an algorithm to prioritize test cases based on the rate of severity detection associated with dependent faults. In this study, the authors listed the goals for test cases and test case prioritization goals. Prioritization goals listed in this study are increasing test suite's rate of fault detection, increasing coverage, confidence, increasing rate of high-risk fault detection, and revealing the faults related to changes.

However, the authors did not explain the sources from where they have identified these goals.

Kwon et al. [68] proposed an information retrieval and coverage based regression test prioritization technique. Increasing test suite's rate of fault detection was considered the goal of the technique, whereas information sources utilized are code coverage and fault detection rate. The authors suggested using mutation faults in the absence of actual faults. To measure the test suite's rate of fault detection, the authors used the APFD metric.

In a survey, Engstrom et al. [7] stressed the need to define the organization-specific regression testing goals. However, the authors did not mention any such goals in the study. White et al. [41] performed an industrial study. The authors listed a few goals concerning regression testing. The goals observed in this study are, early defect detection based on changes and critical defect detection. The study also presents the metrics like module dependencies, execution cost, time, number of test cases executed, and code changes. It is reported in various studies [15, 17, 31] that most of the regression techniques presented in the literature are using effectiveness (increasing test suite's rate of fault detection) as a goal. To measure the effectiveness, the authors are using APFD and APFDc metrics. In most cases, authors utilize test execution and fault detection history to evaluate their techniques concerning effectiveness (i.e., increasing test suite rate of fault detection). However, fault mutation can also be utilized to compensate for the absence of actual faults [68].

From the recent systematic reviews representing the regression testing research up to 2022 and survey by Yoo and Harman [9], and the primary studies presented above, we learned that the common goal of most techniques is increasing test suite's rate of fault detection. Early identification of critical faults and coverage are also mentioned as goals of some techniques. The most utilized metrics in all studies are APFD, APFDc, and code coverage metrics. Most of the reviewed literature present the goals of regression testing techniques, and a few studies considered the regression testing goals a primary concern. Only a couple of studies considered this aspect from an industry perspective. Furthermore, we could not find a precise mapping between the goals and metrics. This fact motivated the authors to conduct a study to investigate research and industry perspectives on regression testing goals and related aspects. The current study is the continuation of our earlier work [4, 29], and it extends [4] by adding literature findings, and perspective of more practitioners representing more domains (see Table 3.1 in Section 3.1).

3.3 Methodology

The study aims to characterize the regression testing goals from research and practice perspectives, and it also strives at comparing two perspectives on regression testing goals. Table 3.2 presents the research questions that further elaborate the study's aim. To answer the research questions, we have chosen to conduct the literature review and a survey. For the literature review, we selected the studies where the authors discuss regression testing goals, information needs, and metrics. We did not opt to conduct the systematic literature review (SLR) because, along with the in-depth analysis, an SLR covers the breadth of the existing literature relevant to the research questions [64]. An in-depth analysis of regression testing techniques is not the goal of this study. The only aim was to identify the regression testing goals, information needs, and metrics. For selecting relevant literature, we performed systematic searches that helped to include a reasonable number of relevant studies. We could have two alternatives to understand

Table 3.2: Research questions

RQ	Motivation
RQ1) What are the goals of regression testing discussed in the literature, and what are the corresponding information needs and metrics to evaluate these goals?	The objective is to better understand which goals are considered by the researchers while proposing or evaluating regression test selection and prioritization techniques. Which information needs they utilize to achieve the goals. Moreover, to evaluate the goals, what metrics have been proposed by the researchers. We will also investigate to see if there is any mapping between the success goals, information needs, and metrics (e.g., what metrics could be used to evaluate a specific success goal?).
RQ2) What are the goals of regression testing defined by the practitioners, and what are the corresponding information needs and metrics to evaluate these goals?	The objective is to know if the practitioners define any goals to determine the success in regression testing. Moreover, to see if they use/define any information needs to achieve the goals and evaluate these with the prescribed metrics.
RQ3) How are the findings from the literature and the survey related?	The aim is to create an integrated view of findings on regression testing goals, information needs, and metrics and provide actionable guidelines for practitioners and researchers.

the practitioners' perspective of regression testing goals, i) case study and ii) survey. A case study investigates a phenomenon in deeper detail. It is a suitable method for the situations where context is important, and analysis of the cause-effect relationship is the aim [63]. In contrast, a survey helps to identify the characteristics of a larger population and it is a suitable method where the aim is to collect the opinions of a large sample [63]. In our case, we were interested to know the perception of as many practitioners as possible. Simultaneously, we were also keen to know some insight about the regression testing goals and other associated practices. Therefore, we decided to survey by opting for two data collection methods interviews and an online questionnaire. Interviews provided us an opportunity to have direct interaction with the practitioners and understand their perceptions. The online questionnaire helped us to reach the broader population and collect the information about regression testing goals from a larger sample.

3.3.1 Literature review

Study selection

To answer the first research question (RQ1), we have conducted a literature review of 33 selected papers. Though we did not conduct systematic literature review (SLR), we used systematic searches and followed established methods to extract and present the selected papers' data. However, we do not claim the exhaustive searches of the studies, and we did not incorporate quality assessments. The reason was that the aim was to get a view of what goals, questions, and metrics exist concerning regression testing.

Search strategy: We followed snowball search strategies for the selection of studies [45]. Snowball search strategy helps find all relevant studies and still not get too many irrelevant papers to be excluded manually in the subsequent steps [47]. The first step in snowball searches is to find the start set, then we have to iterate the backward and forward snowball iterations [45].

Finding the start set: To find a start set for snowball searches, we used keywords based search to identify a basic set of papers, and used the following search string:
(“regression testing” OR “retesting”) AND (“goal” OR “desired effect”) AND (“metric” OR “measure” OR “information need”). We applied the search string in IEEE, Scopus, and Inspec. We found a total of 175 research papers. We did title scanning of these 175 papers and selected 62 relevant papers for further processing. Later we read the abstracts of the selected 62 papers, and after applying the inclusion/exclusion criteria, we selected 13 research papers to include in the start set for snowball searches.

Snowball iterations: By taking the selected 13 articles as start-set, we performed snowball iterations (see Table 3.3). In the backward snowballing, we examined the references of every paper in the start-set. For the forward snowballing, we reviewed the studies that were citing any of the papers in the start-set. For the identification of citations and searching for papers, we used google scholar. In each iteration, the papers were selected based on the inclusion and exclusion criteria mentioned below. In the event of selection, new papers further went through the snowball iterations. In the first iteration we found 16 related papers, and in the second iteration we found only four papers. We stopped the process after the second iteration because we could not find new papers related to our topic.

Table 3.3: Snowball iterations

Iterations	**References**	No of Studies
Start set	[1–13]	13
Iteration 1	[15–30]	16
Iteration 2	[31–34]	4
	Total No of studies included	**33**

Additional searches: Since our initial searches were focused on the regression testing goals, information needs, and metrics, we did not focus on any development context during the snowball iterations. However, to overcome this limitation, besides the snowball searches, we looked at systematic reviews of regression testing published during the last six years (i.e., 2017 to 2022) to see if there are studies published lately considering the context-specific regression testing (goals, information needs, and measures). We searched the Scopus database to find the systematic reviews on regression testing, and we found eleven systematic reviews published during the last six years.

Inclusion and exclusion criteria: Table 3.4 presents the inclusion and exclusion criteria we used for the selection of primary studies. We selected regression testing studies that include goals, information needs, and metrics regardless of the development domain. Other constraints that were applied are the language of the article, publication stage, and availability of the article in full text.

Table 3.4: Inclusion and exclusion criteria

	Inclusion Criteria	Exclusion Criteria
1	Article is written in English	Article is written in a language other than English
2	Article is focusing on any aspect of regression testing goals, information needs, or metrics	Article related to regression testing but not considering any aspect of regression testing goals, information needs, or metrics
3	Article is peer reviewed (i.e., journal, conference, and workshop)	Article is not peer reviewed (Grey literature)
4	Article is available in full text (i.e., the article is downloadable)	Article is not available in full text

Data extraction

Before the data extraction, we went through the reading of selected studies, and after the first round of reading, we started identifying the goals, information needs, and metrics. We used different colors (green for goals, grey for information needs, and yellow for metrics/measures). After finishing with the color-codes, we assigned appropriate labels (where required), and finally, we extracted data by using the data extraction form (see Table 3.5). Data extraction was performed jointly by the first and second authors.

Table 3.5: Data extraction form

Data Item	**Description**
Title	Title of the selected study
Authors	Authors' names and affiliations
Publication	Type, year, and venue of publication
Research Method	Stated objectives of research, and chosen research methodology
Goals	Regression testing success goals, along with the authors' description of the goals
Information needs	Information required to fulfill the achievement of the goal
Metric	Metrics/measures used/mentioned in the study to evaluate the achievement of the goals

Validating literature findings

To ensure the correctness and consistency of data extracted from literature, the first author reviewed the second author's data. The second author reviewed the data extracted by the first author. Issues were discussed and resolved jointly. Finally, the third author did a random check of the extracted data.

3.3.2 Survey

To answer the second research question (RQ2), we have conducted a survey comprising of interviews and an online questionnaire.

The interviews allow an in-depth investigation of any phenomenon, while the questionnaire provides an opportunity to broaden the scope of findings [47]. We have chosen to conduct interviews with the testing practitioners, as we were interested in understanding the practitioners' perspectives in a detailed manner. Later, to know the perspective of the testing practitioners at large, we distributed an online questionnaire among the practitioners of various companies. Along with the interview guide and online questionnaire detail, the following subsections present the steps carried to conduct the survey.

Sample selection

For the surveys, sample selection from the target population is a crucial step [50]. Considering the challenge of selecting a representative sample of all testing practitioners worldwide using probability sampling, we chose non-probability sampling methods (i.e., convenience and snowball sampling). Non-probability sampling provides an easy way to select samples using non-random sampling techniques, including convenience sampling, quota sampling, or snowball sampling [35]. To ensure the selection of suitable participants for the survey, we set a pre-condition that the participant must have worked or is currently working in regression testing. We made this characteristic mandatory for the interviews and also embedded this requirement in the online questionnaire. If a survey respondent has no experience in regression testing, he will not be able to continue with the questionnaire's subsequent sections. However, we did not put any boundaries concerning the years of experience. The reason for not limiting years of experience was that the more answers from people with regression test experience we gain, the more comprehensive the GQM model would be. The less experienced people may miss out on some of the goals in the organization due to lack of experience, they still contribute valuable input to the tree by providing a few goals/measures.

Interview participants: For semi-structured interviews, we used the convenience and snowball sampling methods [50]. We started with convenience sampling and contacted practitioners with experience in regression testing using our contact networks. Five participants responded to our first attempt. We started scheduling interviews with these respondents. Later we opted for snowball sampling and asked the participating respondents to refer us to practitioners experienced in regression testing, who can willingly participate in the study. With these five participants' help, we reached six testers who gave their consent to participate in the study (see Table 3.7 Section 3.4.2).

Online questionnaire participants: For the online questionnaire, we used the snowball sampling approach [50]. We asked the interview participants to provide us further contacts of practitioners with expertise in regression testing. We also sent LinkedIn messages to the testing practitioners and posted the link to the questionnaire in two testers' groups. From all these sources (i.e., contact snowballing, LinkedIn messages, and testing groups), we received 45 responses to our online questionnaire. The detail of online questionnaire participants is presented in Section 3.4.2, please see also Figure 3.1 and Figure 3.2.

Interview steps

Eleven practitioners of nine companies participated in the interviews. Seven of eleven respondents had testing experience ranging from 10 to 15 years. While two of eleven respondents had two years of testing experience, and two had testing experience of one year. Complete detail of interview participants is presented in Table 3.7 (Section 3.4.2).

Interview guide: We designed the interview guide (see Appendix A) based on the guidelines of Runeson et al. [46]. We opted for the open-ended questions in the interview guide that allowed the interviewees to present their views freely. The second author developed the interview guide, and the first author reviewed and revised it. Later, the third and fourth authors reviewed the interview guide and provided their feedback. The comments were discussed among the authors, and necessary changes were made in the interview guide. Finally, to test our interview guide, we conducted pilot interviews with two experts, and based on the feedback from these experts, we finalized the interview guide.

The interview guide is divided into three sections, including introduction, background, and regression testing. The introduction section explains the context and purpose of the study. The background section consists of questions to capture the interview respondents' background information, including their current role, testing experience, current projects/product under test, etc. In the regression testing section, questions were organized to understand practitioners' perspectives on regression testing in general and the success of regression testing in particular. Then, the questions to capture practitioners' viewpoint on regression testing goals, information needs, and metrics. In the end, we also added some questions to know the response of practitioners regarding the metrics we identified from the literature.

Interview conduct: We conducted semi-structured interviews, mainly containing open-ended questions, except the questions related to the metrics identified from the literature. To avoid researchers' bias we did not include any question that could lead to a desired answer. For interview questions please see Appendix A.

The interviews with open-ended questions make it hard to capture the complete responses by taking notes. There is a high chance of missing the essential aspects of the discussion. Besides taking notes, with the participants' prior consent, we audio-recorded all the interviews to ensure not to leave any piece of information from the participants' responses. Each interview took approximately 30 minutes.

Analysis: Data collected from the interviews were subject to qualitative analysis. For the analysis of qualitative data, we used thematic analysis, in the thematic analysis, the data is identified into themes and codes based on the frequency and relevance of the collected data. To carry out data analysis using thematic analysis, we followed a five steps process presented by Lacey et al. [51].

- *Transcription:* Since all the interviews were audio-recorded, the first step was to transcribe the interviews. The first and second authors transcribed the audio records. In the next step, both transcribers verified each other's transcripts. We also used notes taken during the interviews to complement the transcripts generated from audio recordings.

- *Organizing data:* The transcribed data is organized in some specific order to make it uncomplicated and easily accessible. At the first stage, we assigned id numbers to each interview, we also assigned ids to sections of interview transcripts. We eliminated the information from the transcripts that were possibly revealing the identity of the respondents or their organizations.

- *Familiarization with the data:* Since the interviews were conducted by the first and second authors alternatively. Therefore to completely understand the context of interviews, we repeatedly went through the listening of recordings and reading of transcripts.

- *Coding:* For the preliminary coding, we used different colors to categorize different themes in the transcripts. The investigation's primary focus was on three themes, which mainly correspond to the research questions (goals, information needs, and metrics). We used green color to highlight the goals, grey color to distinguish information needs, and yellow color to represent the metrics.

- *Themes:* To define more specific labels, we clustered the definitions based on the similarity of views. For instance, one of the interviewees stated a goal as *“The goal is that no fault with priority 1 (high-risk faults) should slip through. We* *want to make sure that the customer should not find any such fault.".* Another interviewee stated that *"We try to maintain a 100% success rate, we do not want any fault slippage to our customer"*. Similarly, one interviewee stated that *"All test cases in regression test pack should be executed with 100% pass, the goal is that customers should not find any fault"*. In all these statements, we can see that practitioners do not want any fault to be slipped to the customer. Therefore we grouped all these statements into one cluster. After arranging the goals and measures into relevant clusters, we assigned appropriate labels to goals, information needs and metrics by using literature findings. For instance, the goals discussed here were assigned a label of *"No or controlled fault slippage"*.

Validation: In addition to the above steps of interpreting and analyzing, after assigning labels, we validated our interpretations with the selected interview participants. In the feedback, we did not receive any complaint of misinterpretation or misquote from any of the respondents. All of them were agreed that our interpretations are appropriate and closer to their perspective.

Online questionnaire steps

Characterization of subjects: In response to our invitation to participate in the online questionnaire, we received 45 testing practitioners' responses. The respondents are QA Engineers, test leads, test managers, and test analysts. Regarding testing experience, the respondents' experience lie in the range of 2 to 15 years. The survey respondents are working on the products from different domains, including accounting/finance, automobiles, embedded systems, etc. For detail please see the Figures 3.1 and 3.2 (Section 3.4.2).

Questionnaire design: To conduct an online survey on regression testing goals, using google forms, we prepared a questionnaire [1]. The purpose was to expand the scope of our findings, and to some extent, validate the information collected from the literature and interviews. The questionnaire was designed by following the guidelines provided in [48] and guided by the results obtained from the literature and interviews. The questionnaire is divided into three sections. In the first section, the objective, motivation, and terminology of the research are explained. In the second section, questions are included to collect the respondents' background, the product under test, and information about the organization. In this section, there is a question about the respondent's experience in regression testing. They have to answer this question as yes or no, and respondents having experience in regression testing could proceed to the remaining

[1]https://drive.google.com/file/d/1aKDObyHGq6E5UTgtEXqJ_ZfGWpXz1sQL/view?usp=sharing

part of the survey. This ensured the quality and validity of the survey. The last section contains the questions specific to the research topic. There are a total of 21 questions in this section, and all are close-ended. In this section, we embedded the questions on information needs and metrics with their respective goals. If a respondent selects a goal, then the possible list of information needs and metrics is displayed. This helped to keep track of the right information/metrics for the right goals. We used a five-point Likert scale for the regression testing goals, which allowed respondents to disagree with any goal provided in the questionnaire. For information needs and metrics, we used the nominal scale. In addition to a specific list of information needs and metrics, the questionnaire provided free text space for respondents to include information needs and the metrics of their choice. These steps helped avoid researchers' bias.

Questionnaire validation: To evaluate the survey instrument, one of the methods is the pilot execution of the survey. The purpose of the pilot survey is to identify possible problems with the questionnaire [49]. To test our survey questionnaire's validity, we conducted the pilot survey with the two practitioners. They did not raise any significant issue in the questionnaire except suggesting to elaborate on the study's purpose. Based on the feedback, we made a few changes to the questionnaire.

Questionnaire conduct: We used google forms to distribute the questionnaire to the potential respondents. With the help of interview participants, we contacted 14 practitioners and requested them to forward the questionnaire to the people working in regression testing. We also sent 80 requests using LinkedIn messages, and we posted our questionnaire in two testing groups. As an outcome of these invites, we received 45 responses.

Analysis: Data collected using the questionnaire was subject to quantitative analysis. We were supposed to present the Likert scales' summaries for the goals, information needs, and metrics selected/identified by the respondents. We used descriptive statistics for the analysis of the data [44]. The results are presented in form of summary tables and graphs (see Figures 3.1, 3.2, & 3.3 and Table 3.6 in Section 3.4.2).

3.3.3 Threats to validity

This study employed the literature review and a survey as the research methods. There could be potential threats to the validity of the results obtained through literature and survey. The following subsections discuss the threats to validity and possible mitigation strategies, following the guidelines provided in [46, 47].

Construct validity: This aspect of validity could be associated with the choice of treatment for the study and its expected outcomes. In our case, it could be linked to selecting studies for the literature review, selecting survey participants, and creating the GQM model.

For the literature review, while selecting the primary studies, we opted snowballing technique [45]. However, we cannot guarantee the exhaustive searches, but the consistency of findings is the evidence that we retrieved a sufficient amount of relevant studies.
While designing the survey instruments, we carefully followed the respective guidelines [46, 48] for the design of the interviews and online questionnaire. Further, we conducted pretests by conducting pilot interviews and surveys. Based on the outcomes of pretests, we augmented our survey instruments. Concerning the survery participants, we used convenience sampling to select the initial participants. Later we used the snowball sampling method to select the participants further. Given the specific focus of the study (experience in regression testing), it was not easy to recruit practitioners. To reduce this threat to validity to some degree, we created our GQM model from three sources (Figure 3.4): the literature, the interviews with company representatives, and the survey. The consequence may be that we may have missed including some goals and measures in this model. Therefore, we do not claim to have developed an all-encompassing model for regression testing goals, information needs and measures. Rather, we created a baseline that companies can work with and extend based on their context.

Internal validity: Internal validity threats mainly deal with the credibility (data collection, sample selection) of the study, i.e., whether the obtained results are valid or not. Internal validity refers to the factors that affect the outcome of the research. We followed the well-defined search strategies to find the relevant studies, and employed systematic procedures for data extraction and analysis. We used audio recordings for the interviews and selected the interviewees based on their experience and interest in the regression testing. Furthermore, after transcribing the interviews and assigning the appropriate labels, we validated our interpretations from the interview participants. For the survey, the questionnaire is updated and revised before distributing it. In the questionnaire, we used the multiple-choice questions, and to avoid the researchers' bias, in every question, we provided the option for the free-text response.

External validity: The external validity threats refer to the concept of generalization of the results. Along with the literature review of 33 research papers, this study results from 11 interviews and 45 responses to the online questionnaire. The practitioners who participated in this study represent various organizations working on diversified domains and from different countries. Even though we have added data to the body of knowledge, since we employed convenience sampling to select interview participants, it may threaten external validity. However, we have provided the interview and questionnaire respondents' background information, which may help generalize the context.

Conclusion validity: The conclusion validity threat deals with the quality of the conclusions drawn from the collected data. We ensured the triangulation for all aspects of data that is data collection and interpretation. This study's conclusions are the outcome of data collected from multiple sources (literature review, interviews, and online questionnaire). We employed well-defined methods for data interpretation and analysis. We also verified our interpretations from the selected respondents.

3.4 Results and analysis

3.4.1 Literature review

To answer RQ1, we have selected 33 research papers. The selected studies are those in which authors consider regression testing goals or propose or evaluate the regression testing techniques based on goals. Various selected studies are also specifying the information needs and metrics that could be used to evaluate the goals' achievement. Besides the initial searches, we also looked at 11 systematic reviews of regression testing published during the last six years (i.e., 2017 – 2022) to determine whether these systematic reviews lead to additional goals, information needs, and metrics. The detailed review of these studies is presented in Section 3.2 (Related work).The findings of these studies are merged in the results. Table 3.6 presents a mapping of goals and corresponding metrics, along with the information needed to aid the assessment of the goal. We have created the mapping using the authors' descriptions in the studies and the following proposition:

*"To achieve/evaluate goal **G**, based on information needs **IN**, use the metric **M**."*

A | Regression testing goals identified from literature

This section presents a brief description of the regression testing goals found in the literature.

G1. **Increasing test suite's rate of fault detection:** Finding maximum faults early and quickly is the objective of any testing process, and it corresponds to the *effectiveness* of any testing method/technique [2, 31]. The goal is listed in 72% of the included studies.

Table 3.6: GQM mapping of regression testing goals, information needs, and metrics – Literature

Goal	Information need	Metric
G1: Increasing test suite's rate of fault detection [1, 2, 5, 6, 8–12, 12, 13, 16, 17, 17, 21, 22, 28, 29, 31–34] [43, 71–79]	Coverage based information [5, 32, 34] [71, 72, 74, 77, 78], Requirements information [28] [43, 77, 79], Test execution/-fault detection history [6, 9, 13] [43, 71, 72, 76–78]	APFD [2, 6, 9, 11–13, 17, 22, 28, 32–34, 36] [71, 73–79], APFDc [2, 5, 9, 22, 31] [71, 73–79], Test case failure rate [13]
G2: Early identification of critical faults [2, 5, 13, 15, 22, 23, 28, 29, 32, 33] [71–74]	Module criticality/Test criticality [5] [71, 72] Fault dependency matrix [10] [79], Changes in requirements [33] [43, 77, 78], Test execution/-fault detection history [15, 24, 25]	APFDc [2, 5, 22] [71, 73–79], APFDD [10, 15] [74], Fault severity [2, 5] [72, 74, 79]
0G3: Detection of faults related to changes [2, 3, 19, 32]	Changes in requirements [33] [43, 77, 78], Code changes [3, 19, 24] [43, 77, 78]	BFCP [3]
G4: Coverage [2, 4, 5, 11, 25, 29, 30, 32, 34] [43]	Coverage based information [25, 26, 32, 34] [71, 72, 74, 77, 78]	Code coverage metrics [5, 11, 13, 26] [43, 72, 73, 77, 79]
G5: No or Controlled fault slippage [4, 29]		
G6: Confidence [2, 4, 11, 29, 32]	Changes in requirements, Coverage based information, and Product complexity [4]	

G2. **Early identification of critical faults:** Finding highly critical faults early in the testing process is another performance goal for regression testing. It refers to detecting the faults that could have a severe impact on the system under test and can exist in critical modules. This goal appeared in 30% of the included studies.

G3. **Detection of faults related to changes:** Early detection of faults introduced by the developers due to changes and bug fixes is another performance goal because the presence of such faults could break the regression testing [3]. Such faults should be detected as early as possible. The goal is listed in 10% of the included studies.

G4. **Coverage:** Covering maximum code with a small number of test cases is the goal of regression testing techniques. These techniques are referred to as coverage-based techniques, and 25% of the included studies refer to this goal.

G5. **No or controlled fault slippage:** Fault slippage is a phenomenon where the testing process fails to find a fault in software under test, and the product is delivered to the subsequent phases (e.g., release). This goal is highlighted in only two included studies, and both these studies represent the practitioners' perspective.

G6. **Confidence:** The testers should have confidence in their regression testing process and ultimately they must have confidence about the reached quality of software under test [29, 32]. Confidence is listed as regression testing goal in the studies which are representing the perspective of practitioners (e.g., [4, 29], this goal appeared in 11% of the included studies.

B | Information needs and metrics identified from literature

Information needs: The fulfillment of regression testing goals is subject to the selection/prioritization of the test cases, and it requires practitioners to know various aspects, generally termed as information needs. We present here the information needs required to fulfill the regression testing goals identified in this study.

IN1: **Requirements information:** Based on the importance, stability, and fault-proneness of the requirements, practitioners can prioritize the test cases to increase the fault detection rate. Further requirements information can help the practitioners locate the source of defects more conveniently [28].

IN2: **Test case execution/fault detection history:** Using test execution history, we can evaluate various metrics for a test case, for example, fault detection rate, detection of severe faults, and test case failure rate. Various authors have specified the use of test case execution and fault detection history for the test case selection and test case prioritization techniques [6, 9].

IN3: **Module criticality/Test criticality:** To estimate fault severity, two possible information could be used, i. information related to module criticality (importance of the module under test), ii. information related to test criticality (ability of a test case to detect severe faults) [5].

IN4: **Changes in requirements/code changes:** Bug-fixing changes might break the regression testing [3]. The code changes are the source of a majority of new defects in the system. Therefore knowledge of changes is an essential information need for the regression testing process [19].

IN5: **Coverage based information:** The testers can utilize the coverage-based information to select/prioritize test cases. It will help achieve regression testing goals and ultimately increase the efficiency and effectiveness of the regression testing. Various authors mention utilizing this information for the test case selection and prioritization techniques [5, 32, 34].

IN6: **Fault dependency matrix:** The fault dependency matrix could be used to identify the leading faults. Such faults are considered severe faults, and identifying these faults is in the scope regression testing goal (i.e., early identification of critical faults) [10].

Metrics: Various metrics concerning regression testing techniques are presented in the literature. Below, we discuss the metrics relevant to evaluating the identified goals. It is significant to highlight that most metrics require data from test execution history, which might not always be available. Therefore, it is suggested that practitioners maintain historical data to assess metrics. For example, practitioners need to record the fault detection history to measure the average percentage of fault detection (APFD). They should record coverage-based information with test execution history to evaluate the code coverage metric. Furthermore, change-logs are required to assess the bug-fixing-change-impact-prediction (BFCP). The practitioners need to maintain the fault dependency matrix to measure the average percentage of fault dependency detected (APFDD).

M1 To measure the rate of fault detection, **APFD** (average percentage of faults detected) could be used [17, 32]. APFD measures the average of total percentage of faults detected by executing all the test cases present in the test suite. The APFD value is directly proportional to the fault detection rate.

M2 The **APFDc** (cost-cognizant weighted average percentage of faults detected) is used to measure the rate of fault detection and cost efficiency of test suite. It provides a mechanism to measure the varying test cases along with the fault cost and severity [22].

M3 **Test case failure rate** represents the ratio of number of times a test case has failed to a number of times it has been executed. A test case with higher failure rate is a potential test case to be included in the regression suite.

M4 **APFDD** (average of the percentage faults dependency detected) measures how quickly dependency among the faults can be detected during the execution of a test suite. APFDD values range from 0 to 100, higher value means faster dependency detection [15].

M5 Tang et al. [3] introduced an information retrieval (IR) based approach, **BFCP** (Bug-fixing change impact prediction). BFCP could help predict whether a bug-fixing change will break the regression testing. By mining the source code change history, it identifies the bug-fixing changes that can break the regression testing before running the regression test cases.

M6 **Code coverage metrics** estimate how much code is tested by a test set. Askarunisa et al. [5] introduced various coverage based metrics to evaluate the test coverage at various levels of detail, including, the average percentage of statement coverage (APSC), the average percentage of branch coverage (APBC), the average percentage of loop coverage (APLC), and the average percentage of condition coverage (APCC).

M7 **Severity measure** helps to identify the test cases that can reveal higher number of severe faults. Severity value could be assigned to the faults based on their impact on the product [56]. **ASFD** (average severity of faults detected) is a metric used to measure the severity of faults [53].

3.4.2 Survey

The survey findings consist of the interview results and the online questionnaire results. The following subsections present the findings from both means of the survey.

Interviews

We have conducted semi-structured interviews with eleven practitioners from nine different companies. The participating practitioners' testing experience ranges from 1 to 15 years. Concerning development approaches, all the participants reported using Agile/Scrum. Table 3.7 presents details about the interview participants. The practitioners who participated in the interviews represent four different countries (Sweden, Belgium, India, and the USA). They work on different product, including health care, mobile gaming, IT services, Telecom, retail and distribution systems, customized solutions, and infotainment systems. Although the participants represented different companies, there is overlap concerning the development domains. For example, participants I-1 and I-8 are working on health care systems, I-4 and I-5 are working on IT services. However, we can not infer that commonalities in the product domains impact the perspectives.

The primary focus of investigations was to know the perception of practitioners about regression testing in general and regression testing goals and metrics in particular. Besides collecting the background information of the participants, we asked them

Table 3.7: Interview participants

ID	**Role**	**Testing Experi- ence** (in years)	**Product**	**Approach**	Releases Per Year	**Location**
I-1	Lead QA engineer	15	Health care	Agile	2	Sweden
I-2	Program test manager	12	Retail and distribution	Agile	4	Belgium
I-3	Test lead	12	Web & Mobile products	Scrum	Releasing Patches Frequently	Sweden
I-4	Senior test lead manager	12	IT services	Scrum	4	USA
I-5	Technical test lead	11	IT services and consulting	Scrum	3	India
I-6	Senior test analyst and QA lead	10	Multiple domains	Agile	2 to 6	Sweden
I-7	Test lead	10	Infotainment systems	Scrum	Not fixed	India
I-8	SQA engineer	2	Health care	Scrum	6	India
I-9	Software tester	2	Mobile Gaming and casino	Scrum	6	Sweden
I-10	Software tester	1	Telecom	Scrum	4	Sweden
I-11	Test analyst	1	Financial Services	Scrum	6 to 12 Patches	Sweden

to tell us how they are performing regression testing in their company, the scope of regression testing, and the goals and metrics they use to assess their success in regression testing. To get an overall overview, we asked the participants to elaborate on, "What is regression testing for them? Why and when they need to perform the regression testing?" The practitioners' perception of regression testing is presented here:

I-1 *“After adding new features or bug fixing we go back and try to see if this change has broken something else.”*

I-2 *“In the event of any change, we have to perform regression testing to ensure that we have not damaged the quality of the existing software product's functionality.”*

I-3 *“To ensures that nothing is broken and everything is working in the system.”*

I-4 *“To make sure that whatever the bug fixed in the previous release, those do not break the existing working functionality.”*

I-5 *"When we introduce new changes to the application, we perform regression testing on the other areas of the application that are not part of new changes. To make sure that the new changes do not affect the other parts of the application. The functionality of the other parts is working correctly."*

I-6 *"Regression testing is to verify the existing functionality did not get affected with the update to the existing code, and that is the main idea of this."*

I-7 *"Regression testing focuses on finding out the side effects that might cause because of bug fixes, or it might be side effects because of the implementation of the new features. Therefore our primary focus is to identify the side effects of the changes."*

I-8 *"Because of changing technology, we need to add new features to our product, and after adding new features, we perform regression testing to ensure that the current product is working fine."*

I-9 *"In general, regression testing is system testing that ensures the software's quality after the changes to the system. In our case, the changes are enhancements, patches, or any configuration changes. Along with the changes, regression testing helps identify new faults that could be the results of the bug-fixes".*

I-10 *"After every release, we need to make sure that the previous functionalities are workings. We have to ensure that all bugs are fixed, and there are no new bugs introduced so that the old functionalities are working together with the new functionalities."*

I-11 *"Regression testing checks that after any changes, if the system is working? It is to test whether the system is working according to the mentioned functionality. Hence, regression testing is a kind of functional re-testing."*

The practitioners use existing system tests for regression testing. They run the regression tests after modifications or bug-fixing to see if the changes did not negatively affect the unchanged parts of the system. All the participants told us that they run a selected set of test cases while performing regression testing. However, the criteria for selecting the subset of test cases from the larger test suites vary among different perspectives. For instance, three of them (I-5, I-6, & I-7) select and prioritize test cases based on changes and their possible impact on the other functionalities. In some cases, practitioners (I-1, I-8, & I-9) told us that they have a predefined set of test cases applied to test if the basic functionality is working correctly after any system changes. Along with running the predefined set of test cases, they also run some sanity tests to ensure that other major functionalities are also working correctly. Three participants (I-2, I-3, & I-4) told us that they prioritize the test cases based on the importance of functionality. For instance, test cases that test the core functionalities will have the highest priority. One participant (I-10) revealed that they prioritize the test cases based on robustness, and one participant (I-11) told us that they prioritize the test cases based on the business impact.

Table 3.8: Regression testing goals – interviews results
I-1 to I-11 are the practitioners' IDs and (✓) means that the goal was defined by the respective practitioners.

Goal	**I-1**	**I-2**	**I-3**	**I-4**	**I-5**	**I-6**	**I-7**	**I-8**	**I-9**	**I-10**	**I-11**
G1: Increasing test suite's rate of fault detection				✓					✓		
G2: Early identification of critical faults	✓		✓						✓		
G3: Detection of faults related to changes			✓				✓				
G4:Coverage							✓	✓		✓	
G5: No or controlled fault slippage	✓	✓		✓		✓			✓		✓
G6: Confidence			✓	✓						✓	✓
G7: Test suite maintenance					✓		✓				
G8: Team's awareness of changes and overall application knowledge					✓			✓			

A | Regression testing goals defined by the interview participants

To know the practitioners' goals for regression testing, we asked: *"What are the goals that you think are essential to achieve success in regression testing?"* To grasp the right perception of the practitioners, many times we needed to rephrase this question from different angles. Table 3.8 presents the summary of regression testing goals identified from interviews, whereas, practitioners' definitions of the goals are presented in the paragraphs to follow.Table 3.8 presents the summary of regression testing goals identified from interviews, whereas practitioners' definitions for these goals are presented in the subsequent paragraphs.
Of the eleven, six practitioners defined **G5: No or controlled fault slippage** as their goal, with varying descriptions. The statements of the practitioners regarding this goal are:

I-1 *"The goal is that no fault with priority 1 (high-risk faults) should slip through. We want to make sure that the customer should not find any such fault."*

I-2 *"All test cases in regression test pack should be executed with 100% pass, the goal is that customers should not find any fault."*

I-4 *"There should be no priority 1 or priority 2 defects in the system while releasing it to the client."*

I-6 *"The goal is that we should not let a fault slip through that can break the existing application."*

I-9 *"We try to avoid hotfixes after release, or at least we try to reduce the number of hotfixes."*

I-11 *"We try to maintain a 100% success rate, we do not want any fault slippage to our customer."*

Four practitioners defined **G6: Confidence** as their goal. The practitioners want to be confident about the reliability and the reached quality of the product. The statement of the practitioners regarding confidence are as follow:

I-3 *"The goal is to increase the confidence in the reliability of the system under test at a faster rate, and this could be only done with opting smarter approaches for regression testing."*

I-4 *"With the regression testing, I want to be confident that nothing should be broken in the system under test."*

I-10 *"We want to be confident that it is safe enough to release the product to the customer."*

I-11 *"The goal is to gain the customers' confidence and trust, and this is only possible if we are confident that we have tested enough and it is safe to release the product."*

Three practitioners stated that **G2: Early identification of critical faults** is their goal, the practitioners perspective in this regard is:

I-1 *"We try to identify the high risk faults, early detection of critical faults is our goal."*

I-3 *"Early identification of critical bugs is our goal, we have to make it sure that there should be no severe faults in the key user functionalities."*

I-9 *“Critical bugs are show stopper. Severity detection of bugs and their early detection is one of our essential goals.”*

Three practitioners defined **G4: Coverage** as their success goal, the statements of the practitioners are:

I-7 *“What percentage of code is covered, and what percentage of test cases covered against defects found? It is an important parameter to assess the success in regression testing.”*

I-8 *“Whenever we get a new build, we have to make it sure that all basic functionalities have been covered.”*

I-10 *“Covering all robust test cases is an important goal.”*

G3: Detection of faults related to changes is the goal specified by the two practitioners, their statements are:

I-3 *“Early evaluation of changes and detection of faults related to changes is an essential goal in regression testing.”*

I-7 *“How many defects found based on changes and against changes is an important goal to be measured.”*

Two practitioners stated **G1: Increasing test suite rate of fault detection** is their goal. The practitioners’ perspective regarding defect detection is:

I-4 *“Early detection of bugs is an important goal, which saves time for developers and testers. What is the rate of fault detection is important.”*

I-9 *“An increased rate of fault detection provides you confidence about your underlying regression testing approach and strategy.”*

Two practitioners stated that **G7: Test suite maintenance** is their goal. Test suite maintenance is listed as a regression testing challenge in the existing studies [7, 29, 54, 55]. The practitioners' perspective regarding test suite maintenance is:

I-5 *“To keep the test suite updated all the time so that it really helps with future releases. A well-maintained test suite is always an essential requirement for your success in regression testing.”*

I-7 *“ To make it sure that we should not miss any issues, we have to keep the test suite updated. ”*

Two practitioners stated that **G8: Team's awareness of changes and overall application knowledge** has a significant impact on the success of regression testing. In our interview-based multi-case study [29], the practitioners highlighted it as a success criterion required to aid the achievement of various goals. The practitioners' perspective regarding this goal is:

I-5 *"To keep the team educated always with the new changes and with the overall application knowledge is important."*

I-8 *" What are the fixes developers have made in the newer version? We need to learn that. We have to review the release note of the old version and get aware of the product. Having knowledge of such things is crucial for the success of regression testing."*

B | Information needs and metrics defined by the interview participants

The next essential part of our investigations was to know the response of practitioners about information needs and metrics/measures to be used to achieve/evaluate the success in regression testing. We asked a series of questions for instance, we asked:

i. Do you measure or evaluate the goals?

ii. How do you measure?

iii. Which are the information-needs necessary to achieve the goals?

iv. Which metrics do you use to evaluate the success goals?

While responding to the question, "Do you measure or evaluate the goals?", the response of the interview participants was a mix of yes and no. The majority of them responded, yes, we do measure, a couple of participants straightforwardly said no, we do not measure, and some of the participants told us that to some extent, they analyze the results.

In response to the second question, "How do you measure?", one of the participants revealed that they are using an agile-based tracking system to track the fulfillment of the goals. Some participants narrated that they make guesses based on their experience and product knowledge. Whereas, a couple of participants are using defect count as a measure to evaluate their goals. They have a defined threshold to decide if they can release the product. For instance, defect rate per unit time and the number of critical defects vs. total defects are used to evaluate the success. Another measure that is being used is the ratio between the number of defects and test cases.

From the responses of the participants regarding the questions, *"Which are the information-needs necessary to achieve the goals?"* and *"Which metrics do you use to evaluate the success goals?"*. We learnt that companies are making some sort of assessments to evaluate the success in regression testing. Regarding information needs, five practitioners (I-1, I-2, I-5, I-6, and I-8) stated that the use of **requirements information** helps fulfill the regression testing goals. Four practitioners (I-3, I-4, I-5, and I-7) discussed the use of **test case execution/fault detection history**. Two practitioners (I-2 and I-5) highlighted the importance of **knowledge of changes** for success in regression testing, and one practitioner (I-9) said that they maintain the **code coverage information** to track the coverage of high-risk cases. Regarding metrics, three practitioners (I-3, I-4, and I-7) reported the use of **fault detection rate** (number of defects found). Three practitioners (I-2, I-5, and I-8) stated that they track the **test case failure rate** to evaluate success. Two practitioners (I-2 and I-9) said that they measure the **coverage rate** to evaluate the success goals. In the majority of the companies, judgments are made based on the guesses of practitioners. However, while making assessment guesses, the practitioners consider some real-time outcomes like the number of defects found vs. the number of test cases executed.

C | Interview participants' responses on metrics identified from literature

In the last part of the interview, we presented the metrics that we found from literature and asked them to see if they recognize these metrics and what is their opinion about the usefulness of these metrics for measuring the success. Six metrics got recognition from the interview participants (see Table 3.9). Three participants (I-3, I-8, and I-11) endorsed APFD, three participants (I-1, I-3, and I-11) endorsed APFDc, two participants (I-2 and I-8) voted for test case failure rate, and two (I-1 and I-8) endorsed the fault severity measure. One participant (I-8) endorsed BFCP, and one participant (I-7) opted for code coverage. Three participants (I-4, I-6, and I-10) stated that they were not aware of the metrics presented in the literature. In contrast, two participants (I-5 and I-9) said they do not use the metrics defined in the literature.

The perspective of individual practitioners who did not recognize the metrics identified from literature are presented here:

I-4 I am unaware of these metrics.

I-5 In actual projects, we would not be using any of the metrics defined in the literature.

I-6 We are not familiar with the metrics given in the literature.

I-7 Some of the metrics presented in the literature may not be applicable in some areas. However, code coverage is a metric that is measurable.

I-9 We do not consider the metrics given in the literature, but somehow we follow these metrics as summary.

I-10 I am unable to answer this question because I am unaware of these metrics.

Table 3.9: Literature metrics of regression testing, recognized by the interview participants

Metric	Endorsed by
APFD (Average percentage of faults detected)	I-3, I-8, I-11
APFDc (Cost-cognizant weighted average percentage of faults detected)	I-1, I-3, I-11
Test case failure rate	I-2, I-8
APFDD (Average percentage of fault dependency detected)	-
Fault severity measure	I-1, I-8
BFCP (Bug fixing change impact prediction)	I-8
Code coverage	I-7

Online questionnaire

To illustrate the research perspective, we used results from the literature. To highlight the industry perspective, we interviewed the practitioners representing nine companies from four different countries. We conducted interviews openly asking about goals, information needs, and metrics to check saturation (do we find more goals and metrics prior to surveying a larger set of people). Then we see yes, we got two new goals but did not learn anything new for the metrics. We also got more qualitative information here (deeper insights). From the interview results, we observed that practitioners have a different perspective of regression testing goals and metrics. Although the practitioners have their own goals, to some extent, they recognize the literature goals. However, almost half of them did not support the metrics/measures presented in the literature.

To have insight from a larger set, we opted for an online questionnaire based survey. To avoid any misinterpretations, we provided a brief description of each goal. We listed all goals that we found in the literature and interviews. We received 45 correct responses of the practitioners working in different roles and having different experiences. Figure 3.1 provides the detail of participating practitioners' roles and experience in the field of software testing. The most-reported role is QA engineer with (23 of 45) 51% of the participants, followed by the test lead with (9 of 45) 20% of the participants, test manager (6 of 45) 13%, test analyst (4 of 45) 9%, and test architect (3 of 45)

7%. The majority of the respondents (21 of 45) 47% have an experience between two to five years, followed by (11 of 45) 24% having more than ten years of experience, (10 of 45) 22% have an experience between five to ten years, and (3 of 45) 7% of the respondents have an experience between one to two years. Concerning the company size, most of the respondents represented large scale companies. As (27 of 45) 60% of the respondents are from companies with more than 1000 employees, (6 of 45)13% are from companies with 500 - 1000 employees, (8 of 45) 18% are from companies with 100- 500 employees, and (4 of 45) 9% are from companies with less than 100 employees.

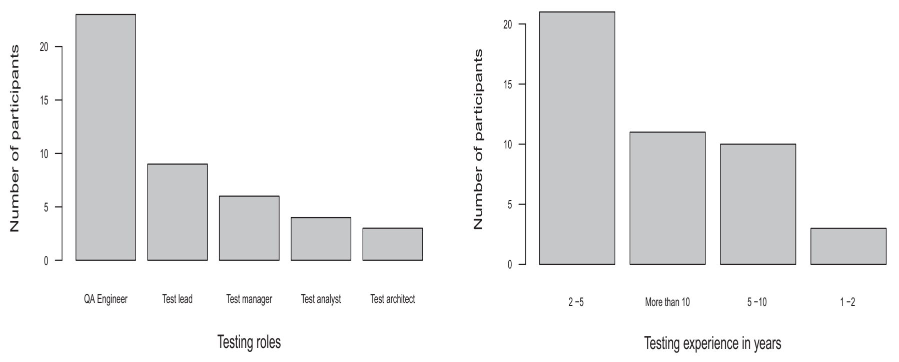

Figure 3.1: Role and testing experience of the survey respondents

The respondents are working on different domains, including accounting and finance, automobile systems, business services, embedded systems, Telecom, mobile applicant ions, and medical devices. Figure 3.2 presents the detail of product domain on which survey respondents are working. Regression testing is highly important for 58% of the respondents, important for 20%, and moderate for 22% of the respondents. Among the respondents 27 of 45 said that they implement selective regression testing (i.e., running a selected sub of test cases). Whereas 18 of 45 stated that they implement re-test all policy (i.e., running all test cases in the regression suite). Concerning product releases, the majority of respondents replied that they have multiple releases for their products every year. Only two of 45 respondents revealed that they have one release for their product per year.

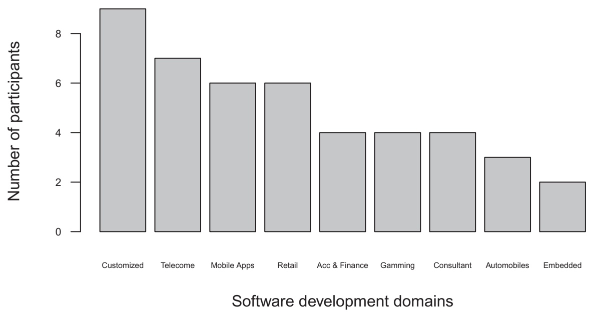

Figure 3.2: Software development domains on which survey respondents are working

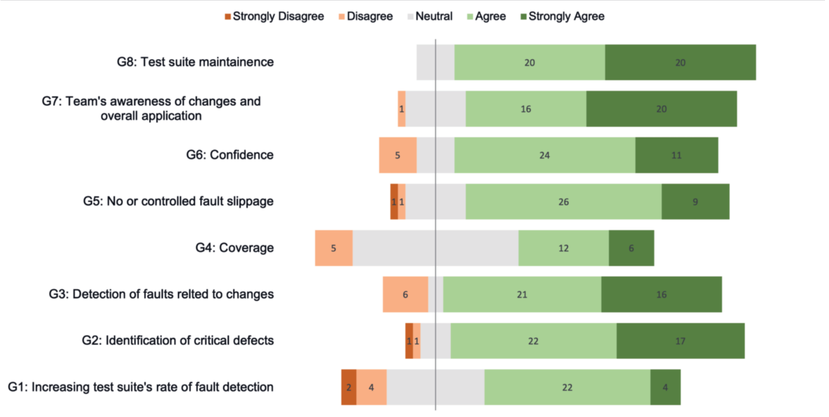

Figure 3.3: Regression testing goals – questionnaire results

A | Regression testing goals

Figure 3.3 presents the response of survey respondents on regression testing goals, and an overall mapping of goals, information needs, and metrics is presented in Table 3.10.

Of the 45 respondents, 26 (58%) practitioners selected **increasing test suite's rate**

of fault detection (G1) as their goal, 13 were neutral, and six opposed this option. 39 (87%) of the respondents were agreed that **early identification of critical faults** (G2), is a goal for regression testing, four opted to neutral, and two disagreed. 37 (82%) were agreed that the **detection of faults related to changes** (G3)is an essential goal for regression testing, two were neutral, and six disagreed with this option. Only 18 (40 %) chose **coverage** (G4) as their goal, 22 (49 %) chosen to remain neutral, and five disagreed. 35 (78%) were agree that **no or controlled fault slippage** (G5) is their goal, eight respondents opted to stay neutral, and only two respondents disagreed. 35 (78%) practitioners chosen **confidence** (G6) as their goal, five remained neutral, and five disagreed. Besides these goals, 89% of the survey respondents suggested that **test suite maintenance** (G7) is an efficient way contributes to the success of regression testing. Similarly, 80% of the respondents emphasized that the **team's awareness of changes and overall application knowledge** (G8) are the primary requirements for success in regression testing.

B | Information needs and metrics

In the survey questionnaire, we embedded the information needs and metrics with every goal. We also provided the free text space to allow the respondents to state any goal, information need, or metric which we may not have listed in the questionnaire. The survey respondents did not mention any new goals. However, they listed a few information needs and metrics other than those we listed in the questionnaire.

Information needs: The survey respondents have listed a few information needs. The most mentioned information need is the **requirements information** as it was listed against five different goals. For example, 73% of the respondents mentioned it against G8, 69% listed it against G7, 63% against G6, 42% against G5, and 5% mentioned it against G1. **Code changes** was listed as information need for two goals, 71% of the respondent mentioned it as information need to achieve G7, and 41% of the respondents thinks it is required to achieve G3. 57% of the respondents against G8 lists **past fault detection history**. Similarly, **fault dependence** is listed against G2 by 37% of the respondents. Along with the listed information needs, the survey respondents have stated their own information needs, including **business impact** against two goals G2 and G4, **domain knowledge** against G8, and **coverage of impacted modules** G3 and G5. Table 3.10 presents the detail of information needs along with the respective goals and metrics.

Metrics: From the interviews, we learned that practitioners do not use any metrics defined in the literature to evaluate the success goals. They mainly make guesses based on their experience and knowledge. Only six interview participants endorsed a

Table 3.10: GQM mapping of goals, information needs, and metrics – Survey

Goal	G-Response (%age)[1]	Information needs	IN-Response (%age)[2]	Metrics	M-Response (%age)[3]
G1: Increasing test suite's rate of fault detection	58%	Requirements information	5%	Test case failure rate	62%
				APFD	56%
				Code coverage metrics	26%
G2: Early identification of critical faults	87%	Business Impact	81%	Severity measure	77%
		Fault dependency	37%	Code coverage metrics	28%
				APFDc	23%
G3: Detection of faults related to changes	82%	Code changes	41%	BFCP	69%
		Coverage of impacted modules	77%	APFD	26%
				Test case failure rate	9%
G4: Coverage	40%	Business impact	58%	Code coverage metrics	38%
G5: No or controlled fault slippage	78%	Requirements information	42%	Code coverage metrics	42%
		Coverage of impacted modules	61%		
G6: Confidence	78%	Requirements information	63%	Test case failure rate	63%
				Code coverage metrics	45%
				Pass percentage of the total scenarios executed	5%
G7: Test suite maintenance	95%	Requirements information	69%		
		Code changes	71%		
G8: Team's awareness of changes & overall application knowledge	80%	Domain knowledge	80%		
		Requirements information	73%		
		Past fault detection history	57%		

1. G-Response (%age)= Practitioners' response for goals as sum of percentages of Strongly agree and Agree.
2. IN-Response (%age)= The percentage is calculated against the number of respondents who have selected the corresponding goal.
3. M-Response (%age)= The percentage is calculated against the number of respondents who have selected the corresponding goal.

few metrics defined in the literature. Therefore, it was interesting to see if the survey respondents recognize the metrics given in the literature. Using the results obtained from the literature, we listed a set of metrics for each goal. The respondents could select the one, many, or none for the goals they opted to agree to or strongly agree. They were provided with the free text space to provide their own choices if different from those provided. The majority of practitioners listed metrics/measures against each goal. They selected varying choices of metrics against each goal. However, the majority preferred to choose from the list of given options. A few of the respondents provided metrics other than the given list. For instance, two respondents mentioned **pass percentage of the total number of scenarios executed** as a metric to evaluate G6. Table 3.10 presents the complete set of metrics along with the respective goals. Three metrics **test case failure rate**, **APFD**, and **code coverage metrics** were respectively selected by 62%, 56%, and 26% of the respondents for G1. The metrics **severity measure**, **code coverage metrics**, and **APFDc** were selected for G2 by 77%, 28%, and 23% respondents. **BFCP**, **APFD**, and **Test case failure rate** were listed by 69%, 26%, and 9% respondents to evaluate G3. **Code coverage metrics** is listed by 38% of the respondents to evaluate G4, and 42% of the respondents listed it to evaluate G5. To have confidence (G6) in the regression testing 63% respondents suggested to measure the test case failure rate, 45% opted to evaluate the code coverage metrics, and 9% suggests to evaluate the pass percentage of total scenarios executed. For G7 and G8, the survey respondents did not suggest any metrics.

3.4.3 Using GQM to integrate research and practice perspectives

From the interview results (see Table 3.8), it is evident that the majority of the practitioners defined more than one goal for success in regression testing. The same trend was observed in the responses to the online questionnaire (see Figure 3.3). Although the authors of most techniques proposed in the literature focus on a single goal, they mention other goals. It reflects that only a single goal can not guarantee success. Furthermore, the identified goals have a certain level of interdependence. For instance, increasing test suite's coverage (G4) without increasing fault detection rate (G1) will be useless. Besides its maximum coverage (G4) and a reasonable fault detection rate (G1), if a test suite misses the critical faults (G2), it will not solve the purpose. Similarly, Controlling fault slippage (G5) requires that the underlying regression testing technique should be able to select/prioritize the fault-revealing test cases (G1), it should identify all critical faults (G2), and cover all changed/impacted modules (G3, G4). Finally, to be confident (G6) about the success in regression testing, testers want to ensure that no critical fault is being slipped through (G5) to the customer. This highlights the need for a holistic map of goals and associated selection/prioritization strategies.

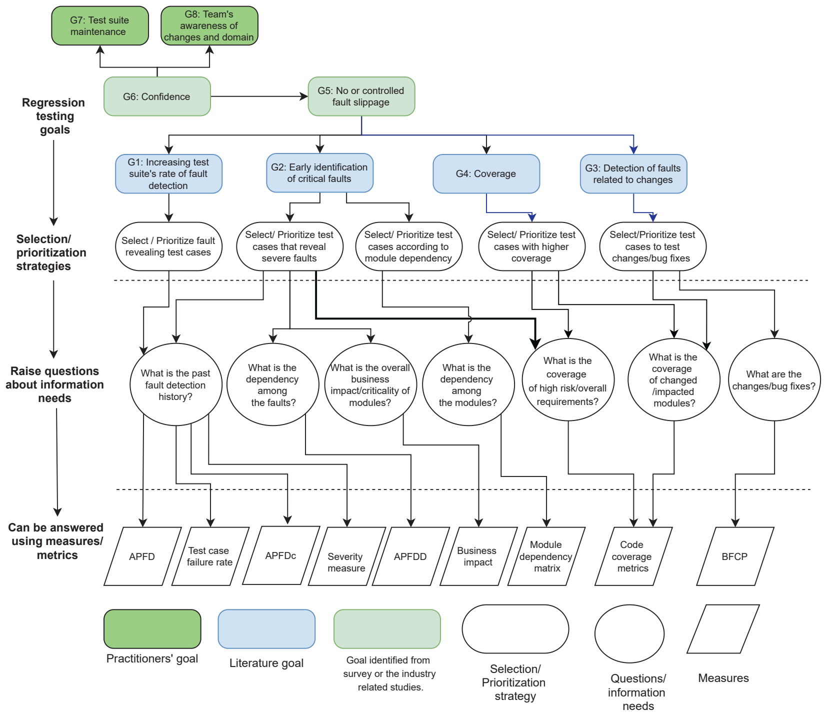
G7: Test suite maintenance
G8: Team's awareness of changes and domain
Regression testing goals
G6: Confidence
G5: No or controlled fault slippage
G1: Increasing test suite's rate of fault detection
G2: Early identification of critical faults
G4: Coverage
G3: Detection of faults related to changes
Selection/ prioritization strategies
Select / Prioritize fault revealing test cases
Select/ Prioritize test cases that reveal severe faults
Select/ Prioritize test cases according to module dependency
Select/ Prioritize test cases with higher coverage
Select/Prioritize test cases to test changes/bug fixes
Raise questions about information needs
What is the past fault detection history?
What is the dependency among the faults?
What is the overall business impact/criticality of modules?
What is the dependency among the modules?
What is the coverage of high risk/overall requirements?
What is the coverage of changed /impacted modules?
What are the changes/bug fixes?
Can be answered using measures/ metrics
APFD
Test case failure rate
APFDc
Severity measure
APFDD
Business impact
Module dependency matrix
Code coverage metrics
BFCP
Practitioners' goal
Literature goal
Goal identified from survey or the industry related studies.
Selection/ Prioritization strategy
Questions/ information needs
Measures

Figure 3.4: A GQM based model of regression testing goals, information needs, and metrics.

Using GQM approach, we have created a model that maps goals, information needs, and metrics (see Figure 3.4). The model provides a combined view of the literature and survey findings, and it would be helpful for practitioners in adopting regression testing strategies suitable to their context. It will also help researchers to propose new techniques tailored to the industry's needs.

We have classified the goals into three categories. Practitioners' goals (identified from the survey) are shown in dark green, goals shown in light green represent goals identified from studies conducted in an industry context and from the survey. Goals in light blue represent goals identified from studies proposing or evaluating regression testing techniques. Our model gives confidence (G6) a central place and suggests controlling the fault slippage (G5). To control the fault slippage, ensure the early identification of critical faults (G2), detect the faults related to changes and bug fixes (G3), ensure that all essential paths have been covered (G4), and try to maximize the rate of fault detection (G1). Furthermore, two goals test suite maintenance (G7) and team's awareness of changes and overall application knowledge (G8) identified from the survey have been placed on top and considered essential criteria to be confident in the regression testing process. The model is created based on the following proposition:

*To achieve a goal **"G"**, calculate metric **"M"** using relevant information needs **"IN"** and opt for the respective regression testing strategy.*

Using the model, we can derive the following guidelines:

1. To increase test suite's rate of fault detection (G1), use the APFD value calculated using fault detection history and select/prioritize the fault revealing test cases.
2. To achieve the early detection of critical faults (G2), use the APFDc and severity measure calculated using test execution/fault detection history and select/prioritize test cases that reveal severe faults.
3. To achieve the early detection of critical faults (G2), use the APFDD value calculated using fault dependency matrix and select/prioritize test cases that reveal leading faults.
4. To detect the faults related to changes (G3), use the BFCP calculated using the information of changes/fixes and select test cases that test changes.
5. To achieve increased coverage (G4), use coverage metrics calculated using the coverage information of test cases and select test cases with higher coverage.

Application of these guidelines would not be difficult for practitioner's. For instance, if practitioners want to achieve the goal increasing test suite's rate of fault detection (G1), they can use the past execution history of the test cases to measure the rate of fault detection, and based on these metrics, they can select/prioritize the fault revealing test cases. Similarly, to assess the achievement of a goal, the practitioners can use the real-time test results to see if they have achieved their goal(s). For example, to measure the achievement of increasing test suite's rate of fault detection (G1), using the fault detection data for each test case, they can calculate the average percentage of fault detection (APFD).

3.5 Discussion

In this study we have investigated the literature and the industry perspectives on regression testing goals, information needs, and metrics. From the findings of 33 studies selected initially and 11 additional systematic reviews, we learnt that regression testing goals are not specific to product domain, development environment, or technique type. For example, increasing test suite's rate of fault detection is mentioned as a goal of regression testing techniques proposed for web applications [74], continuous integration environment [78], and of the techniques where authors did not highlight any domain or environment. Similarly, authors of machine learning [71], history-based, search-based, and coverage-based [72, 76] regression testing techniques use increasing test suite's rate of fault detection as one of the goals for their techniques. From the findings of survey, we learnt that the practitioners have their preferences for regression testing goals. However, we can not conclude that this variation of choices depends on the product domain or development environment. Besides preferring specific goals, most survey respondents endorsed the regression testing goals found in the literature. The following paragraphs provide a brief discussion on the study's findings.

The performance of a test process could be gauged by the number of faults detected during the process. *“Increasing test suite's rate of fault detection”*, is one of the regression testing goals identified in this study. The goal appeared in various studies, and the survey respondents also identified this goal. This goal corresponds to the effectiveness of regression testing and could be achieved by adding those test cases in the regression test suite, which have more fault detection capability. Using fault detection history and requirements information could help in selecting the fault-revealing test cases. The benefit of increasing test suite's rate of fault detection early in the testing process is quicker feedback on the system under test, early start of debugging, and ultimately reduced testing time and cost [31]. Some of the faults are crucial and can break the product. Finding such faults early in the regression testing process is critical. *“Early identification of critical faults”* is the part of the findings of this study. We found this goal in the literature, and the survey respondents also identified it. Critical faults could be of two types, 1) faults that affect the core functionality of the system under test, 2) leading faults, the faults that cause the other faults to appear later in the operations. Uncovering the critical faults needs to identify the modules that are badly affected and then prioritizing the test cases which cover the identified modules [23]. If a testing process fails to identify such faults early, there could be adverse outcomes. More precisely, the overall goal could be to identify more severe leading faults early in the testing process [2]. Early evaluation of changes could help identify critical faults, especially faults related to changes and bug fixing. *Detection of faults related to changes* is also identified as a regression testing goal in this study. Achieving this goal requires selecting the test cases based on the changes/bug fixes in the system. Detection of these faults is crucial, especially for scenarios like fixing critical bugs in an emergency and running tests under tight time schedules [19].

In selective regression testing, an important aspect that a testing practitioner considers is how much code would be covered by the selected test cases [38]. One of the interview participants stated that *"Whenever we get a new build, we try to find the defects based on the changes and find the percentage of code covered against these newly found defects. So code coverage can also be considered as a part of success criteria. It is also imperative because it reduces the effort and cost of regression testing."* The coverage alone could not be a goal of any regression testing process because maximizing coverage can not guarantee fault detection. Instead, it will help in minimizing the test execution time and cost. The coverage can be evaluated using code coverage metrics like method coverage, statement coverage, and branch coverage. The techniques using coverage-based metrics could additionally use the program mutation to measure the fault exposing potential. For example, Kwon et al. [68] used mutation score to determine the fault detection capability of their technique in the absence of test execution/fault history. Program mutation refers to introducing a small change in the source code, and the changed version would be referred to as a mutant. A mutant is killed if a test case can identify it. The number of mutants killed by a test case is referred to as the mutation score of the test case, and it could be the reflection of the fault detection capability of that test case [9]. Although various authors have been using mutation testing to evaluate their techniques, applying mutation testing at an industrial scale is challenging because of the time required to execute each mutant against the test suite under evaluation [69].

While releasing a product to the customer, the team wants to ensure that customer should not find any fault after release. In our previous study [4], the practitioners labeled this goal as *"no-fault slippage"*. We argue that setting a goal of no-fault slippage does not mean that there would be no fault slip through. Therefore more appropriate would be to set a goal of controlling or minimizing the fault slip through. The majority of interview participants and survey respondents highlighted that controlling fault slippage to the customer is one of the essential success goals of regression testing. Fault slippage is the primary reason for higher rework costs. Damm et al. [14] introduced a metric called fault slip through (FST) to determine the faults that would have been more cost-effective to find earlier in the testing process. Keeping fault slippage rate as low as possible may help the managers decide about releasing the product, provided if they are confident that no known fault is supposed to be slipped-through [4, 29]. This study suggests that to control the fault slippage the practitioners need to focus on the other goals (G1, G2, G3, & G4).

Practitioners frequently use the term confidence concerning their success in regression testing. They want to be confident about their regression testing process that they have uncovered and fixed all such bugs that can break the system under test. *"Confidence"* appeared as a regression testing goal in the studies conducted in the industry context. In a focus group workshop [4], the practitioners identified ten essential questions if answered correctly, then a tester can be confident about the underlying regression testing process. The questions are regarding the changes to the system, experience of the testing team, coverage of critical parts, testing of modifications, and test outcomes. Considering the finding of our previous work and discussion with the practitioners in the current study, we suggest that confidence is a subjective goal, and it can subsume various goals, depending upon the perception of practitioners involved. To be confident, the practitioners must achieve the other measurable goals, for instance, no or controlled fault slippage, early detection of critical faults, and detection of faults related to changes.

Test suite maintenance is another goal mentioned by most of the survey participants. It refers to adding new test cases to test the changes and delete the test cases that have become irrelevant/obsolete because of changes in the requirements. For success in regression testing, another essential aspect is the knowledge and experience of the team members. The knowledge refers to the domain, requirements, and changes of the system under test. Test suite maintenance and the team's knowledge are essentially required to be confident in the regression testing process. However, in practice, test suite maintenance and educating team members are challenges for the practitioners working with large-scale systems because of the tight deadlines. The practitioners have to perform too much testing in short span of time [29].

While interacting with the practitioners during this and our past studies [4, 29, 43], we learned that practitioners do not evaluate the achievement of the goals as they do not have any mechanism to follow the goals' achievement. Instead, they rely on expert judgment to guess the achievement of their goals. However, making a judgment without a formal mechanism may negatively impact the outcomes. Moreover, without a formal mechanism, the practitioners may overlook some essential aspects while making assessments/judgments [62]. As a step forward, we have proposed a GQM model to guide the practitioners to follow the goals. However, better information maintenance strategies are required to ensure achieving/evaluating regression testing goals. The practitioners are aware of this as they recognized that information maintenance is a challenge in the companies, and there is a desire to improve the information maintenance strategies [29]. We argue that the GQM could potentially be used in many organizations that conduct regression testing. What the organizations would have to do is to prioritize the goals for their specific context. They can add their goals that are not yet captured in the model. Further, the organizations have to choose the metrics they wish to use. One factor here is the cost of collecting the metrics, which may vary with the test framework used (e.g., lacking the ability to collect the measures automatically). Thus, depending on the context, different measures would be chosen. The companies could use the method in [70]. Having a starting point (our model) will help them use the method and select relevant measures.

The validation of GQM is not part of this work, and it is a proposal based on the literature and survey findings. However, in future, we are aiming to extend this proposal and validate it from industry practitioners. The evaluation will entail prioritizing the goals to decide on measures. As the GQM is not static, new goals and measures will be identified with further contexts and developments. We plan to create guidelines for updating and extending the GQM with forthcoming studies. The current GQM serves as a baseline for people to use. With further usage, it will become completer and more comprehensive. In the future, we also would like to investigate the importance of different goals, questions and measures depending on context. This allows practitioners to select the right metrics. We plan to follow the approach suggested by Gencel et al. [70].

3.6 Summary and conclusions

The study explored the regression testing goals, information needs, and metrics from the research and practice perspectives. The quantitative and qualitative data is collected using the literature review, interviews, and online questionnaire. The purpose was to present an integrated view of literature and industry perspectives on regression testing goals. To present the literature perspective of regression testing goals **(RQ1)** we have conducted a literature review of 33 research papers. In addition, we also looked at the 11 systematic reviews published between 2017 –2022. Except for a couple of studies explicitly focusing on regression testing goals, most of the studies discuss regression testing goals while proposing, evaluating, or reviewing regression testing techniques. From the literature, we found six regression testing goals, two of them, are identified from the studies representing the practitioners' perspective. Most of the authors evaluate their techniques in terms of fault detection rate by using the APFD metric. The information needs mentioned to evaluate the fault detection rate are “requirement information”, “test case execution/fault detection history”, and “coverage based information”. Other goals mentioned in the literature are “early identification of critical faults”, “detection of faults related to changes”, and “coverage”. The goals identified from the industry-related literature are “no or controlled fault slippage” and “confidence”. A complete mapping of regression testing goals, information needs, and metrics found from the literature is presented in Table 3.6. Till now, there is a lack of literature review on the topic of regression testing goals. This study provides a step forward in this context.

To present the practitioners' perspective **(RQ2)** we conducted a survey comprising 11 interviews and 45 responses to an online questionnaire. We observed that the interview participants have varying perspectives on regression testing goals. In the overall survey results, we learned that, besides recognizing the literature goals, the practitioners emphasize on their own goals including i) test suite maintenance and ii) team's awareness of changes and overall application knowledge. They recognized only a few of the information needs and metrics identified from the literature. The practitioners also suggested some information needs and metrics including, domain knowledge, business impact, coverage of impacted modules, and pass percentage of executed scenarios. A complete list of goals, information needs, and corresponding metrics selected/defined by the survey respondents is presented in Tables 3.8, 3.10.

To compare the research and practice perspective on regression testing goals, we have created a goals-questions-metrics (GQM) model **(RQ3)**. The model presents an integrated view of the literature and the practitioners' perspectives. Researchers can utilize this model to align their research closer to the industry context while proposing the new regression testing techniques. Similarly, the practitioners can utilize this model to better follow the goal-based regression testing strategies. Based on the findings of our study, we suggest that researchers should consider multi-objective strategies while proposing and evaluating regression testing techniques. They need to incorporate no or controlled fault slippage (G6) as a primary goal of the proposed techniques. It will provide confidence to the practitioners that applying such techniques will help control the fault slippage.

The results provide a basis for future research on the evaluation of regression testing, and the GQM model presented in this study is a step forward in this direction. Furthermore, this study's findings will help the researchers propose new methods to align their research with the practitioners' regression testing goals. Hence, contributing to the adoption of research on regression testing in the industry. The identified goals and metrics will also help the practitioners to access the new techniques while adopting them. The metrics listed in this study can allow the practitioners to try out new metrics since many of these metrics are not incorporated in the industry. This study's overall contribution would be reducing the gap in the research and practice of regression testing.

3.7 References

[1] H. Do and G. Rothermel, "On the use of mutation faults in empirical assessments of test case prioritization techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733–752, 2006.

[2] S. Jafrin, D. Nandi, and S. Mahmood, "Test case prioritization based on fault dependency," *International Journal of Modern Education and Computer Science*, vol. 8, no. 4, p. 33, 2016.

[3] X. Tang, S. Wang, and K. Mao, "Will this bug-fixing change break regression testing?" in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2015*. IEEE, 2015, pp. 1–10.

[4] N. M. Minhas, K. Petersen, N. Ali, and K. Wnuk, "Regression testing goals-view of practitioners and researchers," in *24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*. IEEE, 2017, pp. 25–32.

[5] M. A. Askarunisa, M. L. Shanmugapriya, and D. N. Ramaraj, "Cost and coverage metrics for measuring the effectiveness of test case prioritization techniques," *INFOCOMP*, vol. 9, no. 1, pp. 43–52, 2010.

[6] R. H. Rosero, O. S. Gómez, and G. Rodríguez, "15 years of software regression testing techniques–a survey," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 05, pp. 675–689, 2016.

[7] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *International Conference on Product Focused Software Process Improvement*. Springer, 2010, pp. 3–16.

[8] R. Wang, S. Jiang, D. Chen, and Y. Zhang, "Empirical study of the effects of different similarity measures on test case prioritization," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[9] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[10] I. Kayes, S. Islam, and J. Chakareski, "The network of faults: a complex network approach to prioritize test cases for regression testing," *Innovations in Systems and Software Engineering*, vol. 11, no. 4, pp. 261–275, 2015.

[11] S. Nayak, C. Kumar, and S. Tripathi, “Effectiveness of prioritization of test cases based on faults,” in *3rd International Conference on Recent Advances in Information Technology (RAIT), 2016*. IEEE, 2016, pp. 657–662.

[12] X. Zhao, Z. Wang, X. Fan, and Z. Wang, “A clustering-bayesian network based approach for test case prioritization,” in *IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), 2015*, vol. 3. IEEE, 2015, pp. 542–547.

[13] G. Chaurasia, S. Agarwal, and S. S. Gautam, “Clustering based novel test case prioritization technique,” in *IEEE Students Conference on Engineering and Systems (SCES), 2015*. IEEE, 2015, pp. 1–5.

[14] L.-O. Damm, L. Lundberg, and C. Wohlin, “Faults-slip-through—a concept for measuring the efficiency of the test process,” *Software Process: Improvement and Practice*, vol. 11, no. 1, pp. 47–59, 2006.

[15] M. I. Kayes, “Test case prioritization for regression testing based on fault dependency,” in *3rd International Conference on Electronics Computer Technology (ICECT), 2011*, vol. 5. IEEE, 2011, pp. 48–52.

[16] S. Elbaum, D. Gable, and G. Rothermel, “Understanding and measuring the sources of variation in the prioritization of regression test suites,” in *Proceedings. Seventh International Software Metrics Symposium, 2001. METRICS 2001.* IEEE, 2001, pp. 169–179.

[17] S. Elbaum, A. G. Malishevsky, and G. Rothermel, “Test case prioritization: A family of empirical studies,” *IEEE transactions on software engineering*, vol. 28, no. 2, pp. 159–182, 2002.

[18] G. Rothermel and M. J. Harrold, “Analyzing regression test selection techniques,” *IEEE Transactions on software engineering*, vol. 22, no. 8, pp. 529–551, 1996.

[19] A. Srivastava and J. Thiagarajan, “Effectively prioritizing tests in development environment,” in *ACM SIGSOFT Software Engineering Notes*, vol. 27. ACM, 2002, pp. 97–106.

[20] S. Yoo, M. Harman, P. Tonella, and A. Susi, “Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge,” in *Proceedings of the eighteenth international symposium on Software testing and analysis*. ACM, 2009, pp. 201–212.

[21] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, “An empirical study of the effect of time constraints on the cost-benefits of regression testing,” in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM, 2008, pp. 71–82.

[22] S. Elbaum, A. Malishevsky, and G. Rothermel, “Incorporating varying test costs and fault severities into test case prioritization,” in *Proceedings of the 23rd International Conference on Software Engineering*. IEEE Computer Society, 2001, pp. 329–338.

[23] H. Kumar and N. Chauhan, “A module coupling slice based test case prioritization technique,” *IJ Modern Education and Computer Science*, vol. 7, no. 7, pp. 8–16, 2015.

[24] W. E. Wong, J. R. Horgan, S. London, and H. Agrawal, “A study of effective regression testing in practice,” in *Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on*. IEEE, 1997, pp. 264–274.

[25] S. Yoo and M. Harman, “Using hybrid algorithm for pareto efficient multi-objective test suite minimisation,” *Journal of Systems and Software*, vol. 83, no. 4, pp. 689–701, 2010.

[26] G. Wikstrand, R. Feldt, J. K. Gorantla, W. Zhe, and C. White, “Dynamic regression test selection based on a file cache–an industrial evaluation,” in | *2009 International Conference on Software Testing Verification and Validation*. IEEE, 2009, pp. 299–302.

[27] S. Kim, T. Zimmermann, E. J. Whitehead Jr, and A. Zeller, “Predicting faults from cached history,” in *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 2007, pp. 489–498.

[28] M. J. Arafeen and H. Do, “Test case prioritization using requirements-based clustering,” in *IEEE Sixth International Conference on Software Testing, Verification and Validation (ICST), 2013*. IEEE, 2013, pp. 312–321.

[29] N. M. Minhas, K. Petersen, J. Börstler, and K. Wnuk, “Regression testing for large-scale embedded software development – exploring the state of practice,” *Information and Software Technology*, vol. 120, p. 106254, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584919302721

[30] S. Yoo and M. Harman, “Pareto efficient multi-objective test case selection,” in *Proceedings of the 2007 international symposium on Software testing and analysis*. ACM, 2007, pp. 140–150.

[31] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, “Cost-cognizant test case prioritization,” Technical Report TR-UNL-CSE-2006-0004, University of Nebraska-Lincoln, Tech. Rep., 2006.

[32] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, “Test case prioritization: An empirical study,” in *Proceedings of IEEE International Conference on Software Maintenance, 1999.(ICSM’99).* IEEE, 1999, pp. 179–188.

[33] T. Muthusamy and K. Seetharaman, “A new effective test case prioritization for regression testing based on prioritization algorithm,” *International Journal of Applied Information Systems (IJAIS)*, vol. 6, no. 7, pp. 21–26, 2014.

[34] B. Jiang, Z. Zhang, W. K. Chan, and T. Tse, “Adaptive random test case prioritization,” in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2009, pp. 233–244.

[35] A. S. Acharya, A. Prakash, P. Saxena, and A. Nigam, “Sampling: Why and how of it,” *Indian Journal of Medical Specialties*, vol. 4, no. 2, pp. 330–333, 2013.

[36] H. Do, G. Rothermel, and A. Kinneer, “Prioritizing junit test cases: An empirical assessment and cost-benefits analysis,” *Empirical Software Engineering*, vol. 11, no. 1, pp. 33–70, 2006.

[37] A. Pravin and S. Srinivasan, “S. srinivasan:—effective test case selection and prioritization in regression testing,” *Journal of Computer Science*, 2013.

[38] G. Rothermel and M. J. Harrold, “A safe, efficient regression test selection technique,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, no. 2, pp. 173–210, 1997.

[39] E. Engström, P. Runeson, and G. Wikstrand, “An empirical evaluation of regression testing based on fix-cache recommendations,” in *Third International Conference on Software Testing, Verification and Validation (ICST), 2010.* IEEE, 2010, pp. 75–78.

[40] M. Skoglund and P. Runeson, “A case study of the class firewall regression test selection technique on a large scale distributed software system,” in *International Symposium on Empirical Software Engineering, 2005.* IEEE, 2005, pp. 10–pp.

[41] L. White and B. Robinson, "Industrial real-time regression testing and analysis using firewalls," in *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.* IEEE, 2004, pp. 18–27.

[42] I. Eusgeld, F. Freiling, and R. H. Reussner, *Dependability Metrics: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, October 5-November 1, 2005, Advanced Lectures*. Springer, 2008, vol. 4909.

[43] N. bin Ali, E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, S. Kunze, and M. Varshosaz, "On the search for industry-relevant regression testing research," *Empirical Software Engineering*, pp. 1–36, 2019.

[44] J. Linaker, S. M. Sulaman, M. Höst, and R. M. de Mello, "Guidelines for conducting surveys in software engineering v. 1.1," *Lund University*, 2015.

[45] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 38.

[46] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[47] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[48] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research: part 3: constructing a survey instrument," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 2, pp. 20–24, 2002.

[49] B. Kitchenham and S. L. Pfleeger, "Principles of survey research part 4: questionnaire evaluation," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 3, pp. 20–23, 2002.

[50] ——, "Principles of survey research: part 5: populations and samples," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 5, pp. 17–20, 2002.

[51] A. Lacey and D. Luff, *Qualitative data analysis*. Trent focus Sheffield, 2001.

[52] V. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, vol. 2, no. 1994, pp. 528–532, 1994.

[53] H. Srikanth, L. Williams, and J. Osborne, “System test case prioritization of new and regression test cases,” in *2005 International Symposium on Empirical Software Engineering, 2005.* IEEE, 2005, pp. 10–pp.

[54] M. J. Harrold and A. Orso, “Retesting software during development and maintenance,” in *Proceedings of the Frontiers of Software Maintenance Conference*, 2008, pp. 99–108.

[55] D. Brahneborg, W. Afzal, and A. Čaušević, “A pragmatic perspective on regression testing challenges,” in *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017, pp. 618–619.

[56] R. Kavitha and N. Sureshkumar, “Test case prioritization for regression testing based on severity of fault,” *International Journal on Computer Science and Engineering*, vol. 2, no. 5, pp. 1462–1466, 2010.

[57] S. R. Dalal and A. A. McIntosh, “When to stop testing for large software systems with changing code,” *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 318–323, 1994.

[58] B. Zachariah, “Optimal stopping time in software testing based on failure size approach,” *Annals of Operations Research*, vol. 235, no. 1, pp. 771–784, 2015.

[59] A. Jangra, G. Singh, C. Kant *et al.*, “When to stop testing,” in *International Conference on High Performance Architecture and Grid Computing*. Springer, 2011, pp. 626–630.

[60] A. K. Shrivastava and N. Sachdeva, “Generalized software release and testing stop time policy,” *International Journal of Quality & Reliability Management*, 2019.

[61] P. Kapur, A. Shrivastava, and O. Singh, “When to release and stop testing of a software,” *Journal of the Indian Society for Probability and Statistics*, vol. 18, no. 1, pp. 19–37, 2017.

[62] M. Usman, K. Petersen, J. Börstler, and P. S. Neto, “Developing and using checklists to improve software effort estimation: A multi-case study,” *Journal of Systems and Software*, vol. 146, pp. 286–309, 2018.

[63] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for software engineering research,” in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[64] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Tech. Rep., 2007.

[65] Y. Lu, Y. Lou, S. Cheng, L. Zhang, D. Hao, Y. Zhou, and L. Zhang, “How does regression test prioritization perform in real-world software evolution?” in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 535–546.

[66] J. Chi, Y. Qu, Q. Zheng, Z. Yang, W. Jin, D. Cui, and T. Liu, “Relation-based test case prioritization for regression testing,” *Journal of Systems and Software*, vol. 163, p. 110539, 2020.

[67] A. Bertolino, A. Guerriero, B. Miranda, R. Pietrantuono, and S. Russo, “Learning-to-rank vs ranking-to-learn: Strategies for regression testing in continuous integration,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1–12.

[68] J.-H. Kwon, I.-Y. Ko, G. Rothermel, and M. Staats, “Test case prioritization based on information retrieval concepts,” in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 19–26.

[69] L. Chen and L. Zhang, “Speeding up mutation testing via regression test selection: An extensive study,” in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018, pp. 58–69.

[70] Ç. Gencel, K. Petersen, A. A. Mughal, and M. I. Iqbal, “A decision support framework for metrics selection in goal-based measurement programs: Gqm-dsfms,” *Journal of Systems and Software*, vol. 86, no. 12, pp. 3091–3108, 2013.

[71] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, “Test case selection and prioritization using machine learning: a systematic literature review,” *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–43, 2022.

[72] M. Rehan, N. Senan, M. Aamir, A. Samad, M. Husnain, N. Ibrahim, S. Ali, and H. Khatak, “A systematic analysis of regression test case selection: A multi-criteria-based approach,” *Security and Communication Networks*, vol. 2021, 2021.

[73] M. S. Abdul Manan, D. N. Abang Jawawi, and J. Ahmad, “A systematic literature review on test case prioritization in combinatorial testing,” in *2021 The 5th International Conference on Algorithms, Computing and Systems*, 2021, pp. 55–61.

[74] M. Hasnain, I. Ghani, M. F. Pasha, C. H. Lim, and S. R. Jeong, "A comprehensive review on regression test case prioritization techniques for web services," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 14, no. 5, pp. 1861–1885, 2020.

[75] M. d. C. de Castro-Cabrera, A. García-Dominguez, and I. Medina-Bulo, "Trends in prioritization of test cases: 2017-2019," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 2005–2011.

[76] M. Khatibsyarbini, M. A. Isa, D. N. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," *Information and Software Technology*, vol. 93, pp. 74–93, 2018.

[77] H. de S. Campos Junior, M. A. P. Araújo, J. M. N. David, R. Braga, F. Campos, and V. Ströele, "Test case prioritization: A systematic review and mapping of the literature," in *Proceedings of the 31st Brazilian Symposium on Software Engineering*, 2017, pp. 34–43.

[78] J. A. P. Lima and S. R. Vergilio, "Test case prioritization in continuous integration environments: A systematic mapping study," *Information and Software Technology*, vol. 121, p. 106268, 2020.

[79] A. Rahmani, S. Ahmad, I. E. A. Jalil, and A. P. Herawan, "A systematic literature review on regression test case prioritization," *International Journal of Advanced Computer Science and Applications*, vol. 12, pp. 253–267, 2021.

[80] A. Samad, H. Mahdin, R. Kazmi, and R. Ibrahim, "Regression test case prioritization: A systematic literature review," *International Journal of Advanced Computer Science and Applications*, vol. 12, pp. 655–663, 2021.

[81] S. Dalal, K. Solanki *et al.*, "Challenges of regression testing: A pragmatic perspective." *International Journal of Advanced Research in Computer Science*, vol. 9, no. 1, 2018.

Appendix A

Interview guide

Introduction

Personal introduction The lead Interviewer will tell the participant about research team, their background and training, and their research interest in regression testing.

Study goal The lead interviewer explains the study goal to the participant.

Goal: The goal of the study is to compare the literature and practitioners' perspectives on regression testing goals, needs, and metrics. The purpose this interview is to know that which are the regression testing goals that practitioners set for their success in regression testing. Further, the interviewers are interested to investigate if the practitioners measure their success in regression testing. The following points are the focus of the interview.

- ○ Regression testing goals,
- ○ Metrics to assess the goals,
- ○ Information needs used to compute the metrics,

Benefit: This study will provide the basis for proposing a mapping of the selected goals with the information needs and required metrics. The said mapping will help the practitioners in selective regression testing, it will also help the testing researchers to align their research to the industry needs. We believe your opinion is valuable. This investigation gives you (interviewee) a chance to contribute to the improvement of the regression testing research and practice.

Interview process Interviewer describes the overall process, that how the interview will take place.

◦ *Interview duration:* The interview will be completed in about 30 minutes time.

◦ *Interview questions:* There may be some questions that the interviewee perceives as not suitable or challenging to answer. It is possible that a question appropriate for one person may not be ideal for the other.

◦ *Counter questions:* The interviewee may feel free to ask counter questions for the clarification of an interview question and can disagree with any statement of the interviewer.

◦ *Answers:* The interview participants need not worry about their answers, as we can not rate any answer as correct or incorrect. We expect they will answer the questions based on their knowledge and experience.

Respondent background

In this section, the interviewers are interested to know about the participant's professional background, organizational role and responsibilities.

Question 1: Could you please briefly describe your professional background?

◦ For how long you have been working with this organization?
◦ What is your role in the organization?
◦ For how long you have been taking up this role?
◦ What kind of products does your organization deal with?

Question 2: How will you define your expertise?

◦ Software Engineering,
◦ Software Development,
◦ Software testing.

Question 3: Please specify about your current job.

◦ Your current team,
◦ Your role in the team.

Interview part to explore the regression testing goals

We are heading to the core part of this interview, and we are interested to know about the practitioners' perspective on regression testing goals. We will also be interested to know about the information needs and metrics used to measure these goals. Please feel free to add detail at any point of the interview that you think we missed asking or forgetting to describe.

Defining regression testing We know the academic definition of regression testing, and we are interested in learning that perception of regression testing that prevails in practice.
Question 1: How do you perceive regression testing?
Question 2: How do you perform regression testing?
Question 3: Are you satisfied with the regression testing approaches used in your team/organization?

Success in regression testing To determine the success of any activity, we measure it with the predefined goals, that is, if the goals have met or not.
Question 1: At your company / team do you define success goals?
Question 2: What are the goals that you think are essential to achieve success in regression testing?
Question 2: Which are the information needs necessary to achieve the goals?
Question 2: Do you measure the success? or Do you measure or evaluate the goals?
Question 3: How do you measure?
Question 4: How will you determine that the desired goals have been achieved? Or Which metrics do you use to evaluate the success goals?

Closing questions We mentioned earlier that this research aims to compare the literature and practitioners' perspective on regression testing goals. Since you have given us a walkthrough of your regression testing process and your success goals and measures, we want to know your opinion on the literature findings. We have identified the following goals and measures from the literature.
Question 1: Which of these goals do you think are aligned to your perspectives?
Question 2: Which metrics do you think can be used in your environment? Question 3: Do you want to share some more information that you think is important to consider that we may have missed?

Chapter 4

Regression testing for large-scale embedded software development – exploring the state of practice

4.1 Introduction

Testing is an essential aspect of any software development project. The companies are facing various testing challenges, especially for large and complex products [3, 17]. The most challenging testing activity in large-scale systems development is regression testing, which can consume up to 80% of the total testing cost [20, 25]. Regression Testing (RT) is a retest activity to ensure that system modifications do not affect other parts of the system and that the unchanged parts of the system are still working as they did before a change.

RT is a challenge for the software industry, especially for large-scale embedded software development companies in the context of systems with continuous integration and delivery. Considering the recent systematic literature review studies conducted on the topic of regression testing: [24, 26, 32, 33] we can conclude that it is a well-researched area with a large number of suggested techniques. Despite extensive research on RT, research results are not finding their way into practice. There are several reasons, like differences in terminology, availability of research results and a lack of empirical evaluation of RT techniques in real industrial environments [19, 25].

Other factors that affect the transition of research to the industry are, communication-gap between practitioners and researchers, consideration of testing challenges at the different level of abstraction, and differences in the vocabulary and context regarding the concepts of testing [21]. It is essential that researchers consider real-world situations. Without a focus on real industrial needs and practices, a majority of new techniques proposed in the literature will not fit with existing practices. Testing is not only a technical challenge, but also a socio-technical issue [17].

In this paper, we have investigated the current state of regression testing practice in two large-scale embedded software development companies. The contributions of this study are as follows:

- Regression testing definition: Presented the practitioners' perspective of regression testing.
- Regression testing practices: Presented how practitioners undertake the regression testing activity in the companies. How they are selecting/prioritizing and what are the challenges the practitioners are facing in their current practices.
- Improvements: Highlighted the suggestions of the practitioners to overcome the identified challenges.
- Regression testing goals: Presented the goals and criteria that could be used to evaluate the success in regression testing.

The remainder of this paper is structured as follows. Section 4.2 discusses the related work. Section 4.3 describes our research methodology. Threats to validity are discussed in Section 4.4. Results of the study are presented and discussed in Section 4.5. A summary, organized along our research questions, as well as conclusions can be found in Section 4.6.

4.2 Related work

There is a large body of research on software testing ([1, 4]), but most of this research is focused on methods and tools. There is only little work on the perspectives of testing practitioners in industry on regression testing. We found only 14 articles [10–16, 18, 22–25, 28, 34] related to our work, which we discuss further below. Out of these 14 studies, nine deal specifically with regression testing, whereas five have a broader focus. Other than regression testing specific studies, we included those papers in the related work, where authors are discussing any aspect of regression testing. It is worth mentioning, that some of the works (especially, studies presented in [18, 22, 25]) are not recent. This further underlines the need for investigations on the topic, specifically with a focus on industry cases. Ali et al. conducted a systematic review on industry-relevant regression testing research [34], which also emphasizes the lack of industrial research in terms of studying regression testing interventions and classifying them.

We did not conduct a systematic review for the related work. However, in our searches, we focused on the literature that seemed most relevant:

- general testing practices, as we are focusing on industry testing,
- automation, as this is considered an important aspect, given that regression testing is conducted repeatedly, and
- studies explicitly focusing on regression testing with a specific focus on industry applications.

To identify studies on the mentioned areas, we used the search terms (“Regression Testing", “Testing practice", “Automated testing", “Testing challenges", “Testing tools", and “Testing methods"). With respect to the main focus of the related works, we organized them into four groups: (1) general testing practices, (2) testing methods and tools, (3) automated testing, and (4) regression testing.

General testing practices Two studies, both of them surveys, investigated general testing practices ([11, 12]). Dias-Neto et al. [11] conducted a survey to identify the software testing practices in South America. The study was carried out with the practitioners from Brazil and Uruguay. The authors highlight the testing practices that the practitioners consider as essential and beneficial. For instance, the practitioners think testing documentation as useful for current and future testing activities. The practitioners acknowledge the importance of test management and error reporting tools. The essential testing types are system and regression testing. The authors also highlight some weaknesses of the companies. For example, they indicate that the companies do not measure their testing activity and the tendency of using test automation tools is not encouraging.

Kassab et al. [12] conducted a web-based survey to explore how software practitioners are using testing. The focus of their research was to study the overall testing practices. Authors indicate that the use of black box testing techniques is more common as compared to white box testing. Regression testing is one of the testing levels getting more attention of the companies. There is a trend in the companies to outsource the regression testing activity. Among the surveyed companies majority of telecommunication and gaming companies prefer the outsourcing of regression testing, and they are satisfied with this practice. The authors highlight requirement coverage as the most used metric in the companies, followed by the test execution rate. Test stopping criteria for the majority of the companies is the deadline of the project.

Testing methods and tools Ng et al. [18] investigated software testing practices in ICT companies in Australia. Their focus was on testing methodologies/techniques, tools, metrics, and standards. The authors highlighted that the training share for testing staff is low, according to the results presented in the study, universities and training colleges offer only 10.7% of the total training courses for testing staff. Regarding regression testing, the authors report that 69.2% of the organizations are using regression testing for all of the applications they are developing. Regarding the frequency of regression testing, 53.3% of the organizations repeat regression testing for every new version of the product and 28.9% reported the use of regression testing after every change.

Automated testing Two studies focused on the state of practice in testing automation ([13, 15]). Rafi et al. [13] highlighted the benefits and challenges of automated testing using a systematic literature review followed by a practitioner survey. They found only few studies discussing the benefits and limitations of automated testing. The authors conclude that automation is beneficial in an environment where excessive regression testing is performed and it helps in improving test coverage. The key limitations are initial setup costs, training, and availability of reliable testing tools. Furthermore, a majority of testers believe that automated testing cannot replace manual testing.

Kasurinen et al. [15] studied the current state of practice and required improvements in software test automation in an interview-based empirical study. The authors suggest that most of the practitioners are satisfied with their current test policy and testing practices, and they are not thinking of any change in it. Regarding automation, the authors reveal that only 26% of the test cases were automated, and the majority of these test cases are related to unit and regression testing. The automation of regression testing is a common practice among the companies, and regression testing was the most practiced testing type in the sample organizations.

Regression testing Nine studies focused specifically on regression testing aspects ([10, 14, 16, 22–25, 28, 34]).

In a recent systematic review, Ali et al. [34] synthesized the industry-relevant research on regression testing. The authors introduced three taxonomies to support the communication of industry relevant research on regression testing to the testing practitioners. These taxonomies capture aspects of industrial-relevance regarding the regression testing techniques. The authors defined context, effect, and information taxonomies to enable the practitioners to compare the research proposals and access their applicability and usefulness for their specific context. In this study, the authors mapped 26 industrially evaluated regression testing techniques with the taxonomies. The identified techniques are addressing the aspects of test case selection, prioritization, and minimization.

Brahneborg et al. [28] extracted the challenges corresponding to the existing methods of regression testing from a set of empirical studies. The authors classified the challenges into two categories, 1) method related challenges, 2) Organization related challenges. Among method related challenges they highlighted, handling failures, performance measurement, handling fault distribution, scalability of techniques, and tool support. Whereas regarding organization related challenges, the authors describes existence of a structured test suite (test suite maintenance), information availability, knowledge and skills of testers and developers, and management support.

Minhas et al. [10] conducted a focus group based study to investigate the views of industry practitioners and software engineering researchers concerning regression testing. The authors explore the perceptions of both communities about the scope of regression testing. They also identify the regression testing success goals. The authors listed confidence, high precision, and fault-slippage as the essential goals of regression testing. They conclude that the perception of the practitioners and researchers about regression testing is alike, and there are similarities in views concerning regression testing goals. However, the authors highlighted the differences in the priorities of goals among the practitioners and the researchers. Finally, the authors indicate the need for measuring the regression testing task to enable the testers to measure success. The goals identified in the study are summarized in Table 4.2.

Parsons et al. [23] conducted case-studies and an online survey to investigate the adoption of regression testing strategies in agile development. The authors analyzed different contextual factors that can have an impact on regression testing strategy. The focus of their research was to identify the organizational maturity regarding regression testing and operational aspects of regression testing in the surveyed organizations. The authors found that the maturity of the organization is the primary factor for successful regression testing. The authors conclude that organizations can get potential benefits of investments in regression testing. The authors highlighted the need for investigations in the areas of change and risk management regarding regression testing.

Yoo and Harman [24] surveyed the literature on regression test case minimization, selection, and prioritization. They specified the state of the art, trends and issues concerning these areas of regression testing. The authors conclude that the trend to evaluate regression testing techniques is getting a significant increase in the research. However, the majority of empirical studies are carried out with systems under test of less than

10,000 lines of code and test suite sizes of less than 1,000 test cases. They also found that almost 60% of the empirical studies on regression testing are using programs from the software infrastructure repository (SIR[1]) ([5]). It indicates that evaluation of regression testing techniques in real industrial context is limited. The authors argued that there is a potential risk of over-fitting the research on regression testing techniques to the programs that are readily available. They suggested that for the future research on regression testing, researchers should opt for alternative data sources and focus should be on the transfer of technology to industry.

Juergens et al. [14] carried out an industry case study to highlight the challenges concerning regression test selection techniques when applied in manual system tests. They suggest that the testing effort could exceed the expected limit when applying selective regression testing techniques on manual testing. The authors also think that under-specification of manual tests can reduce the conclusiveness of results. Finally, they suggest strategies to improve the manual regression test selection.

Engström and Runeson [16] investigated regression testing practices and challenges and pointed out that a majority of the challenges are related to testability issues and good practices are related to test automation. The authors note that most findings of their study are related to testing in general and are not specific to regression testing. However, there was a consensus among the practitioners regarding the concept and importance of regression testing. Some key findings of this study are provided in Table 4.2.

Harrold and Orso [25] analyzed the state of research and practice in regression testing. The authors conducted the review of research on regression testing and informal survey with the researchers and practitioners. They concluded that hardly a few methods and tools proposed in the research are in use of industry. The authors identified various issues that are hindering the transition of proposed techniques to practice. They also highlighted the technical/conceptual issues, needed to be addressed by the research. Issues identified by Harrold and Orso are listed in Table 4.2.

Lin [22] conducted a literature-based survey on the regression testing research and recent practices. Their goal was to identify the gaps in regression testing research and its industrial practices. The author suggests that there are some gaps and efforts should be made towards the implementation of regression testing research in industry. Addressing these gaps should be the future direction in research.

Summary A summary of related work is presented in Table 4.1. Of the 14 studies discussed above, 9 focused on aspects of regression testing. Overall, the related work shows that there is a gap between research and practice. New techniques should have to

[1]http://sir.unl.edu/portal/index.php

be evaluated in the real industrial context. The researchers should work on technology transfer of the regression testing techniques.

Table 4.1: Summary of related work. The first column indicates the subsection in Section 4.2 (GTP: General Testing Practices, TMT: Testing Methods and Tools, AT: Automated Testing, RT: Regression Testing).

	Ref.	**Methods**	**Year**	**RT**[1]	**Focus of the study**
GTP	[11]	Survey	2017	No	Characterizing the testing practices in South America.
	[12]	Survey	2017	No	Overall testing practices.
TMT	[18]	Survey	2004	No	Current state of testing practices, testing methodologies/techniques, testing tools, metrics, and standards in Australian ICT companies.
AT	[13]	SLR and Survey	2012	No	Benefits and challenges of automated testing.
	[15]	Interviews	2010	No	Current state and required improvement in test automation.
RT	[34]	SLR	2019	Yes	Industry relevant regression testing research.
	[28]	Literature study	2017	Yes	Identification of regression testing challenges.
	[10]	Focus Group	2017	Yes	Regression Testing goals, information needs and metrics.
	[23]	Case study and survey	2014	Yes	Regression testing strategies and the factors that influence the adoption.
	[24]	Literature Survey	2012	Yes	A detailed analysis of trends and issues in regression testing concerning minimization, selection and prioritization.
	[14]	Case study	2011	Yes	Regression test selection challenges when applied to system manual tests.
	[16]	Focus group and survey	2010	Yes	Regression testing practices and challenges.
	[25]	Literature review and survey	2008	Yes	An analysis of the state of the research and the state of the practice.
	[22]	Literature survey	2007	Yes	Identification of gaps between the regression testing research and practice.

[1]Whether the work focuses on regression testing aspects.

In order to build a mapping between the existing literature and findings of our study, we extracted related information from some of the studies presented in the related work. Considering the scope and method the studies that could be regarded as closely relevant to our work are [16, 25]. Both studies are of exploratory nature and purpose to investigate the current state of practice concerning RT. We extracted information related to practices, selection and prioritization criteria and challenges from these studies. We did extract some information regarding the selection and prioritization criteria from the literature survey by Yoo and Harman [24]. For the regression testing challenges, we also utilized the study by Brahneborg et al. [28]. Regarding RT goals we extracted the information from [10, 31]. We have listed the key findings of these studies in Table 4.2. It is essential to specify that the literature findings are not exhaustive because finding practices, challenges, etc. from the literature was not the goal of this study.

Table 4.2: Literature findings on RT state of practice

Aspect	ID	Description	Ref
RT Practices	LPr1.	In industry test suite maintenance is largely manual.	[25]
	LPr2.	For RT, many organizations rerun all test cases (retest all).	[16, 25]
	LPr3.	A most common approach is running core set of test cases.	[16, 25]
	LPr4.	In large number of organization, test case are selected randomly on the basis of experience of testers.	[25]
	LPr5.	Mostly organizations use in-house build techniques and tools	[25]
	LPr6.	Some practitioners prefer to run as many as possible.	[16]
	LPr7.	Start RT as early as possible.	[16]
	LPr8.	Run RT before each release.	[16]
	LPr9.	Complete re-test for critical parts.	[16]
	LPr10.	Focus is on functional test cases	[16]
	LPr11.	Selection of test cases depends on the situation.	[16]
	LPr12.	The amount and frequency of RT depends upon the various factors.	[16]
Prioritization	LPc1.	Change.	[16, 24, 25]
	LPc2.	Cost.	[25]
	LPc3.	Running time.	[25]
	LPc4.	Criticality.	[25]
	LPc5.	Complexity of the test cases.	[25]
Selection	LSc1.	Change.	[16, 24, 25]
	LSc2.	Historical test data on test case effectiveness.	[24, 25]
	LSc3.	Timing data on the last time a test case was run.	[25]
	LSc4	Traceability between requirements to test cases.	[24, 25]
	LSc5.	Situation based selection.	[16]
	LSc6.	Areas affected by the changes.	[16]
Challenges	LC1.	Identification of obsolete test cases.	[25]
	LC2.	Selection of relevant test cases.	[16, 25]
	LC3.	Test case prioritization.	[25]
	LC4.	Test suite augmentation.	[25]
	LC5.	Removing redundant test cases.	[16, 25]
	LC6.	Creating effective test cases.	[16, 25]
	LC7.	Manual testing (expensive and time consuming).	[25]
	LC8.	information maintenance.	[25, 28]
	LC9.	Test suite maintenance.	[16, 25, 28]
	LC10.	Test suite assessment.	[25]
	LC11.	Time to RT.	[16]
	LC12.	Balance between manual and automated RT.	[16]
	LC13.	Execution of automated RT.	[16]
	LC14.	Time to analyze results.	[16]
	LC15.	Management support.	[28]
Goals	LG1	Increasing rate of fault detection (effectiveness)	[10, 31, 34]
	LG2	Increasing coverage of test suite (coverage)	[10, 31, 34]
	LG3	Increasing confidence regarding the system reliability (confidence)	[10, 31]
	LG4	Identifying high risk (critical/sever) faults	[31, 34]
	LG5	Identifying change specific faults (effected areas)	[31]
	LG6	The customer should not find fault in the product (fault slippage to the customer)	[10]
	LG7	Finishing RT in limited time and low cost (efficiency)	[10, 34]
	LG8	Running most effective test cases (inclusiveness)	[10]
	LG9	Excluding non effective test cases (precision)	[10, 34]

4.3 Methodology

We conducted a multi-case study in two large companies to investigate the current state of practice regarding regression testing. Data were mainly collected through interviews. Interviews provide an opportunity for direct interaction with the respondents and resolving issues with interpretations during the interview sessions. We choose to conduct semi-structured interviews, since it allows improvisation and exploration of the studied objects [9].

4.3.1 Research questions

RQ1 How is regression testing perceived by the practitioners in industry?

RQ2 What is the current state of regression testing practice in industry?

- RQ2.1 How are the practitioners selecting and prioritizing test cases?
- RQ2.2 What are the key challenges for the practitioners regarding regression testing?

RQ3 What are the possible improvements regarding regression testing practices suggested by the practitioners?

RQ4 How do the practitioners evaluate their success in regression testing?

4.3.2 Case companies

We conducted this study from the platform of EASE [2], which is an industry-academia collaboration project. The companies participated in this study are Sony Mobile Communications AB and Axis Communications AB, from hereafter we will use Sony and Axis to refer the companies. Both Sony and Axis are large-scale embedded software development companies, and both are the collaborators in the EASE project. Sony is a multinational telecommunication company producing smartphones and other smart products. The core part of the software is used in different versions and variants of the devices. The estimated size of different modules varies between 30 and 100 thousand lines of code (KLOC). The approximate number of releases is 20 releases per quarter. At Sony, a majority of the code is written in C. Axis is manufacturing network cameras and other surveillance devices, and claim to be the inventors of the first network camera. Regarding the software for their products, the core platform is similar

[2]EASE – the Industrial Excellence Centre for Embedded Applications Software Engineering http://ease.cs.lth.se/about/

for all products. The estimated size of the platform is 20 millions lines of code. The approximate number of releases for the complete product line at Axis is 100 releases per quarter. For the product development at Axis, the programming languages used are C and C++. For the testing tools, they are using Python. Both case companies invest a substantial effort in regression testing. Each of them has a common code base for their products from which they have release multiple products over several years.

Table 4.3: Overview of interviewees.

	PID	Team[1]	Current Role	Exp[2]
Company A	P1	SWSE	Test Architect &Team Lead	10
	P2	SWSE	Manager Verification	9
	P3	SWSE, FCS	Verification Engineer	13
	P4	FCS	Test Architect	10
	P5	FCS	Verification Engineer	10
	P6	FCS	Product Owner	11
Company B	P7	FWD	Tech Lead	6
	P8	FWD	Engineering Manager	8
	P9	FWR	Program Manager	2
	P10	PTS	Technical Coordinator	7
	P11	PTS	Test Area Maintainer	6

[1] SWSE (Software Security & Enterprise), FCS (Flash Core Security), FWD (Firmware Development), FWR (Firmware Release & Upgrade), PTS (Platform Storage).
[2] Experience in number of years the practitioner is working with the current company.

From Sony, the practitioners from two teams (Software Security & Enterprise (SWSE) and Flash Core Security (FCS)) participated in the study. These teams are working on two different parts of a product, and some modules of the software for both parts are developed within the respective teams. Along with the in-house development, the company is also using modules developed by their vendors. The SWSE team is working on the enterprise part of the product, and performs testing for software security and enterprise at the system level. Testing for software security is automated, whereas testing for the enterprise part is manual. For test automation a third-party tool is used. The stated reason for using a third-party tool is lower costs and higher quality. The team uses test automation tools provided by the vendors who are also providing different modules of their product. The FCS team is working on the boot & loader part and performs testing mainly at the component level. Testing for the loader is automated, whereas testing for the boot part is manual. The team uses an in-house developed tool for automated testing and currently working on complete test automation and the adoption of an open source tool. The reason for relying on internally developed tools is that the teams' perception is that third-party tools might not fit their context and would require extensive effort to adapt.

From Axis, practitioners from three teams (Firmware Development (FWD), Firmware Release & Upgrade (FWR), and Platform Storage (PTS)) participated in the study. The FWD team is responsible for the new product development and new feature additions to products. They perform testing at unit, integration, and system level. Mainly regression testing is automated except the testing of new features.

The FWR team works on upgrades and new releases. Their primary responsibility is pre-release (Beta testing), performance, and stability testing. Performance and stability testing is automated, but complemented by manual exploratory testing. Regression testing is regarded as input to release. The PTS team is responsible for the storage part of the platform and performs testing at the unit and system level. For regression testing, the policy of the team is to run all test suits during nightly trials. At Axis, the practitioners are mainly relying on an in-house developed automated testing framework. The reason for using an in-house developed testing framework is to ensure testing performance and stability. Furthermore, the testing tools are tightly tied with their APIs. The automation framework is common for the whole platform.

4.3.3 Data collection

The data was mainly collected through interviews. We conducted semi-structured interviews with a total of eleven testing practitioners.

Selection of participants

Before the actual study, with the help of project contact persons, we conducted two separate workshops at both companies. The purpose was to present our research objectives to the managers and potential participants. We did highlight the required experience and roles of the potential participants. These workshops enabled us to understand the overall hierarchy of the companies, their team structures, and the working procedures. The selection of the participants was based on convenience sampling [29]. Convenience sampling is a non-probability non-random sampling method. We refer our selection as convenience sampling because we selected those participants who were fulfilling our selection criteria of role and experience, and who were willing to participate. A summary of the participants is presented in Table 4.3.

Interview design

Based on the research questions, we prepared an interview guide consisting of open-ended questions. We did not restrict ourselves to the pre-defined questions, we added/improvised the interview questions during the sessions. The first author developed the interview guide, while the second author helped during the revisions, and an independent researcher reviewed the interview guide. The interview guide consists of seven sections: introduction, background, current practices, selection and prioritization, challenges and improvements, and success criteria. The complete interview guide is presented in A.

Interview execution

In total, we conducted eleven interviews (five at Axis, and six at Sony) with the representatives of five teams. The first and second author participated in the interviews at Axis, whereas the first and fourth author participated in the interviews at Sony. The first author guided all the eleven interviews, while the second and fourth authors assisted in their respective sessions. Observer triangulation was utilized during all interviews. Besides the person guiding an additional researcher took the interview notes.

Each interview took about one hour and was audio-recorded with the consent of the participants. During the interview, we complemented our documentation with mind-maps and free text notes. The template of mind-map used for this study is presented in Figure 4.1. The template consists of six nodes, five nodes represent the main topics of our study (i.e. current practices, selection and prioritization criteria, challenges, improvements, and evaluation), whereas one node of the mid-map shows the background information of the participant.

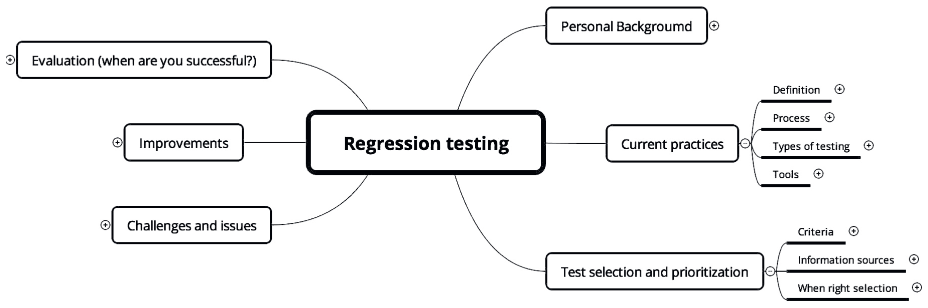

Figure 4.1: Mind-map used for data management and classification.

4.3.4 Interpreting, analyzing and validating interview transcripts

To minimize misunderstands and misinterpretations, we documented the interviews by three different means: structured mind-maps, free text notes, and audio recordings. The mid-map was the master document for the recording of interview results. Based on the research questions, we already structured the mind-maps according to the main themes of our study (see Figure 4.1). For the analysis of interview notes and audio transcribed transcripts we followed the steps defined by Cruzes and Dybå [8]. We also took inspiration from the method followed by the Petersen and Wohlin in their study [27].

Step 1: Transcribing the interview recordings The first step was to finalize the interview transcripts. In this step we transcribed the audio recordings, the first author transcribed all the audio records and the second author randomly verified these transcripts. We did use the free text notes as a separate source of raw data.

Step 2: Clustering of categories The second step was to cluster the raw data into meaningful groups. We used color coding to separate the concepts in the transcripts and identified the similarities and differences in the themes of different transcripts. For example, the statements were grouped according to the categories, “current practices”, “selection and prioritization criteria”, “challenges”, etc. As an example, we have listed some statements of interviewees in the in Table 4.4. We clustered the raw data separately for each participating team.

Step 3: Clustering of subcategories In the third step, we assigned the labels (themes) to the statements already clustered in step 2. In this step, beside the labeling process, we also did restructure the statements where it was necessary. Table 4.4 presents the labels along with the restructured statements.

Step 4: Mapping categories and subcategories to mind-maps In this step, we mapped the results generated from free text notes and audio generated transcripts to the mind-maps, and updated the mind-maps accordingly.

Step 5: Generation of Results From the final copy of mind-maps we generated Tables (4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, & 4.12) of results according to the research questions presented in the section 4.3.1.

Table 4.4: Analysis procedure adopted for step 2 and step 3.

SNo	Original statement	Category	Subcategory	Restructured statement
1	i) Working very close with our development team, Developers cannot merge any thing with out the approval of testers ii) Decision making approach: Before heavy documents and test plans. Scoped out a lot of overhead, now it is within the project, in collaboration with the developers and with the project manager and QA (decision within the development team).	Practice	Process	Developers and testers collaborate while deciding the scope for the testing of changes and new features.
2	i) If we don't find issues with regression testing, we try exploratory to find something ii) We do some exploratory testing to find new issues.	Practice	Testing type	Use of exploratory testing as an alternative to RT.
3.	i) Usually when we are closer to the freeze of the code deadline we try to make a little bigger scope. ii) We run full scope at the of development, when developers cannot make changes. iii) If there is a change in the core of the product we run the full scope.	Practice	Process	Regression testing is performed with full/bigger scope at code-freeze, before release, or in case of changes in the core of the system.
4.	We tag our test cases like sanity, scope, mandatory, tagged test cases are suppose to be really important test cases.	Practice	Prioritization	Use of priority tags for the test cases with respect to the relevant importance.

Step 6: Validation of results In qualitative studies, the researcher's bias can influence the interpretation of the results. It can be avoided by validating the interpretations from the sources.

To validate our interpretation of results, we conducted two workshops with the participants of our study and the representatives of the companies for EASE. We used validation sheets to record the issues regarding the interpretation of the results. Feedback of the study participants and companies representatives was positive, they identified a few minor revisions (mainly related to the RT practices), which we adjusted accordingly. Lately, we provided the final results to the companies representatives for the final verification, their response was encouraging, and they did not raise any issue in the final results.

4.4 Threats to validity

This study is exploratory and based on the experience and perceptions of industry practitioners. The data was collected through semi-structured face-to-face interviews. We asked open-ended questions to capture viewpoints without any restriction. The purpose was to avoid researcher bias and get insights into current industry practices.

In our discussion of threats to validity, we follow the guidelines by Runeson and Höst [9].

Construct validity This aspect of validity is regarding the underlying operational measures, concepts, and terms of the study. In our case, the selection of the participants regarding their competence and relatedness to the study area was a potential threat. Similarly, missing any critical aspect from the interview design could also be a threat. To mitigate these, we conducted pre-study workshops to explain the purpose of the study to the focal persons in the companies and recruit appropriate participants for the study (Ref: Section 4.3.3). Regarding the interview design, we have carefully designed and reviewed the interview guide (see Section 4.3.3 and A).

Internal validity This aspect of validity is essential if causal relations are examined. Generally, we can state that studying causal relationships was not in the scope of this study. While describing the challenges and improvements we did analyze the possible dependencies among the identified challenges. We also studied the relationship between the identified challenges and improvements. To avoid inconsistencies and researcher bias while analyzing the results, we consulted the participants to validate our interpretations of dependencies/relationships.

External validity This aspect of the validity refers to the generalization of findings. Participants of this study represented five different teams of two large, multi-national companies working on embedded software systems in the communications domain. We have linked our findings with the findings from existing literature, which indicates similar findings. A general challenge of case studies is generalizability, but their advantage is the amount of qualitative information that can be collected and the depth of analysis this makes possible. The effort needed for qualitative data collection and analysis limits the number of cases that can be studied, though. With a questionnaire, for example, more data points could be obtained, but qualitative information on the desired level of detail could not be obtained. Given the limitations of case studies, Ghaisas et al. [30] suggest to describe the cases in sufficient detail and generalize by analyzing the similarity of cases. To support such an analytical generalization of our results, we have presented a detailed discussion of the cases under study in Subsection 4.3.2.

Reliability To ensure the reliability of the results, we assured the correctness of the collected data and interpretation of the results, see sections 4.3.3 and 4.3.4. For each round of interview two authors participated. We did ensure the results triangulation, by involving multiple authors in the results interpretations, and we did validate the results in two workshops with the participants of the study.

4.5 Results and discussion

This section presents the results regarding the practitioners' perspectives on regression testing, test case selection and prioritization criteria, challenges, improvements, and success criteria. The subsections are organized according to the research questions presented in Section 4.3.

4.5.1 The Practitioners' perceptions of regression testing (RQ1)

The purpose of our first research question was to elicit what participants think about regression testing. Our first interview question was *"What is regression testing for you?"*. Instead of standard definitions of regression testing [6, 7], our participants provided practical descriptions of regression testing with close relations to the perspectives (teams) they were representing.

From Table 4.5, we can see that the practitioners covered three aspects while defining regression testing: 1) The timing or frequency of regression testing, 2) the scope of regression testing, and 3) the overall goal of regression testing. Regarding timing, the participants define the need for regression testing after changes or fixes, before release, whereas some participants suggest running regression testing continuously. About the scope of RT, the practitioners are flexible, some of the participants describe running RT with smaller scope, whereas some prefer to adopt a "re-test all" policy. Finally, the participants are agree on the goal of RT, (i.e. to get confidence about the integrity of the system after changes, fixes or before the release). Difference in the initiation timing of RT and scope does not means that teams / companies are perceiving RT differently. In fact this represent that on which development level a team is employing the RT and how often they are making changes or adding new features to the releases.

We thus synthesized the views of all participants from both the companies and finalized the following definition:

Regression testing is a repetitive activity which is applied to ensure that changes/fixes/upgrades did not affect the behavior of the system/product negatively and nothing was broken or destroyed in the functionality.

4.5.2 Regression testing practices (RQ2)

The objective of the second research question was to understand how the practitioners conduct regression testing in their projects. Figure 4.2 presents the regression testing process in the case companies and Table 4.6 summarizes the regression testing prac-

Table 4.5: The practitioners' definitions of regression testing, as response to interview question “What is regression testing for you?”.

CID[1]	**PID**[2]	**RT Definition**
SWSE	P1.	To verify that introduced **changes/fixes** have not changed the behavior of the system in a negative way.
	P2.	Small sanity check on the **changing part of system**, try to figure out nothing has broken.
	P3.	Run selected test cases **continuously**, to check If something has broken or if everything has broken.
FCS	P4.	For **every release**, along with the testing of the new features, it is much important to make sure that old functionality is intact.
	P5.	To make it sure that everything still works, need to run a regression test for **every new fix** from developers. It could differ concerning what are the changes.
	P6.	It is a validation that primary use cases are working, regression testing is a hygiene thing to make sure that any basic functionality have not broken. It is not expected to find much during regression testing.
FWD	P7.	To test what happened with other products, did **changes** destroy other products?
	P8.	Regression testing is a **constant qualifier**, and it is a process to test over and over.
FWR	P9.	To make sure that there is no regression, no degrades. Regression testing is **input to release**.
PTS	P10.	Regression testing is to verify that the functionality that was working previously, still works correctly or something has broken that was not expected while **making changes**.
	P11.	To verify that during **changes or adding new features** the overall quality of the functionality, performance of the database has not decreased.

[1] Company/team ID according to Table 4.3.
[2] Participant ID according to Table 4.3.

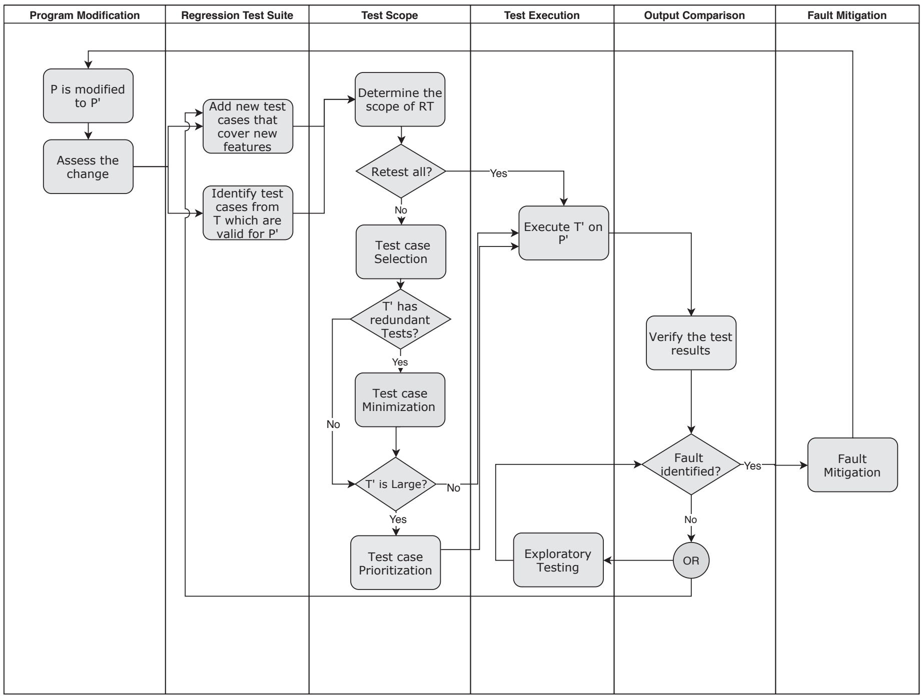

Figure 4.2: Regression testing process in the case companies

tices identified in our study. Regression testing is instantiated after any modification (change, fix, or upgrade). The practitioners start the process by assessing the change and adding relevant test cases to the RT suite. The scope of RT is determined to decide about the adoption of a retest-all vs selective RT strategy. In the case of retest-all, the next step would be running the RT suite and in case of selective RT, a series of decisions are carried out, see Figure 4.2 for detail. After running the RT suite, test results are verified and in case defects are found, defect mitigation is invoked. If no defects are found by running the current RT suite, two alternatives could be adopted; to augment the RT suite or to continue with exploratory testing.

From Table 4.6, it is evident that the practitioners do collaborate with each other while making any decision regarding RT. The companies are using a mix of automated and manual RT, at Sony the practitioners specified that almost 50% of their testing activity is automated, whereas participants from Axis claimed that majority of the RT

Table 4.6: Regression testing practices.

ID	Description of practice	SWSE	FCS	FWD	FWR	PTS	CPL [1]
Pr1.	*Collaboration*: Developers and testers collaborate while deciding the scope for the testing of changes and new features.	✓	✓	✓	✓	✓	
Pr2.	*Execution frequency*: The frequency of executing tests depends upon the type/importance of the functionality.	✓					
Pr3.	*Execution frequency*: Testing as much as possible with selected test cases near the completion of the project.					✓	LPR6
Pr4.	*Reuse of test cases*: Using existing test cases for the testing of changes.	✓			✓		
Pr5.	*Adding new test cases*: New test cases are added for new features and issue leakage.	✓	✓				
Pr6.	*Running full scope*: Regression testing is performed with full/bigger scope at code freeze, before release, or in case of changes in the core of the system.	✓	✓		✓	✓	LPr2
Pr8.	*Nightly testing*: RT is executed with full scope in nightly runs.			✓		✓	LPr2
Pr7.	*Day time testing*: Select a scope that should run (fast) in a day time and cover bigger part.	✓		✓		✓	
Pr9.	*Weekly round of RT*: Run a weekly round of RT with a smaller scope.	✓					
Pr10.	*Scope Selection*: Mostly run RT with selected scope because of time constraint.		✓				
Pr11.	*Running fix suite*: Using fix set of test cases (that cover the core/basic functionality) for RT.	✓					LPr3
Pr12.	*RT Automation*: The companies are using both manual and automated RT.	✓	✓	✓			
Pr13.	*Testing tools*: For test automation using third party tools.	✓					
Pr14.	*Testing tools*: For test automation using in-house developed tool.		✓	✓	✓	✓	LPR5
Pr15.	*RT suite size*: Regression test suites are fairly large.	✓	✓				
Pr16.	*Exploratory testing*: Use of exploratory testing to complement RT.	✓	✓		✓	✓	
Pr17.	*Test strategy*: For every new project a detailed test strategy / plan is developed.	✓	✓	✓		✓	
Pr18.	*Priority tags*: Using tags with test cases to determine the priority.		✓				
Pr20.	*Traceability labels*: Use of test case labels to link with respective modules.					✓	
Pr19.	*Tickets*: Use of tickets for highlighting the issues, it helps to identify the type and nature of the error.				✓		
Pr21.	*When to introduce RT*: Sometime early start of regression testing is preferred to catch defects at early stages.	✓				✓	LPr7
Pr22.	*RT level*: Applying RT at System level.	✓		✓		✓	
Pr23.	*RT level*: Applying RT at Component level.		✓				
Pr24.	*RT goal*: The goal of regression testing is to have confidence that product is in good shape.			✓	✓		

CPL: Corresponding practice(s) in literature.

is automated. For test automation the companies are using in-house build test tools, except one team who is currently using a third party tool. All three teams at Axis are using the same test automation and management tool. The scope of RT depends on change/fix and time-line of the project. For changes, testers prefer to run the smaller scope (selected test cases), in case of complex changes (e.g., in the core) they try to opt for a wider scope. Near the completion of project (code freeze or near the release) the practitioners favor the execution of RT with full scope (re-test all). The practitioners at Axis also highlighted that during day time they run selected test cases and during nightly testing they prefer to run full scope. The participants from SWSE (Sony) highlighted that they run weekly rounds of RT with selected scope, whereas FCS (Sony) adopt the selection of scope because of time constraints.

The practitioners reuse of existing test cases as regression test suite, they do augment new test cases in the existing test suites in case of adding of new functionality or any issue leakage. The practitioners label the test cases with “test tags” and/or “test case labels” according to the respective importance and/or modules. The use of labels conceptually differs in both the companies, at Sony labels are used to highlight the relative priority of the test case, and at Axis labels are used to link the test cases with respective modules.

Regarding test strategy or test planning, two teams SWSE (Sony) and PTS (Axis) highlighted this aspect, both teams claimed it a company wide activity. At Sony, a high level test strategy exists that serves the basis for the design of detailed test strategy. For every new project test architect design the detailed strategy, it includes the details about the modules to test, scope of testing, test cases, and etc. At Axis, there exists an organization wide test plan. Preparing and updating the project specific test plan and test scripts is the responsibility of the QA team. Some of the practices identified in our study are already defined in the existing literature (see Table 4.2), we have created the mapping of literature identified practices in the last column of Table 4.6. Considering the results of a survey-based study conducted by Dias-Neto et al. [11] in which authors surveyed the overall software testing practices we can conclude that practices identified in our study are purely regression testing related. As opposed to the survey conducted by Engström and Runeson [16] where authors specified that the practices identified in their survey are general testing practices.

Test case selection and prioritization (RQ2.1)

An important aspect of regression testing practices is test case selection and prioritization, which we investigated in RQ2.1. A summary of selection and prioritization criteria along with the information sources used for decision making can be found in Tables 4.7 and 4.8, respectively. We did map the information sources with the selection/prioritization criteria in Table 4.7 under the column heading “UIS”. The last column “CCrL” in Table 4.7 lists the selection/prioritization criteria available in the literature (Literature findings are listed in Table 4.2).

Table 4.7: Test selection and prioritization criteria.

ID[1]	Criteria	SWSE	FCS	FWD	FWR	PTS	UIS[2]	CCrL[3]
SPCr1.	Change (size, complexity,and location).	✓	✓	✓		✓	IS1, IS2, IS7	LPc1, LPc5
PCr2.	Risk.	✓			✓	✓	IS2, IS4	
PCr3.	Critical functionality.	✓	✓				IS1, IS2, IS3, IS7	LPc4
PCr4.	Defect priorities.				✓		IS6	
SCr5.	Situation based scope assessment.		✓				IS2, IS7	LSc5
SCr6.	Coverage(feature/module).			✓		✓	IS2, IS5	
SCr7.	Affected areas.			✓		✓	IS1, IS2	LSc6
SCr8.	Deadlines.	✓	✓		✓		IS8	
SCr9.	Least recently used test cases.	✓					IS2, IS5	
SCr9.	Most frequently failed test cases.			✓			IS2, IS5	

[1] SPCr: Selection/Prioritization Criteria, PCr:Prioritization Criteria, SCr: Selection Criteria
[2] UIS: Utilized Information Sources.
[3] CCrL: Corresponding Criteria in Literature

Table 4.8: Information sources utilized for test selection and prioritization.

ID	Information sources	SWSE	FCS	FWD	FWR	PTS
IS1.	Developers' feedback.	✓	✓	✓	✓	
IS2.	Experience and Knowledge of system.	✓	✓	✓		✓
IS3.	Issue ranking.	✓				
IS4.	Team meetings.	✓		✓		✓
IS5.	Test history.	✓				
IS6.	Test tags.		✓			
IS7.	Feature to test traceability.			✓		✓
IS8.	Project managers.	✓	✓		✓	

The primary criteria for regression test suite selection and prioritization is ‘change’. The practitioners assess the size, complexity, and location (area) of change. The participants from all the teams highlighted this criterion. Feedback from developers, and experience and knowledge of the practitioner about the system are the information sources used to assess the change. Experience and knowledge of system and feedback from developers are the primary sources of information for majority selection/prioritization criteria. For instance, scope assessment, fix regression test suite, critical functionality, and coverage are the criteria where the practitioners are using their experience to access these criteria.

Other relevant criteria that are used for the test selection or prioritization are risk, deadlines, and critical functionality. Three out of five teams are using these criteria. Measurement of risk is based on change (area and complexity of change), the practitioners are sometimes using issue ranking to prioritize the risk. Knowledge of the practitioner about risk areas is another source of information for risk measurement. Deadlines are central concerning the scope selection. If the project is on schedule and deadlines are relaxed then testers prefer to run full scope (complete test suite) for regression testing. If deadlines are tight (often, it is the case), then the practitioners opt for an adaptive (selected) scope. In this case, they do prioritize the test cases on the basis of the critical (essential) functionality. The selected scope consists of the test cases that cover the critical functionality and the primary use cases (basic functionality). The companies are using a predefined test suite that covers the primary use cases (basic functionality). They update this suite (add new test case) in case developers add new functionality (features) or if they think that something (test case) is missing.

Some selection criteria are team specific. For instance, the SWSE team selects the test cases that have not been executed recently (least recently used test cases), they do choose the test cases concerning the date of use. The testers in the FWD team are using most recently failed test cases and most frequently failed test cases during the selection process.

Key challenges in regression testing (RQ2.2)

The identified challenges are summarized in Table 4.9. They can be classified into two categories: 1) management related challenges and 2) technical challenges. Management related challenges include "C1: Time to test", "C2: Information management", "C4: Communication", "C9: Lack of strategy", "C11: Developers interest in testing", and "C13: Management support". Whereas technical challenges include "C3: Obsolete test cases (Test suite maintenance)", "C5: Test case selection", "C6: Test case prioritization", "C7: Evaluating and improving (Lack of assessment)", "C8: Low coverage", "C10: Traceability", and "C12: Tool support". Although only one team identified lack of management support as a challenge initially, all teams agreed that a lack of management support might lead to most other challenges.

Figure 4.3 presents the relationship between the identified challenges. Based on the

Table 4.9: Regression testing challenges.

ID	Challenge	SWSE	FCS	FWD	FWR	PTS	CCL[1]
C1	*Time to test*: Too much testing in a short time. Lack of time is the primary hindrance for the assessments and innovation of regression testing.	✓	✓	✓		✓	LC11
C2	*Information management*: Because of poor information maintenance and lack of documentation it is hard to extract the required information.	✓	✓	✓	✓		LC8
C3	*Test suite maintenance*:There are tests which are not failing for a long time; the issue of obsolete test cases is a big challenge as it affects efficiency and effectiveness.	✓	✓	✓		✓	LC9
C4	*Communication*: Lack of information about the changes and upcoming plans from the other teams makes it challenging to trace the changes.	✓	✓	✓	✓		
C5	*Test case selection*: Instead of the dynamic test scope focus is on fixed test suites. Adding value to the scope by selecting relevant test cases is a challenge.	✓	✓	✓	✓	✓	
C6.	*Test case prioritization*: Inappropriate assumptions about the priorities and what to select.		✓	✓		✓	LC3
C7	*Lack of assessment*: Lessons learned on a project are often broad in nature, not possible to reflect on the success. Time to test is one the factors that limits the option of evaluations and improvements.	✓	✓	✓		✓	LC10
C8	*Low coverage*: Detection of new bugs is a challenge this is because of, running the same test cases for a long time that is causing the low test coverage.	✓	✓	✓			
C9	*Lack of strategy*: Focus is on project success instead of organizational success, no long-term strategy.		✓	✓			
C10	*Traceability*: Finding trace-links between tests and other artifacts is a challenge.	✓		✓			
C11	*Developers interest in testing*: Developers' least interest in quality, delivering their code without testing.	✓					
C12	*Tool support*: The testers have to go through excessive manual work in RT because of the unavailability of good verification tools.	✓					LC13
C13	*Management support*: Lack of understanding about the importance of verification, and not allocating sufficient time to test.		✓				LC15

[1] CCL: Corresponding challenges in literature

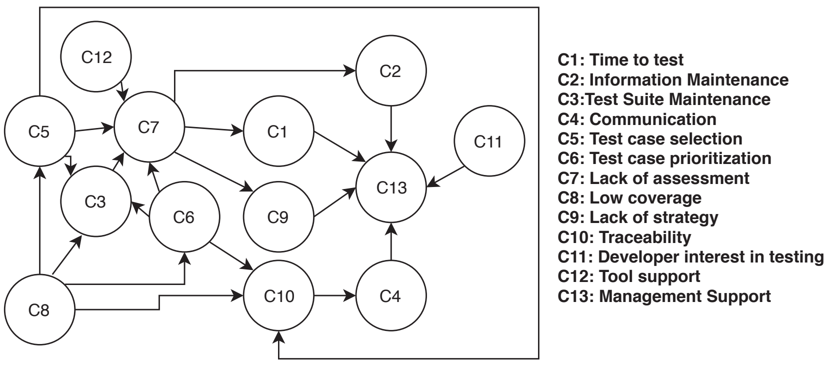

Figure 4.3: Relationship between RT challenges.

descriptions of the practitioners, two challenges are said to be related if the presence of one challenge can cause the existence of another challenge, or if the resolution of one challenge can resolve the dependent challenge. For instance, practitioners describe time to test (C1) as *"[t]oo much testing in a short time. Lack of time is the primary hindrance for assessment and innovation of regression testing."* From this description, we can see that lack of time affects evaluating and improving (C7). Similarly, practitioners define management support (C13) as *"[l]ack of understanding about the importance of verification, and not allocating sufficient time to test."* From this description, we can conclude that the existence of the challenge time to test (C1) might be caused by (lack of) management support (C13).

In Figure 4.3, we linked the challenges to each other with arrows. The arrowhead is towards the challenge that can affect the challenge that is on the tail side of the line. For example, challenge C13 (management support) can cause the presence of the challenges C1, C2, C4, C9, and C11. From the figure we can see that the central challenge is C13 (management support) which could cause all other related problems directly or indirectly. Time to test (C1), information management (C2), and lack of strategy (C9) are root causes of technical challenges. Among the technical challenges, C7 (lack of assessment) is a root of other technical challenges including C3, C5, C6, C8 and C12). Similarly, traceability (C10) can effect selection (C5) and prioritization (C6). Finally C3, C5, C6 and C10 can cause low coverage (C8).

In the last column of Table 4.9 we mapped the challenges identified from the literature (see Table 4.2) with the challenges identified in our study. Seven challenges (C1, C2, C3, C6, C7, C12, & C13) identified in our study are similar to the challenges identified in the studies [16, 25, 28]. Interesting aspect is that first study by Engström and Runeson [16] was conducted in 2010 and the other study by Harrold and orso [25] was carried out in 2008. Despite the voluminousness research on regression testing [24], it is evident that the challenges are not fully addressed after 10 years. It is an indicator that either published research is not fulfilling the industrial needs or the industrial stakeholders are unable to exploit the intended benefits from the available research. Along with the identification of challenges with the help of practitioners, it is also important to work on improvements in a close collaboration with the practitioners.

4.5.3 Suggested improvements for regression testing (RQ3)

In RQ3, we investigated improvements that the practitioners seek in their testing environment. Table 4.10 summarize the possible improvements we identified with the help of the practitioners.

A majority of identified improvements correspond to the challenges presented in Table 4.9. Last column (CC) of Table 4.10 list the challenges that could be addressed by the respective improvements. There is often a one-to-one mapping between the challenges and the suggested improvements. For instance, C2 (information management) is identified as a challenge in both the companies, the practitioners even who did not recognize it as challenge agree to build and maintain a standard information repository (I-1). Similarly, C3 (test suite maintenance) is highlighted as a challenge. To address the issue, the practitioners suggest to work on the elimination of irrelevant and obsolete test cases (I-2: Test suite minimization) and updating test suites by adding new relevant test cases (I-3: Test suite augmentation). Working on test suite minimization will also be helpful for reduction of testing time, that ultimately will be helpful to cope with the challenge of time to test (C1). Another important suggestion that can be helpful for the reduction in time to test is to introduction of parallelized testing (I-12). Overall test optimization (I-14) will also improve the situation.

Availability of good verification tools (C12) is an issue of concern for the companies, because of this fact testers have to do a lot of manual work in RT. In this regard, it is suggested to identify and adopt the verification tools (I-4) appropriate for the companies environment. To cope with the challenge of communication (C4), the practitioners think that there is a need to work for the improved collaboration (I-5). Although, the practitioners (developers and testers) do have a certain level of communication, but there is no formal mechanism for the communication. Specifically, with reference to information sharing regarding changes and change plan. In the presence of well maintained information repository this issue should be minimized. Test case selection and test case prioritization are of central importance in regression testing, especially for the large-scale development. Because right selection with appropriate priority order

Table 4.10: Identified improvements.

ID	Improvements	SWSE	FCS	FWD	FWR	PTS	CC[1]
I-1.	*Information repository*: Need to build and maintain a standard information repository, to make information extraction easier and faster.	✓	✓	✓	✓		C2
I-2.	*Removing irrelevant test cases*: Need to identify and remove irrelevant/obsolete test cases, to reduce the test cost and improve the efficiency and effectiveness.	✓		✓	✓	✓	C1, C3
I-3.	*Test suite augmentation*: Need to update test suites by adding new relevant test cases, to improve the coverage and effectiveness.	✓					C3
I-4.	*Good verification tools*: Need to identify and add good verification tools, to reduce the dependence on manual testing.	✓	✓				C12
I-5.	*Improved collaboration*: There is a need to regulate the collaboration mechanism between developers and tester, to improve the information sharing (What developers are changing? and what testers have to test?)	✓	✓	✓		✓	C4
I-6.	*More exploratory testing*: To find new bugs, need to do more exploratory testing.	✓	✓				C8
I-7.	*Evaluate*: To measure success and improve RT, need to collect test execution statistics and evaluate.		✓	✓		✓	C7
I-8.	*Improved test strategy*: Need to improve the testing strategy and innovate testing methods.		✓				C9
I-9.	*Test History*: Need to maintain test execution history dynamically, so that it could be reused to innovate testing.		✓			✓	C5, C6
I-10.	*Early start of regression testing*: To ensure the stability, need to introduce early start of regression testing.				✓		C1
I-11.	*Test case classification*: To classify test cases according to severity, need to introduce a standard template for error reporting, that should allow linking tests to errors.				✓		C6, C10
I-12.	*Parallelized testing*:To cope with the time constraints, need to introduce the concept of parallelized testing.					✓	C1
I-13.	*Selective RT*: Need to improve the selection mechanism of test cases, selection should be without any compromise.					✓	C5
I-14.	*Test optimization*: To shorten the release cycle, need to work on overall optimization of RT.				✓		C1, C3, C5, C6

[1] CC: Corresponding challenge(s).

can improve the coverage and defect detection rate. The companies lack in both areas (C5: Test case selection and C6: Test case prioritization). From the identified improvement (I-7, I-8) provides the basis for the improvement in selection and prioritization methods. Exploratory testing (I-6) could be an immediate solution choice for the practitioners.

4.5.4 Goals and criteria for successful regression testing (RQ4)

In response to our question "How do you evaluate the success in regression testing?" the majority of the participants responded that they don't use any objective metrics, but apply subjective judgements. Subsequently, we did change our question, "If you have to evaluate the success, which would be your success goals?" Tables 4.11 and 4.12 summarize goals and criteria regarding the success of regression testing. The success criteria refer to the conditions that are essential to achieving the regression testing goals.

Table 4.11: Regression testing success goals.

ID	Goal	SWSE	FCS	FWD	FWR	PTS	CSC[1]	CGL[2]
SG1.	*Customer satisfaction*: The released product is working and the customer is not complaining.	✓	✓				SC6	
SG2.	*Critical defect detection*: RT is regarded as successful if it can find the critical bugs (no critical bugs should be delivered to the customer).	✓	✓	✓	✓	✓	SC1, SC2, SC3, SC7	LG4
SG3.	*Confidence*: The tester should be confident about the achieved quality.	✓	✓		✓	✓	All SCs	LG3
SG4.	*Effectiveness*: In term of fault detection, the goal is to find as many bugs as it is possible.	✓	✓	✓	✓	✓	SC1, SC2, SC3, SC5, SC7	LG1
SG5.	*Controlled fault slip-through*: How many issues have slipped to the customer is important, it provide a measure to success. The goal is to keep fault-slip through as low as possible.	✓	✓	✓			SC1, SC2	LG6
SG6.	*Efficiency*:Running the planned scope for RT in a limited time.	✓	✓	✓	✓	✓	SC1, SC2, SC3	LG7

[1] CSC: Corresponding Success Criteria.
[2] CGL: Corresponding goals in the literature

Table 4.12: Regression testing success criteria.

ID	Criteria	SWSE	FCS	FWD	FWR	PTS
SC1.	*Right selection*: Based on requirements selecting and defining the right test cases. Selection of appropriate test cases is the key for successful RT.	✓	✓	✓		
SC2.	*Knowledge of changes*: For successful regression testing, QA should be well aware of changes in the system.	✓		✓		
SC3.	*Early start of RT*:The success criteria in RT is to start of RT as early as possible.			✓		
SC4.	*Coverage*: Coverage is one the criteria to evaluate the success.	✓	✓	✓		
SC5.	*Tester's Experience*: Knowledge and experience of tester is the subjective measure of confidence in RT.	✓	✓			
SC6.	*Customer's feedback*: The customer feedback is the measure of confidence in the software testing.	✓	✓			
SC7.	*Quality of test cases*: Carefully designed test cases can guarantee the finding issues and good coverage.	✓	✓			✓

We identified six regression testing goals (SG1–SG6) along with the seven success criteria. From Table 4.11 it is evident that the goals “Critical defect detection” (SG2), “Confidence” (SG3), “Effectiveness” (SG4), and “Efficiency” (SG6) are highlighted by the majority of the participating teams. It is interesting that “Customer satisfaction” (SG1) is highlighted by only two teams, whereas internal business perspectives (SG2, SG4, SG6) are highlighted by all five teams. This is surprising since the case companies are market-driven with customer satisfaction as a main success factor. The customers might not care about how much regression testing a company performs as long as the products work and the customers' expectations are fulfiled. Regarding regression testing success criteria (SC1–SC7), they were mainly defined by three teams (SWSE, FCS and FWR). The other teams defined only one success criterion and none, respectively. In Table 4.11, we mapped the success criteria to the respective goals and found that all criteria corresponded to goal “confidence”. However, confidence is a subjective term and difficult to measure. Testers could be confident about the testing results based on their experience and knowledge. We also identified RT success goals from related studies ([10, 31, 34], see Table 4.2) and related them to the RT success goals found in our work (see column CGL in Table 4.11).

These results are encouraging, although, initially, the majority of the practitioners responded that they don't use objective metrics. The practitioners are aware of the importance of success criteria and they could indicate how to measure them. Presently, they are not making objective assessments because of lack of environment support, availability of resources, and an appropriate mechanism for the assessment.

4.6 Summary and conclusions

We have conducted an interview-based study in two large communication companies and investigated various aspects of regression testing in practice. In the following, we summarize our findings regarding each research question.

The definition of RT that emerged from the practitioners **(RQ1)** (see Section 4.5.1) is in line with the standard definition presented in ISO, IEEE, system and software engineering vocabulary [7]. The goal of regression testing is to get confidence that the system changes have not introduced unwanted system behavior rather than to find errors. The scope of regression testing depends on the timing in the projects (e.g., small fix or a major release) and risk analysis for incoming changes. This study confirms our previous viewpoints [10] and Engström et al. [16].

Looking at the regression testing practices **(RQ2)**, our respondents are using manual and automated regression testing, there is a lot of manual work that practitioners have to do. For test automation mainly the companies are relying on in-house developed tools. Inability to find a third party tool is an interesting findings that points towards specific testing requirements or processes that can not be captured by a general regression testing tool support. Another possible explanation is that our respondents prefer to change the tools rather than wait for tool vendors to provide the necessary adaptations. This allows for faster and better alignment between the testing process, testing strategy and the tool support. Test strategy definition seems to be an ad hoc practice among our respondent which confirms the need for in-house and flexible tooling.

The testers and developers collaborate in the decision making regarding various aspects of RT. Only SWSE and FWR respondents confirmed to reuse test cases. This may explain how exploratory testing is used as a replacement for regression testing when they fail to find issues with fixed regression suites. Greater test reuse seems to be hindered by a lack of traceability labels (mentioned by one group) or tickets (also mentioned by one group), see Table 6 for details.

The scope and time of change drive regression testing activities. Our practitioners mostly applied a static regression testing scope to cover the basic/core functionality of the respective systems. Size, position and complexity of change drive test case selection, supported by domain knowledge and experience and dialog with developers. When pressed by short deadlines, our practitioners limit the scope of regression testing. The selection/prioritization criteria identified in our study are closely related to the selection/prioritization criteria described in related studies [16, 24, 25]. We noticed that for test coverage, our practitioners mentioned feature or module coverage. They did not talk about code coverage. A reason for that might be that the case companies are not developing the complete product-lines using in-house development. They are also using code provided by their external vendors. Therefore, the case companies might perform regression testing mainly at the system level, and the coverage criteria they are using are feature or module coverage.

Looking at the 12 identified challenges (RQ2.2), information management, test suite maintenance, communication, test case selection, and test case prioritization are common for both companies. We identified fourteen challenges from the related studies [16, 25]. Six of the challenges identified in our study can also be found in the related work (see Table 4.9). We have also identified relationships among the challenges (see Figure 4.3). There could be further relationships, for instance, a lack of communication might impact traceability. Similarly, traceability might affect test suite maintenance. A majority of the participants mentioned communication and test suite maintenance, but only two groups highlighted traceability as a challenge. This may mean that our respondents underestimate the role of traceability in regression testing. We believe that more research should be directed towards understanding the relations between the challenges and how the solutions can mitigate them.

We identified two improvement categories for regression testing practices **(RQ3)**:

1) Improvements related to management related challenges, 2) improvements related to technical interventions. The latter ones are related to test case selection, test case prioritization, test suite minimization, test suite augmentation, and assessment of regression testing. Yoo and Harman [24] presented a literature survey of 159 papers on test case selection, minimization and prioritization techniques, which includes a large number of techniques proposed in these three areas. Despite a lot of work in these areas, the practitioners still think that there is a need for improvement. This indicates that either the techniques proposed in the literature are not fulfilling the requirements of industry or that the practitioners are not aware of these techniques. The synthesis of industry-relevant research on regression testing by Ali et al. [34] is a step forward to communicate RT techniques to industry practitioners. Surprisingly, our respondents pointed out good verification tools as a necessary improvement despite developing in-house and heavily tailored solutions themselves. Another interesting aspect are irrelevant or obsolete test cases that appear to be a similar challenge. In a survey by Wnuk et al. [2], over 80% of the respondents confirmed negative influence of obsolescence on their processes.

The success goals identified in our study are, customer satisfaction, critical defect detection, confidence, effectivenss, controlled fault slip-through, and efficiency **(RQ4)**. Our study also reveals some preconditions which can guarantee the success of RT. For instance, right selection of test cases, and knowing the changes are the conditions that are essential for the success of RT. Regarding goals findings of this study complement the findings of previous studies [10, 31, 34]. Similarity in the success goals identified in two different studies indicate that there is an awareness and urge in the industry regarding evaluating the success of regression testing.

4.7 References

[1] A. Bertolino, "Software testing research: Achievements, challenges, dreams," in *Proceedings of the Workshop on the Future of Software Engineering (FOSE07*, 2007, pp. 85–103.

[2] K. Wnuk, T. Gorschek, and S. Zahda, "Obsolete software requirements," *Information and Software Technology*, vol. 55, no. 6, pp. 921 – 940, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584912002364

[3] N. B. Ali, K. Petersen, and M. V. Mäntylä, "Testing highly complex system of systems: an industrial case study," in *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*. IEEE, 2012, pp. 211–220.

[4] A. Orso and G. Rothermel, "Software testing: a research travelogue (2000–2014)," in *Proceedings of the Workshop on Future of Software Engineering (FOSE14)*, 2014, pp. 117–132.

[5] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Software Engineering*, vol. 10, no. 4, pp. 405–435, 2005.

[6] I. S. C. Committee *et al.*, "Ieee standard glossary of software engineering terminology (ieee std 610.12-1990). los alamitos," *CA: IEEE Computer Society*, 1990.

[7] ISO/IEC, "ISO/IEC/IEEE24765:2010: Systems and software engineering–vocabulary," 2010.

[8] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*. IEEE, 2011, pp. 275–284.

[9] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[10] N. M. Minhas, K. Petersen, N. B. Ali, and K. Wnuk, "Regression testing goals-view of practitioners and researchers," in *24th Asia-Pacific Software Engineering Conference Workshops (APSECW), 2017*. IEEE, 2017, pp. 25–31.

[11] A. C. Dias-Neto, S. Matalonga, M. Solari, G. Robiolo, and G. H. Travassos, "Toward the characterization of software testing practices in south america: looking at brazil and uruguay," *Software Quality Journal*, vol. 25, no. 4, pp. 1145–1183, 2017.

[12] M. Kassab, J. F. DeFranco, and P. A. Laplante, "Software testing: The state of the practice," *IEEE Software*, vol. 34, no. 5, pp. 46–52, 2017.

[13] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," in *Proceedings of the 7th International Workshop on Automation of Software Test*. IEEE Press, 2012, pp. 36–42.

[14] E. Juergens, B. Hummel, F. Deissenboeck, M. Feilkas, C. Schlogel, and A. Wubbeke, "Regression test selection of manual system tests in practice," in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR)*, 2011, pp. 309–312.

[15] J. Kasurinen, O. Taipale, and K. Smolander, "Software test automation in practice: empirical observations," *Advances in Software Engineering*, vol. 2010, 2010.

[16] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *Proceedings of the International Conference on Product Focused Software Process Improvement*. Springer, 2010, pp. 3–16.

[17] J. Rooksby, M. Rouncefield, and I. Sommerville, "Testing in the wild: The social and organisational dimensions of real world practice," *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5-6, p. 559, 2009.

[18] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen, "A preliminary survey on software testing practices in australia," in *Software Engineering Conference, 2004. Proceedings. 2004 Australian*. IEEE, 2004, pp. 116–125.

[19] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proceedings of the Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2011, pp. 99–102.

[20] E. Engström, P. Runeson, and G. Wikstrand, "An empirical evaluation of regression testing based on fix-cache recommendations," in *Proceedings of the Third International Conference on Software Testing, Verification and Validation (ICST), 2010*. IEEE, 2010, pp. 75–78.

[21] E. Engström, K. Petersen, N. bin Ali, and E. Bjarnason, “Serp-test: A taxonomy for supporting industry-academia communication,” *Software Quality Journal*, pp. 1–37, 2016.

[22] X. Lin, “Regression testing in research and practice,” *Computer Science and Engineering Department University of Nebraska, Lincoln*, pp. 1–402, 2007.

[23] D. Parsons, T. Susnjak, and M. Lange, “Influences on regression testing strategies in agile software development environments,” *Software Quality Journal*, vol. 22, no. 4, pp. 717–739, 2014.

[24] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[25] M. J. Harrold and A. Orso, “Retesting software during development and maintenance,” in *Proceedings of Frontiers of Software Maintenance FoSM*. IEEE, 2008, pp. 99–108.

[26] R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, “Effective regression test case selection: A systematic literature review,” *ACM Comput. Surv.*, vol. 50, no. 2, pp. 29:1–29:32, 2017.

[27] K. Petersen and C. Wohlin, “A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case,” *Journal of systems and software*, vol. 82, no. 9, pp. 1479–1490, 2009.

[28] D. Brahneborg, W. Afzal, and A. Čaušević, “A pragmatic perspective on regression testing challenges,” in *Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on*. IEEE, 2017, pp. 618–619.

[29] B. Kitchenham and S. L. Pfleeger, “Principles of survey research: part 5: populations and samples,” *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 5, pp. 17–20, 2002.

[30] S. Ghaisas, P. Rose, M. Daneva, K. Sikkel, and R. J. Wieringa, “Generalizing by similarity: Lessons learnt from industrial case studies,” in *Proceedings of the 1st international workshop on conducting empirical studies in industry*, 2013, pp. 37–42.

[31] S. Jafrin, D. Nandi, and S. Mahmood, “Test case prioritization based on fault dependency,” *International Journal of Modern Education and Computer Science*, vol. 8, no. 4, p. 33, 2016.

[32] M. Khatibsyarbini, M. A. Isa, D. N. Jawawi, and R. Tumeng, “Test case prioritization approaches in regression testing: A systematic literature review,” *Information and Software Technology*, 2017.

[33] S. U. R. Khan, S. P. Lee, N. Javaid, and W. Abdul, “A systematic review on test suite reduction: Approaches, experiment’s quality evaluation, and guidelines,” *IEEE Access*, vol. 6, pp. 11 816–11 841, 2018.

[34] N. bin Ali, E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, S. Kunze, and M. Varshosaz, “On the search for industry-relevant regression testing research,” *Empirical Software Engineering*, pp. 1–36, 2019.

Appendix B

Interview guide

Introduction

Personal introduction Interviewers tell the participant about themselves, their background and training, and interest in the area of software testing.

Study goal The lead interviewer explains the study goal to the participant.

Goal: The goal of the study is to know the state of regression testing practice in the large-scale embedded software development. The purpose is to know that how companies are managing their test systems, specially with reference to regression testing. The following points are the focus of the interview.

- ○ Current Practices,
- ○ Test Selection and prioritization,
- ○ Challenges and issues,
- ○ Improvements, and
- ○ Evaluation of success goals.

Benefit: This study will provide the basis for improving the different aspects of regression testing considering the different views of people within the organization. We believe your opinion is valuable. This investigation gives you (interviewee) a chance to contribute to the improvement of the regression testing environment.

Interview process Interviewer describes the overall process, that how the interview will take place.

○ *Interview duration:* The interview will be completed in about an hour time.

○ *Interview questions:* there may be some questions that the interviewee perceives as stupid, silly, or difficult to answer. It is possible that an appropriate question for one person may not be suitable for the other.

○ *Counter questions:* The interviewee may feel free to ask counter questions for the clarification of an interview question and can disagree with any statement of the interviewer.

○ *Answers:* We believe that in an interview, we can not rate any answer as right or wrong. The interviewee need not worry about in this regard, and we expect he/she will answer the questions on the basis of knowledge and experience.

Respondent background

The purpose of this section is to know about the participant's professional background, current role and responsibilities.

Question 1: Could you please briefly describe on your professional background?

○ Your qualification,
○ Overall Experience,
○ Time in the current company.

Question 2: How will you define your expertise?

○ Software Engineering,
○ Software Development,
○ Software testing.

Question 3: Please specify about your current job.

○ Your current team,
○ Your role in the team.

Question 4: Can you please brief us about your current project(s).

Interview part to explore the RT state of practice

This is the central part of this interview, and we are interested to know about the current practice, selection and prioritization criteria, challenges, and improvements regarding regression testing. In the final part of the interview, we will discuss the regression testing success goals. We will start by asking our questions regarding current practices. Please feel free to add detail at any point of the interview that you think we missed to ask or you forget to describe.

Defining regression testing The purpose is not to get the academic definition of regression testing. The interviewers are interested to know the perception of the practitioner.
Question 1: What is regression testing for you?

Questions regarding current practices The aim is to know how practitioner's team is performing regression testing.
Question 1: Can you please give us a walk-through of overall process and highlight where and how often regression testing is performed?
Question 2: Have you been involved in decision making regarding (When to do regression testing? Which test cases should be executed? How to select a subset of candidate test cases?)?
Question 3: In your team regression testing is manual or automated?
Question 4: For automation which tools are in use of your company / team?
Question 5: Decision are taken by the individuals or by the QA team? (Who are the people involved in decision making?)

Selection and prioritization Although, selection and prioritization are regarding as the part of practice. Considering the importance of selection and prioritization, we are asking focused questions.
Question 1: Do you use some predefined criteria for the selection and / or prioritization of test cases?
Question 2: Which information you use while making decisions for selection and/or prioritization?
Question 3: Do you or your team maintain the required information? Is this information readily available?
Question 4: When have you made the right selection of test cases?
Question 5: How do you evaluate / know whether the selection was right?

Challenges and improvements Challenges are the obstacles that can hinder the smooth and successful execution of the operations. Like other working environments, practitioners working in software development organizations are facing different issues. Our interest is to know those issues which are recurring and require attention.
Question 1: What are the challenges for testers regarding regression testing?
Question 2: Do you have any mitigation strategies to overcome these challenges?
Question 3: What are the challenges, you think need to pay more attention?

Question 4: Considering the underlying challenges, can you identify the areas of improvement?

Success Criteria To determine the success of any activity, we measure it with the predefined goals, that is, if the goals have met or not.
Question 1: What is your criteria of success of regression testing? Do you measure the success?
Question 2: At your company / team do you define success goals?
Question 3: For a successful regression testing what are the goals that should be achieved?
Question 4: How will you determine that the desired goals have been achieved?

Closing Questions We mentioned earlier that the goal of this research is to identify potential problems and come up with suggested solutions. Your opinion counts!
Question 1: In your opinion which is the most important area that should have to be the focus of this research?
Question 2: Do you want to share some more information which you think is important to consider, that we may have missed?

Chapter 5

On the search for industry-relevant regression testing research

5.1 Introduction

Regression testing remains an unsolved and increasingly significant challenge in industrial software development. As a major step towards quality assurance, regression testing poses an important challenge for the seamless evolution (e.g., continuous integration and delivery) of large-scale software. Similarly, dealing with variability (e.g., in software product lines/product variants) makes regression testing of industrial software a non-trivial matter. Testing is highly repetitive at all levels and stages of the development, and for large and complex systems precision in regression test scoping becomes crucial.

These challenges have led to a large body of academic research. There is even a multitude of systematic literature reviews classifying and analysing the various proposed techniques for regression testing. For example, there are eleven literature reviews on regression testing published since 2010 ([16, 17, 36, 39, 43, 54, 66, 70, 72, 78, 79]).

Despite this extensive body of research literature, research results have shown to be hard to adopt for the practitioners ([5–8, 74, 76]). First of all, some results are not accessible for practitioners due to the discrepancies in terminology between industry and academia, which in turn makes it hard to know what to search for in the research literature. Furthermore, many empirical investigations are done in controlled experimental settings that have little in common with the complexity of an industrial setting. Hence, for practitioners, the relevance of such results is hard to assess. [74] surveyed regression testing practices, which highlighted the variation in regression testing contexts and the need for holistic industrial evaluations.

There are today a significant number of industrial evaluations of regression testing. Unfortunately, also these results are hard to assess for the practitioners, since there are no conceptual models verified by practitioners to interpret, compare, and contrast different regression testing techniques. [8] conducted an in-depth case study on the procedures undertaken at a large software company to search for a relevant regression testing technique and to evaluate the benefits of introducing it into the testing process at the company. This study further emphasises the need for support in matching the communication of empirical evidence in regression testing with guidelines for identifying context constraints and desired effects that are present in practice.

To respond to this need, in this paper, we review the literature from a relevance and applicability perspective. Using the existing literature reviews as a seed set for snowball sampling [20], we identified 1068 papers on regression testing, which are potentially relevant for our study. To gain as many insights as possible about relevance and applicability we have focused the review on large-scale industrial evaluations of regression testing techniques, as these studies in many cases involve stakeholders and are more likely to report these issues.

Both relevance and applicability are relative to a context, and we are not striving to find a general definition of the concepts. In our study, we are extracting factors that may support a practitioner (or researcher) in assessing relevance and applicability in their specific cases. We define relevance as a combination of desired (or measured) effects and addressed context factors and include every such factor that have been reported in the included studies. Similarly, applicability, or the cost of adopting a technique, may be assessed by considering the information sources and entities utilised for selecting and/or prioritising regression tests. For each of these facets, we provide a taxonomy to support classification and comparison of techniques with respect to industrial relevance and applicability of regression testing techniques.

The original research questions stem from an industry-academia collaboration[1] (involving three companies and two universities) on decision support for software testing. Guided by the SERP-test taxonomy [75], a taxonomy for matching industrial challenges with research results in software testing, we elicited nine important and challenging decision types for testers, of which three are instances of the regression testing

[1]EASE- the Industrial Excellence Centre for Embedded Applications Software Engineering `http://ease.cs.lth.se/about/`

challenge as summarised by [17]: regression test minimisation, selection, and prioritisation. These challenge descriptions (i.e., the generic problem formulations enriched with our collaborators' context and target descriptions) guided our design of the study.

To balance the academic view on the regression testing literature, we consulted practitioners in all stages of the systematic review (i.e., defining the research questions, inclusion and exclusion criteria, as well as the taxonomies for mapping selected papers).

The main contributions provided in this report are:

- three taxonomies designed to support the communication of regression testing research with respect to industrial relevance and applicability, and
- a mapping of 26 industrially evaluated regression testing techniques (in total 38 different papers) to the above-mentioned taxonomies.

The remainder of the paper is structured as follows: Section 5.2 summarises previous research on assessing the industrial relevance of research. It also presents an overview of existing systematic literature reviews on regression testing. Research questions raised in this study are presented in Section 5.3. Section 5.4 and Section 5.5 detail the research approach used in the study and its limitations, respectively. Sections 5.6 to 5.8 present the results of this research. Section 5.9 and Section 5.10 present advice for practitioners and academics working in the area of regression testing. Section 5.11 concludes the paper.

5.2 Related work

In this section, we briefly describe related work that attempts to evaluate the relevance of software engineering research for practice. We also discuss existing reviews on regression testing with a particular focusing on the evaluation of the industrial relevance of proposed techniques.

5.2.1 Evaluation of the industry relevance of research

Software engineering being an applied research area continues to strive to establish the industrial practice on scientific foundations. Along with the scientific rigour and academic impact, several researchers have attempted to assess the relevance and likely impact of research on practice.

[62] proposed a method to assess the industrial relevance of empirical studies included in a systematic literature review. The criteria for judging relevance in thier

proposal evaluates the realism in empirical evaluations on four aspects: 1) subjects (e.g. a study involving industrial practitioners), 2) context (e.g. a study done in an industrial settings), 3) scale (e.g. evaluation was done on a realistic size artifacts) and 4) research method (e.g. use of case study research). Several systematic reviews have used this approach to assess the applicability of research proposals in industrial settings (e.g. [1, 85]).

Other researchers have taken a different approach and have elicited the practitioners' opinion directly on individual studies ([59–61]). In these studies, the practitioners were presented a summary of the articles and were asked to rate the relevance of a study for them on a Likert scale.

The *Impact project* was one such initiative aimed to document the impact of software engineering research on practice [46]. Publications attributed to this project, with voluntary participation from eminent researchers, covered topics like configuration management, inspections and reviews, programming languages and middle-ware technology. The approach used in the project was to start from a technology that is established in practice and trace its roots, if possible, to research [46]. However, the last publications indexed on the project page[2] are from 2008. One of the lessons learned from studies in this project is that the organisations wanting to replicate the success of other companies should *"mimic successful companies' transfer guidelines"* ([40, 46]). Along those lines, the study presently read attempts to identify regression testing techniques with indications of value and applicability from industrial evaluations [57].

To address the lack of relevance, close industry-academia collaboration is encouraged ([19, 46, 57]). One challenge in this regard is to make research more accessible to practitioners by reducing the communication-gap between industry and academia [75]. SERP-test [75] is a taxonomy designed to support industry academia communication by guiding interpretation of research results from a problem perspective.

5.2.2 Reviews of regression testing research

We identified eleven reviews of software regression testing literature ([16, 17, 36, 39, 43, 54, 66, 70, 72, 78, 79]). Most of these reviews cover regression testing literature regardless of the application domain and techniques used. However, the following four surveys have a narrow scope: [43] and [16] target testing web-based applications, and [70] focus on identifying security-related issues, while [79] only considers literature where researchers have used Genetic Algorithms for regression testing. The tertiary study by [68] only maps the systematic literature studies in various sub-areas of software testing including regression testing. Instead of focusing only on regression

[2] `https://www.sigsoft.org/impact.html`

testing research, [47] reviewed the literature on test case selection in general. They identified that only six of the selected studies were performed on large-scale systems, and only four of these were industrial applications.

In the most recent literature review, [54] reviewed empirical research on regression testing of industrial and non-industrial systems of any size. They mapped the identified research to the following dimensions: evaluation metrics used in the study, the scope of the study, and what they have termed as the theoretical basis of the study (research questions, regression testing technique, SUT, and the dataset used). Their approach indicates a similar aim as other literature reviews: to identify "the most effective" technique considering the measures of "cost, coverage and fault detection". However, they do not take into consideration the aspect of the relevance and likely applicability of the research for industrial settings.

Among the identified reviews, only five discuss aspects related to the industrial application ([17, 36, 66, 72, 78]). [78] found that 64% of the included 120 papers used datasets from industrial projects in their evaluation. They further recommend that future evaluations should be based on non-proprietary data sets that come from industrial projects (since these are representative of real industrial problems) [79]. [17] identified that a large majority of empirical studies use a small set of subjects largely from the SIR[3] repository. They highlight that it allows comparative/replication studies, and also warn about the bias introduced by working with the same small set of systems. Similarly, [72] concluded that most empirical investigations are conducted on small programs, which limits the generalisation to large-systems used in industry. [36] also found that 50% of the 65 selected papers on regression test prioritisation included in their review use SIR systems. Furthermore, 29% of the studies use the same two systems from the repository.

[66] reviewed the state of research and practice in regression testing. Authors presented the synthesis of main regression testing techniques and found that only a few techniques and tools developed by the researchers and practitioners are in use of industry. They also discussed the challenges for regression testing and divided the challenges into two sets (transitioning challenges and technical/conceptual issues). Along with the review of research on regression testing authors also presented the results of their discussions (an informal survey) with researchers and practitioners. They were intended to understand the impact of existing regression testing research and the major challenges to regression testing.

Unlike existing literature reviews, this study has an exclusive focus on research conducted in industrial settings. This study provides taxonomies to assist researchers in designing and reporting research to make the results more useful for practitioners.

[3] Software Infrastructure Repository http://sir.unl.edu/

Using the proposed taxonomies to report regression testing research, will enable synthesis in systematic literature reviews and help to take the field further. One form of such synthesis will be the technological-rules [33] (as extracted in this paper) with an indication of the strength-of-evidence. For practitioners, these taxonomies allow reasoning about the applicability of research for their own unique context. The study also presents some technological-rules that are based on the results of this study which practitioners can consider research-informed recommendations from this study.

5.3 Research questions

In this study, we aggregate information on regression testing techniques that have been evaluated in industrial contexts. Our goal is to structure this information in such a way that it supports practitioners to make an informed decision regarding regression testing with a consideration for their unique context, challenges, and improvement targets. To achieve this goal we posed the following research questions:
RQ1: *How to describe an industrial regression testing problem?* Regression testing challenges are described differently in research ([38]) and practice ([74]). To be accessible and relevant for practitioners, research contributions in terms of technological rules ([33]) need to be interpreted and incorporated into a bigger picture. This, in turn, requires alignment in both the abstraction level and the terminology of the academic and industrial problem descriptions. To provide support for such alignment, we develop taxonomies of desired effects and relevant context factors by extracting and coding knowledge on previous industrial evaluations of regression testing techniques.
RQ2: *How to describe a regression testing solution?* Practitioners need to be able to compare research proposals and assess their applicability and usefulness for their specific contexts. For this purpose, we extract commonalities and variabilities of research proposals that have been evaluated in industry.
RQ3: *How does the current research map to such problem description?* To provide an overview of the current state of the art, we compare groups of techniques through the lens of the taxonomies developed in RQ1 and RQ2.

5.4 Method

To capture what information is required to judge the industrial-relevance of regression testing techniques, we relied on: 1) industrial applications of regression testing techniques reported in the literature, 2) existing research on improving industry-academia collaboration in the area of software testing, 3) and close cooperation with practitioners.

To develop the three taxonomies presented in Section 5.6 and 5.7 and arrive at the results presented in Section 5.8 we conducted a systematic literature review of regression testing research, interleaving interaction with industry practitioners throughout the review process.

The process followed can be divided into six steps, which are visualised in Figure 5.1. Research questions were initially formulated within a research collaboration on decision support for software testing (EASE). To validate the research questions and the approach of constructing a SERP taxonomy [75] a pilot study was conducted (Step 1, Section 5.4.3). Based on the pilot study, a preliminary version of the taxonomy was presented to the researchers and practitioners in EASE, together with a refined study design for the extensive search. Through the extensive search of the literature (Step 2, Section 5.4.4) we identified 1068 papers on regression testing. This set was then reduced (Step 3, Section 5.4.5) by iteratively coding and excluding papers while refining the taxonomy (Step 4, Section 5.4.6). Finally, the constructed taxonomies were evaluated in a focus group meeting (Step 5, Section 5.4.7) and the regression testing techniques proposed in the selected papers were mapped to the validated version of the taxonomies (Step 6, Section 5.4.8).

5.4.1 Practitioners' involvement

As shown in Figure 5.1 (ovals with shaded background), practitioners were involved in three steps. For validating the selection criteria (Step 3a) a subset of selected papers was validated with practitioners. In Step 4a, the initial taxonomy was presented to EASE partners in a meeting. This meeting was attended by five key stakeholders in testing at the case companies. In Step 5, for taxonomy evaluation, we relied on a focus group with three key practitioners. The three practitioners came from two companies which develop large-scale software-intensive products and proprietary hardware. The participating companies are quite different from each other; *Sony Mobile Communications* has a strict hierarchical structure, well-established processes and tools, and is globally distributed, while the development at *Axis Communications AB, Sweden* still has the entrepreneurial culture of a small company and has less strictly defined processes. The profiles of the practitioners involved in the study are briefly summarized below:

Practitioner P1 is working at Axis. He has over eight years of experience in software development. At Axis, he is responsible for automated regression testing from unit to system-test levels. His team is responsible for the development and maintenance of the test suite. The complete regression test suite comprises over 1000 test

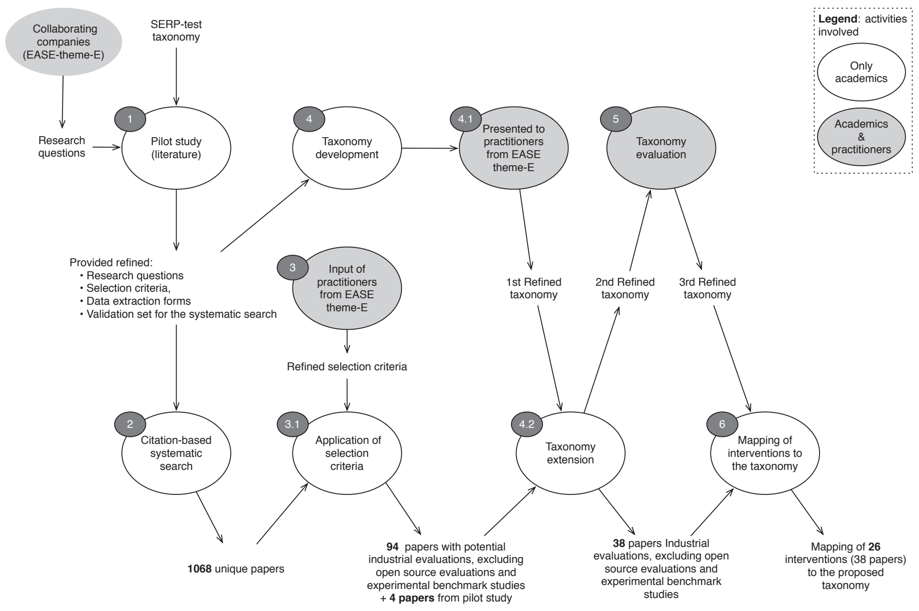
Collaborating companies (EASE-theme-E)
SERP-test taxonomy
Research questions
1 Pilot study (literature)
4 Taxonomy development
4.1 Presented to practitioners from EASE theme-E
5 Taxonomy evaluation
Legend: activities involved
Only academics
Academics & practitioners
Provided refined:
• Research questions
• Selection criteria,
• Data extraction forms
• Validation set for the systematic search
3 Input of practitioners from EASE theme-E
Refined selection criteria
1st Refined taxonomy
2nd Refined taxonomy
3rd Refined taxonomy
2 Citation-based systematic search
3.1 Application of selection criteria
4.2 Taxonomy extension
6 Mapping of interventions to the taxonomy
1068 unique papers
94 papers with potential industrial evaluations, excluding open source evaluations and experimental benchmark studies + 4 papers from pilot study
38 papers Industrial evaluations, excluding open source evaluations and experimental benchmark studies
Mapping of 26 interventions (38 papers) to the proposed taxonomy

Figure 5.1: A description of the flow of activities including alternately reviewing the literature and interacting with practitioners from the research project EASE.

cases that take around 7 hours to execute. He was also involved in previous research-based initiatives to improve regression testing at Axis [76].

Practitioner P2 also works at Axis communications. He has over 12 years of software development and testing experience. He is responsible for both automated and manual regression testing at the system-test level. He has recently overseen a complete assessment and review of the manually executed test-cases in the regression test suite.

Practitioner P3, works at Sony Mobile Communications. He has over 18 years of experience in software development with responsibilities primarily include software testing and overall automation and verification strategies. His current role as verification architect covers testing at all levels including regression testing. Within the EASE project, he has collaborated with researchers in several research-based investigations at his company.

5.4.2 Need for a literature review

The broader context of this study is a collaborative research project EASE (involving two academic and three industrial partners) working towards decision support in the context of software testing. As shown in Figure 5.1, the research questions and the need for a systematic literature review were identified in the context of this project. We considered the literature to answer the following two questions in the pilot study:

1. *Have existing systematic literature reviews taken into consideration the industrial relevance and applicability of regression testing techniques?* We identified 11 systematic literature studies ([16, 17, 36, 39, 43, 54, 66, 70, 72, 78, 79]), and they have been briefly discussed in Section 5.2. These studies have not addressed the research questions of interest for the current study.

2. *Are there sufficient papers reporting an industrial evaluation of regression testing techniques?* Once we had established the need to analyse the existing research from the perspective of industrial relevance, we conducted a pilot study to:

   - identify if there are sufficiently many published papers to warrant a systematic literature review,
   - develop an initial taxonomy that serves as a data extraction form in the main literature review, and
   - identify a set of relevant papers that serve as a validation set for our search strategy in the main literature review.

5.4.3 Pilot study

By manually searching through recent publications of key authors (identified in previous literature reviews discussed in Section 5.2) and by skimming through the top most-relevant results of keyword-based searches in Google Scholar, we identified 36 papers. Using a data extraction form based on the SERP-test taxonomy [75], data were extracted from these papers. Data extraction was done independently by at least two reviewers and results were consolidated by discussion. This validation was considered useful for two reasons: firstly, through the cross-validation, we developed a shared understanding of the process. Secondly, since the results were to be used as a guide for data extraction in the main literature review, it was necessary to increase the reliability of this initial step.

The pilot study indicated that sufficient literature exists to warrant a systematic literature review. The results of analysing the extracted information were useful for formulating the data extraction forms for the main literature review.

5.4.4 Search strategy

Using the following search string, we identified the existing systematic literature studies on regression test optimization as listed in Table 5.1:
(“regression test” OR “regression testing”) AND (“systematic review” OR “research review” OR “research synthesis” OR “research integration” OR “systematic review” OR “systematic overview” OR “systematic research synthesis” OR “integrative research review” OR “integrative review” OR “systematic literature review” OR “systematic mapping” OR “systematic map”))

Additionally, we also used [17] survey as it has a thorough coverage (with 189 references) and is the most-cited review in the area of regression testing. Using the references in the papers listed in Table 5.1, and the citations to these papers were retrieved in August 2016 from Google Scholar. We identified a set of 1068 papers as potentially relevant papers for our study. One of the systematic reviews, by [54] as discussed in Section 5.2, was not used for snowball-sampling as it was published yet when the search was conducted.

Table 5.1: Systematic literature studies used as start-set for snowball sampling

ID	No. of References.	No. of Citations
[70]	75	5
[43]	69	1
[16]	71	0
[79]	24	4
[36]	80	14
[78]	24	25
[72]	73	135
[47]	46	1
[39]	59	0
[17]	189	515

Using the 36 papers identified in the pilot-study (see Section 5.4.3) as the validation-set for this study, we calculated the precision and recall ([15, 53]) for our search strategy. 36 papers in a validation-set are reasonable for assessing the search strategy of a systematic literature review [53].

Recall = *100* ***** *(# of papers from the validation-set identified in the search)* **/** *(total # of papers in the validation set).*

Precision = *100 * (total # of relevant papers (after applying the selection criteria) in the search results)* **/** *(total # of search results).*

$$Recall = \frac{32}{36} * 100 = 89\%$$

Only four of the papers in the pilot-study were not identified by our search strategy ([18, 37, 48, 49]). These papers neither cite any of the literature reviews nor were they included by any of the literature reviews comprising the starting set for search in this study. We also included these four papers to the set of papers considered in this study.

As shown in Figure 5.1, after applying the selection criteria 94 relevant papers were identified. These papers were used to extend the taxonomy. Using this number, we calculated the precision of our search strategy as follows:

$$Precision = \frac{94}{1068} * 100 = 8\%$$

An optimum search strategy should maximise both precision and recall. However, our search strategy had high recall (with 89% recall it falls in the high recall range, i.e. $\geq$ 85% [15]) and low precision. The precision value was calculated considering the 94 papers that were used in extending the taxonomies.

The value of recall being well above the acceptable range [15] of 80% adds confidence to our search strategy. Furthermore, such low value of precision is typical of systematic literature reviews in software engineering e.g. approx. 5% [79] approx. 2% ([9, 85]), and below 1% ([36, 43, 72]).

5.4.5 Selection of papers to include in the review

We applied a flexible design of the study and inclusion criteria were iteratively refined. The notion of "industrial" was further elaborated after the first iteration. To make the set of papers more manageable, we decided to exclude open source evaluations and industrial benchmark studies. The assumption was that such reports contain less information about application context and limitations in terms of technology adoption. The following inclusion-exclusion criteria were the ones finally applied:

- **Inclusion criteria:** peer-reviewed publications that report empirical evaluations of regression testing techniques in industrial settings. It was detailed as the following, include papers that:
  - are peer-reviewed (papers in conferences proceedings and journal articles)
  - report empirical research (case studies, experiments, experience reports ...)

  - report research conducted in industrial settings (i.e. uses a large-scale software system, involves practitioners or reports information on the real application context including the process).
  - investigate regression testing optimization techniques (i.e. regression test selection, prioritization, or minimization/ reduction/ maintenance)

- **Exclusion:** exclude papers that:
  - are non-peer reviewed (Ph.D. thesis, technical reports, books etc.)
  - report a non-empirical contribution (analytical/ theoretical/ proposals)
  - report evaluation in non-industrial settings.

We decided to use lines of code (LOC), if reported, as an indicator for the scale of the problem instead of the number of test cases in the test suite or turnaround time of a test suite (and similar metrics) for the following reasons:

- LOC/kLOC is the most commonly reported information regarding the size of a SUT.
- Size and execution time of individual test cases in a test suite varies a lot, therefore, an aggregate value reporting the number of test cases or the execution time of test cases is not very informative.

Techniques that work well on a small program may work on large programs. However, this is yet to be demonstrated. Practitioners seem to trust the results of research conducted in environments similar to their [2]. Previous research on assessing the industrial relevance of research has also relied on the realism in the evaluation setting regarding the research method, scale, context and users ([62, 85]).

We performed pilot selection on three papers to validate the selection criteria and to develop a shared understanding among the authors. Each author independently applied the selection criteria on the these randomly chosen papers. We discussed the decisions and reflected on the reasons for any discrepancies among the reviewers in a group format.

After the pilot-selection, remaining papers were assigned to each author randomly to apply selection criteria. Inclusion-exclusion was performed at three levels of screening: ‘Titles only’, ‘Titles and abstracts only’, and ‘Full text’. If in doubt, the general instruction was to be more inclusive and defer the decision to the next level. Each excluded paper was evaluated by at least two reviewers.

Additionally, to validate that the studies we were selecting were indeed relevant, during the paper selection phase of this study, a sample of eight papers from the included papers was shared with practitioners. They labelled the paper as relevant or

Table 5.2: The list of papers included in this study

Study ID	Reference.	Study ID	Reference.
S1	[76]	S20	[41]
S2	[37]	S21	[52]
S3	[48]	S22	[32]
S4	[49]	S23	[58]
S5	[81]	S24	[71]
S6	[35]	S25	[21]
S7	[23]	S26	[73]
S8	[22]	S27	[30]
S9	[13]	S28	[63]
S10	[11]	S29	[34]
S11	[14]	S30	[64]
S12	[12]	S31	[45]
S13	[24]	S32	[65]
S14	[27]	S33	[51]
S15	[28]	S34	[84]
S16	[26]	S35	[50]
S17	[29]	S36	[77]
S18	[25]	S37	[80]
S19	[42]	S38	[67]

irrelevant for their companies and also explained their reasoning to us. This helped us to improve the coverage of information that practitioners are seeking, which they consider will help them make informed decisions regarding regression testing.

After applying the selection criteria on 1068 paper and excluding open source and industrial benchmarks we had 94 remaining papers. Four papers from the pilot-study were also added to this list. These 98 papers were randomly assigned to the authors of this paper for data-extraction and taxonomy extension. After full-text reading and data extraction, 38 papers were included as relevant papers (see list in Table 5.2), which represent 26 distinct techniques. All excluded papers were reviewed by an additional reviewer.

5.4.6 Taxonomy extension

Table 5.3 presents an abstraction of the data extraction form, which was based on the first version of our taxonomy that was developed in the pilot study (see Step-4 onwards in Figure 5.1 that produced the “1st Refined taxonomy" and Section 5.4.3 for details of the pilot study). We followed the following steps to validate the extraction form and to develop a shared understanding of it:

1. Select a paper randomly from the set of potentially relevant papers.
2. All reviewers independently extract information from the paper using the data extraction form.
3. Compare the data-extraction results from individual reviewers.
4. Discuss and resolve any disagreements and if needed update the data extraction form.

This process was repeated three times before we were confident in proceeding with data extraction on the remaining set of papers.

Table 5.3: Data extraction form

Item	Value	Remarks
1) Meta information		
2) Description of testing technique		
3) Scope of technique		
4) High-level Effect/Purpose		
5) Characteristics of the SUT		
6) Characteristics of the regression testing process		
7) Required sources of information		
8) Type of required information		
9) Is this an industrial study?		
10) If yes, could the SUT be categorised as closed source?		
11) Is the paper within the scope of the study? If not, please explain the reason.		

The extracted information was used to develop extensions of SERP-test taxonomy [75] relevant to our focus on regression testing techniques. Separate taxonomies for “addressed context factors”, “evaluated effects” and “utilised information sources” were developed (shown as step 4.2 in Figure 5.1). The initial version of these taxonomies was developed in a workshop where six of the authors participated. Each of the taxonomies were then further refined by two of the authors and reviewed independently by a different pair of authors. This resulted in what is referred to as “2nd refined taxonomy" in Figure 5.1. This version of the taxonomy was further validated with practitioners, which is discussed in the following section.

5.4.7 Taxonomy evaluation

Once data analysis was complete, and we had created the three taxonomies presented in Section 5.6, Section 5.7 and Section 5.8, we conducted a focus group with three key stakeholders from the companies (brief profiles are presented in Section 5.4.1). In this focus group, moderated by the second author, we systematically collected practitioners' feedback on the context and effect taxonomies because these two taxonomies are supposed to describe the practitioners' need.

Practitioners were asked to assess the relevance of each of the nodes in the taxonomies (as presented in Table 5.4) and grade these from 1 to 3, where 1) means very relevant (i.e. we are interested in this research), 2) possibly relevant and 3) means irrelevant (i.e. we are not interested in such research). The practitioners were asked to respond based on their experience and not only based on their current need.

The feedback from the practitioners was taken into account, and some refinements to the taxonomies were made based on it. As this is primarily a literature review, we decided not to add factors that were not presented in the included papers although initial feedback pointed us to relevant factors in the studies. Neither did we remove factors completely from the taxonomies (although we removed some levels of detail in a couple of cases). The feedback was mainly used to evaluate and improve understandability of the taxonomies and changes were mainly structural.

5.4.8 Mapping of techniques to taxonomy

As shown in Figure 5.1 after incorporating the feedback from the practitioners in the taxonomy, we mapped the 26 techniques to our multi-faceted taxonomy. The reviewer(s) (one of the authors of the study) who were responsible for data extraction from the papers reporting the technique mapped the paper to the taxonomy. Two additional reviewers validated the mapping, and disagreements were resolved through discussion and by consulting the full-text of the papers. The results of the mapping are presented in Table 5.5.

5.5 Limitations

In this section, we discuss validity threats, our mitigation actions, and the limitations of the study.

5.5.1 Coverage of regression testing techniques:

To identify regression testing techniques that have been evaluated in industrial settings, we used snowball sampling search strategy. Snowball sampling has been effectively used to extend systematic literature reviews ([69]). The decision to pursue this strategy was motivated by the large number of systematic literature studies (as discussed previously in Section 5.2) available on the topic. Some of these reviews (e.g. [17] and [72]) are well cited, indicating visibility in the community. This increases the likelihood of finding recent research on the topic.

The search is not bound to a particular venue and is restricted to citations indexed by Scopus and Google Scholar before August 2016. We choose Scopus and Google scholar because of their comprehensive coverage of citations [4]. We are also confident in the coverage of the study as out of the 36 papers in the validation set, only four were not found (see Section 5.4).

To reduce the possibility of excluding relevant studies, we performed pilot selection on a randomly selected subset of papers. Furthermore, all excluded papers were reviewed independently by at least two of the authors of this paper. In cases of disagreement, the papers were included in the next phase of the study, i.e. data extraction and analysis.

5.5.2 Confidence in taxonomy building process and outcome

The taxonomies presented in this paper were based on data extracted from the included studies. To ensure that no relevant information was omitted, we tested the data extraction form on a sample of papers. This helped to develop a shared understanding of the form.

Furthermore, to increase the reliability of the study, the actual data extraction (from all selected papers) and the formulation of facets in the taxonomies were reviewed by two additional reviewers (authors of this paper).

As shown in Figure 5.1, the intermediate versions of the taxonomy were also presented to practitioners and their feedback was incorporated in the taxonomy. Possible confounding effects of their participation is due to their representativeness. The impact of practitioner feedback was mainly on the understandability and level of detail of the proposed taxonomies and a confounding effect could be that the understandability of the taxonomy is dependant of dealing with a context similar to our practitioners' . The two companies are large-scale and the challenges they face are typical for such contexts [75, 86]. All participants have many years of experience of testing (as described in Sec 5.4.1). Therefore, their input is considered valuable for improving the validity of our study, which focuses on regression testing research of large-scale software systems.

The taxonomies presented were sufficient to capture the description of challenges and proposed techniques in the included studies and the practitioners consulted in this study. However, new facets may be added by both researchers and practitioners to accommodate additional concerns or aspects of interest.

5.5.3 Accuracy of the mapping of techniques and challenges

All mappings of included papers to the various facets of the three taxonomies were reviewed by an additional reviewer. Disagreements were discussed, and the full-text of the papers was consulted to resolve them. Despite these measures, there is still a threat of misinterpretation of the papers, which could be further reduced for example by consulting the authors of the papers included in this study to validate our classification. However, due to practical reasons we did not implement this mitigation strategy.

5.6 RQ1 – Regression testing problem description

In response to RQ1, we created taxonomies of addressed context factors and desired effects investigated in the included papers.

The taxonomies created in this study follow the SERP-taxonomy architecture [3], i.e. they cover four facets, *intervention*, *context constraints*, *objective/effect* and *scope*. A SERP-taxonomy, should include one taxonomy for each facet. In our case, we create the regression testing taxonomies by extending an existing SERP-taxonomy (i.e. SERP-test [75]) by adding the details specific to regression testing. More precisely, we develop extensions for three out of four SERP facets: *context factors* (extends context in SERP-test), *desired effects* (extends objective\improvements in SERP-test) and *utilised information entities and attributes* (extends intervention in SERP). We do not extend the scope taxonomy further since regression testing is in itself a scope entity in SERP test, which all reviewed techniques target.

The taxonomy creation was done in three steps (considering both the researcher's and the practitioner's perspective on the regression testing challenge): firstly we, together with our industry partners, defined an initial set of factors and targets which were important to them; secondly we extracted information regarding these factors in the included papers and extended the taxonomies with details provided in the reports, and finally we evaluated the extended taxonomies in a focus group meeting with our industry partners to get feedback on its relevance and understandability to them in their search for applicable regression testing techniques. The items of the final taxonomies are visible in Table 5.4

At the highest abstraction level, all categories of our proposed taxonomies were considered relevant when describing a regression testing challenge (i.e. characteristics of the system, the testing process and test suite and people related factors in the context taxonomy and similarly improved coverage, efficiency, effectiveness and awareness in the effect taxonomy).

The taxonomies reported in this paper are the revised version that addresses the feedback from this focus group. Due to the dual focus when creating the taxonomies, we believe they could provide guidelines for both researchers and practitioners in defining the real-world regression testing problems they address, or wish to address consistently to support the mapping between research and practice.

5.6.1 Investigated context factors

The purpose of the context taxonomy can be summarised as: *Provide support for identifying characteristics of an industrial environment that make regression testing challenging and hence support the search for techniques appropriate for the context.*

Table 5.4 shows a taxonomy of contextual factors that were investigated in the included papers, as well as considered relevant by our industry partners. To be classified as an investigated context factor the mere mentioning of it in general terms was not considered sufficient, only in cases where the authors of the study include a discussion or explanation of the effect a factor has on regression testing and why it is considered in their study we include it as an investigated context factor.

Since we only include factors that have been discussed in the literature, the context taxonomy is not extensive but can still be used as a guideline for describing regression testing problems and solutions. We identified three main categories of relevant contextual factors (*system related, process related*, and *people related*) that have been addressed in the included papers.

System related context factors

System related context factors include factors regarding the system (or subsystem) under test, such as *size, complexity* and *type*. How size and complexity are measured varies in the studies, but a common measure of size is lines of code. Factors that are reported to add to the complexity are *heterogeneity* and *variability* (e.g. in software product lines). Some techniques are designed to address the specific challenges of applying regression testing to a certain type of systems (e.g. *web-based systems, real-time systems, embedded systems, databases* or *component-based systems*).

In the focus group meeting, *embedded systems* as a type of system were considered to be a relevant factor, characterising the regression testing challenges, but the other suggested system types were not - mainly on account of them not being applicable to the specific situation of the practitioners in the group. We interpret that the abstraction level is relevant and choose to keep the system types in the context taxonomy only where an explanation of what makes the context challenging from a regression testing perspective is given in any of the included studies (i.e. system types that are mentioned but not explained from a regression testing challenge perspective are removed from the

Table 5.4: A taxonomy of context, effect and information factors addressed in the included papers and considered relevant by our industry partners

Context taxonomy		Factors along Study ID
Investigated context factors	System-related factors	**Size** e.g. Large-scale S1, S2, S5 – S35 **Complexity** e.g. Heterogeneneous S1, S6, S9 – S12, S19, S20, S26, S29, S30, S31, S35 or Customizeable or using product-line approach S3, S4, S6, S13 – S18, S24 – S26, S35, S38 **Type of the system** e.g. Web-based/SOA S3, S4, S6, S30, S31, S36 Real time S3, S4, S7,S8, S13 – S18, S27 Embedded S1, S24 – S27 Database applications S19, S20, S36 Component-based S6, S9,S10,S11,S12, S31
	Process-related	**Testing process** (e.g. Continuous S1, S3, S4, S26) **Test technique** e.g. Manual testing S5, S33, Combinatorial S19, S20 Scenario-based testing S6 or Model-based testing S35
	People-related factors	**Cognitive factors** e.g. lack of experience S13 – S18, S26 or that new tools need to be easy to implement and use S22 **Organizational factors** (e.g. Distributed development S6)

Effect taxonomy		References
Desired effects	Test Coverage	**Feature-coverage** S3,S4, S13 – S18 **Input (Pairwise)** S19,S20
	Efficiency and effectiveness	**Reduction of test suite** S5 – S18, S23 – S25, S27 S28, S30 – S32, S35 – S37 **Reduced testing time** S1, S3, S4, S5, S7, S8, S13 – S18, S23, S28, S29, S30, S32, S33, S36 **Improved precision** S1, S7 – S12, S24, S25 **Decreased time for fault detection** S2 – S4, S21, S22, S26, S29, S37 **Reduced need for resources** S2, S13 – S18, S29, S30 **Fault detection capability** S7, S8, S13 – S21 – S4, S24 – S26, S28, S29, S34 **Severe fault detection** S3, S4, S21 **Reduced cost of failure** S9 – S12, S19, S20, S33
	Awareness	**Transparency of testing decisions** S26

Information taxonomy		References
Utilised information entities and attributes	Requirements	**No. of changes in a requirement**, **Fault impact**, **Subjective implementation complexity**, **Perceived completeness**, **Perceived traceability** S21 **Customer assigned priority** S21, S33
	Design artefacts	**System models** S13 – S18, S27, S35 **Code dependencies** S19,S20S37
	Source code	**Code changes/ Revision history** S1, S2, S5, S7,S8, S24, S25, S38 **Source file** S2, S7, S8, S30, S37 **No. of Contributors** S32
	Intermediate code	**Class dependencies** S6 **Code changes (method or class)** S2, S6, S28
	Binary code	**Revision history** S6, S29 **Component changes** S9 – S12, S31 **Binary files** S6, S9 – S12, S23, S29
	Test cases	**Target variant** S26 **Type of test** S26 **Model coverage** S13 – S20 **Functionality coverage** S3, S4**Static priority** S26 Age S26 **Fault detection probability (estimated)** S22, S29, S33 **Execution time (estimated)** S22, S29 **Cost (estimated)** S22, S33 **Link to requirements** S21, S22 **Link to faults** S21 – S4 **Link to source code** S6 – S8
	Test Execution	**Execution time** S29, S32 **Database-states** S36 **Invocation counts** S28 **Invocation chains** S28, S31 **Runtime component coverage** S31 **Method coverage** S28 **Code coverage** S5, S23, S29, S37 [67] S38 **Browser states**S36 **Class coverage** S6
	Test reports	**Execution time** S4, S13 – S18, S3, S28 **Verdicts** S1 – S4, S13 – S18, S26, S32, S34 **Severity** S28, S33 **Link to packages and their revisions** S1 **Link to branch** S32 **Build type** S32 **Link to failure** S13 – S18 **Test session** S13 – S18, S26 **Variant under test** S32
	Issues	**Link to fixed file / link to source code** S24, S25 **Fix-time** S32 **Link to test case** S24, S25, S37 **Failure severity** S3, S4

taxonomy). A similar approach was used for the other system related factors of which only one, *variability*, was considered very important by the practitioners.

Process related context factors

Process related context factors include factors of the development or testing process that may affect the relevance of a regression testing technique, such as currently used *processes* and *testing techniques*. Some regression testing techniques address new regression testing challenges arising with highly iterative development strategies such as continuous integration (which also was the first and main challenge identified by our collaborators and a starting point for this literature review). How testing is designed and carried out (e.g. *manual, black-box, combinatorial* or *model based*) may also be crucial for which regression testing technique is relevant and effective.

Of the testing process characteristics, *use of a specific tool* was considered irrelevant while the *use of testing technique* (all three examples) was considered very important. Thus, we removed the testing tool as a context characteristic and kept the examples of testing techniques, *manual testing*, and *combinatorial testing*. Black box testing was removed as it is covered by the information taxonomy. From the literature, we added two more examples of test techniques that affect the applicability of regression testing solutions, *Scenario based testing* and *Model based testing*. *The frequency of full regression test* (i.e. how often is the complete regression test suite run) was considered important, and we rephrased it to *continuous testing* in the final taxonomy. Also, *size* and *long execution times of test suites* were considered important but since it is covered by the desired effects, we removed it from the context taxonomy.

People related context factors

People related context factors refer to factors that may cause, or are caused by, distances between collaborating parties and stakeholders in the development and testing of the software system. The terminology used stems from [82]. *Cognitive* context factors include the degree of knowledge and awareness, while *organisational* factors include factors that may cause, or are caused by, differences in goals and priorities between units.

People related issues were important to all participants in the focus group, but the message about which factors were most important was mixed. *Ease of use* got the highest score. A new node *Cultural distances* was proposed as well, however, we have not found any such considerations in the selected set of papers, and thus did not include it in the taxonomy. This branch of the taxonomy showed to have overlaps with the effect taxonomy (e.g. *Lack of awareness* and *Need for quick feedback*), and we decided to remove such nodes from the context taxonomy and add them to the effect taxonomy instead.

General reflection

A reflection about the overall context taxonomy is that it is not obvious which characteristics are relevant to report from a generalisation perspective. Even in industrial studies, the problem descriptions are in many cases superficial and many context factors are mentioned without any further explanation as to why they are relevant from a regression testing perspective. Some factors mentioned are crucial only to the technology being evaluated, and not necessarily an obstacle preventing the use of other technologies. One such example is the type of programming language - it was initially added to the taxonomy, as it is a commonly reported aspect of the cases used for empirical evaluation. However, it was finally removed as it was considered a part of the constraints of a solution, rather than characterising trait of the addressed problem context.

5.6.2 Desired effects

The desired effect of a technique is basically about the reported types of improvement(s) achieved by applying the technique, such as 'improving efficiency' or 'decreasing execution time'. To be recognised as a desired effect, in our setting, the effect of the technique has to be evaluated in at least one (industrial/large scale) case study, rather than just being mentioned as a target of the technique without any evidence. Accordingly, the effect has to be demonstrated as a *measurement* showing the effect of the proposed technique on regression testing.

Table 5.4 shows a taxonomy of effect (-target) factors. The proposed effect taxonomy provides a categorisation of the various effects (improvements) identified in the research while simultaneously, it meets the level of information (or detail) required (or considered relevant) by our industrial partners. The improvements (effects) of techniques are categorised into three main categories: *Improved test coverage*, *Improved efficiency and effectiveness* and *increased awareness*.

Improved test coverage

Improved coverage refers to the effects aiming at improving (increasing) the coverage of any type of *entity* by the selected test suite. The type of entity, which is under consideration, depends on the context and the proposed solution. We identified two main desired effects for coverage in the included papers, namely *increased feature coverage* and *improved combinatorial-input coverage (Pairwise)*.

Improved efficiency and effectiveness

Efficiency and effectiveness cover cost reduction factors such as *reduced number of test cases* and *reduced execution time* with a consideration for how comprehensively faults are detected. In principle, efficiency does not look into how well a testing technique reveals or finds errors and faults. Improving only the efficiency of a technique will lead to a testing activity that requires less amount of time or computational resources, but it may not be effective (i.e. comprehensively detect faults). Efficiency and Effectiveness are often distinguished in the research literature, while in practice they are equally important objectives and are most often targeted at the same time. Thus, we treat them as one class in our taxonomy. *Reduction of test suite* often leads to a set of test cases requiring less resource (memory) and less amount of time to be generated, executed, and analysed. Note that test suite reduction in the research literature is often referred to as a technique as such ([17]). It is then used interchangeably with test suite maintenance referring to the permanent removal of test cases in contrast to the temporary removal or ordering of test cases in “test case selection” or “test case prioritisation”. However, “reduction of the number of test cases” is at the same time the most common measure of the effectiveness of a regression test selection technique in industrial evaluations. It is used in evaluation of regression testing techniques when comparing with the current state of practice (both in the maintenance case and the selection case) in a particular context. Thus, we add it as a desired effect in our taxonomy.

Reduction of testing time considers any time/resource-related aspect, also referred to as ‘cost’ in some studies. *Improved precision* refers to the ability of a selection technique in avoiding non-fault revealing test cases in the selection. High precision results in a reduction of test suite while it also indicates a large proportion of fault detecting test cases. Hence, precision is considered a measure of both efficiency and effectiveness. *Decreased time for fault detection* i.e. the aim is to reduce the time it takes to identify faults, which is relevant when reflecting on the outcomes of a prioritisation technique for regression testing. *Reduced need for resources* i.e. reduces the consumption of a resource e.g. memory consumption. *Improved fault detection capability* also referred to as ‘recall’ or ‘inclusiveness’, measures how many faults are detected regardless of their severity. *Improved detection of severe faults* refers to the extent to which a technique can identify severe faults in the system. *Reduced cost of failures*, here the focus is on the consequence (measured in cost factors) of false negatives in the selection.

Increased awareness

Increased awareness refers to improvements related to the testing process (activities) per se and the people involved in the process. *Improved transparency of testing decisions* has been considered in the existing research and identified as a relevant target by our industrial partners. It aims at transparently integrating regression testing techniques into daily/normal development activities such that the stakeholders understand the working of the technique and trust the recommendations regarding the test-cases they produce.

General reflection

A general reflection regarding the effect-taxonomy is that "what is measured in research is not always what matters in practice". The taxonomy was initially based solely on the different variants of measurements used in the studies and rather fine-grained in some aspects. Different levels of code coverage are for example a popular measurement in literature but were not considered relevant by the practitioners in the focus group. All proposed coverage metrics except feature coverage were considered irrelevant by the participants. Our interpretation is *not* that code coverage is considered useless as a test design technique, but that improving code coverage is not a driver for applying regression testing (not for the practitioners and not for any of the stakeholders in the industrial evaluations). Although code coverage is not presented as a desired effect in our taxonomy, it still appears as a characteristic of a technique (information attribute) since there are some examples of techniques in our included papers that utilise measures of code coverage to propose a regression testing scope.

Regarding the variants of measurements under effectiveness and efficiency, the granularity level was considered too high and many of the nuances in the measurements were hard to interpret from a practical point of view. Only three of the low-level categories were considered relevant for at least one of the participants, 'detection of severe faults' was important for all three while 'precision' and 'test suite reduction' were important to one of the participants.

5.7 RQ2 – Regression testing solution description in terms of utilised information sources

To answer RQ2, i.e., *"how to describe a regression testing solution?"*, we considered the following choices for developing a taxonomy: 1) based on the underlying assumptions (e.g., history-based and change-based), 2) based on the techniques (e.g., firewall, fixed-cache, and model-based), or 3) based on the required information (e.g., test case execution information, code complexity, and version history).

We decided in favour of the third option, in particular, because it allows for reasoning about what information is required to implement a regression testing technique. From a practitioner's point of view the concerns regarding: a) whether a technique would work in his/her context, and b) whether it can help achieve the desired effect, are already covered with the context and the effect taxonomy. Thus, while the effect and context taxonomies enable narrowing down the choice of techniques, the aim of the information taxonomy is to support practitioners in reasoning about the technical feasibility and the estimated cost of implementing a technique in their respective unique context.

Hence, the purpose of the information taxonomy can be summarised as *to provide support in comparing regression testing solutions by pinpointing relevant differences and commonalities among regression testing techniques (i.e., the characteristics affecting the applicability of a technique)*. We consider this classification particularly useful for practitioners as it helps one identify relevant techniques in their context. For example, if a certain test organisation does not have access to source code, they can focus on techniques that do not require it.

Similarly, knowing what information is required to implement a technique, the interested reader can: 1) identify if this information is currently available in their organisation 2) investigate how to derive it from other available information sources, or 3) analyse the feasibility of collecting it. Hence, a practitioner can make an informed decision about the applicability of a technique in their context by considering the possibility and the cost of acquiring the required information.

The information taxonomy (as shown in Table 5.4) uses the structure <entity, information> to identify what information is required to use a certain technique. Thus, we coded the entities and the utilised information about their various attributes/facets used by each of the techniques. Some examples of entities, in this case, are design artefacts, requirements or source code. The respective information about the various attributes/facets of these three example entities may include dependencies between various components, the importance of a requirement to a customer or code metrics.

From the papers included in this review, the following nine entities (and different information regarding them) were used by the regression testing techniques: 1) Requirements, 2) Design artefacts, 3) Source code 4) Intermediate code, 5) Binary code, 6) Closed defect reports, 7) Test cases, 8) Test executions, and 9) Test reports.

5.7.1 Requirements

Very few regression testing techniques included in this study have used information related to requirements (such as the importance of required functionality for the customer). Only two papers explicitly make use of information regarding the requirements [51, 52]. Such information can be coupled with requirement coverage (i.e., the number of requirements exercised by a test case) to optimise regression testing with respect to the actual operational use of the SUT [32, 52].

The information about several attributes of requirements such as their priority and the complexity of their implementation are stored in requirement management systems. However, the information regarding requirement coverage may as well be stored in the test management system with respect to their corresponding test cases.

One reason for the lack of techniques utilizing requirements as input for regression testing could be that often the traceability information from requirements to source code to test cases is not maintained [31]. Furthermore, it is significantly more difficult to recreate these traceability links than, e.g., linking source code to test cases.

The following five techniques are based on the requirements and feature coverage by test-cases: RTrace [52], MPP [48, 49], TEMSA [25, 26, 29], and FTOPSIS [32].

FTOPSIS [32] uses multi-criteria decision-making as well as fuzzy logic, where both objective and subjective (expert judgement) data about requirements can be used to prioritise test cases in a regression suite. Krishnamoorthi and Mary's approach RTrace [52] expects an explicit link between test cases and requirements for their proposal. However, they do not describe how the case companies were documenting this information. They also do not suggest how such links can be generated. TEMSA [25, 26, 29] develop and use feature models and component family models, to ensure feature coverage in regression test selection of a software product line system. MPP [48, 49] uses the coverage of functionality of the system by individual test cases as a criterion to prioritise test cases.

5.7.2 Design artefacts

Wang et al. [24–29] use feature models and component feature models. These models along with an annotated classification of test cases are used for test case selection. Similar to the approach of Wang et al., Lochau et al. [50] also exploit models (in their case, delta-oriented component test models and integration test models). They also used existing documentation from the industrial partners and interviews with practitioners to develop and validate these models.

For an automotive embedded system, Vöst and Wagner [30] propose the use of system architecture (system specifications) for test case selection. System specifications and test case traces were used to create a mapping between components and test cases using them.

5.7.3 Source code

In IEEE standard for software test documentation, regression testing is defined as: “selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or components still complies with its specified requirements” [10]. Therefore, several regression testing techniques attempt to leverage available information to localise changes in a software system that can then inform the decision of which test cases to run and in which order. Some such techniques, work with source code and its version history to identify the change. Using source code has two advantages, first, several static and dynamic approaches exist to link test-cases to source code (e.g. once we have localised the change, identifying change traversing test-cases to select or prioritise is possible). The second advantage is that most commercial organisations use a configuration management system. Thus, the techniques that utilise revision history are relevant for industrial settings.

For example, FaultTracer [67], CCB [81], Fix-cache [21, 71], EFW [22, 23], and Difference-Engine [76] utilise revision history of source code for regression testing. Similarly, THEO [65] uses the number of contributors to the code base as input.

Several techniques require access to actual source code to work. Carlson et al. [80] propose the use of a clustering approach that computes and uses code metrics as one of the criteria for test case prioritisation. REPiR [37] uses information retrieval techniques to prioritise test cases that cover the changes in source code. It relies on the likelihood that similar keywords are used in source code literal and comments as in the test cases.

GWT-SRT [64] instruments source code to be able to generate traces that are used to connect test-cases and source code. This information along with control flow graphs (to isolate code changes) are used for selective regression testing in the context of web applications.

5.7.4 Intermediate and binary code

In cases when the source code is either not available, or it is not feasible to use it, there are some techniques that work on intermediate and binary code instead of source code to localise change between two versions of a software system. REPiR [37] and CMAG [63] use intermediate code to identify changes. While I-BACCI [11–14], Echelon [34], OnSpot [58] and Pasala and Bhowmick's proposed technique [45] work with binaries to localise change.

5.7.5 Issues

Some techniques utilise information typically residing in issue management systems [21, 65, 71]. Provided that an issue originates in a fault revealed by a test case, the attributes of that issue may be used to recreate a link between the said test case and the source files that were updated to fix the issue [21, 71]. Herzig et al. [65] utilise information about the closed defect reports (e.g. the time it took to fix the issue) in a cost model weighing the cost of running a test case against skipping it. The technique described by Marijan et al. [48, 49] uses the severity information from defect reports, prioritising test cases that reveal faults of high severity.

Among the included papers, no proposal presents an approach to recreate links between defect reports and mapping to related test cases. Therefore, if practitioners want to use one of the techniques that leverage fault coverage by test cases or other fault-related information (like the severity of faults etc.) they must document and maintain the links between these artefacts.

5.7.6 Test cases

Test cases refer to the specification of the tests and are static information entities (i.e., the information is documented at the design of the test and stored and maintained in a repository typically a test management system). 50% of the evaluated techniques rely on such information. What attributes of the test specifications being used vary between the different techniques, but it could be divided into traceability information and properties of the test case per se.

Traceability information is typically used for coverage optimisation selection [11, 22, 26, 32, 35, 42, 49, 52, 80, 81]; e.g. links to other information entities such as source code and requirements, or explicit coverage targets such as model coverage [26, 42] or functionality coverage [49].

Three techniques utilise the property attributes of the test cases (e.g age and estimated cost) solely for test prioritisation [34, 51, 73] and hence they are not dependent on static traceability links. Two are risk-based [51, 73] while one recreates traceability links dynamically [34], see Section 5.7.7.

5.7.7 Test executions

Test executions refer to an implicit information entity, meaning that its information attributes may be dynamically collected but are not stored and maintained for other purposes. Just as for the 'Test cases' described above the attributes of the 'Test executions' could be divided into coverage information (e.g. 'invocation chains' [45, 63], 'covered system states' [77], 'runtime component coverage' [45] and 'code coverage' [34, 35, 58, 63, 67, 80, 81]) and intrinsic properties of the executions (e.g. 'execution time' [34, 65], or 'invocation counts' [63])

Dynamically collected coverage information is used for similarity-based and coverage-based optimisation of regression tests [77, 80, 81] as well as change-based prediction of regression faults [34, 58, 67] while dynamically collected property attributes of the test executions are typically used for history-based cost optimisation of regression tests faults [63, 65].

5.7.8 Test reports

Test reports refer to the static records of the test executions, this information could either be automatically captured or entered manually by the testers. Such attributes are used for history-based optimisation of regression tests and most commonly used for regression test optimisation are verdicts [26, 49, 65, 73, 76, 84], time stamps [26, 49, 63] and links to the tested system configuration [65, 76]. Several information attributes of the test reports are similar to the test execution attribute or the test case attribute, but differ in how it is derived and maintained. As an example, test execution time could be an attribute of all three test information entities but as an attribute of a test case it is an estimation; as an attribute of a test execution, it is measured at runtime; and as an attribute of the test report, it is further recorded and maintained.

5.8 RQ3 – Mapping of current research

26 different techniques (reported in 38 papers) were classified under three taxonomies: context, effect and information (see Table 5.4). This mapping (see Table 5.5) helps to select techniques that address relevant context factors and deliver the target benefits for a given scope (regression test selection, prioritization or minimization). The information taxonomy helps to reason about whether the information is available or can be reasonably acquired in the unique context of a particular company. We demonstrate the use of this taxonomy in Section 5.9 in the form of technological rules [33] derived from this mapping (see Section 5.9.

5.8.1 Addressed context factors

In terms of system-related factors, when describing the context where the proposed techniques are applicable, 22 of the included techniques consider the scalability of

Table 5.5: Mapping of techniques to the taxonomy

Technique	Study ID	Scope			Addressed context factors			Desired effects			Utilised information (entities)								
		Selection	Prioritization	Minimization	System-related	Process-related	People-related	Test coverage	Efficiency and Effectiveness	Awareness	Requirements	Design artefacts	Source code	Intermediate code	Binary code	Test cases	Test execution	Test reports	Issues
TEMSA	S13 – S18	✓	✓	✓	✓		✓	✓	✓			✓				✓		✓	
History based prioritization (HPro)	S26	✓	✓		✓	✓	✓		✓	✓						✓		✓	
classification tree testing (DART)	S19, S20	✓			✓	✓		✓	✓			✓				✓			
I-BACCI	S9 – S12	✓			✓				✓						✓				
Value_based	S33		✓		✓	✓			✓		✓					✓		✓	
multi-perspective prioritisation (MPP)	S3, S4		✓		✓	✓		✓	✓							✓		✓	✓
RTrace	S21		✓		✓				✓		✓					✓			
Echelon	S29		✓		✓				✓						✓	✓	✓		
Information retrieval (REPiR)	S2		✓		✓				✓				✓	✓					
EFW	S7, S8	✓		✓	✓				✓				✓			✓			
Fix-cache	S24, S25	✓			✓				✓				✓						✓
Most Frequent Failures	S34	✓			✓				✓									✓	
Continuous Multi-scale Additional Greedy prioritisation (CMAG)	S28	✓	✓		✓				✓					✓			✓	✓	
GWT-SRT	S30	✓			✓				✓				✓						
Clustering (based on coverage, fault history and code metrics)	S37	✓	✓						✓			✓	✓				✓		✓
FTOPSIS	S22		✓		✓		✓		✓							✓			
Difference engine	S1	✓			✓	✓			✓				✓					✓	
Change and coverage based (CCB)	S5	✓			✓	✓			✓				✓				✓		
Similarity based minimisation	S36			✓	✓				✓								✓		
THEO	S32	✓			✓				✓				✓				✓	✓	✓
DynamicOverlay / OnSpot	S23	✓			✓				✓						✓		✓		
Class firewall	S6	✓			✓	✓	✓		✓					✓	✓	✓	✓		
model-based architectural regression testing	S35	✓			✓	✓			✓			✓							
component interaction graph test case selection	S31	✓			✓				✓						✓		✓		
keyword-based-traces	S27	✓			✓				✓			✓							
FaultTracer	S38	✓			✓								✓				✓		

techniques for large-scale software systems. Another 13 techniques take the complexity and the type of system under test into account. 9 techniques consider process related context factors. While only 4 techniques consider people-related factors.

5.8.2 Desired effects

Most techniques (25 out of the total 26) target improved effectiveness and efficiency in terms of finding known faults. Three techniques focus on improving coverage (which is considered a proxy for achieving confidence in the testing or confirming the quality of a system). Only one technique targets increased awareness in the decision-making process in the scope of regression testing.

5.8.3 Information sources

Except regression testing techniques that rely solely on the history of test-cases, most techniques use information beyond the test cases to select, prioritise or minimise the regression testing suite. Such approaches rely on documented/generated links between test cases and other artefacts. The 26 techniques included in this study utilise information contained in nine different types of software engineering artefacts.

Only two and five techniques use information related to requirements and design artefacts, respectively. Attributes of source code are used in nine techniques while seven techniques only rely on intermediate or binary code among other information sources. Ten techniques use information about test cases. Moreover, ten and eight techniques use information from test executions and test reports. Only four techniques make use of the issue reports.

5.9 Suggestions for practitioners

Practitioners may utilise the results of this study in different ways. The mappings of techniques to each category may guide the practitioner looking for relevant research. The taxonomies could also be used to compare a set of possible solutions found in the research, or those designed by engineers at a company.

In Table 5.4, three taxonomies for describing regression testing problems and techniques were introduced. In Table 5.5, we present a mapping of the included papers to the three taxonomies. A practitioner can use the taxonomies and the mapping to identify recommendations that are likely to help them design a solution in their unique context. The mapping of techniques to the three taxonomies may be read as technological rules [33] i.e., “To achieve <**effect**> in <**context**> apply <**technique**>”, which in turn should be interpreted as recommendations (or advise) extracted from research.

Such rules could be formulated in detail for a single technique (i.e. one row in the mapping, as in example TR1) or with fewer details for a set of techniques (by including only the common denominators for that set, TR2) or in more general terms by selecting nodes at a higher level in the taxonomy (TR3). Three illustrative examples (TR1-3) are given:

TR 1: To reduce the regression test suite and testing time when regression testing large scale software systems utilise the following information attributes: #contributors of a piece of code, measured execution time, verdict, build type, variant under test and link to tested branch from test reports and fix time in issue reports. This example was extracted from an evaluation of a tool called THEO ([65]).

TR 2: To increase feature coverage, reduce testing time and improve fault detection capability when regression testing customisable, real-time systems, utilise information about verdicts and execution time in test reports. (This rule is based on the intersection of the classification of two different techniques, Multi-perspective prioritisation [49] and TEMSA [28], which have been evaluated in several studies [24–29, 48, 49].)

TR 3: To improve efficiency and effectiveness when regression testing large scale complex systems, utilise information attributes of the test reports. (This rule is a generalisation of TR2 and is supported by the same studies and another two [73, 76].)

From the research communication perspective, we argue that formulating such rules (albeit by compromising some details) will help to communicate our research in particular to industry practitioners.

By selecting the most important aspects of the two problem-defining taxonomies (i.e. desired effects and addressed context factors), for the organisation, one or more research-based recommendations (in terms of technological rules) may be extracted from the mapping in this study together with pointers to the matching literature. These recommendations could then be used as input to the design of the specific solution for the organisation.

5.10 Recommendations for researchers

As for testing, in general, the value of a regression testing technique could be measured either in terms of its ability to increase confidence in testing or in terms of its ability to improve fault detection with limited time and resources. Those high-level goals are shared by researchers and practitioners but with some variations when it comes to details and priorities [88]. The recommendations given here are based on our comparison of what we found in the literature and the feedback from our industry partners.

Evaluate coverage of regression testing techniques at the feature level Confidence is achieved by measuring any type of coverage. However, of the facets of our effect taxonomy, coverage was the least important for the practitioners and at the detailed level of our taxonomy only feature coverage was considered relevant. This is also reflected in the literature [26, 37, 49]. Few techniques were evaluated with respect to coverage and of the few the majority focused on feature coverage.

Focus evaluation on the detection of severe faults and reduction of cost From the practitioners' perspective, the ability of a technique to detect severe faults and to reduce cost in terms of man- and machine-time was considered more important. While confidence, and coverage, always being priorities in the design of new tests, the pursuit of trust in the regression test suite is often the root cause of increasing costs and decreasing the effectiveness of regression testing - as more and more test cases are added just in case [74]. Hence, the main driver for most industrial studies on regression testing is cost reduction and precision of regression testing. 70% of the techniques in our study were evaluated with respect to cost reduction. Furthermore, effectiveness should be considered in terms of the number of severe faults, rather than in the number of faults in general as there is also a cost-benefit trade-off in the issue backlog. Only 40% of the techniques in our study were evaluated with respect to fault detection and of those only two considered severity [49, 52].

Report addressed context factors in industrial evaluations To support generalisation of results between industrial contexts, relevant contextual factors need to be identified and clearly reported. Here relevant context factors denote context factors that are either causing the problems to be solved or affecting the applicability or effect of the solution. Such relevance may be observed or assumed by the stakeholder or the researcher.

Despite being a selection of industrial evaluations, reporting the context factors seems to be of low priority. In 10% of the evaluations, we did not find any context descriptions at all. For the remaining 90% at least system factors such as size, complexity and type of system under test are reported. Only 30% described the current process, which will affect (or be affected by) the adoption of the technique and only 15% reported people-related factors.

Rather than providing a general and extensive description of the case context, as proposed in previous research [44] we recommend a careful selection of context factors to report and discuss. Such selection could be guided by the context-taxonomy provided in Table 5.4.

Study if and how the proposed context factors are relevant In most cases, the relevance of the context factors described in the included papers is assumed rather than observed. Furthermore, they are described on a rather high abstraction level. Thus, there is room for more research on the relevance of various context factors for regression testing.

Study people-related factors As a research area struggling to gain traction in industry [55, 56, 83] we believe that there is a need to investigate people-related factors. The need for transparency and understandability that leads to trust in the results of regression testing techniques was highlighted by the industry partners. Only one study in our included set has investigated these factors [73]. Future research in this direction that takes into account the needs and concerns of potential users may increase the likelihood of successful adoption of regression testing techniques in the industry.

5.11 Conclusion

In this paper, we report an extensive search for and an in-depth analysis of applicability and relevance of regression testing techniques reported in the literature. We focused on techniques that have been applied to large-scale software systems in industrial contexts. Based on the literature review and in collaboration with our industrial partners, we propose three taxonomies that enable practitioners and researchers to assess and compare the relevance and applicability of regression testing techniques for a given industrial context.

The taxonomies extend three out of the four facets of the SERP-test taxonomy [75]: i.e. addressed context factors, desired improvements and characteristics of the techniques from an applicability point of view (required information and underlying assumptions). It allows for characterisation of regression techniques and helps to determine which of these techniques could be suitable in a given context and to indicate what benefits could be expected from its application. The identification of information needs for these techniques further assists a reader to reason about the implications regarding the cost of adoption [87]. In this report, we apply the taxonomies on the 38 papers that are included in the review. However, initially, we identified more than 1000 papers on regression testing and there are many techniques, not included in the review, that may still be relevant and applicable in some industrial contexts. The aim of the taxonomies is to support assessment also of them and of new proposals or adaptations of techniques.

In our interaction with the practitioners, we identified some discrepancies in researcher's focus and the perceived needs in the industry. One such example is the commonly used measure of effectiveness in terms of various levels of code coverage. Some types of information (such as coverage information) that are extensively studied in the literature were found to be only partially relevant to the practitioners (e.g., only at the feature level) for evaluating the benefits of a regression testing technique. At the same time, some extremely relevant information in choosing technique (e.g. the context information) is completely neglected in some existing studies.

For academia, this study can help to use evaluation criteria and contexts that are representative of industrial needs. This work also supports reporting the research results to facilitate readers making an informed decision with a consideration for relevance to their context, potential benefits and the likely investment required to use the technique.

5.12 References

[1] H. Munir, M. Moayyed, and K. Petersen, "Considering rigor and relevance when evaluating test driven development: A systematic review," *Information and Software Technology*, vol. 56, no. 4, pp. 375–394, Apr. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0950584914000135

[2] M. V. Zelkowitz, D. R. Wallace, and D. Binkley, "Culture conflicts in software engineering technology transfer," in *NASA Goddard Software Engineering Workshop*. Citeseer, 1998, p. 52.

[3] K. Petersen and E. Engström, "Finding relevant research solutions for pracitical problems - the SERP taxonomy architecture," in *International Workshop on Long-term Industrial Collaboration on Software Engineering (WISE 2014).*

[4] M. Thelwall and K. Kousha, "ResearchGate versus Google Scholar: Which finds more early citations?" *Scientometrics*, vol. 112, no. 2, pp. 1125–1131, Aug. 2017. [Online]. Available: https://link.springer.com/article/10.1007/s11192-017-2400-4

[5] A. Rainer, T. Hall, and N. Baddoo, "A preliminary empirical investigation of the use of evidence based software engineering by under-graduate students." in *Proceedings of the 10th International Conference on Evaluation and Assessment in Software Engineering*. [Online]. Available: https://uhra.herts.ac.uk/dspace/handle/2299/2270

[6] A. Rainer, D. Jagielska, and T. Hall, "Software engineering practice versus evidence-based software engineering research," in *Proceedings of the ACM Workshop on Realising evidence-based software engineering (REBSE '05)*, pp. 1–5. [Online]. Available: http://doi.acm.org/10.1145/1082983.1083177

[7] A. Rainer and S. Beecham, "A follow-up empirical evaluation of evidence based software engineering by undergraduate students," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*.

[8] E. Engström, R. Feldt, and R. Torkar, "Indirect effects in evidential assessment: a case study on regression test technology adoption," in *2nd international workshop on Evidential assessment of software technologies*, pp. 15–20.

[9] H. Edison, N. B. Ali, and R. Torkar, "Towards innovation measurement in the software industry," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1390–1407, 2013.

[10] *IEEE standard for software test documentation IEEE Std. 829-1983*, IEEE Std., 1998.

[11] J. Zheng, L. Williams, and B. Robinson, “Pallino: automation to support regression test selection for cots-based applications,” in *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, ASE*, 2007, pp. 224–233.

[12] J. Zheng, B. Robinson, L. Williams, and K. Smiley, “Applying regression test selection for cots-based applications,” in *Proceedings of the 28th International Conference on Software Engineering, ICSE*, 2006, pp. 512–522.

[13] ——, “A lightweight process for change identification and regression test selection in using COTS components,” in *Proceedings of the 5th International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, ICCBSS*, 2006, pp. 137–143.

[14] J. Zheng, “In regression testing selection when source code is not available,” in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE*, 2005, pp. 452–455.

[15] H. Zhang, M. A. Babar, and P. Tell, “Identifying relevant studies in software engineering,” *Information & Software Technology*, vol. 53, no. 6, pp. 625–637, 2011.

[16] A. Zarrad, “A systematic review on regression testing for web-based applications,” *JSW*, vol. 10, no. 8, pp. 971–990, 2015.

[17] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” *Softw. Test., Verif. Reliab.*, vol. 22, no. 2, pp. 67–120, 2012.

[18] Z. Xu, Y. Kim, M. Kim, M. B. Cohen, and G. Rothermel, “Directed test suite augmentation: an empirical investigation,” *Softw. Test., Verif. Reliab.*, vol. 25, no. 2, pp. 77–114, 2015.

[19] C. Wohlin, “Empirical software engineering research with industry: top 10 challenges,” in *Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry, CESI*, 2013, pp. 43–46.

[20] ——, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE*, 2014, pp. 38:1–38:10.

[21] G. Wikstrand, R. Feldt, J. K. Gorantla, W. Zhe, and C. White, “Dynamic regression test selection based on a file cache an industrial evaluation,” in *Proceedings of the International Conference on Software Testing Verification and Validation, ICST*. IEEE, 2009, pp. 299–302.

[22] L. J. White and B. Robinson, “Industrial real-time regression testing and analysis using firewalls,” in *Proceedings of the 20th International Conference on Software Maintenance, ICSM*, 2004, pp. 18–27.

[23] L. J. White, K. Jaber, B. Robinson, and V. Rajlich, “Extended firewall for regression testing: an experience report,” *Journal of Software Maintenance*, vol. 20, no. 6, pp. 419–433, 2008.

[24] S. Wang, A. Gotlieb, S. Ali, and M. Liaaen, “Automated test case selection using feature model: an industrial case study,” in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2013, pp. 237–253.

[25] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, and M. Liaaen, “Multi-objective test prioritization in software product line testing: an industrial case study,” in *Proceedings of the 18th International Software Product Line Conference, SPLC*, 2014, pp. 32–41.

[26] S. Wang, S. Ali, T. Yue, O. Bakkeli, and M. Liaaen, “Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search,” in *Proceedings of IEEE/ACM 38th International Conference on Software Engineering Companion, ICSE-C*, 2016-05, pp. 182–191.

[27] S. Wang, S. Ali, A. Gotlieb, and M. Liaaen, “Automated product line test case selection: industrial case study and controlled experiment,” *Software and System Modeling*, vol. 16, no. 2, pp. 417–441, 2017.

[28] S. Wang, S. Ali, and A. Gotlieb, “Cost-effective test suite minimization in product lines using search techniques,” *Journal of Systems and Software*, vol. 103, pp. 370–391, 2015.

[29] ——, “Minimizing test suites in software product lines using weight-based genetic algorithms,” in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*, 2013, pp. 1493–1500.

[30] S. Vöst and S. Wagner, “Trace-based test selection to support continuous integration in the automotive industry,” in *Proceedings of the International Workshop on Continuous Software Evolution and Delivery, CSED*, 2016, pp. 34–40.

[31] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis, "Linking requirements and testing in practice," in *Proceedings of the 16th IEEE International Requirements Engineering, RE.* IEEE, 2008, pp. 265–270.

[32] S. Tahvili, W. Afzal, M. Saadatmand, M. Bohlin, D. Sundmark, and S. Larsson, "Towards earlier fault detection by value-driven prioritization of test cases using fuzzy TOPSIS," in *Information Technology: New Generations*, S. Latifi, Ed. Springer International Publishing, 2016, pp. 745–759.

[33] M. D. Storey, E. Engström, M. Höst, P. Runeson, and E. Bjarnason, "Using a visual abstract as a lens for communicating and promoting design science research in software engineering," in *Proceedings of ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2017, pp. 181–186.

[34] A. Srivastava and J. Thiagarajan, "Effectively prioritizing tests in development environment," in *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA '02. ACM, 2002, pp. 97–106.

[35] ——, "A case study of the class firewall regression test selection technique on a large scale distributed software system," in *Proceedings of the International Symposium on Empirical Software Engineering, ISESE*, 2005, pp. 74–83.

[36] Y. Singh, A. Kaur, B. Suri, and S. Singhal, "Systematic literature review on regression test prioritization techniques," *Informatica (Slovenia)*, vol. 36, no. 4, pp. 379–408, 2012.

[37] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry, "An information retrieval approach for regression test prioritization based on program changes," in *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering, ICSE*, 2015, pp. 268–279.

[38] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Trans. Software Eng.*, vol. 22, no. 8, pp. 529–551, 1996.

[39] R. H. Rosero, O. S. Gómez, and G. D. R. Rafael, "15 years of software regression testing techniques - A survey," *Int. J. Software Eng. Knowl. Eng.*, vol. 26, no. 5, pp. 675–690, 2016.

[40] H. D. Rombach, M. Ciolkowski, D. R. Jeffery, O. Laitenberger, F. E. McGarry, and F. Shull, "Impact of research on practice in the field of inspections, reviews and walkthroughs: learning from successful industrial uses," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 6, pp. 26–35, 2008.

[41] E. Rogstad, L. C. Briand, and R. Torkar, “Test case selection for black-box regression testing of database applications,” *Information & Software Technology*, vol. 55, no. 10, pp. 1781–1795, 2013.

[42] E. Rogstad and L. C. Briand, “Cost-effective strategies for the regression testing of database applications: Case study and lessons learned,” *Journal of Systems and Software*, vol. 113, pp. 257–274, 2016.

[43] D. Qiu, B. Li, S. Ji, and H. K. N. Leung, “Regression testing of web service: A systematic mapping study,” *ACM Comput. Surv.*, vol. 47, no. 2, pp. 21:1–21:46, 2014.

[44] K. Petersen and C. Wohlin, “Context in industrial software engineering research,” in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 401–404.

[45] A. Pasala and A. Bhowmick, “An approach for test suite selection to validate applications on deployment of COTS upgrades,” in *Proceedings of the 12th Asia-Pacific Software Engineering Conference, APSEC*, 2005, pp. 401–407.

[46] L. J. Osterweil, C. Ghezzi, J. Kramer, and A. L. Wolf, “Determining the impact of software engineering research on practice,” *IEEE Computer*, vol. 41, no. 3, pp. 39–49, 2008.

[47] E. N. Narciso, M. E. Delamaro, and F. de Lourdes dos Santos Nunes, “Test case selection: A systematic literature review,” *Int. J. Software Eng. Knowl. Eng.*, vol. 24, no. 4, pp. 653–676, 2014.

[48] D. Marijan, A. Gotlieb, and S. Sen, “Test case prioritization for continuous regression testing: An industrial case study,” in *Proceedings of IEEE International Conference on Software Maintenance*, 2013, pp. 540–543.

[49] D. Marijan, “Multi-perspective regression test prioritization for time-constrained environments,” in *Proceedings of IEEE International Conference on Software Quality, Reliability and Security, QRS*, 2015, pp. 157–162.

[50] M. Lochau, S. Lity, R. Lachmann, I. Schaefer, and U. Goltz, “Delta-oriented model-based integration testing of large-scale systems,” *Journal of Systems and Software*, vol. 91, pp. 63–84, 2014.

[51] Q. Li and B. Boehm, "Improving scenario testing process by adding value-based prioritization: an industrial case study," in *Proceedings of the International Conference on Software and System Process*. ACM, 2013, pp. 78–87.

[52] R. Krishnamoorthi and S. A. S. A. Mary, "Factor oriented requirement coverage based system test case prioritization of new and regression test cases," *Information & Software Technology*, vol. 51, no. 4, pp. 799–808, 2009.

[53] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*. Chapman & Hall/CRC, 2015.

[54] R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, "Effective regression test case selection: A systematic literature review," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 29:1–29:32, 2017.

[55] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation, ICST*, 2011, pp. 99–102.

[56] N. J. Juzgado, A. M. Moreno, and S. Vegas, "Reviewing 25 years of testing technique experiments," *Empirical Software Engineering*, vol. 9, no. 1-2, pp. 7–44, 2004.

[57] S. T. R. Jr. and W. E. Riddle, "Software technology maturation," in *Proceedings of the 8th International Conference on Software Engineering*, 1985, pp. 189–200.

[58] M. U. Janjua, "Onspot system: test impact visibility during code edits in real software," in *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE*, 2015, pp. 994–997.

[59] D. Lo, N. Nagappan, and T. Zimmermann, "How practitioners perceive the relevance of software engineering research," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*. Bergamo, Italy: ACM Press, 2015, pp. 415–425. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2786805.2786809

[60] X. Franch, D. M. Fernandez, M. Oriol, A. Vogelsang, R. Heldal, E. Knauss, G. H. Travassos, J. C. Carver, O. Dieste, and T. Zimmermann, "How do Practitioners Perceive the Relevance of Requirements Engineering Research? An Ongoing Study," in *2017 IEEE 25th International Requirements*

Engineering Conference (RE). Lisbon, Portugal: IEEE, Sep. 2017, pp. 382–387. [Online]. Available: http://ieeexplore.ieee.org/document/8049144/

[61] J. C. Carver, O. Dieste, N. A. Kraft, D. Lo, and T. Zimmermann, "How Practitioners Perceive the Relevance of ESEM Research," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*. Ciudad Real, Spain: ACM Press, 2016, pp. 1–10. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2961111.2962597

[62] M. Ivarsson and T. Gorschek, "A method for evaluating rigor and industrial relevance of technology evaluations," *Empirical Software Engineering*, vol. 16, no. 3, pp. 365–395, 2011.

[63] S. Huang, Y. Chen, J. Zhu, Z. J. Li, and H. F. Tan, "An optimized change-driven regression testing selection strategy for binary java applications," in *Proceedings of ACM symposium on Applied Computing*. ACM, 2009, pp. 558–565.

[64] M. Hirzel and H. Klaeren, "Graph-walk-based selective regression testing of web applications created with google web toolkit," in *Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering (SE)*, 2016, pp. 55–69.

[65] K. Herzig, M. Greiler, J. Czerwonka, and B. Murphy, "The art of testing less without sacrificing quality," in *Proceedings of the 37th International Conference on Software Engineering*, ser. ICSE '15. IEEE Press, 2015, pp. 483–493.

[66] M. J. Harrold and A. Orso, "Retesting software during development and maintenance," in *Proceedings of Frontiers of Software Maintenance FoSM*. IEEE, 2008, pp. 99–108.

[67] M. Gligoric, S. Negara, O. Legunsen, and D. Marinov, "An empirical evaluation and comparison of manual and automated test selection," in *Proceedings of ACM/IEEE International Conference on Automated Software Engineering, ASE*, 2014, pp. 361–372.

[68] V. Garousi and M. V. Mäntylä, "A systematic literature review of literature reviews in software testing," *Information & Software Technology*, vol. 80, pp. 195–216, 2016.

[69] K. R. Felizardo, E. Mendes, M. Kalinowski, E. F. d. Souza, and N. L. Vijaykumar, "Using forward snowballing to update systematic reviews in software engineering," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2016, pp. 53:1–53:6.

[70] M. Felderer and E. Fourneret, “A systematic classification of security regression testing approaches,” *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 3, pp. 305–319, 2015.

[71] E. Engström, P. Runeson, and G. Wikstrand, “An empirical evaluation of regression testing based on fix-cache recommendations,” in *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation, ICST*, 2010, pp. 75–78.

[72] E. Engström, P. Runeson, and M. Skoglund, “A systematic review on regression test selection techniques,” *Information & Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[73] E. Engström, P. Runeson, and A. Ljung, “Improving regression testing transparency and efficiency with history-based prioritization - an industrial case study,” in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation, ICST*, 2011, pp. 367–376.

[74] E. Engström and P. Runeson, “A qualitative survey of regression testing practices,” in *Proceedings of the 11th International Conference on Product-Focused Software Process Improvement PROFES*, 2010, pp. 3–16.

[75] E. Engström, K. Petersen, N. B. Ali, and E. Bjarnason, “SERP-test: a taxonomy for supporting industry-academia communication,” *Software Quality Journal*, vol. 25, no. 4, pp. 1269–1305, 2017.

[76] E. D. Ekelund and E. Engström, “Efficient regression testing based on test history: An industrial evaluation,” in *Proceedings of IEEE International Conference on Software Maintenance and Evolution, ICSME*, 2015, pp. 449–457.

[77] P. Devaki, S. Thummalapenta, N. Singhania, and S. Sinha, “Efficient and flexible GUI test execution via test merging,” in *Proceedings of the International Symposium on Software Testing and Analysis, ISSTA*, 2013, pp. 34–44.

[78] C. Catal and D. Mishra, “Test case prioritization: a systematic mapping study,” *Software Quality Journal*, vol. 21, no. 3, pp. 445–478, 2013.

[79] C. Catal, “On the application of genetic algorithms for test case prioritization: a systematic literature review,” in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*. ACM, 2012, pp. 9–14.

[80] R. Carlson, H. Do, and A. Denton, “A clustering approach to improving test case prioritization: An industrial case study,” in *Proceedings of the 27th IEEE International Conference on Software Maintenance, ICSM*. IEEE, 2011, pp. 382–391.

[81] G. Buchgeher, C. Ernstbrunner, R. Ramler, and M. Lusser, “Towards tool-support for test case selection in manual regression testing,” in *Workshops proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation, ICST*, 2013, pp. 74–79.

[82] E. Bjarnason, K. Smolander, E. Engström, and P. Runeson, “A theory of distances in software engineering,” *Information & Software Technology*, vol. 70, pp. 204–219, 2016.

[83] A. Bertolino, “Software testing research: Achievements, challenges, dreams,” in *Proceedings of the Workshop on the Future of Software Engineering, FOSE*, 2007, pp. 85–103.

[84] J. Anderson, S. Salem, and H. Do, “Improving the effectiveness of test suite through mining historical data,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM Press, 2014, pp. 142–151.

[85] N. B. Ali, K. Petersen, and C. Wohlin, “A systematic literature review on the industrial use of software process simulation,” *Journal of Systems and Software*, vol. 97, pp. 65–85, 2014.

[86] N. B. Ali, K. Petersen, and M. Mäntylä, “Testing highly complex system of systems: an industrial case study,” in *Proceedings of ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2012, pp. 211–220.

[87] N. B. Ali, “Is effectiveness sufficient to choose an intervention?: Considering resource use in empirical software engineering,” in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2016, pp. 55:1–55:6.

[88] N. M. Minhas, K. Petersen, N. B. Ali, and K. Wnuk, “Regression testing goals-view of practitioners and researchers,” in *24th Asia-Pacific Software Engineering Conference Workshops (APSECW), 2017*. IEEE, 2017, pp. 25–31.

Chapter 6

Lessons learned from replicating a study on information-retrieval based test case prioritization

6.1 Introduction

Replications help in evaluating the results, limitations, and validity of studies in different contexts [46]. They also help establishing or expanding the boundaries of a theory [38, 46].

During the previous four decades, software engineering researchers have built new knowledge and proposed new solutions, many of these lack consolidation [37]. Replication studies can help in establishing the solutions and expanding the knowledge. Software engineering researchers have been working on replication studies since the 1990s. Still, the number of replicated studies is small, and a more neglected area is the replication of software testing experiments [33, 34, 37, 38]. Most software engineering replication studies are conducted for experiments involving human participants; few replications exist for artefact-based experiments [38]. In the artefacts-based software engineering experiments, the majority of the authors use the artefacts from the software infrastructure repository (SIR) [7]. Do et al. [22] introduced SIR in 2005 to facilitate experimentation and evaluation of testing techniques (mainly regression testing technique) and to promote replication of experiments and aggregation of findings.

Researchers have been proposing different techniques to support regression testing practice, and some of them are evaluating their techniques in an industry context. However, adopting these techniques in practice is challenging because, in most cases, the results are inaccessible for the practitioners [11]. Moreover, most regression testing techniques proposed in research have been evaluated using open-source data sets [7]. Adopting these techniques in practice is more challenging because practitioners do not know the context these techniques can fit. Replications of existing solutions for regression testing can be helpful in this regard, provided the availability of data and automation scripts for future replications.

Attempts have been made to replicate regression testing techniques. The majority of these replications are done by the same group of authors who originally proposed the techniques [39, 40, 42]. There is a need for conducting more independent replications in software engineering [22]. However, evidence of independent replications in regression testing is low [38].

Overall, we would highlight the following research gaps concerning replications:

- *Gap 1: Only a small portion of studies are replications:* Among the reasons for a lower number of replications in software engineering is the lack of standardized concepts, terminologies, and guidelines [37]. Software engineering researchers need to make an effort to replicate more studies.

- *Gap 2: Lack of replication guidelines:* There is a need to work on the guidelines and methodologies to support replicating the studies [36].

- *Gap 3: Lack of replications in specific subject areas:* Software testing as a subject area has been highlighted as an area lacking replication studies [38]. According to Da Silva et al. [38] the majority of replication studies focuses on software construction and software requirements. Despite a well-researched area, the number of replication studies in software testing is at the lowest than other software engineering research areas according to Magalhães et al. [36].

- *Gap 4: Lack of studies on artefact-based investigations:* Only a few replicated studies focused on artefact-based investigations [38]. That is, the majority of studies focused on experiments and case studies involving human subjects. Artefact-based replications are of a particular interest as they require to build and run scripts for data collection (e.g., solution implementation and logging), and at the same time compile and run the software systems, which are the subject of study.

Considering the gaps stated above, we formulate the following research goal:

Goal: *To replicate an artefact-based study in the area of software testing, with a focus on reflecting on the replication process and the ability to replicate the findings of the original study.*

To achieve our research goal, we present the results of our replication experiment in which we evaluated an IR-based test case prioritization technique proposed by Kwon et al. [26]. The authors introduced a linear regression model to prioritize the test cases targeting infrequently tested code. The inputs for the model are calculated using term frequency (TF), inverse document frequency (IDF), and code coverage information [26]. TF and IDF are the weighing scheme used with information retrieval methods [55]. The original study's authors used open-source data sets (including SIR artefacts) to evaluate the proposed technique. We attempted to evaluate the technique using four software programs to see if the replication confirms the original study's findings. We selected two software programs from the original study and two new cases to test the technique's applicability on different software programs.

Our research goal is achieved through the following:

1. *Objective 1: Studying the extent to which the technique is replicable.* Studying the extent to which the technique is replicable and documenting the detail of all steps will help draw valuable lessons. Hence, contributing with guidance for future artefact-based replications (Gap 2, Gap 4).

2. *Objective 2: Evaluating the results of the original study [26].* Evaluating the results through the replication provides an assessment of the validity and the robustness of the results of the original study. Overall, we contribute to the generally limited number of replication studies (Gap 1) in general, and replication studies focused on software testing in particular (Gap 3).

The organization of the rest of the paper is as follows: Section 6.2 provides a brief introduction to the concepts relevant to this study. Section 6.3 presents a brief discussion of some replications carried out for test case prioritization techniques. Along with the research questions and summary of the concepts used in the original study, Section 6.4 describes the methodologies we have used to select the original study and conduct the replication. Threats to the validity of the replication experiment are discussed in Section 6.4.6. Section 6.5 presents the findings of this study, Section 6.6 provides the discussion on the findings of replication study, and Section 6.7 concludes the study.

6.2 Background

This section provides a discussion on the topics related to our investigation.

6.2.1 Regression testing

Regression testing is a retest activity to ensure that system changes do not affect other parts of the system negatively and that the unchanged parts are still working as they did before a change [7, 21]. It is essential but expensive and challenging testing activity [19]. Various authors have highlighted that testing consumes 50% of the project cost, and regression testing consumes 80% of the total testing cost [4, 5, 19, 25]. Research reports that regression testing may consume more than 33% of the cumulative software cost [12]. Regression testing aims to validate that modifications have not affected the previously working code [21, 42].

Systems and Software Engineering–Vocabulary [23], defines regression testing as:

> *1. "Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements."*
> *2. "Testing required to determine that a change to a system component has not adversely affected functionality, reliability or performance and has not introduced additional defects."*

For larger systems, it is expensive to execute regression test suites in full [21]. To cope with this, one of the suggested solutions is test case prioritization. It helps to prioritize and run the critical test cases early in the regression testing process. The goal of test case prioritization is to increase the test suite's rate of fault detection [24].

A reasonable number of systematic literature reviews and mapping studies on various aspects of regression testing provides evidence that regression testing is a well-researched area [1–15]. Despite a large number of regression testing techniques proposed in the literature, the adoption of these techniques in the industry is low [16–19]. The reasons are that the results of these techniques are not accessible for practitioners due to the discrepancies in terminology between industry and academia [11, 19, 20]. There is a lack of mechanisms to guide the practitioners in translating, analyzing, and comparing the regression testing techniques. Furthermore, various authors use controlled experiments for their empirical investigations, and in most cases, it is hard to assess that these experiments are repeatable and could fit in an industrial setting [11]. Replication of empirical studies could lead us to the desired solution, as it can help to confirm the validity and adaptability of these experiments [46].

6.2.2 Replication

Replication is a means to validate experimental results and examine if the results are reproducible. It can also help to see if the results were produced by chance or if the results are the outcome of any feigned act [32]. An effectively conducted replication study helps in solidifying and extending knowledge. In principle, replication provides a way forward to create, evolve, break, and replace theoretical paradigms [37, 46]. Replication could be of two types 1) internal replication –a replication study carried out by the authors of the original study themselves, 2) external replication –a replication study carried out by researchers other than the authors of the original study [31, 37].

In software engineering research, the number of internal replications is much higher than external replications [35, 38]. Da Silva et al. [38] reported in their mapping study that out of 133 included replication studies, 55% of the studies are internal replications, 30% are external replications, and 15% are the mix of internal and external. Furthermore, the results of 82% of the internal replications are confirmatory, and the results of 26% of external replications conform to the original studies [38]. From the empirical software engineering perspective, Shull et al. [46] classify replications as exact and conceptual replication. In an exact replication, the replicators closely follow the procedures of the original experiment, whereas, in a conceptual replication, the research questions of the original study are evaluated using a different experimental setup. Concerning exact replication, if the replicators keep the conditions in the replication experiment the same as the actual experiment, it would be categorized as exact dependent replication. If replicators deliberately change the underlying conditions of the original experiment, it would be referred to as exact independent replication. Exact dependent and exact independent replications could respectively be mapped to strict and differentiated replications. A strict replication compels the researchers to replicate a prior study as precisely as possible. In contrast, in a differentiated replication, researchers could intentionally alter the aspects of a previous study to test the limits of the study's conclusions. In most cases, strict replication is used for both internal and external replications [37].

6.2.3 Information retrieval

IR-based techniques are used to retrieve the user's information needs from an unstructured document collection. The information needs are represented as queries [29, 30]. An information retrieval (IR) system is categorized by its retrieval model because its effectiveness and utility are based on the underlying retrieval model [56]. Therefore, a retrieval model is the core component of any IR system.

Amati [56] defines the information retrieval model as:

"A model of information retrieval (IR) selects and ranks the relevant documents with respect to a user's query. The texts of the documents and the queries are represented in the same way, so that document selection and ranking can be formalized by a matching function that returns a retrieval status value (RSV) for each document in the collection. Most of the IR systems represent document contents by a set of descriptors, called terms, belonging to a vocabulary V."

Some of the retrieval models are the vector space model (VSM), probabilistic relevance framework (PRF), binary independence retrieval (BIR), best match version 25 (BM 25), and language modeling (LM). VSM is among the popular models in information retrieval systems. It uses TF-IDF (term frequency and inverse document frequency) as a weighing scheme [55].

Since the technique [26] we are replicating in this study uses the concepts of TF-IDF weighing scheme, we briefly present TF and IDF.

Term frequency (TF) and inverse document frequency (IDF) are statistics that indicate the significance of each word in the document or query. TF represents how many times a word appears in the document or query. IDF is an inverse of document frequency (DF). The DF of a word indicates the number of documents in the collection containing the word. Therefore a high IDF score of any word means that the word is relatively unique and it appeared in fewer documents [30].

6.3 Related work

Most of the replication studies on test case prioritization were conducted by the same group of authors, who primarily re-validated/extended the results of their previously conducted experiments (see [39, 40, 42]). Below we discuss studies that are closely related to our topic (i.e., test case prioritization).

Do et al. [39] conducted a replication study to test the effectiveness of the test case prioritization techniques originally proposed for C programs on different Java programs using the JUnit testing framework. The authors' objective was to test whether the techniques proposed for C programs could be generalized to other programming and testing paradigms. The authors who conducted the replication study were part of the original studies, so by definition, it could be referred to as an internal replication. However, concerning the implementation perspective, the replication study would be regarded as differentiated replication.

Do and Rothermel [40] conducted an internal replication study to replicate one of their studies on test case prioritization. The original study used hand-seeded faults. In the replication study, the authors conducted two experiments. In the first experiment, the authors considered mutation faults. The goal was to assess whether prioritization results obtained from hand-seeded faults differ from the results obtained by mutation faults. The authors used the same software programs and versions used in the original study. They also replicated the experimental design according to the original study. To further strengthen the findings, later in the second experiment, the authors replicated the first experiment with two additional Java programs with different types of test suites.

Ouriques et al. [43] conducted an internal replication study of their own experiment concerning the test case prioritization techniques. In the original study, the authors experimented with software programs closer to the industrial context. The objective of the replication study was to repeat the conditions evaluated in the original study but with more techniques and industrial systems as objects of study. Although the authors worked with the test case prioritization techniques, they clearly stated that the methods examined in their research use a straightforward operation of adding one test case at a time in the prioritized set. They do not use any data from the test case execution history, and hence, regression test prioritization is not in the scope of their study.

Hasnain et al. [44] conducted a replication study to investigate the regression analysis for classification in test case prioritization. The authors' objective to replicate the original study was to confirm whether or not the regression model used in the original study accurately produced the same results as the replicated study. Along with the program and data set used in the original study, the authors also used an additional open-source Java-based program to extend the original study's findings. It is an external replication study as all authors of the replication study are different from that of the original study. The authors of the replicated study validated the results of the original study on an additional data-set other than the one used in the original study, the replication is not strict.

In the above discussion of related studies, we learned that most replication studies conducted for test case prioritization are primarily internal replications. We could only find a single external replication study [44]. The authors of this study conducted the replication of a classification-based test case prioritization using regression analysis. Our study is similar to this study based on the following factors, 1) our study is an external replication, 2) we also use two software artefacts from the original study and two additional artefact. In many ways, our study is unique; for example, 1) we are replicating a technique that focuses on less tested code, whereas Husnain et al. replicated a technique that is based on fault classification and non-faulty modules, 2) we have provided a step by step guide to support future replications, and 3) we provide automated scripts to execute the complete replication study.

6.4 Methodology

For reporting the replication steps, we followed the guideline proposal provided by Carver [45]. It suggests reporting the following for a replication study:

1. Information about the original study (Section 6.4.2)
2. Information about the replication (Section 6.4.3)
3. Comparison of results to the original study (Section 6.5.2)
4. Drawing conclusions across studies (Section 6.7)

6.4.1 Research questions

In the presence of the constraint regarding experimental setup and data, we have to rely on the information presented in the original study (see Section (6.4.2). We decided not to tweak the original study's approach and followed the steps proposed by the authors and executed the technique on one of the artefacts used by the authors. The differential aspects of the replication experiment are the mutants and the automation of the major steps of the technique. According to the classification provided by Shull et al. [46], our work can be classified as *exact independent replication* of the test case prioritization technique presented in [26].

To achieve the objectives of the study we asked the following two research questions:

RQ1. *To what degree is the study replication feasible given the information provided?*

RQ1.1 To what degree is the study replicable with the software programs used in the original study?

RQ1.2 What is the possibility to replicate the study with the additional software programs?

The answer to RQ1 corresponds to Objective 1. While answering RQ1, the motive was to see the possibility to replicate the technique presented in the original study using different software programs.

RQ2. *Does the replication confirm the findings of the original study?* The answer to RQ2 corresponds to Objective 2. The motive of RQ2 was to see if the replication results conform to the finding of the original study. To ensure that there should be no conscious deviation from the basic technique, we followed the steps and used the tools mentioned in the original study. Finally, we evaluated the replication results using the average percentage of fault detection (APFD) as suggested by the original study's authors.

6.4.2 Information about the original study

Selection of target study

Selection of a target study for replication is a difficult process, and often it is prone to biases due to various reasons [57]. For example, clinical psychology research reports that authors tend to choose targets that are easy to set up and execute [57]. The selection of target must be purpose-based, either by following systematic criteria (see, e.g., [57]) or other justifiable reasons. In our case, the selection of the target is based on the needs identified from our interaction with industry partners [11, 20, 21] and reported facts in the related literature [7, 9].

For the selection of the target study, our first constraint was test case prioritization, whereas the underlying criteria were to search for a technique that can help control fault slippage and increase the fault detection rate. During our investigations [21], we identified that test case prioritization is among the primary challenges for practitioners, and they are interested in finding techniques that can overcome their challenges and help them follow their goals (see also [20]). Increasing the test suite's rate of fault detection is a common goal of regression test prioritization techniques [14, 58], whereas controlling fault slip through is among the goals of the practitioners [20, 21].

Our next constraint was selecting a study where authors used the SIR system to evaluate their technique(s). [9] reported that out of 65 papers selected for their systematic reviews on regression test prioritization 50% are using SIR systems. [7] also reported that most of the authors evaluate their techniques using SIR artefacts. They highlight that use of SIR systems allows replication studies.

The final constraint was to select a target study that uses IR methods for the prioritization technique. Recent studies report that test case prioritization techniques based on IR concepts could perform better than the traditional coverage-based regression test prioritization techniques [27, 28].

We searched Google Scholar with the keywords *"regression testing", "test case prioritization", "information retrieval (IR)", "software infrastructure repository (SIR)"*. Our searches returned 340 papers. After scanning the abstracts, we learned that there is not a single technique that explicitly states controlling fault slippage as its goal. However, the technique presented in [26] focused on less tested code, and the goal was to increase the fault detection rate of coverage-based techniques using IR methods. Ignored or less tested code could be among the causes of fault slippage. Therefore we considered the technique by Kwon et al. [26] for further evaluation. We evaluated this technique using the rigor criteria as suggested by Ivarsson et al. [59]. The authors suggest evaluating the rigor of empirical studies based on context, design, and validity threats. After considering all the factors mentioned above and applying the rigor rubrics, the study presented in [26] was used as a target for replication.

Describing the original study

Kwon et al. [26] intended to improve the effectiveness of test case prioritization by focusing on infrequently tested code. They argued that test case prioritization techniques based on code coverage might lack fault detection capability. They suggested that using the IR-based technique could help overcome the limitation of coverage-based test case prioritization techniques. Considering the frequency at which code elements have been tested, the technique uses a linear regression model to determine the fault detection capability of the test cases. Similarity score and code coverage information of test cases are used as input for the linear regression model. Kwon et al. [26] stated that the technique they proposed is the first of its type that considers less tested code and uses TF-IDF in IR for test case prioritization. The authors claimed that their approach is also first in using linear regression to weigh the significance of each feature regarding fault-finding. They divided the process into three phases, i.e., validation, training, and testing, and suggested using the previous fault detection history or mutation faults as validation and training data.

Kwon et al. [26] suggested the following steps to implement the proposed technique:

1. *Coverage of each test case*
2. *Set IDF threshold with validation data (previous or mutation faults)*
3. *Calculate TF/IDF scores of each test case*
4. *Use coverage and sum of TF/IDF of a test case as predictor values in the training data*
5. *Use previous (mutation) faults as response values in the training data*
6. *Estimate the regression coefficients (weight of each feature) with the training data*
7. *Assign predictor values (coverage and TF/IDF scores) to the model to decide the test schedule*
8. *Run the scheduled test cases*

To evaluate the proposed test case prioritization technique based on information retrieval and coverage information (IRCOV), Kwon et al. [26] used four open-source Java programs XML-Security (XSE), Commons-CLI (CCL), Commons-Collections (CCN), and Joda-Time (JOD). Kwon et al. [26] highlighted that the fault information of the software programs was not sufficiently available, and they were unable to evaluate their approach using available information. Therefore, the authors simulated the faults using mutation. To generate the mutants, they used the mutation framework MAJOR [53, 54]. To reduce the researcher's bias and achieve reliable results, they applied ten-fold validation and divided the mutation faults into ten subsets and assigned each subset to training, validation, and test data.

Concepts used in the original study

The original study [26] makes use of IR concepts. It views a "document" as a test case, "words" as elements covered (e.g., branches, lines, and methods), and "query" as coverage elements in the updated files. TF and IDF scores of the covered elements determine their significance to a test case. The number of times a test case exercises a code element is counted as a TF value. The document frequency (DF) represents the number of test cases exercising an element. IDF is used to find the unique code elements as it is the inverse of DF.

Since the focus of the proposed technique was on less-tested code, the IDF score has more significance, and it is required to minimize the impact of TF. To minimize the impact of TF score on the test case prioritization, they used Boolean values for TF (i.e., $TF = 1$ if a test case covers the code element, $TF = 0$ otherwise). To assign an IDF score to a code element the IDF threshold is used. Kwon et al. [26] define the IDF threshold as:

IDF threshold: *"The maximum number of test cases considered when assigning an IDF score to a code element."*

The IDF threshold is decided by the validation data that consists of faults and related test cases from the previous test execution history or mutation faults.

Finally, the authors used the similarity score between a test case (document) and the changed element (query) to indicate the test cases related to modifications. The similarity score is measured using the sum of TF-IDF scores of common elements in the query.

Key findings of the original study

Using four open-source Java programs, the authors compared their technique with random ordering and standard code-coverage-based methods (i.e., line, branch, and method coverage). They measured the effectiveness using Average Percentage of Fault Detection *(APFD)*.

The authors concluded that their technique is more effective as it increased the fault detection rate by 4.7% compared to random ordering and traditional code coverage-based approaches.

6.4.3 Information about the replication

In this section, we present contextual information, i.e. data availability, roles involved, and replication steps.

Data availability

Kwon et al. [26] did not provide detail on the experimental setup and raw data. Furthermore, they did not publish the automation scripts and data in any open source repository. We contacted them and asked if they could share the experimental package with us. One of them informed us that they had lost the data and did not have any backups related to this study. .

Roles involved

The selection of candidate study for replication was a done through consensus among the authors. Whereas in the subsequent steps every author had a specified role. The first author conceptualized the whole replication process, including the logical assessment of the study to be replicated. The second author who is an industry practitioner wrote the automation scripts and sat up the environment according to the requirements of the software programs. Both the first and second authors jointly performed the replication steps. The first author provided his input for every step, while the second author carried out the actual implementation. The third and fourth authors reviewed study design and implementation steps.

Replication steps

We aimed to make an exact replication of the original study, and therefore we followed the procedure strictly as presented in the original study [26]. The original study IRCOV was built using line, branch, and method coverage. Therefore we also used the line, branch, and method coverage to replicate the IRCOV model. The sequence of events followed in the replication experiment is shown in Figure 6.1.

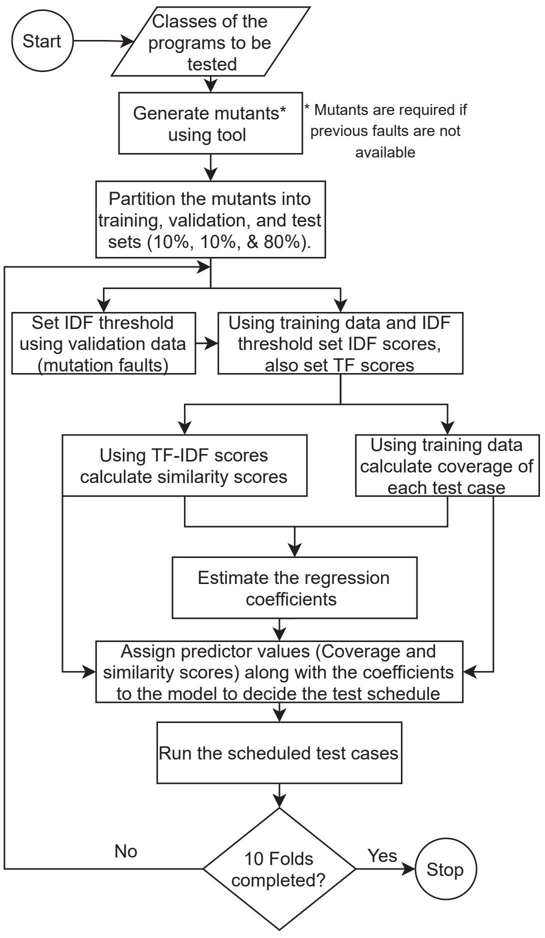
Start
Classes of the programs to be tested
Generate mutants* using tool
* Mutants are required if previous faults are not available
Partition the mutants into training, validation, and test sets (10%, 10%, & 80%).
Set IDF threshold using validation data (mutation faults)
Using training data and IDF threshold set IDF scores, also set TF scores
Using TF-IDF scores calculate similarity scores
Using training data calculate coverage of each test case
Estimate the regression coefficients
Assign predictor values (Coverage and similarity scores) along with the coefficients to the model to decide the test schedule
Run the scheduled test cases
10 Folds completed?
No
Yes
Stop

Figure 6.1: Steps followed to replicate the original study

Replication objects: We attempted to test the replication of the IRCOV with six software programs, using four software programs from the original study and two additional software programs. Table 6.1 presents the details of the software programs used in the replication of IRCOV. The software programs are Common CLI, XML Security, Commons-Collections (CCN), Joda-Time (JOD), Commons-Email, and Log4j. We were able to implement IRCOV with Commons-CLI, but due to various constraints discussed in Section 6.5, we failed to replicate IRCOV with XML Security, Commons-Collections (CCN), and Joda-Time (JOD), Commons-Email, and Log4j.

Table 6.1: Software programs used in replication

Program	Version	LOC	Test Classes	Used in Kwon et al. [26]	Repository
Commons-CLI	1.1, 1.2	13210	23	Yes	SIR & GitHub
XML Security	2.2.3	21315	172	Yes	SIR & GitHub
Commons-Collections	v4.3	128907	490	Yes	GitHub
Joda-Time	v2.10.10	147025	240	Yes	GitHub
Commons-Email	fd6b6**c35a	83154	20	No	GitHub
Log4j	b956**8b969	169646	63	No	SIR & GitHub

Commons-CLI[1] is a library providing an API parsing of command-line arguments. XML-Security[2] for Java is a component library implementing XML signature and encryption standards. Commons-Collections [3] provides java data structures. Joda-Time [4] provides a quality replacement for the Java date and time classes.

To see if the technique (IRCOV) is replicable with other software programs, we selected Commons-Email and Log4j. Commons-Email[5] is built on top of the JavaMail API, and it aims to provide an API for sending email.

Log4j[6] is a Java based logging utility. Log4j 2 was released in 2014 to overcome the limitations of its predecessor version Log4j 1. We obtained the software programs from GitHub and used the test suites provided with the programs.

Mutant generation: The fault information of the software programs was not available, and therefore we used mutation faults instead. In the original study, Kwon et al. [26] used mutation faults as well. For the mutation, we used the tool (MAJOR) [53, 54].

Partitioning mutants into training, validation, and test sets: As per the description in the original study, we classified the mutants into training, validation, and test sets (10%, 10%, and 80%, respectively). To classify the data, we used an online random

[1]https://commons.apache.org/proper/commons-cli/
[2]http://santuario.apache.org/javaindex.html
[3]https://commons.apache.org/proper/commons-collections/
[4]https://www.joda.org/joda-time/
[5]https://commons.apache.org/proper/commons-email/
[6]https://logging.apache.org/log4j/2.x/

generator[7]. We applied the ten-fold validation technique to ensure the reliability of the results and avoid any bias. To create ten folds of each data set (i.e., training, validation, and test sets), we wrote automation scripts [51].

IDF threshold: The purpose of setting up an IDF threshold is to ensure that prioritized test cases should detect faults in less tested code elements. The IDF threshold is decided using validation data containing information of faults and of test cases detecting the faults. To calculate the IDF threshold the authors of the original study [26] suggested using a ratio from 0.1 to 1.0 in Equation 6.1.

$$IDF\ Threshold = no\ of\ test\ cases \times ratio \tag{6.1}$$

We trained the regression model with each threshold using validation data and selected the ratio that led to the minimum training error for the IDF threshold. Based on the minimum training error, Table 6.2 presents the chosen values for the IDF threshold of all ten folds of Commons-CLI. We assigned IDF values to only those code elements whose DF was not above the IDF threshold.

Calculating TF and IDF score: As suggested in the original study [26], we use Boolean values for TF (i.e., $TF = 1$ if the test case covers the element, $TF = 0$ otherwise). The purpose to fix the TF values as 0 or 1 was to ensure that only test case would be prioritized that are targeting less tested code. The IDF score is more significant in this regard. As suggested in the original study [26], we used Equation 6.2 to calculate the IDF score.

$$IDF = 1 + log\left(\frac{\#\ of\ test\ cases}{\#\ of\ test\ cases\ covering\ the\ element}\right) \tag{6.2}$$

Similarity score: The similarity score directly contributes to the IRCOV model. In the regression model (see Equation 6.4), x_2 refers to the similarity score of each test case. We have calculated the similarity scores using Equation 6.3 as suggested in [26].

$$Similarity\ Score(t,q) = \sum_{e \in t \cap q} tf - idf_{\mathrm{e,t}} \tag{6.3}$$

Since TF values are 1 or 0 (i.e., if a test case excises a code element, then TF is 1; otherwise, it is 0), practically similarity scores are the sum of IDF scores of the elements covered by a particular test case.

[7] https://approsto.com/random-line-picker/

Coverage information: The coverage measure is aslo used in the regression model. In Equation 6.4, x_1 refers to the coverage size of each test case. To measure code size (line of code) and coverage of each test case, we used JaCoCo[8].

IRCOV model: We used Equation 6.4 for the linear regression model as suggested in the original study [26].

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{6.4}$$

In Equation 6.4, x_1 is the size of the coverage data for each test case, and x_2 refers to the similarity score of each test case. The value of y represents each test case's fault detection capability, which is proportional to the number of previous faults detected by the test case. In the regression model, three coefficients need to be calculated (i.e., θ_0, θ_1, & θ_2). Here θ_0 represents the intersect, whereas, to calculate θ_1 and θ_2 Kwon et al. [26] suggested using Equation 6.5, which uses y value and respective values of x_1 and x_2. Here y could be calculated using Equation 6.6, where as x_1 and x_2 respectively represent the size of coverage and similarity scores of each test case.

$$theta = (X^T X)^{-1} X^T \vec{y} \tag{6.5}$$

$$y = \sum_{n=1}^{n} \frac{f_{\text{i}}}{log(t_{\text{i}}) + 1} \tag{6.6}$$

Prioritization based on fault detection capability: After having the values of coefficients and variables of regression model (i.e., θ_0, θ_1, θ_2, x_1, and x_2), we determined the fault detection capability of each test case using the IRCOV model (see Equation 6.4). Finally, we arranged the test cases in the descending order of the calculated fault detection capability.

Evaluating the technique: After having a prioritized set of test cases, we ran them on the 50 faulty versions of each fold we created using test data set of mutants. To evaluate the results, we used the average percentage of fault detection (APFD) (see Equation 6.7).

$$APFD = 1 - \frac{TF_1 + TF_2 + ... + TF_{\text{N}}}{nm} + \frac{1}{2n} \tag{6.7}$$

[8]https://www.eclemma.org/jacoco/

6.4.4 Analysis of the replication results

We implemented IRCOV for line, branch, and method coverage. As described above, for all coverage types, we calculated the APFD values for each fold, and we also captured the intermediate results (see Table 6.2).

To compare the replication and the original study results, we translated the APFD values for Commons-CLI from the original study. Then we plotted the APFD values of the original and replication study in the box plot, a statistical tool to visually summarize and compare the results of two or more groups [42, 49]. Box plot of APFD values enabled us to visually compare the replication and original study results.

To triangulate our conclusions, we applied hypothesis testing. We used Wilcoxon signed-rank test to compare the results of IRCOV original and IRCOV replication results. Also in the original study Kwon et al. [26] used Wilcoxon signed-rank test to compare the IRCOV results with the baseline methods. Wilcoxon signed-rank test is suitable for paired samples where data is the outcome of before and after treatment. It measures the difference between the median values of paired samples [50]. In our case, we were interested in measuring the difference between the median APFD values of IRCOV original and IRCOV replication. Therefore, the appropriate choice to test our results was Wilcoxon signed-rank test.

We tested the following hypothesis:

H_{0LC}: There is no significant difference in the median APFD values of original and replication study using line coverage.

H_{0BC}: There is no significant difference in the median APFD values of original and replication study using branch coverage.

H_{0MC}: There is no significant difference in the median APFD values of original and replication study using method coverage.

6.4.5 Automation of replication

The replication was implemented using Python scripts. They are available in [51]. Figure 6.2 presents the details of automation steps for the replication of IRCOV. The original study's authors proposed that ten-fold-based execution is needed (when historical data is not available) to evaluate their original technique. Therefore, our implementation (fold_generator) [51] generates ten folds of the object program at the first stage. Thereafter, it generates fifty faulty versions of each fold, whereas each version contains 5-15 mutants (faults). After generating the faulty versions, the script makes the corresponding changes in the code. Finally, the tests are executed, and their results are extracted. Later, using the test results, we calculate the APFD values of each fold. The

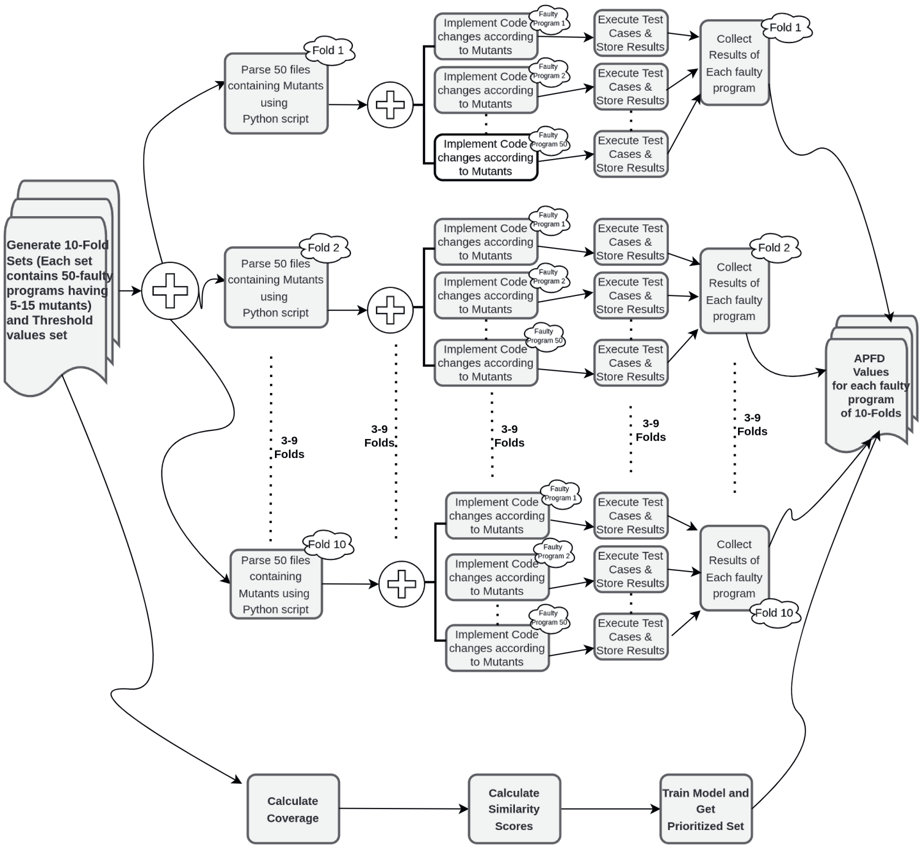
Generate 10-Fold Sets (Each set contains 50-faulty programs having 5-15 mutants) and Threshold values set
Fold 1
Parse 50 files containing Mutants using Python script
Fold 2
Parse 50 files containing Mutants using Python script
Fold 10
Parse 50 files containing Mutants using Python script
Implement Code changes according to Mutants
Faulty Program 1
Faulty Program 2
Faulty Program 50
Execute Test Cases & Store Results
Collect Results of Each faulty program
3-9 Folds
APFD Values for each faulty program of 10-Folds
Calculate Coverage
Calculate Similarity Scores
Train Model and Get Prioritized Set

Figure 6.2: Steps to automate the replication of IRCOV

calculation of APFD values is the only step not handled in our script. We used excel sheets to calculate APFD values.

6.4.6 Threats to validity

Internal validity

Internal validity refers to the analysis of causal relations of independent and dependent variables. In our case, we have to see if the different conditions affect the performance of IRCOV. IRCOV depends upon two inputs, coverage of each test case and a similarity score calculated based on TF-IDF. We used the test cases available within the software programs. Therefore, we do not have any control over the coverage of these test cases. However, the choices of mutants can impact the similarity score. To avoid any bias, we generated the mutants using a tool and used a random generator to select the mutants for different faulty versions of the software programs. Furthermore, we trained IRCOV sufficiently before applying it to test data by following the ten-fold validation rule. Since we measured the performance of IRCOV using the APFD measure, the results of the successful case were not significantly different from the original study's results. Therefore we can argue that our treatment did not affect the outcome of IRCOV. Hence minimized the threats to the internal validity.

Construct validity

Construct validity is concerned with the underlying operational measures of the study. In our case, since it is a replication study and we followed the philosophy of exact replication [46]. Therefore, if the original study suffers of any aspects of construct validity, the replication may do so. For instance, the use of mutation faults could be a potential threat to the construct validity because of the following two reasons

- Mutation faults may not be representative of real faults.
- Possible researchers' bias concerning the nature of mutation faults.

Concerning the first reason, the use of mutation faults to replace the real faults is an established practice and researchers claim that mutation faults produce reliable results and hence can replace the real faults [40, 41]. To avoid any bias, we used an automated mutation tool to generate the mutants. Also to select the mutants for validation, training, and test set we used an automated random selector. Hence no human intervention was made during the whole process. Furthermore, we discussed the strengths and weaknesses of different tools.

External validity

External validity is the ability to "generalize the results of an experiment to industrial practice" [48]. The software programs used in the replication study are small and medium-sized Java programs. Therefore, we can not claim the generalizability of results to large-scale industrial projects. The results produced in replication align well with the results of the original study. However, we could not demonstrate the use of the technique on the additional software programs.

6.5 Results

This section presents the findings from the replication. The results are organized according to research questions listed in Section 6.4.

6.5.1 RQ1. Degree to which the replication is feasible to implement.

The first goal was to see the possibility to replicate the IRCOV technique with the four software programs described in the original study [26] (RQ1.1) and with two additional software programs (RQ1.2).

Concerning RQ1.1, out of four software programs used in the original study, we could completely replicate the IRCOV technique with only one software program Commons-CLI. Whereas, with the other three software programs, (i) XML Security, (ii) Commons-Collections (CCN), and (iii) Joda-Time (JOD), the replication was partially successful or unsuccessful. Concerning RQ1.2, we attempted to replicate IRCOV with two software programs, (i) Commons-Email and (ii) Log 4j, but we were unable successfully to replicate the IRCOV with these additional software programs as well. The detail of completely successful, partially successful and unsuccessful cases are discussed in the subsequent paragraphs. *Successful replication implementation:* We successfully replicated IRCOV with Commons-CLI. After going through the steps presented in Section 6.4.3, for every fold, we were able to calculate the respective coverage information and similarity score of each test case. Table 6.2 presents the intermediate results for the replication of IRCOV with Commons-CLI. These include, training error, chosen value of IDF threshold, regression coefficient θ_0, coverage weight θ_1, and weight for similarity score θ_2).

To evaluate the performance of IRCOV, we have calculated APFD values for all ten folds of each coverage type (branch, line, and method) (see Table 6.3). For branch coverage, the APFD value ranges from 0.548 to 0.874, whereas the average APFD

Table 6.2: Simulation parameters for Commons-CLI. (MC = Method coverage, LC = Line coverage, & BC = Branch coverage)

Fold Name	Coverage Type	Training Error	IDF Threshold	θ_0	θ_1	θ_2
	MC	1.0694	7	-0.3478	0.0187	0.1426
Fold1	LC	0.9770	2	0	0	0
	BC	0.8876	2	0	0	0
	MC	0.3195	5	-0.7976	0.0323	-0.1472
Fold2	LC	0.3533	6	-0.6084	0.0088	-0.1343
	BC	0.3567	5	-0.3386	0.0178	-0.2095
	MC	0.6411	6	-0.0286	0.0008	0.0796
Fold3	LC	0.6404	6	-0.0498	0.0004	0.0736
	BC	0.6405	6	-0.0380	0.0008	0.0736
	MC	0.4783	6	-0.0687	0.0097	0.1677
Fold4	LC	0.4551	6	-0.1086	0.0032	0.1365
	BC	0.4947	6	0.1240	0.0045	0.1683
	MC	0.1838	5	0.0309	0.0068	0.0558
Fold5	LC	0.1856	4	0.0859	0.0018	0.0612
	BC	0.1876	4	0.1406	0.0038	0.0516
	MC	0.2247	2	-0.5284	0.0194	0.3548
Fold6	LC	0.1795	2	-0.4869	0.0052	0.3470
	BC	0.1549	2	-0.3978	0.0119	0.3149
	MC	0.1382	10	-0.1479	0.0115	-0.0141
Fold7	LC	0.1364	10	-0.0833	0.0030	-0.0234
	BC	0.1390	10	0.0028	0.0065	-0.0235
	MC	0.2020	6	0.4389	-0.0024	0.0839
Fold8	LC	0.2046	6	0.3401	-0.0001	0.0715
	BC	0.2046	6	0.3286	-0.00001	0.0694
	MC	0.1490	6	0.1652	-0.0032	0.1473
Fold9	LC	0.1532	6	0.0540	-0.0002	0.1344
	BC	0.1517	6	0.0862	-0.0012	0.1434
	MC	0.0339	10	-0.1253	0.0017	0.0267
Fold10	LC	0.0339	10	-0.1127	0.0004	0.0261
	BC	0.0343	10	-0.0920	0.0007	0.0278

value for branch coverage is 0.747. The APFD values for line coverage range from 0.610 to 0.874, and the average APFD value for line coverage is 0.809. Finally, the APFD value for method coverage ranges from 0.585 to 0.865, and the average APFD for method coverage is 0.772. These results show that the IRCOV model performed best for the line coverage as the mean APFD for line coverage is highest among all.

Partial or unsuccessful replication: Our first unsuccessful replication was concerning XML Security. We did not find all the program versions used in the original study (Study [26]). Therefore, we decided to use the versions that have slightly similar major/minor release versions. We downloaded available XML Security versions 1, 1.5 and 2.2.3. The first two downloaded versions (version 1 and version 1.5) were not compiling due to the unavailability of various dependencies. The logs from the compilation failures are placed in folder "LogsXmlSecurit" available at [52].

We were able compile the third XML Security version 2.2.3, but we could not continue with it, because this version contained several failing test cases (see [52]). With already failing test cases it was difficult to train the model correctly and get the appropriate list of prioritized test cases.

The second and third attempts were made on Commons-Collection and Joda-Time. Compared to other projects used in replication experiment these projects contain more test classes (see Table 6.1). The mutants were generated for each of these projects. Out of fifty faulty versions, the first faulty version of Commons-Collection (Fold1) took 40 minutes in execution, and similarly, the first faulty version of Joda-Time (Fold1) took 36 minutes in execution. This was a limitation since, to train the model, we had to use ten folds, and for each fold, we had to execute 50 faulty versions containing five to fifteen faults in each faulty version. Our estimate shows that it would take 2000 minutes to train Commons-Collection and 1810 minutes to train Joda-Time. It would take 64 hours (approx) before we get the final results of these two programs – provided that the Internet connection remains stable and working. Furthermore, parsing and analyzing the data of many test classes is also a time-consuming task. Due to these technical and resource limitations, we abandoned the replications for these two projects. We have provided further discussion on these issues in Section 6.6.

The fourth unsuccessful replication attempt was executed on Commons-Email. This time the replication was unsuccessful because of faulty mutants generated by the mutant software. For instance, it suggested replacing variable names with 'null' (see Listing 1 & 2). The actual code was $this.name = null$; while after mutant injection, the code turned to $this.null = null$.

Listing 6.1: Faulty mutant generated by the tool

```
35:EVR:<IDENTIFIER(java.lang.String)>:<DEFAULT>:org.apache.commons.mail.
ByteArrayDataSource@setName(java.lang.String):214:name |==> null
```

Listing 6.2: Code generated after the insertion of faulty mutant

```
public void setName(final String name)
   {
   // this.name = null;
       this.null =  null;  }
```

Another type of faulty instances were when MAJOR suggested to modify a line in the code that resulted in Java compilation errors (such as “unreachable statement”). There were several such faulty mutants that made the program fail to compile, and hence no further processing was possible. The detail of all faulty mutants is available in the folder “CommonsEmail” at [52].

We also made unsuccessful attempts to change the mutant generator to rectify this problem. However, each mutant generator presented a new set of problems. The lessons learnt from usage of different mutant generators are described in next section.

The fifth replication attempt was executed on Log4j. We followed all the steps (using our automated scripts) proposed by the authors of the original study. We successfully generated the mutants for this program. However, the replication was stopped at the point when the steps to train the model failed. The proposed approach in the original study is based on the coverage information of each code class and test-class. This time the issue was caused by the low coverage of the test cases. During the training of the model, we realized that because of low coverage of the test cases, we were unable to calculate the values of regression coefficients, and as a result, we could not generate the prioritized set of test cases. We developed a Jupyter notebook to describe each steps of this partially successful replication (see [51]). Compared to the other software programs selected in this study, with 169646 LOC, Log4j is a large program. Thus, a lot of time was needed to train the model for Log4j. For all ten folds, with fifty faulty versions of each fold and with five to fifteen faults in each faulty version, it required approximately 60 hours to train the model.

Key findings: Concerning RQ1, the replication was only feasible in one of four cases; the key reasons are listed below.

1. The inability to use the system under test was caused by compatibility issues (unavailability of system versions and dependencies).
2. Already failing test cases made the replication fail.
3. Mutant generators created issues in running the replication, workarounds were difficult to implement.
4. Test cases require a certain level of coverage to train the model.
5. More effort is required to train the model for large-sized software programs.

Table 6.3: APFD values for all ten folds of each coverage type

Folds	Branch Coverage	Line Coverage	Method Coverage
Fold 1	0.874	0.874	0.865
Fold 2	0.816	0.866	0.790
Fold 3	0.646	0.643	0.613
Fold 4	0.757	0.816	0.755
Fold 5	0.725	0.715	0.721
Fold 6	0.796	0.829	0.829
Fold 7	0.841	0.839	0.839
Fold 8	0.610	0.610	0.585
Fold 9	0.548	0.622	0.594
Fold 10	0.736	0.803	0.803

6.5.2 RQ2. Comparison of the results to the original study.

Figure 6.3 presents the APFD boxplots of the original and replication study for Commons-CLI. Boxplots with blue patterns represent the original study results, and boxplots with gray patterns represent the replication study results. We can see that in all cases, the APFD values of the original study are slightly better compared to the values of the replication. We applied statistical tests to detect whether the results of the replication and the original study differ.

To compare the replication results for branch, line, and method coverage of Commons-CLI with the original study's results, we applied Wilcoxon singed-rank test. The results are significant if the p-value is less than the level of significance [47]. In our case, the difference between the two implementations would be significant if the p-value is less than 0.05.

Table 6.4 presents the results of statistical test. The p-value for branch coverage is 0.475, which is greater than 0.05 (significance level). Therefore, we can not reject the null hypothesis. That means we can not show a significant difference in the APFD values for branch coverage of Commons-CLI between the replication and the original study.

Similarly, the p-value for line coverage is 0.415, greater than the set significance level. Based on the statistical results, we can not reject the null hypothesis. This implies that we can not show a significant difference in the APFD values for the line coverage of Commons-CLI between the replication and the original study.

Finally, the p-value for method coverage is 0.103, based on this result, we can not reject the null hypothesis. Therefore no significant difference in the APFD values for the method coverage of Commons-CLI between the replication and the original study.

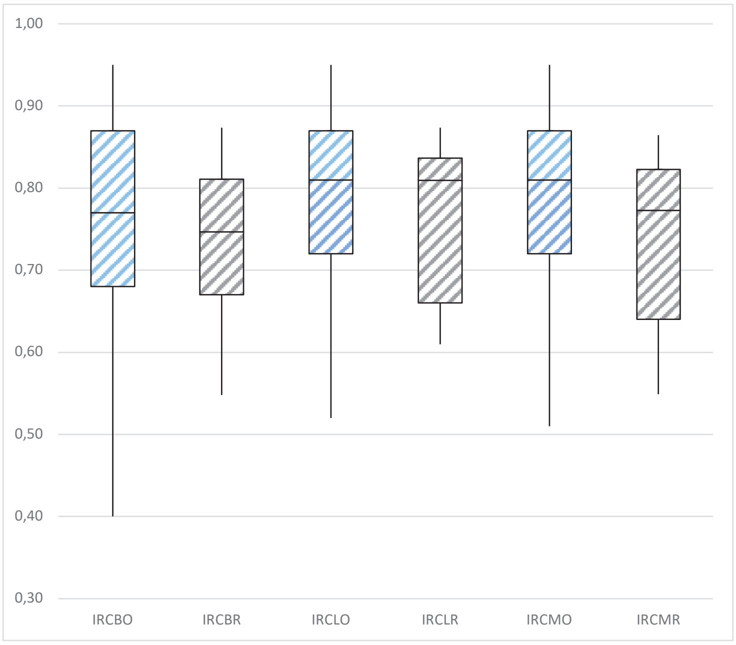

Figure 6.3: APFD boxplots for IRCOV Original vs IRCOV replication

IRCBO= IRCOV branch coverage original, IRCBR= IRCOV branch coverage replication
IRCLO= IRCOV line coverage original, IRCLR= IRCOV line coverage replication
IRCMO= IRCOV method coverage original, IRCMR=IRCOV method coverage replication

Table 6.4: Statistical results of replication compared to the original study for Commons-CLI.

Coverage	α	p-value	95% Conf. Int.
Branch	0.05	0.475	0.646 - 0.816
Line	0.05	0.415	0.668 - 0.845
Method	0.05	0.103	0.652 - 0.827

From the t-test results, we can conclude that for all three coverage types (branch, line, and method), we did not find any significant difference between the replication and the original study. Therefore, we can state that the replication experiment did not deviate from the original result to a degree that would lead to the test detecting a significant difference.

Key findings: Concerning RQ2, we compared the replication results of the successful case (i.e., Commons-CLI) with the original study's results. Below are the key findings for RQ2.

1. The statistical test did not detect a significant difference in the APFD values of the replication and the original study concerning the three coverage measures investigated.
2. We conclude that the results of the original study are verifiable for Commons-CLI.

6.6 Discussion

6.6.1 Lessons learned of replicating artefact-based studies in software testing

We replicated the study presented in [26] with the intent of promoting artefact-based replication studies in software engineering, validating the correctness of the original study, and exploring the possibilities to adopt regression testing research in the industry.

Overall, it is essential to capture and document assumptions and constraints concerning the techniques that are replicated, as well as the conditions for being able to run a replication. We highlight several factors of relevance that were observed.

Conditions concerning System under Test (SuT) complexity: From the replication results, we learned that besides the various constraints, the technique (IRCOV) presented in [26] is replicable for small and medium software programs provided the availability of context information. The technique with its current guidelines is difficult to implement with large-size software programs because it requires a significant amount of effort to train the model. The restriction of 10-folds, fifty faulty versions for every fold, and 5 to 15 faults in every faulty version would require a substantial effort. For example, while attempting to replicate the original study with Commons-Collection and Joda-Time, we estimated that it would take approximately 67 hours to train the model for each of these software programs. This limitation can be managed by reducing the number of faulty versions for each fold, but this may degrade the accuracy and increase the training error.

Conditions concerning the characteristics of the test suite: Test cases available with Log4j have low coverage, limiting the chance of correctly training the model and generating a reasonable prioritization order of the test cases. Coverage is one of the primary inputs required for the test case prioritization using the IRCOV model. Another problem we encountered was the presence of already failing test cases in one of the versions of XML Security. Test cases are used to calculate the coverage score and similarity scores of the project. If a handful of test cases fail (as in XML Security version 2.2.3), wrong coverage information and similarity scores are calculated. This results in the wrong prioritization of test cases as well faulty training of the model (which is used to identify prioritized test cases). Another drawback with failing test cases concerns the use of mutations. If tests are already failing and when mutants are introduced, then the effectiveness is unreliable as tests are already failing because of other issues. Further conditions may be of relevance in studies focusing on different aspects of software testing. Here, we would highlight how important it is to look for these conditions and document them. This is also of relevance for practice, as it demonstrates under which conditions a technique may or may not be successful in practice.

Availability of experimental data for artefact-based test replications: One of the constraints regarding the replicability of the IRCOV technique is the availability of experimental data. For example, the authors of the original study [26] stated that they used in-house built tools to conduct the experiment but, they did not provide any source of these tools, also not including details of the automation tools. Therefore, it took a significant effort to set up the environment to replicate IRCOV with the first program. There are various limitations concerning the data sets and tools required to work with the recommended steps. Regarding data sets, we have recorded the findings in Section 3.4. These include the compatibility of SIR artefacts. For example, because of various dependencies, we faced difficulties while working with XML Security version 1. While working with version 2.2.3 of XML Security, we encountered errors in the version. Therefore, we could not collect the coverage information. Ultimately, we were unable to replicate the technique with any of the versions of XML Security.

Reflections on mutant generators: In the absence of failure data, the authors of the original study suggested using mutation faults, and they used the MAJOR mutation tool to generate the mutants. In one of our cases (Commons-Email), the mutation tool (MAJOR) generated inappropriate mutants that led to the build failure. Therefore, no further progress was possible with this case. One option could be to remove the faulty mutants and include only correct mutants. In our opinion, removing faulty mutants may bring different results from the original study, thus limiting the validity of replication.

Table 6.5: Comparison of mutant generators

No	Mutation Tool	Benefits	Challenges
1	MAJOR[9]	(i) Easy to use. (ii) Most commonly used mutant generator.	(i) Faulty mutant generated. (ii) Needs upgrade to latest Java versions (iii) Documentation needs improvement.
2	µJava[10]	(i) IDE plugin available (ii) User decides what types of mutants can be generated.	(i) Exporting mutants separately is not supported (ii) Does not support latest Java versions (iii) GUI crashes often while generating mutants.
3	Jester[11]	Two types of Jester versions, a complete version and a simple version.	Latest update is more than 10 years ago. We were unable to generate mutations or start the program despite of following all steps.
4	Jumble[12]	(i) Support recent Java versions. (ii) Integration with IDE Supported.	Unable to generate mutants despite following examples. Latest update was 6 years ago.
5	PIT[13]	The most recent and complete mutant generator. Mutants are generated and tests are executed. A report is generated for the user.	(i) Unable to export the mutants. (ii) Lack of diversity in the mutants. (iii) Each execution produced exact same mutants.

To overcome the difficulty with replication of project 3 (Commons-Email), we tried different open-source mutation generators available. Each of these presented various benefits and challenges that are documented in Table 6.5. After trying out different mutation tools, we learned that among the available options, MAJOR is an appropriate tool for Java programs, as it generates the mutants dynamically.

Reflections on the requirements of the experimental setup:

We faced limitations while attempting the replication of technique with larger software programs. The time required to train the model with larger programs was much longer. We do not know whether the original authors experienced similar problems

[9]https://mutation-testing.org/
[10]https://cs.gmu.edu/ offutt/mujava/
[11]http://jester.sourceforge.net/
[12]http://jumble.sourceforge.net/
[13]https://pitest.org/

since the original study did not discuss similar issues. We suggest that original studies explicitly report any special requirements (e.g., hardware, software, etc.) concerning experimental setup. A collaboration between the original authors and replicators may help resolve such issues. However, as mentioned in the earlier sections, we could not establish a collaboration with the authors of the original study.

Reflections on the IRCOV technique: Besides the various limitations highlighted earlier, the IRCOV technique is replicable for smaller software programs provided the availability of required information. The replication results of the successful case show that, at least for Commons-CLI, the original authors' claim about the performance of the IRCOV technique was verifiable. The technique needs to be modified, though, to decrease the time required for programs with many test classes. Then, the technique presented in the original study can be valuable from an industry perspective because it focuses on prioritizing test cases detecting faults in less tested code while taking coverage of test cases into account during the prioritization process. Besides increasing the test suite's rate of fault detection, it can help the practitioners work with one of their goals (e.g., controlled fault slippage). Looking at regression testing in practice, the practitioners recognize and measure the coverage metric [21]. The only information that needs to be maintained in the companies is failure history. In the presence of actual failure data, we do not need to use the mutants to train the IRCOV model extensively, and we can reduce the number of faulty versions for each fold and the number of folds.

Overall, pursuing the first research question (RQ1) provided us with a deeper insight into the various aspects and challenges related to external replication. The lessons learned in this pursuit are interesting and to provide recommendations in the context of replication studies in software engineering. From the existing literature, it was revealed that the trend of replication studies in software engineering is not encouraging [33, 38]. The studies report that the number of internal replications is much higher than external replications [35, 38]. While searching the related work, we observed that in the software testing domain, compared to the internal replications, external replications are few in numbers. There could be several reasons for the overall lower number of replication studies in software engineering, but we can reflect on our experiences concerning the external replications as we have undergone an external replication experiment.

One issue we would like to highlight is the substantial effort needed to implement the replication. Replication effort can be substantially reduced with more detailed documentation of the original studies, the availability of appropriate system versions and their dependencies, and the knowledge about prerequisites and assumptions. Better documentation and awareness of conditions may facilitate a higher number of replications in the future.

6.6.2 General lessons learned for artefact-based replications

Table 6.6 provides an overview of challenges we encountered during the replications. It lists the possible impact of each challenge on the results of replication, and the table also presents a list of recommendations for researchers. The following provides a brief discussion on the lessons learned in this study.

Table 6.6: Recommendations drawn from the challenges/lessons learned

Challenge	Impact	Recommendation
Documentation of original experimental setup	Replicators have to invest additional effort to understand the context of the study.	Original authors need to maintain/publish a comprehensive documentation of experimental setup.
Collaboration with the authors of original studies	In the absence of experimental data and support from original authors can make the replication process more complicated.	In the event of request from the replicators the authors of the original study provide assistance in the form of essential information regarding the original experiment.
Issues with the open-source data sets	Replication experiments may fail due to these issues.	open-source repositories need to maintained and be up to date.
System under Test (SuT) and tools compatibility issues	Any compatibility issue of the tools required to replicate the original experiment can create a bottleneck for the replication.	Such tools (e.g., Mutation tools in our case) need to be maintained to make them compatible with new development frameworks. The same applies to the system under test.

Documenting the original experiment: The authors of the original studies need to maintain and provide comprehensive documentation concerning the actual experiment. The availability of such documents will help the independent replicators understand the original study's context. In the absence of such documentation, the replicators need to invest more effort to understand the original study's context. In this regard, we suggest using open-source repositories to store and publish the documentation. The documentation may contain the detail of the experimental setup, including the tools used to aid the original experiment, automation scripts (if used/developed any), and the internal and final results of the study. Furthermore, the authors can also include detail about any special requirements or considerations that need to be fulfilled for the successful execution of the experiment.

Collaboration with the original authors: Because of page limits posed by the journals and conferences, every aspect of the study can not be reported in the research paper. Sometimes, the replicators need assistance from the original authors regarding any missing aspect of the study. Therefore, it is essential that in case of any such query from the replicators, the original study's authors should be able to assist them. Such cooperation can promote replication studies in software engineering. In our opinion, lack of collaboration is one reason for fewer replication studies in software engineering. However, it is important to still conduct the replications as independently as possible due to possible biases (i.e., avoiding to turn an external replication into an internal one).

Maintaining open-source repositories: Open-source repositories (one example being SIR) provide an excellent opportunity for researchers to use the data sets while conducting software engineering experiments. A large number of researchers have benefited from these data sets. We learned that some of the data sets available in repositories are outdated and need to be maintained. Such data sets are not helpful, and studies conducted using these data sets would be complex to adopt/replicate. It is therefore essential that authors explicitly state the versions they used in their own studies. In addition, we recommend that authors of original studies as well as replications ensure that the dependencies or external libraries are stored to facilitate the replications of system under test.

Tools compatibility: In many cases, the authors need to use open-source tools to assist the execution of their experiment. Such tools need to be well maintained and updated. In case of compatibility issues, these tools can hinder the replication process. For example, the study we replicated uses a mutation tool (MAJOR). Although it is one of the best available options, the tool generated inappropriate mutants for one of our cases due to some compatibility issues. Ultimately, after a significant effort, we had to abandon the replication process for that case. Here, we also would like to highlight that one should document the versions of the tools and libraries used (also including scripts written by the researchers - e.g., in python).

Documenting successes and failures in replications: Besides the significance of documenting every aspect of the original experiment, recording every single event of replication (success & failure) is critical for promoting future replications and industry adoptions of research. We recommend storing the replication setups and data in open-source repositories and providing the relevant links in the published versions of the articles.

Automation of replication: A key lesson learned during the replication of the original study is that the documentation of the setup and execution of replication could be automated with the help of modern tools and programming languages. This automation will help in reproducing the original results and analysis for researchers reviewing or producing the results from the studies. We have provided programming scripts that describe and document all the steps (and the consequences of these steps).

6.7 Conclusions

This study reports the results of a replication experiment to evaluate a test case prioritization technique using information retrieval (IR) concepts proposed initially by Kwon et al. [26]. We attempted to replicate the original study using six Java programs: Commons-CLI, XML Security, Commons-Collection, Joda-Time, Commons-Email, and Log4j. In the first research question (RQ1), the aim was to see if the technique is replicable, and in the second research question (RQ2), we aimed to see if the replication results conform to the ones presented in the original study.

We have faced various challenges while pursing RQ1, these challenges are related to the availability of original experimental setup, collaboration with the original authors, system under test, test suites, and compatibility of support tools. We were able to replicate the technique only with Commons-CLI, which is a smaller program. Based on our experience we can conclude that the technique is replicable for small software programs (such as Commons-CLI) subject to the availability of required information. However, it is hard to implement the technique with the larger software programs because it requires a substantial effort to implement it for a larger program. Concerning RQ1, the important concluding point is that it is not feasible to externally replicate an experiment when context information and relevant data are not available. Furthermore, without the support of the original authors, it becomes a challenging task.

To verify the original study's results (RQ2), we compared the replication results for Commons-CLI with the ones presented in the original study. These results validated the original study's findings as the statistical test confirms no significant difference between the APFD values of the replication and the actual experiment. However, we may say that our results partially conformed with the original study because we could not replicate the technique with all selected artefacts due to missing dependencies, broken test suites, and other reasons highlighted earlier.

The technique can be helpful in the industrial context as it prioritizes the test cases that target the less tested code. It can help the practitioners to control fault slippage. However, it needs some improvements in training and validation aspects to scale the technique to the industry context. To support the future replications/adoption of IRCOV, we have automated the IRCOV steps using Python (Jupyter notebook).

We plan to work with more artefacts with actual faults to test the technique's (IRCOV) effectiveness in the future, and we plan to see the possibilities of scaling it up for larger projects. In addition to that, we want to evaluate our proposed guidelines (under lessons learned) using different studies from industrial contexts.

6.8 References

[1] R. H. Rosero, O. S. Gómez, and G. D. R. Rafael, "15 years of software regression testing techniques - A survey," *Int. J. Software Eng. Knowl. Eng.*, vol. 26, no. 5, pp. 675–690, 2016.

[2] M. Felderer and E. Fourneret, "A systematic classification of security regression testing approaches," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 3, pp. 305–319, 2015.

[3] A. Zarrad, "A systematic review on regression testing for web-based applications," *JSW*, vol. 10, no. 8, pp. 971–990, 2015.

[4] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information & Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[5] R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, "Effective regression test case selection: A systematic literature review," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 29:1–29:32, 2017.

[6] C. Catal, "On the application of genetic algorithms for test case prioritization: a systematic literature review," in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*. ACM, 2012, pp. 9–14.

[7] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Softw. Test., Verif. Reliab.*, vol. 22, no. 2, pp. 67–120, 2012.

[8] D. Qiu, B. Li, S. Ji, and H. K. N. Leung, "Regression testing of web service: A systematic mapping study," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 21:1–21:46, 2014.

[9] Y. Singh, A. Kaur, B. Suri, and S. Singhal, "Systematic literature review on regression test prioritization techniques," *Informatica (Slovenia)*, vol. 36, no. 4, pp. 379–408, 2012.

[10] C. Catal and D. Mishra, "Test case prioritization: a systematic mapping study," *Software Quality Journal*, vol. 21, no. 3, pp. 445–478, 2013.

[11] N. bin Ali, E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, S. Kunze, and M. Varshosaz, "On the search for industry-relevant regression testing research," *Empirical Software Engineering*, pp. 1–36, 2019.

[12] M. Khatibsyarbini, M. A. Isa, D. N. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," *Information and Software Technology*, vol. 93, pp. 74–93, 2018.

[13] A. Bajaj and O. P. Sangwan, "A systematic literature review of test case prioritization using genetic algorithms," *IEEE Access*, vol. 7, pp. 126 355–126 375, 2019.

[14] J. A. P. Lima and S. R. Vergilio, "Test case prioritization in continuous integration environments: A systematic mapping study," *Information and Software Technology*, vol. 121, p. 106268, 2020.

[15] O. Dahiya and K. Solanki, “A systematic literature study of regression test case prioritization approaches,” *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 2184–2191, 2018.

[16] A. Rainer, D. Jagielska, and T. Hall, “Software engineering practice versus evidence-based software engineering research,” in *Proceedings of the ACM Workshop on Realising evidence-based software engineering (REBSE ’05)*, 2005, pp. 1–5. [Online]. Available: http://doi.acm.org/10.1145/1082983.1083177

[17] A. Rainer and S. Beecham, “A follow-up empirical evaluation of evidence based software engineering by undergraduate students,” in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 2008, pp. 78–87.

[18] E. D. Ekelund and E. Engström, “Efficient regression testing based on test history: An industrial evaluation,” in *Proceedings of IEEE International Conference on Software Maintenance and Evolution, ICSME*, 2015, pp. 449–457.

[19] E. Engström and P. Runeson, “A qualitative survey of regression testing practices,” in *Proceedings of the 11th International Conference on Product-Focused Software Process Improvement PROFES*, 2010, pp. 3–16.

[20] N. M. Minhas, K. Petersen, N. Ali, and K. Wnuk, “Regression testing goals-view of practitioners and researchers,” in *24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*. IEEE, 2017, pp. 25–32.

[21] N. M. Minhas, K. Petersen, J. Börstler, and K. Wnuk, “Regression testing for large-scale embedded software development – exploring the state of practice,” *Information and Software Technology*, vol. 120, p. 106254, 2020.

[22] H. Do, S. Elbaum, and G. Rothermel, “Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact,” *Empirical Software Engineering*, vol. 10, no. 4, pp. 405–435, 2005.

[23] ISO/IEC/IEEE, “International standard - systems and software engineering–vocabulary,” *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, Aug 2017.

[24] S. Elbaum, A. G. Malishevsky, and G. Rothermel, “Test case prioritization: A family of empirical studies,” *IEEE transactions on software engineering*, vol. 28, no. 2, pp. 159–182, 2002.

[25] M. J. Harrold and A. Orso, “Retesting software during development and maintenance,” in *Proceedings of the Frontiers of Software Maintenance Conference*, 2008, pp. 99–108.

[26] J.-H. Kwon, I.-Y. Ko, G. Rothermel, and M. Staats, “Test case prioritization based on information retrieval concepts,” in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 19–26.

[27] Q. Peng, A. Shi, and L. Zhang, “Empirically revisiting and enhancing ir-based test-case prioritization,” in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 324–336.

[28] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry, “An information retrieval approach for regression test prioritization based on program changes,” in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 268–279.

[29] S. Yadla, J. H. Hayes, and A. Dekhtyar, “Tracing requirements to defect reports: an application of information retrieval techniques,” *Innovations in Systems and Software Engineering*, vol. 1, no. 2, pp. 116–124, 2005.

[30] H. Fang, T. Tao, and C. Zhai, “A formal study of information retrieval heuristics,” in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 49–56.

[31] M. Shepperd, N. Ajienka, and S. Counsell, “The role and value of replication in empirical software engineering results,” *Information and Software Technology*, vol. 99, pp. 120–132, 2018.

[32] N. Juristo and O. S. Gómez, *Replication of Software Engineering Experiments*. Springer Berlin Heidelberg, 2012, pp. 60–88. [Online]. Available: https://doi.org/10.1007/978-3-642-25231-0_2

[33] M. Cruz, B. Bernárdez, A. Durán, J. A. Galindo, and A. Ruiz-Cortés, “Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018,” *IEEE Access*, vol. 8, pp. 26 773–26 791, 2019.

[34] A. Santos, S. Vegas, M. Oivo, and N. Juristo, “Comparing the results of replications in software engineering,” *Empirical Software Engineering*, vol. 26, no. 2, pp. 1–41, 2021.

[35] R. M. Bezerra, F. Q. da Silva, A. M. Santana, C. V. Magalhaes, and R. E. Santos, “Replication of empirical studies in software engineering: An update of a systematic mapping study,” in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2015, pp. 1–4.

[36] C. V. de Magalhães, F. Q. da Silva, R. E. Santos, and M. Suassuna, “Investigations about replication of empirical studies in software engineering: A systematic mapping study,” *Information and Software Technology*, vol. 64, pp. 76–101, 2015.

[37] J. L. Krein and C. D. Knutson, “A case for replication: synthesizing research methodologies in software engineering,” in *RESER2010: proceedings of the 1st international workshop on replication in empirical software engineering research*. Citeseer, 2010, pp. 1–10.

[38] F. Q. Da Silva, M. Suassuna, A. C. C. França, A. M. Grubb, T. B. Gouveia, C. V. Monteiro, and I. E. dos Santos, “Replication of empirical studies in software engineering research: a systematic mapping study,” *Empirical Software Engineering*, vol. 19, no. 3, pp. 501–557, 2014.

[39] H. Do, G. Rothermel, and A. Kinneer, “Empirical studies of test case prioritization in a junit testing environment,” in *15th international symposium on software reliability engineering*. IEEE, 2004, pp. 113–124.

[40] H. Do and G. Rothermel, “On the use of mutation faults in empirical assessments of test case prioritization techniques,” *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733–752, 2006.

[41] J. H. Andrews, L. C. Briand, and Y. Labiche, “Is mutation an appropriate tool for testing experiments?” in *Proceedings of the 27th international conference on Software engineering*, 2005, pp. 402–411.

[42] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, “The effects of time constraints on test case prioritization: A series of controlled experiments,” *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 593–617, 2010.

[43] J. F. S. Ouriques, E. G. Cartaxo, and P. D. Machado, “Test case prioritization techniques for model-based testing: a replicated study,” *Software Quality Journal*, vol. 26, no. 4, pp. 1451–1482, 2018.

[44] M. Hasnain, I. Ghani, M. F. Pasha, I. H. Malik, and S. Malik, “Investigating the regression analysis results for classification in test case prioritization: A replicated study,” *International Journal of Internet, Broadcasting and Communication*, vol. 11, no. 2, pp. 1–10, 2019.

[45] J. C. Carver, “Towards reporting guidelines for experimental replications: A proposal,” in *1st international workshop on replication in empirical software engineering*, vol. 1. Citeseer, 2010, pp. 1–4.

[46] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, “The role of replications in empirical software engineering,” *Empirical software engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[47] J.-B. Du Prel, G. Hommel, B. Röhrig, and M. Blettner, “Confidence interval or p-value?: part 4 of a series on evaluation of scientific publications,” *Deutsches Ärzteblatt International*, vol. 106, no. 19, p. 335, 2009.

[48] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[49] D. F. Williamson, R. A. Parker, and J. S. Kendrick, “The box plot: a simple visual method to interpret data,” *Annals of internal medicine*, vol. 110, no. 11, pp. 916–921, 1989.

[50] J. D. Gibbons, “Location tests for single and paired samples (sign test and wilcoxon signed rank test),” 1993.

[51] M. Irshad, “Automation scripts to replicate ircov,” https://github.com/MohsinIr84/replicationStudy/, 2021.

[52] N. M. Minhas and M. Irshad, “Data set used in the replication of an ir based test case prioritization techniques (ircov),” https://data.mendeley.com/drafts/ccnzpxng54, 2021.

[53] R. Just, F. Schweiggert, and G. M. Kapfhammer, “Major: An efficient and extensible tool for mutation analysis in a java compiler,” in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 612–615.

[54] R. Just, “The major mutation framework: Efficient and scalable mutation analysis for java,” in *Proceedings of the 2014 international symposium on software testing and analysis*, 2014, pp. 433–436.

[55] T. Roelleke, “Information retrieval models: Foundations and relationships,” *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 5, no. 3, pp. 1–163, 2013.

[56] G. Amati, *Information Retrieval Models*. Springer, New York, NY, 2009, pp. 1523–1528. [Online]. Available: https://doi.org/10.1007/978-1-4614-8265-9_916

[57] M.-M. Pittelkow, R. Hoekstra, J. Karsten, and D. van Ravenzwaaij, “Replication target selection in clinical psychology: A bayesian and qualitative reevaluation.” *Clinical Psychology: Science and Practice*, vol. 28, no. 2, p. 210, 2021.

[58] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, “Test case selection and prioritization using machine learning: a systematic literature review,” *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–43, 2022.

[59] M. Ivarsson and T. Gorschek, “A method for evaluating rigor and industrial relevance of technology evaluations,” *Empirical Software Engineering*, vol. 16, no. 3, pp. 365–395, 2011.

Chapter 7

Checklists to support decision making in regression testing

7.1 Introduction

Practitioners working in large-scale software development face many challenges in their regression testing activities [20]. They have to choose between full regression testing (re-test all) and selective regression testing. Furthermore, in selective regression testing, test case selection is a complex decision-making activity [20, 31]. There are various factors that testers need to consider while selecting a subset of tests from a large test suite [24]. In selective regression testing, the goal is to maximize the coverage and fault detection ratio with a selected regression test set. While releasing a product, practitioners want to control the fault slippage and to be confident that they have tested enough, there are no critical faults in the release, and they have achieved the desired quality [23, 24, 32, 33]. Achieving these goals requires support from organizational testing process. During the analysis of various embedded systems and windows application-based projects, Kasoju et al. [1] reported a lack of a structured testing process in organizations.

In our recent industrial studies [23, 24], we have identified the goals and challenges of regression testing in large-scale embedded software development companies. The identified challenges could be divided into two groups; process related challenges and technical challenges. The primary cause of the identified challenges is the absence of a well structured regression testing process. Instead of having a structured testing process for various testing activities, practitioners rely on expert judgment [20, 24].

However, evaluating or making a judgment without a structured mechanism symbolizes adhocism and may negatively impact the outcomes [4]. There is a possibility that even the experienced practitioners may overlook some essential aspects while making assessments and judgments [4]. Similarly, it is hard for the new team members to make good guesses with limited knowledge about the product. Especially on what activities need to be considered before the start of regression testing and when they can decide to stop regression testing. To complement the expert judgments, checklist-based guidelines co-designed with expert practitioners could be a step towards potential solutions. Such checklists could help document and reuse the best testing practices and help cope with various regression testing challenges that practitioners face. It would be easy for the team members to grasp the organizational testing policies/activities quickly in such a case. They can benefit from checklists and become familiar with usual team practices. Without a checklist, it is highly likely that a new practitioner, for instance, can omit a necessary test or violate any team policy [7]. However, it is essential that checklists should be designed based on practitioners' needs and should be aligned with the teams' current practices. Checklists prepared without involving practitioners could be misused or ignored [10].

This study aims to assist practitioners in improving regression testing by

1. identifying regression testing activities considered essential in practice and
2. introducing regression testing checklists based on essential activities.

The proposed checklists will help the managers to assess the readiness of team/team members. The practitioners will use checklists to keep track of essential activities, and they would not miss any necessary steps while performing regression testing. We worked with 25 senior testing practitioners of 12 companies to identify the essential regression testing activities and to design the regression testing checklists. Later, 23 practitioners of 10 companies participated in the evolution and evaluation of the proposed checklists.

The organization of the rest of the paper is as follows: Section 7.2 provides a brief introduction to checklists in software engineering practice and research. Along with the research questions, Section 7.3 provides a summary of methods opted in this study. Threats to the validity of this study and mitigation strategies to minimize the threats are discussed in Section 7.4. Section 7.5 presents the findings of this study, Section 7.6 provides the discussion on the process and outcomes of this study, and Section 7.7 concludes the study.

7.2 Background and related work

7.2.1 Significance of checklists

Practitioners of various disciplines use checklists as a cognitive aid to ensure the correct completion of any task.

> *"If the knowledge exists and is not applied correctly, it is difficult not to be infuriated."**(Gawande [34] The Checklist Manifesto)***

A checklist is a standardized tool that enlists the required process criteria for the practitioners performing a specific activity. It provides support in recording the presence or absence of the essential process tasks [12].

"Checklists seem able to defend anyone, even the experienced, against failure in many more tasks than we realized. They provide a cognitive net. They catch mental flaws inherent in all of us—flaws of memory and attention and thoroughness." ***(Gawande [34] The Checklist Manifesto)***

Two popular uses of checklists are, using checklists as mnemonic systems or as evaluation tools. The first is used as a reminder system to help practitioners avoid omitting any essential task. It also assures that practitioners follow the organizational framework and utilize best practices. Such checklists help minimize human error and improve overall performance. In contrast, the evaluative checklists can aid in the standardization of evaluation by providing assessment guidelines and ultimately improving the evaluation process 's credibility [12].

Using a checklist to document any process is not a new concept. For example, in the aviation industry since the 1930s, it has been a standard operating procedure for the pilots and other aviators to use checklists [5]. Pilots are using the checklists before, during, and after the flight [11]. In medicine, checklists are used as a decision aid to identify a medical condition and decide on an appropriate course of treatment. In comparison, surgical checklists are recommended as a safety measure to reduce the margin of human error, and any adverse effects during surgery [13].

Social and behavioral scientists are using self-reporting questionnaires as an assessment mechanism. Usually, such questionnaires include checklist items that enable the goal-based assessment of a phenomenon [6].

Software engineers are using checklists in various tasks, including the audit of requirement/design specifications and code inspection [4]. Checklists can help make a process repeatable, and the practitioners can use various checklists in the software development life cycle. For instance, they can use release checklists to assure that no essential steps are skipped. At the start, a checklist does not have to be exhaustive. If some items are missing, we can add the missing or new items later. Improvising a checklist is always helpful in adding future goals [7]. In the subsequent sections, we present some related work regarding checklists in software engineering research and practice.

7.2.2 Use of checklists in software engineering

Perry [2] provided generic checklists to aid software testing teams in different phases of their work. The author does not provide any checklist specific to regression testing. However, the checklists presented in this book can be taken as inspiration to introduce any testing-related checklists.

Based on their experience with a project, Heroux and Willenbring [7] advocate using checklists to make processes repeatable. They suggest using checklists at various stages of a project. For instance, release checklists, developer checklists, and commit checklists. The authors stressed that checklists help practitioners remember essential but easily omitted steps. The checklists can help new team members to get familiar with the team practices. Heroux and

Willenbring suggest that starting with simple checklists and improving them through iterations will help improve the related processes.

Brykczynski [8] surveyed 117 software inspection checklists from 24 sources. The author suggests that checklists help the reviewers in a software inspection process by providing recommendations to find the defects. Brykczynski further recommends that checklists should be updated regularly. These should not be longer than a page and should be based on relevant items based on questions. The author classified the existing checklists according to their application type, including requirements, design, code, testing, documentation, and process.

Brito and Dias-Neto [9] conducted empirical studies to evaluate their checklist based technique (TestCheck). TestCheck is used to inspect the software testing artefacts. TestCheck consists of three separate checklists for assessing the test plan, test case, and test procedure. The authors aimed to evaluate and improve their technique through a series of evaluations in this study. They believe that their approach needs to undergo the evaluation process further, and there is a need to provide the tool support for the smooth transition of the technique to the industry.

Usman et al. [4] proposed checklists to improve the software effort estimation. The authors revealed that expert judgment is the most common practice for effort estimation that could lead to wrong estimates in the absence of any process support. Usman et al. proposed a process to develop and evolve the estimation checklists. They started with understanding the current estimation process and identifying the relevant checklist factors from the existing literature. The authors developed and improved the checklist in different iterations based on the findings. They validated the proposed checklist in two steps (i.e., statically (a trial use) and dynamically (a real use). The authors claimed that the use of checklists could increase practitioners' confidence in their estimates.

Madaio et al. [10] designed a checklist to understand organizational challenges and opportunities around fairness in AI. The authors conducted semi-structured interviews and co-design workshops with 48 practitioners of 12 companies to understand the needs and concerns of practitioners and develop the AI fairness checklists. Madaio et al. concluded that checklists could help formalize the ad-hoc process and empower individual advocates. They were hopeful that their proposal could support the practitioners in addressing AI ethics issues. Although Madaio et al. [10] did not evaluate their checklist, however, in future, they are planning to conduct pilot studies with different teams.

Petersen et al. [3] proposed a context checklist for industrial software engineering research and practice. They have listed three primary purposes of the proposed checklist, i) to help record the experience in projects in an industrial setting, ii) to help decide between the use of past decisions vs experience and knowledge, and iii) to support researchers in deciding the contextual information to report in primary studies and information to extract in secondary studies. The authors evaluated the proposed checklists with the practitioners and researchers using interviews and questionnaires. Based on the feedback, Petersen et al. revised the checklists to overcome the deficiencies identified by the practitioners and researchers.

Molléri et al. [16] presented a checklist to support the survey research in software engineering. Using 12 methodological studies, the authors identified stages and recommended practices of the survey process. They used thematic analysis and vote-counting methods to aggregate knowledge from the existing selected studies. The authors evaluated the checklist by applying it to the existing surveys and analyzing the results. Later the authors collected the experts' feedback on the proposed checklist and improved the checklist.

Höst and Runeson [14] proposed two separate checklists to support the software engineering researchers and reviewers for conducting and reviewing case studies. The authors conducted a literature survey to identify the existing checklists in the first stage. They merged all the checklists found in the literature into a single checklist and classified the items according to different case study phases. Later, the authors reduced the size of the checklist by grouping similar items. After validating the checklist with the PhD students, the authors updated it to accommodate the validation feedback.

Kitchenham et al. [15] merged two checklists into a single checklist to evaluate the quality of software engineering experiments. The authors constructed the checklist using the findings of two studies [17, 18]. The authors performed a two-step validation of their proposed checklist by applying the checklist criteria to the selected papers from human-centric software engineering experiments.

7.2.3 Summary

From the review of the related literature, we learned that use of checklists in software engineering research and practice is evident. The checklists in software engineering research are used to assess the quality of adopted empirical research and provide guidelines to conduct empirical studies systematically. Software engineering professionals use checklists to assess activities, including effort estimation, code reviews, and testing. However, to the best of our knowledge, no checklists are available explicitly designed for regression testing. We are introducing regression testing checklists to support practitioners in decision-making and keeping track of essential regression testing activities. We considered industry context while designing the checklists and designed them with the input of senior testing practitioners. The proposed checklists will open a way forward for assessing and improving regression testing.

7.3 Methodology

The primary objective of this study is to support practitioners in structuring the regression testing process by introducing regression testing checklists. We followed an iterative approach to design, evolve, and evaluate the regression testing checklists. We intended to find the answer to the following **research questions** to fulfil the study's objective.

RQ1 What activities do practitioners consider while planning, performing and assessing regression testing?

RQ2 What checklists and checklist items can be helpful for practitioners while planning, performing and assessing regression testing?

RQ3 What is the perspective of practitioners about the proposed checklists?

7.3.1 Research approach

Figure 7.1 presents the overview of the approach we followed to design and evolve the regression testing checklists. We involved the practitioners in our process, from checklists' activities identification to checklists' verification. We conducted individual and group interviews with senior testing practitioners to identify the activities that should be considered during the overall regression testing process. Based on the input from practitioners, we mapped the practitioners' decisions with the goals and respective activities. This mapping provided us with a basis for the regression testing checklists. We handed over the initial draft of checklists to the practitioners and asked them to assess each checklist item to see if the item is relevant. Based on their feedback, we made improvements to the checklists. Finally, the industry practitioners evaluated the proposed checklists. They assessed checklists concerning comprehensiveness, usefulness, and relevance to their context.

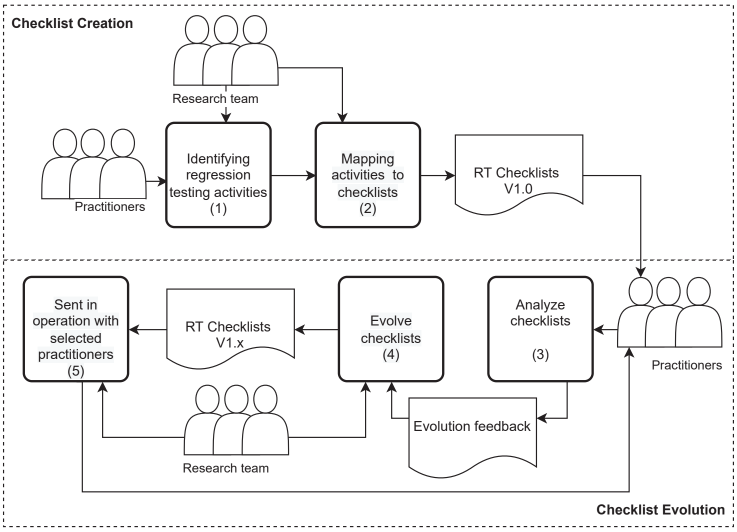

Figure 7.1: Overview of the approach used to design and evolve the regression testing (RT) checklists.

7.3.2 Selection of participants

We followed a snowball sampling approach [25] to select the participants for our study. Since we were interested in recruiting senior testing practitioners, we imposed a constraint that participating practitioners must have five years or more of experience in software testing.

We contacted seven senior testing practitioners from three large Swedish companies, two of whom had already participated in our previous studies on regression testing [22–24]. We received responses from five practitioners. We conducted introductory workshops with them to present the study idea and to finalize the operational aspects of the study. Also, we requested them to provide us with the contacts of senior testing practitioners from their companies or contact network. All five practitioners consented to participate in the study and introduced us to a few more practitioners who fulfilled the criteria of the required experience. Along with contacting the practitioners from the companies we had already worked with, we also sent various requests through Linked In. We conducted introductory workshops with willing practitioners, one workshop for each company regardless of the number of participants. We continued with this approach and stopped after getting the consent of 25 senior testing practitioners from 12 companies.

7.3.3 Study participants

Our goal was to suggest checklists that could be useful and fit in the industry context. We designed the checklists in an iterative process and involved the industry participants in all phases (i.e., checklist design, evolution, and evaluation). Our findings represent the perspective of senior testing practitioners. The participants' experience ranges from five to twenty-three years, and their average experience is twelve years. The participants' organizational roles are test engineer, senior test engineer, test lead, QA manager, senior SQA manager, test architect, and head QA unit (see Table 7.1).

Further, most participants work in large-scale environments. The size of a company can be classified based on the number of professionals working in it, and the volume of projects [40]. A large company will have more than 250 practitioners working [39]. In our sample, only two companies, C6 and C8, have less than 250 practitioners working. These companies use agile methodologies, including scrum, CI/CD, and DevOps. The domains represented in our study are financial, banking, healthcare, transport, surveillance and security, telecommunication, AI solutions, and security systems. For more detail on contextual information of the companies represented by the participants, please see Table 7.2.

7.3.4 Data collection

We used multiple data collection methods to introduce our research goals to the prospective participants, collect data on regression testing practice, evolve checklists, and verify the checklists.

Introductory workshops

Our study required to involve senior testing practitioners in all phases, and we needed a long-term commitment from them. We conducted 12 online workshops to introduce our research idea of working on regression testing checklists to prospective participants. The example of the content used in these workshops is presented in Appendix C.2. Apart from the short introduction, the

Table 7.1: List of practitioners who participated in checklist design and evolution.

Participant ID	Role	Experience in years
P1	Manager SQA	15
P2	Test Lead	10
P3	Test Lead	11
P4	Senior Test Lead	12
P5	Test Lead	10
P6	Senior Manager QA	17
P7	Test Architect	8
P8	Test Lead	23
P9	Tech Lead	16
P10	Senior Test Engineer	20
P11	Senior SQA Engineer	11
P12	SQA Eengineer	6
P13	Senior SQA Engineer	9
P14	Test Lead	11
P15	Senior SQA Engineer	7
P16	Test Manager	9
P17	Test Engineer	8
P18	Test Engineer	6
P19	Test Manager	15
P20	Test Engineer	5
P21	Test Engineer	6
P22	Test Lead	12
P23	QA Lead	22
P24	Head SW QA	20
P25	Test Engineer	15

workshops were mainly Q/A-based discussions. These workshops aimed to ensure senior testing practitioners' informed and consented participation.

Interviews (checklists creation)

In the second phase, we conducted seven individual and five group interviews to collect data on regression testing practice. The average duration for individual interviews was 60 minutes, and for group interviews, it was 75 minutes. We followed the guidelines by Runeson and Höst [26] while designing the interview questionnaire and conducting the interviews. To create the regression testing checklists, we were primarily interested to know the activities considered essential for regression testing in the companies. In this regard, we collected the opinion of senior testing practitioners using semi-structured interviews. In semi-structured interviews, the investigators do not need to follow a fixed pattern (e.g., order of questions) with every participant, and by following the discussion, the investigators can improvise the interview questions [26]. The interview questions were open-ended, and we improvised them, given the participants' context.

Table 7.2: Contextual information of the companies represented by the participants (size classification: *"small < 50, large > 250"* [39]).

CID	Employees	Size	Leading Participant	Product domain	Approach	Test Team Size	Participants
C1	3000	Large	Manager IT/SQA	Financial	Agile	18	P1, P2, P3
C2	500	Large	Senior QA Lead	Financial	Agile, Scrum	18	P4, P5, P6
C3	8000	Large	Test lead	Transport	Agile, DevOps	20	P7
C4	4000	Large	Tech lead	Surveillance / Security	Agile	17	P8
C5	10000	Large	Senior Tester	Charging System	Agile, DevOPs	50	P10
C6	150	Medium	Senior SQA Engineer	Healthcare	Agile, Scrum	10	P11, P12, P13, P14
C7	5000	Large	Senior SQA Engineer	Telecom	Agile, CI-CD Pipelines	10	P15, P16, P17, P18
C8	200	Medium	Test Manager	Telecom	Agile, DevOps	10	P19, P20, P21
C9	3000	Large	Test Lead	AI Solutions	Agile	75	P22
C10	20000	Large	Test Manager	Security Systems	Agile	10	P23
C11	13000	Large	Head SQA	Banking	Agile	25	P24
C12	1000	Large	Test Engineer	Hardware & Software	Agile, Scrum	12	P25

During these interviews, we asked questions to understand the regression testing practice in the companies, for example, what regression testing activities are considered essential, and what stopping criteria the testing practitioners use. The detailed interview questionnaire is available in Appendix C.3. To ensure the quality of the interview instrument, we underwent expert reviews. A senior practitioner (reviewer 1) with ten years of research and development experience in software testing and an academic researcher (reviewer 2) with sixteen years experience in software engineering research evaluated the interview instrument. Reviewer 1 agreed with the instrument and did not suggest adding or updating anything in the instrument. However, reviewer 2, who also has experience developing checklists, suggested some changes, including adding a question about the product domain and changing the phrasing of Question 10. The interview instrument in Appendix C.3 represents the version after incorporating reviewers' feedback.

Workshops (checklists evolution)

Workshops provide a practical approach to designing, evaluating, or co-creating any artefact of interest [36]. We conducted online workshops for *checklists evolution*. We designed an evolution tool to evolve the checklists generated based on interview results. Along with the initial draft of checklists, we shared the evolution tool with the practitioners who participated in the study's second phase (i.e., interviews). The first author participated in five workshops conducted with the participants of companies C1, C2, C6, C7, and C8. The participants from C3, C4, C9, C11, and C12 conducted the evolution of the checklists on their own. The participants from C5 and C10 could not participate in the evolution phase.

The participants reflected on each item of the checklist, and for every item, they rated the checklist items based on the question given below.

- Is the checklist item relevant? Yes, no, don't know

If practitioners considered the checklist item to be relevant, they chose "yes", if they did not find it relevant, the option chosen would be "no". If they were indecisive, they reported it as "Do not know". We asked the participants to suggest additional items if they thought we had missed any. In case of suggesting a new item, the practitioners needed to specify the checklist for which item was proposed, and they could also provide additional comments to motivate their suggestion. The tool for evolving checklists is described in Appendix C.4.

Survey (checklists evaluation)

After having the checklists finalized, we requested the practitioners to provide us their feedback based on checklists' trial run and team discussion. We used an online survey for *checklists evaluation*. To design the survey questionnaire, we followed the guidelines by Kitchenham and Pfleeger [28]. We created evaluation questionnaire using Google forms and sent it to all the participants. The evaluation questionnaire was created on the following parameters:

- comprehensiveness,
- usefulness,
- customizability, and
- adoptability

We also added a question to ask if the participants' companies are willing to use the checklists. The complete evaluation questionnaire is provided in Appendix C.5.

7.3.5 Data analysis

Detailed analysis was required for the qualitative data, whereas summaries and graphs were required for the data collected during the checklist evolution workshops and survey (i.e., evaluation of checklists). We followed thematic analysis to analyze the qualitative data and took inspiration from the methods described in [37, 38]. Figure 7.2 presents the detail of the steps carried out for the data analysis, whereas the steps are outlined in the following subsections.

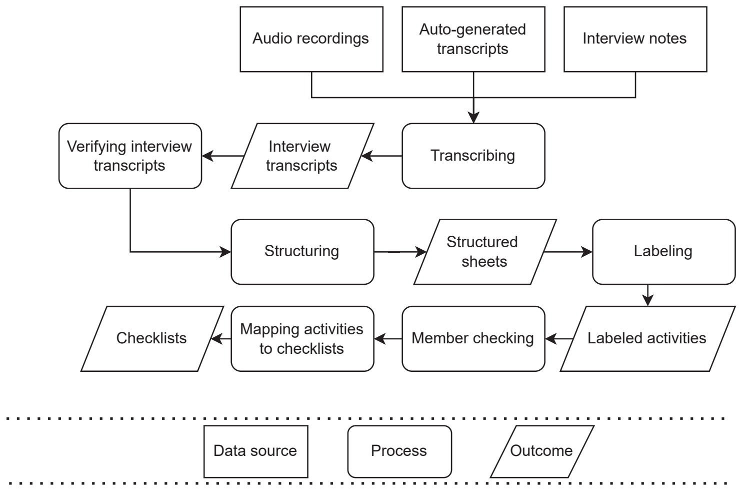

Figure 7.2: Data analysis steps.

Transcribing

All interviews were conducted using online meeting tools and recorded with the participants' consent. During the interviews, we enabled the automated transcription facility provided by the online meeting tool, and one note taker took notes. We finalized the interview transcripts using auto-generated transcripts, interview notes, and audio recordings. We ensured to transcribe participants verbatim to avoid any bias and misinterpretation. On average, we invested four hours in transcribing an interview. Later, with the help of an independent volunteer, the first author verified the transcripts by comparing them with sources to ensure we did not miss any vital information.

Structuring

Lacey [38] suggests organizing the data into easily retrievable sections after finalizing the transcripts. We converted the transcripts into a structured excel sheet (see Table 7.3). We extracted the information under column headings which were derived from interview questions. However, finding desired information against the relevant questions was not straightforward. We often needed to scan the answers to multiple questions to find the relevant information. This step helped us familiarize ourselves with the data. We used colour coding to differentiate between emerging themes and to add more clarity.

Table 7.3: Structure of transcription sheet – RT (regression testing).

Heading	Description
CID	ID for the Participants' companies.
State of RT	How regression testing is performed in the case companies?
Significance of RT	How significant RT is for the case companies?
Frequency	Frequency of releases.
Significant activities	RT activities that are significant for the participating companies.
Essential aspects before RT	The aspects that practitioners suggest to consider before the start of RT.
Essential aspects after RT	The aspects that practitioners suggest to consider after RT.
RT goals	The goals that practitioners set for RT.
RT CL	If the participants already using any checklists for RT?
CL Usefulness	The perception of practitioners about the usefulness of prospective RT checklists.

Labeling

We have reported participants' original statements regarding the state of regression testing practice and essential activities in Appendix C.1 (Table C.1). After having interview results in a structured form with appropriate colour codes the next step was to have appropriate labels for similar themes. We used labels for the activities considered essential in the companies before and after regression testing. We followed the philosophy of axial coding while assigning the labels to activities [35] and assigned appropriate labels by grouping similar statements. For example, we grouped the following three statements: i. selecting a smaller but effective subset of test cases, ii. selecting the right test cases, and iii. good knowledge of test cases helps in selecting right test cases. We labelled the mentioned group of statements as *"selection of right test cases"*. In assigning a label to a group of similar statements, we ensured that we used a label that reflected the perspective of the practitioners involved. We continued this exercise until we grouped all similar statements and labelled appropriately.

Member checking

Along with the participants' original statements regarding the state of regression testing practice and essential activities, we sent the labelled activities to the participants. We asked them to verify if we had interpreted their perspectives correctly. However, we did not receive any corrections from them. Maybe this was because we did not deviate from participants' statements while defining the labels for the activities.

Mapping

Once having the lists of activities considered essential in the companies, the next step was to create checklists' items based on these activities. We took inspiration from [2] while creating checklists' items from the activities identified in step 3 (labelling). We divided activities into

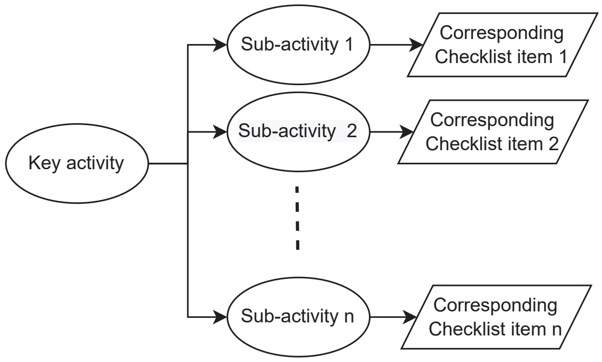
Key activity
Sub-activity 1
Sub-activity 2
Sub-activity n
Corresponding Checklist item 1
Corresponding Checklist item 2
Corresponding Checklist item n

Figure 7.3: Mapping regression testing activities to checklist items

sub-activities (where possible) and then mapped the sub-activities to the checklist items. Figure 7.3 presents the procedure of how regression testing activities mapped to the checklist items. For example, the regression testing activity labeled in the previous step *selection of right test cases* was divided into three sub-activities 1) Identifying test cases related to changes, and 2) Identifying test cases related to impacted modules. Later these activities were mapped to the following checklist items.

- Have the test cases associated to changed parts been identified?
- Have the test cases associated to impacted module been identified?

7.4 Validity threats

This study's results are based on industry practitioners' experiences and perceptions. We opted for workshops, semi-structured interviews, and online questionnaires as data collection methods. There are various aspects related to this study that can pose threats to its validity. In the following, we have discussed potential validity threats and the strategies we opted to mitigate these threats. In our discussion of threats to validity, we follow the guidelines by Runeson and Höst [26] and Wohlin et al. [27].

Construct validity: This aspect of validity could be associated with the choice of treatment for the study and its expected outcomes. In our study, we can link it to the creating the data collection instruments and the process of selecting participants. We used well-known guidelines and followed the established procedures to minimize this threat to validity. While designing the data collection instruments, we followed the guidelines by Runeson and Höst [26] for the interview questionnaire, and Kitchenham and Pfleeger [28] for the design of survey questionnaires. To avoid inconsistency and bias, we involved a senior researcher and a practitioner to review the instruments. Based on their feedback, we augmented our data collection instruments. Concerning the selection of participants, we used snowball sampling to reach the senior testing practitioners. Another threat could be the validity of our study's outcome (i.e. regression testing checklists). We went through a systematic procedure to design regression testing checklists (see Section 7.3.1).

Internal validity: This aspect of validity is crucial if causal relationships are examined in the study. Generally, internal validity refers to the study's credibility concerning underlying data collection, interpretation, and analysis methods since these can impact the validity of obtained results. Here are some of the measures that we took to mitigate threats to internal validity of our study. We selected the interview participants based on their experience and interest in the regression testing. Concerning data collection, with the prior consent of the participants, we recorded all interviews. We generated the structured transcription sheets using recordings, interview transcripts, and notes taken during the interviews. We also validated our interpretations from the interview participants. For the survey questionnaire, we used multiple-choice questions. To avoid the researchers' bias, we also provided the option for the free-text response in the evolution instrument (see Appendix C.4).

External validity: The external validity threats refer to the concept of generalization of the results. Although, we did not claim the generalizability of our checklists. However, the similarity in views of practitioners from 12 different companies working on diversified domains indicates the possible generalizability of the proposed checklists. We have provided the contextual information of the participants' companies. We have also provided the detail of data collection instruments (See Appendices C.3, C.4, & C.5). This may help generalize the context and replicate the study in future.

Reliability : This aspect concerns the extent to which the data and analysis depend on the specific researchers. The results are reliable if they are free of biases, and independent researchers can reproduce them using similar methods. We took various measures to minimize the threats to the reliability of the study. We have explained all aspects of data collection, analysis, and reporting in Section 7.3. Data collection instruments are also available in the study. Further, the study participants reviewed and validated all the results generated in this study.

7.5 Results

7.5.1 State of regression testing practice in companies

Table C.1 (Appendix C.1) provides a summary of how regression testing (RT) is performed in the participants' companies. Eleven of 12 participating companies consider regression testing an unavoidable activity. Participants from one company said that they use regression testing based on the nature of changes. They only choose to perform regression testing if changes are in critical areas. They skip regression testing for simple changes and perform exploratory testing. Participants of three companies (C3, C10, & C12) said that they perform exploratory testing after regression testing to ensure all risk areas are working correctly. Participants from C7 said that they use smoke testing before regression testing to check if the build is stable. The frequency of regression testing varies among the companies, and it mainly depends on the domain and criticality of the product/module under test. Testing practitioners set regression testing goals, 10 of 12 do it informally, and only two have a defined mechanism for setting and assessing the goals. The participant from C4 stated that *"Regression testing goals are significant for our organization. We take these as a challenge and are continuously working on them. The test team can decide to stop regression testing once goals are achieved."*. All companies' participants said that test managers/leads decide to stop regression testing based on their judgement. The considered information includes all test cases in regression suite executed, pass/fail ratio, pass rate above threshold (e.g. 90% or above), and bug severity. Participants from C2 decide to stop regression testing when fewer bugs appear or they witness bug repetition. Three companies (C3, C5, C9) have transitioned from manual to fully automated regression testing and introduced the CI/CD pipelines. The scope of regression testing is defined based on the changes and their impact. In all companies, regression testing is performed with a selected set of test cases whenever a change occurs (adding a new feature or fixing a bug). Near the release, they prefer to run the complete regression suite. The participant from C11 revealed that their regression suites are huge, and running all tests is costly. To cope with the cost, thet are experimenting with running all tests for the most commonly used features instead of running all tests in the regression suite – the participant suggested a slogan for it "running all those tests that matter". Concerning the regression test plan, the majority consider it as a part of the test plan, and they do not have a separate regression test plan. In three companies (C7, C8, C10), the regression plan is part of the sprint planning meeting. During every sprint, they take the essential decisions about regression testing. For example, what are the new fixes (tickets), what do they have to test, and how much they should test?

7.5.2 Regression testing activities (RQ1)

Table 7.4 outlines the activities considered essential for regression testing in the participating companies. Practitioners consider selecting the right test cases (e.g., more coverage with fewer test cases) as a key to their success. This could only be possible if they have good domain knowledge, understand the system specifications, and know the changes and their impact. To understand the impact of changes, the practitioners need to know the dependencies among the modules/subsystems. All changes must have been tested sufficiently, and all tickets/changes must be checked in (code freeze) before the start of regression testing. Availability of the required test environment and data are also essential to start the regression testing. In the end, the practitioners ensure to run the planned regression tests completely. They generate the test reports, analyze the results, and decide subsequent actions based on the test results. They look at the pass vs failure test cases and decide to release the product if the pass percentage is above the defined threshold. They can decide to release the product with the medium severity bugs, and fixing bugs will be part of the next release. However, in case of severe faults, they must stop the release. In many cases, the teams decide on the goals before the start of regression testing, and they assess the achievement of their goals after finishing the regression testing activity. A complete and company-wise summary of regression testing activities considered essential in the participants' companies is presented Table C.2 (Appendix C.1).

Table 7.4: Activities considered essential for regression testing (RT) in the companies.

A#	Activities before RT	Companies
1.	Acquiring domain knowledge	C1, C2, C5, C6, C7, C8, C9, C11, C12
2.	Ensuring communication of changes	C1, C3, C5, C6, C12
3.	Knowing new features/Changes	C3, C4, C5, C6, C7, C8, C9,C10, C12
4.	Ensure that changes are freezed	C3, C6, C10, C11, C12
5.	Ensure changes have been tested	C2, C4, C7, C9, C10, C12
7.	Identifying impact of changes	C1, C2, C4, C5, C6, C7, C9, C10, C12
8.	RT scope is decided	C1, C2, C4, C8, C12
9.	Selection of right test cases	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12
10.	Having required test environment	C10, C11
11.	Organizing test data	C2, C6, C7, C8, C10, C11, C12
12.	Defining regression testing goals	C1, C2, C3, C4, C12
13.	Making regression test plan	C7, C8, C10
14.	Assigning responsibilities	C2,C8
15.	Test suite maintenance	C1, C6, C7, C9, C11
A#	**Activities after RT**	**Companies**
1.	Having the planned regression test suite executed	C1, C3, C5, C6, C8, C9, C10, C12
2.	Creating test reports	C2, C6, C7, C10, C11, C12
3.	Analyzing test results	C1, C2, C3, C5, C6, C7, C8, C9, C10, C11, C12
4.	Assessing goals achievement	C1, C3, C4, C5, C6, C7, C9, C11
5.	Assessing ratio of pass vs fail	C1, C4, C5, C6, C7, C9, C11, C12
6.	Ensure pass percentage is above threshold	C2, C5, C6, C7, C9, C11, C12
7.	Critical bugs have been identified and resolved	C1, C2, C3, C4, C8, C10, C11, C12

7.5.3 Practitioners' opinion on prospective regression testing checklists (RQ2)

During the interviews, we asked the practitioners' opinions on the regression testing checklists. Most participants were convinced about the usefulness of regression testing checklists, provided these checklists cover essential aspects only. They pointed out that checklists can help add structure to regression testing practice, and practitioners will not skip any essential step while performing regression testing.

> *Checklists can help in adding formalism to practice. Although we have a well-managed plan, there are still things that we miss. If we have a small checklist that can add value.**(Senior SQA Engineer)***
>
> *A guideline could be an asset for any tester that can help him do essential things before regression testing. It will reduce the impact of team members leaving and new members being added. **(Test Engineer)***
>
> *It will help streamline the practice. However, it should be a short checklist not to hinder the job. We would be interested in using the proposed checklists. **(Head QA unit)***

Some form of regression testing checklists are in place in a couple of companies (e.g., C3, C4). However, these checklists are application-specific.

> *We are already using the checklists, but our checklists are product-specific. During my experience, I have worked with waterfall, agile, and now DevOps. In waterfall, we had too many checklists, but in our current environment, we have only essential checklists. I think checklists are helpful since they help the practitioners not forget to do any essential activity. **(Test Lead)***

The participants from these companies voiced the usefulness of generic checklists that can guide the practitioners to stick to the plan and not miss any essential steps. Some of the participants highlighted that they do not use any formal checklists. However, they informally follow the lists of essential items. For example, at C2, senior test leaders informally assess the readiness of their team members by asking random questions.

We are doing something similar to checklists informally, but we are not using any pre-defined checklists. We assign regression testing of different modules to the practitioners based on their relevant knowledge of modules. So informally, we gauge the readiness of the team members. However, we do it using our first-hand knowledge. Being an experienced manager, I know the skills of my team member. When we induct a new member, we provide a chance to get on board and help him gain the domain knowledge. We check the readiness through informal chats. ***(Senior Manager QA)***

Some participants were reluctant to give their opinions before seeing the actual checklists. For example:

I would like to see what checklists emerge, and then I will decide about their usefulness. If it is helpful for my context, then it is useful. Generally, I agree that checklists are a useful thing for any environment. ***(Senior Tester)***

While responding to our question about the types of prospective checklists, most practitioners suggested checklists to assist practitioners before and after regression testing. There was a divided opinion on the checklists types before regression testing. Some participants suggested using a single checklist, and the others proposed having two checklists (one for individual testers and another for team activities) before regression testing. Table 7.5 provides a summary of practitioners' opinion about checklists types.

Table 7.5: Types of regression testing (RT) checklists suggested by participants.

Checklist type	Suggested By
Checklists to track the activities before regression testing (Individual)	C1, C4, C6, C7, C9, C10, C12
Checklists to track the activities before regression testing (Team)	C1, C4, C6, C7, C9, C10, C12
Checklists to track the activities before regression testing (Combined)	C2, C3, C8, C11
Checklists to track the activities after regression testing (Exit criteria)	C1, C2, C3, C4, C7, C8, C11, C12

RT Checklists creation and evolution

Checklists creation: Based on the input form the participants about the checklist types presented in Table 7.5, we decided to opt for the following three checklists.

1. Checklist to track the activities before regression testing (Individual)
2. Checklist to track the activities before regression testing (Team)
3. Checklist to track the activities after regression testing (Exit criteria)

To decide the checklists' items we considered the regression testing activities identified by practitioners (See Table 7.4). For each activity we created relevant checklist items and mapped the individual items to the respective checklists, in the result of this exercises we created the checklists presented in Tables 7.6, 7.7. & 7.8. To check the readiness of the teams members, test manager can ask them to fill the checklist provided in Table 7.6. Later, to ensure team 's readiness, after collecting the individual checklists test manager can fill the checklist presented in Table 7.7. Finally, while stopping regression tests, test manager together with team members can fill in the checklist presented in Table 7.8. In every checklist table, we have provided two additional columns "status" and "comments". Using the status column, the stakeholders can report the status concerning the checklist item. For example, for the checklist item *" Are you aware of dependencies among the subsystems?"*, the concerned stakeholder can fill in *"Yes I am aware"*, *"Yes, but not 100%"*, *''No, it is not applicable"*. In the comments column, the stakeholders can further explain the status. For example, if a stakeholder chooses *"Yes, but not 100%"*, the further explanation can be added in the comments column as *"I am in the phase of acquiring knowledge of dependencies"*.

Table 7.6: Checklist to assess the readiness of testers to be filled by test team members (CL_1 – $V_{1.0}$).

CLI	Checklist item	Status	Comments
1	Are you aware of the team 's regression testing goals?		
2	Do you have essential knowledge of system specifications?		
3	Are you aware of dependencies among the subsystems?		
4	Are you aware of new changes in the system?		
5	Have you analyzed the impact of changes on the unchanged parts of the system?		
6	Are you confident of performing regression testing on your own?		
7	Have you been trained for the tools used for regression testing within the team/organization?		
8	Are you aware of the criticality of the subsystems to be tested?		
9	Do you have access to test data?		

Checklists evolution During the checklists' evolution phase, the participants were asked to give their opinion on the relevance of checklists and the individual items. We shared the initial draft of checklists and evolution forms with the practitioners who participated in the study's first phase. The practitioners evaluated checklist items and provided their opinion.

Tables 7.9, 7.10, and 7.11 presents the summary of feedback from the study participants. Concerning the relevance of all three checklists, we received "Yes" from all participants. However, on the individual items, a few participants chose the options of "No", and "Do not know".

Table 7.7: Checklist to determine the team 's readiness to be filled by test manger (CL_2 – $V_{1.0}$).

CLI	Checklist item	Status	Comments
1	Have the regression testing goals been defined?		
2	Are the test team members aware of system specifications?		
3	All the changes been checked in?		
4	Have the changes been communicated to the test team?		
5	Have the changes been tested in isolation?		
6	Has the change impact been determined?		
7	Is the regression test suite up to date?		
8	Have the test cases associated to changed parts been identified?		
9	Have the test cases associated to impacted module been identified?		
10	Has the regression testing scope been determined?		
11	Has the regression testing been incorporated in the test plan?		
12	Has the regression test plan been developed?		
13	Are the required resources available?		
14	Has the decision been taken between manual vs automated regression testing?		
15	Have clear responsibilities assigned to team members?		
16	Is the testing team agreed to start regression testing?		

Table 7.8: Checklist to determine exit criteria of regression testing to be filled by test manager together with team members (CL_3 – $V_{1.0}$).

CLI	Checklist item	Status	Comments
1	Have the regression testing test suites been executed completely?		
2	Has the pass rate of regression testing suites reached the threshold?		
3	Have all severe /critical defects been resolved?		
4	Have all medium severity defects been closed?		
5	Have all metrics been collected?		
6	Have defined regression testing goals been achieved?		
7	Is the test team agreed to test closure?		

For instance in the checklist presented in Table 7.9 the checklist item # 6 *"Are you confident of performing regression testing on your own?"* was rejected by four respondents, three chose "Do not know", and three considered this item relevant. Whereas most respondents considered all other items of this checklist relevant. In the checklist presented in Table 7.10 checklist item # 11, 12, & 16 received less recommendations. Here item # 11 was recommended by 5 respondents, three rejected it and two chose "Do not know". Item # 12 received six recommendations, two rejections, whereas two respondents chose "Do not know". Item # 16 received six recommendations, two rejections, and two respondents chose "Do not know". Most respondents recommended all items of the checklist presented in Table 7.11, except item # 4 & 5. One respondent rejected the item #4 and two chose "Do not know". For item #5 non of the respondents rejected it, however, four chose “Do not know”.

Based on the respondents’ feedback for the checklists’ version 1.1, we suggest removing item #6 from the checklist presented in Table 7.9. Also we suggest considering inclusion of item # 5 in Table 7.10 and item # 4 in Table 7.11 in the final version. We leave the choice to the practitioners about the usage of items # 11, 12, & 16 of the checklist presented in table 7.10 and # 5 of the checklist presented in Table 7.11.

Table 7.9: Evolution of checklist to know the readiness of testers to be filled by test team members ($CL_1 - V_{1.1}$).

CLI	Is the checklist item relevant?	Yes	No	Don’t Know
1	Are you aware of the team 's regression testing goals?	9	1	0
2	Do you have essential knowledge of system specifications?	9	0	1
3*	Are you aware of dependencies among the subsystems?	10	0	0
4*	Are you aware of new changes in the system?	10	0	0
5	Have you analyzed the impact of changes on the unchanged parts of the system?	8	1	0
6	Are you confident of performing regression testing on your own?	3	4	3
7	Have you been trained for the tools used for regression testing within the team/organization?	10	0	0
8	Are you aware of the criticality of the subsystems to be tested?	9	0	1
9	Do you have access to test data?	8	0	2

Suggestions by respondents: Test team lead of C4 has reflected on some of the items included the checklists (i.e., item # 3 & 4 in Table 7.9, item # 1 & 16 in Table 7.10). The participant voted yes for these items but argued that the inclusion of these items would depend upon the situation. In this regard, the participant provided an example of item #16 in Table 7.10 and suggested that the team should have to be agreed to start regression testing in most cases. Still, there could be exceptions in this regard. If needed for the project, the product owner can decide on an early start.

Senior QA lead of C2 suggested the following items be included in checklists (Table 7.11):

- Is QA sign-off document ready?
- Are the stakeholders agree on QA sign-off?

The test manager of C8 suggested the following items be included in the checklists (Table 7.10):

- Do we need to add new test cases in the regression suite?
- What is the trade-off between manual vs automated testing?
- Have we discussed the scope of regression testing in the sprint planning meeting?

Table 7.10: Evolution of checklist to know the readiness of test team to be filled by test manager ($CL_2 – V_{1.1}$).

CLI	Is the checklist item relevant?	Yes	No	Don't Know
1*	Have the regression testing goals been defined?	9	1	0
2	Are the test team members aware of system specifications?	9	0	1
3	All the changes have been checked in?	8	1	1
4	Have the changes been communicated to the test team?	9	0	1
5	Have the changes been tested in isolation?	7	1	2
6	Has the change impact been determined?	8	0	2
7	Is the regression test suite up to date?	9	0	1
8	Have the test cases associated with changed parts been identified?	10	0	0
9	Have the test cases associated with the impacted module been identified?	9	0	1
10	Has the regression testing scope been determined?	10	0	0
11	Has the regression testing been incorporated into the test plan?	5	3	2
12	Has the regression test plan been developed?	6	2	2
13	Are the required resources available?	9	0	1
14	Has the decision been taken between manual vs automated regression testing?	8	1	1
15	Have clear responsibilities been assigned to team members?	9	1	0
16*	Is the testing team agreed to start regression testing?	6	2	2

Table 7.11: Evolution of checklist to determine exit criteria of regression testing (RT) to be filled by test manager together with team members ($CL_3 – V_{1.1}$).

CLI	Is the checklist item relevant?	Yes	No	Don't Know
1	Have the regression testing test suites been executed completely?	10	0	0
2	Has the pass rate of regression testing suites reached the threshold?	9	0	1
3	Have all severe /critical defects been resolved?	10	0	0
4	Have all medium severity defects been closed?	7	1	2
5	Have all metrics been collected?	6	0	4
6	Have defined regression testing goals been achieved?	10	0	0
7	Is the test team agreed to test closure?	9	0	1

QA unit head of C11 suggested the following two items to be included in the checklists.

- Are there any pending changes that will be deployed during RT cycle? (Table 7.10)
- Has the regression test report been consolidated and shared? (Table 7.11)

We have provided our reflection on these suggestions in Section 7.6 (discussion).

7.5.4 Checklists evaluation (RQ3)

We evolved the checklists in two iterations since the practitioners approved most of the items included in the first version of checklists except for a couple of cases mentioned in the preceding section. Therefore, we did not iterate the checklists further. We sent the second version (version 1.1) of the checklists and an evaluation questionnaire to the practitioners and asked them to provide feedback on the following questions:

Q1 Do you think the proposed checklists cover all essential aspects that must be considered before and after regression testing?

Q2 The proposed regression testing checklists are helpful in your team/organization context?

Q3 The proposed regression testing checklists are customizable for your team/organization context?

Q4 Do you think checklists are easy to adopt in your organization 's context?

Q5 Are you willing to use the proposed checklists in your team/organization?

Figure 7.4 provides the detailed evaluation feedback provided by the study participants. Concerning Q1 (i.e., completeness), five of ten respondents chose “strongly agree”, three “agree”, one “disagree”, and one chose to remain neutral. For Q2 (i.e., utility), four out of ten respondents chose "strongly agree", six chose "agree", and none of the respondents denied the usefulness of our proposed checklists. Similarly against Q3 (i.e. customizable) we received three “strongly agree”, seven “agree”, no one voted against it. For Q4 (i.e., easy to adopt) one respondent opted “strongly agree”, two “agree”, six “neutral”, and one opted “disagree”. Finally, for Q5 (i.e., willing to use) three respondents chose “strongly agree”, three “agree”, three “neutral”, and one chose “disagree”.

Overall the feedback was hopeful, except for Q4 (i.e., easy to adopt). We expected this response because, during the interviews, many of the participants highlighted the fact that even if they want to adopt the checklists or any other process improvement tool, maybe they will get a negative response from higher management of the companies.

7.6 Discussion

We conducted this study to support testing practitioners in formalizing and improving the regression testing practice by introducing regression testing checklists. We opted for a multi-step co-design approach and involved 25 practitioners of 12 companies in first two phases and 23 practitioners from 10 companies in last two phases of our study. Most participants represented large-size companies, and the average experience of the participants was 12 years. Therefore, we can say that the findings of this study represent the perspective of senior testing practitioners working in large-scale development environments. We started our process with workshops where we presented our research idea to the prospective participants and discussed the modalities of their participation in the study. In the subsequent steps, we built our understanding of

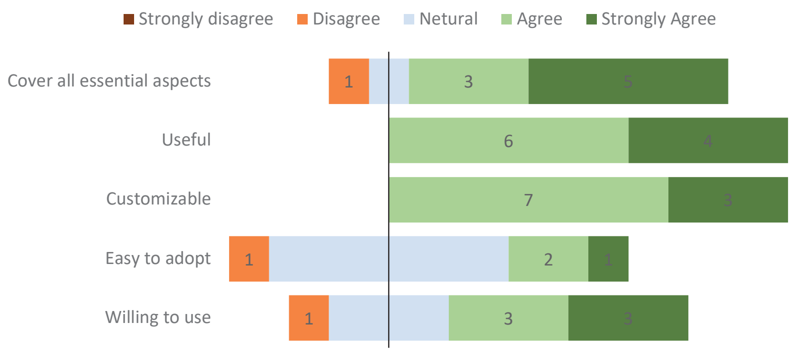

Figure 7.4: Evaluation feedback from the participants on the final version of regression testing checklists.

the current state of regression testing practice in the participants' companies, investigated the regression testing activities considered essential by the participants, and investigated their opinion on regression testing checklists.

During the data collection phases, we observed that practitioners not only recognize the significance of checklists, but some are also using some form of regression testing checklists. However, they pointed out that their checklists are application-specific and cannot be generalized. They emphasized the need for checklists to help practitioners to keep track of essential regression testing activities.

7.6.1 Regression testing activities

We identified regression testing activities directly from the practitioners by following a bottom-up approach. Using the same interview structure, we conducted group and individual interviews with the testing practitioners. However, the group interviews allowed the participants to build argument-based discussions on our questions. The participating practitioners reflected on their team perspectives; they sometimes complemented the points their fellow team members raised, and in some cases, they built the arguments against a raised point. The group interviews took more time compared to the individual interviews. One possible limitation of our investigations could be the different perspectives of the practitioners working in different companies. We did not see many variations in this regard. The practitioners working in different companies consider many of the identified regression testing activities equally important. This commonality allowed us to group similar activities under a single label, and we did so using thematic analysis. We verified our interpretations of regression testing activities from the participants To ensure that participants agreed to our themes, we get validated these from the participants. We classified the identified regression testing activities as “activities considered before regression testing” and “activities considered after regression testing”. Some of the activities identified in this study are also available in empirical studies on regression testing. We have provided a few examples in Table 7.12. However, presenting activities concerning applicability is a unique contribution to our knowledge. Furthermore, while investigating the essential regression testing activities, the aim was to make the investigation more representative by incorporating the views of practitioners working in diverse environments and development domains.

Table 7.12: Regression testing activities identified from selected studies

Factor identified from literature	Ref	Identified in our study
Understanding requirements specifications	[19, 20, 22, 24]	Acquiring domain knowledge
Understanding Changes	[19–22, 24]	Knowing new feature/changes
Identification of affected areas (Dependencies)	[20, 22, 24]	Identifying impact of changes
Modules to be tested	[20]	Deciding RT scope
Selection of right test cases	[19, 20, 24]	Selection of right test cases
Collaborate with developers	[24]	Communicating changes
Preparing test reports	[20, 22, 24]	Creating test reports
Analyzing test results	[20, 22, 24]	Analyzing test results

7.6.2 Checklists creation and evolution

We followed a systematic approach for the checklist creation and evolution (See Figure 7.1). Using the activities considered essential for regression testing by senior testing practitioners (See Table 7.4) and input of practitioners about the prospective checklists we created three checklists (See Tables 7.6, 7.7, & 7.8). These checklists would help remind testing practitioners of the essential measures to be taken before and after regression testing. Since we evolved the checklists only for two iterations, we do not claim the comprehensiveness of the proposed checklists. These checklists provide a basis for structuring the regression testing process, and practitioners can improvise the checklist during its use. Practitioners working on domains other than represented in this study can customize these checklists according to their needs.

During the evolution phase a few checklists’ items receive fewer recommendations, we highlighted these items in different colours (i.e. red, cyan, and gray). Our interpretation of the colours is that we exclude them from the checklists if red. If cyan, we suggest inclusion, and if gray, we leave the choice of inclusion or exclusion to the practitioners. We received suggestions from three participants concerning including a few items in the checklists, and we have presented these suggestions in the results. We did not enforce these items to be included in the final checklists because we consider that these items are adding further detail to already existing items. For example, items suggested by the senior QA lead of C2 *“Is QA sign-off document ready?” and “ Are the stakeholders agree on QA sign-off?”* are the further interpretation of item #7 of the checklist presented in Table 7.11. The items suggested by the test manager of C8 *" Do we need to add new test cases in the regression suite?"* could correspond to the checklist item # 7, 8, & 9, *" What is the trade-off between manual vs automated testing?"* is similar to item # 14, and *"Have we discussed the scope of regression testing in the sprint planning meeting?"* could correspond to item # 10 & 12 of the checklists presented in Table 7.10. QA unit head of C11 suggested the inclusion of two items, one *" Are there any pending changes that will be developed during RT cycle?"* in Table 7.10 and the other *"Has the regression test report been consolidated and shared?"* in Table 7.11. The item suggested for Table 7.10 is the further interpretation of item # 3. However, we consider the suggestion of including checklist item *"Has the regression test report been consolidated and shared?"* in Table 7.11 to be valuable, and we plan to add it to the checklist in future evaluations with more practitioners.

Furthermore, if the respective practitioners consider these items essential for their environment they can additionally include these items to their local checklists.

7.6.3 Checklists evaluation

We opted for an opinion-based evaluation of the proposed checklists by the study participants. The practitioners' opinion was based on their experience in testing, a trial run of checklists, and discussion among the team members. Practitioners from two companies (C5 & C10), who participated in the study's initial phases, could not participate in the study's evolution and evaluation phases. In their feedback, 80% of the respondents think checklists are comprehensive, besides the fact that we only went through two iterations of checklists design and evolution. Considering the communication and cognitive gap between regression testing research and practice a reported fact [29, 30], we were a little dubious if the proposed checklists are applicable in varying contexts of participating companies. However, the evaluations were affirmative beyond our expectations. 100% of our respondents think that the proposed checklists are helpful in their team/organization context, and 100% responded that the checklists could be customized in their team/organization context.

We added a question to ask the participants if they were willing to use these checklists. The participants from six companies showed willingness to use the checklists on an experimental basis. At this stage we are not including the usage-based feedback of practitioners in the current study because practitioners could not give us a definite timeline for providing the usage results of checklists. However, they assured us they would send us their feedback once they completed at least one usage cycle of the checklists. We plan to publish the usage data of checklists and improved version of the checklists in our future work.

7.6.4 Implications

This study has its implications for regression testing research and practice. In the following, we briefly discuss the implications for regression testing practice and the implications for regression testing research.

Implications for practice

During our interactions with the practitioners for our various studies (e.g., [22–24]), we observed that regression testing practice lacks documented structure. Most regression testing decisions are based on expert judgement, and activities are ad-hoc. The practitioners are aware of this fact and realize the need to introduce some structure in the regression testing activities.

The checklists proposed in this study are meant to help practitioners to keep track of regression testing activities. These are easy to adopt as the checklists' items represent activities that are considered essential by the practitioners for regression testing. The proposed checklists will remind practitioners not to miss an activity required for success. These simple checklists will aid the test managers in making necessary decisions concerning regression testing. For example, when to start and when to stop regression testing. Since the checklists are designed in collaboration with senior testing practitioners from varying contexts, therefore,these are scalable to the industry context. Using the feedback loop introduced for the design of the checklists, the practitioners can improvise the checklists by adding, removing, or updating the checklist items.

The proposed checklists will introduce a repeatable process at the team and organizational levels. Practitioners can reflect on the outcomes of adopted regression testing activities. Repetitive use of successful activities would enable practitioners to define and document the regression testing process according to their organizational context, which will be an ultimate step toward improving the regression testing process.

Implications for research

From the research perspective, the study has two kinds of implications 1) Specific implications for regression testing research and 2) Implications for empirical research.
Implications for regression testing research: Regression testing is a well-researched area, and many regression testing techniques have been proposed in the literature [21, 22]. However, supporting regression testing practice in decision-making is an area overlooked by software engineering researchers. In this study, by incorporating the practitioners' perspectives, we have proposed checklists to support practitioners in decision-making. The study will open up new horizons for regression testing researchers. They can work to support regression testing practice, for example, test management- activities, supporting practitioners in essential regression testing activities, and improving the regression testing process.
Implications for empirical research: The challenging part of our study was to engage the practitioners for a longer period since we needed to involve them from identifying regression testing activities to the final evaluation of the checklists. Our experience in this regard can be helpful to the software engineering researchers involved in empirical research. The following steps helped us engage practitioners through all phases of our study.

Introductory workshops: A practical approach to engaging practitioners in the studies is conducting introduction workshops and convincing them about the worth of the idea for practice.

Validate the findings: After the interpretation of the findings, getting validated by the participating practitioners will serve two purposes 1) it will increase the investigators' confidence in the results, 2) it will give a sense to practitioners concerning the significance of research for investigators, and 3) it will increase practitioners' trust in the relevance of the results to their organizational context. .

Keep them updated: Another way of keeping practitioners' engagement alive is to keep them updated about the progress and results.

Communicate the final results: After finalizing the results, communicate these to study participants. Also, discuss the plan of action with them. It will help for future engagements.

7.7 Conclusion

We conducted a multi-step co-design study to create and evolve regression testing checklists to help practitioners improve the regression testing activities by keeping track of essential regression testing activities. Twenty-five practitioners from twelve companies participated in the first two phases of the study (i.e., until checklists creation). In the latter two phases (i.e., checklists evolution and evaluation), twenty-three practitioners from ten companies participated in the study.
As a result of **RQ1**, we identified regression testing activities considered essential in the companies. The identified activities can be classified as 1) activities to be considered before regression testing, and 2) activities to be considered after regression testing.
In the next step, we transformed the identified activities into the respective regression testing checklist. Two primary types for the checklists were finalized as a result of **RQ2**, i.e., i) two checklists to track the pre-regression testing activities, and ii) one checklist to track the post-regression testing activities. Later, we evolved the checklists based on the feedback of participating practitioners.
Finally, the same practitioners evaluated the checklists and provided us with their feedback after a trial run and discussions among their team members **RQ3**. The practitioner 's feedback was positive about the various aspects of the checklists, except one, where we asked them *"Do you think checklists are easy to adopt in your organization 's context?"*, 60% of the respondents chose to stay neutral. The reason not to take a clear stance by the majority was the constraints of getting support from higher management. 60% of the respondent showed their willingness to use the checklists at the team level. This shows the practitioners found checklists helpful in improving their regression testing practice. In the current study, we do not include any data concerning the usage of checklists. However, in future, we plan to collect the usage data from the participants who are willingly using the checklists. Further, we aim to evaluate the checklists from more practitioners other than the ones who participated in this study. Based on this data, we will see the possibility of improving and generalizing the checklists.

7.8 References

[1] A. Kasoju, K. Petersen, and M. V. Mäntylä, “Analyzing an automotive testing process with evidence-based software engineering,” *Information and Software Technology*, vol. 55, no. 7, pp. 1237–1259, 2013.

[2] W. E. Perry, *Effective methods for software testing: Includes complete guidelines, checklists, and templates*. John Wiley & Sons, 2007.

[3] K. Petersen, J. Carlson, E. Papatheocharous, and K. Wnuk, “Context checklist for industrial software engineering research and practice,” *Computer Standards & Interfaces*, vol. 78, p. 103541, 2021.

[4] M. Usman, K. Petersen, J. Börstler, and P. S. Neto, “Developing and using checklists to improve software effort estimation: A multi-case study,” *Journal of Systems and Software*, vol. 146, pp. 286–309, 2018.

[5] W. Y. Higgins and D. J. Boorman, “An analysis of the effectiveness of checklists when combined with other processes, methods and tools to reduce risk in high hazard activities,” *Boeing Technical Journal*, 2016.

[6] R. Van de Schoot, P. Lugtig, and J. Hox, “A checklist for testing measurement invariance,” *European Journal of Developmental Psychology*, vol. 9, no. 4, pp. 486–492, 2012.

[7] M. A. Heroux and J. M. Willenbring, “Barely sufficient software engineering: 10 practices to improve your cse software,” in *2009 ICSE workshop on software engineering for computational science and engineering*. IEEE, 2009, pp. 15–21.

[8] B. Brykczynski, “A survey of software inspection checklists,” *ACM SIGSOFT Software Engineering Notes*, vol. 24, no. 1, p. 82, 1999.

[9] J. Brito and A. C. Dias-Neto, “Conducting empirical studies to evaluate a technique to inspect software testing artifacts,” *CLEI Electronic Journal*, vol. 16, no. 1, pp. 10–10, 2013.

[10] M. A. Madaio, L. Stark, J. Wortman Vaughan, and H. Wallach, “Co-designing checklists to understand organizational challenges and opportunities around fairness in ai,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–14.

[11] B. M. Hales and P. J. Pronovost, “The checklist—a tool for error management and performance improvement,” *Journal of critical care*, vol. 21, no. 3, pp. 231–235, 2006.

[12] B. Hales, M. Terblanche, R. Fowler, and W. Sibbald, “Development of medical checklists for improved quality of patient care,” *International Journal for Quality in Health Care*, vol. 20, no. 1, pp. 22–30, 2008.

[13] A. Chaparro, J. R. Keebler, E. H. Lazzara, and A. Diamond, “Checklists: A review of their origins, benefits, and current uses as a cognitive aid in medicine,” *ergonomics in design*, vol. 27, no. 2, pp. 21–26, 2019.

[14] M. Host and P. Runeson, “Checklists for software engineering case study research,” in *First international symposium on empirical software engineering and measurement (ESEM 2007)*. IEEE, 2007, pp. 479–481.

[15] B. Kitchenham, D. I. Sjøberg, O. P. Brereton, D. Budgen, T. Dybå, M. Höst, D. Pfahl, and P. Runeson, “Can we evaluate the quality of software engineering experiments?” in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010, pp. 1–8.

[16] J. S. Molléri, K. Petersen, and E. Mendes, “An empirically evaluated checklist for surveys in software engineering,” *Information and Software Technology*, vol. 119, p. 106240, 2020.

[17] T. Dybå and T. Dingsøyr, “Strength of evidence in systematic reviews in software engineering,” in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, 2008, pp. 178–187.

[18] B. A. Kitchenham, O. P. Brereton, D. Budgen, and Z. Li, “An evaluation of quality checklist proposals-a participant-observer case study,” in *13th International Conference on Evaluation and Assessment in Software Engineering (EASE) 13*, 2009, pp. 1–10.

[19] M. J. Harrold and A. Orso, “Retesting software during development and maintenance,” in *Proceedings of the Frontiers of Software Maintenance Conference*, 2008, pp. 99–108.

[20] E. Engström and P. Runeson, “A qualitative survey of regression testing practices,” in *Proceedings of the International Conference on Product Focused Software Process Improvement*, 2010, pp. 3–16.

[21] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[22] N. bin Ali, E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, S. Kunze, and M. Varshosaz, “On the search for industry-relevant regression testing research,” *Empirical Software Engineering*, pp. 1–36, 2019.

[23] N. M. Minhas, K. Petersen, N. B. Ali, and K. Wnuk, “Regression testing goals-view of practitioners and researchers,” in *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*. IEEE, 2017, pp. 25–31.

[24] N. M. Minhas, K. Petersen, J. Börstler, and K. Wnuk, “Regression testing for large-scale embedded software development–exploring the state of practice,” *Information and Software Technology*, vol. 120, p. 106254, 2020.

[25] B. Kitchenham and S. L. Pfleeger, “Principles of survey research: part 5: populations and samples,” *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 5, pp. 17–20, 2002.

[26] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[27] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[28] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research: part 3: constructing a survey instrument," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 2, pp. 20–24, 2002.

[29] X. Lin, "Regression testing in research and practice," Lincoln, NE, USA, Tech. Rep., 2007.

[30] E. Engström, K. Petersen, N. B. Ali, and E. Bjarnason, "SERP-test: a taxonomy for supporting industry-academia communication," *Software Quality Journal*, vol. 25, no. 4, pp. 1269–1305, 2017.

[31] S. Dalal, Sudhir, and K. Solanki, "Challenges of regression testing: A pragmatic perspective," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 1, pp. 499–503, 2018.

[32] S. Jafrin, D. Nandi, and S. Mahmood, "Test case prioritization based on fault dependency," *International Journal of Modern Education and Computer Science*, vol. 8, no. 4, p. 33, 2016.

[33] S. Nayak, C. Kumar, and S. Tripathi, "Effectiveness of prioritization of test cases based on faults," in *3rd International Conference on Recent Advances in Information Technology (RAIT), 2016*. IEEE, 2016, pp. 657–662.

[34] A. Gawande, *Checklist manifesto, the (HB)*. Penguin Books India, 2010.

[35] A. Böhm, *Theoretical Coding: Text Analysis in Grounded Theory.* Sage London, 2004.

[36] K. Thoring, R. Mueller, and P. Badke-Schaub, "Workshops as a research method: Guidelines for designing and evaluating artifacts through workshops," 2020.

[37] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2011, pp. 275–284.

[38] A. Lacey and D. Luff, *Qualitative data analysis*. Trent focus Sheffield, 2001.

[39] E. Lindgren and J. Münch, "Raising the odds of success: the current state of experimentation in product development," *Information and Software Technology*, vol. 77, pp. 80–91, 2016.

[40] N. M. Minhas and J. Iqbal, "Software process improvement practices–a pakistani perspective," in *International Workshop on CMMI based Software Process Improvement in Small and Medium Sized Enterprises*, 2011, p. 29.

Appendix C

C.1 State of regression testing practice in the case companies

Table C.1: State of regression testing (RT) in the case companies.

CID	RT State
C1	The company performs regression testing at the functional level to study the impact of changes on the other parts of the system. The decision on when to perform regression testing is based on the changes. Regression testing is mandatory if the changes are in the leading areas. 80% of the module in the product require regression testing. The team informally sets some quality goals and tries to keep an eye during the regression testing if goals are followed. In the end, the senior practitioners review the results and, based on their experience and knowledge, gauge the achievement of the goals.
C2	The company performs regression testing during every sprint and before every release cycle. The selection of test cases is based on the areas linked to the changed parts. Change impact decides the scope of regression testing. The senior QA lead informally assesses the team 's readiness before the start of regression testing. The testing team focuses on maximizing coverage and controlling the ripple effect (i.e., ensure including the modules that are affected by the changes) in the regression testing plan. The team decides to stop regression testing when fewer bugs appear or they witness the bug repetition.
C3	The company performs regression testing before every release cycle; most regression tests are automated. Regression testing is mandatory if the changes are in the critical areas. In some cases, they skip regression testing and perform exploratory testing instead. The company is following regression testing goals formally, as the goals are part of their DevOPs chain. The company is using application-specific checklists that help them to decide on release.

C4 The company representative believes that their regression testing is in perfect shape. They perform regression testing with selected tests during the day and full suite during the night. Functional and non-functional tests are used for regression testing and are performed 24/7. The most critical time is while releasing the product. At the release, the controlled fault-slippage policy is followed.

C5 The company is transitioning towards automated regression testing, and DevOps pipelines are used that are defined using Jenkins jobs. The automated regression suite will get triggered via Jenkins jobs whenever a new commit is checked. Currently, the company 's product is in developing mode. Therefore, whenever there is a new commit, they run regression testing. Before transitioning to the new technology, they have been using a set of rules, based on which they decide the scope of regression testing.

C6 At the start of every sprint, the team decides which tickets (changes) would be fixed, and at the end of the sprint, while performing regression testing, the goal is to cover all the tickets. The regression suite is created subject to the new fixes and their impact (affected areas). The affected areas are determined using the information provided by developers. Testers also use their experience and knowledge of specifications to do so. The modules with minor changes undergo sanity check instead of regression testing.

C7 The regression testing is performed before every release, and the focus is that core functionality should always work. Besides the regular regression cycle, selective regression testing is also performed. Test cases are selected based on the knowledge of changes and impact. Regression testing is performed in three stages i. Development environment, ii. Staging environment, and iii. Production environment

C8 Regression testing is performed after every release. The scope of regression testing is decided based on the fixes reported in the release notes. The goal is to ensure that all dependent areas work correctly after any change. In case of finding any defects, hotfixes are done. Before the start of regression testing, it is ensured that the build is stable, which is done through smoke testing.

C9 The company is following agile and has CI/CD pipelines in place. After the functional testing of any module, regression testing is triggered with a selected scope. Near the release, regression testing is performed with full scope (running all tests in the regression suite).

C10 The regression testing philosophy followed in the company is to test what has to work. Risk areas are identified, and a checklist about what must work is created. Later run regression testing according to the checklist. Exploratory testing is also used to complement regression testing.

C11 The changes are classified as major and minor. Minor change refers to a change request that can be handled independently, whereas a major change means upgrading the system or adding a new module. The policy is all testing for major changes, and sanity tests are performed for minor changes. Based on the product knowledge, which areas would be tested are decided. The regression suites are significantly large, and it takes a lot of time to run all tests. Therefore, the QA team is transitioning from test it all to test all that matters.

C12	The company manufactures automated doors, and regression testing is considered crucial for the company 's products. They have a three-week sprint, and there are new features and changes in the product during every sprint. For every new addition, they perform regression testing with a selected set of test cases. The requirements specification guides the selection of test cases, and they have good traceability. Sometimes if it is hard to decide on the set of tests, then exploratory testing is preferred. The company release its product every three months, and they freeze the software three weeks before the release. At release, they run all the test cases in the regression suite.

Table C.1: State of regression testing (RT) in the case companies.

Table C.2: Regression testing activities considered essential in the case companies.

CID	Before RT	During RT	After RT
C1	Study impact, Acquire domain knowledge, Assure that relevant test cases are updated, Test suites are maintained and updated, Assure relevant specification/change documents have been delivered to testers.	Monitor fault detection and coverage, as these are primary goals	Assure the completion of planned regression testing, Assess the results and determine the quality goals, Assess the ratio of pass vs fail cases
C2	Using domain knowledge, assess the impact of changes, Take measures to reduce the ripple effect during regression testing, Determine the scope of regression testing, Select most appropriate test cases, the goal is to maximize coverage, Talk to the testers about their readiness, assigning roles and responsibilities	Monitor coverage and ensure damage is minimized	Assess the results (specially pass percentage) and ensure reporting the results to relevant people,
C3	Ensure requirements and changes have been communicated, Identification of critical areas to be tested	See if the goals are being achieved	Completion of planned regression tests, goals achievement, Use the statistics to assess the goals
C4	What new features are being developed now, what are the other changes, impact of changes, what have to be tested should be known	Who should be contacted if any queries, what is the stage of the release cycle, how many problems can be afforded right now	See if the goals are achieved, Ensure that fault slippage is controlled (assess pass vs fail ratio)
C5	Knowing what new features have been added, Knowing the impact of the new features, Writing test cases for new features, Selecting test cases to test the impact	Monitor the failing cases	Analyze the results to see i. Why tests are failing, ii. Are these failing due to code error or any other reason. If regression suite is failing then stop the release.
C6	Make sure that all tickets are fixed by the developers, What are the changes and impact of changes, Developers notes are available and complete, Test suites are maintained, Test data is maintained	Make sure that all tickets are being covered	Analyze test reports, Pass vs fail test cases, Make sure that everything is covered what was planned, what is pass percentage

C7	Domain knowledge (Changes, impact of changes, build to be tested), Changes have been tested, Build is stable, Selection of right test cases, Maintenance of regression suite, What is to be included in the test suite, A well manged regression test plan, Test data availability,	Continuous status updating (e.g., pass, fail)	Generating comprehensive test reports, analyzing failures, decide the subsequent steps based on pass percentage
C8	Clear understanding of requirements, understanding of system flows, What was implemented in before this release, what is coming in this release, Test data, performance issues, make a regression testing plan, right people for right job	Should be able to execute all planned tests without any problem	Analyze the results, Are we delivering the functionality according to the requirements, nothing is broken
C9	Knowledge of product, knowledge of changes, impact of changes, changes must be functionally tested, optimizing regression suites	Monitor the stability and performance	How much have been covered, pass and fail ratio, analyzing failures, stop based on coverage and pass fail ratio
C10	RT plan, right test data, right test environment, what has changed, impact of change (knowledge of backward dependencies), all code changes are checked in, all changes have been unit tested,RT plan is in place, scope of regression testing, what are risk	Monitor the activity to see the test logs what is failing and why	Analyze test reports, maintain regression testing documentation, Executed the planned scope, what issues are found, mitigated the identified risk, test reports have been generated
C11	Domain knowledge, Complete knowledge of the current module, Test data availability, Test environment, Contingency plan of resources, all changes have been freezed, appropriate tracking mechanism	Monitor the activity and try to analyze the tests	Planned scope covered, Analyze the failures, Pass/fail ratio, assess goals
C12	Do we have a clear regression testing plan, a good knowledge of requirements specifications, what new features/changes have been added, what is the impact, which areas need to be tested, which test will be used	Monitoring coverage, ensure that all critical areas have been identified,	generate test reports, analyze results, e.g., how much covered, what is the pass percentage

Table C.2: Regression testing activities considered essential in the case companies.

C.2 Introductory workshop

Introduction

The lead investigator (author 1) introduced the research team and the overall purpose of our research on regression testing. Later, he requested the introduction of participants, their company's environment, and the testing team.

After the introductory session, the lead investigator presented the study idea based on the following key elements.

Significance of checklists

Practitioners of various disciplines use checklists as a cognitive aid to ensure the correct completion of any task. A checklist is a standardized tool that enlists the required process criteria for the practitioners performing a specific activity. It supports recording the presence or absence of the essential process tasks. Checklists are not new for software practitioners, some famous checklists used in software industry are code review and inspection checklists, user stories checklists, sprints checklists, testing checklists, etc. We could not find an explicit checklist on regression testing.

Regression testing checklists

Regression testing checklists could help document and reuse the best testing practices and help cope with various regression testing-related challenges that practitioners face. It would be easy for the new members to grasp the organizational testing policies/activities quickly in such a case. They can benefit from checklists and become familiar with usual team practices. Without a checklist, it is highly likely that a new practitioner, for instance, can omit a necessary test or violate any team policy.

Participants' Role

We are aiming to co-design regression testing checklists. The objective is to propose such checklists that would be easy to adopt yet improve the regression testing process in the companies. In this regard, the input from testing practitioners is significant. Besides this introductory workshop, your participation would be required in at least three more phases (i.e., interviews, checklists evolution, and checklists evaluation). In the next phase, we will conduct interviews and would be interested to know your opinion on various aspects of regression testing. The role of participants would be critical during the last two phases (checklists evolution and evaluation). We expect the team discussions and trial runs in the evolution and evaluation phases.

Definition of terms

We would like to describe the terms we will use in the subsequent phases of this study. For instance, we will be using the terms readiness, essential aspects of regression testing, regression testing goals, and assessment of regression testing goals. By essential aspects, we mean the steps that should be taken before the regression testing, during the regression testing, and after the regression testing. Our opinion is that before starting regression testing, all involved should be ready for that. We would be interested to know the measures taken to assess the readiness. Concerning regression testing goals, these could correspond to the pre-defined objectives that a practitioner wants to achieve by applying a regression testing process or technique. By assessing regression testing goals, we mean to see if the goals have been achieved (maybe by using some metrics).

Study benefit

The outcome of this study would be regression testing checklists that would have been co-designed with and validated by practitioners. Hence, the checklists will be scalable to the industry context. Our aim is to make these checklists easy to adopt and benefit the practitioners in the regression testing activity.

Non-disclosure assurance

We will be interested to know about you and your team/organization. We assure you that the outcome of this study will only be used for research purposes, and we will not reveal the identity of any individual or company in any publication or presentation.

Questions

We welcome your questions concerning the prospective study, its procedures, our team, etc.

C.3 Interview guide

Participant 's background

Question 1. Could you please tell us about your professional background?

1. Your overall experience?
2. For how long you have been with this organization?
3. How do you describe your current role in the team?
4. What is the size of your team?
5. How will you describe your organization?
   - (a) The size of your organization
   - (b) Development environment (e.g., DevOps, CI/CD, etc.)
   - (c) Product(s) domain

Question 2. Could you please tell us about regression testing within your organizational/team context?

1. How significant is regression testing for your organization/team?
2. Can you please give us a walkthrough of how regression testing is performed at your organization/team?
3. How frequently regression testing is performed?

Question 3. What are the essential aspects that need to be considered before the start of regression testing?
Question 4. What are the essential aspects that need to be considered after the regression testing?
Question 5. Do you think we should check the readiness of the individuals and team before starting regression testing activity?
a. What questions should be relevant to assess the readiness of individual and team?
Question 6. Do you think that the testing team needs to set the regression testing goals before the start of regression testing?
Question 7. What type of goals/quality parameters you/your team set before the start of regression testing?

Question 8. How would you ensure that you have achieved the regression testing goal(s) Post regression testing analysis?
Question 9. What is your opinion about the usefulness of regression testing checklists in practice?

1. What type of checklist(s) can add value to the regression testing activity?
2. What are the questions you think we need to ask in this regard?

Question 10. What are the aspects that need to be considered during the regression testing activity?

Question 11. Would you like to add any further points that you think are significant for regression testing? Maybe we have missed asking?

C.4 Checklist evolution

Based on the results obtained from five groups and seven individual interviews with the practitioners of 12 companies, we have created the following checklists:

1. Checklists to track the pre-regression testing activity
   (a) Checklist to determine the readiness of an individual tester
   (b) Checklist to determine the team's readiness
2. Checklist to track the post regression testing activities

Guiding note: Please provide feedback on each checklist item. You can rate the included items as Yes, No, or Do not know. If you agree that the item is relevant, then please check yes. If the item is irrelevant, then please check No. If you are undecided, you can check Don't know.

We provided checklists in similar tables as given below. The practitioners were supposed to rate every item concerning its relevance in the respective checklist. For every checklist we asked the following question:

Is the checklist relevant? Yes No Don't Know

Checklist evolution form

No.	Is the checklist item relevant?	Yes	No	Don't Know
1				
2.				
3.				
4				
5.				
.				
.				

We asked the respondents if they wanted to suggest any new item they could add to the table given below.

Add new items for checklists

No.	New Item?	Respective checklists?	Comments
1			
2.			
3.			
4			
.			
.			

C.5 Checklist evaluation

Regression testing checklist evaluation practitioners' feedback

Guiding note: Thank you for participating in the first three phases of our study and helping us designing the regression testing checklists. Now we are interested to know your opinion on the final version of checklists. We have sent you the copy of checklists earlier. If you did not receive the final version of checklists please let us know at "*nasir.mehmood.minhas@bth.se*". We assure you that your identity will not be revealed in any of our publication, we are collecting your information just for our analysis purposes. We expect your feedback based on your experience, a trial run of the checklists, and discussion among the team members.

Your Name:	______________________
Role in the team:	______________________
Company:	______________________
email:	______________________

Question 1: Do you think the proposed checklists cover all essential aspects that must be considered before and after regression testing?
a. Strongly agree b. Agree c. Disagree d. Strongly disagree

Question 2: The proposed regression testing checklists are helpful in your team/organization context?
a. Strongly agree b. Agree c. Disagree d. Strongly disagree

Question 3: The proposed regression testing checklists are customizable for your team/organization context?
a. Strongly agree b. Agree c. Disagree d. Strongly disagree

Question 4: Do you think checklists are easy to adopt in your organization 's context?
a. Strongly agree b. Agree c. Disagree d. Strongly disagree

Question 5: Are you willing to use the proposed checklists in your team/organization?
a. Strongly agree b. Agree c. Disagree d. Strongly disagree

ABSTRACT

Background. Regression testing is a complex and challenging activity and consumes a significant portion of software maintenance costs. Researchers are proposing various techniques to deal with the cost and complexity of regression testing. Yet, practitioners face various challenges when planning and executing regression testing. One of the main reasons is the disparity between research and practice perspectives on the goals and challenges of regression testing. In addition, it is difficult for practitioners to find techniques relevant to their context, needs, and goals because most proposed techniques lack contextual information.

Objective. This work aims to understand the challenges to regression testing practice and find ways to improve it. To fulfil this aim, we have the following objectives:

1) understanding the current state of regression testing practice, goals, and challenges,

2) finding ways to utilize regression testing research in practice, and

3) providing support in structuring and improving regression testing practice.

Method. We have utilized various research methods, including literature reviews, workshops, focus groups, case studies, surveys, and experiments, to conduct the studies for this thesis.

Results. The results indicate disparities in research and practice perspectives on regression testing. The practitioners rely on expert judgment instead of a well-defined regression testing process. They face various challenges in regression testing, such as time to test, test suite maintenance, communication, lack of assessment, and issues in test case selection and prioritization.

We have proposed a GQM model representing research and practice perspectives on regression testing goals. The proposed model can help reduce disparities in research and practice perspectives and cope with the lack of assessment.

We have created regression testing taxonomies to guide practitioners in finding techniques suitable to their product context, goals, and needs. Further, based on the experiences of replicating a regression testing technique, we have provided guidelines for future replications and adoption of regression testing techniques.

Finally, we have designed regression testing checklists to support practitioners in decision-making while planning and performing regression testing. Practitioners who evaluated the checklists reported that the checklists covered essential aspects of regression testing and were useful and customizable to their context.

Conclusions. The thesis points out the gap in research and practice perspectives of regression testing. The regression testing challenges identified in this thesis are the evidence that either research does not consider these challenges or practitioners are unaware of how to replicate the regression testing research into their context. The GQM model presented in this thesis is a step toward reducing the research and practice gap in regression testing. Furthermore, the taxonomies and the replication experiment provide a way forward to adopting regression testing research. Finally, the checklists proposed in this thesis could help improve communication and regression test strategy. Moreover, the checklists will provide a basis for structuring and improving regression testing practice.

BLEKINGE TEKNISKA HÖGSKOLA • BTH •

ISSN: 1653-2090
ISBN: 978-91-7295-444-1

2022:07