Raimundas Matulevičius

Process Support for Requirements Engineering

A Requirements Engineering Tool Evaluation Approach

Doctoral thesis
for the degree of doktor ingeniør

Trondheim, June 2005

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Computer and Information Science

NTNU
Innovation and Creativity

NTNU
Norwegian University of Science and Technology
Doctoral thesis
for the degree of doktor ingeniør
Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

ISBN 82-471-7171-6 (printed vers.)
ISBN 82-471-7170-8 (electronic vers.)
ISSN 1503-8181

Doctoral theses at NTNU, 2005:142

Printed by NTNU-trykk

ABSTRACT

Requirements engineering (RE) tools are software tools which provide automated assistance during the RE process. The need for automated support varies in different projects. RE-tool support could clearly be useful if an organisation deals with requirements specifications containing many requirements which need to evolve over time. However, the mainstream RE practice relies on office tools (e.g. text editors and modelling tools) rather than targeted RE-tools (e.g., CaliberRM, RequisitePro, and DOORS) provided by various companies and research groups.

Reasons for not using the RE-tools include financial causes, such as high tool price and perceived low return on investment. The part of the problem also lies in the difficulty to evaluate such tools before acquisition to support the RE process. Hence, to support the completeness and effectiveness of RE-tool evaluation, a sound framework providing methodological guidelines to the evaluators is needed.

This work proposes an RE-tool evaluation approach (R-TEA), which provides a systematic way of the RE-tool assessment using two evaluation frameworks for the RE-tools. Both frameworks contain lists of features which provide a structure for the RE-tool comparison and assessment.

The framework for the functional RE-tool requirements consists of three dimensions: representation, agreement, and specification. The representation dimension deals with the degree of formality, where requirements are described using informal, semiformal and formal languages. The agreement dimension deals with the degree of agreement among project participants through communication means. The specification dimension deals with the degree of requirements understanding and completeness at a given time moment.

The second framework categorises the non-functional RE-tool features to process, product, and external requirements. Process requirements characterise constraints placed upon the user's work practice. Product requirements specify the desired qualitative characteristics of RE-tools. External requirements are derived from the user's internal and external environment.

Both evaluation frameworks are applied to a specification exemplar which application initiates preparation of the requirements specification for the RE-tool selection. The requirements specification contains RE-tool requirements which specify what the RE-tool should do, what characteristics, constraints and properties it should have. Assessment of the RE-tools' compatibility to the specified RE-tool requirements is performed using different evaluation techniques. Decision about RE-tool selection is made after summarising all the assessment results.

In comparison to the existing tool assessment approaches, the R-TEA method targets only the RE-tool domain and guides the user of the evaluation frameworks through RE-tool assessment. In comparison to the existing RE-tool frameworks the proposed evaluation frameworks provide a more complete and consistent RE-tool assessment by combining both functional and non-functional RE-tool features and by providing a comprehensive explanation of the terminology.

A prototype tool is developed supporting the frameworks and R-TEA in order to facilitate the RE-tool assessment. The R-TEA method is tested in a number of case studies. The findings report on positive trends of the frameworks, prototype and the R-TEA method. Furthermore, a number of RE-tool weaknesses is highlighted during the case studies. A fragment-based RE-tool prototype is implemented in order to emphasis the research efforts of the automated RE process support, and to investigate the RE-tool weaknesses in a design experiment environment.

CONTENTS

PREFACE xv

CHAPTER 1. INTRODUCTION 1
- 1.1 Background 1
- 1.2 Motivation and Objective 3
- 1.3 Research Problem and Questions 5
- 1.4 Claimed Contribution 7
- 1.5 Research Methodology 10
- 1.6 Structure 11

CHAPTER 2. INFORMATION SYSTEMS AND PROCESS IMPROVEMENT 17
- 2.1 Information System 17
  - 2.1.1 Development Methodology 19
  - 2.1.2 Development Cycles 20
  - 2.1.3 Modelling Perspectives 22
- 2.2 Process Improvement 24
  - 2.2.1 Plan-Do-Check-Act 26
  - 2.2.2 Total Quality Management 26
  - 2.2.3 Experience Factory 28
  - 2.2.4 Capability Maturity Model 31
  - 2.2.5 ISO/IEC 15504 33
  - 2.2.6 Surveys on the Process Improvement Approaches 35
- 2.3 Chapter Summary 38

CHAPTER 3. REQUIREMENTS ENGINEERING 39
- 3.1 Requirements Engineering Definition 39
  - 3.1.1 NATURE Framework 41
  - 3.1.2 3-activity Model 42
  - 3.1.3 4-activity Model 43
  - 3.1.4 Development/Management Model 44
- 3.2 Requirements Engineering Process 45
- 3.3 Requirements Specification 47
  - 3.3.1 Requirements Specification Qualitative Properties 48
  - 3.3.2 Requirements Specification Standards 50
- 3.4 Empirical Studies of Requirements Engineering 52
  - 3.4.1 Representation Activities 54
  - 3.4.2 Agreement Activities 54
  - 3.4.3 Specification Activities 55
  - 3.4.4 RE Share of Total Development Time 56
  - 3.4.5 Automated Support for RE 56
- 3.5 RE Process Improvement 58
- 3.6 Requirements Engineering Tools 61

3.6.1 Tool Weaknesses ... 61
3.6.2 Overview of RE-tools ... 63
3.6.3 Taxonomy for RE-tools ... 68
3.7 Chapter Summary ... 70

CHAPTER 4. SOFTWARE TOOL EVALUATION FRAMEWORKS AND APPROACHES ... 73
4.1 Semiotic Quality Framework ... 73
4.2 RE-tool Evaluation Frameworks ... 76
4.2.1 Wieger's Requirements ... 79
4.2.2 Lang and Duggan Requirements ... 79
4.2.3 INCOSE Framework ... 80
4.2.4 Priority-based Framework ... 82
4.2.5 Role-based Framework ... 82
4.2.6 Comparison of Frameworks ... 86
4.3 COTS Evaluation Methods ... 88
4.3.1 Procurement-Oriented Requirements Engineering ... 89
4.3.2 Off-The-Shelf Option ... 91
4.3.3 COTS Acquisition Process ... 92
4.3.4 Scenario-based Selection ... 92
4.3.5 Social Technical Approach to COTS Evaluation ... 94
4.3.6 Quality-based Selection ... 95
4.3.7 Comparison of Approaches ... 97
4.4 Chapter Summary ... 97

CHAPTER 5. FRAMEWORKS FOR FUNCTIONAL AND NON-FUNCTIONAL RE-TOOL REQUIREMENTS ... 101
5.1 Framework Definition ... 101
5.2 Framework for Functional RE-tool Requirements ... 103
5.2.1 Features of the Representation Dimension ... 104
5.2.2 Features of the Agreement Dimension ... 106
5.2.3 Features the Specification Dimension ... 106
5.2.4 Semiotic Quality and Frameworks for Functional RE-tool Requirements 108
5.2.5 Framework for Verification and Validation ... 111
5.3 Framework for Non-functional RE-tool Requirements ... 112
5.3.1 Non-functional Requirements Models ... 113
5.3.2 Requirements of the Evaluation Process ... 113
5.3.3 Overview of the Framework ... 114
5.3.4 Process Requirements ... 115
5.3.5 Product Requirements ... 116
5.3.6 External Requirements ... 120
5.4 Relationships between Frameworks ... 121
5.5 Chapter Summary ... 125

CHAPTER 6. RE-TOOLS EVALUATION APPROACH 127
6.1 Guidelines for RE-tool Evaluation Methodology....127
6.2 Specification Exemplar and Application Guidelines....129
6.2.1 Specification Exemplar for RE-tool Evaluation 129
6.2.2 Specification Exemplar Application.... 131
6.2.3 Demonstrative Example 134
6.3 Comparison with Other Tool Evaluation Approaches 139
6.3.1 PORE vs. R-TEA.... 139
6.3.2 OTSO vs. R-TEA 140
6.3.3 CAP vs. R-TEA 140
6.3.4 Scenario-based Selection vs. R-TEA.... 141
6.3.5 STACE vs. R-TEA 141
6.3.6 Quality-based Selection vs. R-TEA.... 141
6.3.7 Comparison Summary 142
6.4 Chapter Summary....145

CHAPTER 7. CASE STUDIES 149
7.1 Case Study Objectives....149
7.2 Case Study A: Framework Feature Importance 151
7.2.1 Problem Description 151
7.2.2 Environment Description.... 153
7.2.3 Discovering Environmental Needs 154
7.2.4 Evaluation of RE-tools 155
7.2.5 RE-tool limitations 157
7.2.6 Maintaining Requirements Specification 158
7.2.7 Threats to Validity 161
7.2.8 Lessons Learned 162
7.3 Case Study B: Comparison of Two RE Processes 163
7.3.1 RE Process Comparison 164
7.3.2 Comparison of Requirements Specifications.... 167
7.3.3 Threats to Validity 168
7.3.4 Lessons Learned 170
7.4 Case Study C: Comparison of Evaluation Techniques....171
7.4.1 RE-tools and Evaluation Techniques.... 172
7.4.2 Case Study Design.... 172
7.4.3 Case Study Results 174
7.4.4 Threats to Validity 177
7.4.5 Lessons Learned 178
7.5 Case Study D: Testing the RE-tool Evaluation Approach 179
7.5.1 Case Study Design.... 180
7.5.2 Result Analysis Method.... 183
7.5.3 Case Study Results 185
7.5.4 Threats to Validity 189

7.5.5 Lessons Learnt 189
7.6 Summary of Case Studies 190
7.7 Chapter Summary 194

CHAPTER 8. PROTOTYPES 197
8.1 Prototype: Framework for Functional RE-tool Requirements 197
8.1.1 Session Management 200
8.1.2 Evaluation Management 201
8.1.3 Maintenance of Evaluation Rationale 205
8.1.4 Questionnaire about Quality of Requirements Specification 206
8.1.5 Prototype Improvements 207
8.2 Environment for RE-tool Weakness Consideration 208
8.2.1 Management Level 210
8.2.2 Repository Level 213
8.2.3 User Identification Level 214
8.2.4 Representative Example 215
8.2.5 Evaluation and Extensibility 219
8.3 Chapter Summary 221

CHAPTER 9. CONCLUSIONS AND FUTURE WORKS 225
9.1 Conclusions on the Research Questions 225
9.2 Contribution 229
9.3 Limitations 231
9.4 Future Work 231
9.5 Final Remarks 233

BIBLIOGRAPHY 235

APPENDIX A 249
APPENDIX B 253
APPENDIX C 261
APPENDIX D 265
APPENDIX E 267
APPENDIX F 269
APPENDIX G 277
APPENDIX H 279
APPENDIX I 283
APPENDIX J 287

TERM INDEX 291

LIST OF FIGURES

Figure 1.1 Software Development Problems 2
Figure 1.2 Structure of the Work 12

Figure 2.1 Social Reality Construction in an Organisation 18
Figure 2.2 Waterfall Model 21
Figure 2.3 Prototyping Model 21
Figure 2.4 Process and its Environment 24
Figure 2.5 Shewhart Improvement Cycle 26
Figure 2.6 Quality Improvement Paradigm 30
Figure 2.7 Relations between Goals, Questions and Metrics 30
Figure 2.8 Capability Maturity Model 31
Figure 2.9 Requirements Engineering Process Maturity Model 32
Figure 2.10 Model-based Assessment Process in ISO/IEC 15504 33
Figure 2.11 Two Dimensional Architecture of ISO/IEC 15504 34

Figure 3.1 Three Dimensional Framework for the RE process 42
Figure 3.2 3-activity RE Process Model 43
Figure 3.3 Requirements Development/Management Model 45
Figure 3.4 RE Process in Requirements Development/Management Model 45
Figure 3.5 A Spiral Model for the RE process 46
Figure 3.6 Complexity of the RE Process 47
Figure 3.7 Requirements Specification Outline 51
Figure 3.8 Template for the Requirements Model Specification 51
Figure 3.9 RE Share Among all Software Development in Lithuania 56
Figure 3.10 Most Frequently Used Tools for the RE Activities in Lithuania 57
Figure 3.11 Most Frequently Used Tools for the RE Activities in Norway 57
Figure 3.12 Most Frequently Used Tools for the RE Activities in Finland 58
Figure 3.13 SE Tools Taxonomy - Requirements Engineering Tools 69

Figure 4.1 The Semiotic Quality Framework 75
Figure 4.2 The Framework for Quality of Modelling Languages 76
Figure 4.3 COTS Selection Process 89
Figure 4.4 PORE Process for COTS Selection 90
Figure 4.5 PORE Processes 91
Figure 4.6 CAP Information Flow 92
Figure 4.7 How Scenario Will Work with Candidate 'A' 93
Figure 4.8 COTS Acquisition Process in Scenario-based Selection 93
Figure 4.9 STACE Framework 95
Figure 4.10 STACE Evaluation Process 96

Figure 5.1 Framework for Evaluation of Functional RE-tool Requirements 104
Figure 5.2 Framework for Evaluation of Non-functional RE-tool Requirements 115
Figure 5.3 Relationship between Frameworks 123

Figure 6.1 RE-tools Evaluation Approach 132

Figure 6.2 Specification Example of the Functional Features 134
Figure 6.3 Specification Example of the Non-functional Features 135
Figure 6.4 Specification Example of the Relationships between Features 136
Figure 6.5 Requirements Specification constructed from the Example 137
Figure 6.6 R-TEA Example (RE-tool_A) 138
Figure 6.7 R-TEA Example (RE-tool_B) 138

Figure 7.1 Case Study *A* Design 152
Figure 7.2 Representation of the Problem. Static View 153
Figure 7.3 Representation of the Problem. Dynamic Relationships 154
Figure 7.4 Case Study *B* Design 165
Figure 7.5 Fragment of the Questionnaire to Evaluate the Requirements Specification 167
Figure 7.6 Fragment of the Feature-based Questionnaire 173
Figure 7.7 Design of Case Study C 174
Figure 7.8 Subjective Evaluations of "the Best" RE-tool 176
Figure 7.9 Cases study *D* Design 181
Figure 7.10 Prototype of the Framework for Evaluation of Functional RE-tool Requirements 181
Figure 7.11 First task: Preparation of Three Requirements Specifications 182
Figure 7.12 Result Analysis Method 183

Figure 8.1 Overview of the Prototype Components 198
Figure 8.2 Administrator Application in the Framework for Evaluation of Functional RE-tool Requirements 199
Figure 8.3 Framework Application in the Framework for Evaluation of Functional RE-tool Requirements 199
Figure 8.4 Login to the Prototype 200
Figure 8.5 Saving Evaluation Session 200
Figure 8.6 Open Evaluation Session 200
Figure 8.7 Feature Description Window 201
Figure 8.8 Framework Feature Selection 202
Figure 8.9 Framework Feature Selection 202
Figure 8.10 Requirements Specification for RE-tool Acquisition 203
Figure 8.11 RE-tool Selection for Evaluation 203
Figure 8.12 RE-tool Evaluation 204
Figure 8.13 Result Analysis Window for Framework Dimensions 205
Figure 8.14 Result Analysis Window for Framework Feature (FEF1.1) and Activities..... 205
Figure 8.15 Final Evaluation Report 206
Figure 8.16 Comment Window (Rationale Maintenance) 206
Figure 8.17 Questionnaire for Requirements Specification Quality 207
Figure 8.18 Repository-based Requirements Specification 209
Figure 8.19 FB-RET architecture 211
Figure 8.20 FB-RET Class Diagram 213
Figure 8.21 FB-RET Repository 215

Figure 8.22 FB-RET Main Window 216
Figure 8.23 Creation of New Master Product in FB-RET 216
Figure 8.24 Creation of New Phase Product in FB-RET 217
Figure 8.25 Definition of New Fragments in FB-RET 217
Figure 8.26 Fragment Upload from Domain in FB-RET 218
Figure 8.27 Requirements Specification in FB-RET 218

Figure B.1 Responses to Question 3 253
Figure B.2 Responses to Question 5 254
Figure B.3 Responses to Question 6 254
Figure B.4 Responses to Question 7 255
Figure B.5 Responses to Question 19 259
Figure B.6 Responses to Question 20 (tool categories) 259
Figure B.7 Responses to Question 20 (individual tools) 260
Figure B.8 Responses to Question 21 260

Figure F.1 Example of Requirements Document 269
Figure F.2 RequisitePro. Requirements Properties (1) 269
Figure F.3 Selected Requirements in the Requirements Document 270
Figure F.4 RequisitePro. Requirements Properties (2) 270
Figure F.5 RDT. Initiation of Requirements Import 271
Figure F.6 RDT. Document Import Parse Window (1) 271
Figure F.7 RDT. Document Import Parse Window (2) 272
Figure F.8 RDT. Document Import Parse Window (3) 272
Figure F.9 RDT. Document Import Parse Window (4) 273
Figure F.10 RDT. Document Import Parse Window (5) 273
Figure F.11 CORE. Window of Element Extractor 274
Figure F.12 CORE. Database Browser Window 275
Figure F.13 CORE. Window of Element Properties 276

LIST OF TABLES

Table 1.1 Reasons for Project Failure 2

Table 2.1 Steps of the corporate feedback cycle 29
Table 2.2 The Capability Levels in ISO/IEC 15504 34
Table 2.3 Comparison of the CMM and EF Approaches 35
Table 2.4 Comparison of the CMM and ISO/IEC 15504 Approaches 36
Table 2.5 Taxonomy Applied to TQM, EF, CMM and ISO/IEC 15504 37

Table 3.1 Qualitative Property Definitions 49
Table 3.2 RE Activities Used in Industry 53
Table 3.3 Survey of RE-tools 64

Table 4.1 Metrics of COTS Quality 77
Table 4.2 Wieger's (1999) Requirements 79
Table 4.3 INCOSE (2002) Requirements 81
Table 4.4 Priority-based Evaluation Framework 83
Table 4.5 Role-based Framework 84
Table 4.6 Framework Comparison 88

Table 5.1 Activities of the Representation Dimension 105
Table 5.2 Activities of the Agreement Dimension 107
Table 5.3 Activities of the Specification Dimension 109
Table 5.4 Coverage of RE-tool Framework by Semiotic Quality Framework 110
Table 5.5 Verification and Validation Support by RE-tool Framework 112
Table 5.6 Non-functional Process Requirements 115
Table 5.7 Non-functional Product Requirements 117
Table 5.8 Non-functional External Organisational Requirements 121
Table 5.9 Non-functional External Requirements for Business Parties 122

Table 6.1 Evaluation Techniques 133
Table 6.2 Example of Specification Exemplar (Evaluation results) 139
Table 6.3 Comparison of Evaluation Approaches and R-TEA 144

Table 7.1 How R-TEA Method Addressed in Case Studies 151
Table 7.2 Overall Evaluation of RE-tools According to Organisational Needs in Particular Environment. 156
Table 7.3 Features and Activities for the Evaluation Framework Suggested by Respondents during the Quantitative Analysis 156
Table 7.4 Comparison along the Representation Dimension 166
Table 7.5 Comparison along the Agreement Dimension 166
Table 7.6 Comparison along the Specification Dimension 167
Table 7.7 Comparison of the Qualitative Properties of the Requirements specifications 169
Table 7.8 Session A Results 175
Table 7.9 Session B Result 175
Table 7.10 Evaluation of Non-functional Features in Session A 177

Table 7.11 Time Schedule for the Experiment Phases 177
Table 7.12 Evaluation of Tool Functionality and Requirements Specification Quality 186
Table 7.13 t-test of the RE-tool Functionality Assessment Using Evaluation Techniques. 186
Table 7.14 t-test of the Requirements Specification Quality Evaluation Using Evaluation Techniques 186
Table 7.15 Descriptive Analysis of Evaluation Techniques (evaluation scale 1-to-6) 187
Table 7.16 Correlations Coefficient between Evaluation Technique Measures 188
Table 7.17 t-test of the Requirements Specification Quality, where Requirements Specifications are Prepared with RE-tools and Office Tools 188
Table 7.18 Descriptive Analysis of Tool Usage (evaluation scale 1-to-6) 188
Table 7.19 Correlation Coefficient between Tool Measures 188
Table 7.20 Comparison of Three Case Studies 191
Table 7.21 The Paired t-test for Agreement about Framework Features 192
Table 7.22 ANOVA test for Agreement about Framework Features 192
Table 7.23 Range of Feature Importance 193
Table 7.24 Rate of Not-tested Framework Features 193
Table 7.25 The Paired t-test for the Rate of Not-tested Framework Features 193

Table 8.1 Evaluation and Extensions of the FB-RET Prototype 221

Table B.1 Findings of Question 8 255
Table B.2 Findings of Question 9 256
Table B.3 Findings of Question 10 256
Table B.4 Findings of Question 11 256
Table B.5 Findings of Question 12 257
Table B.6 Findings of Question 13 257
Table B.7 Findings of Question 14 257
Table B.8 Findings of Question 15 257
Table B.9 Findings of Question 16 258
Table B.10 Findings of Question 17 258
Table B.11 Findings of Question 18 258

Table C.1 Evaluation Form for the Requirements Representation Dimension 262
Table C.2 Evaluation Form for the Requirements Agreement Dimension 263
Table C.3 Evaluation Form for the Requirements Specification Dimension 264

Table D.1 Raw Material of Case Study A 265

Table G.1 Raw Material of Case Study C 277

Table J.1 Evaluation of the RE-tool Functionality 287
Table J.2 Quality Evaluation of the Requirements Specification 288
Table J.3 Raw Material of the Evaluation Technique Assessment 289
Table J.4 Raw Material of the RE Process Automated Support 290

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfilment of the requirements for the degree *doktor ingeniør*. The work has been carried out at the Information Systems Group, within the department of Computer and Information Science (IDI), under supervision of Professor Guttorm Sindre. Part of the work was conducted while on a shorter research stay at the Software Engineering Research Group (SERG) at Lund University in Sweden, with Associate Professor Björn Regnell.

Acknowledgement

I would like to thank Professor Arne Sølvberg for inviting me to Trondheim for the doctorial studies and for providing an excellent environment for the research. I acknowledge and thank my supervisor Professor Guttorm Sindre for his time, open door, patience, discussions and valuable comments. I also enjoyed the freedom that was given during the pursuing of my research directions.

Part of the work has been carried out at the Engineering Research Group at Lund University in Sweden. I would like to thank Associate Professor Björn Regnell for inviting me to work with his group. I also thank fellow colleagues there, in particular, Johan Natt och Dag and Lena Karlsson, for inspiring discussions and practical support.

Thanks to all at IDI for the stimulating working atmosphere. Special thanks go to Sari Hakkarainen, Csaba Veres, Christian Mönch, Xiaomeng Su, Darijus Strašunskas, Yun Lin, Jennifer Sampson, and Lillian Hella for a practical and scientific cooperation, and to all fellow members, past and present, of the IS-group for providing the needed environment and atmosphere for this work.

I would like to thank to my friends both in Norway and Lithuania, it is a great pleasure to record my appreciation of the joy I shared with them and the help I received from them.

Special thanks go to my mother and father for supporting and believing in me during all years. I would also like to thank to my bother and his family for the continuous e-mail correspondence, regards and encouragement both from them and my parents. My sincere thank goes to Sigita for her support, understanding and encouragement all the way through.

Raimundas Matulevičius
June 29, 2005

Chapter 1

Introduction

The first chapter discusses background, motivation and objectives of this work. The research problem and research questions are defined and the claimed contributions are presented. The chapter continues with a discussion over the research methodology, and finishes with an overview of the thesis structure.

1.1 Background

Information systems development is a complex activity which comprises operating information, allocating material and human resources and managing all of them usually with a computerised software system. Kotonya and Sommerville (1998) define an information system as primarily concerned with processing information which is held in some kind of databases. Such systems are usually implemented using computer hardware and are built on top of commercial operating systems. There are several arguments for distinguishing between software and systems. However every large project involves hardware, networks, people, and procedures to follow, in other words systems of some kind (Alexander and Stevens, 2002). In most projects the system requirements engineering primarily focus on software requirements engineering.

Requirements define what the system is required to do and the circumstances under which it is required to operate. Requirements engineering (RE) is considered to be one of the most important stages in the whole system and software development process (Figure 1.1). Errors done during the RE process could be very expensive in the later development stages and during the system maintenance and use. Alexander and Stevens (2002) and Leffingwell and Widrig (2000) refer to the Standish Group and ESPITI (European Software Process Improvement Training Initiative) studies which identify three most common reasons for the project failure: 1) lack of user involvement, 2) incomplete requirements and requirements specifications, and 3) changing requirements and specifications.

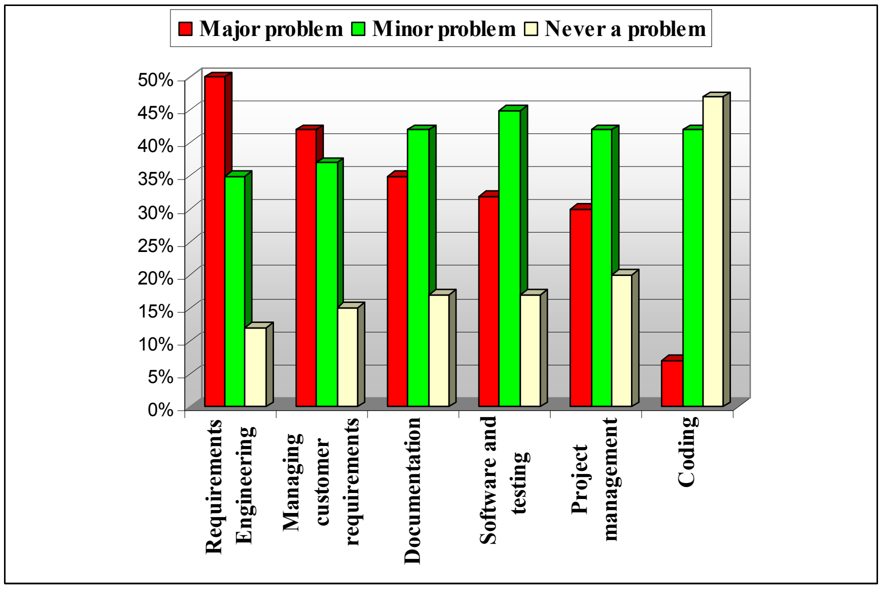


Figure 1.1 Software Development Problems adapted from (Leffingwell and Widrig, 2000)

In Table 1.1 five of eight major reasons for failure are requirements-based. The other three are related to management; and none of them are technical (Alexander and Stevens, 2002). Systems do not do what users really want, or systems are too expensive, or they are not used for the full effectiveness by the users who have paid for them.

Table 1.1 Reasons for Project Failure adapted from (Alexander and Stevens, 2002)

Incomplete requirements	13.1%
Didn't involve users	12.4%
Insufficient resources/schedule	10.6%
Unrealistic expectations	9.9%
Lack of managerial support	9.3%
Changing requirements	8.7%
Poor planning	8.1%
Didn't need it any longer	7.4%

Furthermore the RE process itself remains complex and problematic (Sommerville and Sawyer, 1997). The RE process is over budget and not predictable. People involved in RE complain that they do not have enough time and resources to do their job properly. They complain about understandability or the completeness of the requirements documents being produced. System designers complain about rework resulting from poor requirements. Finally customers fail to use all the system capabilities, there are a high volume of change request immediately after a system is delivered to customers, and it takes a long time to agree on system changes resulting from new requirements. Therefore, RE process improvement remains an important topic.

Making changes and introducing new techniques in an organisation is always difficult. There should be enough time to implement these new techniques. People who must apply the techniques should be kept fully informed during the introduction process. Before starting improvement activities for the RE process it is important to gather knowledge about 1) the RE process itself and about its maturity level; 2) budget and timescale for improvements; and 3) the people involved in implementing the RE process improvements.

The techniques to support and to improve the process could be characterised in three different ways, like *work avoidance* (reuse of methodology and techniques in development and reuse of software artefacts, e.g., requirements and requirements specifications), *working smarter* (apply new RE methods to execute the process, develop techniques for training users, support maintenance of the software products, work out techniques for requirements negotiation, development of standards both for RE process and documents), and *working faster* (use software tools, e.g., using targeted RE-tools to support the RE process). This work addresses RE process support and improvement by emphasising working faster and suggesting a methodology to acquire RE-tools in order to automate the RE process. Boehm (1999) discusses that introduction of new software tools could improve the productivity by 8 percents (in comparison to application of new methods – 17 percents and reuse of prepared artefacts – 47 percents). Eight percent of improvement is already a significant result for the RE process improvement. On the other hand introduction of new tools require organisational changes, and in most cases it means adoption of new engineering methods. Furthermore, software tools facilitate reuse of already developed artefacts across related domains by suggesting knowledge repositories and linking them across different projects.

1.2 Motivation and Objective

Requirements engineering (RE) *tools* are software tools which provide automated assistance during the RE process and support the RE activities (Matulevičius, 2004b). The literature and vendors of these tools usually call these tools as requirements management tools. However these tools do support different requirements engineering activities such as requirements elicitation, requirements documentation and analysis, requirements validation and requirements specification. Therefore they are call as RE-tool in this work.

The need for automated support may vary in different projects; and if a company does not have a mature RE process, automation won't necessarily help as there are other basic process improvements that should be done first. On the other hand if the company deals with system requirements specifications containing many requirements which need to evolve over time, RE-tool support could clearly be useful (Kotonya and Sommerville, 1998; Kaindl *et al*. 2002; Matulevičius, 2004a). However, empirical studies (Nikula et al, 2000, Hofmann and Lehner, 2001; Karlsson et al, 2002, Matulevičius 2004c) report that the mainstream RE practice relies on office tools (e.g. text editors, drawing, and modelling tools) rather than targeted RE-tools (e.g. DOORS, CaliberRM, RequisitePro, DOORS and CORE) provided by various companies or research groups.

Reasons for not using RE-tools include financial causes, like high RE-tool price, low return on investment. Companies consider it to be difficult to adapt RE tools to their organisational needs. Many software companies are not aware that there might be significant gains in taking up advanced tool support. A part of the reason might be that it is difficult to evaluate the available RE-tools. Hofmann and Lehner (2001) stress that a lack of well defined RE process and a lack of team training in the selected tools caused the insufficient support for the RE activities. In order to adopt a tool, an infrastructure must be set to support the tool. A company must be willing to invest in putting such an infrastructure in the organisation (El Emam and Madhavji, 1995). This includes personnel training, tool support groups, funding for the tool implementation. However, the management of software companies usually have unrealistic expectations, as, for example, immediate pay-off.

Because of their limited use in practice it is difficult to evaluate RE-tools in terms of their impact on an organisation's processes. Similarly, it is difficult to examine tools in an experimental situation, as it is difficult to control for the variation in system developers' capabilities. Moreover, RE-tools provide the greatest benefit for large projects with stakeholders who frequently change their minds about requirements, while a controlled experiment normally requires prescribed tasks of a fairly limited size. It would be hard to create experimental tests that would provide a realistic evaluation of the tools, and for small- and medium-size organisations the cost of thus evaluating several RE-tools empirically might be prohibitive. There is also a need for a cheaper kind of evaluation that can be done analytically rather than empirically. For instance, RE-tools can be evaluated from a theoretical point of view - using information provided by vendors. They can be tried out on some realistic examples, but without the rigour of a controlled experiment. A potential problem of such evaluations, however, is that they easily become ad hoc and subjective. Hence, to support the completeness and effectiveness of such evaluations, they should be grounded in a sound evaluation framework providing methodological guidelines to the evaluators. The objective of this work is to develop an RE-tool acquisition method, which would help to elicit the environment needs in order to adapt the RE-tool(s) in an inexpensive way and in a short amount of time.

1.3 Research Problem and Questions

In this section the research problem and questions are explained. The working title of this work has been "*Process Improvement in Information System Requirements Engineering*". The title had three major parts which are informally explained below and considered in more detail in the first part of this work.

Information System is a system for the dissemination of data between persons in order to increase their knowledge which is related to different aspects of the organisation including, e.g., its production processes, marketing, management, process constraints, policies and guidelines.

Process Improvement means the improvement of the process itself and the improvement of a product produced by this process. In this work both approaches are taken in account since the use of the tools helps to improve the process and the quality of the product, which is produced by the process.

Requirements Engineering is a systematic sequence of actions, during which the list of requirements for a new system is elicited, analysed, validated and documented into a formal, complete and agreed requirements specification.

However this title involves several issues which should be discussed. One could argue why this work addresses only RE for the information systems. As RE for information systems is primarily dealing with a software or application RE (Kotonya and Sommerville, 1998, Alexander and Stevens, 2002) the approach taken in this work could be applicable to various kind of system development. But this work does not address hardware RE which also necessary is when developing embedded, command or control systems.

Process improvement could be seen as increase of productivity (Boehm, 1999) when applying new development methods, reusing of information, or applying the software tools in the system development process. However, one could argue that application of different techniques and means (e.g., applying new requirements elicitation, analysis, documentation and validation methods, training of the workers, developing new methods for RE, reusing RE artefacts, and automating the RE activities) should not be considered as the process improvement, but rather the enactment of the process support. The improvement is not only achieved by the application of techniques or tools. Process improvement much depends on the organisation's environment, the knowledge and work practice of the workers, projects' size and management.
Taking in account the discussion above the title for this work is defined as follow:

"*Process Support for Requirements Engineering*"

The second part of the title "*An Requirements Engineering Tool Evaluation Approach*" defines more precisely the means of the RE process support and improvement. This work focuses on the evaluation and acquisition of the RE-tools. The research problem addressed is:

Research Problem: *How to improve the RE process by acquiring RE-tools?*

This work targets the RE process support and improvement by using RE-tools to automate the RE process. But the work does not develop an RE-tool itself, because of the limited resources, time available and a high competition of the existing commercial RE-tools in the market. Instead, the work discusses the RE-tool evaluation approach which would guide through evaluation process and help to select a qualitative RE-tool in order to automate the RE process. The general belief is that qualitative process support should lead to a better requirement specification. The approach could be helpful both for the RE-tool users and RE-tool vendors. For the RE-tool users the approach provides guidelines, properties and criteria on how to assess the adequacy of an RE-tool to organisational settings before acquisition; for the RE-tool vendors, the work helps to evaluate and improve both the quality and functionality of the developed RE-tool(s).

In order to narrow the investigation focus, the research problem is divided into two main research questions:

RQ.1: *What software tools provide better means to support the RE process and to maintain high-quality requirements artefacts?*

This question investigates automated support for the RE process. The main focus is on two types of software tools: office tools which include text editors (e.g., MS Office), drawing tools (e.g., MS paint), and modelling tools (e.g., Visio and Rational Rose), and targeted RE-tool (e.g., DOORS, Rational RequisitePro, CaliberRM, and CORE).

RQ.2: *How to evaluate and acquire software tools for the RE process support according to organisational needs so that they could lead to improvement of the RE process?*

The question involves the analysis of *requirements of the evaluation process*. Such requirements characterise what are the evaluation settings, for example, they define evaluation frameworks, methodological guidelines, techniques and participants. In particular the second research question addresses the tool evaluation frameworks and approaches which are applied during the assessment of tool suitability to an environment, and their performance during the tool assessment process. The working hypothesis is that if good quality methods were used for RE-tool acquisition, it would help to select the appropriate RE-tool, which would help to prepare the requirements specifications of high quality.

Both questions are considered through an analytical investigation, a survey, several case studies and prototype tools. Taken together the answers to these questions constitute the main contributions of this work.

1.4 Claimed Contribution

The objective of this work is to investigate how to evaluate the need for the automated support in the organisation and how to acquire the most suitable RE-tools to support the RE process. The main contributions of this work are:

C.1: *Two frameworks for evaluating RE-tools; the first is framework for evaluation of the functional RE-tool requirements, the second framework is for evaluation of the non-functional RE-tool requirements.*

The framework for functional requirements consists of three dimensions: representation, agreement, and specification. The representation dimension deals with the degree of formality, where informal, semiformal and formal languages are used. The agreement dimension deals with the degree of agreement among project participants through communication means. The specification dimension deals with the degree of system requirements understanding at a given time. The knowledge reuse and standards are ways to achieve a higher degree of specification completeness faster.

The second framework categorises the non-functional RE-tool features into process, product and external RE-tool requirements. Process requirements are constraints placed upon the evaluator's work practice. Product requirements specify the desired qualitative characteristics of RE-tools. External requirements derived from the RE-tool user's environment, are divided into organisational requirements and requirements for business parties. Organisational requirements describe the costs and business issues. Requirements for business parties deal with the RE-tool vendor performance, reliability, reputation, customer base and track records.

The evaluation frameworks are presented in Chapter 5. They are developed according to an analytical literature study in Chapters 2 and 4, and surveys of industrial experience in Chapter 3. They contribute to the answer of the first research question *RQ.1*, as they provide assessment criteria which are tested in empirical investigation of industrial RE practice in Chapter 3 and in case studies in Chapter 7.

C.2: *Requirements specification exemplar, used for the RE-tool evaluation.*

The specification exemplar initiates the evaluation process according to the proposed evaluation frameworks. It could also be used as an evaluation technique which guides the RE-tool assessment so it contributes to the answer to the second research question *RQ.2*.

C.3: *Methodology to use evaluation framework for the RE-tool acquisition.*

The RE-tool evaluation approach (R-TEA) is inspired from the general tool acquisition approaches discussed in Chapter 4. The R-TEA method applies a specification exemplar which is constructed from the proposed evaluation frameworks. In comparison to existing tool evaluation approaches for commercial-off-the-shelf products, the R-TEA method targets the RE-tool domain and guides the user of the evaluation frameworks through RE-tool assessment. In comparison to the existing RE-tool evaluation frameworks, the proposed evaluation frameworks provide a more complete and consistent RE-tool assessment by combining both functional and non-functional frameworks. Frameworks are also accompanied with methodological guidelines which provide help in the selection of the RE-tool(s) and are maintained by a prototype tool. The claimed contribution helps to answer research question *RQ.2*.

C.4: *Prototype tool for facilitating the RE-tool evaluation.*

A prototype of the framework for evaluation of functional requirements is created to facilitate the assessment of the RE-tools. A framework prototype uses the guidelines of the RE-tool evaluation approach and implements the evaluation framework for the functional RE-tool requirements. The framework prototype is used in the case study, and it contributes to the answer of the second research question *RQ.2*.

C.5: *Prototype – FB-RET – an RE-tool prototype to test weakly supported RE-tool features*.

While considering the first research question *RQ.1* a number of weak aspects of the automated RE process support was discovered. In order to ease analysis of the discovered weaknesses an experimental-design environment (FB-RET) is proposed in section 7.2. The FB-RET prototype emphasises the analysis of different RE activities and processes while implementing them in a unified environment and testing in the experimental design afterwards.

Finally, this work is partly based on papers presented at international conferences and workshops:

- Matulevičius R., Sindre G., Requirements Engineering Tool Evaluation Approach. *Accepted for the Proceedings of the 14th International Conference on Information Systems Development* (*ISD 2005*), Karlstad, Sweden, August, 2005.
- Matulevičius R., Sindre G., Overview of the Evaluation Approaches and Frameworks for Requirements Engineering Tools. *Accepted for the Proceedings of the 14th International Conference on Information Systems Development* (*ISD 2005*), Karlstad, Sweden, August, 2005.
- Matulevičius R., Prototype of the Evaluation Framework for Functional Requirements of RE-tools. *Accepted for the Proceedings of the 13th IEEE*

International Requirements Engineering Conference (*RE'05*), Paris, France, August-September, 2005.

- Matulevičius R., Sindre G., Requirements Specification for RE-tool Evaluation: Towards a Specification Exemplar. *Proceedings of the 17th International Conference Software and Systems Engineering and their Applications* (*ICSSEA 2004*), Paris, France, November-December, 2004.
- Matulevičius R., Karlsson L., Sindre G., How Evaluation Techniques Influence the RE-Tool Evaluation: An Experiment. *Proceedings of the industrial experience track of the European Software Process Improvement Conference* (*EuroSPI 2004*), Trondheim, Norway, November, 2004.
- Matulevičius R., Survey of Requirements Engineering Practice in Lithuanian Software Development Companies. Vasilecas O., Čaplinskas A., Wojtkowski W., Wojtkowski W. G., Zupancic J. Wrycza S. (eds.): *Proceedings of the 13th International Conference on Information Systems Development* (*ISD 2004*). *Advances in Theory, Practice and Education.* To be published by Kluwer Academic/Plenum Publishers. Vilnius, Lithuania, September, 2004, pp. 327-339.
- Matulevičius R., Process Improvement in Requirements Engineering by Acquisition of RE-tools. Glinz M. (eds.): *Proceedings of the Doctorial Consortium at the 12th IEEE International RE Conference* (*RE'04*), Kyoto, Japan, September, 2004, pp. 37-40.
- Matulevičius R., How Requirements Specification Quality Depends on Tools: A Case Study. In Persson A., Stirna J. (eds.) *Proceedings of the 16th International Conference on Advanced Information System Engineering* (*CAiSE'2004*), Riga, Latvia, (2004), Springer-Verlag Berlin Heidelberg 2004, pp. 353-367.
- Matulevičius R., Validating an Evaluation Framework for Requirement Engineering Tools. In Krogstie J., Halpin T. and Siau K., (eds.) *Information Modeling Methods and Methodologies (Adv. Topics of Database Research)*, 2004, Idea Group Publishing, pp. 148-174.
- Matulevičius R., Validating an Evaluation Framework for Requirement Engineering Tools. *Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design* (*EMMSAD'03*), Klagenfurt/Velden, Austria, June 16-20, 2003.
- Matulevičius R., Usability and Adaptability of Evaluation Framework for Requirements Engineering Tools. *Proceedings of the 2003 International MultiConference in Computer Science & Engineering* (*SERP'03*), Monte Carlo Resort, Las Vegas, Nevada, USA, June 23-26, 2003.

- Matulevičius R., Strašunskas, D. Evaluation Framework of Requirements Engineering Tools for Verification and Validation. In: Genero M., Grandi F., van den Heuvel W.-J., Krogstie J., Lyytinen K., Mayr H.C., Nelson J., Olivé A., Piattini M., Poels G., Roddick J. F, Siau K., Yoshikawa M., Yu E. S. K. (eds.): *Advanced Conceptual Modeling Techniques, ER 2002 Workshops: ECDM, MobIMod, IWCMQ, and eCOMO*, Tampere, Finland, October, 2002, Revised Papers, Springer-Verlag Berlin Heidelberg 2003, pp. 251-263.
- Matulevičius R., Process Improvement in Information System Requirements Engineering. *Scientific Proceedings of Riga Technical University*, *Information System Development (ISD 2002) Doctoral Consortium*, Riga, Latvia, 2002.

1.5 Research Methodology

Mathematics, science, and engineering have special historical relationships with computing, so different research methods may be used. Denning (2000) classifies three major research approaches: theory, experimentation and design. All three research approaches constantly interact in the process of research.

Theoretical approach involves building of conceptual frameworks and notations for understanding relationships among objects in a domain and the logical consequences of axioms and laws. Theory characterise the analytical method when a formal theory is proposed and then compared with empirical observations. *Experimentation* is a process of exploring models of systems and architectures within the given application domains and testing whether those models can predict new behaviour accurately. Experimentation describes an empirical method when a model is proposed and evaluated through empirical studies, for example, case studies and experiments. *Design* characterises constructing of systems that support work in given organisations or application domains. Design describes an engineering method, which studies the solutions and evaluates them.

Generally the research process involves three fundamental questions to be raised:

1) What is the problem that is being focused on, and why is this important? The question explains a need to search for understanding, for a sense of having found a satisfying explanation of some aspects of reality.
2) What is the best way to approach a solution to this problem? The question describes how the understanding is achieved by means of statements of general laws and principles– laws applicable to the widest possible variety of phenomena.
3) How can the proposed solution be validated (i.e., was the problem solved)? The question analyses if the laws or principles can be tested experimentally.

The research of this work addresses RE process improvement. The research objective targets the development of the RE-tool evaluation approach which contributes to selection of the RE-tools in order to improve both the RE process and product. In order to analyse the defined research questions, both descriptive and prescriptive research methods are used; and they include literature study, survey (Dillman, 2000), case studies, and experiment (Wohlin *et al.*, 2002), leaving aside field study and action research (Sankaran, 2001). One could argue that application of the RE-tool assessment method especially in an organisation that is planning to acquire an RE-tool would be valuable and useful; however the analytical literature study and the empirical survey (Matulevičius, 2004a) of software development organisations revealed some major problems in conducting field studies and/or action research. It was difficult to find an industrial organisation, which would be interested in participating in such an investigation. The organisations are not interested and not willing to automate the RE process. A part of the problem is that often their RE process is immature and needs further consideration. Therefore, all together the research phases in this work include:

1) An extensive analytical literature study in Chapters 2, 3, and 4. Post-mortem analysis of software development projects with focus on RE is followed with a survey research (Dillman, 2000) in targeted geographical areas. The analysis highlights the existing RE problems and the RE process difficulties and shows weak automated support of the RE process.
2) Both analytical and descriptive studies contribute to the proposal of the conceptual framework for evaluating the RE-tools before acquisition to the environmental settings. Chapters 5 and 6 present a theoretical approach to the RE-tool assessment which is summarised to the RE-tool evaluation approach (R-TEA).
3) The validation of the proposed method comprises both experimental empirically-based (Wohlin *et al.*, 2002) and design research approaches. Chapter 7 presents a number of case studies in order to explore different conceptual elements of the R-TEA method. Building a prototype and testing the proposed R-TEA method is the scope of Chapter 8. Two prototypes are implemented in order to target validity issues from a design point of view.

The first prototype tool supports the proposed R-TEA method itself. The second prototype describes an experimental-design environment for analysis of poorly supported features of existing commercial RE-tools.

1.6 Structure

The thesis is organised in four parts and nine chapters (Figure 1.2):

PART I, State-of-the-Art contains three chapters related to the theoretical overview of the current practices and research in information systems, process improvement and RE.

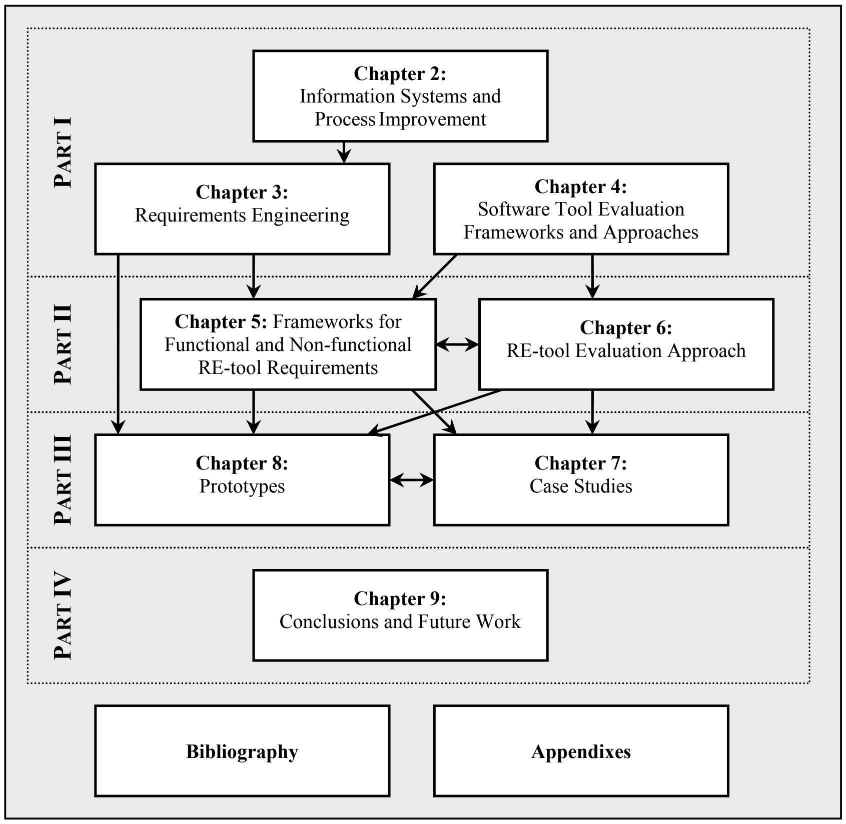

PART I
Chapter 2: Information Systems and Process Improvement
Chapter 3: Requirements Engineering
Chapter 4: Software Tool Evaluation Frameworks and Approaches
PART II
Chapter 5: Frameworks for Functional and Non-functional RE-tool Requirements
Chapter 6: RE-tool Evaluation Approach
PART III
Chapter 8: Prototypes
Chapter 7: Case Studies
PART IV
Chapter 9: Conclusions and Future Work
Bibliography
Appendixes

Figure 1.2 Structure of the Work

Chapter 2: *Information Systems and Process Improvement* has a purpose to survey the information system development methodology, development cycles and modelling perspectives. The second half of the chapter overviews and compares the process improvement approaches.

Chapter 3: *Requirements Engineering* provides a definition of requirements engineering (RE). Next it surveys the existing RE models and makes a separation between the process and product of RE. In addition the chapter overviews the empirical investigations of RE and characterises the possible targets of the RE process improvement. Finally the chapter surveys the currently available RE-tools and concludes with the problem definition to evaluate these tools before acquisition by the organisation.

Chapter 4: *Software Tool Evaluation Frameworks and Approaches* surveys the frameworks and approaches to evaluate software tools before acquisition to the environment settings. The chapter considers a semiotic quality framework first. Next an overview of the existing evaluation frameworks to compare RE-tool and the general assessment approaches to consider the commercial-off-the-shelf products is provided in the chapter.

PART II, Theoretical Approach contains two chapters about the RE-tool evaluation approach, which consists of two evaluation frameworks and guidelines for framework application.

Chapter 5: *Frameworks for Functional and Non-functional RE-tool Requirements* introduces two frameworks. The functional RE-tool requirements framework separates the RE-tool functionality into three orthogonal dimensions, such as requirements representation, requirements agreement and requirements specification. The non-functional RE-tool requirements framework characterises three non-functional groups: such as process, product and external. Finally the relationships between both frameworks are defined in the chapter.

Chapter 6: *RE-tool Evaluation Approach* proposes the guidelines for the framework application. The requirements specification for the RE-tool assessment is prepared according to the specification exemplar constructed by the means of the evaluation frameworks. Next the chapter describe six steps of the RE-tool evaluation approach and compares it with the similar tool evaluation approaches.

PART III, Evaluation and Validation contains two chapters which evaluates the proposed theoretical approach in a number of case studies and experimental design.

Chapter 7: *Case Studies* reports on four case studies, which applies the evaluation frameworks in order to compare several RE-tools. The case studies also analyse the validity issues of the RE-tool evaluation approach.

Chapter 8: *Prototypes* presents two prototypes. The first half of the chapter introduces the prototype which is constructed to support the evaluation framework of the functional RE-tool requirements. The second part of the chapter presents the environment to investigate the RE-tool weaknesses discovered both in the analytical and empirical investigations.

PART IV, Conclusions and Future Work summarises the major findings and discusses the future work.

Chapter 9: *Conclusions and Future Work* presents the conclusions of the research problem and questions and states the claimed contributions of this work. Next the limitations to the study are described along with the recommendations for the future work.

The final chapter is *Bibliography.* It includes a list of all references cited throughout the work. Next, a number of appendixes present instruments and raw material gathered in the market research and cases studies.

Appendix A describes the questionnaire used in the market research in order to investigate the RE activities executed in software organisations.

Appendix B presents the findings of the market research which investigates the activities performed in the software development organisations while executing the RE process.

Appendix C includes the evaluation form adapted according to the evaluation framework for the functional RE-tool requirements. The evaluation form is used in Case studies A and C.

Appendix D presents the results of Case study A where the importance and agreement of the evaluation framework features are analysed.

Appendix E introduces the test problem which is used in case studies to investigate the performance and correctness of the evaluation frameworks and the RE-tool evaluation approach. The problem is based on the electricity domain and describes the process of the electrical fault handling.

Appendix F shows a sample of the evaluation scenarios for three RE-tools (RequisitePro, RDT and CORE). The evaluation scenarios are defined for the same problem (presented in Appendix E).

Appendix G shows the results of Case study C which consist of two sessions executed in different universities. The appendix presents the findings in the Norwegian University of Science and Technology.

Appendix H includes the questionnaire and findings from the second part of the Case study C. The study was executed in the Lund University.

Appendix I provides the questionnaire used in Case study D which investigates the correctness and performance of the RE-tool evaluation approach.

Appendix J presents raw material of Case study D where the validity of the RE-tool evaluation approach was considered using the framework prototype.

Finally the work concludes with a list of terms used through this work.

PART I

STATE-OF-THE-ART

The purpose of this part is to survey the state-of-the-art of the process improvement in information system requirements engineering. The part consists of three chapters.

The second chapter *Information Systems and Process Improvement* overviews information system development methodology and the assessment approach of the organisational maturity. First, the information systems are defined as the spreading of knowledge among workers in the organisation. Next, an overview of the existing methodological approaches of information system development is made. The chapter also presents the existing system development life cycles and modelling perspectives used in software system and tool development.

Before starting the RE process improvement it is important to gather knowledge about the RE process itself and its maturity level, budget and timescale for improvements, and the people involved in implementation of the RE process improvements. Therefore, the existing process improvement approaches are analysed in the chapter. The approaches include the Plan-Do-Check-Act paradigm (Shewhart, 1936; Deming, 1986), Total Quality Management (Deming, 1986), Experience Factory (EF) and Quality Improvement Paradigm (Basili and Rombach, 1991; Basili, 1993; Basili and Caldiera, 1995), SEI Capability Maturity Model (Humphrey, 1989; Paulk, Curtis, Chrissis, and Weber, 1993) and ISO/IEC 15504 (El Emam, Drouin and Melo, 1998; Rout, 2001).

As discussed in the first chapter RE is one of the most important system development phases. The third chapter *Requirements Engineering* defines RE describes the RE models, and surveys RE activities, process and resulting products – requirements specification. Next, the industrial practice from different geographical areas is overviewed. The empirical investigation of the RE practice shows that software and system development organisations need to make the RE process evolve towards a more mature way, they need to teach employees to perform RE activities. Furthermore the RE process needs to be automated by adopting the RE-tools to support different RE activities. The chapter ends with a commercial RE-tools survey which highlights weak aspects of the RE-tools, and the description of the problem to evaluate the RE-tools before their acquisition for the organisational needs.

The fourth chapter *Software Tool Evaluation Frameworks and Approaches* introduces a number of RE-tool frameworks, lists of requirements, and general tool evaluation approaches. However, they are not perfectly applicable for the selection of the RE-tools for the organisational needs. For instance the NATURE framework (Pohl, 1994; Pohl, 1996) and the semiotic quality framework (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003) are too abstract for the tool evaluation purposes. Furthermore, the chapter surveys the RE-tool evaluation frameworks which provide lists of functional and non-functional RE-tool requirements as standalone instruments are not followed with guidelines on how to apply and use them for the tool assessment. Therefore an analytical overview and comparison of the existing general COTS (commercial-off-the-shelf) product selection approaches is performed. However the approaches do not specifically target the RE-tool domain; and their application is time consuming and domain knowledge demanding. The chapter raises a problem on how to assess the RE-tools before acquiring the one which is the most appropriate to the organisational environment.

Chapter 2

Information Systems and Process Improvement

Before staring to consider the RE process improvement possibilities it is important to investigate the surrounding settings and to define the concepts of information systems and process improvement models. Development of *information systems* (IS) is not a straightforward process, but comprises cycles, repetitions and continuous changes as the environmental circumstances evolve. Therefore, the need of IS development process improvement emerges, when the IS development processes are reconsidered in order to lead to better, faster, more qualitative IS definition, development and maintenance.

In this chapter an overview of the IS development methodologies, development cycles and modelling perspectives is made. Later, *process improvement* is defined, and the existing process improvement models in order to assess organisational maturity are discussed.

2.1 Information System

An *information system* (IS) is a system for the dissemination of data between persons in order to increase their knowledge (Krogstie and Sølvberg, 1996). The knowledge considered is usually related to different aspects of organisation including its production processes, marketing, management, process constraints, policies and guidelines. Organisations are constantly under the pressure of change from internal as well as external forces. Most organisations are supported by a portfolio of application systems that likewise have to be changed, often rapidly, for the organisation to be able to keep up and extend its activities.

An organisation consists of persons who view the world in their own specific way, because each of them has different experiences arising from work and other activities. The *local reality* (Figure 2.1) refers to the way a person perceives the world in which he or she lives and the way the world is for the individual. It is some kind of knowledge about how different parts of the world fit together, what is related to what, and how to interpret actions and events. When the social actors of an organisation act, they *externalise* their local reality. The ways in which persons externalise their local reality are by speaking and constructing languages, artefacts, and institutions. What they do is to construct *organisational reality* by making something that other persons have to relate to as being part of the organisation.

Organisational reality is the social order to institution that exists because everybody relates to it in their actions. The organisational reality may consist of different things, such as institutions, language, artefacts, and technology. *Internalisation* is the process of making sense out of the actions, institutions, artefacts etc. in the organisation, and making this organisational reality part of the individual local reality.

Note that this linear presentation does not mean that the processes of externalisation and internalisation occur in a strict sequence. Externalisation and internalisation may be performed simultaneously. Also, it does not mean that only organisational reality is internalised by individuals. Other externalisations also influence the construction of the local reality of an individual.

A *computerised organisational information system* is a system for the dissemination of data within an organisation which are based on the use of computers (Krogstie and Sølvberg, 1996). An *application system* is a subsystem of the computerised organisational information system being adapted to the needs of the organisation. Development of an application system is the process of producing a new application system in the organisation based on the current IS and the knowledge of internal and potentially external actors. In the following sections an overview of development methodology, development cycles and modelling perspectives used during the development of the information system is given.

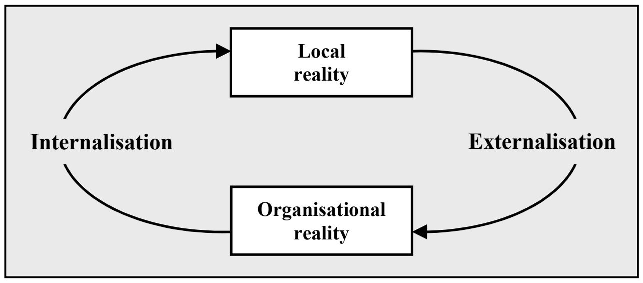


Figure 2.1 Social Reality Construction in an Organisation

2.1.1 Development Methodology

A methodology is a system of rules, approaches, and tools to aid development and/or maintenance of systems. Krogstie and Sølvberg (1996) suggest a framework for information system methodology classification. It is based on a number of questions (e.g. why, when, what, how, who, and for how long) which are asked while defining, developing, and maintaining a system.

Why is the problem attacked in a chosen way? This question addresses "*Weltanschauung*", which differentiates between three views of the world: objectivistic, constructivistic and mentalistic. *Objectivistic* world view describes that reality exists independently of any observer and merely needs to be mapped to an adequate description. *Constructivistic* world view says that reality exists independently of any observer, but what each person posses is a restricted mental model only. Finally, *mentalistic* world view claims that to talk about the reality as such does not make sense because we can only form mental constructions of the perceptions. The distinction into objectivistic and constructivistic is parallel to the distinction between objectivistic and subjectivistic done by Hirschheim and Klein (1989), who also distinguish along the order-conflict dimension. The order emphasises a social world characterised by order, stability, integration, consensus, and functional coordination. The conflict or coercion view stresses change, conflict, disintegration, and coercion.

When is a methodology applied? This aspect analyses the *coverage in process*. The methodology addresses planning changes to the IS and its support; development of application systems, use and operation of application systems; maintenance and evolution of application systems; management of planning, development, operations, and/or maintenance of application systems.

What part of the IS-portfolio is supported by the methodology? The methodology describes *coverage in product* and concerns 1) planning, development, operating, use, and/or maintenance of one single application system; 2) a family of related application systems; 3) the whole portfolio of application systems in an organisation; 4) the totality of goals, business process, people, and technology used within the organisation.

How does the methodology help in achieving the goals of IS? The question is concentrated on IS and methodology *reuse*, having in mind different aspect of it, like: motivation (why reuse is done), substance (artefact reuse, process reuse), development scope (whether the reusable entities are from a source external or internal to a project or organisation), management mode (how reuse is conducted), and technique (how reuse is implemented).

Who is involved and where are the changes done? *Stakeholders* in IS could be divided into several groups: those who are responsible for IS development, introduction and maintenance (the project manager, system developers, communications experts, technical authors, training and user support staff); those with financial interest, responsible for the application systems sale or purchase; and those who have an interest in its use (e.g., end-users, indirect users and their managers).

How is the knowledge represented? Knowledge about the process and the product of IS development and maintenance can be represented using both linguistic and non-linguistic means such as audio and video. Representational languages can be informal, semi-formal, or formal, having a logical and/or an operational semantics.

For how long has the methodology been used? The methodology *maturity* can be differentiated between maturity description (is the methodology properly described?), support by tools (is the methodology supported by tools?), practical applications (is methodology used by many organisation, supporting a large part of the portfolios in these organisations?), and its scientific background (is the methodology undergoing a conscious evolution based on experience with it and scientific study of the use of the methodology?)

2.1.2 Development Cycles

The most common system development activities are identified in (Loucopoulos and Karakostas, 1995; Sommerville, 1997). The activities include software specification (the functionality of the software and constraints on its operation is defined); software development (the software to meet specification must be produced); software validation (the system is validated in order to ensure that it does what customer wants); and software evolution (the system changes in order to meet customer needs). Different decomposition of these activities specify development processes, such as waterfall, spiral, prototyping, operational, transformational, the knowledge-based and domain-based models.

The *waterfall* model describes system development as a stepwise transformation from the problem domain to the solution through a number of phases which are wholly satisfied before their successors begin (Figure 2.2).

The *spiral* model of system development recognises the iterative nature of development and the need to plan and assess risks. The following activities (Figure 3.5) must be performed in each of the development phases: plan next phase; determine objectives, alternatives, constraints; evaluate alternatives, identify and resolve risks; and verify next level product.

The *prototyping* (evolutionary) model is a technique which constructs and experiments with a "mock-up" version of the system, in order to gain some preliminary understanding of the functionality and behaviour required from it (Figure 2.3). The engineers develop an initial implementation, expose it to user comments and refine it after through many versions until an adequate system has been developed.

The *operational* model is a system model that can be evaluated or executed in order to generate the behaviour of the system. The operational model claims that

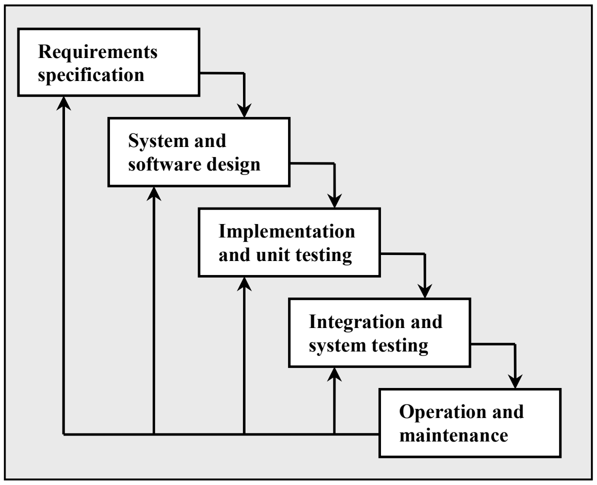

Requirements specification
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 2.2 Waterfall Model
adapted from (Sommerville, 1997)

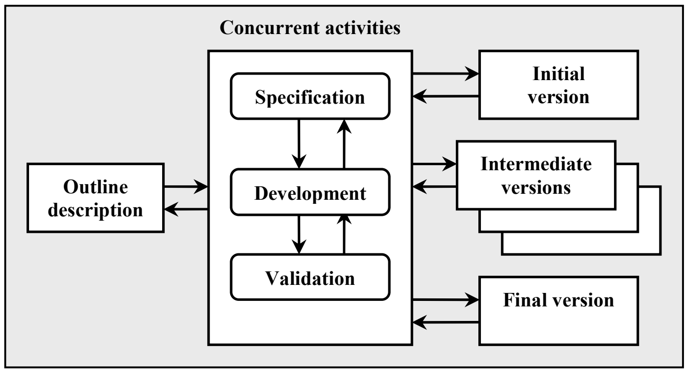


Figure 2.3 Prototyping Model
adapted from (Sommerville, 1997)

decisions about the structuring and of the domain problem should be made in the early stages of the development cycle. The operational model deliberately intertwines ‘what’ and ‘how’ decisions in an executable specification model.

The *transformational* model attempts to automate labour-intensive stages of development such as design and implementation by using a concept of a transformation. A transformation is defined as a mapping from a more abstract object to a less abstract one. The transformational model advocates the use of a series of transformations that changes a specification into a concrete system.

The *knowledge-based* model implies the usage of the (intelligent) software tools which support the development process. The term “intelligent” implies that the tools incorporate a knowledge base consisting of knowledge (guidelines) about how to perform development phases; or/and knowledge about characteristics of some problem domain, which can be employed in various development phases.

The *domain analysis* model realises the existence of similarities between applications belonging to the same problem domain and advocates that the results from one application can be reapplied to the analysis of a similar domain. Domain analysis model suggests that common concepts of similar applications of the same domain (e.g. aircraft, medical, and economical) can be organised in a library so that they can be reused in future applications.

2.1.3 Modelling Perspectives

In IS development a modelling *perspective* is defined as a rule or assumption concerning how data should be structured. Perspectives separate modelling language according to the core phenomena classes that are represented in a language. A traditional distinction regarding perspectives (Krogstie and Sølvberg, 1996) is between the structural, functional, behavioural, rule, object, communication, actor and role perspectives.

Approaches within the *structural* perspective concentrate on describing the static structure of a system. The main construct of such languages are the "entity". Other terms used for this role with some differences in semantics are object, concept, thing, and phenomena. The main modelling languages, which support this paradigm are entity-relationship (ER) diagrams (Chen, 1976), and reference modelling language (RML) (Sølvberg, 1999).

The main phenomena class in the *functional* perspective is the process. A process is defined as an activity which based on a set of phenomena transforms them to a possibly empty set of phenomena. The best know modelling language with a process perspective is data flow diagrams (DFD) (Gane and Sarson, 1979).

In most languages with a *behavioural* perspective the main phenomena are states and transitions between states. State transitions are triggered by events. A finite state machine (FSM) is a hypothetical machine that can be in only one of a given number of states at any specific time (Davis, 1988). In response to an input, the machine generates an output, and changes state. There are two language-types commonly used to model FSM: state transition diagrams and state transition matrices. Petri-Net is another well-known behaviourally oriented modelling language.

The general structure of a *rule* is *“if condition then expression*” where condition is descriptive, indicating the scope of the rule by designating the conditions in which the rule apply, and the expression is prescriptive. Example of the rule perspective language is an issue argumentation modelling approach (Chung, Nixon, Yu, and Mylopoulos, 2000) to consider non-functional requirements and the first-order logic-based external rule language (ERL) which is the part of Tempora project[1].

The basic phenomena of *object* oriented modelling languages are objects which have a unique and unchangeable identifier and a local state consisting of a collection of attributes with assignable values; the object process which is the trace of the events during the existence of the object; and class, which define a set of objects that share the same definitions of attributes and operations compose an object class. One example of the object perspective is the object modelling technique (OMT) (Rybinski, 1987) and unified modelling language (UML) (Rumbaugh, Jacobson and Booch, 1999).

The work within *communication* perspective is based on language/action theory from philosophical linguistics. The basic assumption of language/action theory is that persons cooperate within work processes through their conversations and through mutual commitments taken within them. Speech act theory is developed by Searle and Habermas, and comparison of their approaches is performed by Dietz and Widdershoven (1992).

The main phenomena of languages within *actor* and *role* perspectives are actor (alternatively agent) and role. Examples of this paradigm are agent-oriented language for building and eliciting real-time requirements (ALBERT) (Dubois, Du Bois and Petit, 1993) and TROPOS/i* (Yu, 1995).

One model in a given language would thus seldom be sufficient. With this in mind more and more approaches are based on the combination of several modelling perspectives and languages. There are four general ways to tackle this:

1. Use existing single-perspective languages as they are defined, without trying to integrate them further. This is the approach followed in many existing CASE-tools.
2. Refine common approaches to make a set of formally integrated, but still partly independent set of languages.
3. Develop a set of entirely new integrated conceptual modelling languages.
4. Create frameworks that can be used for creating the modelling languages that are deemed necessary in any given situation.

Despite of increased management load, there are a number of advantages of combining different approaches. A consequence is that it requires much better tool support to be practically applicable. Due to the increased possibilities of consistency checking and traceability across models, in addition to better possibilities for the models to serve as input for code-generation, and to support validation techniques such as execution, explanation generation, and animation the second of these approaches has been receiving increased interest, especially in the academic world. Basing integrated modelling languages on well-known modelling languages also have advantages with

[1] ESPRIT-3 project, finished in 1994

respect to perceptibility, and because of the existing practical experience with these languages.

2.2 Process Improvement

Different approaches to *process improvement* have different goals and focus on different aspects of process. There is also a considerable confusion about the meaning of the terms *process* and *improvement*. In this section some interpretations and outline on the view to both the process and its improvement is discussed.

The Webster dictionary (Walker *et al.*, 2003) defines a *process* as “a course or method of operations in the production of something”. Thus a process is structured and goal oriented. They can be broken down into operations, where the sum of operations equals the process. The process produces an output, and the structure of the process is the mean to receive this production. In this sense the IS development process could be defined as a course of operations, which produces an information system. Further, a process is executed in an organisation (Thunem, 1997); the actions are performed by people, who could play different roles. The process produces a set of products, consuming resources during its execution as shown in Figure 2.4.

The process quality much depends on this environment and the people who execute this process (Sørumgård and Sindre, 1995). There is a significant difference in the way the different roles perceive the process. *End users* and *customers* are primarily concerned about attributes of the product being developed, e.g. its price, quality and defect density. As for the process, they consider effort (which determine price), timeliness and communication with the development organisation to be important.

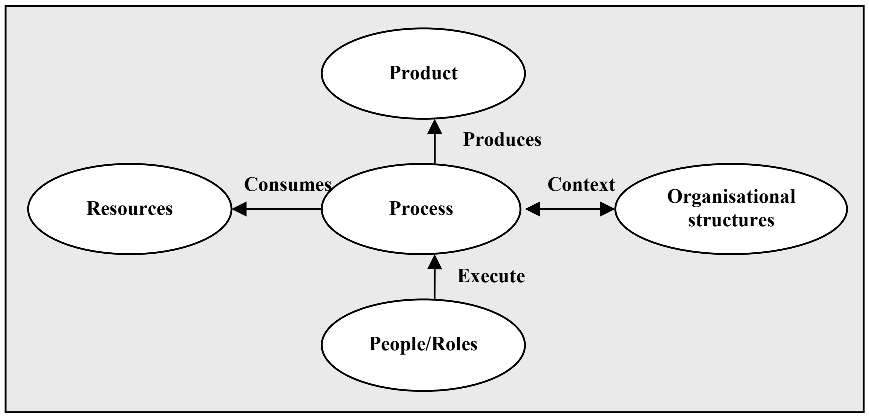


Figure 2.4 Process and its Environment adapted from (Thunem, 1997)

Managers (i.e. managers of the development organisation) are primarily concerned about productivity, cost and quality similarly to the customers. A cost/benefit analysis will have large influence on the manager's opinion about the process, since the manager is administrating the process, he requires it to be easy to change and adapt.

Developers have a completely different view of the process. They will be more likely to consider the process as a self contained entity with a notion of quality independent on other entities. Their view is that a process which allows them to use and develop their skills is good. Psychological aspects such as motivation, appreciation and stress are crucial.

In the Webster dictionary (Walker *et al.*, 2003) *improvement* is "the act of improving", and to improve means "to make better the quality, condition", or "to increase the value or profit". Thus, improvement deals with achieving higher process and product quality, and with increasing the product value. This means bringing something into a more desirable or excellent condition. Further, improvement considers the notion of productivity, as it deals with increasing profit.

Consequently, the combination of *process* and *improvement* is not interpreted uniformly. Sørumgård in 1997 divided the existing interpretations into two views:

– Improvement *of the process*, i.e., the process is the target of improvement. Here process improvement is considered as a vehicle for improvement. There must be a direct relationship between the process and the improvement target, which is defined like product. In this case it is aimed on improving product quality, and the process is a tool to measure the product quality. Business Process Reengineering (Hammer and Champy, 1995) and ISO/IEC 15504 standard (El Emam, Drouin and Melo, 1998; Rout, 2001) are the approaches, which support this paradigm.
– Improvement of *something by means of the process*, i.e., the process is a vehicle for accomplishing improvement. This approach characterises a process itself and its quality, which could be described like process efficiency, time and resource scheduling, predictability and conformance (Thunem, 1997). The approach recognises the quality control activities (Pohl, 1996), which improve the production process. This could result in higher quality products and would reduce the rework. Examples of this paradigm are the Experience Factory (Basili, 1993) and SEI Capability Maturity Model (Paulk, Curtis, Chrissis, and Weber, 1993).

Kotonya and Sommerville (1998) define four questions, which should be answered when planning process improvements: 1) what are the problems with the current process; 2) what are the improvements goals; 3) how can process improvements be introduced to achieve these goals; and 4) how should improvements be controlled and managed. There is a number of process improvement approaches defined by various authors, for instance:

– The Plan-Do-Check-Act (PDCA) paradigm or "Shewhart cycle" (Shewhart, 1936; Deming, 1986);

- Total Quality Management (TQM) (Deming, 1986);
- Experience Factory (EF) and Quality Improvement Paradigm (QIP) (Basili and Rombach, 1991; Basili, 1993; Basili and Caldiera, 1995);
- SEI Capability Maturity Model (CMM) (Humphrey, 1989; Paulk, Curtis, Chrissis, and Weber, 1993);
- ISO/IEC 15504 (El Emam, Drouin and Melo, 1998; Rout, 2001).

In the following these models are surveyed in some details.

2.2.1 Plan-Do-Check-Act

The Plan-Do-Check-Act (PDCA) paradigm, sometimes referred to as "Shewhart cycle" (Shewhart, 1939) uses feedback mechanism to optimize a single process/ production line. PDCA cycle depicted in Figure 2.5 provides the basic philosophy for a disciplined cyclic approach to continuous improvement. The goal of this approach is to improve a single production process model. The activities are:

- *Plan*, i.e. to define the problem and to state the improvement objective;
- *Do*, i.e. to identify possible problem causes, establish baselines and test changes;
- *Check*, i.e. to collect and evaluate data;
- *Act*, i.e. to implement system changes and determine effectiveness.

The approach uses feedback loop and statistical quality control to experiment with methods for improvement and to build predictive models of products. It influenced the creation of later approaches such as the Quality Improvement Paradigm (Basili, 1993).

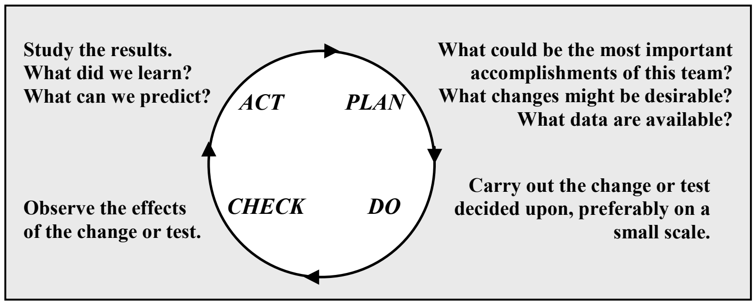

Study the results.
What did we learn?
What can we predict?
ACT
PLAN
What could be the most important accomplishments of this team?
What changes might be desirable?
What data are available?
Observe the effects of the change or test.
CHECK
DO
Carry out the change or test decided upon, preferably on a small scale.

Figure 2.5 Shewhart Improvement Cycle adapted from (Deming, 1986)

2.2.2 Total Quality Management

The Total Quality Management (TQM) represents a management approach for improving process quality and focuses on teamwork (Deming, 1986). It proposes that the involvement of all people within company is essential for improving the quality of process. The organisation is seen as a process whose only constant purpose is continuous improvement of process quality. The approach is based on 14 points proposed by Deming in 1986:

1. "Create constancy of purpose towards improvement", i.e. to replace short-term reaction with long-term planning.
   "Adopt the new philosophy". The implication is that management should actually adopt his philosophy, rather than merely expect the workforce to do so.
2. "Cease dependence on inspection". If variation is reduced, there is no need to inspect manufactured items for defects, because there won't be any.
3. "Move towards a single supplier for any one item." Multiple suppliers mean variation between feedstocks.
4. "Improve constantly and forever", i.e. constantly strive to reduce variation.
5. "Institute training on the job". If people are inadequately trained, they will not all work the same way, and this will introduce variation.
6. "Institute leadership". Deming makes a distinction between leadership and mere supervision. The latter is quota and target-based.
7. "Drive out fear". Deming sees management by fear as counter- productive in the long term, because it prevents workers from acting in the organisation's best interests.
8. "Break down barriers between departments". TQM has the concept of the 'internal customer', that each department serves not the management, but the other departments that use its outputs.
9. "Eliminate slogans". Another central the TQM idea is that it's not people who make most mistakes - it's the process they are working within. Harassing the workforce without improving the processes they use is counter-productive.
10. "Eliminate management by objectives". Deming saw production targets as encouraging the delivery of poor-quality goods.
11. "Remove barriers to pride of workmanship". Many of the other problems outlined reduce worker satisfaction.
12. "Institute education and self-improvement".
13. "The transformation is everyone's job".

Management commitment towards global quality orientation is supported by selective process management in TQM. This process management delivers raw material from which teamwork can derive concrete proposals for process improvement. Management again has to ensure that the teamwork results are instituted and not lost. The underlying principles are similar to the PDCA cycle as it comprises the continuous cycle of organisational process improvement. The main principles of process management include:

– *Continuous improvement* comprises continuous improvement of goods and services produced in an organisation, continuous expand and growth to create a future.

– *Management commitment* means that top management should enact the improvement process. Management have to change the culture of the organisation and create an environment where continuous improvement can flourish.
– *Customer focus* characterises the customer involvement. The objective is to establish a common definition of quality with the customer and to satisfy customers' needs.
– *Right work, right first time.* The goal of all organisational activities is to do the right work (obtain the customers' true requirements) and to do the work right the first time (ensuring efficiency and effectiveness).
– *Error prevention* ensures that the necessary checks and balances are performed to prevent errors.
– *Metrics* are used to measure processes and performances.
– *Corrective actions* are performed for problem solving. It is important to determine the root of the problem rather than fixing the symptoms.
– *Teamwork* is required to achieve improvements Employees involvement, group brainstorming, and team efforts are common practices.

Process improvement includes both dynamic (organisational issue, job responsibilities and performance reward) and technical (tools, techniques, methodologies and training) aspects.

2.2.3 Experience Factory

The Experience Factory (EF) is the name for the organisation that supports reuse of experience and collective learning within a software organisation (Basili and Rombach, 1991; Basili, 1993; Basili and Caldiera, 1995). The EF supports the Quality Improvement Paradigm (QIP) which is the result of the application of the PDCA cycle and focuses on continuous improvement and feedback into the process, based on packaging and reuse of experience. Both EF and QIP provide a mechanism for continuous process improvement, based on experimentation, packaging and reuse of experience-based upon a business's needs. QIP consists of six steps (Table 2.1) and implements two feedback cycles (Figure 2.6):

– The *project feedback cycle* is the feedback that is provided to the project during the execution phase. It provides analytic information about performance at intermediate stages by comparing data with the nominal range for similar projects.
– The *corporate feedback cycle* is the feedback that is provided to the organisation. Its purpose is to understand what happened by capturing experience across application domain and packing experience for reuse purposes.

Thus, the QIP method considers two parts: a *project organisation*, which is responsible for the process of characterising it environment, setting goals and choosing models and executing the process; and *experience factory*, responsible for analysing and supplying experience from previous projects.

The goal/question/metric (GQM) approach is the mechanism used by QIP for defining and evaluating a set of operational goals using measurement (Figure 2.7). The GQM approach comprises three levels, which are required to obtain the measures:

– Conceptual level (Goals). A goal is defined on its purpose, for an object, for a specific reason, with respect to a quality model, from point of view, relative to an environment.
– Operational level (Questions). Questions are selected to characterise the object under the constraints implied by the goal, and to determine its quality.
– Quantitative level (Metrics). The metrics are used to answer the questions in a quantitative way

Table 2.1 Steps of the corporate feedback cycle

No	Step	Description
1.	Characterise the project and environment	The purpose of this step is to recognise the environment of the project in order to provide a context for setting goals. The characterisation can be also served as a basis for selecting completed projects exhibiting similar characteristics from which experiences, processes and objects may be reused.
2.	Set measurable goals	The GQM approach is applied in order to arrive at a set of goals that may be quantitatively assessed. The goals are set relative to the environment that was characterised in the first step.
3.	Choose process models	This involves selecting a process model based on the context and the goals. If one of the goals is to validate a technique one may choose a process model resembling a controlled experiment. At this step the process is also tailored to the particular project needs.
4.	Execute the process	As the selected process is executed, data are collected. An important point is that the data collection is an integrated part of the development process.
5.	Analyse data	When the execution is completed, the data are analysed relative to the goals of the project. The data can be used to characterise and understand, evaluate and analyse, predict and control, and to motivate and improve.
6.	Package the method	The new experience gained in the project must be made available to the rest of the organisation for use in other projects. Thus, the relevant models are refined, and new models, if needed, are developed.

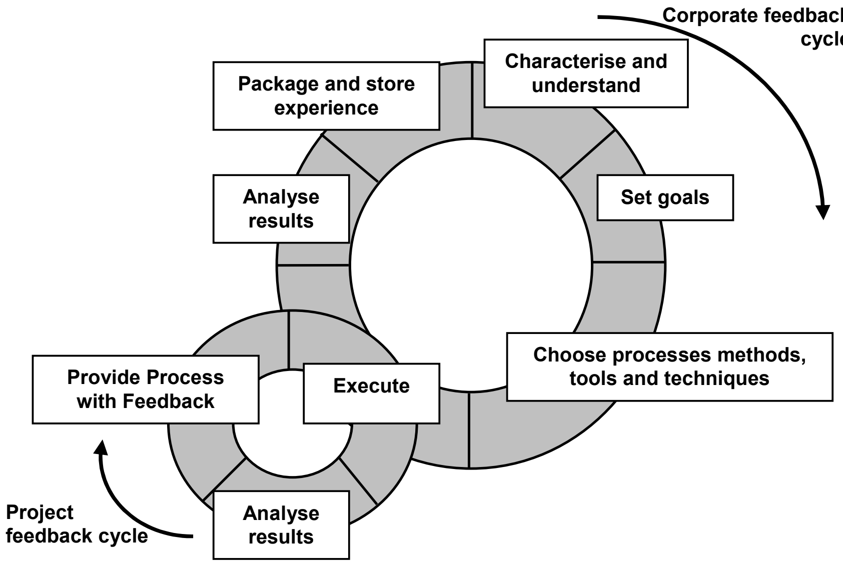

Corporate feedback cycle
Package and store experience
Characterise and understand
Analyse results
Set goals
Choose processes methods, tools and techniques
Provide Process with Feedback
Execute
Analyse results
Project feedback cycle

Figure 2.6 Quality Improvement Paradigm adapted from (Basili and Cadiera, 1995)

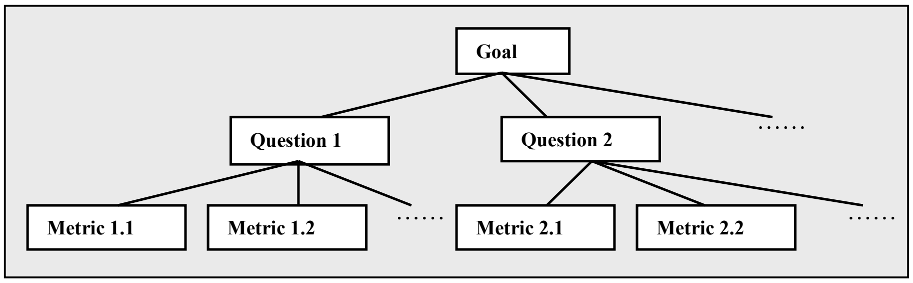


Figure 2.7 Relations between Goals, Questions and Metrics adapted from (Basili, 1993)

The QIP approach recognises the fact of improving the software process and product requires the continual accumulation of evaluated experiences in a form that it could be easily understood, modified, and stored in a repository of integrated experience models (Basili, 1993). The responsibility of the experience factory is to manage the experience, both in terms of analysing and packaging the data collected from the project, and in terms of supporting the experience base where the models and experience are stored.

2.2.4 Capability Maturity Model

The SEI Capability Maturity Model (CMM) proposes a maturity paradigm for organisations, where maturity has a greater capacity for process improvement (Humphrey, 1989; Paulk, Curtis, Chrissis, and Weber, 1993). The CMM consists of a six-step procedure to go about the five maturity levels. The CMM principles are also found in the TQM approach (Deming, 1986), as the goal of CMM is to satisfy the customer by achieving a predictable development process. The idea of the CMM is that organisation should first assess their maturity then introduce process changes

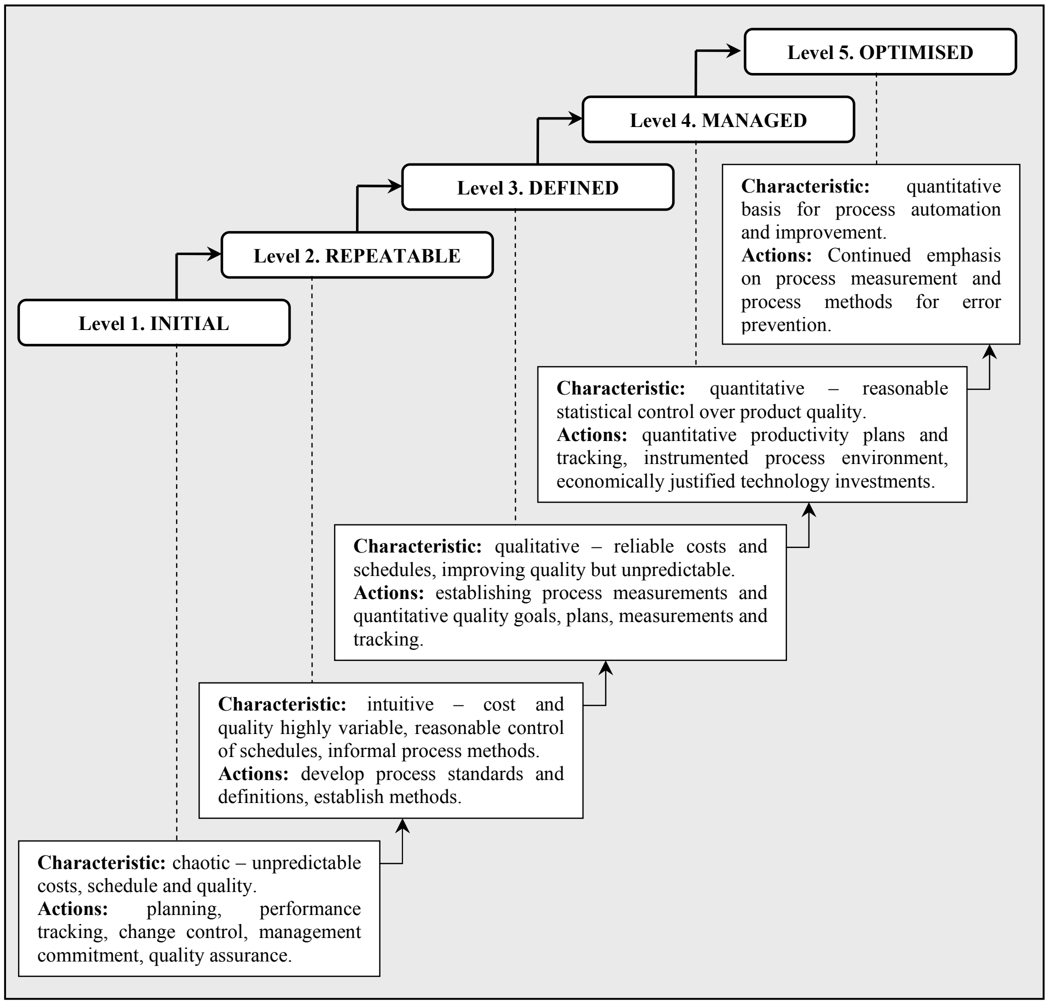

Level 5. OPTIMISED
Level 4. MANAGED
Level 3. DEFINED
Level 2. REPEATABLE
Level 1. INITIAL
Characteristic: quantitative basis for process automation and improvement.
Actions: Continued emphasis on process measurement and process methods for error prevention.
Characteristic: quantitative – reasonable statistical control over product quality.
Actions: quantitative productivity plans and tracking, instrumented process environment, economically justified technology investments.
Characteristic: qualitative – reliable costs and schedules, improving quality but unpredictable.
Actions: establishing process measurements and quantitative quality goals, plans, measurements and tracking.
Characteristic: intuitive – cost and quality highly variable, reasonable control of schedules, informal process methods.
Actions: develop process standards and definitions, establish methods.
Characteristic: chaotic – unpredictable costs, schedule and quality.
Actions: planning, performance tracking, change control, management commitment, quality assurance.

Figure 2.8 Capability Maturity Model
adapted from (Humphrey, 1989; Paulk, Curtis, Chrissis, and Weber, 1993)

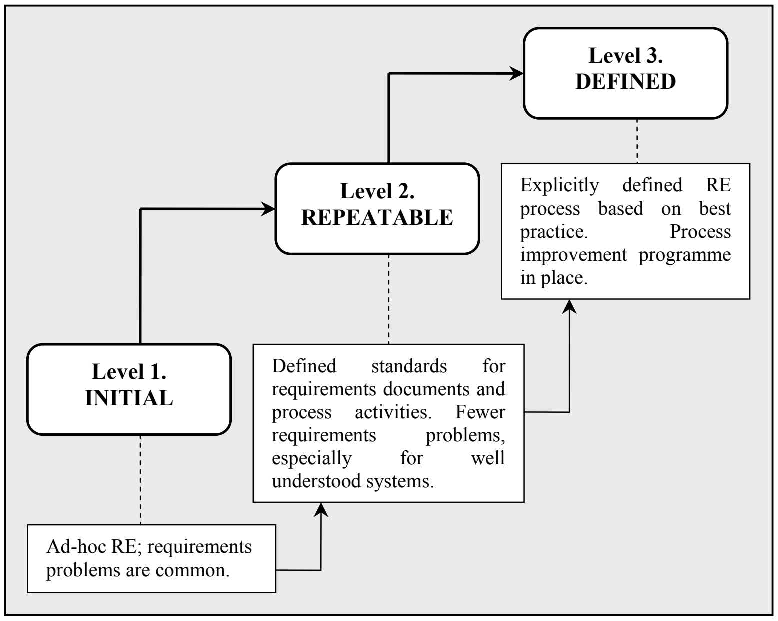

Level 3.
DEFINED
Explicitly defined RE process based on best practice. Process improvement programme in place.
Level 2.
REPEATABLE
Defined standards for requirements documents and process activities. Fewer requirements problems, especially for well understood systems.
Level 1.
INITIAL
Ad-hoc RE; requirements problems are common.

Figure 2.9 Requirements Engineering Process Maturity Model adapted from (Sommerville and Sawyer, 1997)

which enables them to progress up on the maturity 'ladder' in a five stage process (Figure 2.8). Improvement requires the organisation to take the following six steps: 1) understand the current status of the development process; 2) develop a version of the desirable process; 3) establish a list of required process improvement actions in order of priority; 4) produce a plan to accomplish the required actions; 5) commit the resources to execute plan; and 6) start over at the first step. The overall goal of the CMM approach is to lift an organisation to level five, i.e. to enable continuous process improvement based on process measurements and process methods for error prevention.

CMM has evolved into different approaches which include models for systems engineering (Bate *et al.*, 1995), systems acquisition (Cooper and Fisher, 2002), and requirements engineering (Sommerville and Sawyer, 1997; Beecham, Hall and Rainer, 2004, Sommerville and Ransom, 2005) in Figure 2.9. The differences among these discipline-specific models include their architecture, content, and approach. The recent proposal of CMMI (SEI, 2002) tends to specify a framework for supporting the integration of other discipline-specific CMM models.

2.2.5 ISO/IEC 15504

ISO/IEC 15504 (also known as SPICE – Software Process Improvement for Capability dEtermination) provides guidelines on how to perform a model-based process assessment according to the standard (El Emam, Drouin and Melo, 1998; Rout, 2001). It describes both the processes (Figure 2.10) that an organisation may perform to acquire, supply, develop, operate, evolve and support software and as well as the process attributes that characterise the capability of those processes. Further it provides a reference model, which documents a common basis for different models and methods for software process assessment, ensuring that results of assessments can be reported in a common context.

The reference model is two dimensional in ISO/IEC 15504 (Figure 2.11). The process dimension is characterised by process statements which are the essential measurable objectives of a process. The process capability dimension is characterised by a series of process attributes applicable to any process. The attributes represent measurable characteristics that are necessary to manage a process and to improve its capability to perform. Capability levels are described in Table 2.2. Each process attribute describes an aspect of the overall capability of managing and improving the effectiveness of a process in achieving its purpose and contributing into capability levels. The process guidelines include an eight-step model which forms a continuous cycle of improvement.

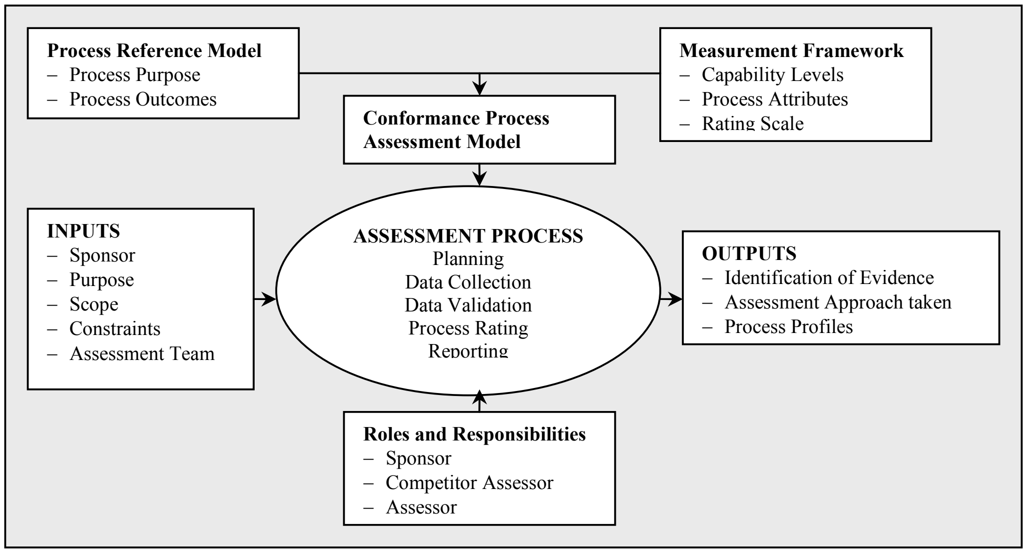


Figure 2.10 Model-based Assessment Process in ISO/IEC 15504

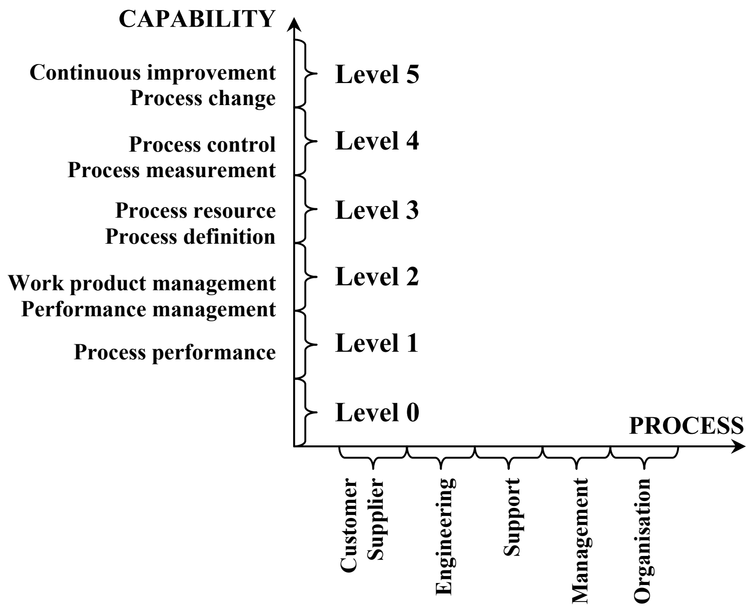


Figure 2.11 Two Dimensional Architecture of ISO/IEC 15504

Table 2.2 The Capability Levels in ISO/IEC 15504

Capability Level	**Description**
Level 0 *Incomplete*	There is general failure to attain the purpose of the process. There are little or no easily identifiable work products or outputs of the process.
Level 1 *Performed*	The purpose of the process is generally achieved. The achievement may not be rigorously planned or tracked. There are identifiable work products for the process, and these testify to the achievement of the purpose.
Level 2 *Managed*	The process delivers work products according to specified procedures and is planned and tracked Work products conform to specified standards and requirements.
Level 3 *Established*	The process is performed and managed using a defined process based upon good software engineering principles. Individual implementations of the process use approved, tailored versions of standard. Processes to achieve the process outcomes are documented.
Level 4 *Predictable*	The defined process is performed consistently in practice within defined control limits, to achieve its defined process goals.
Level 5 *Optimising*	Performance of the process is optimised to meet current and future business needs, and the process achieves repeatable in meeting its defined business goals.

2.2.6 Surveys on the Process Improvement Approaches

There are a number of works which compare the process improvement approaches. Sørumgård (1997) compared the CMM and the EF approaches in pair using a list of characteristics and a textual description (Table 2.3).

Elsewhere Paulk (1998) analyses the CMM and ISO/IEC 15504 approaches according to their architecture. The CMM is characterised as a staged model, as it describes organisational capability in terms of maturity levels that represents stages of capability. The model provides a roadmap for organisational improvement. The ISO/IEC 15504 model is described as a continuous model which defines the terrain of process maturity from the perspective of the individual process. The comparison of both architectural approaches is depicted in Table 2.4. The CMMI project (SEI, 2002) aims to harmonize these two architectures so that CMMI model can have both staged and continuous representations.

Both these surveys address only certain properties. A wider approach to process improvement comparison is suggested by Halvorsen and Conradi (2001). They compare the process improvement approaches (Table 2.5) according to a proposed taxonomy which comprises 5 categories and 25 characteristics. The general attributes or features, are specified for each process improvement approach and related to how it was constructed or designed. The *general* category concerns characteristics that describe how the process improvement approach is used. The characteristics in the *organisational* category are related to the environment in which the approach is used. The *quality* category deals with characteristics related to the quality dimension by pointing out aspects such as how progression is measured, whose quality perspective is employed and what that means in terms of quality indicators and causal relations.

Table 2.3 Comparison of the CMM and EF Approaches adapted from (Sørumgård, 1997)

Characteristics	CMM	EF
Improvement goals	Lacks definition of the process improvement goals.	The organisation is responsible for setting the process improvement goals.
Adaptability	If the predefined goals do not fit the approach is obsolete.	The approach evolves according to goals based on the context.
Measurement	Applies measurement to achieve statistic process control; however measurement does not become an issue until level three.	Starts measurement from the beginning and applies it for checking whether goals are reached.
Application	May be used for evaluating and comparing organisations.	Specific for each organisation.
Target processes	All processes are taken to require management, thus management processes are considered important.	Is based on the view that not all software is the same, thus different processes are required.
Validation	External validation based on application within different domain.	Internal validation through heavy reliance on experiment.

Table 2.4 Comparison of the CMM and ISO/IEC 15504 Approaches adapted from (Paulk, 1998)

Characteristics	CMM	ISO/IEC 15504
Vital view	Attention is focused on the vital view issues in process improvement that are generally true for any organisation.	Less important process issues can down out the vital few issues when they are clashes over improvement priorities.
Organisational capability	Organisational capability is explicitly described in terms of maturity levels.	Organisational capability is implicit; it can be intuitively understood by looking at the organisational processes, the process attributes and their dependencies.
Process evolution	Key process areas are a snapshot of the evolving process.	The evolution of processes from ad hoc to continuously improving is "fully described.
Guidance	Extensive guidance in the key practices provide significant help in understanding what a key practice or goal means, although it is typically oriented towards the practices of large organisations and projects in a contracting environments.	Abstract processes and process attributes can be difficult to interpret. No particular organisational improvement path is prescribed.
Extendibility	It may be difficult for non-expert to extend the CMM principles to new focus area.	Adding processes and integrating with other models is a relatively straightforward definition, with the application of the capability dimension for rating the processes.

Finally, the *result* category treats characteristics that describe the results of employing a process improvement approach, but also the costs of reaching these results and the methods used to validate them.

All the process improvements, although they focus on different aspects, are based on the feedback loops (Shewhart, 1936; Deming, 1986). Therefore it becomes possible to define the improvement setting when planning the continuous process improvement strategy. Pohl (1996) proposes five major characteristics of process improvement:

– There must be an explicit process definition;
– Process performance must be based on the current process models and definitions;
– Data about process execution which should be traceable, must be recorded;
– The captured execution data must be evaluated and related back to the process;
– The relation of the recorded data and the corresponding process definition.

The experience that is collected during an execution of the improvement loop is vital for evaluating the process performance. This information should be directly linked to the process definition. It also helps to choose the appropriate facilities, means, tools, methods and techniques for the process improvement.

Table 2.5 Taxonomy Applied to TQM, EF, CMM and ISO/IEC 15504 adapted from (Halvorsen and Conradi, 2001)

Category	**Characteristic**	**TQM**	**EF**	**CMM**	**ISO/IEC 15504**
General	Geographic origin/ Spread	Japan/ World	US/World	US/World	World/World
	Scientific origin	Quality control	Partly TQM	TQM, SPC	CMM, Bootstrap, Trillium
	Development/ Stability	Entire post-war era	Since 1976	Since 1986	Under development
	Popularity	High (in Japan)	Medium	Top (in US)	Growing
	Software specific	No	Yes	Yes	Yes
	Prescriptive/ Descriptive	Descriptive	Descriptive	Both	Both
	Adaptability	Yes	Yes	Limited	Yes
Process	Assessment	None	None	Org. maturity	Process maturity
	Assessor	ND	ND	Internal and external	Internal and external
	Process improvement method	PDCA	QIP	IDEAL	SPICE Doc. Part 7
	Improvement initiation	Top-down	Iterative bottom-up	Top-down	Process instance
	Improvement focus	Management processes	Experience reuse	Management processes	Management processes
	Analysis techniques	7QC, 7MP, SPC, QFD	GQM	Assessment questionnaires	Several (manual and automated) Required.
Organi-sation	Actors/ Roles/ Stakeholders	Customer, employees, management	Experience factory, project organisation	Management	Management
	Organisation size	Large	All	Large	All
	Coherence	Internal and external	Internal	Internal	Internal
Quality	Quality perspective	Customer	All	Management	Management
	Progression	Continuous	Continuous	Staged	Continuous (staged at process instance level)
	Causal relation	ND	**F1'**(*Experience reuse*) ⇒**Q**(*Process*) ⇒**Q**(*Product*) but feedback loops here	**F1'**(*key process areas*)⇒ **F2'**(*maturity level*) ⇒**Q**(process) ⇒**Q**(product)	**F1'**(*Process attributes*)⇒**F2'** (*Capability level*)⇒**Q**(*Process*) ⇒**Q**(*Product*)
	Comparative	No	No	Yes, maturity level	Yes, maturity profile
Result	Goal	Customer satisfaction	Organisation specific	Process improvement, supplier capability, determination	Process assessment
	Process artefacts	Plans, diagrams	Experience packages, GQM models	Process documentation, assessment result	Process profile, assessment record
	Certification	No	No	No	No
	Implementation cost	ND	ND	ND	ND
	Validation	None	Experimental and case studies	Surveys and case studies	Document review, trials (case studies and surveys)

ND – not determined

2.3 Chapter Summary

In this chapter a notion of *Information system* (IS) is explained as the system supporting knowledge dissemination in an organisation. Next, IS development settings which characterise the IS development process are discussed. They include methodology, development cycles and modelling perspectives. Further a notion of *process improvement* is defined as 1) the improvement of the process itself and 2) improvement of the process product by means of the process. In this work the focus is placed on both process improvement definitions. In the later chapter the means of the process improvement are presented. Before enacting process improvement itself, it is important to investigate the environmental maturity. Therefore an overview of the existing process improvement approaches to evaluate organisational maturity is given.

In the next chapter the notion of *requirements engineering* (RE) is introduced. Further, the role of RE in an information systems development process, objectives, targets, and means for the RE process improvement are discussed.

CHAPTER 3

Requirements Engineering

In the previous chapter IS development approaches and process improvement models are considered. One of the most important system development phases is requirements engineering (RE). The purpose of this chapter is to identify targets, objectives and means to the RE process support and improvement.

The term *requirements engineering* is defined and several models which characterise the RE activities and the RE process are defined in this chapter, first. The output of the RE process is a complete requirements specification represented in formal language and agreed among all project stakeholders. Next, empirical RE surveys which report on a number of RE problems are discussed in the chapter. Finally, the objectives, targets and means for the improvement of the RE process are identified. The chapter ends with an overview of RE-tools and concludes with a need for a cheep and non-expensive method to assess these tools before acquisition to environmental settings.

3.1 Requirements Engineering Definition

The term *requirement* is defined differently by various authors. The Webster's dictionary (Walker *et al.*, 2003) defines *requirement* as "that which is required". The dictionary defines the term *require* as having need, finding something necessary. Elsewhere *requirements* is refined as

1. a condition or capability needed by a user to solve a problem or achieve an objective (Leffingwell and Widrig, 2000);
2. a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document (Leffingwell and Widrig, 2000);
3. a documented representation of a condition or capability as in 1) and 2) (Loucopoulos and Karakostas, 1995).

Kotonya and Sommerville (1998) define *requirements* as "a statement of a system service or constraint". Elsewhere in (Ferdinandi, 2002) *requirement* is a need or desire to be satisfied by the product or service.

Davis in 1993 defines a *requirement* as what a system should do, without specifying how to do. However, this definition becomes problematic, when the separation between functional and non-functional requirements is made. *Functional requirements* describe what the system should do. *Non-functional requirements* are constraints to the system and they describe how functional requirements should be reflected in the system. Thus, in this work both aspects of requirements are supported. A requirement is either functional or non-functional, where

- *functional requirements specify what a system should do;* and
- *non-functional requirements describe system characteristics, constraints and properties (without specifying how they should be implemented in the system).*

Requirements engineering (RE) is an activity which aims at discovering, documenting and maintaining a set of requirements. The use of the term *engineering* implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent and relevant (Sommerville and Sawyer, 1997). Loucopoulos and Karakostas (1995) stress the importance of cooperative work when executing different RE activities, in particular, developing requirements through analysing the problem, documenting the observations in a variety of representation forms, and checking the accuracy of the knowledge gained. This definition is reflected by the three dimensional NATURE[2] framework for the RE process (Pohl, 1994, Pohl, 1996). This work basically supports the definition of Zave (1997):

> *"Requirements engineering (RE) is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families."*

This definition highlights the importance of "real-world goals" that motivate the development of a system. These represent the 'why' as well as the 'what' of a system. The definition also refers to "precise specifications". These provide the basis for RE activities like *analysing* requirements, *validating* that they are indeed what the stakeholders want, *defining* what designers have to build, and *verifying* that they have done so correctly upon delivery. Finally, the definition refers to specifications' "evolution over time and across software families", emphasising the reality of a changing world and the need to reuse partial specifications, as engineers often do in other branches of engineering (Nuseibeh and Easterbrook, 2000).

[2] NATURE project, finished in 1996.

The literature defines different RE models, which are based on the various combinations of the requirements activities. The requirements activities are elicitation, negotiation, analysis, agreement, verification and specification. However, various authors identify, specify and combine them differently. This section describes four RE models, based on the requirements activities:

- The NATURE framework (Pohl, 1994; Pohl, 1996);
- 3-activity model: requirements elicitation, requirements specification, and requirements validation (Loucopoulos and Karakostas, 1995);
- 4-activity model: requirements elicitation, requirements analysis and negotiation, requirements documentation, and requirements validation (Kotonya and Sommerville, 1998);
- Requirements development/management model (Ferdinandi, 2002).

The following sections present these RE models in detail.

3.1.1 NATURE Framework

The NATURE framework describes the RE process in a three dimensional orthogonal space (Pohl, 1994; Pohl, 1996). The three dimensions of the RE process are (Figure 3.1):

- The *requirements representation dimension* deals with the degree of representation formality. The dimension has the goal to transform informal requirements model into formal model representation.
- The *requirements agreement dimension* deals with the degree of requirements agreement. The dimension has the goals to gain a common agreement about the requirements model.
- The *requirements specification dimension* deals with the degree of requirements understanding at the certain time moment. The dimension has the goal to improve an opaque system comprehension into a complete requirements specification, where completeness is measured by standards and guidelines.

At the beginning of a RE process the knowledge about the system is vague. Therefore the specification is *opaque*, based on *personal views,* and mainly *informal representations* are used. The desired output of the requirement specification process is expected to be precise. Therefore, the resulting specification is a *complete specification*, which is expressed using *formal language*, and is *commonly agreed* by all stakeholders.

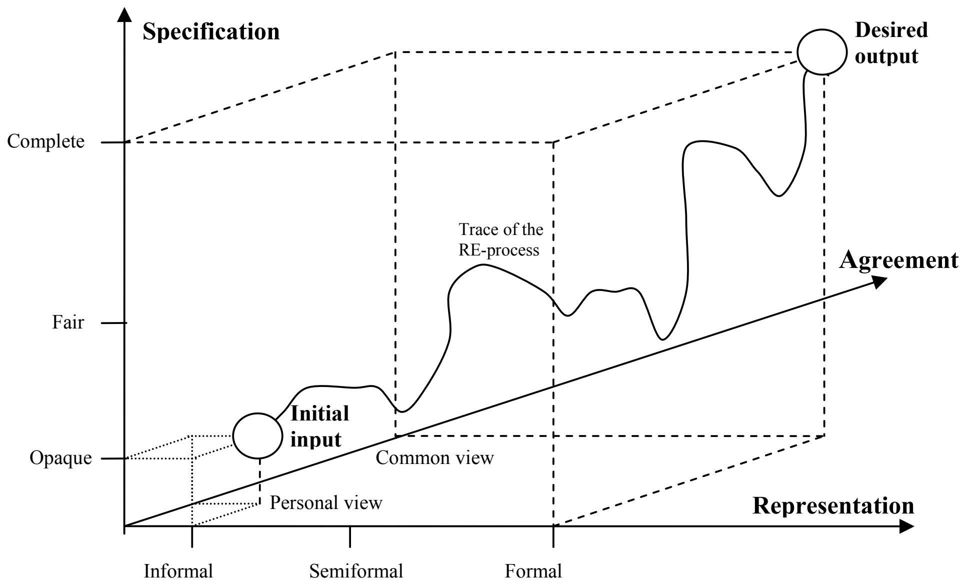


Figure 3.1 Three Dimensional Framework for the RE process adapted from (Pohl, 1993)

3.1.2 3-activity Model

Loucopoulos and Karakostas (1995) define the RE process consisting of three RE activities (Figure 3.2) – elicitation, specification, and validation. The activities are described in following way:

- *Requirements elicitation* is the activity of acquiring necessary knowledge which is used to produce a formal requirements model. The main elicitation activities comprise the identification of all the sources of requirements knowledge, acquisition of this knowledge, decision on the relevant parts of the knowledge to the problem at hand, understanding of the elicited knowledge and its impact on the software requirements, and reuse of knowledge acquired in similar problem domains.
- *Requirements specification* is the activity which receives as input the deliverables of requirements elicitation in order to create a formal model of the requirements. The basic specification activities are to analyse and assimilate the requirements knowledge, synthesize and organise it into a coherent and logical requirements model. A requirements specification model could be viewed as a document which defines the desired functionality, without showing how such functionality is going to be achieved (Davis, 1993).

- *Requirements validation* is the activity which attempts to certify that the produced formal requirements model satisfies the users' needs. The main validation activities comprise a preparation (e.g. settings for tests, case studies, experiments, and customer evaluations), performance and result analysis. Requirements validation is an ongoing process of RE which aims to ensure that the right problem is being tackled at any time.

Further the RE process model (Loucopoulos and Karakostas, 1995) focuses on in-house development, when the system is developed inside a company for companies own use, and contract development projects, when a supplier company develops a system to the customer company (Lauesen, 2002). In these projects the requirements specification activity and its output could be defined as a contract between customers and developers. However, the RE process model do not address projects, where the customer is not defined (e.g. commercial off-the-shelf development).

The 3-activity RE process model could be compared to the NATURE framework, as both approaches correspond to the fundamental RE concerns – understanding a problem, describing a problem, and attaining an agreement on the nature of the problem.

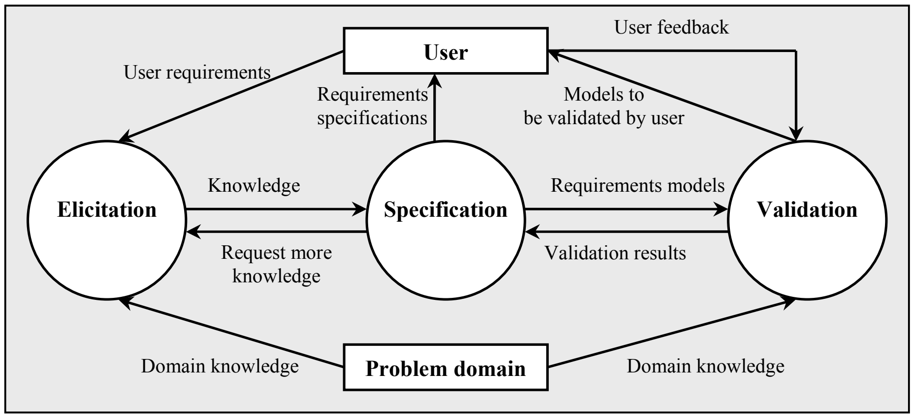


Figure 3.2 3-activity RE Process Model adapted from (Loucopoulos and Karakostas, 1995)

3.1.3 4-activity Model

Kotonya and Sommerville (1998) separate the RE process into two parts – RE activities and requirements management. According to them the RE process consists of four activities:

- *Requirements elicitation* that describes the requirements discovery and acquisition process. Requirements are discovered through consultations with stakeholders, from system documents, domain knowledge, and market studies.

- *Requirements analysis and negotiation* characterise the requirements which are accepted to a requirements model. The process helps to solve the requirements conflicts, it deals with requirements completeness and analyses the available budget to develop the system.
- *Requirements documentation* is the activity, where the requirements model is documented in the appropriate level of detail.
- *Requirements validation* checks the requirements model for consistency and completeness. This activity is intended to detect problems in the requirements model before it is used for the system development.

The 4-activity model is similar to the three dimensional NATURE framework and the 3-activity model, as it comprises the same RE principles, like formality, agreement, and understandability of the requirements model. In comparison to the 3-activity model, the 4-activity model separate specification into two activity groups – requirements analysis and negotiation, and requirements documentation.

In parallel with the all the RE activities, it is a process of *requirements management* which concerns managing changes of the requirements model. Changing requirements is necessary as errors in the requirements model are discovered, as new requirements emerge. The management activity should keep the track of changes and ensure that changes are made to the requirements model in a controllable way. The other two models – the NATURE frameworks and 3-activity model - do not separate requirements management, but address it as integrated part of the main activities.

3.1.4 Development/Management Model

Ferdinandi (2002) separates RE into two activity groups (Figure 3.3): the requirements process or development and the requirements management. Requirements process/development consists of two levels. The first level has the purpose to allocate responsibilities, to involve the stakeholders in the RE process and to divide the requirements into logical groups. The second level consists of the RE activities, which form the RE process (Figure 3.4). Elicitation, specification and validation are similar to the activities described in the 3-activity and 4-activity models. The purpose of requirements approval is to determine whether it is cost effective to continue with the current requirements model, or it is better to reconceptualise the model and implement only a portion of it.

Ferdinandi (2002) defines the requirements management in the same way as the 4-activity model. The activity is characterised like a process of setting baselines for the approved requirements and requirement model and managing any changes and their impact on other requirements and the whole requirement model.

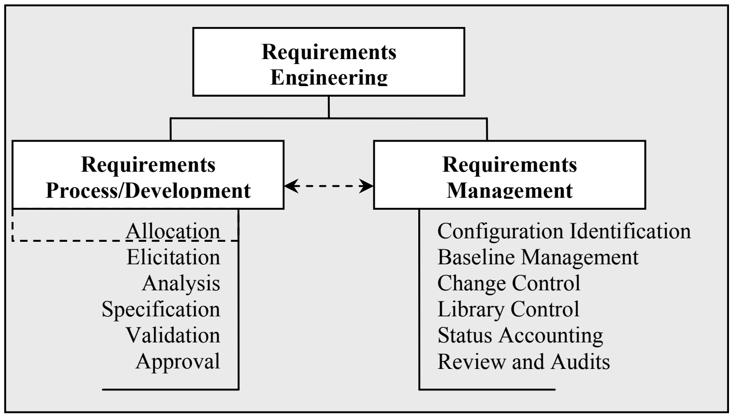


Figure 3.3 Requirements Development/Management Model adapted from (Ferdinandi, 2002)

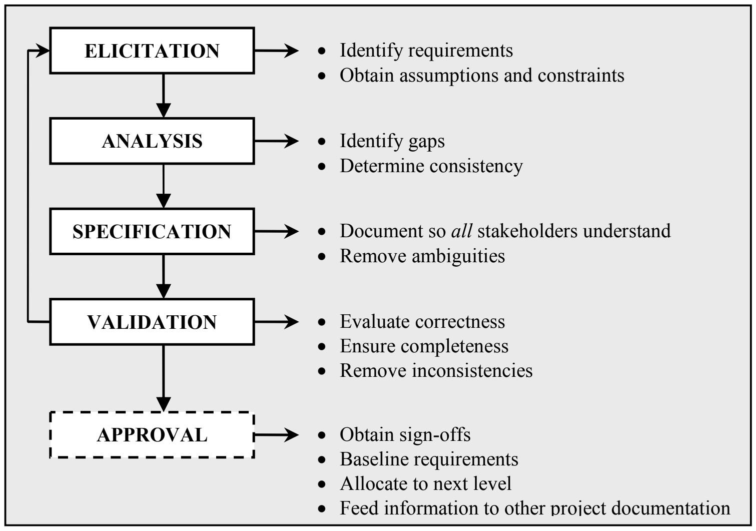


Figure 3.4 RE Process in Requirements Development/Management Model adapted from (Ferdinandi, 2002)

3.2 Requirements Engineering Process

The *RE process* is described as a sequence of actions, during which the list of requirements for a new software system is elicited, analysed, validated and documented into a formal, complete and agreed requirements specification (Pohl, 1994; Loucopoulos and Karakostas, 1995; Pohl, 1996; Zave, 1997; Kotonya and

Sommerville, 1998). All the RE models tend to describe the RE process; however, only the three dimensional framework (Pohl, 1994; Pohl, 1996) shows the continuous process and describes it in representation, agreement and specification dimensions.

Both Loucopoulos and Karakostas (1995) and Kotonya and Sommerville (1998) relate their RE models with the waterfall, spiral (Figure 3.5), prototyping, operational, transformational, knowledge-based and domain analysis software development model. However, these RE models do not characterise precisely the continuous RE process, but they tend to separate it into different activities. This is not usual in practice where the RE activities are closely interrelated or executed in parallel.

The above summarised RE models describe the RE process as incremental through the time. Firesmith (2003b) argues, that the effective RE process is iterative, incremental, parallel, and time boxed. However, the RE process heuristics (Lindland, Sindre and Sølvberg, 1994; Sindre and Krogstie, 1995) and the empirical findings of Nguyen and Swatman in 2003 show that the RE process is not a smooth and incremental evolution, but it is characterised by occasional "crisis" points (Figure 3.6). At the crisis points the requirements model is reconceptualised, the problem space is reshaped and the requirements model is simplified and restructured. The new model has a new architecture reflecting a new perception and understanding of the requirements problem. The complexity of the requirements model is reduced, and the reconceptualisation creates the new level of requirements abstraction. The newly restructured model becomes a base for further requirements development stages.

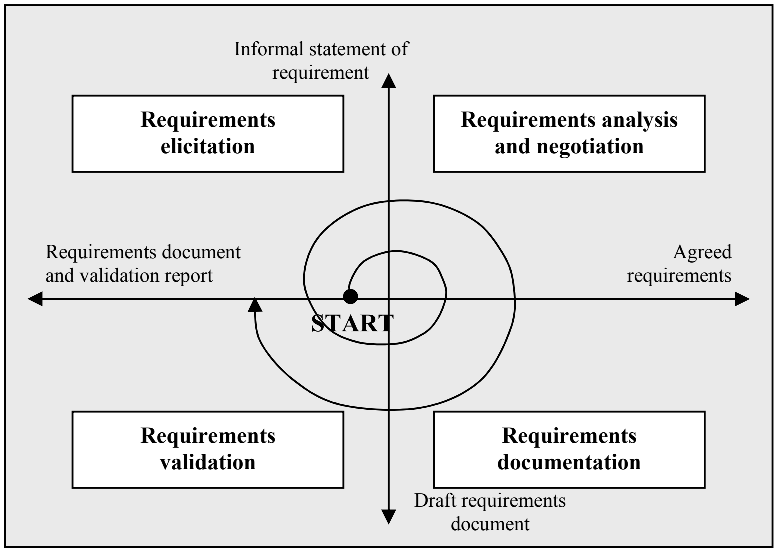

Informal statement of requirement
Requirements elicitation
Requirements analysis and negotiation
Requirements document and validation report
Agreed requirements
START
Requirements validation
Requirements documentation
Draft requirements document

Figure 3.5 A Spiral Model for the RE process adapted from (Kotonya and Sommerville, 1998)

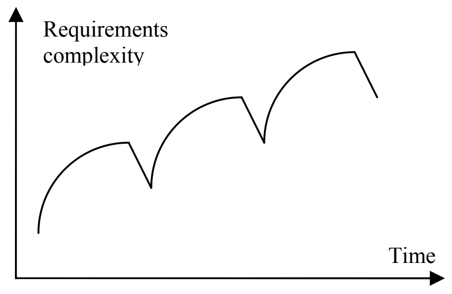


Figure 3.6 Complexity of the RE Process adapted from (Nguyen and Swatman, 2003)

Deming (1986) and Humphrey (1989) argued that process management should be based on support and understanding of this process. The heuristic definition of the RE process (Sindre and Krogstie, 1995; Nguyen and Swatman, 2003) suggests a list of challenging RE process improvements, such as monitoring and managing the RE process, understanding and documenting the RE process, supporting the creativity, education of the RE professionals. Similar challenges for the RE process improvement are suggested in (Kaindl *et al.*, 2002). Monitoring of the RE process, and maintainance of the requirements rationale behind a requirement model (Loucopoulos and Karakostas, 1995; Nguyen and Swatman, 2003) should be supported using RE-tools. The RE-tools would provide the engineer a flexible environment, which promotes requirements model reflectivity and creativity, and supports problem reconceptualisation and specification restructurization.

3.3 Requirements Specification

The term *requirements specification* is used in two ways in the literature. First, it is defined as the *process* or RE activity undertaken to specify requirements (Pohl, 1994; Loucopoulos and Karakostas, 1995; Kotonya and Sommerville, 1998). This definition deals with the *understandability* at the certain time moment.

Second, the requirements specification is characterised as a *document*, which contains a complete description of *what* the system should do without describing *how* it should do it (Davis, 1993; Davis *et al.,* 1993; IEEE std. 830, 1998; Kulak and Guiney, 1998). The second meaning is also supported by the RE definition (Zave, 1997) provided in section 3.1. Here, the requirements specification is the output of the RE process, where the requirements and requirements model is described using a formal language and is agreed among all the stakeholders (Pohl, 1994; Loucopoulos and Karakostas, 1995). Additionally, the requirements specification should hold a list of properties and qualitative characteristics.

This section considers the second meaning of the requirements specification. The section analyses the qualitative properties of the requirements specification and surveys the requirements specification standards.

3.3.1 Requirements Specification Qualitative Properties

The literature (Davis, 1993; Davis *et al.,* 1993; Costello and Liu, 1995; IEEE std. 830, 1998; Khwaja and Urban, 2002; Mora and Denger, 2003) defines metric taxonomies for requirements specification. Typically, the desirable properties are completeness, consistency, correctness, traceability, understandability, verifiability, and maintainability. Whether the requirements are documented in formal or natural language, each requirement should be individually countable, it should be consistent with all other requirements, and carry any annotations in use by the program (Costello and Liu, 1995). Conventions used to specify requirements should be consistent for all specifications within a given level and a traceability matrix should exist for each specification. To increase the accuracy of data collection, it is recommended to store requirements specification electronically in a form that permits individual requirements to be identified and traced to requirements in higher or lower specification level.

Davis *et al.* (1993) describes the comprehensive list of the qualitative properties for the quality of the requirements specification. Further in (Krogstie, 2001a), the requirements specification quality is considered with respect to the goals and means of the semiotic quality framework[3], which divides the quality into physical, empirical, syntactic, pragmatic, semantic, perceived semantic, and social.

The property of physical quality is that a requirements specification should be *electronically stored*. *Reusability* of a requirements specification could be considered through the physical representation. But it also influences other quality types, such as semantic (domains for actual reuse), and syntactic (level of formality reuse).

Empirical quality is understood as the ergonomic representation of the requirements model and it considers *understantability* and *concision* of a requirements specification.

The goal of syntactic quality is syntactic correctness. Syntactic quality is not precisely stated in (Davis *et al.,* 1993), although, some semantic qualitative properties could be reduced in order to analyse the syntactic quality.

Qualitative properties for empirical, semantic, and pragmatic qualities are defined in Table 3.1. Most of the properties concern semantic quality, which has the goals of feasible validity and completeness. Goal of the pragmatic quality is comprehension and it analyses if a requirements specification is *executable* (*interpretable* or *prototypable*), *organised* and *cross-referenced*.

[3] The semiotic quality framework is presented in section 4.1 in detail.

Table 3.1 Qualitative Property Definitions

Qua-lity	Qualitative properties	Definition
Empirical	*understandable*	A reader with a minimum explanation easily comprehends all classes.
	concise	A requirements specification is short as possible without affecting any other quality of it.
Semantic	*complete*	A requirements specification possess the following: 1) Everything that the software is supposed to do is included in the document. 2) Definitions of the responses of the software to all realizable classes of input data in all realizable classes of situations are included. 3) All pages are numbered, all figures and tables are numbered, named, and referenced; all terms and units of measure are provided; and all referenced material and sections are presented. 4) No section is marked "to be determined".
	correct	Every requirement represents something required of the system to be built.
	internally consistent	No subset of individual requirements stated therein conflicts.
	external consistency	No requirement stated therein requirements document conflicts with any already base-lined project documentation.
	precise	Numeric quantities are used whenever possible and appropriate levels of precision are used for all numeric quantities.
	traced	The origin of each requirement is clear.
	annotated by relative importance	A reader can easily determine which requirements are the most important.
	annotated by relative stability	A reader can easily determine which requirements are most likely to change.
	annotated by version	A reader can easily determine which requirements will be satisfied in which version of the product.
	traceable	A requirements specification is written in a manner that facilitates the referencing of each individual statement.
	verifiable	There exists a finite cost effective technique that can be used to verify that every requirement is satisfied by the system to be built.
	achievable	There exists at least one system design and implementation that correctly implements all the requirements stated in the requirements document.
	design-independent	There exists more than one system design and implementation that correctly implements all the requirements stated in the requirements document.
	at right level of detail	It is described how the requirements document is being used.
	unambiguous	Every requirement stated therein has only one possible interpretation.
	modifiable	A requirements specification structure and style are such that any changes can be made easily, completely and consistently.
	not redundant	The same requirement is not stated more than once.
Pragmatic	*executable (interpretable, prototypable)*	There exists a software tool, capable of inputting the requirements specification and providing a dynamic behavioural model.
	organised	A requirements specification contents are arranged so that readers can easily locate information, and logical relationships among adjacent sections are apparent.
	cross-referenced	Cross-references are used in the requirements specification to relate sections containing requirements to other sections.

Social quality is dealing with the agreement about requirements specification. This quality type is not analysed in (Davis, 1993; Davis et al., 1993). However, the importance of the argumentation tool support through the requirements specification is mentioned by Krogstie (2001a).

3.3.2 Requirements Specification Standards

International standards (NASA-std-2001-91, 1991; IEEE std. 830, 1998) describe how the requirements specification should be prepared. The basic difference between the standards is that they suggest own ways to represent the requirements model. However, the main requirements specification structure remains the same. The requirements standards 1) represent the requirements model at different level of requirements abstraction and in different viewpoints; 2) they separate the requirements representation into functional and non-functional requirements and 3) specify requirements priorities and traceable relationships. This section surveys the IEEE standard 830-1998 in some details.

The IEEE standard 830-1998 recommends that requirements should have three main sections (Figure 3.7). The first section introduces the purpose and scope of the requirements specification. This section also provides the glossary of definitions, acronyms and abbreviations. The second section is concerned with the description of the factors that affect the intended system and its requirements.

The third section (Figure 3.8) provides templates to describe specific requirements. The organisation of the third section may differ depending on the requirements approach and modelling perspective that are used. Each requirement should include a unique identifier and a measure of priority. Essential requirements must be market as such. References that trace the requirements back to the user requirements document must accompany each requirement. Any other requirements source must be stated. Each requirement must be verifiable. The standard requires that functional requirements would be structures top-down in this section. Non-functional requirements can appear at all levels of the hierarchy and functions, and by the inheritance they apply to all functional requirements below. Non-functional requirements may be attached to functional requirements by cross-references or by physically grouping them together in the specification.

In order to support and improve the preparation of the requirements specification, a powerful tool support could be useful. The application of the targeted RE-tools could be a challenge to improve the preparation of the requirements specification. First, an RE-tool would allow reuse of the knowledge gathered in other projects. It would also guide a user in order to prepare the requirements specification, and to maintain the qualitative characteristics of the requirements specification.

Finally, the RE-tool would suggest the requirements specification standards or means to define an organisational standard.

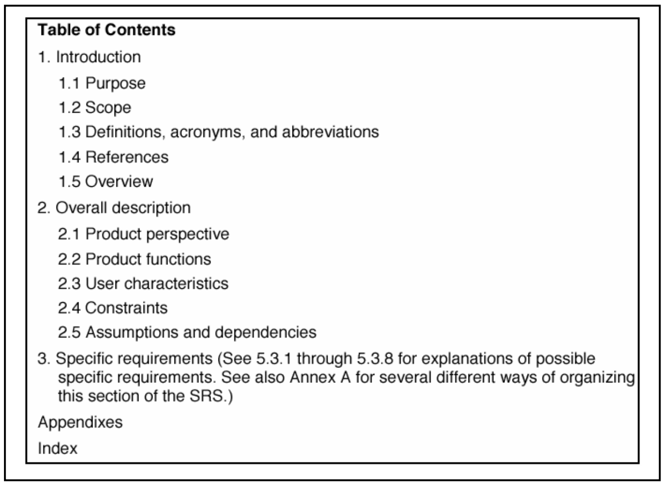


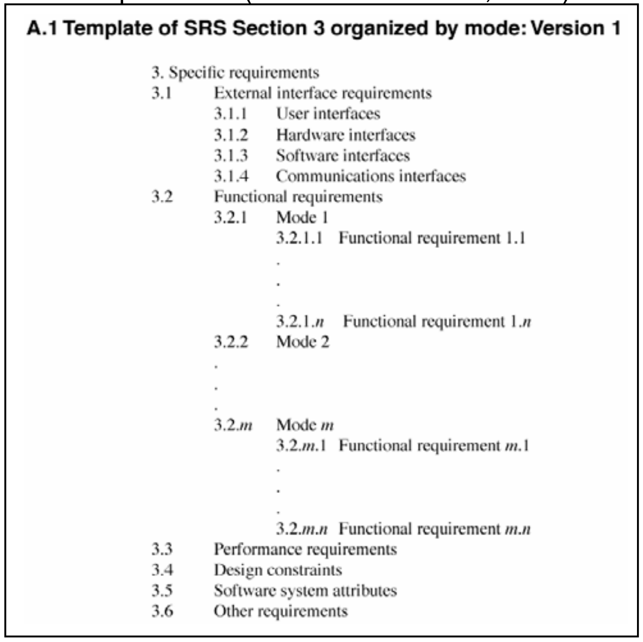
Table of Contents

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, acronyms, and abbreviations
    - 1.4 References
    - 1.5 Overview
2. Overall description
    - 2.1 Product perspective
    - 2.2 Product functions
    - 2.3 User characteristics
    - 2.4 Constraints
    - 2.5 Assumptions and dependencies
3. Specific requirements (See 5.3.1 through 5.3.8 for explanations of possible specific requirements. See also Annex A for several different ways of organizing this section of the SRS.)

Appendixes

Index

Figure 3.7 Requirements Specification Outline adapted from (IEEE standard 830, 1998)



A.1 Template of SRS Section 3 organized by mode: Version 1

```
3. Specific requirements
3.1      External interface requirements
         3.1.1    User interfaces
         3.1.2    Hardware interfaces
         3.1.3    Software interfaces
         3.1.4    Communications interfaces
3.2      Functional requirements
         3.2.1    Mode 1
                  3.2.1.1  Functional requirement 1.1
                  .
                  .
                  .
                  3.2.1.n  Functional requirement 1.n
         3.2.2    Mode 2
         .
         .
         .
         3.2.m    Mode m
                  3.2.m.1  Functional requirement m.1
                  .
                  .
                  .
                  3.2.m.n  Functional requirement m.n
3.3      Performance requirements
3.4      Design constraints
3.5      Software system attributes
3.6      Other requirements
```

Figure 3.8 Template for the Requirements Model Specification adapted from (IEEE standard 830, 1998)

3.4 Empirical Studies of Requirements Engineering

Empirical surveys of the RE practice can be divided into several groups: general analysis, which concern only RE issues (Chatzoglou, 1997; Curtis, Krasner and Iscoe, 1998; El Emam and Madhavji, 1995; Hofmann and Lehner, 2001), works which analyse RE for a particular application domain (Karlsson *et al.*, 2002), and studies, which include geographical coverage (Lubars, Potts, and Richter, 1993; Nikula, Sajaniemi, and Kälviäinen, 2000; Ekremsvik and Tiset, 2003; Matulevičius, 2004a).

Chatzoglou (1997) presents a survey of 107 projects developed in 74 different organisations, in which an attempt is made to identify the relationships between developers, project characteristics and different aspects of the RE process. Curtis, Krasner and Iscoe (1998) investigate how domain knowledge, fluctuating and conflicting requirements, and communication breakdowns influence the software productivity and quality in 17 large software development projects. El Emam and Madhavji (1995) perform a field study, which has a purpose to formulate recommendation to practitioners for improving the RE processes.

Based on field study of 15 RE teams, Hofmann and Lehner (2001) identify RE practices that contribute to project success, particularly in terms of team knowledge, resource allocation and RE process. Karlsson *et al.* (2002) present an empirical study on RE for software packages using semi-structured interviews. Seven employees at five software companies with a market-driven development focus were interviewed.

Many empirical studies report about the industrial RE practice in particular geographical areas. Ten organisations in the United Stated are interviewed in order to find out how requirements are defined, interpreted, analysed, and used (Lubars, Potts, and Richter, 1993). Nikula, Sajaniemi, and Kälviäinen (2000) performed interview based investigation in 12 Finish software development organisations. Two other studies use Web-based questionnaires to collect information in Norwegian (Ekremsvik and Tiset, 2003) and Lithuanian (Matulevičius, 2004a) software development organisations.

A comparison of the RE processes and activities highlighted in the mentioned empirical studies, is performed based on the NATURE framework dimensions (Pohl, 1994, Pohl, 1996) and RE approaches (Loucopoulos and Karakostas, 1995; Kotonya and Sommerville, 1998). Table 3.2[4] shows the most general RE activities used in industrial companies. The main activities of the representation dimension, are requirements description using informal, semiformal, formal languages, and requirements traceability. Activities analysed in the agreement dimension, include collaboration, stakeholder and role definition, and communication means. Finally in

[4] See Appendixes A and B for the overall results of the survey (Matulevičius, 2004a).

the specification dimension the activities are requirements reuse, requirements documentation and preparation of final requirements specification.

Table 3.2 RE Activities Used in Industry adapted from (Matulevičius, 2004a)

Dimension	Feature	Activity	Importance
Representation dimension	Requirements description	Requirements description, using informal language and conversations.	High
		Requirements descriptions, using semi-formal definitions.	High
		Requirements descriptions, using formal definitions.	Medium
	Informal language	Natural language.	High
	Semiformal languages	State charts, DFD, Use Case diagrams, Use Case templates, UML, ER diagrams.	Medium
	Requirements traceability	We keep different versions of requirements specification.	High
		We keep information about requirements source.	High
		We keep traceable relationships with requirements specification and design.	Medium
Agreement dimension	Requirements discovery	Analysing existing (similar) software.	High
		Performance of surveys (market, stakeholders).	High
		Meetings and conversations with (potential) software stakeholders.	High
		Reuse of domain specific requirements from predefined repositories.	Medium
	Roles and stakeholders	Project managers.	High
		Software developers.	High
		Marketing personnel.	High
		Domain experts.	High
		End-users.	High
		Training and user support staff.	Medium
	Requirements negotiation and collaboration	Means of brainstorm.	High
		Means of discussion/negotiation.	High
		Maintenance of rationale behind requirements.	High
	Requirements attributes	Sorting according to attributes.	Medium
		Definition of views according to attributes.	Medium
		Definition of requirements attributes.	Medium
	Requirements groups	Functional requirements.	High
		End-users requirements.	High
		Architectural requirements.	Medium
Specification dimension	Requirements reuse	Selection and extraction of domain specific requirements from other projects.	Medium
	Requirements process documentation	Reports and documentation, which help to understand requirements.	High
		Reports and documentation of representations.	High
		Reports and documentation about requirements agreement.	High
	Requirements specification	Standard for requirements specification, defined by organisation.	High

3.4.1 Representation Activities

The requirements representation dimension deals with representation forms and relationships between them. Table 3.2 shows high importance of informal and semi-formal specification languages. The most important representation language is found to be a natural language. There is an interesting difference between semi-formal and formal languages: semi-formal languages are used for the requirements representation, but there is no language about which it would be would be reach a common agreement. Formal languages are used quite rarely.

Similar use of natural language was observed in (Nikula, Sajaniemi and Kälviäinen, 2000; Karlsson *et al.*, 2002; Ekremsvik and Tiset, 2003; Firesmith, 2003b). Semi-formal techniques as observed in other studies are ER diagrams (El Emam and Madhavji, 1995; Ekremsvik and Tiset, 2003; Lubars, Potts and Richter, 1993; Davies, Green and Rosemann, 2003), state charts (Hofmann and Lehner, 2001; Karlsson *et al.*, 2002), data flow diagrams (Karlsson *et al.*, 2002; Ekremsvik and Tiset, 2003; Davies, Green and Rosemann, 2003), UML (Karlsson *et al.*, 2002), and Use Case templates (Ekremsvik and Tiset, 2003). In (Lubars, Potts and Richter, 1993; Hofmann and Lehner, 2001) the requirements were observed to be described using knowledge and quality function deployment matrices. Formal languages are not used for RE in (Lubars, Potts, and Richter, 1993), but (Matulevičius, 2004a) observed support for formal languages, but exact formal languages were not properly identified.

Findings indicate high importance of relationships between requirements definitions, requirements source, and through continues requirements development. Several companies analysed in (Lubars, Potts and Richter, 1993; Hofmann, and Lehner, 2001) maintain a traceability matrix to track a requirement from its origin through its specification to its implementation.

3.4.2 Agreement Activities

The requirements agreement dimension deals with agreement about requirements model. Common practice to reach agreement about requirements model is to use face-to-face negotiation and discussions. The communication ensures that requirements are interpreted properly (El Emam and Madhavji, 1995; Hofmann and Lehner, 2001). However, communication gaps among stakeholders are observed in (Curtis, Krasner, and Iscoe, 1998; Karlsson *et al.*, 2002). The knowledge sharing and change facilitation is observed by Chatzoglou (1997) and Curtis, Krasner and Iscoe (1998). The maintenance of requirements rationale in customer-specific projects is not worth the effort (Lubars, Potts and Richter, 1993), but in market-driven projects the requirements rationale for decisions and assumptions should be recorded. But Matulevičius (2004a) findings observe that the rationale is kept and it helps to solve the emerging conflicts about the requirements.

The most important requirements agreement activities are meetings, surveys, and analysis of similar systems. These activities are considered in (Lubars, Potts, and Richter, 1993; El Emam, and Madhavji, 1995; Chatzoglou, 1997; Hofmann and Lehner, 2001; Matulevičius, 2004a). Hofmann, and Lehner (2001) and Karlsson *et al.* (2002) argue about the requirements databases and document analysis as the requirements source.

Sorting and viewpoint definitions support the involvement of different stakeholders: project managers, software developers, domain experts, marketing personnel and end users. Experienced project manager should be able to use the RE tools, and have knowledge of the system development process (El Emam and Madhavji, 1995; Chatzoglou, 1997). Software developers play an important role during requirements analysis and validation (Karlsson *et al.*, 2002). Users should participate in the project from the very beginning (Lubars, Potts and Richter, 1993; El Emam and Madhavji, 1995; Chatzoglou, 1997; Hofmann and Lehner, 2001). Domain experts are the most regarded participants (Lubars, Potts and Richter, 1993; El Emam and Madhavji, 1995; Chatzoglou, 1997; Curtis, Krasner and Iscoe, 1998; Hofmann and Lehner, 2001). In general domain experts understand a domain as well as the users, but domain experts can often correct the users.

In (Matulevičius, 2004a) most of the software development companies are departments of some international organisations. The central offices consider software marketing and support requirements. Results indicate that companies are dealing with functional and architectural requirements and do not analyse the non-functional ones.

3.4.3 Specification Activities

The requirements specification is written according to standards and guidelines. However the software development companies often prefer organisational standards to standards agreed by international communities (Ekremsvik and Tiset, 2003; Matulevičius, 2004a). In order to support the final specification, a big amount of documentations is needed (Matulevičius, 2004a). However, the creation of requirements documents is not self-evident (Lubars, Potts and Richter, 1993; Nikula, Sajaniemi and Kälviäinen, 2000). Projects that are developing products for a potential market, tend not to document requirements in much detail, if at all, and seldom follow any standards other than internal corporate or division-wide guidelines.

Selection and extraction of domain specific requirements from other project is indicated with weak importance (Lubars, Potts and Richter, 1993; Matulevičius, 2004a). The reuse activities are performed manually, in the best case using “copy-paste” typing operations.

3.4.4 RE Share of Total Development Time

RE is a new and not mature activity (Kaindl *et al.*, 2002). Matulevičius (2004a) analysis shows that the time spent for RE is 25-50% of the whole development (Figure 3.9). The result corresponds to other findings, where the average amount of RE time equals to 38.6 % of total project duration (Hofmann and Lehner, 2001), and which argue, that companies invest heavily in terms of time (Chatzoglou, 1997). However the surveys might also consider and include not only the process of new software development, but also the software support, which takes part after software release.

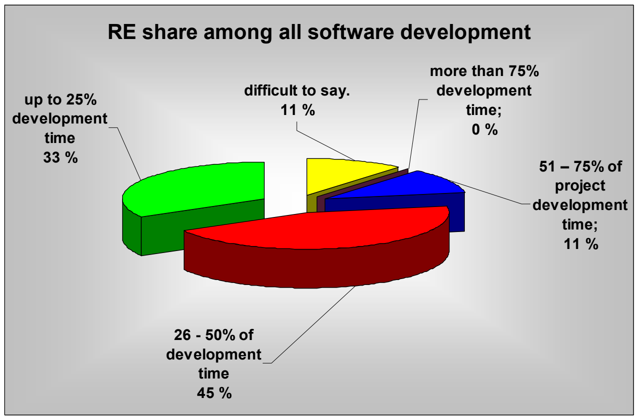


Figure 3.9 RE Share Among all Software Development in Lithuania

3.4.5 Automated Support for RE

In Lithuanian and Norwegian companies the mainstream of RE practice relies on word processors and modelling tools rather than the RE tools (Figures 3.10 and 3.11). Other surveys also report the lack of the automated RE support. The most commonly used tools are text editors and spreadsheets (Lubars, Potts and Richter, 1993; Chatzoglou, 1997; Ekremsvik and Tiset, 2003), web sites accessible to all stakeholders (Hofmann and Lehner, 2001), and Visio (Davies, Green and Rosemann, 2003). Most commonly used tools in Finnish companies (Nikula, Sajaniemi and Kälviäinen, 2000) are configuration management and testing tools where only some of the companies are using CASE (modelling) tools and none of the interviewed organisations apply RE-tools in their work (Figure 3.12). Some projects combine text editors and database management systems for traceability, but such integrations can only be used if configuration management policies are enforced (Lubars, Potts and Richter, 1993).

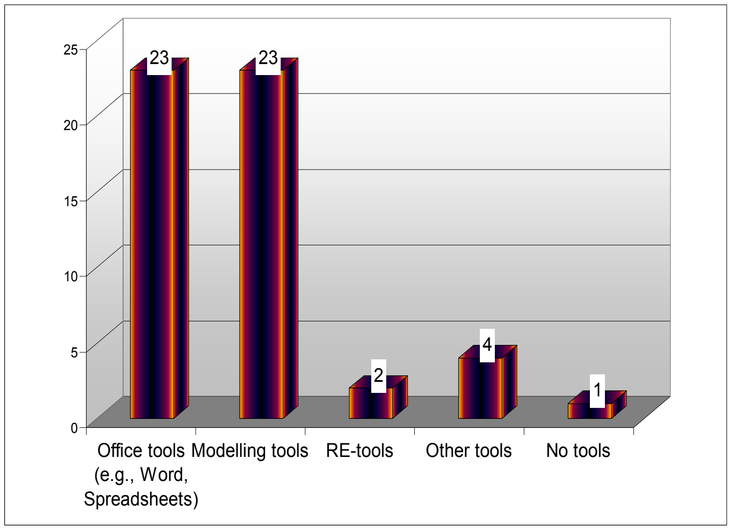


Figure 3.10 Most Frequently Used Tools for the RE Activities in Lithuania (28 organisations)

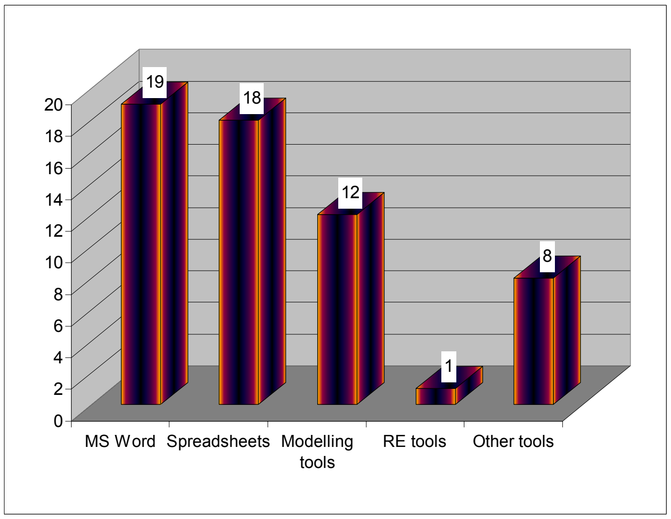


Figure 3.11 Most Frequently Used Tools for the RE Activities in Norway (22 organisations)

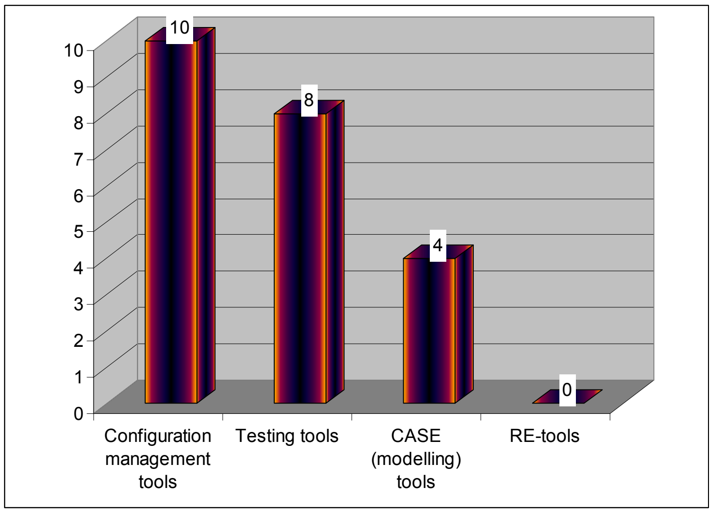


Figure 3.12 Most Frequently Used Tools for the RE Activities in Finland (12 organisations)

Reasons for not using RE tools include financial causes (Matulevičius, 2004a), such as high RE tool price (31%), assumed low return on investment (21%), companies consider it difficult to adapt RE tools to their organisational needs (38%). The Norwegian practice (Ekremsvik and Tiset, 2003) reports similarly the large tool costs and non-awareness of the existing RE tools as the main reasons for not using them. A lack of well defined RE process and a lack of team training in the selected tools were observed as causes of the non-sufficient support for the RE activities (Hofmann and Lehner, 2001). In order to adopt a tool, an infrastructure must be set to support the tool and a company must be willing to invest in such an infrastructure in the environment (El Emam and Madhavji, 1995). This includes personnel training (Karlsson *et al.,* 2002), tool support groups (Davies, Green and Rosemann, 2003), and funding for the tool implementation (El Emam and Madhavji, 1995). However, the management of such companies usually have unrealistic expectations (e.g., immediate pay-off).

3.5 RE Process Improvement

In Chapter 2 different process improvement maturity models are surveyed. The RE process maturity level is only one of the factors which affect the quality of the final requirements specification. Other factors are the abilities and the experience of the people involved in the process, and the novelty, difficulty and size of the problem as well as the time and resources available. An organisation possessing such qualities will use methods and techniques for RE, and will have defined standards for requirements documents, and requirements descriptions. Such an organisation may use RE-tools to support process activities. It will have management policies and procedures in place to ensure that the process is followed and may use process measurements to collect information about the process to help assess the value of process changes.

The objectives of process improvement may include quality improvement, schedule reduction and resource reduction (Kotonya and Sommerville, 1998). *Quality improvement* means that the outputs produced by the process are of higher quality. In case of RE the requirements specification contains fewer errors, is more complete and better reflect the real needs of system stakeholders. *Schedule reduction* defines the process output which is produced more quickly. In the case of requirements, this means the less time is needed to produce the final version of the requirements specification. *Resource reduction* means that fewer resources, such as staff time, are needed to execute the process. Therefore, a smaller team of engineers can produce the final requirements document.

The techniques to improve the process can be characterised in three different ways (Boehm, 1999): work avoidance (reuse of software artefacts), working smarter (apply different methods to execute the process), and working faster (use the software tools). They are general improvement dimensions. A specific company might have more specific improvement objectives such as more reuse of requirements across different systems, and more involvement of end-users in the RE process. Taking into account the complexity of the RE domain, the ways of improving the RE process could depend on the organisational environment.

The goals of potential improvement (Kaindl *et al.*, 2002; Nguyen and Swatman, 2003) are an introduction of new methods for RE, development of techniques for training users, supporting maintenance of the software products, reuse of methodology and techniques in development, working out techniques for interviewing and requirements negotiation, development of standards both for RE process and documents.

Curtis, Krasner and Iscoe, (1998) describe RE process success factors such as knowledge integration, change facilitation, and broad communication. Hofmann and Lehner (2001) identify the best practice of RE being involvement of customers and users throughout RE, assigning skilled project managers and team members, providing specification template, developing complementary models together with prototypes, maintaining traceability, and using peer reviews and scenarios to validate requirements. El Emam and Madhavji (1995) suggests seven factors for RE process improvement: package consideration, managing the level of detail of functional process models, examining the current system, user participation, managing uncertainty, benefits of computer aided system engineering (CASE) tools, and project management capabilities. Chatzoglou (1997) shows that the requirements capture and analysis are iterative. Allocation of the resources (time, efforts, cost, and people) depends on the project type, the team members' and users' attitudes, and the projects management. Karlsson *et al.*, (2002) suggest to improve the RE process by solving both the problems of communication gaps between marketing and development, and of balancing the influence of marketing and development on requirements decisions.

Lubars, Potts, and Richter (1993) report on use of organisational solutions rather than technological ones and RE-tools instead of general-purpose tools. Nikula, Sajaniemi, and Kälviäinen (2000) indicate that the key software development needs are the development of RE process adaptation, RE process improvement and automation of RE practices in small and medium-size enterprises. In (Matulevičius, 2004a) the study concludes with challenging issues, which include the necessity to define the RE process in a mature way, the need to adopt requirements reuse methods, and the need for the automated RE support with the repository-based RE tools.

Finally, the following list of possible RE process improvements could be highlighted from the empirical studies:

– Define RE process in a more mature-complete way (Lubars, Potts, and Richter, 1993; Chatzoglou, 1997; Nikula, Sajaniemi, and Kälviäinen, 2000; Hofmann and Lehner, 2001; Matulevičius, 2004a).
– Train employees in more advanced tool and method usage (Hofmann and Lehner, 2001; Karlsson *et al.*, 2002);
– Improve requirements reuse by adopting requirements repositories (Matulevičius, 2004a). Reuse of requirements specification would be effective, if requirements repositories are introduced. Repository tends to support storage of requirements metadata and traceability when relevant requirements are modified.
– Improve stakeholder communication (Lubars, Potts, and Richter, 1993; El Emam and Madhavji, 1995; Hofmann and Lehner, 2001; Karlsson *et al.*, 2002);
– Improve automated RE support (Lubars, Potts, and Richter, 1993; El Emam and Madhavji, 1995; Hofmann and Lehner, 2001; Karlsson *et al.*, 2002; Ekremsvik and Tiset, 2003; Matulevičius, 2004a).

The market suggests RE-tools which allow integration with other modelling environments and support RE activities such as scope management, version and configuration control, quality assurance and quality control. The requirements specifications of different types, contents, and level of details could be automatically generated from the requirements repositories using appropriate selection criteria and templates. A full support of RE processes by RE-tools is essential, as the size and the complexity of existing current information systems require collaboration and process co-ordination of different participants in the maintenance of the RE processes. Successful acquisition of an RE-tool to the organisation could lead both to improvement of the quality of the RE process and product.

3.6 Requirements Engineering Tools

Requirements engineering tools (RE-tools) are software tools that provide automated assistance during the RE process and support the RE activities (Matulevičius, 2004b). According to this RE-tool definition, only tools that support the entire RE process are considered to be RE-tools. For example, word processors support documentation, and modelling tools provide facilities for requirements modelling. However, these tools are not RE-tools because they do not maintain the whole RE process. Such software tools could be used for different RE activities, but their output should be included into the RE process.

The literature and system developers usually call these tools as requirement management tools. Kotonya and Sommerville (1998), and Ferdinandi (2002) describe requirements management as a part of the RE process and manage changes of system requirements. We use the term *requirements engineering tools* instead of *requirements management tools*, as vendors usually use this term. The functionality of those tools covers RE activities, such as elicitation, analysis, negotiation, and validation, i.e., not only management of project changes.

3.6.1 Tool Weaknesses

Modern information systems development cannot be accomplished without reasonable tools support. (Harrison, Ossher, Tarr, 2000) and RE is one on the key phases of the systems development. Many RE-tools are described as computer aided system engineering (CASE) tools which are expected to provide task related support for software developers in analysing, designing and implementing a set of software systems or their components according to a method. The method can be defined as a language and a set of rules, which define by whom, when and how such representations are derived and/or used. In the 80's there was a great optimism about usage of such tools, software development companies invested much in acquiring CASE tools in their projects. However, surveys (Orlikowski, 1993; Lending and Chervany, 1998) report that the use of CASE tools is not successful used in the organisations. Although the CASE tools support better management of experience, they tend to change the organisational working processes (Orlikowski, 1993), which are not very acceptable by the system developers. Therefore Lending and Chervany (1998) recommend the increase of the perception of the tool usage by training and reinforcement. This training should include concrete examples of how the CASE tool produce beneficial results for the system developers themselves.

The list of CASE tool weaknesses to support the development process itself is identified by Kelly, Lyytinen, and Rossi (1996). Weaknesses are divided into the following aspects:

- lack of mechanism for integrating sets of methods while maintaining consistency between various models,
- lack of support of multiple users to create, modify and delete sets of partly overlapping model instances,
- inadequate catering for multiple representational requirements raging from fully diagrammatic to fully textual or matrix representation. These are dictated by different method families.
- failure to provide consistent mapping mechanism between different representational paradigms.
- lack of flexibility and evolvability in method support ranging from syntactic variation in methods to crafting totally new method components.
- insufficient catering for different information-related needs of a diverse set of stakeholders.

Being a subset of CASE tools, RE-tools also inherited weaknesses to support RE processes (James, 1996; Matulevičius and Strašunskas, 2003). Most of the RE-tools in use are standalone applications and do not provide any (or provide weak) possibilities for collaboration work. The problem is a lack of collaboration tools - brainstorming, negotiation, rationale, awareness. None of the RE-tools is ideally suited for use by a multidisciplinary, distributed team where the stakeholders have diverse skills and needs. Collaborative work is especially important at the starting stages of requirement management. Possibilities for geographically distributed teamwork can save time and financial resources.

RE-tools also lack reuse possibilities and functionality. In some cases RE-tools have a possibility of setting associations between different projects, but common practice of reuse in RE-tools is done by "copy-paste" functions. Solution for reuse support could be the common domain database, which could be used for knowledge of the same product family and for improving the RE process. Repository for common requirements and traceability between requirements and design would benefit in consistent change integration for product lines.

RE-tools are specially designed for use by skilled specialists who are proficient both in software engineering methods and the functionality of the tool itself. Because of the complex functionality RE-tools are not comfortable for non-technical stakeholders Duitoit and Paech, 2001; Lang and Duggan, 2001; Urquhart, 2001).

Usually an RE-tool deals with informal (in some cases semi-formal) representations of RE processes and system requirements, but usually the representation in formal languages is not supported. This weakness could be explained by the market driven RE-tool development. The RE-tool vendors do not expect to sell more RE-tool licenses if their tool would support the formal requirements representation. On the other hand the tool users need to be tough the use of the formal techniques in order to automate the system development. In the meantime the development of formal techniques with a benefit of automating the system development is more academic-oriented, and is not widely accepted in the practice.

Additional features increase the functionality of the tools, but often users do not use all functionality because it is too complex to get familiar with all features. Such over-functionality can also make tool more difficult to use. Fitting RE-tools to an environment remains problematic because companies employ different methods. The tools vary in their level of support for RE activities and it is difficult to choose the right one among available commercial tools. Evaluations of tools differ depending on the purpose of tool usage: in the industry or in a university course. Software development companies focus on different RE-tools features relative to importance depending on the development methods used by the companies. Thus adding non-functional evaluation criteria like tool adaptation to the work process, tool reliability, performance maintenance, usability, as well as purchase, upgrade, support costs, perceived usefulness and perceived ease of use we get a wide spectrum of features, which are relevant for modern RE-tools.

3.6.2 Overview of RE-tools

RE-tool surveys (LaBudde, 1997; Wieger, 1999; Lang and Duggan, 2001; INCOSE, 2002; Matulevičius and Strašunskas, 2003) at the certain time intervals have little long-term value, as the RE-tool market is constantly changes, new tools appear, or existing RE-tools gets new functional and non-functional features. The purpose of this section is to illustrate the variety of the existing commercial RE-tools in the market (Table 3.3). This section does not emphasise any of these tools, but it just presents the main functionality of the RE-tools according to the vendor documentations and some surveys. The RE-tools here are listed in an alphabetical order. The main source of this survey is Volere Web site[5]; however the existing other tool surveys (e.g., Wieger, 1999; Lang and Duggan, 2001; INCOSE, 2002) could have different attitude and perspectives. Some of these commercial RE-tools are tested in the case studies in Chapter 7 for the validation purposes of the RE-tool evaluation approach (Matulevičius, 2004a; Matulevičius, 2004b; Matulevičius, 2004c, Matulevičius, Karlsson and Sindre 2004).

[5] http://www.volere.co.uk/tools.htm

Table 3.3 Survey of RE-tools

RE-tool	Vendor	Description
Analyst Pro Version 3.5	Goda Software, Inc.	Analyst Pro is a tool for requirements management, tracing and analysis. Analyst Pro utilizes a Configuration Management methodology that enables the development staff to analyse the impact of change on requirements and component assets. The tool incorporates features of importing requirements, requirements sharing, requirements change management, requirements assignment, and requirements graphs.
AxiomSys Version 6.0	Structured Technology Group, Inc.	AxiomSys Version 6.0 builds a structured analysis model of the system using the Yourdon structured analysis method (Yourdon, 2000) as well as providing full integration and comprehensive support for the Hatley-Pirbhai real time extensions and architecture modelling concepts (Hatley and Pirbhai, 1998). AxiomSys provides mechanisms to trace how and where each requirement is fulfilled, and validates the entire model for consistency and logical integrity. Also it provides automated documentation ability as well as providing report templates.
CaliberRM	Borland	CaliberRM facilitates communication among project teams by providing centralized requirement data to distributed team members and allowing documented discussions about requirements as well as allowing project teams to fully define, manage and communicate changing application or system requirements. Changes made to requirement data such as traceability, document references, status, user responsibility and more are recorded in Caliber-RM's central repository. CaliberRM keeps team members up to date on changes made to requirements by automatically notifying responsible individuals of the changes. CaliberRM also enables team members to identify potential requirement problems by highlighting ambiguous and commonly used terms defined in a shared glossary
Catalyze	SteelTrace	Catalyze takes a structured view of requirements breaking them into functional and non-functional. Catalyze automatically generates graphical flows directly form text and maintains text and graphics in lockstep. SteelTrace offers native integrations from Catalyze into leading UML modelling tools Rational Rose and Borland Together Solo and Control Centre. Customisable MS Word profiles allow the same Catalyze project generate to multiple document formats that are round-trippable back to the Catalyze project.
C.A.R.E. version 2.0	SOPHIST Ltd	C.A.R.E. supports the method of object engineering developed by SOPHIST. This method integrates the components of object oriented analysis, acceptance criteria, prototyping, requirements engineering and linguistic methods into a flexible toolkit. C.A.R.E. 2.0 utilizes a security strategy based on Lotus Notes which defines how users are allowed to edit, modify or read requirements documents.
CART	Salford Systems	CART is a decision tree tool that automatically sifts large, complex databases, searching for and isolating significant patterns and relationships. The knowledge is then used to generate reliable, easy-to-grasp predictive models for applications such as profiling customers, targeting direct mailings, detecting telecommunications and credit card fraud, and managing credit risk.
Clear Requirements Workbench	LiveSpecs Software	Clear Requirements Workbench supports four detailed specification techniques (glossaries, action contracts, test procedures, and precise use cases) for the description of definitions, behaviour, and usage.
CORE version 3.1	Vitech Corporation	CORE enables the user to extract the originating requirements from the source documentation, analyse them for completeness, consistency and testability, and trace each requirement to a behavioural model which describes the interactions and process sequences. The system behaviour is represented by user-selectable graphical views which capture the system control logic and data flow in an integrated manner. The user allocates the system functional models to a physical system architecture. Verification and validation are available in CORE to execute and test the models to establish system performance and resource usage.

Continuation of Table 3.3

Cradle Version 4.0	3SL - Structured Software Systems Ltd	Cradle provides a requirements capture facility that scans customer statements and extracts requirements, assumptions and/or domain knowledge, creating cross references back to the original document. When new versions of such documents are registered, Cradle finds the differences and provides an impact assessment. Cradle is fully integrated with Word, Excel and PDF, as well as other text formats. Requirements can be linked to a wide variety of UML, use case, functional, behavioural, dynamic and architectural models, grouped within multiple model domains. Requirements can be allocated to use cases, functions, business processes, operational sequences, which in turn can be allocated to functions, classes etc within components of equipments in multiple candidate architectures. Performance assessment and budget aggregation and apportionment within and across these architectures are fully supported, together with the means to develop these models to hardware and software, including the generation and reverse engineering of source code.
Criterium DecisionPlus	InfoHarvest	DecisionPlus supports the decision making process. The Decision Scores view displays how the alternatives are compare based on the criteria and values. The Contributions view helps to understand why the alternative outranked the others. The Alternatives Scatter Plot helps you see how the alternatives are distributed relative to their scores on the various lowest criteria and how the scatter of alternatives correlates with high decision scores. A visual, structured approach to decision making simplifies the process, helping to tackle large and complex decision opportunities and select the best choice. The tool provides the analysis and reporting instruments to justify the decision.
DOORS/ERS	Telelogic	DOORS handles requirements as discrete objects. Each requirement can be tagged with an unlimited number of attributes allowing easy selection of subsets of requirements for specialist tasks. DOORS includes an on-line change proposal and review system that lets users submit proposed changes to requirements, including a justification. DOORS offers unlimited links between all objects in a project for full multi-level traceability. Impact and traceability reports as well as reports identifying missing links are all available across all levels or phases of a project life cycle.
ERGO	TEC Decision Support Systems	ERGO is a single user decision support environment designed to help users organise their decision criteria and their related priorities, as well as vendor evaluation information. This allows the user to analyse results of the evaluation process and create what-if scenarios. ERGO decision models has a hierarchical structure with no practical limitations on the structure.
EasyRM version 1.05	Cybernetic Intelligence GmbH	EasyRM is a CASE tool covering the initial stages of the software project development, including glossary, requirement and reference documentation management. EasyRM provides users with the following facilities: creation, description, modification and progress tracking of project requirements, classification of requirements, specification of relationships between requirements, maintenance of semantic links from requirements to glossary, maintenance of traceability links from requirements to information sources where these requirements have originated.
Focal Point	Focal Point	Focal Point is an Internet-based platform for market-driven product development, a decision-support system for the core business of product developing companies. The emphasis of the tool is collaboration between the parties interested in gathering the requirements. The tool also supports gap analysis, competition analysis with market definition, and requirements prioritisation with product, release and resource planning.
GMARC	Computer System Architects Ltd.	GMARC provides rapid elicitation of requirements using a generic approach to enhance re-usability and encourage standardization across projects. Features include traceability of requirements hierarchically, historically and inter-task as well as inter-document, identification and correction of subjective requirements and automatic interchange of requirements information between models and specifications. The GMARC tool produces and displays data flow diagram representations of the functional aspects of the specification and then animates these diagrams to verify the dynamic viability of the system.

Continuation of Table 3.3.

IRqA	TCP Sistemas & Ingenieria	IRqA assists in early stages of the software development process. IRqA provides requirements elicitation, analysis, specification and management. Requirements can then be classified according to criteria such as priority, type, status, or any other criteria, using management facets. Requirements analysis and refinement is carried out by building a problem domain model, which allows users to represent the concepts in a certain domain. Two kinds of models are supported: object oriented models, and entity relationship models.
Mesa/AD	Mesa Systems Guild, Inc	Mesa/AD automates the transitions from real-time structured analysis to object-oriented and structured designs. This automation is based on published, widely available methodologies.
METIS version 3.1	Computas AS	METIS provides a visual representation of the linkages and relationships between a company's various functions and its technology infrastructure. A METIS model is typically based on a template and consists of objects and relationships logically grouped in containers.
MooD 2001	The Salamander Organisation	MooD 2001 provides techniques to support the creation and alignment of strategic business models, process design maps, and their implementation through people and systems. MooD enables integrated performance management reporting and traceability at all levels. Using business objects to capture required behaviour, MooD creates OO system requirement models for input to software design. MooD is a repository-based tool that supports multi-user access to organisational models and integration with other software through Windows Open Standard Architecture (OLE and ODBC).
Objectiver	Objectiver	Objectiver relies on KAOS (Bertrand *et al., 1998*), a goal driven methodology and enables users to have a global overview on the system and a systematic link between all the models representing the system. Analysts have the possibility to draw diagrams and to define concepts and relationships over those concepts. Diagrams can be explained with text documents including references to concepts elicited in the diagrams. All these pieces of information can then be put together to generate a requirements document compliant with predefined standards.
OnYourMark Pro version 2.0	Omni-Vista, Inc.	OnYourMark Pro allows users to do intelligent trade-off analysis between requirements, schedules, and development costs. Users enter the difficulty of each requirement (in terms of feature points, function points, or person-days) and OnYourMark Pro automatically computes the total labour effort and ideal schedule.
Rational RequisitePro	IBM	Rational RequisitePro integrates Microsoft Word and a requirements database. Software project teams can gather, enter and manage requirements within your documents or in a database. Automated traceability tracks requirements and changes through implementation and testing. Related requirements can be linked together, so that as changes occur to one requirement users can easily see its impact on other related requirements. RequisitePro allows organising, prioritising, and tracing relationships between requirements.
RDT version 3.0	Igatech	RDT includes a parser that imports text documents then identifies requirements by key words and structure. The tool provides functionality for deriving, allocating and assigning requirements and acceptance test procedures. Requirements can be traced from top level requirements down to the lowest level requirements. The tool is able to classify/categorize requirements during identification using requirements attributes. In addition the tool provides capabilities to capture architecture, functional decomposition in graphical format and display data as a tree view of requirements. RDT is able to generate documentation directly into MS Word, including requirements and test specifications, requirement allocation matrices, parent-child relationships and design documents.

Continuation of Table 3.3

RDD-100 version 4.1.2	Holagent Corporation	RDD-100 include a parser tool that can be defined and developed to help the user identify single or compound requirements. RDD-100 captures and traces requirements using its element relationship attribute repository, where each source document, and the text for each requirement, is stored as a separate element. Graphical hierarchies show how individual pieces of data relate to each other and trace back to their sources. The tool can extract requirements from ASCII form documents. RD-100 also provides the user the capability to interactively manipulate and input data through a variety of diagrams including behaviour diagrams, hierarchical views, functional flow diagrams, N2 charts, IDEF0, and data flow diagrams.
Requirements Traceability Management version 5.3	Integrated Chipware Inc.	Requirements Traceability Management supports multiple users working on the same requirements at the same time by implementing locking control on a requirement-by-requirement basis. The tool utilizes the native tool, which created the graphics object. A class definition tool is included that allows the user to model any type of hierarchical project data (requirement document, hierarchies, and system element structure). Once the hierarchy is defined generic relationships can also be established to allow cross-reference link information to be established between any active data item.
Reqtify	TNI-Valiosys	Reqtify is a requirements monitoring tool. It takes the formalized requirements produced by the requirements activity and monitors their implementation throughout the rest of the project's lifecycle. Reqtify manages requirements traceability and impact analysis across the project's entire lifecycle, enabling quality development in both hardware and software projects.
RMTrak	RBC Product Development	RMTrak uses a document centric approach that allows users to update the requirements independently from the tool. RMTrak provides multiple views (matrix, tree, allocation) to allow the user to access the requirements visually. The tool includes basic requirements traceability analysis tools, like the ability to view childless requirements and orphans within an included report.
RMI version 1.0.2	MathWorks	RMI allows users to coordinate, track, and implement changes in the design specifications throughout the development cycle. Users can access requirements stored in formal requirements management systems, such as DOORS, or in Microsoft Word, Excel, or HTML-formatted documents, and implement the requirements in MATLAB. RMI provides a graphical user interface called the Navigator that displays a tree-structured directory of all the blocks and subsystems in the current model.
QFD/CAPTURE	QFD capture	QFD/CAPTURE is a support tool for any decision making process - from basic to complex. The tool has a project focus, rather than a single matrix focus. This means that it is possible to you can set up a roadmap of the lists, matrices, and documents which will be developed for each particular project. The roadmap indicates links between the matrices. Data which changes in one matrix will cause related changes in the upstream and downstream matrices.
Qualica QFD	Qualica Software	Qualica QFD is a project data repository which combines the database with the flexibility of a spreadsheet program. It helps to master large amounts of information in anything from simple cause-and-effects matrix to complex, multistage QFD projects. Full integration of selected TQM methods like benchmarking, target costing, value analysis, and risk management tools makes Qualica QFD the tool for comprehensive requirements management.
SLATE version 5.0	SDRC	SLATE is integrated and delivered with FrameMaker and MS Word. SLATE does provide several requirements parsers to identify requirements from source text documents in addition SLATE supports OLE automation which allows external tools to "batch update" SLATE information. SLATE is built on top of a commercial multi-user OO database which allows geographically dispersed teams to work on the same database at the same time.
Software through Pictures version 8.2	Aonix	Software through Pictures consists of several tools that link all phases of the project lifecycle. Amongst the family of tools is the Validator/Req which models requirements in UML notation. Validator/Req checks the entered requirements information for logical and operational correctness. Additional features include the ability to generate test cases, testability reports, and requirements-to-test-case traceability matrices.

Continuation of Table 3.3.

Statemate MAGNUM	I-Logix	Statemate MAGNUM is a graphical modelling and provides a direct and formal link between user requirements and software implementation by allowing the user to create a complete, executable specification. Using Statemate MAGNUM requirements analysts can describe the system from three perspectives: functional, behavioural, and structural.
Tofs 01	TOol For Systems	Tofs 01 relies on MS Word's mechanism "find" in order to find key words and identifiers in the requirements text. The tool includes a mechanism to graphically capture system implementation using objects with requirements as attributes to these objects. Tofs 01 includes a requirements manager that allows users to list all requirements with attributes and trace objects within a system, which are concerned by the requirement. The requirements manager can also identify which requirements concern each design object and also to make sure that all requirements are taken care of by one design object or another.
Tracer	Revolutionary Business Concepts, Inc.	Tracer uses the MS Word environment to track changes made to requirements, and the facility to designate a requirement manually by visually marking the text within MS Word. Tracer provides multiple views (matrix, tree, allocation) to allow the user to access the requirements visually. The tool includes basic requirements traceability analysis tools like the ability to view childless requirements, and orphans within an included report.
VeroTrace	Verocel	VeroTrace associates various artefacts such as source code, design components, test procedures, functional test results, coverage results, and all review filenames that support all requirements. It also tracks the review state of requirements and their artefacts, and checks the progress of the requirement within the life cycle processes.
Vital Link	Compliance Automation, Inc.	Vital Link is a multi-user requirements management software tool that integrates a word processor and a relational database. Vital Link uses Adobe FrameMaker which allows users to create and re-use templates, create tables and graphics and type complex mathematical formulas. Vital Link utilizes a relational database which enables users to import existing documents from a variety of different word processors, and automatically parse the document which then can be ready for the user to edit, link entities, add attributes or generate reports. The tool provides traceability between requirements in different levels of specifications. Linkages also can be used to record traceability between all related information: requirements to mission needs statements, to operations concepts, to test, design, and verification documentation. In addition to the parent-child and peer-peer links provided, users can create their own linkage classes.
XTie-RT version 3.1.02	Teledyne Brown Engineering	XTie-RT, amongst other things like functional analysis, risk analysis, and testing it provides automatic parsing of requirements using the user-defined keywords that meet the keyword specification. XTie-RT is built on a proprietary database which supports a point and click query mechanism. It can be configured to support a functional or OO project based methodology. Traceability functions include support for normal parent/child links to manage requirements and support for peer links between items in the database and general documents to provide an audit trail showing compliance to quality standards or contractual conditions.

3.6.3 Taxonomy for RE-tools

Botella *et al.* (2002) emphasise the use of taxonomies for the problem domain description. Different requirements lists (Wieger, 1999; Lang and Duggan, 2001) and evaluation frameworks (Nikiforova and Sukovskis, 2002; Post and Kagan, 2000; Haywood and Dart, 1997; INCOSE, 2002; Franch and Carvallo, 2002; Hoffmann *et al.*, 2004) could help to prepare the RE-tool taxonomy. The selection according to the pre-defined criteria could separate between different tool features and group them into different category leafs.

The INCOSE (2002) requirements working group suggests a taxonomy (Figure 3.13) based on functional RE-tool characteristics. The taxonomy differentiates between

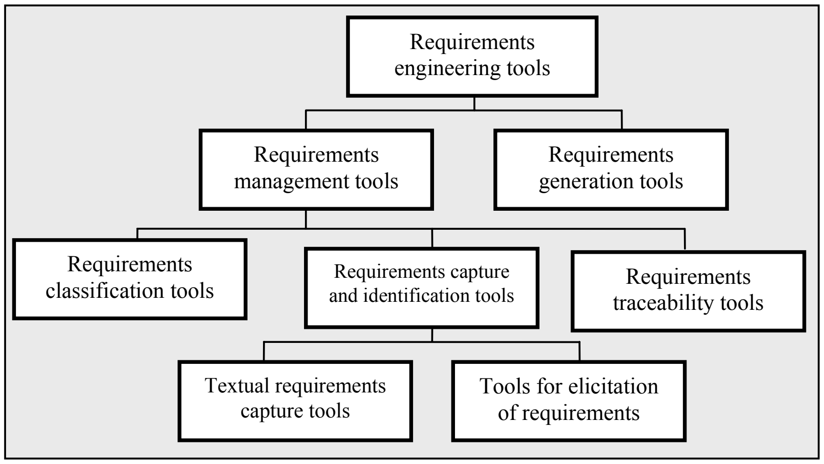

Requirements engineering tools
Requirements management tools
Requirements generation tools
Requirements classification tools
Requirements capture and identification tools
Requirements traceability tools
Textual requirements capture tools
Tools for elicitation of requirements

Figure 3.13 SE Tools Taxonomy - Requirements Engineering Tools adapted from (INCOSE, 2002)

requirements generation tools, requirements traceability tools, requirements classification tools, requirements capture tools, requirements identification tools and requirements elicitation tools.

Requirements generation tools (e.g., Criterium DecisionPlus and Ergo)[6] utilize system simulation results, performance allocations, mission scenarios, and design constraints to generate lower level requirements in an organised and traceable manner. Requirements traceability tools (e.g., CaliberRM, DOORS, and RequisitePro) enable software engineers to link requirements to their source, to changes in requirements, and to modelling elements that satisfy the requirements. Requirements classification tools (e.g., CART) help software engineers classify the requirements based on work to be done so that the requirement analysis activity can be scheduled and tracked. They help software engineers to classify requirements on how they will be used in modelling so that completeness of traceability could be monitored. Requirements capture tools assemble the information and assist software engineers in finding relationships among entities in the information and in moving among the entities. Requirements identification tools aid software engineers in highlighting requirements in the information before them from extraneous information. Requirements elicitation tools (e.g., QFD/CAPTURE and Qualica QFD) assist requirements engineers in drawing out requirements from system stakeholders. Tools include survey and interview techniques, methods and functionality.

[6] Look for the tool description in Table 3.3.

However, the INCOSE taxonomy could mislead during a tool selection process. First, the taxonomy classifies tools only to one category, when the RE-tools usually has a broad functionality and could be classified in several categories. Second, the classification of requirements management tools contradicts to the definition provided to such tools, because it includes RE activities.

The usage of special RE-tools affects the requirements specification quality. On another hand the RE-tools influence the time and resource scheduling and provide the facilities to control the RE process quality. But only introduction of the software tools does not contribute to the process improvement. The organisational maturity level should be investigated, and there is often a need to discuss the process quality independently of specific IS development method. However after the decision to acquire the specialised RE-tool for the RE process support there is a need to evaluate them quickly and in non-expensive way.

It is difficult to evaluate tools in terms of their impact on an organisation's work processes because, as the surveys indicate, the RE-tools are used rarely in practice. Similarly, it is difficult to examine tools in experiments, because it is difficult to control the variance of developers' capabilities and project context. Moreover, RE-tools provide the greatest benefit for large projects (where stakeholders frequently change their minds about requirements), while a controlled experiment normally requires prescribed tasks of a fairly limited size. It would thus be hard to create experimental tests that would provide a realistic assessment of the tools. Furthermore, the cost of thus evaluating several RE-tools empirically might be prohibitive for small- and medium-size organisations. There is also a need for a cheaper kind of evaluation that can be done analytically rather than empirically. For instance, RE-tools can be assessed from a theoretical point of view (e.g., using information provided by vendors), or tools can be tried out on some realistic examples, but without the rigour of a controlled experiment. A potential problem of such evaluation, however, is that they easily become ad hoc and subjective. Hence, hence in order to support the completeness and effectiveness of such evaluations, they should be grounded in a sound evaluation framework providing methodological support to the evaluators.

In the remaining chapters different methods and frameworks are considered in order to assess software tool and to acquire them to organisational settings. This work suggests the RE-tool evaluation approach which uses two evaluation frameworks and a set of guidelines to evaluate and to acquire the RE-tools to an organisational environment in order to improve the RE process.

3.7 Chapter Summary

First, in this chapter terms *requirements* and *requirements engineering* are defined. Next, RE models (Loucopoulos and Karakostas, 1995; Pohl, 1996; Kotonya and

Sommerville, 1998; Ferdinandi, 2002) are presented. The models separate different RE activities, characterise the RE process and the requirements specification. Further, empirical surveys of the RE in different geographical areas (e.g. Norway, Sweden, Finland, Lithuania, and United States) are overviewed. The empirical studies highlight a number of RE problems, like in-mature RE processes, weak reuse of requirements and a weak project stakeholder communication.

One of the identified problems include weak automated support for the RE process. It seems, industrial practice relies on using standard office and modelling tools to perform RE activities rather than targeted RE-tools. The main reasons include financial aspects, tool complexity and difficulty to evaluate these tools and acquire to organisational environment. The survey of RE-tools shows that the appropriate use of these tools could improve the quality of both RE process and requirements specification. The chapter formulates a problem on how to assess and acquire the RE-tool(s) in a quick and non-expensive way to the organisational settings.

In the next chapter an analytical analysis of the existing RE-tool frameworks is made. Next chapter also surveys evaluation approaches for the general commercial off-the-shelf (COTS) products.

CHAPTER 4

Software Tool Evaluation Frameworks and Approaches

The previous chapter concludes with a problem which describes difficulty to evaluate RE-tools before acquisition to the environmental settings. An organisation needs to have an instrument which would help to analyse and compare the available tools. Such an instrument could be defined as an evaluation framework which provides understanding of the artefact quality, its application and guides through the evaluation process.

First, in this chapter a semiotic quality framework which considers quality of conceptual models and modelling languages, is introduced. However the semiotic quality framework is too abstract for the tool evaluation, there for complex extensions for assessment purposes are needed. In order to highlight principles of the framework construction, several CASE tool evaluation approaches are analysed. Further, several lists of RE-tool requirements are considered. As standalone requirements frameworks they lack application guidelines. Therefore, the chapter concludes with a survey of the general COTS (commercial off-the-shelf) product evaluation approaches.

4.1 Semiotic Quality Framework

RE-tools affect both process and product quality. Process quality is addressed, because the tools support a large part of the software engineering process, in particular RE; and product quality is maintained as output of the RE is a requirements specification, which in itself should be of high quality for subsequent software engineering stages. There is a general belief that if the process is of high quality it produces a high quality product. In this sense, it is also possible to make a hypothesis, that if a high-quality tool evaluation process is maintained, its output - the selected RE-tool - will help to prepare high-quality requirements specifications.

The simplest quality evaluation approach is to define quality as a list of desirable properties (Batini, Ceri and Navathe, 1992; Simsion, 2001) or as comprehensive quality frameworks (Pohl, 1994; Lindland, Sindre and Sølvberg, 1994; Krogstie 1998; Becker, Rosemann, von Uthmann 2000; Krogstie and Jørgensen, 2003; Moody and Shanks, 2003). The basic limitation of these approaches is that they focus on the particular information or data models or they are too abstract for the RE-tool evaluation purposes. Despite of this the semiotic quality framework is surveyed because it provides the basis for construction of the RE-tool evaluation approach described in the second part of this work.

The *semiotic quality framework* (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003) distinguishes between quality goals and means to achieve these goals (Figure 4.1). It is closely related to linguistic and semiotic concepts. The semiotic quality framework is an extension of the (Lindland, Sindre and Sølvberg, 1994) quality framework, which includes discussion on syntax, semantics and pragmatics. Krogstie's framework adds two lower level, technical quality aspects (physical and empirical quality) and one higher level, social quality aspect (social quality) to the semantic, syntactic and pragmatic quality types. The semiotic quality framework is also based on a constructivistic view of world, and it recognises that models are created as a part of a dialog between the participants, whose knowledge of the domain changes as the process takes place.

Physical quality deals with two basic goals: externalisation, meaning that the explicit knowledge $\boldsymbol{K_M}$ of some person has been externalised in the model $\boldsymbol{M}$ by the use of a modelling language $\boldsymbol{L}$; and internalisability, meaning that the externalised model $\boldsymbol{M}$ is persistent and available, enabling the other persons involved to make sense of it. *Empirical quality* deals with error frequencies when a model $\boldsymbol{M}$ is read or written by different users, as well as coding and ergonomic of computer-human interaction for modelling tools. *Syntactic quality* is the correspondence between the model $\boldsymbol{M}$ and the language $\boldsymbol{L}$ extension of the language in which the model is written. *Semantic quality* is the correspondence between the model $\boldsymbol{M}$ and the domain $\boldsymbol{D}$. The framework has two semantic goals: validity and completeness. *Pragmatic quality* is the correspondence between the model $\boldsymbol{M}$ and social and technical audience's interpretation ($\boldsymbol{I}$ and $\boldsymbol{T}$) of it. *Perceived semantic quality* is the correspondence between the participants, interpretation $\boldsymbol{I}$ of a model and their current explicit knowledge $\boldsymbol{K_s}$. *Social quality* has the goal of agreement among participant interpretations $\boldsymbol{I}$. *Organisational quality* has to fulfil the goals $\boldsymbol{G}$ of modelling (organisational validity) and address them through the model $\boldsymbol{M}$ (organisational completeness).

The semiotic quality framework is adapted to evaluate the quality of a modelling language (Krogstie, 2001b). Language quality (Figure 4.2) is distinguished between two criteria types: 1) criteria for the underlying (conceptual) basis of the language (i.e. the constructs of the language); and 2) criteria for the external (visual)

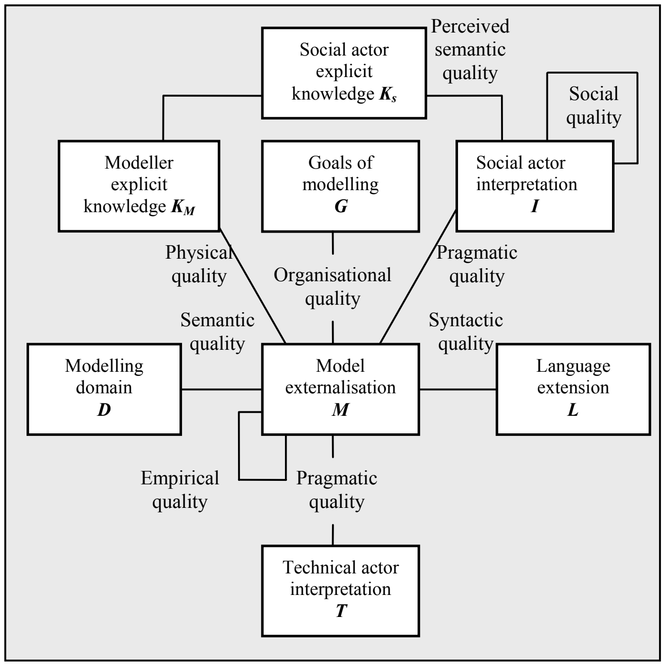


Figure 4.1 The Semiotic Quality Framework adapted from (Krogstie and Jørgensen, 2003)

representation of the language (i.e. the notations). Next, five quality areas are identified with aspect related both to the conceptual basis and the notations:

– *Domain appropriateness* relates the language to the domain and vice versa;
– *Participant language knowledge appropriateness* relates the participant knowledge to the language;
– *Knowledge externalisability appropriateness* relates the language to the participant knowledge;
– *Comprehensibility appropriateness* relates the language to the social audience interpretation;
– *Technical actor interpretation appropriateness* relates the language to the technical audience interpretation.

The semiotic quality framework could be adapted for consideration of the RE process and for analyses of the requirements specification quality (Krogstie, 2001a). In comparison to the NATURE three dimensional framework (Pohl, 1994; Pohl, 1996) analysed in Chapter 3, they both are similar in their structure. In the NATURE framework the representation dimension corresponds to the syntactic quality since they both deal with the relationship between the specification and the languages used. The specification dimension corresponds to the semantic quality since it deals with the completeness. The agreement dimension is related to the pragmatic quality as it describes the audience relationships with the specification. Although both the semiotic quality framework and the NATURE framework provide main fundamental principles for quality evaluation, the frameworks are too abstract and time consuming for the RE-tool assessment.

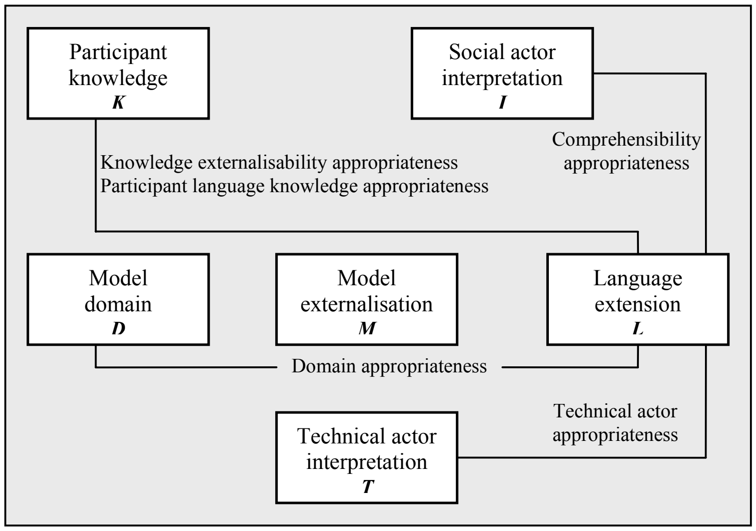

Participant knowledge K
Social actor interpretation I
Comprehensibility appropriateness
Knowledge externalisability appropriateness
Participant language knowledge appropriateness
Model domain D
Model externalisation M
Language extension L
Domain appropriateness
Technical actor appropriateness
Technical actor interpretation T

Figure 4.2 The Framework for Quality of Modelling Languages adapted from (Krogstie, 2001b)

4.2 RE-tool Evaluation Frameworks

RE-tools are subset of computer aided system engineering (CASE) tools, which are expected to provide task related support according to a predefined method. Nikiforova and Sukovskis in 2002 presented a framework for evaluation of CASE modelling tools. The framework is developed using an expert group, which selected a number of CASE tool features. The features include the tool usability (e.g., the usage of the tools is simple, flexible printing, browser window), functionality (e.g., model relationship analysis, document generation, and generation of comments), method and language support (e.g., object orientation support, and UML support), integration with other software (e.g., Web technology support, integration with MS Office, and integration with DBMS).

Post and Kagan (2000) present a market-based approach to consider the effectiveness of CASE tools. The market-based approach is the investigation of the product market with a purpose to determine what features users want to see in a CASE tool. By surveying tool users the desired CASE tool features can be identified. By grouping these features into categories, the usage of the tools can also be examined. By asking for evaluations of existing tools in terms of these features, existing tools can be compared, and specific product aspects can be determined that will support tool improvement and consequently make the products more effective. The evaluation criteria include five major categories: graphics (e.g., ease of changing class relationships, ability to search for diagrams that use a particular object, and ease of look ups for existing definition), teamwork (e.g., version control, multi-user locking, revision history, multi-user access and data dictionary), prototyping (e.g. ability to merge the modified code with existing models, code generator, programming language support, ability to generate code based on models, and inclusion of comments and description), general features (e.g., vendor longevity, vendor stability, vendor support, quality of on-line help facilities, and quality of documentation) and object oriented (e.g. support for class hierarchies and inheritance, support for encapsulation, support for polymorphism, and support for meta-classes) features.

Sedigh-Ali, Ghafoor and Paul (2001) define categories and metrics for risk and quality management when evaluating the COTS (commercial-of-the-shelf) products (Table 4.1). In addition to metrics definition, product impacts and software acquisition decisions should be evaluated. They include:

– the system's expected functionality and the customer's requirements;
– the makeup of the various organisations involved in the project and the level of maturity and capabilities of the participating teams;

Table 4.1 Metrics of COTS Quality adapted from (Sedigh-Ali, Ghafoor and Paul, 2001)

Category	Metric	Measures
Management	**Cost**	Total software development expenditure, including costs of component acquisition, integration, and quality improvement
	Time to market	Elapsed time between development start and component acquisition to software delivery
	Software engineering environment	Capability and maturity of the environment in which the software product is developed
	System resource	Use of target computer resources as a utilization percentage of total capacity
Requirements	**Requirements conformance**	Adherence of integrated product to defined requirements at various levels of software development and integration
	Requirements stability	Level of changes to established software requirements
Quality	**Adaptability**	Integrated system's ability to adapt to requirements changes
	Complexity of interfaces and integration	Component interface and middleware or and integration code complexity
	Integration test coverage	Fraction of the system that has under gone integration testing satisfactorily
	End-to-end user coverage	Fraction of the system that has undergone integration testing satisfactorily
	Fault profiles	Cumulative number of detected faults
	Reliability	Probability of failure-free system operation over a specified period of time
	Customer satisfaction	Degree to which the software meets customer expectations and requirements

- the developer's use of innovative processes and the methods they adopt as a part of the software engineering environment to manage cost and value, including details of developing process models such as the waterfall or spiral models;
- features of pre-existing tool that the system will use.

All three works (Post and Kagan, 2000; Sedigh-Ali, Ghafoor and Paul, 2001; Nikiforova and Sukovskis, 2002) provide similar list of CASE tool features; however, they only emphasise the feature discovery process, not the tool evaluation activities. It is not clear how to acquire and use these frameworks in the different organisational settings. The suggested criteria lists are quite large and their acquisition is needed when organisations have different modelling perspectives, development cycles, or different goals. The authors do not consider the relationships between the features, too. The three approaches suggest some fundamental issues for the tool assessment; however they do not target the RE-tool domain.

RE-tools could be considered as an application family. Requirements specification could produced selecting the requirements for evaluation from the application family model (Mannion *et* al., 1999; Kaindl and Mannion, 2005) which consists of a pool of numbered, atomic, natural language requirements, a domain dictionary and a set of discriminants. A discriminant is seen as a requirement which differentiates between different systems. It is assumed that model contains all requirements in all the existing systems in the application family (and those under construction) and is constructed as a lattice of parent-child relationships. Selection of requirement is performed using the *free selection* when users browse the requirements, and analyse the impacts of making choices at points of variability which helps to focus on what is required. However, lack of already constructed RE-tool application family could be seen as the limitation of this approach.

The common way to analyse and present RE-tool features and requirements is to define the categories and the specific requirements or activities for each category, where specific measures or metrics are used. Such organisation could be considered as *evaluation framework* which provides a skeleton structure for the RE-tool evaluation and comparison process.

There are three most common approaches to determine RE-tool requirements for evaluation purposes: expert-based method (Haywood and Dart, 1997; Nikiforova and Sukovskis, 2002) and market studies (Post and Kagan, 2000) and researchers' personal experience (Hoffmann *et al.*, 2004). The first approach involves the researcher and practitioners in system development, RE, and RE-tools. The sample of respondents is not large and usually direct interviewing is conducted for information gathering and eliciting the requirements and features for the frameworks. The second approach to define an evaluation framework consists of the performance of the market study, when a large number of questionnaire is sent out (by post, e-mail, Internet) in order to investigate the most common trends of the RE practice and RE-tool usage.

The third approach is used by Hoffmann *et al.* (2004). However, the approach is not always applicable and it depends on the situational settings for the experience project, environment, and organisational goals. All the framework definition approaches are based on the techniques used for the requirements elicitation. The major limitation of all of them is availability of the stakeholders, willingness and interest in method or framework definition.

In this section the existing RE-tool evaluation frameworks are surveyed. They are:
- Wieger (1999) requirements;
- Lang and Duggan (2001) requirements;
- INCOSE (2002) framework;
- Priority-based framework (Haywood and Dart, 1997);
- Role-based framework (Hoffmann *et al.*, 2004).

The following subsections present these frameworks in detail.

4.2.1 Wieger's Requirements

Wieger in 1999 provided a RE-tool survey, where 16 RE-tool requirements in a short text summary are used (Table 4.2). However, the requirements are fairly basic and more on a feature level then on the level appropriate for detailed evaluation.

Table 4.2 Wieger's (1999) Requirements

WReq.1.	Parses a source document to load requirements into database;
WReq.2.	Imports requirements from Word tables into database;
WReq.3.	Incorporates non-textual objects such as Excel worksheets and images into database;
WReq.4.	Synchronizes textual SRS with database contents;
WReq.5.	Defines different attributes for different types of requirements and set attribute values for individual requirements;
WReq.6.	Defines requirement baselines;
WReq.7.	Notifies affected project participants by e-mail about requirement changes;
WReq.8.	Defines traceability relationships or links between individual requirements and between requirements and other system elements;
WReq.9.	Tailors usability options;
WReq.10.	Includes learning aids, such as a tutorial and/or sample projects;
WReq.11.	Integrates with other tools, such as testing, design, and project management;
WReq.12.	Defines users and groups and their access privileges;
WReq.13.	Enables threaded discussions on requirements;
WReq.14.	Includes web interface for database query, discussion, and perhaps updating requirement attributes;
WReq.15.	Includes built in change proposal system;
WReq.16.	Includes hardcopy user manuals.

4.2.2 Lang and Duggan Requirements

Lang and Duggan (2001) requirements characterise a requirement management, communication and cooperative work system. They are:

LDReq.1. Maintain uniquely identifiable description of all requirements;

LDReq.2. Classify requirements into logical user-defined groupings;
LDReq.3. Specify requirements using textual, graphical, and model-based descriptions, with support for rich media description (such as images and animated simulations);
LDReq.4. Define traceable associations between requirements;
LDReq.5. Verify the assignments of user requirements to technical design specifications;
LDReq.6. Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests;
LDReq.7. Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed;
LDReq.8. Support standard system modelling techniques and notations;
LDReq.9. Maintain a comprehensive data dictionary of all project components and requirements in a shared repository;
LDReq.10. Generate predefined and ad hoc reports;
LDReq.11. Generate documents that comply with standard industrial templates, with support for presentation-quality output, WYSIWYG preview, and built-in document quality controls;
LDReq.12. Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.

While all stated RE-tool requirements are clearly useful, the list can be criticised for being somewhat unsystematic. LDReq.5 deals with requirements and design traceability. This partly covers and duplicates requirements LDReq.1, LDReq.3, LDReq.4, and LDReq.8. LDReq.1 speaks about the unique description, but does not consider what language should be used for description. It partly duplicates LDReq.3, which covers wide spectrum of informal requirements description. Thus, LDReq.8 covers modelling techniques, which can be both semiformal and formal. LDReq.4 deals with requirement-requirement traceability, but neglects traceability between source and requirements, and between requirements and design. Requirements LDReq.10 and LDReq.11 overlap since both deal with the providing of documents, reports and previews, which are being used to print different views to the process of requirements management.

4.2.3 INCOSE Framework

The INCOSE framework is suggested by the INCOSE working group (INCOSE, 2002) and classifies 52 RE-tool requirements into 14 categories (Table 4.3). The INCOSE provides a RE-tool survey, too, but the survey is based on vendors' information. The second weakness is that the framework terminology is not defined, so the survey obtained some strange results when tool vendors interpreted a feature name in a legitimate but unintended way, making their tools look fully compliant, but if one takes a closer look this is not necessarily the case.

Table 4.3 INCOSE (2002) Requirements

Category	Requirements
Capturing requirements/ identification	**INReq.1.** Input requirements enrichment/analysis (Input document change/comparison analysis) **INReq.2.** Automatic parsing of requirements **INReq.3.** Interactive/semiautomatic requirement identification **INReq.4.** Manual requirement identification **INReq.5.** Batch mode operation (batch-mode document/source-link update). **INReq.6.** Requirement classification.
Capturing system element structure	**INReq.7.** Graphically capture system structure. **INReq.8.** Textually capture system structure.
Requirements flow down	**INReq.9.** Requirements derivation (req. to req., req. to analysis/text) **INReq.10.** Allocation of requirements to system elements **INReq.11.** Bi-directional requirement linking to system elements. **INReq.12.** Capture of allocation rationale, accountability, test/ validation, criticality, issues.
Traceability analysis	**INReq.13.** Identify inconsistencies. **INReq.14.** Visibility into existing links from source to implementation. **INReq.15.** Verification of requirements. **INReq.16.** Requirements performance verification from system elements.
Configuration management	**INReq.17.** History of requirement changes, who, what, when, where, why, how. **INReq.18.** Baseline/Version control. **INReq.19.** Access control.
Documents and other output media	**INReq.20.** Standard specification output. **INReq.21.** Quality and consistency checking. **INReq.22.** Presentation output. **INReq.23.** Custom output features and markings. **INReq.24.** WYSIWYG previewing of finished output. **INReq.25.** Status reporting (technical performance measurement status accounting; requirement progress/status reporting; other ad hoc queries and searches) **INReq.26.** Support and display special character sets.
Groupware	**INReq.27.** Support of concurrent review, mark-up, and comment. **INReq.28.** Multi level assignment/access control.
Interface to other tools	**INReq.29.** Inter-tool communications (interfaces to other tools; external application program interface available; support open database system; import of existing data from various standard file formats) **INReq.30.** Intra-tool communication (exchange of information between same-tool different installations; consistency checking between same-tool datasets)
System environment	**INReq.31.** Single user/multiple concurrent users **INReq.32.** Multiple platforms/operation systems **INReq.33.** Commercial vs. unique database **INReq.34.** Resource requirements (memory requirements; CPU requirements; disk space requirements)
User interfaces	**INReq.35.** Doing one thing while you are looking at other. **INReq.36.** Simultaneous update of open views. **INReq.37.** Interactive graphical input/control of data. **INReq.38.** Which window standard do you follow? **INReq.39.** Executable via scripts or macros. **INReq.40.** Web browser interface. **INReq.41.** Edit-undo function support.
Support and maintenance	**INReq.42.** Warranty **INReq.43.** Network license policy **INReq.44.** Maintenance and upgrade policy **INReq.45.** On-line help **INReq.46.** Internet access/WWW home page location **INReq.47.** Phone support **INReq.48.** Support user's group
Training	**INReq.49.** Tool specific training classes **INReq.50.** Training available at customer's location **INReq.51.** Recommended training time **INReq.52.** Software installation with only basic training
Standards – which one do you comply with?	
What other requirements management features do you as a tool supplier think are important?	

4.2.4 Priority-based Framework

The priority-based framework (Haywood and Dart, 1997) classifies 53 RE-tool requirements according to three priority levels. The framework was created in consultations with practitioners who assigned high, medium and low priorities (Table 4.4). Functions have *high* priority if they are essential for creating, applying and maintaining a requirements model inside an automated environment suitable for the task. The requirements labelled as *medium*, are what would make an RE-tool useful for requirements modelling. Functions with *low* priority are desirable, but are regarded as non-essential.

However, organisations are not homogeneous environments, and priorities depend on various objective and subjective circumstances. The framework is development-oriented; however, a potential weakness is that it does not provide guidelines for how to analyse the RE-tool requirements if user priorities vary in different environments.

4.2.5 Role-based Framework

The role-based framework (Hoffmann et al., 2004) suggests 93 RE-tool requirements (Table 4.5), which are grouped according to project roles - developer, project administrator and RE-tool administrator. The separation reduces the amount of text stakeholders have to read to get an idea of the aspects of a tool most relevant to themselves, but it is not perfectly disjunctive, especially, one user may have several roles in a project. As with (Haywood and Dart, 1997), RE-tool requirements are also assigned high, medium and low priorities. The role-based framework does not consider guidelines for the framework application depending on the context. Also the authors do not provide empirical evidence of the framework's validity. The framework is only focused on functional RE-tool requirements, ignoring non-functional issues.

Table 4.4 Priority-based Evaluation Framework adapted from (Haywood and Dart, 1997)

	Category	Requirements
High Priority	Provide means to create the model, and assistance for user.	**HDReq.1.** Incremental input of requirements. **HDReq.2.** Easy look-up and cross-referencing. **HDReq.3.** Support for various formats (e.g., text, mathematical equations, graphs, pictures, drawings.) **HDReq.4.** Input from other sources (e.g., documents from word processors and spreadsheets). **HDReq.5.** Automatic labelling (including any imported requirements). **HDReq.6.** User defined annotation support (e.g., attributes such as creation date/time, ownership, classification, rationale). **HDReq.7.** Parsing of input to determine if syntactically correct. **HDReq.8.** On-line help. **HDReq.9.** Tailored assistance (to level working on and to experience of user). **HDReq.10.** Concurrency control (needed in multi-user environment).
	Provision for changes and their propagation	**HDReq.11.** Editing facilities for each representational notation. **HDReq.12.** Identification of changed requirements. **HDReq.13.** Dependency specification. **HDReq.14.** Propagation of changes through dependencies. **HDReq.15.** Configuration management. **HDReq.16.** Version management. **HDReq.17.** Impact analysis.
	Requirements can be traced from source onwards and from design backwards.	**HDReq.18.** Consistency checking (e.g., that all requirements have been met in the design and that all design artefacts are traceable back to an original requirement). **HDReq.19.** Automatic annotation (e.g., to provide links). **HDReq.20.** Storage of traces (retrieved easily and displayed in suitable format).
	Requirements to be decomposable or composable whilst maintaining any links between requirements. Users to be made aware of the level at which they are viewing.	**HDReq.21.** Graphical depiction of requirements (e.g., a select and decompose or gathering of symbols into more abstract item). **HDReq.22.** Annotation scheme (e.g., for non-graphical determination of "level" of decomposition).
Medium Priority	Present different views of model, e.g., for different stakeholders, possibly using different notations.	**HDReq.23.** Labelling and annotation of requirements (e. g., according to sub-problem or view). **HDReq.24.** Produce reports (e.g., for user validation). **HDReq.25.** Monitor consistency (e.g., between requirements and between views of requirements). **HDReq.26.** Highlight inconsistencies. **HDReq.27.** Check for completeness. **HDReq.28.** A "viewing" environment for each different format. **HDReq.29.** Automatic generation of one view from another (e.g., automatic translation of formal language into 'readable by non-users' language). **HDReq.30.** Animation of modules.
	Verification/Design Support	**HDReq.31.** Interface with CASE tools (e.g., tracing from design modules back to specification). **HDReq.32.** Completeness checking (e.g., automatic checking program meets specification).
	Test case/use case support	**HDReq.33.** Recording of example scenarios. **HDReq.34.** Automatic generation of test cases. **HDReq.35.** Check complete coverage of all use cases.
	Interfacing to other systems	**HDReq.36.** Requirements management tools. **HDReq.37.** CASE tools. **HDReq.38.** Document preparation tools. **HDReq.39.** Database/Spreadsheet.
	Glossary Support	**HDReq.40.** Means to define system terminology. **HDReq.41.** Data dictionary support.

Continuation of Table 4.4

Low Priority	Prioritise requirements	**HDReq.42.** Display priority in some suitable form.
	Costings	**HDReq.43.** Support for costing methods (e.g., function point counting). **HDReq.44.** Evaluation of changes supported.
	Reuse and access to stored requirements	**HDReq.45.** Catalogue requirements (to promote their reuse and enable other analysis tools to extract information). **HDReq.46.** Browsing capability. **HDReq.47.** Incorporation of reusable modules. **HDReq.48.** Extraction of information (e.g., from documentation.)
	Framework extended or altered to suit particular problem domain.	**HDReq.49.** Basic parameterisation (e.g., specify annotation scheme). **HDReq.50.** Support different modelling notations (e.g., state charts and concept graphs). **HDReq.51.** Meta-level configuration (e.g., for different problem domains).
	All types of requirements such as functional, behavioural etc. to be catered for.	**HDReq.52.** Recording of non-functional requirements. **HDReq.53.** Integrated tools (e.g., for different notations).

Table 4.5 Role-based Framework adapted from (Hoffmann *et al.*, 2004)

Role	Category	Features
Developers	Requirements management information (RMI) model	**HoReq.1.** Every object in the database must be uniquely identifiable over its lifetime. **HoReq.2.** The RMI model should be changeable during the project. **HoReq.3.** Inheritance and reuse should be available for all classes, types and attributes. **HoReq.4.** It could be possible to graphically define and configure the RMI. **HoReq.5.** The tool could support RMIs that are needed when using RE templates.
	Views	**HoReq.6.** The tool must allow views to be defined centrally as well as in a user-specific manner. **HoReq.7.** These views must be freely configurable, including filters on objects, relations, and attributes. **HoReq.8.** The objects must be changeable in the current view. **HoReq.9.** The user must be able to view the requirements in a document-oriented manner. **HoReq.10.** The user must be able to view the requirements in an information-model-oriented manner. **HoReq.11.** Graphical views of the requirements should be available. **HoReq.12.** The tools should allow view to be predefined for user roles. **HoReq.13.** All users must be able to customize the standard views without changing the template.
	Formatting, multimedia and external files	**HoReq.14.** The tool should support the basic formatting. **HoReq.15.** Non-text objects should be saved directly in the database or at least in a configuration management tool, that is tightly coupled with the tool. **HoReq.16.** External objects must be viewed either through a pre-viewer inside the tool or in the native application if called directly from the tool's user interface.
	Change management and comments	**HoReq.17.** The change requests should have public status information like pending, accepted, rejected. **HoReq.18.** There could be a comments or discussion function tightly linked to the requirements, but outside formal changes to requirements.
	Documentation of the history	**HoReq.19.** All changes to the requirements must be tracked and kept in the database. **HoReq.20.** The object in the tool must be versioned. **HoReq.21.** There must be a distinction between major and minor versions. **HoReq.22.** The version number should be incremented automatically when certain changes occur. **HoReq.23.** Changes must be tracked down to the smallest unit of data structures. **HoReq.24.** Changes and old versions must always be available. **HoReq.25.** A tool must allow a requirement to be changed back to any previous state anytime. **HoReq.26.** The tool should visualize the change history in an appropriate way. **HoReq.27.** The tool must generate freely configurable change reports. **HoReq.28.** The tool could analyse changes to provide information about the project status. **HoReq.29.** A comment should be saved with a change to enable it to be understood later on. **HoReq.30.** Changes could be categorized for analysis.

Continuation of Table 4.5

	Baselining	**HoReq.31.** The tool must support baselines
	Traceability	**HoReq.32.** The tool must be able to enforce the creation of change of certain links upon creation or change of a requirement. **HoReq.33.** Link must be directed and an object must be a source and target at the same time. **HoReq.34.** It must be possible to follow link directly in both directions. **HoReq.35.** It must be possible to give the link attributes. **HoReq.36.** It must be possible to create roles for governing what kinds of requirements must have links to what other kind of requirements, if this is not already enforced by the information model. **HoReq.37.** Links must connect any objects in the database, not only in the same subset. **HoReq.38.** Links could be n-ary. **HoReq.39.** The tool must feature a practical, user friendly and concise graphical representation and navigation of the traces.
	Analysis function	**HoReq.40.** The tool must provide information about status and progress of the project. **HoReq.41.** The tool should allow to analyse the inconsistencies in the link structure like finding gaps in the traces. **HoReq.42.** The tool can scan the description texts of the requirements for patterns like unsuitable/inexact language or wrongly used terminology.
	Tool integration	**HoReq.43.** Linking must not lead to redundant data. **HoReq.44.** The connection must be transparent in both tools. **HoReq.45.** Links to external objects must be managed in the same way as internal links. **HoReq.46.** The user should be able to navigate to these links. **HoReq.47.** Access rights to the external objects must be recognised. **HoReq.48.** The links should be able to target the smallest possible structure of the external object. **HoReq.49.** The interface used for tool integration should be active, i.e., synchronisation or change notification should occur automatically. **HoReq.50.** The tool could support tool integration platforms.
	Import	**HoReq.51.** The tool must recognise text marks, formatting, line ends, grammatical structure or keywords to interpret them as the beginning or end of requirement texts. **HoReq.52.** The tool should support a semiautomatic import of requirements from existing documents.
	Document generation	**HoReq.53.** The subset of data to be included in the document must be flexibly configurable, comparable to views. **HoReq.54.** The document generation must be able to include all information available in the tool. **HoReq.55.** The document generator must be able to create document in a certain standard formats. **HoReq.56.** Together with the requirement data the document generator must be able to include meta information. **HoReq.57.** Non-textual objects must be included in the generated documents. **HoReq.58.** The generated documents must be in a standard file format. **HoReq.59.** The tool must generate very large documents with many included external objects quickly. **HoReq.60.** It should be possible to run the document generation automatically as a background task.
		HoReq.61. The document generator must be extensible via a programming interface provided by a tool. **HoReq.62.** Access rights bust be enforced in the document generator.
	Collaborative working on the same task	**HoReq.63.** It must be possible for many user to work on the same data at the same time.
	Checking out for offline use	**HoReq.64.** It must be possible to check out data and a license to work on mobile offline computers without sacrificing consistency and access rights.
	Web access	**HoReq.65.** The tool should have a web interface or another browser based client.

		Continuation of Table 4.5
Project administrator	Central installation and administration of projects	**HoReq.66.** All project-wide information must be held and changed at one place. **HoReq.67.** A history of associated changes must be available.
	Users roles and rights	**HoReq.68.** The administrator must be able to manage user accounts and group and role assignments centrally. **HoReq.69.** Users must be defined centrally for all projects. **HoReq.70.** The tool must allow fine grade access and writing rights to be flexibly granted. **HoReq.71.** Security based on overlapping roles is preferred to security based on hierarchical security level. **HoReq.72.** The security concept must not be compromised by extensions like API programming or scripting. **HoReq.73.** Security among competing suppliers with access to the tool is an important issue. **HoReq.74.** Access rights must be grantable via roles a user is assigned to. **HoReq.75.** A user must be able to perform more then one role at a time. **HoReq.76.** Rights must be grantable down to object and attribute level.
	Size restriction	**HoReq.77.** The database fields must not have a fixed size restriction.
	Workflow management	**HoReq.78.** The tool could support system development via an administrable, organised, and structured process, called workflow. **HoReq.79.** The workflow must not simple restrict the users, but guide them through the process.
	Extensibility	**HoReq.80.** The tool must provide an open and well-documented object model and API which makes all data and functions accessible to extensions. **HoReq.81.** The object model and the API must be stable across versions of the tool. **HoReq.82.** The user interface of the tool must be customizable and extensible with a standard script language.
Tool administrator	Database	**HoReq.83.** The tool must use an appropriate database technology, which must be scalable and reliable. **HoReq.84.** To improve performance, the tool must use a database that uses a modelling paradigm similar to the one used in the tool. **HoReq.85.** The database must be available 24h a day and 365 days a year. **HoReq.86.** The database system must be transaction-safe and the tool must consistently use this feature. **HoReq.87.** The database must have a consistency analysis and data integrity check. **HoReq.88.** It should be possible to run a database backend in a distributed manner. **HoReq.89.** To improve data security and availability, the tool must use a database that is independent of the tool and can be administered independently. **HoReq.90.** It must be possible to backup and restore only a part of the database in the database. **HoReq.91.** It must be possible to export all project data and to import them again at a different time or places from/with different tool. **HoReq.92.** The data should be stored in a universal format.
	Encryption	**HoReq.93.** The information stored in the database of the tool must not be readable to system administrators or intruders.

4.2.6 Comparison of Frameworks

RE-tool frameworks describe tool functionality (Haywood and Dart, 1997; Lang and Duggan, 2001; INCOSE, 2002; Hoffmann *et al.*, 2004), and only some of them analyse non-functional RE-tool requirements, like costs, tool performance, usability, reliability, maintainability, and vendor characteristics (Haywood and Dart, 1997; INCOSE, 2002;). The most common functional requirement categories are:

– *Requirements model representation* using different representation paradigms and languages, like requirements LDReq.1, LDReq.3, and LDReq.8 (Lang and Duggan, 2001), capturing requirements (INReq.1 – INReq.3) and capturing system element structure (INReq.7 and INReq.8) (INCOSE, 2002), means to create the model (HDReq.1 – HDReq.7) test case and use case supporting (Haywood and Dart, 1997), requirements management information model, formatting, multimedia and external files (Hoffmann *et al.*, 2004).

- *Requirements traceability*, like requirements LDReq.4, LDReq.5 (Lang and Duggan, 2001), requirements flow-down, traceability analysis (INCOSE, 2002), requirements traces from source onwards and from design backwards (Haywood and Dart, 1997), traceability (Hoffmann *et al.*, 2004).
- *RE-tool association with other tools* or *import*/*export* to/from, like requirements LDReq.12 (Lang and Duggan, 2001), interface to other tools (INReq.29 and INReq.30) (INCOSE, 2002), verification and design support, interfacing to other systems (Haywood and Dart, 1997), tool integration, and import (HoReq.51 and HoReq.52) (Hoffmann *et al.*, 2004);
- *User and user group definition*, like user roles and rights (Hoffmann *et al.*, 2004);
- Maintenance of requirements *history*, *views* and *baseline*, like requirements LDReq.6 (Lang and Duggan, 2001), configuration management (INReq.17, INReq.18 and INReq.19), groupware (INReq.27 and INReq.28) (INCOSE, 2002), requirements (de)composition (HDReq.21 and HDReq.22), presentation of different views of models (Haywood and Dart, 1997), views, baselining, and documentation on the history (Hoffmann *et al.*, 2004);
- Requirements *attributes* and requirements *prioritisation*, like requirements LDReq.2 (Lang and Duggan, 2001) and prioritise requirements in (Haywood and Dart, 1997);
- *Collaborative work support*, and requirements model *change propagation*, like requirements LDReq.7 (Lang and Duggan, 2001), Provision for changes and their propagation (Haywood and Dart, 1997), change management and comments (HoReq.17 and HoReq.18), collaborative working on the same task, Web access (Hoffmann *et al.*, 2004);
- *Assistance for RE-tool users*, like requirements LDReq.8 (Lang and Duggan, 2001), HoReq.79 (Hoffmann *et al.*, 2004) and training (Haywood and Dart, 1997), assistance for user (requirements HDReq.8, HDReq.9, and HDReq.10), glossary support (M.5) (INCOSE, 2002).
- *Requirements repository functionality*, like requirements HoReq.3 (Hoffmann *et al.*, 2004), reuse and access to stored requirements (Haywood and Dart, 1997), size restriction (PA.3), database (TA.1) (Hoffmann *et al.*, 2004);
- *Report printing* according to different system and requirements views, like requirements LDReq.10, LDReq.11 (Lang and Duggan, 2001), documents and other output media (INReq.20-INReq.26) (INCOSE, 2002), document generation (Hoffmann *et al.*, 2004).

The frameworks (Haywood and Dart, 1997; Hoffmann *et al.*, 2004) prioritise RE-tool requirements according to their importance (high, medium, and low). But none of the frameworks specify traceability between RE-tool requirements – for example, between RE-tool functionality and usability. Lack of traceability could result in poor consistency during the RE-tool evaluation and/or development. None of the frameworks provide terminology explanation, and, therefore, they could mislead during the evaluation, especially, if RE-tools are considered by different evaluators.

Table 4.6 shows some comparison of the evaluation frameworks according to types they comprise, tool support for the framework and the framework use together with the evaluation approach. In the table the 'referenced evaluation' approach means that some proposals to combine the evaluation approach and framework are made; however this does not mean that the framework can't be combined with other evaluation approaches considered in the next section.

Table 4.6 Framework Comparison

Framework	**Types**	**Tool support**	**Referenced evaluation approach**
INCOSE	Functional	No	Quality-based (Carvallo et al., 2004a)
Priority-based (Haywood and Dart, 1997)	Functional and costs	No	None
Role-based (Hoffmann et al., 2004)	Functional and some non-functional (performance, reliability)	No	PORE (Maiden and Ncube, 1998)

4.3 COTS Evaluation Methods

RE-tools can be seen as a kind of commercial off-the-shelf (COTS) products. First, RE-tools are ready made products and potential users can select them from vendor product lists (Oberndorf, 1997). Second, RE-tools are sold in many copies and users are neither controlling the tools' specifications nor development processes (Vigder and Dean, 1997). Finally, tool users do not get access to the RE-tool source code (except in case of open source tools), and vendors are responsible for tool maintenance and improvement (Basili and Boehm, 2001).

There two basic actors during the software procurement: evaluation team and tool users or customers. *Evaluation team* plans, organises and executes the tool evaluation process, coordinates the evaluation actions and proposes a tool after evaluation result analysis. Tool *users* or customers have intention to acquire tool, evaluates the tool suitability, and after the tool selection, use the tool to support their work processes. The tool selection process typically consists of four phases (Finkelstein, Spanoudakis and Ryan, 1996; Feblowitz and Greenspan, 1998; Kunda, 2003): 1) user requirements specification; 2) understanding of the available tools; 3) assessment of the tool compatibility with the requirements; 4) and selection of the "best" tool (Figure 4.3).

Requirements specification is based on the working domain knowledge and existing manual systems. *Understanding of the available tools* involves the consideration of the functionality and relating it to the known processes from the user's experience. During the *assessment of the tool compatibility* the user has to assess the extent to which tools satisfy the requirements. *Selection of the "best" available package* depends on the compatibility with the requirements and the prioritisation of these requirements. The user may have to compromise on requirements not satisfied by any of the tools. Then the user reconsiders the requirements and iterates the selection or reorganises his working practices in order to fit the "best" tool.

This section makes a survey of the existing COTS evaluation approaches. They are:

- Procurement-Oriented Requirements Engineering (PORE), (Maiden and Ncube, 1998);
- Off-The-Shelf Option (OTSO), (Kontio, 1996);
- COTS Acquisition Process (CAP), (Ochs *et al.*, 2000);
- Scenario-based COTS Selection, (Feblowitz and Greenspan, 1998);
- Social Technical Approach to COTS Evaluation (STACE), (Kunda and Brooks, 1999; Kunda, 2003);
- ISO/IEC 9126 Quality-based method (Franch and Carvallo, 2002).

The following subsections present these approaches in detail.

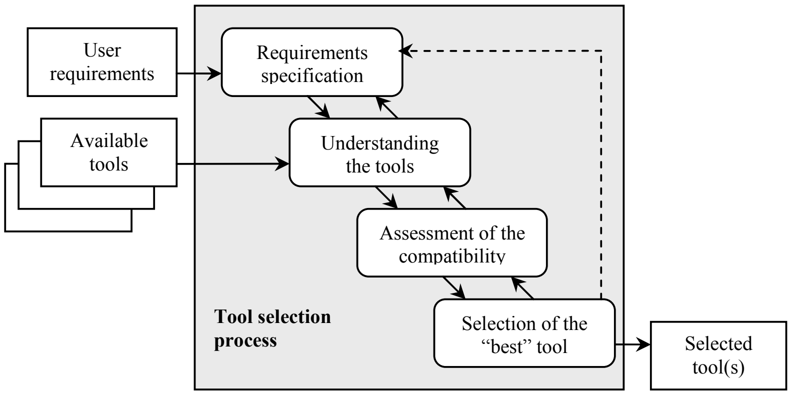


Figure 4.3 COTS Selection Process

4.3.1 Procurement-Oriented Requirements Engineering

The Procurement-Oriented Requirements Engineering (PORE) method integrates techniques for requirements acquisition and COTS selection with process guidance for choosing and using each technique (Maiden and Ncube, 1998). It is template-based and advocates a parallel and iterative requirements acquisition and tool-candidate selection/ rejection (Figure 4.4). The PORE supports engineering team to acquire, describe and analyse customer requirements at the same time as acquiring, modelling and analysing candidate COTS. The main steps for PORE are (Figure 4.5):

– *acquire information* about user requirements, COTS, suppliers and procurement contracts from customers;
– *analyse information* for completeness and correctness;
– use *decision-making* techniques to analyse and determine requirement compliance;
– *reject* (*select*) one or more *candidate* COTS that are non-compliant with customer requirements defined by the evaluation goals.

The application of the PORE method is demonstrated in BANKSEC project (Maiden, Kim, and Ncube, 2002) where a situational meta-model and selection of software component for the banking domain are considered. The SCARLET (Selecting Components Against Requirements) extensions (Maiden and Kim, 2002) to the PORE method include a lightweight process that designs meta-requirements in respect to non-functional tool requirements. PORE is supported by a prototype tool SCARLET-Advisor (Maiden, *et al.*, 2003) which enables process guidance to the evaluation team to select COTS product.

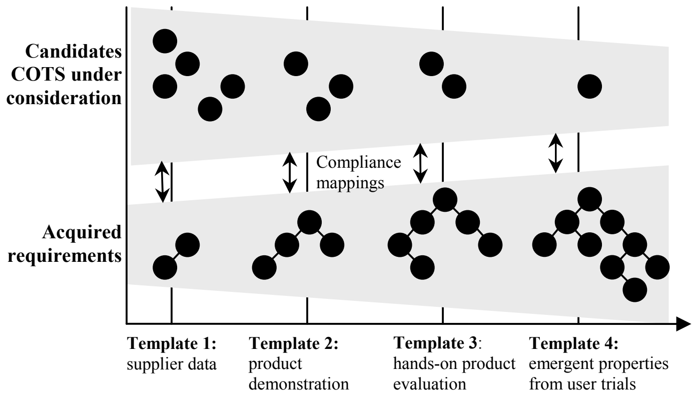

Candidates COTS under consideration
Compliance mappings
Acquired requirements
Template 1: supplier data
Template 2: product demonstration
Template 3: hands-on product evaluation
Template 4: emergent properties from user trials

Figure 4.4 PORE Process for COTS Selection adapted from (Maiden and Ncube 1998)

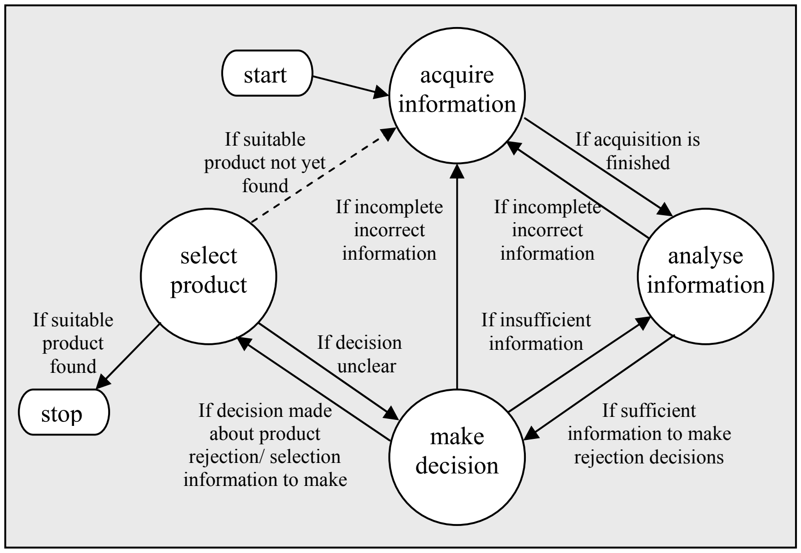

start
acquire information
If suitable product not yet found
If acquisition is finished
If incomplete incorrect information
If incomplete incorrect information
select product
analyse information
If suitable product found
If decision unclear
If insufficient information
stop
If decision made about product rejection/ selection information to make
make decision
If sufficient information to make rejection decisions

Figure 4.5 PORE Processes
adapted from (Achour and Ncube, 2000)

4.3.2 Off-The-Shelf Option

The Off-The-Shelf Option (OTSO) method (Kontio, 1996) describes a way to incorporate the COTS tool into the system, already used in an organisation. The method assumes that tool requirements already exist, and it is based on the requirements specification for defining the evaluation criteria. The main principles of the OTSO method are 1) tasks in the selection process including entry and exit criteria; 2) incremental, hierarchical and detailed definition of evaluation criteria; 3) a model for comparing the costs and value associated with each alternative, making them comparable with each other; 4) use of appropriate decision making methods to analyse and summarise evaluation results.

The OTSO selection process is divided into six phases. In the *search* phase the number of possible alternatives grows rapidly. The most potential candidates are sorted out (*screening* phase) to pick the ones that can be evaluated in more detail. Detailed *evaluation* of alternatives determines how well each of them meets the evaluation criteria. The *analysis* phase interprets the evaluation criteria using multiple criteria decision techniques. Based on decisions made, typically one of the alternatives is selected and *deployed*. Finally, in order to improve the selection process and to provide feedback on potential further reuse of the component, it is necessary to *assess* the success of the reuse component used in a project. OTSO applies a model (e.g. Analytic Hierarchy Process (Saaty, 1980)) for comparing the costs and value associated with each candidate tool makes tools easily comparable. However, OTSO ignores the definition of the requirements specification for the tool selection, thus, giving little support in evaluating whether a tool fits the specific needs of the organisation.

4.3.3 COTS Acquisition Process

The COTS Acquisition Process (CAP) method (Ochs *et al.*, 2000) consists of three parts (Figure 4.6) – initialization, execution and reuse. *Initialization* comprises all the activities related to the definition of the decision basis and the measurement plan. *Execution* which comprises all activities dealing with the identification of possible COTS software alternatives, performing measurements and decision making on the set of existing COTS software alternatives. *Reuse* comprises all activities referring to packing information about COTS software reuse in future CAP enactment.

System design process provides input to the CAP in the form of tool requirements for the perspective system component that shall be implemented through COTS usage. In case that there is an adequate COTS available, *supply process* receive the name and supplier of the selected COTS software for negotiation purposes and eventually buy the COTS product.

In CAP the main criteria categories include functional, non-functional, domain and architecture and strategic tool requirements. They are based on ISO/IEC 9126 standard (1991), and other criteria lists extracted from expert interviews, literature reviews, and applied research activities.

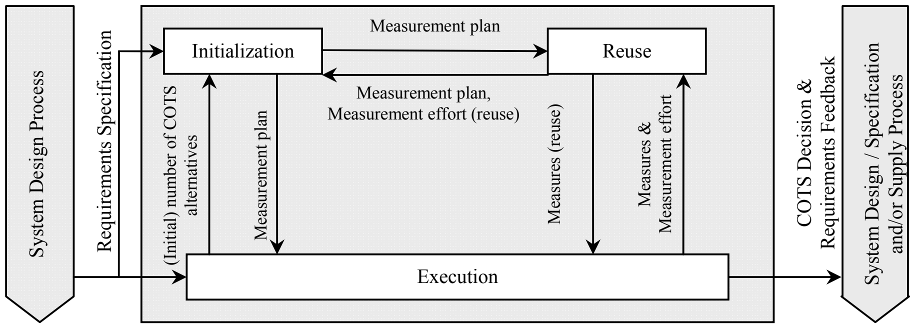

System Design Process
Requirements Specification
Initialization
Measurement plan
Reuse
Measurement plan,
Measurement effort (reuse)
(Initial) number of COTS alternatives
Measurement plan
Measures (reuse)
Measures & Measurement effort
Execution
COTS Decision & Requirements Feedback
System Design / Specification and/or Supply Process

Figure 4.6 CAP Information Flow adapted from (Ochs *et al.*, 2000)

4.3.4 Scenario-based Selection

The scenario-based COTS selection (Feblowitz and Greenspan, 1998) proposes a comparison between *baseline scenarios* which describe how organisation operates, and *tool scenarios*, which are baseline scenario projections into a future, where a tool is applied. Figure 4.7 depicts an *analyst* who rewrites the baseline scenarios adapting it to represent scenarios with tool 'A'. Once the baseline scenarios have been rewritten, the resultant tool scenarios are then compared against the baseline scenario to determine: 1) what differences are between the baseline and tool scenarios; 2) what the impacts of the changes are; 3) what changes to propagate to other parts of the organisational processes; and 4) what the impacts of the propagated changes are.

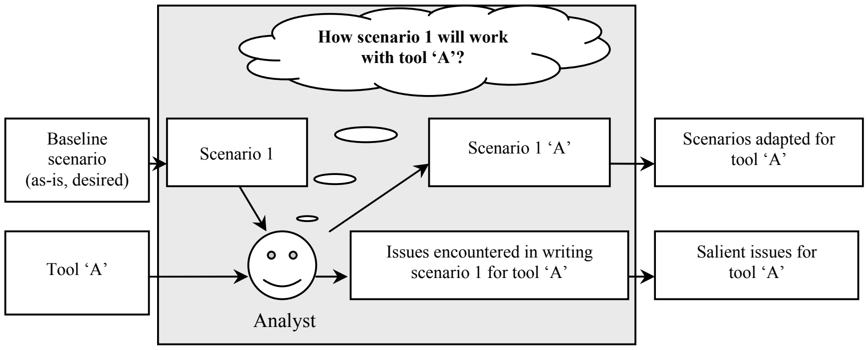

How scenario 1 will work with tool 'A'?
Baseline scenario (as-is, desired)
Scenario 1
Scenario 1 'A'
Scenarios adapted for tool 'A'
Tool 'A'
Issues encountered in writing scenario 1 for tool 'A'
Salient issues for tool 'A'
Analyst

Figure 4.7 How Scenario Will Work with Candidate 'A' adapted from (Feblowitz and Greenspan, 1998)

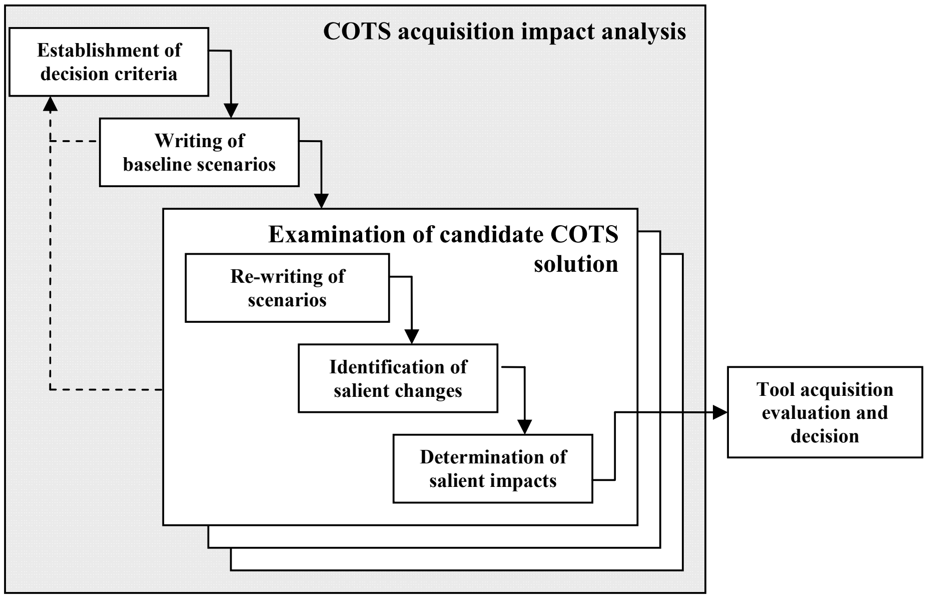

COTS acquisition impact analysis
Establishment of decision criteria
Writing of baseline scenarios
Examination of candidate COTS solution
Re-writing of scenarios
Identification of salient changes
Determination of salient impacts
Tool acquisition evaluation and decision

Figure 4.8 COTS Acquisition Process in Scenario-based Selection adapted from (Feblowitz and Greenspan, 1998)

The scenario-based selection does not analyse the tool coverage of the required functionality, the non-functional requirements, tool interoperability with the systems used in the organisation, fiscal health of the tool vendor, or its ability to provide support of the tool. These criteria should be determined during the tool selection process (Figure 4.8).

4.3.5 Social Technical Approach to COTS Evaluation

The Social Technical Approach to COTS Evaluation (STACE) framework (Kunda and Brooks, 1999; Kunda 2003) comprises four interrelated processes (Figure 4.9): 1) requirements elicitation; 2) social-technical criteria definition; 3) alternatives identification; and 4) evaluation. In the *elicitation*, tool requirements are discovered through consultations with users, from domain knowledge and market studies. In the *social-technical criteria* definition the elicited tool requirements are decomposed into a hierarchical set, and each branch in this hierarchy ends in a measure or metric. *Alternative identification* includes searching for tool candidates. *Evaluation* involves ranking of identified tools against the social-technical criteria by examining capabilities, reading documentation, and experimentation.

Figure 4.10 illustrates the main COTS evaluation process in STACE. Step 1 selects the underlying technology or other keystone issues. The selection process involves 1) defining the evaluation criteria, 2) search and screen for available alternatives, 3) revising the criteria and requirements based on available technologies, 4) assessing and selecting the "best" technology among alternatives. Step 2 defines social-technical evaluation criteria for COTS products based on the selected technology. The criteria should include functionality issues, technology and interface issues, quality characteristics and non-technical issues. Step 3 searches and screens for available COTS products. Search the marketplace to identify candidate COTS components through market surveys and other techniques like Internet search. The search criteria at this stage should be limited to functionality issues. Step 4 revises tool requirements and social-technical criteria based on available COTS products. The selection of the "best" among the packages available depends on the assessment of their compatibility with the requirements specification and the prioritisation of these requirements. Step 5 evaluates the candidate components and selects the "best" COTS product. The evaluation should include ranking of the candidate products according to the evaluator's preferences.

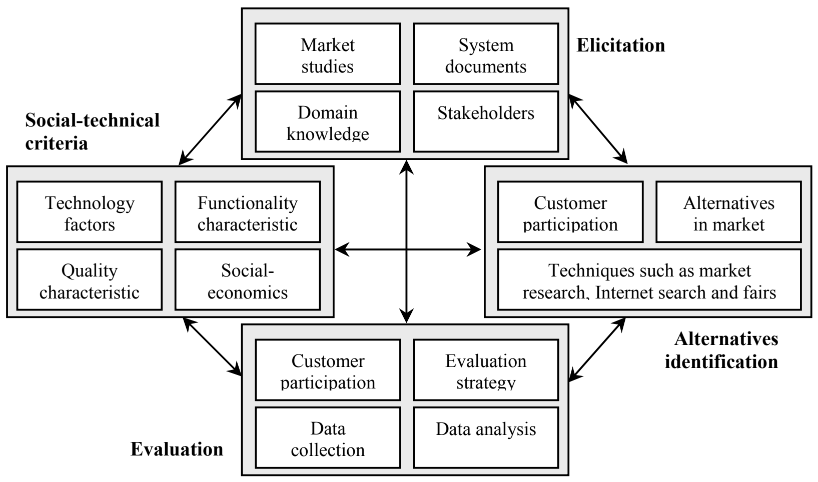


Figure 4.9 STACE Framework adapted from (Kunda, 2003)

4.3.6 Quality-based Selection

The ISO/IEC 9126 quality-based approach (Franch and Carvallo, 2002) constructs a quality model for a tool domain. Following the ISO/IEC 9126 (1991) standard, relevant attributes for a specific tool domain are structured to hierarchy of subcharacteristics and attributes which later on are used as evaluation criteria and metrics. The basic guidelines include:

– *Definition of the domain*: the domain of interest has to be carefully examined and described.
– *Determination of quality subcharacteristics*: division of the ISO/IEC 9126 (1991) standard characteristics into subcharacteristics.
– *Definition of a hierarchy for subcharacteristics*: subcharacteristics are further decomposed with respect to some factors, yielding to a hierarchy of them.
– *Decomposition of subcharacteristics into attributes*: decomposition of the abstract subcharacteristics into more concrete ones – quality attributes.
– *Decomposition of derived attributes into basic ones*: selection of the attributes which are possible to measure in the conceptual model;
– *Determination metrics for basic attributes*: metrics for all the basic attributes should be selected. Metrics could include Boolean, Numerical, Label, and Structures (sets, functions) ones;
– *Definition of relationships between quality subcharacteristics and attribute*: the relationships could include three types: collaboration, damage, and dependency.

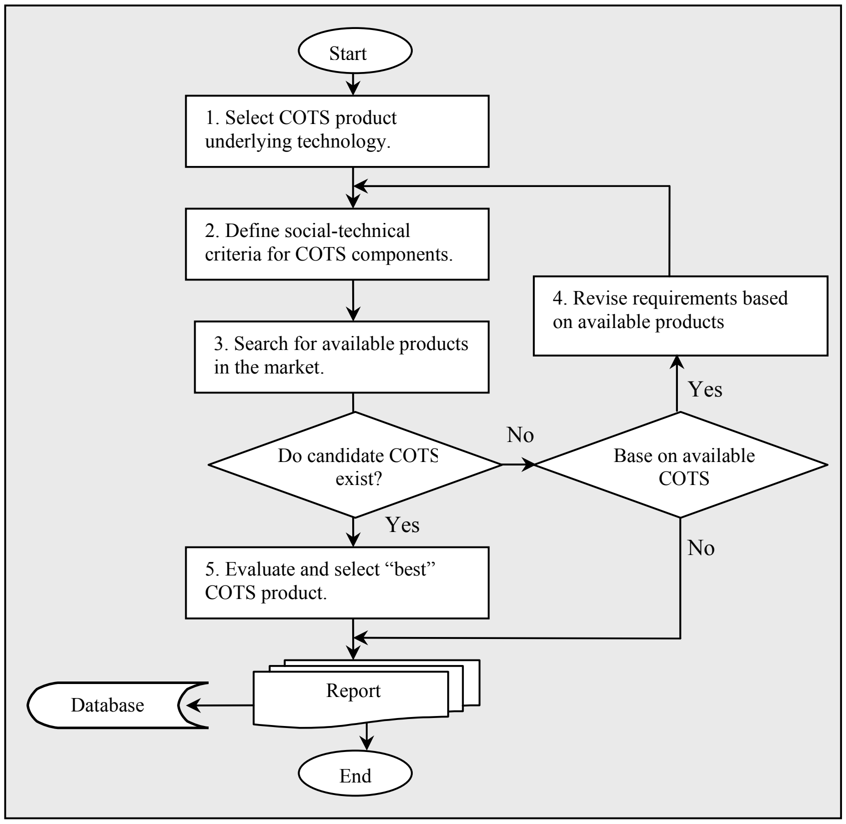

Start
1. Select COTS product underlying technology.
2. Define social-technical criteria for COTS components.
3. Search for available products in the market.
4. Revise requirements based on available products
Do candidate COTS exist?
No
Yes
Base on available COTS
Yes
No
5. Evaluate and select "best" COTS product.
Report
Database
End

Figure 4.10 STACE Evaluation Process adapted from (Kunda and Brooks, 1999)

Next the general tool evaluation process (Figure 4.3) is summarised into the COSTUME (Composite Software system quality model development) model (Carvallo et al. 2004a) which separates four activities: 1) analysing of the environment of the software system; 2) decomposing of the software system actors; 3) building individual quality models for the actors; and 4) combining the individual quality models.

The application of the approach is time-consuming because the quality model should be reconsidered for each particular domain. Although it is possible to reuse the created quality model in different environments, the model itself gets large. Reusable quality models are refined for particular environmental needs, as it is demonstrated in several case studies for assessment of the mail servers in (Franch and Carvallo, 2002) and RE-tools in (Carvallo et al. 2004a). The quality-based approach is also supported by a software tool DesCOTS (description, evaluation and selection of COTS component), which is described in (Grau et. al, 2004; Carvallo et. al, 2004b).

4.3.7 Comparison of Approaches

All approaches correspond to the general COTS selection process in Figure 4.3. However each of them highlights different techniques. PORE (Maiden and Ncube 1998) describes template-based tool procurement. The scenario-based selection (Feblowitz and Greenspan, 1998) maps baseline scenario and tool scenario. The quality-based (Franch and Carvallo, 2002) approach constructs a quality model for tool domain. OTSO (Kontio, 1996), CAP (Ochs *et al.*, 2000), scenario-based selection Feblowitz and Greenspan, 1998) and STACE (Kunda, 2003) assume that evaluation criteria are defined in advance.

The approaches describe only general categories like functional, non-functional tool requirements, tool architecture, vendor and business requirements, which are based on ISO/IEC 9126 standard (1991). The category-criteria definition activities resemble to construction of an evaluation framework. OTSO (Kontio, 1996), PORE (Maiden and Ncube, 1998), and STACE (Kunda, 2003) use Analytic Hierarchy Process (AHP) (Saaty, 1980) for tool requirements prioritisation. However, the AHP method is only applicable when there are few comparisons and when all criteria are independent.

The evaluation could be divided into two types: *one-round* and *multi-round* evaluation. In one-round evaluation, the evaluation is executed one time taking in account all the candidate tools. After each evaluation step, the tool list is reduced and the tool requirements are expanded (e.g. PORE (Maiden and Ncube, 1998)). But most of the approaches (OTSO (Kontio, 1996), CAP (Ochs *et al.*, 2000), scenario-based (Feblowitz and Greenspan, 1998), and STACE (Kunda, 2003)) apply multi-round evaluation, when the full evaluation cycle is repeated with each individual tool. The decision about tool suitability is made after testing all tools.

In summary the general COTS evaluation approaches are criticised for labour-intensive activities to define evaluation criteria. Further, none of the approaches are specifically targeted towards RE-tools. Therefore, the evaluation criteria, measure, and process definition are time consuming and domain knowledge demanding.

4.4 Chapter Summary

In this chapter a survey of the evaluation frameworks and approaches for the tool selection is presented. First some general frameworks, like the semiotic quality framework (Krogstie 1998; Krogstie 2001a; Krogstie and Jørgensen, 2003) and the NATURE framework (Pohl, 1994; Pohl, 1996) are considered. Both frameworks are, however, too abstract for the tool evaluation purposes. Therefore, general CASE tool and targeted RE-tool evaluation frameworks are analysed. However, these frameworks lack application and usage guidelines.

Thus, the existing COTS (commercial-off-the-shelf) product selection approaches (Maiden and Ncube, 1998; Kontio, 1996; Ochs *et al*., 2000; Feblowitz and Greenspan, 1998; Kunda, 2003; Franch and Carvallo, 2002) are overviewed. All the approaches are instances of the general COST product selection process (Finkelstein, Spanoudakis and Ryan, 1996), but focus on different techniques of the tool evaluation and acquisition to the environment. However, the criteria definition is time consuming and domain knowledge demanding, since none of them are directly targeted towards the assessment of the RE-tools. Thus thesis a need in defining a targeted RE-tool evaluation methodology which would guide through the assessment process.

In the following part the RE-tool evaluation approach (R-TEA) is presented. The R-TEA method uses two frameworks for evaluation of functional and non-functional RE-tool requirements which leads to the selection of the RE-tools according to the organisational needs.

PART II

THEORETICAL APPROACH

The second part presents the theoretical approach to the RE-tool assessment. It consists of two chapters.

In Chapter 5 *Frameworks for Functional and Non-functional RE-tool Requirements* two evaluation frameworks are introduced. The first framework characterises functional RE-tool features. The second framework describes non-functional RE-tool features.

The evaluation framework for functional RE-tool requirements is based on the NATURE framework (Pohl, 1994; Pohl, 1996) consisting of three dimensions – requirements representation, requirements agreement, and requirements specification. The requirements representation dimension deals with the degree of formality, where requirements are described using informal, semiformal and formal languages. The requirements agreement dimension deals with the degree of agreement among project participants. It is important to ensure communication among the project participants. The requirements rationale leads to agreement about the requirements model. The requirements specification dimension deals with the degree of requirements understandability. The projects knowledge and existing standards should be applied in order to ensure the quality level of the requirements specification.

Here, the non-functional RE-tool features are classified as process, product and external as suggested by Kotonya and Sommerville (1998). The process requirements are constraints placed upon the development process of the system and the organisational working practice. The product requirements specify the desired qualitative characteristics that an RE-tool must possess. The external requirements are divided to organisational requirements and requirements for business parties. The organisational requirements characterise the RE-tool according to the costs and business issues. The requirements for business parties deal with the vendor performance and reliability, which include the infrastructure and stability, the vendor reputation, customer base and track records.

In Chapter 6 *RE-tool Evaluation Approach* a specification exemplar is constructed according to two proposed frameworks. As the specification exemplar is intended to be sketchy and imprecise, it is the starting point for the construction of the requirements specification for the RE-tool evaluation under the environmental settings. Six guidelines which describe an RE-tool evaluation approach (R-TEA) are presented. Finally, the chapter presents a small example of the R-TEA application is presented and the proposed method with general tool selection approaches are compared in the chapter.

CHAPTER 5

Frameworks for Functional and Non-functional RE-tools Requirements

In Chapter 4 evaluation approaches of the commercial off-the-shelf products and evaluation frameworks for the RE-tools were considered. However, the approaches are not specifically targeted towards RE-tools. And RE-tool frameworks lack application guidelines. In this chapter two evaluation frameworks are proposed. The first framework analyses the functional RE-tool features. The functionality evaluation is based on the NATURE framework (Pohl, 1994; Pohl, 1996) which separates three dimensions: requirements representation, requirements agreement and requirements specification. The second evaluation framework classifies non-functional RE-tool features to process, product, organisational requirements and requirements for business parties. Relationships define dependencies between functional and non-functional RE-tool features.

5.1 Framework Definition

Two frameworks for the RE-tool evaluation are engineered according to the literature study, analyses of related works and empirical investigations of the activities taken part in the industry as discussed in Chapter 3. The validity of the frameworks is considered in several case studies described in Chapter 7.

The purpose of the frameworks is to provide a skeleton structure for the RE-tool evaluation and comparison. An intension of the frameworks is not to be detail, but to provide a list of RE-tool features which should be considered during the RE-tool assessment activities. *Features* represented in the frameworks describes the RE-tool characteristics which tool user could think it might be nice to have in an RE-tool. The features are initial material for starting the RE-tool assessment. The features are not so detail. They do not characterise how the RE-tool should implement feature in itself. The features are the basis for the RE-tool requirements, which should be specified in a requirements specification for the RE-tool selection.

Despite of the discussion above the terms *framework for evaluation of functional RE-tool requirements* and *framework for evaluation of non-functional RE-tool requirements* are used in this work in order to highlight the purposes and goals the proposed frameworks.

When applying the evaluation frameworks, the user needs to consider the features and to decide whether they are important for the RE-tool or not. Application of the frameworks involves the requirements elicitation activities. During the elicitation the user analyses the features suggested in the RE-tool frameworks, prioritise them according to their importance and ends with the requirements specification for the RE-tool selection. The importance is recognised according to the user's practice, experience and the organisational work practice. The requirements specification contains the *RE-tool requirements* which could be either functional RE-tool requirements or non-functional RE-tool requirements. The *functional RE-tool requirements* describe what the RE-tool should do, what behaviour it should have. The *non-functional RE-tool requirements* describe RE-tool characteristics, constraints and properties. The activities of the frameworks' application are considered in Chapter 6.

The requirements model formality much depends on its application and feasibility which defined the relationship between benefit and drawback to achieve the formality. The formality also much depends on of its definition and the cultural environment this definition is applied in. For instance a representation as *formal* can be processed purely based on its form (Snaprud and Kaindl, 1994). Such a representation is not always sufficient since model may still be in complete for the operations. Therefore the separation between formal and operational representations has to be done. At the other extreme there is an *informal* representation (e.g. natural language). When informal and formal representations are combined, a *semiformal* representation is received. A semiformal representation can be more or less formal, depending on the balance between formality and informality.

However in this work the definition used in the NATURE framework (Pohl, 1994), and supported by the semiotic quality framework (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003) are applied. According to them the formality or informality of the model is described according to the language semantics and syntax definition. The *formal* language is a language with a precisely defined vocabulary, syntax and semantics. The *semiformal* language is a language with a precisely defined vocabulary and syntax, but without a precisely defined semantics. Finally, the *informal* language is neither formal nor semiformal. The natural language and the *professional* language (used by a set of people working in a certain kind of area) belong to this category.

Shipman and McCall (1994) identify a number of problems when incrementally describing the model from informal to formal representation. First, this process takes time and effort. Second, formal models seem to be highly error prone and difficult to correct when done wrong. The difficulty of formalisation also include the premature impose of the structure. When new situations are encountered, the understanding of the specific problem, changes. Formalisations based on previous understandings become outdated and counterproductive. This situation is observed in a case study by Nguyen and Swatman (2003) and reflects the process heuristics, as discussed in section 3.2.

However, application of formal languages and preparation of requirements model which could speed up the creation of the system in the later phases, and it is especially useful when developing the system. But the purpose of this work is not RE-tool development, but RE-tool evaluation, before the selection and acquisition to environment. The evaluation frameworks presented in this chapter have to be flexible enough in order to be comprehensible for the users, who might have different knowledge, practice and education. Therefore both evaluation frameworks are defined informally using natural language.

5.2 Framework for Functional RE-tool Requirements

The *framework for evaluation of functional RE-tool requirements* consists of three dimensions, inspired by Pohl's work in the NATURE project (Pohl, 1994; Pohl, 1996): requirements representation, requirements agreement, and requirements specification. The framework focuses on evaluation of RE-tool functional features, which express how the tool behaves. Framework features are adapted from Lang and Duggan (2002) requirements list, which after refinement was fit in the three dimensional space (Figure 5.1) of the RE process. Next, the features are expanded with lists of activities, to be performed in the RE process. Different activities were discovered through investigation of the RE-tools from a theoretical point of view (Matulevičius and Strašunskas, 2002) and through an analysis of the quality types and means of semiotic quality framework (Lindland, Sindre and Sølvberg, 1994; Krogstie 1998). Thus, the selection of the evaluation activities and the tool features depends on an organisational profile – what kind of software development practice and techniques are used in an organisation, and what kind of problem domains an organisation deals with. The list of activities could be extended and improved according to evaluator experience and organisational needs. The framework supports the evaluation by distinguishing between the goals to be reached and the means used to reach them, similarly, as done in the semiotic quality (Krogstie 1998; Krogstie 2001a; Krogstie and Jørgensen, 2003).

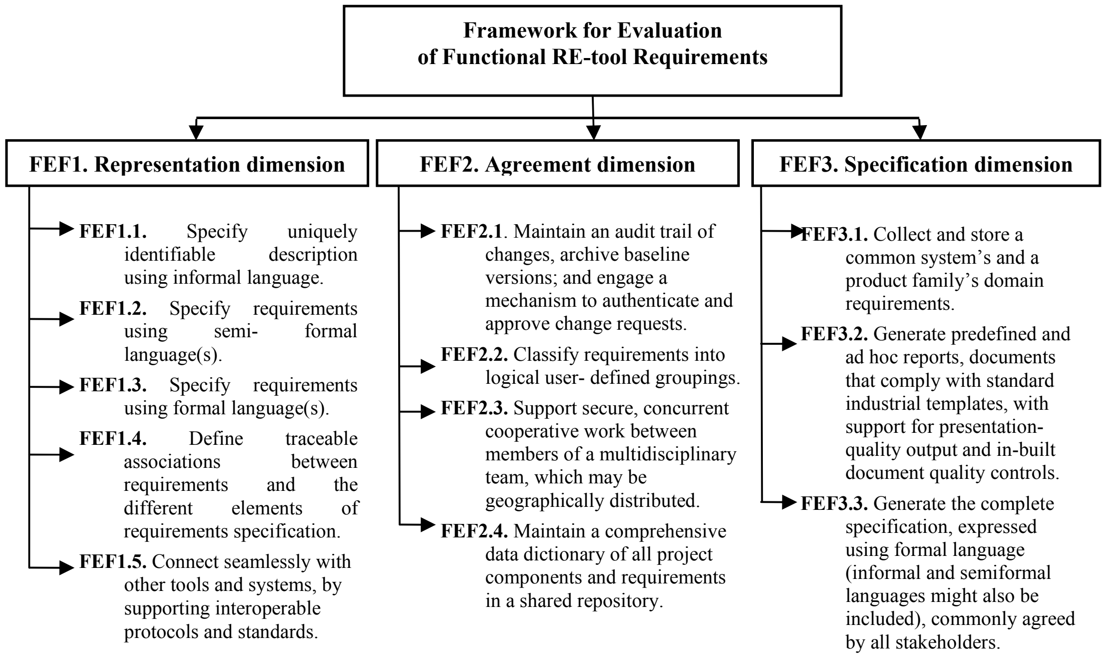

Framework for Evaluation of Functional RE-tool Requirements
FEF1. Representation dimension
FEF2. Agreement dimension
FEF3. Specification dimension
FEF1.1. Specify uniquely identifiable description using informal language.
FEF1.2. Specify requirements using semi- formal language(s).
FEF1.3. Specify requirements using formal language(s).
FEF1.4. Define traceable associations between requirements and the different elements of requirements specification.
FEF1.5. Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.
FEF2.1. Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.
FEF2.2. Classify requirements into logical user- defined groupings.
FEF2.3. Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.
FEF2.4. Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.
FEF3.1. Collect and store a common system's and a product family's domain requirements.
FEF3.2. Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.
FEF3.3. Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.

Figure 5.1 Framework for Evaluation of Functional RE-tool Requirements

5.2.1 Features of the Representation Dimension

The *requirements representation dimension* deals with the degree of formality, where informal, semiformal and formal languages are used to create system requirements model. It is important to keep the traceability between different representations of the same requirements. The RE-tool should also support extensions of modelling techniques and facilities like importing, exporting of different requirements representations, and associating the RE-tool with other development tools. The requirements representation activities are described in Table 5.1.

Table 5.1 Activities of the Representation Dimension

	Features	Activities How does the RE-tool…	Activity description
Representation dimension	**FEF1.1.** Specify uniquely identifiable description using informal language.	**FEF 1.1.1** provide natural language description.	*Natural language* is the language in which humans communicate in everyday life. The requirements could also be defined using *professional language* (Pohl, 1994, Krogstie and Sølvberg, 1996), which is used by a set of persons working in a certain kind of area.
		FEF 1.1.2 allow specifying unique identification (ID) for each separate requirement.	Requirements ID is assigned when requirement is created and entered to a requirements database. It should be unique for each individual requirement.
		FEF 1.1.3 allow importing of requirements and their description from text document.	The initial user needs are usually stored in requirements documents that are text files. The tool should have means to import this information to the tool database, using semi or fully automated functionality.
	FEF1.2. Specify requirements using semi-formal language(s).	**FEF 1.2.1** provide tools for semiformal language description.	*Semiformal language* (Pohl, 1994) is language with a precisely defined vocabulary and syntax, but without a precisely defined semantics (e.g., ER-diagrams, OMT, UML, and DFD). The particular languages maintained in the RE-tool, are specified using non-functional process requirements.
		FEF 1.2.2 provide forward/ backward traceability between semiformal, and informal, formal descriptions.	While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceability relationships between different representation forms.
	FEF 1.3. Specify requirements using formal language(s).	**FEF 1.3.1** provide tools for formal language description.	*Formal language* (Pohl, 1994) is a language with a precisely defined vocabulary, syntax and semantics (e.g., Z-schemas, beta-notations, and aggregation models). The particular languages maintained in the RE-tool, are specified using non-functional process requirements.
		FEF 1.3.2 provide forward/ backward traceability between formal and informal, semiformal descriptions.	While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceability relationships between different representation forms.
	FEF 1.4. Define traceable associations between requirements and the different elements of requirements specification.	**FEF 1.4.1** provide functions for testing traceability between informal, semiformal and formal requirement description.	By defining traceability between different representation forms, it is important to ensure that the requirements model is not changed using different views. The activity is partially duplicated with FEF1.2.2 and FEF1.3.2. However it is important to have it here as the RE-tool evaluation purposes could differ in different environments and previous activities would not be chosen for assessment.
		FEF 1.4.2 create parent-child traceable relations between requirements.	The activity defines hierarchical parent-child traceability. Parent here is understood like a higher abstraction level requirement, which consists of lower abstraction level requirements – children.
		FEF 1.4.3 maintain peer-to-peer traceable relations between requirements.	Peer-to-peer relationship defines requirements trace on the same hierarchical level (on the same abstraction level).

Continuation of Table 5.1

		FEF 1.4.4 maintain traceable relation between various related information.	Related information could be characterised as additional requirements views, figures, requirements documents, elicitation, or negotiation material.
		FEF 1.4.5 maintain forward/ backward traceability between a source of requirements, the requirements and design.	Requirements source is understood as the material (document, related system or person), from where (or whom) requirements are elicited. Design here is the system description, which specifies how the system should be built and implemented.
	FEF 1.5. Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.	**FEF 1.5.1** allow importing/exporting requirements description from/to text documents.	Importing of textual information from text and/ore graphical documents eases requirements model construction. Exporting of textual and/or graphical requirements information helps to analyse them using other software tools. Both import and export functionality could be fully or semi-automated. The activity includes analysis of the interoperability relationships and associations between the RE-tool and other tools used in an organisation.
		FEF 1.5.2 allow importing/exporting requirements description from/to graphical documents.	

5.2.2 Features of the Agreement Dimension

The *requirements agreement dimension* deals with the degree of agreement among project participants. The activities, which describe the requirements agreement dimension, are shown in Table 5.2. The RE-tool should support different groups of users, including people skilled in usage of tools, and people not skilled in RE process, but who know the problem domain. The tool should support communication and collaborative work facilities in order to maintain rationale and to yield agreement among project stakeholders.

5.2.3 Features the Specification Dimension

The *requirements specification dimension* deals with the degree of requirements understanding at a given time moment. Activities for the requirements specification dimension are described in Table 5.3. The requirements specification should be supported by documentation and reports during the whole RE process. A requirements specification is prepared more effectively and efficiently if it is possible to reuse knowledge and experience for other similar problem domains and projects. RE-tools should provide standards in order to ensure the quality level of requirements specification. According to Pohl (1994) framework the final product of RE process is the commonly agreed among all stakeholders, represented in a formal language, and complete requirements specification.

Table 5.2 Activities of the Agreement Dimension

	Features	**Activities** **How does the RE-tool…**	**Activity description**
Agreement dimension	**FEF 2.1.** Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	**FEF 2.1.1** maintain user authentication to the system (i.e. user name, password).	RE-tool users are stakeholders of the RE process. When speaking about the product produced with the RE-tool, users consist of different groups: those who are responsible for product development, introduction and maintenance; those with financial interest, responsible for the sale or purchase; and those who have an interest in product use.
		FEF 2.1.2 allow grouping users into different groups.	The RE-tool has to allow definition of user groups, and assignment the users to them.
		FEF 2.1.3 allow creating different views (according to documents, requirements, attributes) for different groups of stakeholders.	Different user groups could have different interest in requirements activities. The tool should allow display of various requirements views and tool functionality according to the user group or individual user needs.
		FEF 2.1.4 register agreement/ rationale/ discussion/ negotiation/ changes/ history of requirements and by how it was achieved.	*Rationale* is a reference to a set of information which provides an explanation why the requirement has been included. The negotiation, discussion and history maintenance are the means to create and to maintain the requirements rationale.
		FEF 2.1.5 call the earlier requirement description/ versions and register them into history context.	*History* is chronologically ordered states of individual requirements or requirements model. The RE-tool has to support history control mechanism and keep change track. The earlier requirements model (or requirement) versions should be possible to upload.
	FEF 2.2. Classify requirements into logical user-defined groupings.	**FEF 2.2.1** allow specifying attributes/ properties of the requirement.	Definition of requirements attributes and properties include requirements prioritisation activities, where different requirements or their groups are characterised according to the importance, stability, time to market and similar characteristics.
		FEF 2.2.2 provide sorting according to different attributes/ properties.	Requirements sorting and filtering according to defined attributes and properties, helps to narrow analysis scope and to find the relevant requirements to a user.
		FEF 2.2.3 provide filtering according to different attributes/ properties.	
	FEF 2.3. Support secure, concurrent cooperative work between members of a multidisciplinar y team, which may be geographically distributed.	**FEF 2.3.1** provide platform independent interface for geographically distributed users.	Platform independent interface suggests means for cooperative geographically distributed teams to access the requirements model through Web browsers.
		FEF 2.3.2 allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement).	Modification of requirements model versions leads to working on the same requirements data at the same time. Approval of requirements helps to reach agreement about the requirements model. The requirements model is the object of negotiation and agreement about the requirements, and their changes. The RE-tool should ensure that different versions of the requirements model would not overlap but be artefacts to reach the agreement.
		FEF 2.3.3 provide a change approval cycle for multiple change negotiation and approval before posting into common repository.	

Continuation of Table 5.2.

	FEF 2.4. Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.	**FEF 2.4.1** provide the single repository or data and concept dictionary.	Data dictionary provides definitions of the main elements of the requirements model, RE-tool functionality, and related information.
		FEF 2.4.2 provide separate data dictionaries for non-technical users and technical users.	The RE-tool is used by users who have knowledge about the RE and software development process; and users who do not have such a knowledge. The RE-tool has to distinguish between both groups of users providing appropriate data dictionaries or suggesting process scenarios.
		FEF 2.4.3 provide the help system to the users.	The RE-tool has to provide help functionality, where all the RE-tool functions and features would be defined and explained.

5.2.4 Semiotic Quality and Frameworks for Functional RE-tool Requirements

The RE-tool evaluation framework is covered by the semiotic quality framework, (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003), which separates overall quality to physical, empirical, semantic, syntactic, pragmatic, percieved semantic, and social quality types. The way the semiotic quality types are connected with framework dimensions are depicted in Table 5.4 where ✔ means the correspondence between the quality goals and the features of the functional framework.

There are two basic means on the *physical quality*: externalisation and internaliseability. An RE-tool should support basic database functionality using a repository solution for the internal representation of the system requirements model. It should deal with functionality such as version control and configuration management and advance concurrency control mechanisms.

Empirical quality deals with error frequencies when a model is read or written by various users, it also applies to the coding and ergonomics of computer-human interaction for modelling tools. The RE-tool evaluation framework distinguishes a variety of elements, looks for error frequency, checks layout in documents, reports, graphs and diagrams. RE-tools have to prevent and detect such errors in order to cover empirical quality goals.

Syntactic quality has the goal of syntactic correctness. Requirements descriptions should be completed according to the syntax and vocabulary of the language. An RE-tool should provide the means for error prevention and error detection, which may help to prevent syntactic invalidity and incompleteness errors.

Table 5.3 Activities of the Specification Dimension

	Features	**Activities** **How does the RE-tool…**	**Activity description**
Specification dimension	**FEF 3.1.** Collect and store a common system's and a product family's domain requirements.	**FEF 3.1.1** enable selection and extraction of common domain requirements and requirements which differentiate systems in product line.	The activity emphasises the requirements reuse from the requirements repository, which is maintained by the RE-tool. Requirements from the repository could be selected using, for instance, free-based or/and discriminant-based methods (Mannion *et al.,* 1999; Kaindl and Mannion, 2005)
		FEF 3.1.2 incorporate requirements to a concrete project.	Incorporation of requirements (and their groups) into appropriate places of the requirements model helps to create the requirements specification faster as these requirements are complete and agreed in similar projects or domains (or project).
		FEF 3.1.3 adapt/ spread changes in domain requirements to concrete projects within domain.	When the requirements are changing in the requirements repository, the changes are spread to all related requirements within the domain.
		FEF 3.1.4 provide comparison of domain requirements feasibility.	The RE-tool has to support functionality for requirements analyses in the requirements repository by comparing, changing and storing them here.
	FEF 3.2. Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	**FEF 3.2.1** provide wizards for report generation.	Wizards help to prepare reports, which are not designed together with the RE-tool functionality.
		FEF 3.2.2 provide possibility to print report according to views and sorting.	The RE-tool has to maintain requirements sorting, filtering, prioritisation requirements agreement, negotiation and rationale maintenance reports which would be possible to view, send by e-mail, and print them out.
		FEF 3.2.3 provide possibility to print results of rationale, brainstorm and etc.	
		FEF 3.2.4 provide techniques for error checking.	All the reports have to be syntactically correct. The RE-tool should have functionality for error prevention, detection and correction.
	FEF 3.3. Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	**FEF 3.3.1** correspond to standards of software documentation.	The requirements specification should correspond to the international standards (the exact standard maintained by the RE-tool, is determined by the non-functional process requirements).
		FEF 3.3.2 correspond to standards, defined by an organisation.	The RE-tool should have means to define internal organisational standards for requirements specification. The internal requirements specification standards could be adapted following the international ones, or created according the organisational needs.
		FEF 3.3.3 support formal languages for complete, commonly agreed requirements specification.	The output of the RE process is an agreed, complete, represented in formal language requirements specification. The RE-tool has to ensure functionality for completeness, agreement and formality check.

Table 5.4 Coverage of RE-tool Framework by Semiotic Quality Framework
Ext. – externalisation, ***Int.*** – internalisability, ***Min. err. freq***. –minimal error frequency, ***Correct.*** - syntactic correctness, ***Valid.*** – validity, ***Comp.*** – completeness, ***Perc. valid.*** - perceived validity, ***Perc. compl.*** - perceived completeness, ***Compr.*** – comprehension, ***Agr*** – agreement.

Quality Framework		**Physical quality**		**Empirical quality**	**Syntactic quality**	**Semantic quality**		**Perceived semantic quality**		**Pragmatic quality**	**Social quality**
RET evaluation framework		***Ext.***	***Int.***	***Min. err. freq.***	***Correct.***	***Valid.***	***Comp.***	***Perc. valid.***	***Perc. compl.***	***Compr.***	***Agr.***
Representation dimension	**FEF 1.1**	✓		✓	✓						✓
	FEF 1.2	✓		✓	✓					✓	
	FEF 1.3	✓		✓	✓		✓			✓	
	FEF 1.4					✓		✓		✓	
	FEF 1.5	✓				✓	✓				
Agreement dimension	**FEF 2.1**		✓					✓	✓		✓
	FEF 2.2									✓	✓
	FEF 2.3									✓	✓
	FEF 2.4		✓					✓	✓		
Specification dimension	**FEF 3.1**		✓			✓	✓				
	FEF 3.2			✓				✓	✓		✓
	FEF 3.3						✓			✓	✓

Semantic quality is the correspondence between the model and the domain. An RE-tool should provide the means to reach semantic goals - feasible validity and completeness. Some of them could be consistency checking based on a logical requirements description and constructivity, the use of driving questions or baselines to improve completeness of a requirements specification.

Perceived semantic quality is the similar correspondence between the participant's interpretation of a model and participant's current explicit knowledge. Its goals are perceived validity and perceived completeness. To achieve these goals an RE-tool should provide the means for participant training, discussions, statement insertion and deletion.

Pragmatic quality is the correspondence between the model and audience's interpretation of it. Its goal is feasible comprehension. The comprehensive common repository would allow better understanding of the domain. RE-tools should also provide the means for inspection, visualization, filtering, explanation, execution, simulation, and prototyping.

Social quality deals with participant knowledge, including social and technical audience interpretation. The main activities for achieving the feasible agreement goal are model integration and conflict resolution, i.e., pre-integration, viewpoint comparison and conforming, merging, and restructuring.

5.2.5 Framework for Verification and Validation

In general, verification and validation refer to checking processes, which ensure that a system conforms to its specification and meets the needs of a customer. In particular, *verification* means with building the model right, whereas *validation* means building the right model. As no model is absolutely accurate, the purpose for verification and validation is to ensure that the conceptual model is sufficiently accurate. In order to ensure validity of the conceptual model, verification and validation should be performed already in the RE stage as well as in all the system engineering phases.

An RE-tool facilitates verification and validation of information between project stakeholders (Matulevičius and Strašunskas, 2002). Verification and validation is difficult to perform automatically, but a semi-automatic verification and validation process is a desirable feature of RE-tools, as is automatic validation of the further phases. Verification and validation support by the framework is provided in Table 5.5.

In the requirements representation dimension, the use of informal, semi-formal as well as formal representation languages have to be supported. Traceability between the descriptions ensures that they all uniquely identify requirements. Since formal requirements normally are built out of informal ones, process of the verification and validation should not be restrictive. Stakeholders can be expected to verify different descriptions, not only when placing trace mark-up marking trace from informal thought semiformal to formal descriptions, but also when the requirements are imported from or exported to external tools and languages.

In the requirements agreement dimension, the different views and specifications have to be maintained during the whole process. They help to collect information about conflicts and to determine who causes conflicts. Views help to aquire knowledge for verification and validation. The maintainance of communication, conversation, coordination and collaboration processes between participants as well as decision support leads to better and faster agreements and verification and validation of the RE process. Further, an RE-tool must support concurrent cooperative work between members of the multidisciplinary team, which may be geographically distributed. Participants save time and money while performing verification and validation. So the validation helps to increase the participants' in requirements model and specifications leading to a consensus of opinions.

Table 5.5 Verification and Validation Support by RE-tool Framework

	Features	Verification and validation support
Representation Dimension	FEF1.1.	Transformation process between informal, semi-formal and formal representations must be supported. Automatic or semi-automatic transition between formality levels facilitates Verification and validation of specification and agreement between different stakeholders. Verification and validation ensures that each requirement is unique.
	FEF1.2.	
	FEF1.3.	
	FEF1.4.	The impact of changes in one fragment have to be traced to related elements. This benefits in more efficient validation, re-validation could be avoided.
	FEF1.5.	Connection with other tools benefits in easier verification and ensures interchange of produced and validated fragments between different tools used in different system development steps.
Agreement Dimension	FEF2.1.	Version control and configuration management helps to track changes after verification and validation was performed and benefits in facilitated re-validation process and understanding the rationale behind the change.
	FEF2.2.	Cooperative work increases the confidence of product by facilitating the understanding and discussing the model produced. It also facilitates the development of consensus between stakeholders and ensure sufficient accuracy of the model. Possibilities of cooperative work for geographically distributed team reduce expenses for verification and validation.
	FEF2.3.	
	FEF2.4.	
Specification Dimension	FEF3.1.	This feature rise only necessity to validate domain appropriateness and to reuse already validated and verified set of common requirements. It benefits in reduction of delivery time.
	FEF3.2.	The specification could benefit in increased confidence of product. Verification and validation techniques have to be applied. Verification and validation errors and gaps within the specification can be detected. Reuse of requirements specification of already existing systems leads to better insight of the systems behavior and avoids misspecifications.
	FEF3.3.	

In the requirements specification dimension, the system could be improved by applying techniques for requirements verification and validation. The final RE product– requirements specification - must be complete, expressed in formal language, commonly agreeded among all participants and correspond to predefined standards.

5.3 Framework for Non-functional RE-tool Requirements

However, it is not enough to consider the functional RE-tool features during their evaluation. It is necessary to analyse the non-functional features which are the properties that the system must have (Robertson and Robertson, 1999). In literature non-functional requirements are also called as non-behavioural requirements (Davis, 1993) or quality requirements (Lauesen, 2002). This work uses the term *non-functional requirements* in order to distinguish them from functional ones.

5.3.1 Non-functional Requirements Models

Non-functional requirements models are normally defined as feature hierarchies, where the leaves provide some measurable attributes. The Boehm *et al.* (1978) model proposes a multilevel hierarchy or a tree of software criteria. The McCall, Richards, and Walters (1977) model is based on three uses of a system, i.e. system revision, operation and transition. This model defines different factors that describe the external view of the system, as it is perceived by end-users. Each factor is decomposed into criteria that describe the internal view of the system, as the perceived of software developer. The ISO/IEC 9126 (1991) model divides software characteristics down into six categories of characteristics of the software: functionality, reliability, usability, efficiency, maintainability and portability. These can further be broken into sub characteristics that one have measurable attributes. The non-functional requirements are classified into quantitative and qualitative ones (Chung *et al.*, 2000). They are also separated between developer-oriented and usage-oriented quality factors (McCall, Richards, and Walters, 1977; Firesmith, 2003a).

The non-functional requirements models could be used only as the checklists, they do not take in the account that the organisational requirements normally differ from environment to environment. The non-functional requirements models provide only static aspects of an evaluation of the system without consideration of dynamic aspects for the system support.

All the non-functional requirement models mentioned above, consider qualities or abilities, that system should posses. None of them analyses how the system should behave in the environment, what specific processes it should support. Further, the environment requirements describing what internal and external organisational characteristics affect the system, are not considered. Therefore, the above mentioned non-functional requirements as such are not optimised for the tool assessment.

Kotonya and Sommerville (1998) classify non-functional requirements to process, product and external requirements. Process requirements are consequence of organisational policies and procedures and they are derived from user's organisation. Product requirements may be derived directly from the user needs and they specify the need for the delivered product to behave in a particular way. External requirements are all requirements which arise from factors external to the system and its development process. However this classification of non-functional requirement describes the *system development*, but not *system evaluation* requirements.

5.3.2 Requirements of the Evaluation Process

But before starting describing the evaluation framework of non-functional RE-tool requirements, it is important to make distinction between evaluation process requirements and the process requirements which describe how organisation performs.

When assessing the RE-tool(s), the participants have to characterise decisions about evaluation settings. In this work these requirements are called *evaluation process requirements*. They are:

- selection of the evaluation criteria and framework (e.g., Wieger (1999) requirements; Lang and Duggan (2001) requirements, INCOSE, priority-based (Haywood and Dart, 1997), role-based (Hoffmann *et al.*, 2004) frameworks, or two frameworks suggested in this chapter);
- selection of the evaluation approach (e.g., PORE (Maiden and Ncube, 1998), OTSO (Kontio, 1996), CAP (Ochs *et al.*, 2000), scenario-based (Feblowitz and Greenspan, 1998), STACE (Kunda, 2003), quality-based (Franch and Carvallo, 2002) or R-TEA described in Chapter 6);
- about the evaluation and decision support systems (e.g., paper-based tools, like evaluation forms, questionnaires; software tools, like SCARLET-Advisor (Maiden, *et al.*, 2003), DesCOTS (Grau *et al.,* 2004, Carvallo *et al.*, 2004) or framework prototype presented in Chapter 8);
- evaluation activities and responsibilities (e.g., who in an organisation are participating in the RE-tool assessment; how many tool demonstration and evaluation sessions are required);
- evaluation techniques used during the RE-tool assessment (e.g., templates, scenarios, RE-tool tutorials, specification exemplar described in Chapter 6).

The evaluation process requirements are defined before starting the actual RE-tool assessment. The selection of the evaluation process requirements depends on the environment needs and goals. Some proposals suggest combining the quality-based selection (Franch and Carvallo, 2002) and INCOSE framework, PORE (Maiden and Ncube, 1998) and the role-based framework (Hoffmann *et al.*, 2004). However, only R-TEA defined in Chapter 6, combines the evaluation guidelines and two frameworks described in this section, and gives an account to a stepwise RE-tool evaluation.

5.3.3 Overview of the Framework

The *framework for evaluation non-functional RE-tool requirements* is shown in Figure 5.3. The process requirements specify the organisational work processes. The product requirements describe the desired characteristics of the RE-tool according to non-functional requirements models. External requirements divided into organisational requirements and requirements to business parties, specify the requirements placed upon the RE-tool from within and outside the environment. In the following sections the requirements categories are analysed in detail.

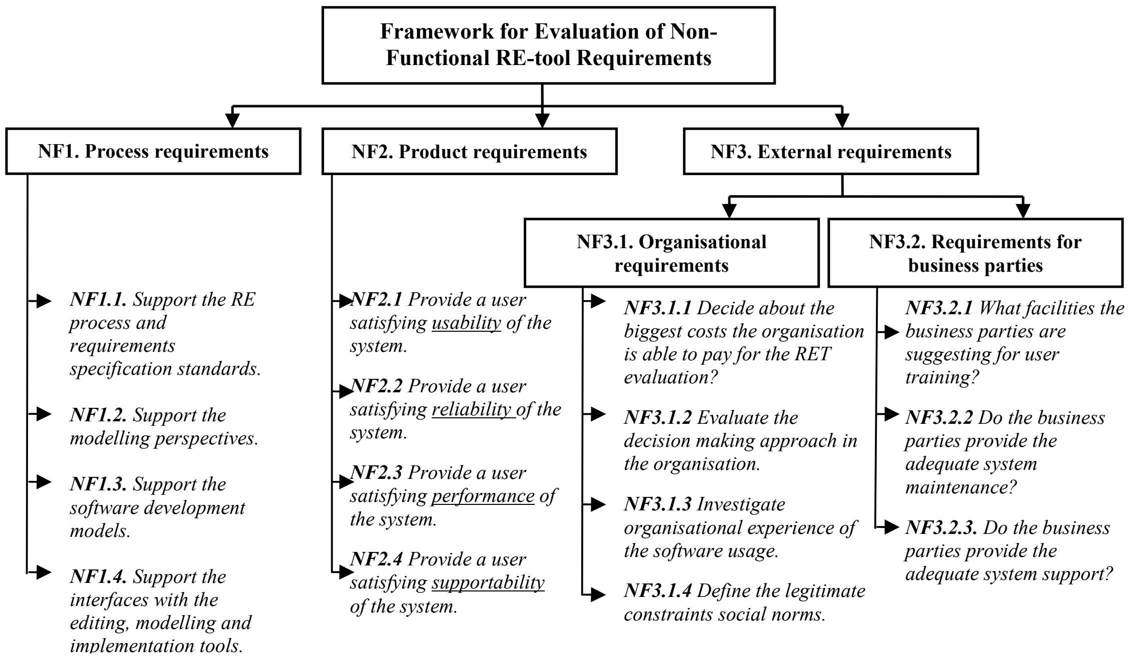

Framework for Evaluation of Non-Functional RE-tool Requirements
NF1. Process requirements
NF2. Product requirements
NF3. External requirements
NF3.1. Organisational requirements
NF3.2. Requirements for business parties
NF1.1. Support the RE process and requirements specification standards.
NF1.2. Support the modelling perspectives.
NF1.3. Support the software development models.
NF1.4. Support the interfaces with the editing, modelling and implementation tools.
NF2.1 Provide a user satisfying usability of the system.
NF2.2 Provide a user satisfying reliability of the system.
NF2.3 Provide a user satisfying performance of the system.
NF2.4 Provide a user satisfying supportability of the system.
NF3.1.1 Decide about the biggest costs the organisation is able to pay for the RET evaluation?
NF3.1.2 Evaluate the decision making approach in the organisation.
NF3.1.3 Investigate organisational experience of the software usage.
NF3.1.4 Define the legitimate constraints social norms.
NF3.2.1 What facilities the business parties are suggesting for user training?
NF3.2.2 Do the business parties provide the adequate system maintenance?
NF3.2.3. Do the business parties provide the adequate system support?

Figure 5.2 Framework for Evaluation of Non-functional RE-tool Requirements

5.3.4 Process Requirements

Process requirements in Table 5.6 are constraints placed upon the user's development process and work practice. Process requirements characterise the RE process in the RE-tool users' organisation. They are related to the process followed while analysing the new system which is produced using an RE-tool (but not particular requirements followed while developing or evaluating the RE-tool itself).

Table 5.6 Non-functional Process Requirements

	Feature **RE-tool should…**	**Working process characteristics**
Process requirements	**NF1.1.** Support the selected RE and requirements specification standards.	IEEE std 830-1998, ISO 9126, NASA-DID-P200, Internal organisational standards.
	NF1.2. Support the selected modelling perspectives.	Structural, functional, behavioural, rule, object, communication, actor and role.
	NF1.3. Support the software development models.	Waterfall, spiral, prototyping, transformational, knowledge-based, domain-based, and RUP.
	NF1.4. Support the interfaces with the editing, modelling and implementation tools.	Editing, communication, modelling, implementation, testing tools, database management systems.

An organisation might specify that an RE-tool have to maintain a predefined modelling perspective (and modelling languages), because the potential users are already trained and skilled in it. Similarly, the organisation might already be using some editing, modelling and implementing tools, so it is important that the RE-tool should have data interchange and interoperability features with these tools. Process requirements should also define what RE process or requirements specification standards should be compatible with the RE-tool.

5.3.5 Product Requirements

Product requirements are features which specify the desired characteristics that an RE-tool must possess (Table 5.7). They could be defined according to non-functional requirements models (McCall, Richards, and Walters, 1977; Boehm *et al.*, 1978; ISO/IEC 9126 standard, 1991) considered in section 5.2.1. In this work four product requirements of the RE-tools are selected. They are usability, reliability, efficiency and supportability. However, other requirements presented and described in the non-functional requirements models, are not excluded from evaluation purposes, but remains as possible candidates if the evaluator sees the need to analyse them.

Usability. Usability considers how easy it is to learn the system, how efficient it is for carrying out day to day task (Leffingwell and Widrig, 2000; Lausen, 2002), also it considers the capability of the system to be understood, learned, used and liked by the user, when used under specified conditions (ISO/IEC 9126 standard, 1991).

Usability could be evaluated by performing usability test and heuristic evaluation techniques (Lausen, 2002). During the usability test the users try to carry out realistic tasks using the tool or a mock-up of it. The tool evaluator observes which problems the user encounters. During the heuristic evaluation the usability expert identifies the problems of the tool usage.

In order to evaluate the tool usability, the evaluator has to consider several usability factors (ISO/IEC 9126 standard, 1991): 1) understandability (the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use); 2) learnability (the capability of the software product to enable the user to learn its application), operability (the capability of the software product to enable the user to operate and control it); and likability (the capability of the software product to be liked by the user). Elsewhere, according to Leffingwell and Widrig (2000), five usability measures should be considered by answering corresponding questions:

– Ease of use. How easy is the system to learn for various groups of users?
– Task efficiency. How efficient is it for the frequent user?
– Ease of remembering. How easy is it to remember for the occasional user?

Table 5.7 Non-functional Product Requirements

	Feature	Measures	Questions
Product requirements	**NF2.1.** Provide a user satisfying <u>usability</u> of the system.	Capability of the system to be: – understood; – learned; – used and liked by the user; – used under specified conditions.	How easy is the RE-tool to learn for various groups of users? How efficient is the RE-tool for the frequent user? How easy is the RE-tool to remember for the occasional user? How satisfied is the user with the RE-tool? How easy is it to understand what the RE-tool does?

	NF2.2. Provide a user satisfying <u>reliability</u> of the system.	Accuracy
		Error tolerance;
		Consistency;
		Recoverability;
		Availability.
			What failure types are evaluated for the RE-tool?
			What is the availability of the RE-tool and its components?
			How long is the RE-tool allowed to be out of operation after it has failed?

	NF2.3. Provide a user satisfying <u>performance</u> of the system.	System performance; Required amount of resources; Data and result accuracy.	What is the minimum and average response time for certain data transaction or operation? What is minimum and maximum throughput for certain operations? What is the capacity of the RE-tool? What is resource utilization of the RE-tool? Do users receive the exact output?
	NF2.4. Provide a user satisfying <u>supportability</u> of the system.	Correctability Extensibility	What type (e.g., corrective, adaptive, preventive, perfective) of maintainability is performed most often? What are reasons for the maintainability activities? How many steps (activities, corrections) should be performed to reach the desirable state of the tool?

– Subjective satisfaction. How satisfied is the user with the system?
– Understandability. How easy is it to understand what system does?

Reliability. Reliability analyses how frequently the system malfunctions and describes the percentage of time it is available (Leffingwell and Widrig, 2000; Lausen, 2002). A number of measures like accuracy (Boehm *et al.*, 1978; McCall, Richards and Walters, 1977), error tolerance (Boehm *et al.*, 1978; ISO/IEC 9126 standard, 1991), consistency (Boehm et al., 1978; McCall, Richards and Walters, 1977) recoverability and availability (ISO/IEC 9126 standard, 1991) are suggested in the non-functional requirements models. Error tolerance is defined as the capability of the software to maintain a specific level of performance in cases of software faults or of infringement of its specific interface. Recoverability is the capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure. Availability is the capacity of the software to be in a state to perform a required function at a given point of time, under stated conditions of use.

While evaluating the reliability it is important to define what the failure and its type are. This means that reliability influences the supportability, because they both are dealing with system correction. Reliability also affects the efficiency because, for example, the reason the system is not available could be the over-consuming of system resources at certain time moments. Reliability also influences the security, because the system could fail because of the illegal breaks into the system. Reliability also deals with accuracy, which define the magnitude of defects in quantitative data.

The basic problem of reliability is to predict when a system would fail. Most commonly used metrics to evaluate reliability are mean time to failures (MTTF) and mean time to repair (MTTR) (Fenton and Pfleeger, 1997, Leffingwell and Widrig, 2000). The list of activities should be carried out while evaluating the reliability and the following questions should be answered:

– What failure types are evaluated? The evaluator should consider the important failure types, which influence the ability to use the system and has impact to user satisfaction.
– What is the availability of the system and separate system components? The evaluator should apply a reliability test to investigate the mean time to failure.
– How long is the system allowed to be out of operation after it has failed? The mean time to repair the system after fail depends on the difficulty of the fail – if the fail is difficult to repair, the time to repair will increase, since the organisation will have to apply to the tool vendor.
– Does the user receive the exact and concrete output? This question considers the accuracy of the reports and results which are produced by the system.

Performance. Performance defines how fast the system responds, how many resources it uses, how accurately it computes values (Boehm *et al.*, 1978; Lausen, 2002). ISO/IEC 9126 standard (1991) defined performance (efficiency) as the capability of the system to provide the required performance relative to the amount of resources used under stated conditions. It incorporates two criteria: storage efficiency (the capability of the software system to provide response and processing times and throughput rates when performing its function under stated conditions) and execution efficiency (the capability of the software to use appropriate resources in an appropriate time when the software performs its function under stated conditions).

In order to evaluate performance of RE-tools evaluators should carry performance tests and to answer the following questions:

– What is the minimum and average response time for certain data transaction or operation? Evaluation should define the most usually performed operations and to evaluate their efficiency in time space.
– What is minimum and maximum throughput for certain operations? The efficiency test should be performed using different test data size.
– What is the capacity of the system? Evaluator should evaluate the number of customers or transactions the system could accommodate.
– What is resource utilization of the system? In order to evaluate needed resources, evaluator could consider memory, disk space, and database size needed to perform operations efficiently.

Maintainability. Maintainability defines how the system could be expanded by adding new features, by adopting the system to standards, modelling perspectives, and by relating the system to other software tools (Leffingwell and Widrig, 2000), or in other works maintainability is defined as the capability of the system to be modified (ISO/IEC 9126 standard, 1991). Boehm *et al.* (1978) considers maintainability as the general utility, which consists of testability, understandability and modifiability. ISO/IEC 9126 standard (1991) characterises maintainability by four measures: analysability, changeability, stability and testability. Analysability is described as the capability of the product to be diagnosed for deficiencies or causes of failures in the software or for the parts to be modified to be identified. Changeability is the capability of the software product to enable a specified modification to be implemented. Stability is defined as the capability of the software to minimize unexpected effects from modifications of the software. And testability is the capability of the software product to enable modified software to be valid.

Firesmith (2003a) defines maintainability as the ease with which a system could be maintained between major releases. The measures include correctability (the ease with minor defects can be corrected between major releases while it is in use by its user) and extensibility (the ease with which an application or component can be enhanced in the future to meet changing requirements or goals).

Maintainability involves several types of changes (Fenton, Pfleeger, 1997). Corrective maintainability involves finding and fixing faults. The changes could be adaptive, when the system changes in some way and is adapted to preserve functionality, performance or the needed specification and working standards are applied. Maintainer also performs preventive maintenance, where the problems are fixed before the system users see them. And finally, perfective maintainability might involve the addition or improvement of existing functionality in a working system. Taking in account all four maintenance activities evaluator could define a measure - mean time to repair (MTTR), which is the average time it takes to implement a change and restore the system to working order. In order to calculate this measure the needed information include problem recognition time, administrative delay time, maintenance tools collection time, problem analysis time, change specification time, and change time (including tests and reviews). Other measures may include (if the information is collected and available) ration of total change implementation time to total number of changes implemented, number of unresolved problems, time spent on unresolved problems, percentage of changes that introduce new faults, number of modules modified to implement a change.

The evaluator also should evaluate whether the organisation has available resources (implementation tool, skilled workers) to perform maintainability themselves or the organisation would have to use services of the tool vendor.

5.3.6 External Requirements

External requirements are the RE-tool features which may be placed on the product and process, and are derived from within and outside the organisation. External requirements represent the social and economical factors and are divided into organisational requirements and requirements for business parties.

Organisational requirements characterise the RE-tool according to the costs and business issues (Table 5.8). Costs include the direct cost, such as the price of the tool and indirect cost, such as the cost of the tool evaluation and user training. Business issues include people and process concerns such as management support, internal organisational politics, staff skills and attitudes.

An organisation consists of individuals who view the world in their own specific way, because each of them has different experience, and education. Activities in an organisation are also influenced by the environmental and social norms, and governmental and economical regulations.

The organisation consists of different hierarchical levels. Workers of different level have perceptions about the system. The organisational requirements should be separated into two interest groups (Seddon, Graeser and Willcocks, 2002). The interest of individual users would include individual productivity, user satisfaction, information quality and perceived usefulness of the system. The interest of management would consider return on investment, return on management, cost savings, sales growth, and system availability.

Requirements for business parties deal with the RE-tool vendors' or consultants' performance and reliability (Table 5.9). They include the infrastructure and stability, the vendor reputation, customer base and track records.

The user satisfaction about the system depends on their ability to perform the certain functionality. Users should be suggested the training on tool functionality, vendors react to the users' requests and they should maintain the network of geographically distributed offices, provide consultancy and assisting services via Internet or phone. The requirements for business parties also involve the system support, when the new releases of the tool are produced. The business parties should help the organisation to adapt new functionality of new tool releases.

Table 5.8 Non-functional External Organisational Requirements

	Features	Measures	Description
Organisational requirements	**NF3.1.1.** Decide about the biggest costs the organisation is able to pay for the RE-tool evaluation?	Tool prices	Direct tool price, tool running costs (like new computers, RAM, etc.)
		Evaluation costs	Evaluation and transition costs (changing from the old to a new working practice), time and resources need to evaluate tool.
		Maintenance and support costs	Future operational costs, support costs, maintenance/ upgrades etc.
	NF3.1.2. Evaluate the decision making approach in the organisation.	Objectivistic (management requirements);	An objectivistic world-view describes an organisation, where stakeholders can map the required RE-tool without changing the organisational processes.
		Constructivistic (users' requirements)	A constructivistic approach supports stakeholder knowledge externalisation and internalisation processes, and in such a way makes the organisational environment part of the individual understanding. The RE-tool selection is the process of negotiation of the environment and RE-tool suitability to this environment.
	NF3.1.3. Investigate organisational experience of the software usage.	Degree of the RE-tool requirements	Different user experience contributes with different degree of abstraction for the tool functional and non-functional feature evaluation. More experienced users could require evaluating bigger number of requirements or only the particular requirements, which would contribute with evaluation of 'needed' tool features.
		Degree of RE-tool evaluation needed	Different user experience contributes with different degree of abstraction for the tool functional and non-functional feature evaluation. More experienced users could require the more detailed tool analysis of a particular feature.
	NF3.1.4. Define the legitimate constraints social norms.	Political constraints and laws	Political constraints and laws specify the norms, which exist in the organisation, and the country where organisation is situated.
		Cultural constraints	Cultural constraints describe the national traditions, customers, which could effect tool evaluation and usage.
		Social constraints	Social constraints characterise the organisational workers relationships, organisational workers hierarchy and relationships between management and workers.

5.4 Relationships between Frameworks

In the quality-based selection (Franch and Carvallo, 2002) the importance of defining relationships between the quality subcharacteristics and attributes is considered. The possible types include collaboration, damage and dependency. Mannion *et al.* (1999) and Kaindl and Mannion (2005) define a set of relationships when selecting requirements from an application family model. The communalities and variability of the systems is considered through direct parent-child links and single adaptor, multiple adaptor and option relationships.

Table 5.9 Non-functional External Requirements for Business Parties

	Features	Measures	Description
Requirements for business	**NF3.2.1.** What facilities the business parties are suggesting for user training?	Availability (costs) of training seminars;	Describes availability of tool training seminars organised by the tool vendor.
		Time spent on the phone before getting an answer;	Does the vendor or consultant provide phone, e-mail and online consultancy services using messenger systems (e.g.: MSN, ICQ and Skype). How long does it take to get answers about the tool usability and functionality?
		Average response time for online help;	
		Average response time for help via e-mail;	
		Availability of information in the vendors' Web page.	Does the vendor of the tool provide the appropriate support in the tool Web page? Is the information informative and relative to the emerging question? How often does the vendor update the Web information?
	NF3.2.2. Do the business parties provide the adequate system maintenance?	Quality of the maintainability services;	The basic supportability (maintainability) metrics: ratio of total change implementation time to total; number of changes implemented; number of unresolved problems; time spent on unresolved problems; percentage of changes that introduce new faults; number of modules modified to implement a change.
		Availability at any time;	Is vendor available at any time when maintainability problems of the tool emerge? How long does it take to get service form the vendor?
	NF3.2.3. Do the business parties provide the adequate system support?	Quality of support during new version releases;	Does vendor representative participate during tool updates? Does the vendor provide teaching services of new tool versions?
		Quality of updates of the current tool version	Do updates solve the old version problems? Are new tool versions more effective, easy to use, etc. than older ones?

In this section relationships between both frameworks are explained and shown in Figure 5.4. The relationships characterise the dependencies between the functional and non-functional features.

R.1. *FEF1.1, FEF1.2, FEF1.3 and NF1.2* – relationship between the modelling perspective and the languages supported by the RE-tool.

The relationship describes links between the representation using informal (FEF1.1), semiformal (FEF1.2) and formal (FEF1.3) languages and the instances of these languages (NF1.2). The evaluator should indicate the particular languages for each formality level, which are used to create requirements model.

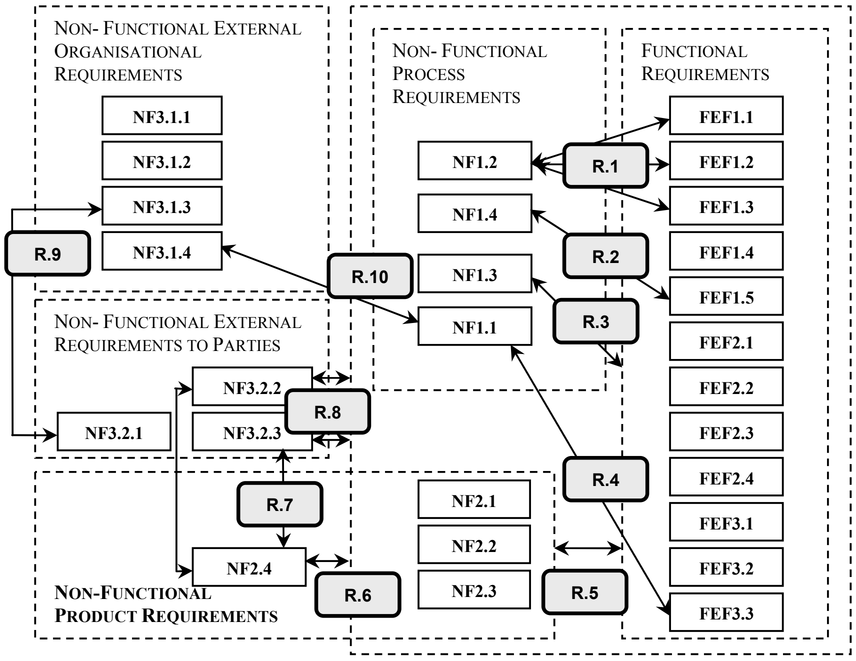


Figure 5.3 Relationship between Frameworks

R.2. *NF1.4* and *FEF1.5* – relationship between the tools already used and the RE-tool.

The relationship describes associations and interoperability between the tools used in the organisation (NF1.4) and the tools which needs to be associated with the RE-tool (FEF1.5).

R.3. *NF1.3 and RE-tool functional features* consider the software development model, used by user.

The relationship should highlight the software development process and analyse if the RE-tool functionality (features FEF1.1, FEF1.2, …, FEF3.3.) support the highlighted development process (NF1.3).

R.4. *NF1.1* and *FEF3.3* – relationship between the RE process and specification standards.

The relationship describes association between the international specification standards (NF1.1) and the standards, which should be maintained in the RE-tool (FEF3.3). The standards could help to prepare the RE process output in a structured and completed way. The relationship should result in selection of the specification standard used in the organisation and in the RE-tool.

R.5. *Non-functional RE-tool features* (*NF2.1*, *NF2.2*, and *NF2.3*) and *functional RE-tool features* – relationship between the RE-tool usability, efficiency, reliability and functionality.

The relationship analyses how the RE-tool functionality (features: FEF1.1, FEF1.2, …, FEF3.3) corresponds to RE-tool usability (NF2.1), reliability (NF2.2) and performance (NF2.3).

R.6. *NF2.4* and the *functionality* consider the need to maintain the non-functional RE-tool features against the functional characteristics by the customer.

The relationship analyses the maintainability (NF2.4) of the RE-tool. Maintainability should be performed here by internal organisational workers. It includes maintenance of the RE-tool functionality (features FEF1.1, FEF1.2, … FEF3.3), and non-functional product features: usability (NF2.1), performance (NF2.2) and reliability (NF2.3).

R.7. *NF3.2.2*, *NF3.2.3* and *NF2.4* – relationship of maintenance and support between the customer and the business parties.

The relationship analyses the degree of maintenance (NF2.4) performed by the internal organisational workers and maintenance (NF3.2.2) and support (NF3.2.3) performed by external business parties (vendors and consultants).

R.8. *NF3.2.2*, *NF3.2.3* and the *functionality* – relationship of the RE-tool maintenance and support by the business parties.

The relationship analyses the maintenance (NF3.2.2) and support (NF3.2.3) of the RE-tool by the business parties (tool vendors and consultants). Maintenance and support should include the RE-tool functionality (features FEF1.1, FEF1.2, … FEF3.3), and non-functional product features: usability (NF2.1), performance (NF2.2) and reliability (NF2.3).

R.9. *NF3.1.3* and *NF3.2.1* – relationship between the customer's knowledge and the need for training.

The relationship describes the correspondence between degree of organisations workers experience (NF3.1.3) and teaching (NF3.2.1) of the RE-tool functionality.

R.10. *NF1.1* and *NF3.1.4* – relationship between the standards and the social, organisational, law and cultural factors.

The relationship describes links between cultural, social, law, and organisational norms (NF3.1.4) and the standards used to prepare the requirements specification (NF1.1). The relationship should conclude how standards should be adapted in the organisation, what norms should be kept in mind, how internal organisational standards should be defined.

They describe the way what should considered together when evaluating the RE-tools. For example, if the importance of semi-formal language (FEF1.2) is highlighted, the targeted language could be specified as a non-functional requirement NF1.2 (relationship R.1). Other relationships characterise what RE-tool maintainability issues when acquiring it to the organisation. For instance R.6 relates the degree of RE-tool maintenance and the tool functionality and usability. All the relationships selected when preparing a requirements specification for the tool acquisition, later on are evaluated when analysing the RE-tools candidates as discussed in Chapter 6.

5.5 Chapter Summary

In this chapter two evaluation frameworks are presented: namely one for functional and one for non-functional RE-tool requirements. The framework for functional RE-tool requirements is based on the NATURE framework (Pohl, 1994; Pohl, 1996) which separates three dimensions of the RE process: requirements representation, requirements agreement and requirements specification. The requirements representation describes how RE-tool(s) could represent requirements model using informal, semiformal and formal languages. The requirements agreement characterise how agreement could be reached among the project stakeholders. The requirements specification describe how complete, formal and agreed requirements specification could be achieved by the means of the RE-tool(s). The functional framework supports the evaluation of the RE-tools according the goals and means to reach these goals, as it is discusses in the semiotic quality framework (Lindland *et al.* 1994, Krogstie 1998).

The evaluation framework for the non-functional RE-tools requirements is described according to (Kotonya and Sommerville, 1998) framework which categorised to work process, product, organisational requirements and requirements for business parties. The work process requirements describe the processes which take place in the organisation when it performs RE for a new software product (but not for the evaluated RE-tool(s)). It is important to separate definition of the work process requirements against the requirements which define the evaluation process. The product requirements characterise the RE-tool abilities, like usability, maintainability, performance and reliability. The organisational requirements characterise the organisational environment. The requirements for business parties present the vendors (or consultants) abilities to support their RE-tool.

A definition of relationships between the evaluation frameworks concludes this chapter. The main emphasis of relationships is placed on the tool maintainability characteristics and the tool acquisition to the organisational environment.

In the next chapter the RE-tool evaluation approach summarises the application of the evaluation frameworks. The R-TEA method guides through the RE-tool assessment.

CHAPTER 6

RE-tool Evaluation Approach

In the previous chapter two evaluation frameworks are presented. They could help during evaluation of the RE-tools. However, only having the frameworks, an evaluator would have difficulty in applying them for evaluation purposes. Therefore rules and guidelines are needed for the framework application. In this chapter it is described how the evaluation frameworks are used to construct a specification exemplar which initiates the RE-tool evaluation approach (R-TEA). The guidelines presented in R-TEA facilitate the application of the frameworks in order to assess and select the RE-tool according to the organisational needs. A demonstrative example of R-TEA application is described and the R-TEA method is compared with the general tool selection approaches surveyed in Chapter 4.

6.1 Guidelines for RE-tool Evaluation Methodology

The working hypothesis is that if a qualitative methodology is to be used for RE-tool acquisition, it would contribute with a properly selected RE-tool, which would help to prepare a qualitative and complete specification. A proper defined methodology saves time afterwards, helps to highlight the evaluation criteria and guides through the process. Here, some guidelines to adapt the RE-tool evaluation methodology are suggested, according to a framework proposed by Krogstie and Sølvberg (1996). The methodological guidelines are:

– *Apply a constructivistic world-view*. An objectivistic world-view describes an organisation, where stakeholders can map the required RE-tool without changing the organisational processes. In comparison to this, a constructivistic approach supports stakeholder knowledge externalisation and internalisation processes, and in such a way makes the organisational environment part of the individual understanding. The RE-tool selection is the process of negotiation of the environment and RE-tool suitability to this environment. Therefore, the methodology should support the constructivistic world-view.

- *Be ready to change work processes*. Sedigh-Ali, Ghafoor and Paul (2001) stress the importance of evaluating the organisational maturity before the tool acquisition. RE-tools may lead to process improvement, but only if the organisation has the ability to take advantage of the tools. If the organisation has an immature process, there are probably other improvement steps that should be taken before considering the tool acquisition. The organisation should be ready to accept the need to reorganise work processes in order to fully utilize automated support.
- *Evaluate the software already available in the organisation*. Usually the RE-tool will not be a stand-alone application in the organisation, but will be used together with other tools relevant for the software development. Hence, the RE-tool ability to interoperate with other tools should be a part of the general tool evaluation.
- *Involve users in the RE-tool evaluation*. The users have different experiences arising from work and other activities; however, they are the true evaluators and can describe lacking points of automated support.
- *Teach users the tool functionality.* Surveys report about complex RE-tool functionality. In order the tool selection to be successful, the evaluation methodology should comprise the necessary teaching activities. The potential tool users should perform themselves the evaluation activities, and the evaluation team should provide the necessary help during the tool tests.
- *Evaluate the maturity*. The methodology maturity could be evaluated by methodology usage time (used for a long time or new), awareness (used in many places, or only by one company or research group), and application area (tried out in practice: academia or/and industry).
- *Reuse the collected information*. Reuse of information could be divided to three cases: 1) reuse of the same type of tool evaluation by the same organisation; 2) reuse of other type of tool evaluation by the same organisation; and 3) reuse of the same type of tools but by a different organisation. The first case could happen if organisational needs or of tool versions changes. In the second case the reuse could contribute with environment description; however a lot of new work must be done investigating new tools, and new organisational needs. The third case of reuse might be significant if the organisations have similar needs.

However, the information accessibility will be a major problem, unless two organisations are divisions within a larger company or have a strong cooperation. On another hand the collected information of the current session should be also packaged for the future needs which include the three cases mentioned above.

6.2 Specification Exemplar and Application Guidelines

The RE-tool evaluation approach (R-TEA) is based on the evaluation frameworks for functional and non-functional RE-tool requirements, which are adapted to an *specification exemplar* (Matulevičius and Sindre, 2004; Matulevičius and Sindre 2005). Specification exemplars generally amount to a self-contained, informal description of a problem in some specification domain (Feather *et al.*, 1997); they are proposed as unique input for the specification process. Exemplars thus define, in a broader sense, model specification tasks. They are to be considered immutable; the user must do his/her best in order to produce a specification.

Specification exemplars are indispensable for illustrating and explaining methodologies (Yu and Cysneiros, 2002). They provide instances that allow abstract constraints and descriptions to come alive through domain settings and scenarios. An exemplar that has become familiar to a research community can be a valuable resource in a number of ways:

- It can be used to capture the set of problems faced by the community. Specific research projects can use it to delineate research objectives without necessarily addressing the entire set of problems.
- It can provide a familiar application context as a basis for discussing the methodological issues. Knowledge and experience can be reused and allow attention to focus on the solutions to problems.
- For someone learning about a new method, the availability of an exemplar can help quickly place the method in relation to other methods. For someone developing a method, an independent exemplar provides an objective vehicle for assessing capabilities and limitations. It can also be invaluable for third parties to compare and evaluate methodologies.
- It can also be a great help to someone from outside the research community, to quickly grasp the issues faced, to see how well they are addressed by the state of the research. Potential users can use it to relate to their domain problems, thus recognizing the strengths and limitations of other approaches.

There are three purposes for using a specification exemplar (Feather *et al.*, 1997): first, it is advancing a single research effort; second, it is promoting research and understanding among multiple research groups; finally, it is contributing to the advancement of software development practice.

6.2.1 Specification Exemplar for RE-tool Evaluation

The example of specification exemplar is shown in Figures 6.2 and 6.3. It is based on two RE-tool evaluation frameworks presented in Chapter 5. The specification exemplar should not be made to favour one particular RE-tool or tool vendor. It should be possible to evaluate tools with a limited functionality. Hence the steps to follow must not break down if some tasks are not supported in the evaluated tool. The specification exemplar does not need to specify whether the task must be done automatically or manually. As RE is an iterative process, an exemplar should contain steps modifying RE-tool requirements, analysing conflicts between RE-tool requirements and restructuring the whole specification. The specification exemplar should also contain the guidelines on how to measure performance of the tools and to apply the specification exemplar itself.

The specification exemplar for RE-tool evaluation supports a well-known domain, and it contributes to advancing a single research effort. The problem aims to suggest the reality check through the choice of domains – organisations, where the tool selection task appears.

But the frameworks do not consider the applicability issues. The specification exemplar facilitates elicitation activities, as it is necessary to handle information from multiple stakeholders. Further, the application of the specification exemplar provides the evaluation according to the specific organisation needs. For example, if elicitation yields a high priority for traceability, there would be more RE-tool requirements (features selected by the users) for which traceability is relevant. With a lower priority on traceability, there would be fewer such RE-tool requirements. In this sense the users are not required to select all the features but just the most relevant ones according to their needs.

The specification exemplar may contribute to advancing the software development practice. The specification exemplar reduces the RE-tool evaluation costs and time, as the criteria are reused under different settings. The specification exemplar contributes with the basis for the architecture, design, implementation and maintenance of the RE-tools. However, the following possible limitations are identified:

- RE-tools acquisition is done fairly seldom. But the specification exemplar could be useful to evaluate the current RE support. The comparison does not require big investments, but it shows the economical benefits.
- The stakeholders could have limited technical knowledge and not be much experienced in the RE-tool acquisition. But the specification exemplar concentrates on the simple evaluation aspects.
- The particular choice of specification exemplar might lead to contamination of experimental data. The specification exemplar could be used as a test scenario, but this could affect the evaluators' attitudes. In order to mitigate this threat, it is suggested to look for the RE-tool tutorials or test scenarios (see next section) which would correspond to the best working process. It will focus on the evaluation, but not on the learning of the RE domain.

The separation between the exemplar and testing scenario is needed, because the organisation could also have different purpose for tool comparison, as different organisations are involved in different project types.

6.2.2 Specification Exemplar Application

The specification exemplar supports the RE-tool evaluation in two ways. First, it provides the evaluation criteria. Second, the specification exemplar is used as the try-on instrument to compare different tools.

The application of the specification exemplar and the evaluation frameworks leads to the guidelines which summarise the RE-tool evaluation approach (R-TEA) in Figure 6.1. As discussed in section 4.3 the evaluation process involves two actor groups: *evaluation team*, which manages the process, and potential *users*, who perform the evaluation activities. R-TEA includes six general guidelines:

1. *Preparation of a requirements specification*. The first phase consists of analysing the specification exemplar and adapting it to the user needs:

a) The evaluation team performs elicitation. The purpose is to highlight the most representative environmental needs. The starting elicitation point is the specification exemplar which consists of both evaluation frameworks. Elicitation should also highlight the level of organisational maturity, the work processes and product modelling paradigm. During the elicitation activities, the user should also specify the additional RE-tool requirements which are not described in the frameworks.

b) The purpose of the prioritisation activity is to determine the most important features and activities for the RE-tool. The evaluation team could apply different prioritisation algorithms (Analytical Hierarchy Process (AHP) (Saaty, 1980), weighted scoring method (WSM) (Kontio, 1996), Planning game).

c) Based on elicitation and prioritisation results, the evaluation team prepares the requirements specification, which contains the most needed user needs and RE-tool requirements, and where they are organised according to their priorities. The requirements specification for the RE-tool could contain the following RE-tool requirements:
   - the framework features which are selected without change from the frameworks and adapted as the RE-tool requirements.
   - the framework features but these features are changed according to the user needs to the RE-tool requirements.
   - new RE-tool requirements, not defined in the frameworks, but elicited from the user work practice.

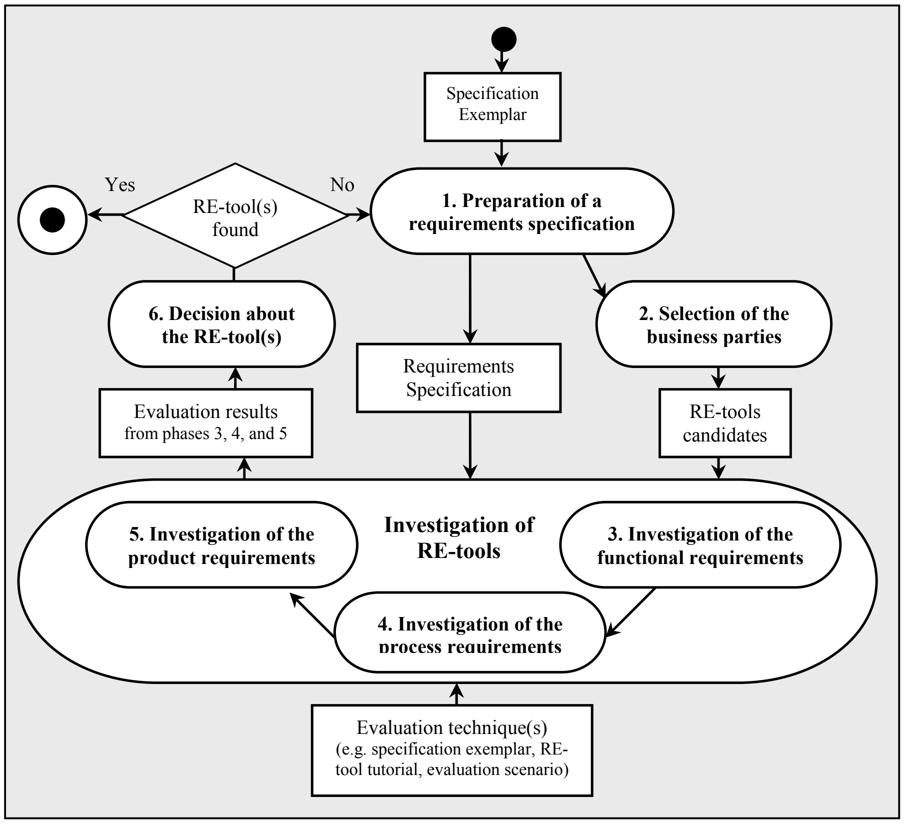

Specification Exemplar
Yes
No
RE-tool(s) found
1. Preparation of a requirements specification
6. Decision about the RE-tool(s)
2. Selection of the business parties
Requirements Specification
Evaluation results from phases 3, 4, and 5
RE-tools candidates
Investigation of RE-tools
5. Investigation of the product requirements
3. Investigation of the functional requirements
4. Investigation of the process requirements
Evaluation technique(s) (e.g. specification exemplar, RE-tool tutorial, evaluation scenario)

Figure 6.1 RE-tools Evaluation Approach

2. *Selection of business parties* involves the investigation of the RE-tool market according to the external requirements. The evaluation team requests trial and demonstration RE-tool versions from the business parties.

Next, phases 3, 4, and 5 describe the evaluation of each of the RE-tool, selected in the phase 2. All three following phases corresponds to the assessment of the tool compatibility as shown in Figure 4.3.

3. *Investigation of the functional requirements* contributes with the functionality evaluation of the RE-tool candidate. In order to test the functional RE-tool requirements an evaluation technique(s) is used. The *evaluation techniques* provide guidelines for getting familiar with the RE-tools selected for evaluation in phase 2. The evaluation techniques are RE-tool tutorials, evaluation scenarios and the requirements specification prepared according to the specification exemplar in phase 1 (Table 6.1). The combination of several evaluation techniques could help to get more

Table 6.1 Evaluation Techniques

Evaluation technique	Advantages	Disadvantages
Requirements specification(or specification exemplar)	– Is prepared in phase 1 according to specification exemplar (or specification exemplar itself); – Is prepared according to user familiar domain; – Emphasises the same evaluation issues.	– No guidance during the RE-tool assessment.
RE-tool tutorial	– Emphasises on teaching the RE-tool functionality; – Is received together with RE-tools; – Provides guidance during the RE-tool assessment.	– Tool evaluation could not be comparable if tutorials emphasise different tool functionality.
Evaluation scenario	– Emphasises on teaching the RE-tool functionality; – Emphasises the same functionality for all RE-tools; – Provides guidance during the RE-tool assessment; – Is prepared for all RE-tool candidates on the same domain; – Domain could be related to the user work practice.	– Preparation is time demanding; – Preparation is domain knowledge demanding.

detail tool evaluation, but the evaluation itself takes more time than using one evaluation technique. Requirements specification as evaluation technique is used in the example in section 6.2.3. Other two evaluation techniques are applied in case studies described in Chapter 7.

After testing all the RE-tool candidates, users fill in the evaluation forms. The evaluation forms where tool features are mapped with the requirements specification, are prepared for each tool. Evaluation form is some kind of questionnaire where users set the evaluation marks in a predefined scale. Evaluation form should also allow writing comments on any related evaluation issue.

When the users are performing the RE-tool testing activities, the evaluation team should perform observation activities. Observation has two purposes. First, the evaluation team could react to the difficulties and problems which arise during tool testing. Evaluation team could provide guidance and teaching remarks, too. Second, the observation could provide the useful information about RE-tool non-functional product characteristics, like RE-tool usability, reliability, and performance.

4. *Investigation of the process requirements*. Non-functional process requirements are analysed in correspondence to the relationships (Figure 5.4). The phase results with the lists of inadequacies between the user RE processes and the RE-tool support for the RE process. The inadequacies are considered in the phase 5 as the maintainability requirements.

5. *Investigation of the product requirements*. Non-functional product requirements are evaluated in the usability, performance and reliability tests. The observation, made in phase 3, also contributes with the evaluation of the non-functional product requirements. The tool maintainability requirements, selected in phase 4, should be investigated together with the business parties' support. The

evaluation team should investigate which maintainability requirements could be fulfilled by the tool users, and which should be redirected to the RE-tool vendors.

6. *Decision about the RE-tool selection* is made after summarising the results from phases 3, 4 and 5. Phase 3 contributes with the evaluation of the functionality, phase 4 addresses how the tool fits the working process and what is the need for maintenance, and phase 5 provides the evaluation of the non-functional product features. After the result analysis, one of three possible decisions should be made:

a) The users start using the "best-evaluated" RE-tool without changing their RE process.
b) The users start using the "best-evaluated" RE-tool, but they have to reconsider their RE process.
c) The "best-evaluated" evaluated tool is not suitable for the users and they need to repeat the RE-tool evaluation (reconsider the requirements specification, and/or search for other RE-tools candidates).

6.2.3 Demonstrative Example

A short demonstrative example of the R-TEA application includes the analysis of the specification exemplar for the RE-tool selection. Let's say, it is important that an RE-tool would maintain the object oriented and structural modelling perspectives.

FEF1.2. The RE-tool should specify requirements using semi- formal language(s).
Description: The feature and its activities define the way how an RE-tool should specify requirements model using semiformal languages.
Children: FEF1.2.1 and FEF1.2.2
Traces to: NF2.1
Priority:___________________

> **FEF1.2.1.** The RE-tool should provide tools for semiformal language description.
> **Description:** Semiformal languages are used to create requirements model. The examples of these languages are ER-diagrams, OMT, UML, DFD, and others. The exact languages should be specified using non-functional process requirements.
> **Parent:** FEF1.1
> **Priority:**___________________
>
> **FEF1.2.2.** The RE-tool should provide forward/ backward traceability between semiformal, and informal, formal descriptions.
> **Description:** While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceable relationships between different representation forms.
> **Parent:** FEF1.1
> **Priority:**___________________

Figure 6.2 Specification Example of the Functional Features

The evaluation process requirements include specification exemplar, two evaluation frameworks defined in Chapter 5, and the R-TEA method. The specification exemplar is also used as evaluation technique to test the RE-tool candidates. The first step is to prepare a requirements specification for the tool acquisition. It means analysing the frameworks and selecting the relevant framework features, activities and relationships (1a). Figures 6.2, 6.3, and 6.4 present a sample of specification exemplar. Figure 6.2 shows functional feature FEF1.2 and its activities. Figure 6.3 presents non-functional process feature NF1.2 and its activities. Figure 6.4 expands the relationship R1. Next step is the RE-tool requirements and relationship prioritisation (1b).

NF1.2. The RE-tool should support the selected modelling perspectives.
Description: Perspectives separate modelling language according to the core phenomena classes that are represented in the language.
Children: NF1.2.1, NF1.2.2, NF1.2.3, NF1.2.4, NF1.2.5, NF1.2.6, and NF1.2.7.
Traced from: FEF1.1
Priority:___________________

> **NF1.2.1.** The RE-tool should support structural modelling perspectives.
> **Description:** The modelling languages, which support this paradigm are entity-relationship (ER) diagrams, and reference modelling language (RML).
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.2.** The RE-tool should support functional modelling perspectives.
> **Description:** The modelling language with a functional perspective is data flow diagrams (DFD).
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.3.** The RE-tool should support rule-based modelling perspectives.
> **Description:** Example of the rule perspective language is goal-based approach to consider non-functional requirements and the external rule language (ERL).
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.4.** The RE-tool should support behavioural modelling perspectives.
> **Description:** There are two language-types commonly used: State transition diagrams (STD) and state transition matrices (STM), and Petri-Net.
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.5.** The RE-tool should support communicational modelling perspectives.
> **Description:** Persons cooperate within work processes through conversations and mutual commitments.
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.6.** The RE-tool should support object modelling perspectives.
> **Description:** Examples of the object perspective are the object modelling techniques OMT and UML
> **Parent:** NF1.2
> **Priority:**___________________
>
> **NF1.2.7.** The RE-tool should support actor and role modelling perspectives.
> **Description:** Examples of this paradigm are agent-oriented language (ALBERT) and (OORASS).
> **Parent:** NF1.2
> **Priority:**___________________

Figure 6.3 Specification Example of the Non-functional Features

R1.			
Functional requirement	**Non-functional requirements**	**Importance**	**Languages**
	NF1.2.1	________	
	NF1.2.2	________	
	NF1.2.3	________	
FEF1.2	**NF1.2.4**	________	
	NF1.2.5	________	
	NF1.2.6	________	
	NF1.2.7	________	

Figure 6.4 Specification Example of the Relationships between Features

In this example the simple importance priority scale of 0-to-10 is used, where 0 means unimportant and 10 means very important features. The outcome of the first phase is a requirements specification for the RE-tool evaluation (1c). Figure 6.5 shows that an object perspective is very important (priority set to 10) and UML should be maintained as modelling language. A structural perspective (and the ER diagrams) is less important to the user (priority 5). All other modelling perspectives are excluded from the further consideration. The RE-tool requirements priorities mean their importance to the RE-tool user. Later on priorities are used to calculate the overall evaluation score for the RE-tool.

The second phase involves the analysis of the RE-tool market and selecting the RE-tools for evaluation. For the purpose of this example two RE-tools are selected:

- RE-tool_A (Figure 6.6) is a prototype described in Chapter 8.
- RE-tool_B (Figure 6.7) is an RE-tool described in (Kaindl, 2004).

In order to evaluate the RE-tool, the requirements specification is adapted to the evaluation form (Table 6.2) where the user is able to provide the evaluation score to the tools features and the comments on any evaluation issue. The evaluation form is used in all the phases of the RE-tool investigation.

The third and fourth evaluation phases involve testing of the RE-tools. The requirements specification (Figure 6.5) used here as testing techniques, is entered to both tools. RE-tool_A provides functionality to specify requirements informally. RE-tool_B uses hypertext as a means to represent requirements semiformally[7] as discussed in (Snaprud and Kaindl, 1994). Both tools allow importing of files, which could maintain different information (including any requirements representations) about the requirements model. In addition RE-tool_B maintains export functionality of the requirements model to Rational Rose UML diagram.

The fifth phase is evaluation of the non-functional product requirements of the RE-tools. It is easy to notice that both RE-tool_A and RE-tool_B are quite complex to use for semiformal requirements model preparation, because users need some other tools for semi-formal requirements representation. Nothing could be said here about

[7] See discussion on informal, semiformal and formal representations in section 5.1.

tool reliability and efficiency of this functionality. The tool maintainability also remains questionable, because users should plug in the additional tools or they have to contact tool vendors for functionality extension which might be costly.

R1.			
Functional requirement	**Non-functional requirements**	**Importance**	**Languages**
FEF1.2	**NF1.2.1**	**5**	**ER diagrams**
	NF1.2.6	**10**	**UML**

<u>**FEF1.2.**</u> The RE-tool should specify requirements using semi- formal language(s).
Description: The feature and its activities define the way how an RE-tool should specify requirements model using semiformal languages.
Children: FEF1.2.1 and FEF1.2.2
Traces to: NF2.1
Priority: 6

<u>**FEF1.2.1.**</u> The RE-tool should provide tools for semiformal language description.
Description: Semiformal languages are used to create requirements model. The examples of these languages are ER-diagrams, OMT, UML, DFD, and others. The exact languages should be specified using non-functional process requirements.
Parent: FEF1.1
Priority: 7

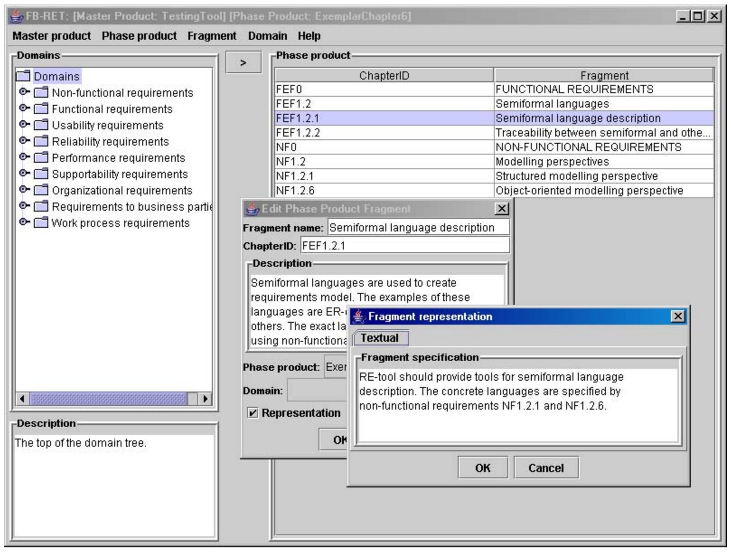
<u>**FEF1.2.2.**</u> The RE-tool should provide forward/ backward traceability between semiformal, and informal, formal descriptions.
Description: While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceable relationships between different representation forms.
Parent: FEF1.1
Priority: 5

<u>**NF1.2.**</u> The RE-tool should support the selected modelling perspectives.
Description: Perspectives separate modelling language according to the core phenomena classes that are represented in the language.
Children: NF1.2.1 and NF1.2.6
Traced from: FEF1.2
Priority: 8

<u>**NF1.2.1.**</u> The RE-tool should support <u>structural</u> modelling perspectives.
Description: The modelling languages, which support this paradigm are entity-relationship (ER) diagrams, and reference modelling language (RML).
Parent: NF1.2
Priority: 5

<u>**NF1.2.6.**</u> The RE-tool should support <u>object</u> modelling perspectives.
Description: Examples of the object perspective are the object modelling techniques OMT and UML
Parent: NF1.2
Priority: 10

Figure 6.5 Requirements Specification constructed from the Example


FB-RET: [Master Product: TestingTool] [Phase Product: ExemplarChapter6]

Master product Phase product Fragment Domain Help

Domains
- Domains
  - Non-functional requirements
  - Functional requirements
  - Usability requirements
  - Reliability requirements
  - Performance requirements
  - Supportability requirements
  - Organizational requirements
  - Requirements to business partie
  - Work process requirements

Description
The top of the domain tree.

Phase product

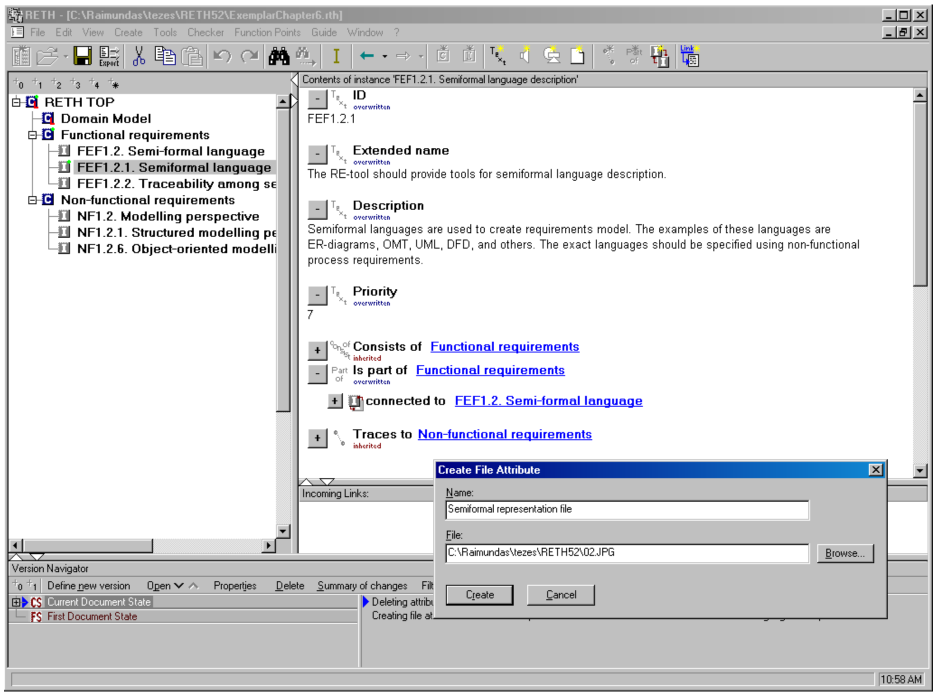
ChapterID	Fragment
FEF0	FUNCTIONAL REQUIREMENTS
FEF1.2	Semiformal languages
FEF1.2.1	Semiformal language description
FEF1.2.2	Traceability between semiformal and othe...
NF0	NON-FUNCTIONAL REQUIREMENTS
NF1.2	Modelling perspectives
NF1.2.1	Structured modelling perspective
NF1.2.6	Object-oriented modelling perspective

Edit Phase Product Fragment
Fragment name: Semiformal language description
ChapterID: FEF1.2.1
Description
Semiformal languages are used to create requirements model. The examples of these languages are ER-… others. The exact la… using non-functiona…
Phase product: Exer…
Domain:
☑ Representation

Fragment representation
Textual
Fragment specification
RE-tool should provide tools for semiformal language description. The concrete languages are specified by non-functional requirements NF1.2.1 and NF1.2.6.

OK Cancel

Figure 6.6 R-TEA Example (RE-tool_A)


RETH - [C:\Raimundas\tezes\RETH52\ExemplarChapter6.rth]

File Edit View Create Tools Checker Function Points Guide Window ?

- RETH TOP
  - Domain Model
  - Functional requirements
    - FEF1.2. Semi-formal language
    - FEF1.2.1. Semiformal language
    - FEF1.2.2. Traceability among se
  - Non-functional requirements
    - NF1.2. Modelling perspective
    - NF1.2.1. Structured modelling pe
    - NF1.2.6. Object-oriented modelli

Contents of instance 'FEF1.2.1. Semiformal language description'

ID
FEF1.2.1

Extended name
The RE-tool should provide tools for semiformal language description.

Description
Semiformal languages are used to create requirements model. The examples of these languages are ER-diagrams, OMT, UML, DFD, and others. The exact languages should be specified using non-functional process requirements.

Priority
7

Consists of Functional requirements
Is part of Functional requirements
connected to FEF1.2. Semi-formal language
Traces to Non-functional requirements

Incoming Links:

Create File Attribute
Name: Semiformal representation file
File: C:\Raimundas\tezes\RETH52\02.JPG
Browse...
Create Cancel

Version Navigator
Define new version Open Properties Delete Summary of changes
CS Current Document State
FS First Document State
Deleting attrib…
Creating file at…

10:58 AM

Figure 6.7 R-TEA Example (RE-tool_B)

Table 6.2 Example of Specification Exemplar (Evaluation results)

Require-ments	Prio-rity	RE-tool_A		RE-tools_B	
		Evaluation	Evaluation comments	Evaluation	Evaluation comments
Functional requirements					
FEF1.2	**6**	2	No functionality to create semi-formal representation. No maintenance of traceability, but the information of the same requirement is stored as one element.	3	Functionality to create semi-formal representation using hypertext. Tool maintains association and aggregation relationships, but they are defined only between informal representations.
FEF1.2.1	**7**	2		4	
FEF1.2.2	**5**	1		3	
Process requirements					
NF1.2	**8**	2	Imported files could maintain any requirements representation using any modelling perspective.	3	Requirements can be exported to Rational Rose UML format. No support for structural perspective.
NF1.2.1	**5**	2		1	
NF1.2.6	**10**	2		5	
Overall:		**77**		**170**	

The sixth phase involves analysis of evaluation results from phases 3, 4 and 5. Table 6.2 shows results of tool evaluation. The tool evaluation is very subjective and it is based only on the user opinion. The overall evaluation is calculated as sum of multiplications between feature priority and tool feature evaluation. The result analysis should also take in account notices from non-functional product requirements evaluation. In this example, most probably, none of RE-tool_A and RE-tool_B would be acquired, and the user would have to reconsider the requirements specification (limit to the informal requirements representation), or to start with investigating the business parties and selecting new RE-tools for evaluation.

6.3 Comparison with Other Tool Evaluation Approaches

Six general tool evaluation approaches – PORE (Maiden and Ncube, 1998), OTSO (Kontio, 1996), CAP (Ochs *et al*., 2000), Scenario-based selection (Feblowitz and Greenspan, 1998), STACE (Kunda and Brooks, 1999; Kunda, 2003), and quality-based selection (Franch and Carvallo, 2002) – are analysed in section 4.3. This section makes an analytical comparison of the R-TEA method and the general evaluation approaches. The analysis highlights the most important similarities and differences between R-TEA and other evaluation approaches and summarises them in Table 6.3.

6.3.1 PORE vs. R-TEA

Both PORE (Maiden and Ncube, 1998) and R-TEA uses quantitative scores and qualitative comments to prioritise tool requirements and evaluate tool compatibility. R-TEA and PORE emphasise maintenance of evaluation rationale and involve the potential tool users in the assessment. In PORE users are 'spectators' and requirements providers. In R-TEA users are tool evaluators at the same time they are taught how to use the tool and its functions. PORE suggests involvement of the tool vendors in evaluation activities. However vendors also provide marketing and tool promotion information, and their involvement could be expensive. R-TEA suggests involvement of domain experts. The experts could not only provide the valuable information of the RE-tool application, but also manage the evaluation activities (as an evaluation team).

PORE is a one round evaluation approach uses an incremental user requirements expansion at the same time reduces number of the tool candidates. PORE is based on the templates which describe evaluation steps and material required and constructs evaluation criteria hierarchy from user requirements. The R-TEA method uses already defined evaluation frameworks for functional and non-functional RE-tool requirements.

6.3.2 OTSO vs. R-TEA

OTSO (Kontio, 1996) consists of six phases: search, screening, evaluation, analysis, deployment and assessment. OTSO assumes that requirements specification for the tool acquisition is prepared in advance. The OTSO search and screening correspond to the second R-TEAS phase.

The OTSO evaluation corresponds to the third, fourth and fifth phases of R-TEA. The major difference is that OTSO does not support any particular tool domain. Therefore in the OTSO evaluation criteria should be constructed according to the goal, question and metric measurement plan (Basili, 1993). Finally the OTSO analysis and deployment correspond to the sixth phase of R-TEA. The OTSO assessment emphasises reuse and improvement of the next OTSO process. This activity is not directly supported in R-TEA.

6.3.3 CAP vs. R-TEA

CAP (Ochs *et al.*, 2000) focus on criteria construction according to predefined in advance requirements specification and the ISO/IEC 9126 standard (1991). In R-TEA the requirements specification is prepared according to specification exemplar, so the main approach focus is on assessment of the RE-tool compatibility. CAP constructs a four-level evaluation criteria hierarchy and develops a GQM (Basili, 1993) measurement plan. R-TEA uses two evaluation frameworks which describes the RE-tool features. CAP provides detail guidelines and defines steps to be performed, their input and output products. R-TEA provides general guidelines and the prototype tool in order to facilitate evaluation process.

6.3.4 Scenario-based Selection vs. R-TEA

The scenario-based selection (Feblowitz and Greenspan, 1998) analyses how selected tool impacts the organisation's work processes, how the work in the organisation gets done, and how services of the organisation are delivered to its customer. The fourth phase of the R-TEA method describes evaluation of the process requirements which characterise how the organisation is working in order to produce new software product with the means of the acquired RE-tool(s). Scenario-based selection does not consider the tool functionality, product requirements, and requirements for business parties. The R-TEA method considers how to relate the process requirements and other types of requirements. The way organisation is (and will be) working, is directly related both to the tool functionality, tool maintainability (product requirement), and vendor capabilities to support and maintain the RE-tool.

6.3.5 STACE vs. R-TEA

STACE (Kunda and Brooks, 1999; Kunda, 2003) comprises four interrelated processes: requirements definition, social-technical criteria definition, alternative identification, and evaluation and assessment. The STACE requirements definition resembles the preparation of the requirements specification in R-TEA. In the STACE social-technical criteria definition, the evaluation criteria are constructed for the particular tool domain. In R-TEA the assessment criteria are provided by two evaluation frameworks which include tool functionality, quality characteristics (like the product requirements) and social-economic factors (like the organisational requirements and the requirements for business parties). In STACE the alternative identification corresponds to the second phase of the R-TEA method. The STACE evaluation or assessment involves investigation of the tool compatibility to the organisational environment and considers tool candidates. R-TEA has three phases (third, fourth, and fifth) which describes the RE-tool assessment. In the sixth phase R-TEA summarises results and makes decision about the RE-tool selection.

As general tool selection approach, STACE is time consuming while defining evaluation criteria for particular tool domain. R-TEA places its focus on the evaluation of the RE-tool candidates and their acquisition to organisational environment.

6.3.6 Quality-based Selection vs. R-TEA

Quality-based selection (Franch and Carvallo, 2002) constructs an evaluation criteria (quality) model according to ISO/IEC 9126 standard. The construction consists of seven steps which describe model subcharacteristics, metrics and relationships between them. The definition of the relationships in quality-based selection approach is similar as it is done between two evaluation frameworks in R-TEA. A quality model has to be constructed for each new tool domain when quality-based selection is applied. Although it might be applicable for the evaluation of the same domain tools, the model still needs to be reconsidered according to the environmental needs. R-TEA starts each time with the specification exemplar which transformed to the requirements specification for the RE-tool acquisition according to the environment needs. The quality-based selection is described by the COSTUME method (Carvallo et al. 2004a) which describes how quality models are constructed for the organisational actors and selected tool candidates, and how both models are mapped. The R-TEA method describes the RE-tool assessment according to two evaluation frameworks.

Construction of the quality models is supported by the prototype tool DesCOTS (Grau et. al, 2004; Carvallo et. al, 2004b). The prototype tool which maintains the functional framework RE-tools and the R-TEA method is presented in Chapter 8.

6.3.7 Comparison Summary

R-TEA may support tool selection in general (Figure 4.3), but it is designed principally for the RE-tool domain. All the evaluation approaches (Kontio, 1996; Maiden and Ncube, 1998; Feblowitz and Greenspan, 1998; Ochs *et al.*, 2000; Franch and Carvallo, 2002; Kunda, 2003) suggest only guidelines for the tool evaluation, but they do not define particular metrics or measures for tool evaluation. Therefore their application is time consuming and domain knowledge demanding. Some of the approaches provide ready-to-use techniques, like templates in PORE (Maiden and Ncube, 1998), or ISO/IEC 9126 standard (1991); however the models should be adapted to environment each time. The R-TEA method uses already constructed RE-tool requirements frameworks to initiate and guide the evaluation process.

None of the tool evaluation methods characterises the tool functionality and user the requirements mapping process. Scenario-based selection (Feblowitz and Greenspan, 1998) introduces the mapping of two working scenarios (scenario without the tool *vs.* scenario with a tool); however this mapping is executed before starting to compare the tools. R-TEA emphasises to use the specification exemplar for tool testing. In addition it suggests two other evaluation techniques, like specification exemplar, RE-tool tutorials and evaluation scenarios.

The R-TEA method selects RE-tools after the market investigation, using the tool costs (Kontio, 1996; Sedigh-Ali and Ghafoor, 2001), and vendor reliability (Kontio, 1996; Kunda, 2003) metrics. Both evaluation frameworks present a wide range of functional and non-functional RE-tool features, like tool functionality (Haywood and Dart, 1997; Lang and Duggan, 2001; INCOSE, 2002; Hoffmann *et al.*, 2004), non-functional product requirements (CAP (Ochs *et al.*, 2000), quality-based selection (Franch and Carvallo, 2002)), external-social requirements (STACE (Kunda, 2003)). In addition the R-TEA method characterises the non-functional process requirements. This is possible because the R-TEA is constructed on the particular domain (e.g. RE-tool selection).

R-TEA supports a constructivistic world-view approach. Both users and evaluation team externalise their knowledge and the RE-tool requirements throughout the evaluation process. The outcome of the externalised knowledge is the requirements specification and the selection of the RE-tool(s) for usage.

R-TEA provides some suggestions on how to assess the RE process and to adapt it to the RE-tool. However, the organisation should be mature and be ready changes. Therefore the organisation has to determine its maturity level before starting the RE-tool assessment.

An RE-tool will not be the only tool used in the organisation. Therefore, it is important to consider the information interoperability between the RE-tool(s) and other tools. R-TEA maintains the RE-tool requirements for assessment relationships between the RE-tool and other tools used in the organisation.

In the R-TEA method two groups of participants are defined. They are evaluation team which manages the evaluation; and potential RE-tool users who perform evaluation and who will use the RE-tool in their practice after that. R-TEA tends to provide teaching of the tool functionality by the means of different evaluation techniques.

R-TEA does not analyse the information reuse activities. However the environment and knowledge collected in the requirements specification and evaluation results, could be easily reused for further purposes: e.g., for second round evaluation of the RE-tools or for evaluation of other type of tools. The R-TEA method is applied in a number of case studies which are analysed in Chapter 7.

Table 6.3 show comparison of the evaluation approaches according to activity focus (activities correspond to ones in Figure 4.3), material required to initiate evaluation approach, detail of provided guidelines, evaluation criteria definition, tool support and maturity which is identified by report year and application of an evaluation approach. Next the tables present some information about the case studies, where the validity of the evaluation approach is investigated.

Table 6.3 Comparison of Evaluation Approaches and R-TEA

		PORE	**OTSO**	**CAP**	**Scenario-based**	**STACE**	**Quality-based**	**R-TEA**
Activity focus		Requirements specification, Understanding the tools	Requirements (criteria) specification, Assessment the compatibility	Requirements (criteria) specification	Requirements specification	Requirements (criteria) specification	Requirements (criteria) specification	Assessment of the compatibility
Material required		Test cases, Vendor demonstration	Templates for criteria construction, Initial criteria taxonomy.	Requirements specification (defined in advance), phase outputs	Stakeholder work scenarios (received by observation or stakeholder stories)	Consultation with stakeholders, system documents, domain knowledge, and market studies.	Literature review, tool web-page analysis.	Specification exemplar, RE-tool tutorials, evaluation scenarios, two evaluation frameworks.
How detail guidelines are?		Detail templates guide the process.	6 broad phases (search, screen, evaluate, analyse, deploy, assess)	Detail guidelines, describing inputs, steps, and outputs	General process guidelines. Detail guidelines how to write scenario.	Guidelines include 5 general phases	COSTUME method. 7 guidelines for construction quality models.	Guidelines include 6 general guidelines.
Evaluation criteria		Not specifically defined, depends on the domain and template guidance.	General evaluation criteria, goal/question/ metric measurement plan.	General evaluation taxonomy, based on ISO/IEC 9126 standard (1991), goal/question/ metric measurement plan.	No evaluation criteria. Focus on tool functionality identification by writing scenarios.	General categories, adapted to a particular domain	Quality model, based on ISO/IEC 9126 standard (1991), extended according to domain	Two evaluation frameworks, one for functional and one for non-functional requirements.
Tool support		SCARLET-Advisor (Maiden, et al., 2003)	No	No	No	No	DesCOTS (Grau *et. al*, 2004; Carvallo *et. al*, 2004b)	Prototype tool (in Chapter 8)
Maturity		Reported 1998, case study (RE-tools), extensions in BANKSEC project and SCARLET approach.	Reported 1995, case studies (map application, hypertext browsers).	Reported 2000, case study (tools for administrative task)	Reported 1998, teaching seminar	Reported 1999, case study (Geographical Information Systems)	Reported 2002, case studies (mail servers, RE-tools)	Reported 2004, academic case studies without purpose to select a tool (RE-tools)
Reported case study	**Tool domain**	RE-tools	Hypertext browsers	Tools for administrative task	NN	Geographical Information Systems	RE-tools	RE-tools
	Tool number	Initially 30, narrowed to 6	4	3	NN	4	5	3
	Time	11 weeks	144 hours	5 months	NN	NN	NN	4-6 hours
	Team size	6-10 potential tool users and evaluation team of 4 persons.	2 evaluators for each tool	NN	NN	NN	NN	2-4 tool user groups and evaluation team of 1 person

NN – not known

6.4 Chapter Summary

In this chapter introduces a specification exemplar which is constructed of two evaluation frameworks is described. The specification exemplar is an evaluation technique which initiates the RE-tool evaluation approach (R-TEA).

The R-TEA method consists of six phases. First, the requirements specification for the RE-tool acquisition is prepared according the specification exemplar where potential tool users select and prioritise the RE-tool requirements. The second phase involves the investigation of the RE-tool market according to the tool surveys and information received from tool vendors Web-pages. Next step is RE-tool assessment according to the functional (phase 3), non-functional process (phase 4) and product (phase 5) requirements. The evaluation team and the potential tool users should determine the tool compatibility and possibilities to maintain and support the RE-tool in the organisation, or to receive professional help from tool vendors (or consultants). Finally, the sixth phase comprises the decision about the RE-tool suitability and selection. In this phase the organisation has to decide either to select the best evaluated RE-tool(s) without or with organisational work process changes or to define new evaluation settings (reconsider the requirements specification and the list of RE-tool candidates) and to repeat the evaluation. The R-TEA method is compared with the existing general tool evaluation approaches such as OTSO (Kontio, 1996), PORE (Maiden and Ncube, 1998), scenario-based selection (Feblowitz and Greenspan, 1998), CAP (Ochs *et al*., 2000), quality-based selection (Franch and Carvallo, 2002) and STACE (Kunda, 2003).

In the next part of this work the R-TEA method is experimentally tested in several case studies in Chapter 7, and implemented into a tool prototype in Chapter 8.

PART III

EVALUATION AND VALIDATION

In the third part of this work the validity of the analytical approach is investigated. The part consists of two chapters which comprise empirical investigation and design-based analysis of the frameworks and the R-TEA method.

In Chapter 7 *Case Studies* four empirical case studies are considered. Case study A (Matulevičius and Strašunskas, 2003; Matulevičius, 2004b) targets the importance of the functional framework features and analyses the agreement about the framework activities. This case study contributes to the validation of the evaluation framework for the functional RE-tool requirements. It also presents how the framework could be adapted to the environment, and identifies the environmental needs, which emerge from the requirements specification maintenance process.

The origin of the evaluation framework for the functional RE-tool requirements allows its application for the RE process assessment (Matulevičius, 2004a; Matulevičius, 2004c). Therefore, Case study B (Matulevičius, 2004c) compares two RE processes. In order to prepare a requirements specification, standard office and modelling tools are used in the first case. In the second case, the requirements specification is executed by the means of the RE-tools. Finally, the quality of both requirements specifications is compared and evaluated by their qualitative properties. The results indicate that the targeted RE-tools provide better support for the RE process than the standard office and modelling facilities. The requirements specification prepared using the targeted RE-tools, is substantially of better quality. The work findings suggest the RE-tool features which could be improved for the qualitative automated support of the RE process.

Case study C (Matulevičius, Karlsson, and Sindre, 2004) considers on a two-session experiment which compares evaluation techniques for RE-tools. First a theoretical framework and test scenarios are used, next tool tutorials and a questionnaire are applied. The findings show that the framework provides a detailed evaluation, but it is more time consuming than the questionnaire-based technique. The results also support the use of test scenarios instead of tool tutorials. Although the scenario preparation is relatively long and requires expert involvement, the scenarios provide comparison of the same tool functions in the same problem domain.

Finally Case study D reports on a student experiment trying to validate a previously proposed framework for evaluating RE-tools. In this experiment, the participants both used several alternative tools for making a requirements specification, and then evaluated the tools by means of the framework. This enables to look at both the participants' performance with the various tools and evaluation approaches, and their perceptions about the same tools. The case study findings indicate advantages of using evaluation frameworks for RE-tool selection. Evaluation scenarios contribute with better satisfaction of the RE-tool usage, but RE-tool tutorials provide better means to learn the RE-tool functionality. The experiment results indicate that RE-tools provide better support than office tools, leading to higher quality specifications which are easier to maintain.

Chapter 8 *Prototypes* presents two prototype tools. The first prototype facilitates the evaluation of the RE-tools. It is implemented according to the evaluation framework for the RE-tools and supports the R-TEA method. The prototype was applied in Case study D. The main architectural components are surveyed, and some improvement possibilities are described in the chapter.

The second prototype introduces an environment for investigating the RE-tool weaknesses in a design-implementation environment. The FB-RET (Fragment-based RE-tool) prototype presents a kernel of the RE-tool and suggests wide possibilities to implement new methods in representing, tracing, communicating, prioritising, reporting, storing and reusing the individual requirements and the requirement model.

Chapter 7

Case Studies

In the second part of this work two evaluation frameworks and an RE-tool evaluation approach (R-TEA) were presented. In this section the validity of the frameworks and the R-TEA method is empirically tested in four case studies. All four case studies are executed in the academic environment. The first two case studies are performed while developing information system MEIS (model evaluation of the information system) which is applied in the introductory course at the university. The last two case studies involve the investigation of different artefacts of the R-TEA method, such as evaluation techniques, frameworks, R-TEA guidelines and a prototype tool used to guide and facilitate assessment of the RE-tools. The findings of the case studies contribute to the validity, performance and usability of the frameworks and the R-TEA method.

7.1 Case Study Objectives

Four case studies presented in this chapter, are executed in an academic environment at the Norwegian University of Science and Technology (NTNU). The third case study was also partly performed at the Lund University in Sweden. The main purpose of all the cases is to investigate usability, performance and reliability of the evaluation frameworks and the RE-tool evaluation approach (R-TEA) defined in the second part of this work. The case studies all address similar issues using different environment settings, subjects and techniques. For instance:

Importance and *agreement* about the framework features and activities is investigated in Case studies A, C and D. All three case studies involved different research subjects who had different experience (researchers in Case study A, and graduate students from NTNU and LU in Case studies C and D), motivation (payment in Case study C and course exercise in Case study D), treatment (presentation of the framework in Case studies A and C, and self study in Case study D) and instruments (paper-based analysis in Case studies A and C, framework prototype in Case study D).

The case study findings are compared in section 7.6, and presented in Table 7.20. Raw material and techniques of the case studies are also provided in the appendixes of this work.

Weaknesses of the RE-tools to support the RE process, are observed through all the studies. In Case studies A and B it was performed a subjective observation of the RE-tool functionality. In Case studies C and D the rate of not tested RE-tool features is calculated. The comparison of not-tested rate is provided in Table 7.23.

The *evaluation techniques* – evaluation scenarios and RE-tool tutorials – are investigated in Case studies C and D. The difference of this investigation is in the research method. In Case study C the observation was performed to get familiar with the evaluation technique performance. In Case study D the subjects had the questionnaire[8] where they provided their perceive opinion about the evaluation techniques[9].

Similarly, Case studies B and D investigated the *RE-tools* against general *office tools* support to the RE process. In Case study B the observation is carried on to investigate the RE process support and maintenance of the requirements specification. In Case study D the subjects were asked to use a questionnaire and to provide their opinion about the tools they were using.

In Table 7.1 several aspects on how the case studies addressed the R-TEA validity, are described. In Case study B two RE-tools are selected according to the results in the tool evaluation in case A and according to several non-functional RE-tool requirements such as tool availability, accessibility and training, usage costs and work process requirements (tool functionality to maintain the semiformal requirements representation).

Although the case studies, mostly, investigate the performance and usability of the evaluation framework for the functional RE-tool requirements, they contributed with validation for the requirements specification preparation for the RE-tool acquisition in the phase 1, RE-tool candidate selection in phase 2, and evaluation of their functional RE-tool requirements in phase 3. The case studies have no purpose of selecting the RE-tools for purchase purpose, so the product and process requirements of the RE-tools were not investigates in detail according to the framework of the non-functional RE-tool requirements. However several questionnaires consider the RE-tool usability, adaptability, performance, maintenance and ease of use.

In the following sections all the case studies are described in detail.

[8] See Appendix I
[9] See Appendix J

Table 7.1 How R-TEA Method Addressed in Case Studies

R-TEA	Cases studies
1. Preparation of a requirements specification	Case studies A, C, and D. The participants were analysing the framework features and prioritising according to their importance.
2. Selection of the business parties	Case studies A, C, and D. The RE-tool candidates were selected after the market studies and the tool availability at the universities.
3. Investigation of the functional requirements	Case studies A, B, C, D. In all the case studies the evaluation framework for the functional RE-tool requirements was applied. In addition, a number of the questionnaires was prepared in order to investigate the RE-tool usability, performance and reliability. Different evaluation techniques (e.g., evaluation scenarios and RE-tool tutorials) were applied and their performance tested in
4. Investigation of the process requirements	
5. Investigation of the product requirements	
6. Decision about the RE-tool(s)	Partially Case study A, and the RE-tools used in Case study B. Case studies C and D did not have the purpose of the tool selection, but they investigated performance, usability, and reliability of techniques used in the R-TEA method.

7.2 Case Study A: Framework Feature Importance

The first case study (Matulevičius and Strašunskas, 2003; Matulevičius, 2004b) describes how an evaluation framework for functional RE-tool requirements could be applied in an environment. The process consists of two phases (Figure 7.1): the preparation phase and the execution phase. The *preparation phase* consists of problem and environment descriptions, discovery of the environmental needs and evaluation of RE-tools candidates. The goal of *problem description* is to characterise the task for which the framework validation process is applied. *Environment description* discusses the organisational settings. The overall goal of the preparation phase is to find the appropriate RE-tool for maintenance of the requirements specification for the information system, used for teaching purposes. The evaluation framework for functional RE-tool requirements is applied in order to discover the most important aspects of the environment. Then the RE-tool candidates are evaluated.

The goal of the *execution phase* is to prepare and maintain the requirements specification for an information system (MEIS – Model Evaluation of the Information Systems) used during the teaching course taken by the third year students in Norwegian University of Science and Technology (NTNU). In this case study the evaluation process of the RE-tools showed the list of RE-tools weaknesses. The editing, drawing, modelling and communication tools, but not targeted RE-tool, were used for *maintenance of the requirements specification*. The observation of the requirements specification maintenance shows requirements for the automated RE process support, which are missing while using not the targeted RE-tools.

7.2.1 Problem Description

The purpose of the case study was to prepare the requirements specification for an information system MEIS. Figure 7.2 represents the static description of the system, and Figure 7.3 gives the representation of dynamic relationships between problem elements.

The system registers two types of users: students and student assistants. Students submit their solutions to the system. Student assistants evaluate the solutions and form reviewer groups. The reviewer groups consist of 3 to 5 students, whose solutions are accepted by student assistants. The following step is the review process. The reviews are done according to the semiotic quality framework by Lindland *et al* (1994). If the review results are essentially different, the student assistant rejects the review, and the reviewer group should evaluate the work again. Otherwise, the review is accepted and the results are sent to the author– the student who delivered the solution. Requirements specification included the following requirements:

– students should upload the solution to the system;
– student assistants should accept or reject the solution;
– if the solution is accepted, student assistants should form the reviewer groups;
– reviewers should evaluate solutions;
– students and student assistants should print reports about the solution evaluation.

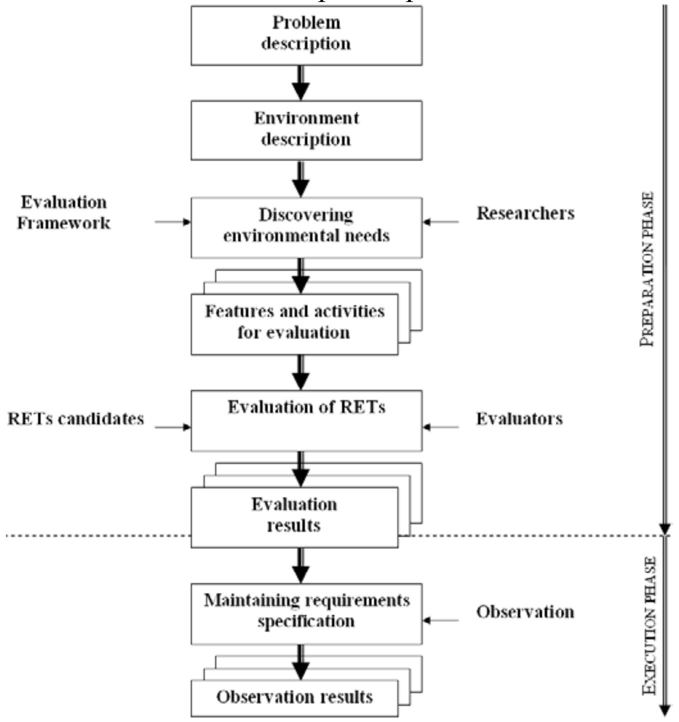


Figure 7.1 Case Study *A* Design

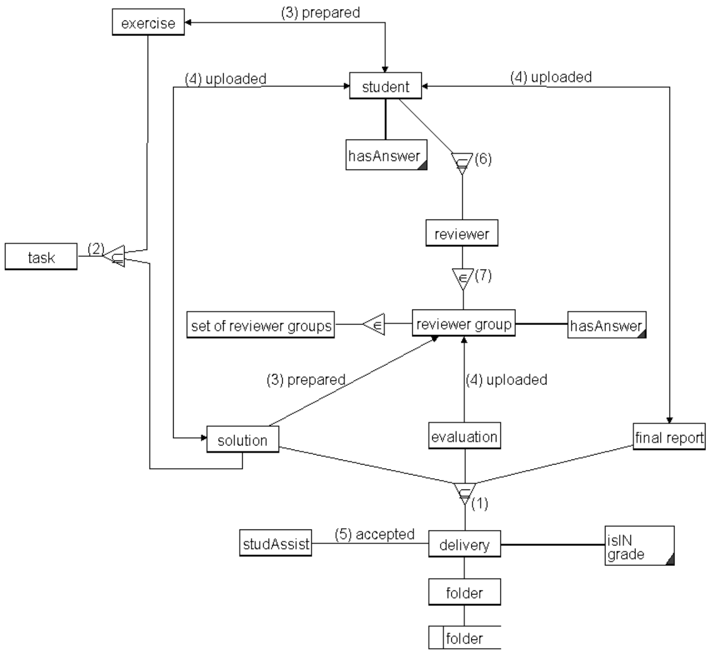


Figure 7.2 Representation of the Problem. Static View

The MEIS system is used for the educational training course, in which 216 graduate students of Norwegian University of Science and Technology participated. The case also included 6 undergraduate students who were student assistants.

7.2.2 Environment Description

The requirements specification for the system is developed in a university. Different stakeholders participate in the process. They perform different roles at various system development stages. The RE process included the following stakeholder groups: *organising actors* like teachers and supervisors of the course; *leading actors* like teaching assistants responsible for coordinating and maintaining different system development phases; *developing actors* like undergraduate students responsible for the system developing and testing. At the different project development phases the actors performed different roles, for example, the leading actors took the role of engineer in the early phase and project manager in the late phase; the developing actors were designing, implementing and testing the system.

The major research interest and experience of the participants includes information systems and processes, workflow analysis and the Semantic Web, information management, maintenance and utilisation of information resources, implementation of decision support systems and intelligent agents.

STUD is a set of students.
DEL is a set of deliveries.
SOL is a set of solutions.
EVAL is a set of evaluations.
FR is a set of final reports.
TASK is a set of tasks.
EXER is a set of exercises.
FLD – is a folder, where solutions are stored.
STASS is a set of student assistants.
REV is a set of reviewers, subset of *STUD*.
GR is a reviewer group consisting of 3 to 5 reviewers.
REVGR is a set of reviewer groups.

$$SOL \cup EVAL \cup FR \subseteq DEL \tag{1}$$

$$EXER \cup SOL \subseteq TASK \tag{2}$$

isIN(z,y) is an attribute relationship which means that delivery z is kept in folder y. The result set is {TRUE, FALSE}. *hasAnswer(z,y)* is an attribute relationship, which means, that z has answer y. Result set is {TRUE, FALSE}. *Grade* is an attribute for exercise acceptance/rejection. It could get values: {GOOD, BAD}

$$prepared(x,y) \equiv \{\forall x \exists z \mid x \in TASK, z \in STUD\} \Rightarrow \{\exists y \mid y \in DEL, hasAnswer(z,y)\} \tag{3}$$

According to task *x*, delivery *y* is prepared - for each task at least one student exists, who has an answer (as delivery - solution or evaluation) to the task.

$$uploaded(z,w) \equiv \{\exists yz \mid y \in EXER, z \in SOL \vee z \in FR, prepared(y,z)\} \Rightarrow \{\forall z \exists w \mid w \in FLD, isIN(z,w)\} \tag{4}$$

Solution *z* is uploaded to folder *w*, - exercise *y* exists, and a solution or a final report for it exists, and a solution and a final report must be prepared according to the exercise), that exists in folder *w*, where a solution or a final report is stored.

$$accepted(x,y) \equiv \{\forall y \exists x \mid y \in DEL, x \in STASS \text{ iff } grade = "GOOD"\} \tag{5}$$

For each delivery exists student assistant, who accepts delivery if it is "good".

$$REV \subseteq STUD \cap \{\forall x \mid x \in REV \text{ iff } \{\exists y,z,w \mid uploaded(y,z), accepted(y,w)\}\} \tag{6}$$

Each element *x* of subset *REV,* if and only if exercise *y* is uploaded to folder *y* and exercise *y* is accepted by student assistant *w*.
GR consists of 3 to 5 reviewers

$$GR \equiv \{\forall x \mid x \in REV, 3 \le x \le 5\} \tag{7}$$

Figure 7.3 Representation of the Problem. Dynamic Relationships

7.2.3 Discovering Environmental Needs

In order to discover environmental characteristics, a quantitative analysis, which included a survey, was carried out. Thirteen researchers were invited to participate in a discussion. Next, the quantitative analysis included a questionnaire[10], containing questions about the importance of the evaluation framework features and activities. Before starting the questioning process the motivating discussion, which provided definitions of the problem domain and the problem itself, was held. The participants were explained the needs and purposes of the quantitative analysis. The respondents were asked to evaluate the general RE-tool features.

The *importance* of evaluation framework features and activities were evaluated in the scale of 0 to 10 (*0* a feature or an activity is not important at all; *10* a very

[10] See Appendix C.

important and useful feature or an activity). The overall importance[11] for each feature and activity is calculated as a mean measure:

$$M_j = \frac{1}{n}\sum_{i=1}^{n} e_{ij}$$

here *i* is a participant index, *j* is a feature index, e_{ij} is an evaluation of *i* participant for *j* feature, *n* – a number of participants. The *agreement* about a single feature or activity is evaluated as the variance measure:

$$V_j = \frac{1}{n-1}\sum_{i=1}^{n}(e_{ij} - M_j)^2$$

If the variance is relatively low, it means strong agreement about a single feature or activity. And if the variance is relatively large, it indicates disagreement between participants on a feature or an activity. In this case study the agreement ranges from 1,4 to 8,8. In order to determine the agreed features and activities the *threshold* (*t*=5) for agreement (variance) is defined. The threshold removes features and activities about which the respondents do not agree (Table 7.2). Participants were also asked to suggest the features and activities to the evaluation framework, which they feel are needed to evaluate and which are useful for a RE-tool. These features and activities are shown in Table 7.3.

7.2.4 Evaluation of RE-tools

Investigation of the RE-tools candidates was done by two evaluators. After individual evaluations, a discussion was held in order to achieve the common agreement for the final scores of RE-tools functionality features. The survey of the evaluation is presented by Matulevičius and Strašunskas (2003), where RE-tools features and activities are evaluated as: High=3 (very good), Medium=2 (average), Low=1 (poor).

The list of RE-tools candidates for evaluation was selected from commercial RE-tools and this includes: *CaliberRM Web v.4.0.*, *Core 3.1 Trial*, *Cradle-4*, *DOORS 5.2.*, *RequisitePro*, *RDT Version 3.0*, *Vital Link*, *XTie-RT 3.1.* Trial, demonstration and evaluation versions were evaluated according to manuals and documentation. RE-tools were also tried out on small examples.

The overall evaluation E_j of the RE-tool *j* may be calculated as the sum of multiplications between two values: feature evaluation C_{ij} (where *i* is a number of the feature), decided during evaluation of the tool, and feature importance M_i, discovered during the quantitative analysis:

$$E_j = \sum_i C_{ij} M_i$$

The overall evaluation of the RE-tools is shown in Table 7.2.

[11] See Appendix D.

Table 7.2 Overall Evaluation of RE-tools According to Organisational Needs in Particular Environment.
FEF1.6 Define and maintain requirements constraints. FEF1.7 Allow requirements definition in the abstract level. FEF1.6 and FEF1.7 are features, suggested by participant during the quantitative analysis. Agreement of the feature FEF1.3 is bigger than the threshold *t*=5. Because of that the feature was not considered during tool evaluation.

Featu-res	Agree-ment	Impor-tance	RE-tools							
			CORE 3.1. Trial	DOORS 5.2	CaliberRM Web v,4.0	Requisite Pro	VitalLink	XTie-RT 3.1	RDT Version 3.0	Cradle-4
FEF1.6	-	10	2	1	2	1	2	2	2	2
FEF1.7	-	10	1	1	1	1	1	1	1	1
FEF1.2	1,4	8,3	3	2	2	2	1	1	1	2
FEF2.3	1,9	8,3	1	2	2	2	1	1	1	1
FEF3.2	1,9	8,3	2	3	2	3	3	2	2	2
FEF2.1	2,6	8,2	2	2	3	3	2	2	2	2
FEF1.1	2,8	8,1	3	2	2	2	3	2	3	2
FEF1.5	3,0	7,9	2	2	2	2	1	2	2	3
FEF3.3	3,4	7,9	1	1	1	2	1	2	2	3
FEF2.4	3,6	7,6	1	3	2	2	2	1	2	1
FEF1.4	3,6	7,5	3	2	1	2	2	2	3	3
FEF2.2	4,3	7,4	2	3	2	3	3	3	3	2
FEF3.1	4,6	7,3	1	2	2	2	1	2	2	3
FEF1.3	8,8	6,5	-	-	-	-	-	-	-	-
Overall evaluation:			**196,4**	**209**	**196,4**	**217,5**	**187,7**	**186,8**	**210**	**218,3**

Table 7.3 Features and Activities for the Evaluation Framework Suggested by Respondents during the Quantitative Analysis

	Description	**Comments**
Features	FEF1.6 Define and maintain requirements constraints.	Requirements constraints specify rules and structure according to which the requirements representation is prepared.
	FEF1.7 Allow requirements definition on the abstract level.	The feature was assigned to the requirements representation dimension, but it also contributes to the specification dimension, because it defines basis for specification reuse.
Activities	Allow import/export of requirements structure.	The activity was added to the FEF1.5 feature list. The respondents made a prediction, that it was not possible to have all the representation possibilities in a single RE-tool. And that additional tool support was needed.
	Allow viewing requirements before acceptance.	The activity extends the feature FEF2.3, which describes the collaborative work facilities. They are dealing with maintaining the rationale behind requirements' elements and suggest the means for agreement of requirements.
	Allow writing comments to requirements and their properties.	The activity was suggested to extend the feature FEF2.3. But it may also be added to FEF2.2, which describes the functionality related to requirements attributes and properties.
	Correspond to standards deployed in a software developing company.	It is important to have a unified requirements specification form subject to the environment characteristics and the working practice. This activity suggests a definition of such a standard in an operating company.

7.2.5 RE-tool limitations

The results of the quantitative analysis and the RE-tools evaluation demonstrated how to adapt the evaluation framework to the environment. First, the quantitative analysis allocates weights to evaluate the functionality of the RE-tools. Weights highlight more important RE-tool characteristics against less important, in a way that evaluation would correspond to the organisational activities better.

Second, the quantitative analysis validates the features and activities of the RE-tools evaluation framework. The performed investigation shows correspondence between analytical and practical studies. Third, the quantitative analysis discovers features and activities, which are important in an environment, but not mentioned in the evaluation framework. The suggested features and activities are shown in Table 7.3. Finally, the quantitative analysis shows the usability issues of the evaluation framework. After the motivating discussion, which took about half an hour, the questioning was performed in a relatively short period of time (13 minutes in average). The framework is easily applicable to the environment.

The evaluation results suggest some RE-tools as possible candidates (RequisitePro, Cradle-4, RDT version3.0, and DOORS 5.2). However, the evaluation showed the limitations of the RE-tools to solve RE problems (Kotonya and Sommerville, 1998):

Business needs are not considered. The RE process is seen as a technical process rather than a business process and it is dominated by technical concerns. RE-tools are designed for use by skilled specialists, proficient both in engineering methods and the functionality of the tool. The complex functionality of RE-tools is not suitable for non-proficient stakeholders. For example FEF3.3, functionality for the requirements specification preparation comprises a complex sequence of functions, and the tools do not suggest any templates to the document.

Stakeholder communication problems. Most of RE-tools are standalone applications and do not provide any (or provide weak) possibilities for collaboration work and communication between stakeholders. According to FEF2.3 evaluation, RE-tools lack brainstorming, discussion and negotiation functionality, which would support the rationale behind the requirements.

Lack of stakeholder involvement. None of the available RE-tools suit ideally for a use by a multidisciplinary, distributed team where the stakeholders have diverse skills and needs. For example, the FEF2.3 shows that the RE-tools does not provide means for collaborative work and the FEF2.1 has limitations in defining separate functionality for different user groups.

Lack of defined responsibilities. RE-tools should provide possibilities to define activity scenarios for each individual participant of the RE process and let people understand their individual responsibilities. Maintaining an audit trail of the RE process, such as engaging a mechanism for authentication and change approval request is important for RE-tools, because people having different educational and practice background are involved in the RE process and may not understand their individual responsibilities. Maintenance of requirements repositories (FEF3.1) would help to keep the predefined scenarios and reuse the guidelines for different user groups (FEF2.1). Help systems and data dictionaries (FEF2.4) should support and help the user to perform operations.

Lack of requirements management. If the RE process does not include effective techniques or methods, it may be introduced in an ad hoc way. But RE-tools like CASE tools operate according to the method, which is defined as a set of rules guiding the use of a RE-tool. Ad-hoc functionality would provide means to react to frequent changes of project requirements and needs during its development.

RE-tools usually deal only with informal (in some cases semi-formal) representations of RE processes and system requirements (FEF1.1 – FEF1.2). The functionality to specify abstract requirements definitions and requirements restrictions is poor, as well (FEF1.7).

The evaluation results could suggest usage the RE-tool with a highest evaluation result. However the study of the evaluation results shows, that the automated RE support is insufficient both for separating activities of RE and for management of the RE processes. Because of the RE-tools limitations, none of the evaluated RE-tools was chosen, but the editing, drawing, modelling, and communicating tools were used to prepare and maintain the requirements specification.

7.2.6 Maintaining Requirements Specification

Standard office tools (MS word and excel), modelling tools (Rational Rose and RML editor), graphical packages (MS paint), and communication tools (ICQ, MSN messengers, and e-mail systems) were used. IEEE std. 830-1998 (1998) recommendation was adapted to the requirements specification. RML editor (Sølvberg, 1999) facilitated in creation of the conceptual model of the problem. Rational Rose was used to describe behavioural model and to prepare the use case diagrams for the individual requirements. The participants chose the communication tools according to their experience. In order to support the communication ICQ and MSN messengers were used. The e-mail correspondence helped in distributing the requirements specification for the participant consideration.

The observation of the preparation and maintenance of the requirements specification was performed. The observation shows the list of shortcomings, needed for an automated support of the RE process. The shortcomings of RE using standard modelling and editing tools are covered by the evaluation framework and they contribute to validation of the features and activities for the evaluation framework. The observed shortcomings are:

Lack of automatic generation of standard requirements specification (feature FEF3.3). The requirements specification should correspond to standards (e.g.: IEEE std 830-1998), which should be maintained by an RE-tool. The requirements specification should separate between different RE phases, like requirements analysis and documentation. It should support different software development methodologies and life cycles, which usually depend on organisational policy. Some of the RE-tool candidates (e.g., Cradle-4) is supposed to cover this feature quite well, because it provides means to define organisational standards for documentation.

Requirements analysis and requirements specification is not separated (feature FEF3.2). Requirements analysis is the activity of learning the aspects of the problem domain to determine how to solve a specific set of users needs. Requirements analysis should be followed with different reports, agreement, negotiation, and documentation. Requirements specification is the activity of documenting a requirements specification. The FEF3.2 is one of the most important features, as the quantitative analysis has demonstrated.

Lack of requirements grouping (feature FEF2.2). The project involves different groups of requirements, for example, requirements for time constraints, functionality, usability, reliability, information storage, source code. The FEF2.2 is quite well supported by the RE-tools candidates (e.g., DOORS 5.2, RequisitePro, VitalLink, XTie-RT 3.1, and RDT Version 3.0) but the tools have many shortcomings concerning different modelling perspectives and participant viewpoints. Requirements grouping activities were carried out manually in our case. This is observed as the shortcoming taking in account a potential development size of the project.

Lack to represent requirements model in different techniques, including informal, semiformal, and formal representations (features FEF1.1, FEF1.2, and FEF1.3). The requirements model includes one logical requirements model, but the variety of project participants demands different representation techniques and requirements groups during all the RE process activities. Different representations are used during elicitation, analysis, validation and other RE activities. In the project two techniques were used for informal requirements representation – natural language and use case templates (Kulak and Guiney, 1998). Natural language requirements specification was the most suitable for the initial phases of RE process. The use case templates provided a way for specification of the structured information (in natural language), which contributed to requirements communication and requirements understandability.

Two techniques were used for semiformal requirements representation: state transition diagrams and reference modelling language diagrams (Sølvberg, 2000). State transition diagram represented the dynamic behaviour of the system. The requirements model, prepared with reference modelling language (Figure 7.2), aims to produce semantically sound models of the real world of the real phenomena, thus separating the real world modelling and traditional data-modelling.

In order to show intentional relationships of the problem elements, the requirements representations were extended with a formal requirements description (Figure 7.3). The set theory notations were used. They allowed the description of the requirements model using rules and predicates.

Cooperative work among multiple users is not supported (feature FEF2.3). The RE-tools should provide means for the collaborative work support. The RE-tool should include multidisciplinary teamwork, which could be geographically distributed. The tool should include not only the means for users working in an organisation, but also for a multidisciplinary teamwork, which could be geographically distributed. RE is the best performed by a cross-functional requirements team that provides an adequate experience base to capture all the requirements and to iterate them in a timely fashion. The FEF2.3 is supported quite weakly, in general case only by providing a web-based user interface. During the project, the requirements specification was distributed to different stakeholders by e-mail. That resulted in shortcomings for maintaining replication and specification versions.

Lack to provide means for communication and maintenance of rationale (feature FEF2.1). It is important to be able to recreate the rationale behind some requirements specification items, as analysed by Loucopoulos and Karakostas (1995). The communication means provide awareness to the participants and allow the collaborative work through all the RE process.

It was quite a challenging task of the project. First, due to the different communication tools (e-mail, MSN and ICQ messenger programs) it is difficult to support and argue appropriately different issues of requirements. Second, the rationale needs to be related to each element of requirements specification. Finally, the maintenance of rationale was kept as the track of e-mails sent about the different RE and system development questions among project participants.

None of the RE-tools candidates provide adequate support for argumentation. However, some of them (e.g., CaliberRM Web v.4.0, RequisitePro) support means for a baseline, and maintain requirements version control (e.g., CaliberRM Web v.4.0).

Lack of maintaining traceability relationship among different requirement elements (feature FEF1.4). It is important to keep traceable relationships among all the related information during the RE process. Traceability is needed to relate requirements, their rationale, source, requirements representation and requirements specification versions. One of the ways to ensure traceable relationships is to maintain requirements repositories. Some of the RE-tools candidates (e.g., CORE 3.1. Trial, RDT Version 3.0, and Cradle-4) provide adequate support for the FEF1.4. In the project traceability between related requirements, requirements properties and other traceable information was kept manually, and this again makes it difficult to perform when the requirements specification expands.

Lack of repository for storing data of a requirements specification (the feature FEF3.1). A RE-tool should support storage of requirements in a requirements repository instead of a paper document. The requirements repository stores requirements-related information such as: individual requirements, requirements metadata, different requirements representations, and requirements models. It should include different information formats like diagrams, tables, formal representations of requirements. The requirements repository benefit is to set traceable relations between various elements of the requirements specification. The repository would provide version control and reuse of already agreed common domain requirements. This feature was very poorly supported by the RE-tools candidates.

Lack of maintaining different data formats according to modelling techniques and tools used (feature FEF1.5). A tool should allow export and import of requirements models, prepared with other modelling tools. This would benefit the means to specify requirements using different paradigms and various modelling techniques. It would be beneficial to adopt requirements data interoperability between different tools.

Difficult to maintain flexible requirements management. Nguyen and Swatman (2003) show that RE process should not be characterised by a smooth and incremental evolution, but by occasional "crisis" points where the requirements models are reconceptualised and simplified. A RE-tool needs to promote design creativity and support reconceptualisation of the requirement model for restructuring the requirements specification.

The maintenance of requirements specification shows the important RE aspects, which are missing while using editing, drawing and modelling tools. The observation was performed along the features and activities of the evaluation framework and the results of the observation shows the limitations of to perform these activities without the adequate automated support. The execution phase demonstrates validation issues for the evaluation framework. It is easy to notice that the features and activities on the evaluation framework cover the shortcomings, arising during preparation and maintenance of requirements specification using standard editing, modelling and drawing tools.

7.2.7 Threats to Validity

The case study involves a number of validity threats. The analysis of the importance of the framework features and activities consists of researchers, but not of RE practitioners with RE experience or experience in using RE-tools. Participants had different computer science backgrounds and experiences. First, the participants describe the organisational reality from their own perspectives. Second, the different educational background is the potential threat that could affect the interpretation and understandability of the questions and reliability of the answers. In order to maintain the uniform interpretation of the questions, the discussion about the project was held, where the project objectives were presented.

Participants' experience and knowledge could produce a large variance of agreement for the features and activities of the evaluation framework during the quantitative analysis. The problem solution could be the setting of flexible threshold in order to remove the most non-agreed features and activities, but still leaving a sufficient list of features and activities.

However, non-agreed features and activities aren't unimportant, as the observation showed. Non-agreed features and activities need further investigation of the environment and the problem during the RE process. For example, feature FEF1.3, which says, that the requirements model should be described using formal techniques, was selected as a non-agreed one after the quantitative analysis (Table 7.2). However, observation of the maintenance of the requirements specification has shown that the formal requirements model is important for some groups of requirements, e.g., to describe intentional relationships between requirements (Figure 7.3).

The evaluation of the RE-tools was performed by the same researchers who had defined the evaluation framework. The preparation and maintenance of the requirements specification was done by one of the same researchers who are assumed to be friendly to the evaluation framework, and who selected the observations according to the framework features and activities.

7.2.8 Lessons Learned

In this case study the usability and adaptability of the evaluation framework for functional RE-tool requirements is one of the key issues during the evaluation of RE-tools. The important needs of stakeholders, who are going to work with an RE-tool, are explained. Further, the adaptability and usability studies show the validity of features and activities for the evaluation framework.

After the evaluation framework highlights the most important environmental needs, it becomes possible to describe RE-tools and to express the RE-tool requirements and needs of an environment. The evaluation of the tools was performed by the same researchers who had defined the evaluation framework in the first place, and it is important to recognise that the evaluation of the RE-tools candidates relies on the subjective opinions of evaluators. But the proposed techniques for the evaluation framework acquisition contribute towards the objective evaluation method, because different organisation's representatives are involved during the framework acquisition and the comparison of the RE-tools candidates.

The quantitative analysis showed that it is not enough to analyse functional RE-tool requirements for the acquisition. The RE-tool functional requirements describe the RE process, but for tools acquisition it is important to consider the RE-tool requirements such as cost of the tool, cost of the tool acquisition, project participant learning, time consumed for tools evaluation, tool adaptation to the environment, and similar requirements defined in the non-functional RE-tool framework in Chapter 5. The collected information of the quantitative research should be reused for the other RE-tools evaluation phases, where tool candidates could be tested with practical engineering examples.

The quantitative analysis provided useful knowledge for the maintenance of the requirements specification, using editing, drawing, communication and modelling tools. The observation of the maintenance of the requirements specification showed that the evaluation framework could be used to highlight the important features of the RE process and to evaluate them and the RE process itself according to a predefined way. The observation shows the critical requirements, needed for the automated support of RE processes.

It is important to recognise that any evaluation depends on the environmental characteristics and the organisational working profile. The framework adaptability and usability tests help to find out the important characteristics of the environment and the needs for an RE-tool acquisition. The execution phase consists of the preparation and maintenance of the requirements specification for the educational training course. The observation of the maintenance for the requirements specification using editing, drawing, communication and modelling tools highlights the requirements for the proper automatic support of the RE process. The results of both investigations – the quantitative analysis and observation – contribute to validation of the evaluation framework for RE-tools.

7.3 Case Study B: Comparison of Two RE Processes

RE could be performed using various software tools. The combination of general tools does not provide adequate quality for the RE process, and many RE activities are done manually as shown in Case study A. The targeted RE-tools could be a solution. The RE-tools tend to improve the quality by suggesting the means to perform RE activities in a more efficient way than manual.

Case study B (Matulevičius, 2004c) is a continuation of the previous case study. In this case study the MEIS functionality was improved with additional features:

- student assistants should provide comments about the solution and evaluation;
- student assistants should disband reviewer groups;
- student assistants should accept or reject evaluations;
- student assistants should provide comments to the evaluations;

– student assistants should print the report about the whole student's performance during the semester.

In order to maintain the requirements specification for the MEIS extensions two RE-tools – CORE 3.1 Trial and RequisitePro – were applied. The engineer created traceability between requirements, functional and design models in CORE. CORE was used to create the requirements model as entity-relationships (ER) diagrams and functional flow block diagrams (FFBD). The tool does not support traceability between the elements on the same abstraction level and does not have discussion facilities. The traceability between requirements on the same abstraction level was created with RequisitePro, which dynamically links the requirements in Word documents to the information stored in a database. The rationale of the requirements model was kept as the discussions over the requirements model in the RequisitePro database.

The purpose of this case study is to compare two RE process, when they are executed using two different sets of tools (Figure 7.4): general tool are used in Case study A, targeted RE-tool are applied in Case study B. In the following the Reprocess and the quality of the requirements specifications is compared along two research questions:

1) What software tools do provide a qualitative support for the RE process?

2) Do the targeted RE-tools provide facilities to create better-quality requirements specification than standard editing and modelling tools?

The RE process is evaluated using the evaluation framework for the functional RE-tool requirements, described in Chapter 5. The requirements specification quality is compared according to the qualitative properties of requirements specification (Davis, 1993; Davis *et al.*, 1993) and the semiotic quality framework (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003).

7.3.1 RE Process Comparison

The evaluation framework separates the RE process into requirements representation, requirements agreement and requirements specification dimensions. In Tables 7.4, 7.5, and 7.6 the evaluations mean: *Yes* – the feature is supported; *No* – the feature is not supported; *Partly* – the support is not adequate.

Comparison of the cases A and B along the *requirements representation dimension* is shown in Table 7.4. In case A the engineers chose the modelling perspectives, what they prefer. The requirements are represented in natural and reference modelling languages, state transition diagrams, the set theory notations, use case templates and use case diagrams. In case B the engineers were dependent on the

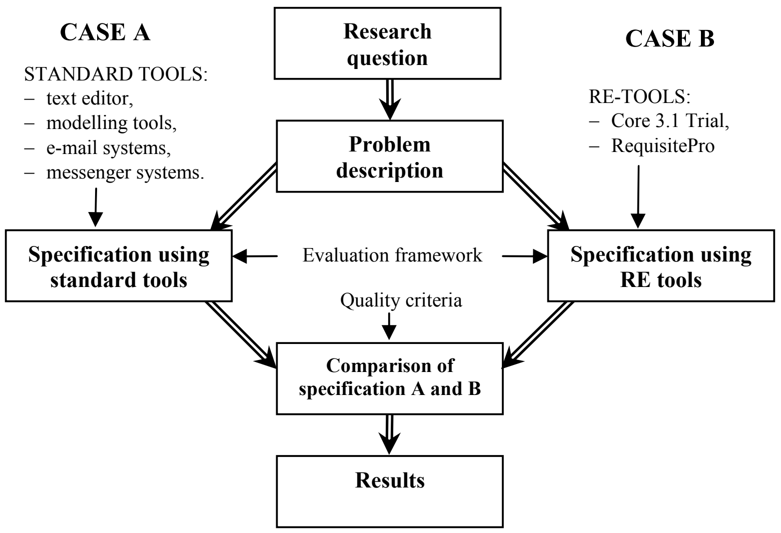


Figure 7.4 Case Study *B* Design

modelling perspective, supported by the RE-tool. The system description was prepared with CORE, using structured natural language, ER and FFBD diagrams.

In case A, requirements traceability was maintained manually. In case B the RE-tools provided different traceability: CORE supported hierarchical relationships between elements; RequisitePro maintained child-parent and peer-to-peer relationships.

The data exchange between the tools lets to use different modelling languages. In case A the representations screenshots were prepared with MS paint. In case B the connection between the RE-tools and the text editors was provided. However, the information transfer between CORE and RequisitePro was done using the text editor.

Comparison of the cases A and B along the *requirements agreement dimension* is shown in Table 7.5. The agreement about the requirements was achieved in face-to-face meetings in both cases. The requirements rationale was registered as the sequence of e-mail correspondence in case A. In case B, RequisitePro maintained discussions over the requirements model. In both cases, the rationale helped to reach understanding about requirements.

In case A the requirements specification versions were registered. The RE-tools registered the requirements revision history in case B. Both cases organised requirements according to the functional behaviour. Requirements views, traceability matrixes in RequisitePro helped to check requirements consistency.

Neither case A nor case B suggested the work for the geographically distributed teams. The requirements specification was distributed using e-mails in case A. In case B, the RE-tools suggested possibilities to access the databases through the

Table 7.4 Comparison along the Representation Dimension

Framework feature	Case A		Case B	
	Evaluation	Comments	Evaluation	Comments
FEF1.1.	YES	Natural language, Use Case templates.	YES	Structured natural language.
FEF1.2.	YES	State transition Use Case and RML diagrams.	YES	ER, and FFBD diagrams in CORE. Association with Rational Rose in RequisitePro (not used in project)
FEF1.3.	PARTLY	Set theory notations, prepared manually.	NO	RE-tools do not support formal requirements representation.
FEF1.4.	PARTLY	Hypertext links defined manually.	YES	Hierarchical traceability in CORE. Peer-to-peer, parent-child, traceability in RequisitePro.
FEF1.5.	NO	Performed manually by creating representation screenshots.	PARTLY	Requirements import/export from/to the text documents. No correspondence between RE-tools.

Table 7.5 Comparison along the Agreement Dimension

Framework feature	Case A		Case B	
	Evaluation	Comments	Evaluation	Comments
FEF2.1.	PARTLY	Rationale as the e-mail correspondence. Difficult to maintain version control.	YES	Rationale is created as RequisitePro discussions. Requirements access and revision registration.
FEF2.2.	PARTLY	Requirements are sorted according to the functionality.	YES	Requirements are sorted and filtered according to the functionality and properties.
FEF2.3.	NO	No cooperative work for the distributed teams.	PARTLY	No cooperative work for the distributed teams. Access of databases through intranet.
FEF2.4.	NO	Data and term dictionaries were maintained manually.	NO	Term dictionaries were created manually in the database. Help was provided to the skilled users.

local organisational intranet. The maintenance of term dictionary is poor both in case A and B.

Comparison of the cases along the *requirements specification dimension* is shown in Table 7.6. In case A "copy-paste" functionality could be used for reuse. In case B RequisitePro suggested the functionality to import related information from the existing projects.

Requirements reports could be prepared manually in case A. The RE-tools suggested a variety of viewpoints in case B. It is possible to print semiformal requirements representations in CORE. RequisitePro prepares the requirements views, traceability matrixes, change history and discussion reports.

Requirements standards lead to requirements understandability. In case A IEEE std. 830-1998 (1998) was adapted manually to the requirements specification. In case B, the requirements specification was prepared using CORE wizard. The defined template could be reused for other projects, too.

Table 7.6 Comparison along the Specification Dimension

Framework feature	Case A		Case B	
	Evaluation	**Comments**	**Evaluation**	**Comments**
FEF3.1.	NO	The reuse is manual by copy-paste functionality.	PARTLY	Reuse of information from o the projects in RequisitePro.
FEF3.2.	NO	Reports could be generated manually.	YES	Reports of views, discussion, representations, traceability.
FEF3.3.	YES	The IEEE std. 830-1998 was adapted by the requirements engineer.	YES	Organisational standard, created according to the RE-tool wizard.

7.3.2 Comparison of Requirements Specifications

The quality of requirements specification is evaluated by four researchers. The evaluator interests include conceptual modelling, RE, traceability and design of requirements repositories. The evaluators were not involved in the requirements specification development, but they were familiar with the MEIS system. Some evaluators were asked to perform double-cross evaluation of both requirements specifications.

The questionnaire (Figure 7.5) describes the requirements specification quality according to the semiotic quality framework (Krogstie 1998; Krogstie, 2001a; Krogstie and Jørgensen, 2003) and the qualitative properties (Davis, 1993; Davis *et al.*, 1993). The qualitative properties are divided to the requirement specification characteristics. For example, to evaluate the document consistency, the requirements behaviour representation, terms, and requirements properties were considered. The evaluators had to perform 39 characteristic evaluations in the scale of 5 to 0 (5 – best, 0 – worst evaluation).

1. Please evaluate the understandability of the

	5	**4**	**3**	**2**	**1**	**0**
Tables	o	o	o	o	o	o
Diagrams						
Singular requirements statements	o	o	o	o	o	o
Whole requirements specification	o	o	o	o	o	o

4. Please evaluate specification annotations.

	5	**4**	**3**	**2**	**1**	**0**
annotating by importance	o	o	o	o	o	o
annotating by relevant stability	o	o	o	o	o	o
Annotating by version	o	o	o	o	o	o

5. Please evaluate the third feature of specification completeness.

	5	**4**	**3**	**2**	**1**	**0**
page numeration	o	o	o	o	o	o
diagram titles	o	o	o	o	o	o
table titles	o	o	o	o	o	o
referenced material	o	o	o	o	o	o
all important terms	o	o	o	o	o	o
all important units	o	o	o	o	o	o
all needed concepts	o	o	o	o	o	o
all needed relationships between separate parts	o	o	o	o	o	o

Figure 7.5 Fragment of the Questionnaire to Evaluate the Requirements Specification

The results indicate that 25 (out of 39) characteristics of the requirements specifications are evaluated higher in case B than in case A. 12 characteristics are found better in the requirements specification produced in case A. And 2 characteristics are evaluated equally. Table 7.7 shows the discussion of the evaluated characteristics and the qualitative properties after analysis of the characteristics of requirements specifications.

7.3.3 Threats to Validity

The following possible threats to the validity of this case study have been identified:

- The case study is executed in the university. But the case study examines the real problem - the system used to manage student exercises.
- The evaluation framework for the functional RE-tool requirements, applied to compare the RE process, is originally created to evaluate the functionality of the RE-tools. However, the literature study did not suggest an evaluation method for the RE process. The evaluation framework for the RE-tool functionality is chosen for comparison, because it addresses the major quality types and it is produced in respect to the RE activities and process.
- The subjective choice of the modelling and RE-tools. The software tools in both cases are chosen according to the participant experience and availability of the tools. RML editor and Rational Rose are used for the teaching purposes. Both CORE and RequisitePro were chosen because of their availability in the Internet and because they got one of the highest scores in the tool evaluation in the Case study A.
- Most, if not all, of the quality properties are subjective (Wilson, Rosenberg and Hyatt, 1996). But the evaluators provided comments on the requirements specifications according the qualitative properties. The comments are used to improve the requirements specifications.
- Participants provided subjective evaluations. The individuals interpret the quality according to their experience. Nevertheless, peer-to-peer method is used in order to collect as objective opinion about the requirements specifications as possible.
- The case study and evaluation design are prepared by the same researcher. However, different participants were involved into the specification process, and the study contributes towards the objective evaluation results.
- The feasibly better requirements specification support and requirements specification in case B could be because of the learning effect. The engineer was already familiar with the problem domain in case B. But both requirements specifications pay emphasis on the different requirements. Even more, the engineer was not familiar with the RE-tools. It results feasibly equal learning efforts in both cases.

The last threat suggests the settings for the similar case study with a higher number of participants and with the specification development in parallel in order to avoid the learning effect.

Table 7.7 Comparison of the Qualitative Properties of the Requirements specifications
A – qualitative property of the requirement document in case A is higher;
B – qualitative property of the requirement document in case B is higher;
= - qualitative property of both requirements specifications is equal; I – indirect assumptions about the qualitative property.

Quality type	**Qualitative property**	**Evaluation results**	**Discussion**
Physical	Electronically stored	I	Both documents are stored electronically. Indirect assumption is that the evaluation is equal.
	Reusable	I	Both documents could be reused by “copy-paste” functionality. RequisitePro suggests the requirements extraction from the projects.
Empirical	Understandable	B	3 (out of 4) characteristics describe requirements specification of case B more understandable.
	Concise	I	The document in case B is longer than in case A (60 page in comparison to 43), but the evaluation shows better its quality.
Syntactic		=	The findings indicate syntactic correctness equal. The reason could be because both case have syntactic error prevention and detection means.
Semantic, Perceived semantic	Complete	B	9 evaluations indicate higher completeness of the requirements specification produced in case B. They include better definition of terms, units, and concepts, relationships, style of referenced material. 6 characteristics evaluated better for case A, concern numeration pages, figures, paragraphs and requirements.
	Consistent	B	Consistency was evaluated by consideration consistency of described system behaviour, terms and requirements definitions.
	Precision	A	Adequacy number of numeric qualities and level of precision was evaluated.
	Traced	B	The origin of separate requirements, the possibility to make design and implementation according to the requirements specification were evaluated.
	Traceable	B	
	Achievable	B	
	Design-independent	B	
	Annotated	B	Annotation level according to element importance, relevant stability and version was considered.
	At the right level of formality	B	Level of the document formality was evaluated for separate requirements and the whole requirements model.
	Unambiguous	=	Level of ambiguity was evaluated.
Pragmatic	Cross-referenced	A	Level cross referencing was evaluated.
	Organised	B	Level organisation was evaluated.
Social			Level of agreement about the requirement model is investigated through RE process.
Not evaluated qualitative properties are *verifiable*, *executable*, *interpretable*, and *prototypable*, because they are domain, design and implementation oriented, and *modifiable*, *correct*, and *not redundant*, because they are dependent on the skills of the evaluators.			

7.3.4 Lessons Learned

Two requirements specifications which differ in the use of the engineering tools, are considered in this case study. In the first case the office and modelling tools are applied, in the second the RE-tools are used. The RE process is analysed by the means of the evaluation framework for functional RE-tool requirements. Four experts investigate the quality of the requirements specifications according to the qualitative properties (Davis 1993; Davis et al. 1993).

The case study findings suggest the management of the requirements specification by the means of the RE-tools, instead of the general tools. The results suggest that the RE-tool vendors should emphasise and support their products to the industrial and academic environments, that the requirements engineers would propagate the use of the RE tools in the companies. The findings also challenge to engineer the quick and inexpensive RE-tool acquisition method according to the environmental settings. Furthermore, the case study findings show the improvements for the RE-tools:

- Improve facilities for the geographically distributed collaborative work. Collaborative work contributes for improvement of requirements analysis, negotiation and strategic planning (Regnell *et al.*, 2000; Damian *et al.*, 2003).
- Improve user training on modelling languages and methods. The tool tutorials present only main functions without consideration of the modelling perspective. The RE-tools are meant for experienced users (James, 1999; Kotonya and Sommerville, 1998), who are familiar with the tool supported modelling language. However, the RE-tools should help both experienced and not skilled users.
- Improve RE support by the ad-hoc functionality. Although the RE tools suggest the basic functionality for the RE activities, they: 1) lack ad hoc functionality (Nguyen and Swatman, 2003); 2) do not provide the pre-configuration to organisational needs (James, 1999); 3) act as CASE tools (Kelly, Lyytinen and Rossi, 1996); 4) lack "how to do" scenarios.
- Provide means to create formal representation. None of the RE-tools represent requirements formally. Formal model contributes with operational semantics, and leads to the executable and interpretable requirements specification (Davis, 1993; Davis *et al.*, 1993).
- Improve information interchange among software tools. The support of different modelling facilities under the same environment would ensure the information control, understandability and flexibility of a requirements specification.
- Improve the reuse of the requirements. The RE-tools, however, support reuse of syntactic aspects. Semantic and pragmatic viewpoints are addressed through the engineer. Maintenance of repositories would allow the reuse of requirements specification at different levels of formality through various projects.

- Support standard requirements documentation. Vendors of RE-tools claim that their products support international standards. However, the RE-tools support standards, defined within the organisation. The tools should provide guidelines of standard applicability.
- Provide requirements numeration schemes. RE-tools should suggest guidelines for the uniquely identifiable requirement numeration according to semantic dependency to requirements group and domain. The functionality would improve the precision, completeness and cross reference of a requirements specification.

The presented requirements specification process is feasible small and thus might be not representative for the larger information system development, where the automated support is most heavily needed.

7.4 Case Study C: Comparison of Evaluation Techniques

As discussed in Chapter 6 RE-tool testing could be performed using different evaluation techniques. For instance, *tool tutorials* provided by tool vendors, suggest teaching on the main RE-tool functionality. On another hand *evaluation scenarios* prepared for different RE-tools on the same problem, allow testing of similar RE-tool functionality across different tool, giving a better possibility for comparison. The purpose of the Case study C (Matulevičius, Karlsson and Sindre, 2004) is to investigate which evaluation techniques do provide better means to receive a qualitative RE-tool assessment. The following research questions are formulated:

1) *How do different evaluation techniques influence the RE-tool evaluation quality?*
2) *Which the RE-tool features are captured during the evaluation process using different evaluation techniques?*

The evaluation quality depends on the abstraction level at which the evaluation is done, as well as completeness and consistency of the evaluated features. The objective dependent variables are the evaluation *time* and the ratio of *not-tested* tool features. The subjective dependent variables are the RE-tool *usability* and *understandability*, the *satisfaction* and *ease of* technique *use*, and the RE-tool functionality *evaluations*.

Case study C consists of two sessions and it was executed at the Norwegian University of Science and Technology (NTNU) and the Lund University (LU). 6 students from NTNU and 30 students from LU participated in the experiment. They worked together in teams of 2 students.

The students from NTNU were in their graduate course. They had knowledge of information systems engineering, CASE tools, conceptual modelling, and the semantic Web. The experiment session was *not* compulsory to them. To motivate the students, some reward, set according to the Norwegian legislation, was paid to them. The students from LU were from the third or fourth year. They had knowledge from courses in RE, and software development methodologies. Six participants also had experience with RE-tools. The session was carried out as a compulsory laboratory exercise of the RE course.

7.4.1 RE-tools and Evaluation Techniques

RE-tools. At NTNU the participants tested three RE-tools: CORE 3.1 Trial, Rational RequisitePro and RDT. Both CORE and RDT are fully functional versions, limited by the size of the created database. They were downloaded from the Internet. NTNU had a teaching and research licence of RequisitePro. At LU three RE-tools included DOORS, CaliberRM and Rational RequisitePro. LU has a teaching and research licence of DOORS. Other two tools were downloaded from the Internet. The vendors provided licenses, which include the evaluation period licences of 2-4 weeks.

At NTNU an ***evaluation framework*** for functional RE-tool requirements described in Chapter 5 was applied in order to evaluate the RE-tool functionality. In order to support the tool tests the ***evaluation scenarios***[12] were prepared for each RE-tool according to a problem[13] from the electricity domain. The problem describes how information system should support the process of electricity fault handling. All the scenarios include requirement import from text documents, traceability definition, preparation of requirements models, and final requirements specification.

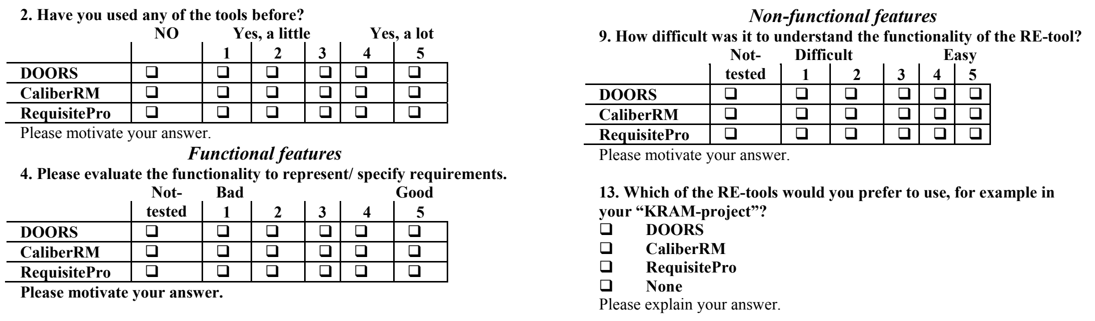
The ***feature-based questionnaire*** used at LU, consists of 13 questions (Figure 7.6[14]). The first two questions regard the respondents' experience with RE-tools. The third question asks about the self-preparation task. Next, questions consider RE-tool features: requirements representation, traceability, prioritisation, collaboration, report printing, usability, understandability, and tool integration with other software. The final question asks which of the RE-tools students would prefer for their projects at different companies.

The RE-tool tutorials applied at LU, are provided by the RE-tool vendors. The tutorials are meant to make the potential customers interested in a tool purchase and teaching the tool functionality.

7.4.2 Case Study Design

The case study consists of four phases (Figure 7.8): introduction, execution, evaluation and analysis. In the *introduction phase* the evaluation techniques are presented to the participants. The tools, the evaluation framework for functional RE-tool requirements and the test task were introduced to the participants in the session at NTNU. In session

[12] See Appendix F.
[13] See Appendix E.
[14] See Appendix H.



2. Have you used any of the tools before?

	NO	Yes, a little			Yes, a lot	
		1	2	3	4	5
DOORS	☐	☐	☐	☐	☐	☐
CaliberRM	☐	☐	☐	☐	☐	☐
RequisitePro	☐	☐	☐	☐	☐	☐

Please motivate your answer.

Functional features

4. Please evaluate the functionality to represent/ specify requirements.

	Not-tested	Bad 1	2	3	4	Good 5
DOORS	☐	☐	☐	☐	☐	☐
CaliberRM	☐	☐	☐	☐	☐	☐
RequisitePro	☐	☐	☐	☐	☐	☐

Please motivate your answer.

Non-functional features

9. How difficult was it to understand the functionality of the RE-tool?

	Not-tested	Difficult 1	2	3	4	Easy 5
DOORS	☐	☐	☐	☐	☐	☐
CaliberRM	☐	☐	☐	☐	☐	☐
RequisitePro	☐	☐	☐	☐	☐	☐

Please motivate your answer.

13. Which of the RE-tools would you prefer to use, for example in your "KRAM-project"?

☐ **DOORS**
☐ **CaliberRM**
☐ **RequisitePro**
☐ **None**

Please explain your answer.

Figure 7.6 Fragment of the Feature-based Questionnaire

at LU the participants were given the self-preparation task - to find the information about the RE-tools in the vendors' Web pages, before the session.

In the execution phase the RE-tools are tested. The student groups at NTNU tested the RE-tools in different order: the first group – RDT, RequisitePro, and CORE; the second – RequisitePro and CORE (software failure caused, that this group did not have time to test RDT); the third – CORE, RDT and RequisitePro. Each RE-tool was tested according to its test scenario. The framework-based form was filled in just after the tool was tested.

At LU there were two laboratory occasions, where different participants took part. The RE-tools were tested according to the tool tutorials in the same order: DOORS, CaliberRM and RequisitePro. The questionnaires were filled in after testing all the RE-tools.

In the *evaluation phase* they are evaluated. The features were evaluated in the scale of 5 to 0 (5 – best, 1 – worst evaluation, 0 – not-tested). At NTNU the RE-tool functionality is evaluated with the evaluation framework for the functional RE-tool requirements. The framework was adapted to the questionnaire[15]. The respondents performed 40 activity evaluations. Next, the participants evaluated the non-functional tool features, evaluation techniques and experiment process with the additional questionnaire. The evaluation phase at LU consisted of one questionnaire, which intends to compare eight features of the RE-tools.

In the *evaluation phase* they are evaluated. The features were evaluated in the scale of 5 to 0 (5 – best, 1 – worst evaluation, 0 – not-tested). At NTNU the RE-tool functionality is evaluated with the evaluation framework for the functional RE-tool requirements. The framework was adapted to the questionnaire[16]. The respondents performed 40 activity evaluations. Next, the participants evaluated the non-functional tool features, evaluation techniques and experiment process with the additional

[15] See Appendix C.

[16] See Appendix C.

questionnaire. The evaluation phase at LU consisted of one questionnaire, which intends to compare eight features of the RE-tools.

Finally, the *analysis phase* calculates and summarises the results (Tables 7.8, 7.9 and 7.10). At NTNU a framework feature evaluation is the average of activity evaluations. At LU a feature evaluation is the average of participants' evaluations for this feature. The weight is the average of the RE-tool feature evaluations. The not-tested ratio is the percentage of tool features, which got evaluation '0'. The weighted evaluation is the multiplication between the weight and the feature evaluation.

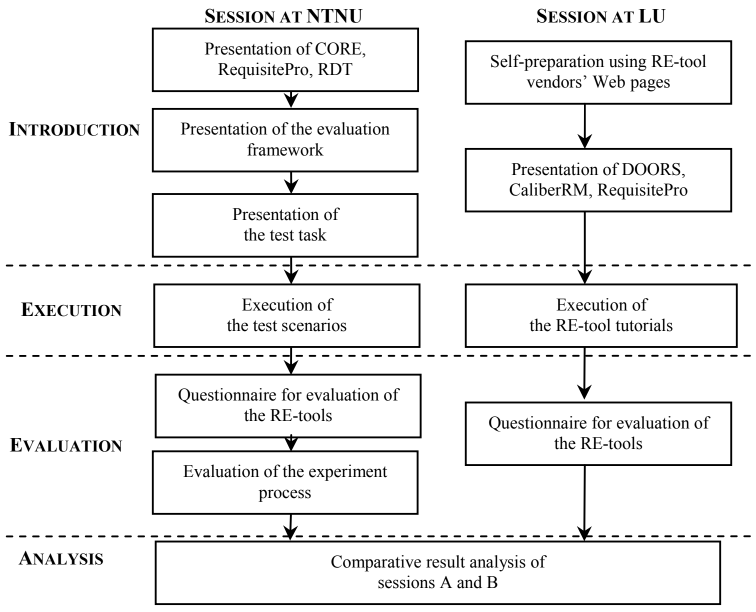

SESSION AT NTNU
SESSION AT LU
INTRODUCTION
Presentation of CORE, RequisitePro, RDT
Presentation of the evaluation framework
Presentation of the test task
Self-preparation using RE-tool vendors' Web pages
Presentation of DOORS, CaliberRM, RequisitePro
EXECUTION
Execution of the test scenarios
Execution of the RE-tool tutorials
EVALUATION
Questionnaire for evaluation of the RE-tools
Evaluation of the experiment process
Questionnaire for evaluation of the RE-tools
ANALYSIS
Comparative result analysis of sessions A and B

Figure 7.7 Design of Case Study C

7.4.3 Case Study Results

The results are analysed according to the objective and subjective variables, defined in the introduction of this case study. The RE-tool functionality evaluations are shown in Tables 7.8[17] and 7.9[18].

Evaluation correspondences. RE-tool features evaluated as well supported are traceability (FEF1.4 and *traceability*), RE-tool integration with other software (FEF1.5; at LU tool integration is evaluated with additional question) and informal

[17] See Appendix G.
[18] See Appendix H.

requirements representation (FEF1.1 and *representation*). Session at NTNU shows that the formal representation is the least supported feature; however the separation between representations was not considered in session at LU. A feature evaluated as weakly supported is the requirements prioritisation (FEF2.2 and *prioritisation*).

Evaluation differences. Two features got relatively different evaluations. They are report printing (FEF3.2 and *reports*) and collaboration (FEF2.1, FEF2.3 and *collaboration*). At NTNU the participants printed reports; however, the functionality was not available at LU. The collaboration is not guided in the tutorials, but was described in the test scenarios.

Features evaluated ***only at NTNU*** are repository functionality (FEF3.1), the user support at different level (FEF2.4), semiformal (FEF1.2) and formal (FEF1.3) requirements representations**.** Feature evaluated ***only at LU*** is the quality of the information, provided by tool vendors on the Internet.

Table 7.8 Session A Results

FEATURES (The feature deal with …)	WEIGHT	NOT TESTED	FEATURE EVALUATION			WEIGHTED EVALUATION		
			RequisitePro	RDT	CORE	RequisitePro	RDT	CORE
FEF1.1. (… informal representation)	**4,25**	**0,0%**	4,78	4,00	3,89	20,32	17,00	16,53
FEF1.4. (… requirements traceability)	**2,91**	**27,5%**	3,17	3,05	2,57	9,22	8,88	7,48
FEF1.5. (… integration with other tools)	**2,68**	**29,2%**	2,85	2,33	2,75	7,64	6,24	7,37
FEF3.2. (… report printing)	**2,66**	**40,6%**	3,17	2,00	2,58	8,43	5,32	6,86
FEF1.2. (… semi formal representation)	**2,39**	**37,5%**	3,42	1,00	2,28	8,17	2,39	5,45
FEF2.3. (… cooperative work)	**2,00**	**58,3%**	1,67	2,17	2,22	3,34	4,34	4,44
FEF2.1. (… user collaboration)	**1,94**	**50,0%**	2,63	2,45	0,90	5,10	4,75	1,75
FEF3.3. (… standard specification)	**1,92**	**54,2%**	1,44	1,83	2,44	2,76	3,51	4,68
FEF3.1. (… repository activities)	**1,86**	**56,3%**	1,71	2,00	1,92	3,18	3,72	3,57
FEF2.2. (… requirements specification)	**1,70**	**50,3%**	1,60	1,90	1,67	2,72	3,23	2,84
FEF2.4. (… user support)	**1,54**	**62,5%**	1,11	2,17	1,56	1,71	3,34	2,40
FEF1.3. (… formal representation)	**1,06**	**75,0%**	2,83	0,00	0,00	3,00	0,00	0,00
OVERALL:		**45,1%**	**30,38**	**24,90**	**24,78**	**75,60**	**62,73**	**63,38**

Table 7.9 Session B Result

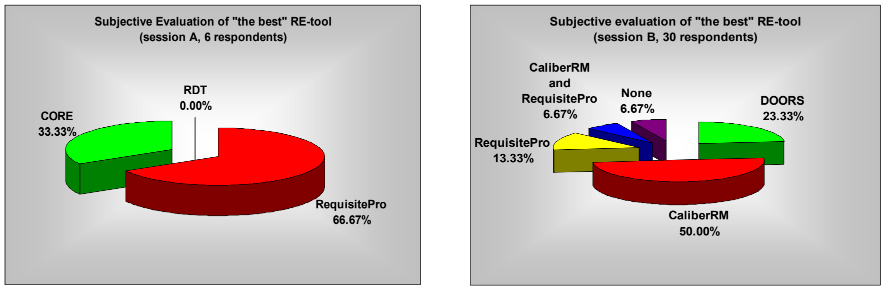
FEATURES	WEIGHT	NOT TESTED	FEATURE EVALUATION			WEIGHTED EVALUATION		
			CaliberRM	RequisitePro	DOORS	CaliberRM	RequisitePro	DOORS
Traceability	**3,60**	**1,1%**	4,20	3,57	3,03	15,12	12,84	10,92
Internet Information	**3,44**	**3,3%**	3,63	3,13	3,57	12,51	10,79	12,29
Representation	**3,28**	**2,2%**	3,77	3,17	2,90	12,35	10,38	9,51
Understandability	**3,26**	**0,0%**	3,90	3,17	2,70	12,70	10,31	8,79
Usability	**3,11**	**1,1%**	3,87	3,03	2,43	12,03	9,44	7,57
Collaboration	**2,16**	**37,8%**	2,37	1,30	2,70	5,10	2,80	5,82
Reports	**1,90**	**44,4%**	2,03	1,80	1,87	3,86	3,42	3,55
Prioritisation	**1,79**	**20,0%**	1,70	1,83	1,83	3,04	3,28	3,28
OVERALL:		**13,5%**	**25,47**	**21,00**	**21,03**	**73,67**	**59,98**	**58,44**

RE-tool usability and understandability was evaluated with additional questionnaire at NTNU. The evaluation at LU contained several questions regarding the non-functional RE-tool requirements. The results (Tables 7.9 and 7.10) indicates positive respondents' opinion about the of non-functional tool features (understandability and usability). At NTNU participants had to answer: "which of the RE-tools helps to prepare "the best" specification?" 4 answers are RequisitePro, 2 - CORE (Figure 7.8a). All the respondents indicated that RDT is the most complex tool. However, the Table 7.8 shows that both CORE and RDT are evaluated equally. One group claimed that CORE is preferred rather than RequisitePro. But the opinion is influenced by the software crash during the evaluation. The functionality evaluation, done by this group, is higher for RequisitePro.

At LU another question was asked: "which tool would you prefer to use in your own project?" 15 respondents would prefer CaliberRM, 7 - DOORS, 4 – RequisitePro, 4 – both CaliberRM and RequisitePro, and 2 respondents would not use any RE-tool (Figure 7.8b). Although, DOORS and RequisitePro functionality is evaluated equally, the questionnaire indicates that 6 respondents had experience of using DOORS. This might have influenced the higher DOORS evaluation.

Satisfaction and ease of technique use. At NTNU the respondents think positively about the framework efficiency and usefulness (Table 7.10). Some problems could be related to feature understandability and it could be a reason for the high not-tested rate. Experiment satisfaction is influenced by the RE-tools and framework performance and understandability. All the participants commented that the evaluation session was very intensive, and they felt a lack of time.

At LU ease of use and satisfaction of the evaluation techniques were not analysed. Nevertheless, most participants found the session interesting and instructive, although, some participants complained about headache after the session due to the small tutorial fonts.



a) in session A **b) in session B**

Figure 7.8 Subjective Evaluations of "the Best" RE-tool

Table 7.10 Evaluation of Non-functional Features in Session A

	NON-FUNCTIONAL FEATURES	EVALUATION
RE-tools	Usefulness of RE-tools	3.83
	Usability of RE-tools	3.67
	Quality of requirements specification, prepared with RE-tools	3.58
	Understandability of RE-tools	3.50
	RE process support with RE-tools	3.50
Framework	Efficiency of the framework	3.83
	Usefulness of the framework	3.58
	Understandability of the framework	3.52
Experiment	Understandability of the experiment	4.17
	Understandability of the task	3.67
	Satisfaction of the experiment	3.50

Time to conclude the evaluation. The time constraints used in experiment, are shown in Table 7.11. At NTNU the session was performed in four hours; however the participants indicated that the time was too short. The first tool test was performed in 70 minutes, the others in approximately 40 minutes. The reason is that the participants had to learn the evaluation framework.

At LU the session was also planned for four hours, but the participants performed it in two and a half, on average. At LU introduction consists of self-preparation (1h 30 min) and introduction of session settings which was given in 15 minutes. The tutorials execution took 2 hours, the RE-tools evaluation was performed in half an hour.

Table 7.11 Time Schedule for the Experiment Phases

Phase	Session A	Session B
Introduction	1 h	15 min (1 h 30 min)
Execution	2 h 45 min	2 h
Evaluation	15 min	30 min

7.4.4 Threats to Validity

The experiment involved students as subjects, and it was executed for educational purposes. The participant number was only 36, which may reduce the external validity. The participants had basic knowledge but limited experience with RE in practice. This situation may be similar to the one experienced by practitioners, who perform the evaluation in industry. The fact that a tool acquisition was not performed, may influence the level of commitment in the experiment.

Time limits and fatigue may influence the results, especially at the NTNU where less time was available and a larger feature number was evaluated. All the tools at the LU were evaluated in the same order, so there may be a learning effect or boredom effect which could influence the responses.

Some participants had experience with RE-tools, and this may have influenced their opinions. But since the experiment provides two independent sessions with correlating results, the participants' prior experience has most likely not affected the results. The experiment at the LU is based on tutorials; therefore, it is possible that the evaluation is affected by the differences between the tutorials.

The evaluations are influenced by limited tool functionality. At the NTNU, CORE and RDT were limited by the database size, and the participants had reliability problem with RequisitePro. At the LU the laboratory computers had no MS Word^{TM} installed, and RequisitePro and CaliberRM tutorials had to be adjusted so that it only included tasks that do not require integration with MS Word^{TM}.

The internal validity regards whether the treatment actually cause the outcome. In this experiment, two sessions differ from each other in two ways: one investigates a framework-based approach using test scenarios, while another explores a feature-based questionnaire and uses tutorials. Therefore it may be difficult to determine which treatment that actually causes the results.

7.4.5 Lessons Learned

The framework for functional RE-tool requirements provides a wide list of RE-tool features. But sometime it is not necessary to evaluate all of them because evaluators could be interested only in some features (e.g., traceability). The prioritisation is a possible improvement for the framework application. First, it would improve the understandability and reduce the not-tested rate. Second, it would decrease the feature number and the evaluation time.

The case study shows that the framework-based evaluation is more time consuming than using the questionnaire. But the framework prepares a more detailed evaluation than questionnaire-based assessment which considers tools on a high level. It is recommended to use the framework, when the purpose is to acquire a tool for targeted RE activities. The questionnaire-based technique evaluates the RE-tools on an abstract level, but the questionnaire construction is time consuming in comparison to the framework application.

The evaluation scenarios allow testing of the same RE-tool functions and they could be created for any problem domain, which is familiar to the evaluators. However, the scenario preparation is relatively time-consuming and involves RE-tool and problem domain experts. The tool tutorials are designed for commercial purposes and emphasise different functionality, and are difficult to compare. But running a tutorial is probably a common way to learn a tool and, therefore, this technique is interesting to use.

Session at LU showed an advantage of a comparative evaluation, when participants evaluated the same feature for all the RE-tools at the same time. Such comparisons provide more consistent results than the single feature evaluation.

The findings of the case study indicate that a combination of both techniques could be interesting. First, the tool candidate number is reduced with a feature-based questionnaire. Next, the detailed tool comparison is performed using the framework.

7.5 Case Study D: Testing the RE-tool Evaluation Approach

The validation could be performed through perception, performance and correctness tests. Perception involves investigation of framework usability, ease to use and user satisfaction (Hands, Peiris and Gregor, 2004). But Case study D analyses an experiment where the performance and correctness of the RE-tool evaluation approach (R-TEA) described in Chapter 6, are tested. The experiment considers whether the evaluation frameworks help to select an RE-tool which would yield a high-quality requirements specification.

Three research questions are formulated:

- ***Q1:*** *Does the evaluation framework help to select tools, which contribute to a high-quality requirements specification?*
- ***Q2:*** *Do evaluation scenarios provide better means to test RE-tools than RE-tool tutorials?*
- ***Q3:*** *Do RE-tools provide better support for the RE process and requirements specification than office tools?*

The first research question analyses the performance and correctness of the evaluation framework for the functional RE-tool requirements. The second research question considers evaluation techniques (*evaluation scenarios* and *RE-tool tutorials*) used during the tool assessment. The third research question investigates how RE-tools and office tools support the RE process and maintain quality of requirements specifications.

The case study was executed at the Norwegian University of Science and Technology (NTNU). 44 students of the 4th year participated in the experiment. The students were attending course Modelling of Information Systems, and the experiment was a part of the practical exercises of the course. The students were divided into ten groups of 4-5 persons. The experiment treatment involved the course material and theoretical lectures given to the students. However, the attendance of the lectures was not compulsory, so the participants had different knowledge and understanding of the experiment settings.

7.5.1 Case Study Design

The case study design is shown in Figure 7.9. As in case study C the ***problem***[19] used in the experiment is a natural language case description of the information system of an electrical power network company, in particular dealing with work processes relating to network fault handling. The problem statement also included the list of requirements which should be maintained in requirements specifications.

Tools. Four RE-tools were selected for the experiment: RDT, CORE, RequisitePro and CaliberRM. The RE-tool trial, evaluation and demonstration versions were downloaded from the Internet. RE-tools were fully functional versions limited by the number of database entries or evaluation period (2-4 weeks).

Office tools included text editors (e.g., MS Word™ and Latex), spreadsheets (e.g., MS Excel™), modelling tools (e.g. Visio), and communication tools (e.g., e-mail systems). The participants chose office tools according to their own preferences.

RE-tool tutorials were provided by the RE-tool vendors and downloaded from the Internet. The tutorials focus on teaching of tool functionality. ***Evaluation scenarios***[20] describe RE-tool functions to prepare an requirements specification for the problem. The scenarios comprise requirements import, maintenance of requirements traceability, requirements model creation, and requirements specification preparation and maintenance. In comparison to the RE-tool tutorials, evaluation scenarios describe the tool functionality, but the evaluation scenarios focus on the same problem, and RE-tool tutorials are meant for tool functionality guidance and self-study.

The ***framework prototype*** (Matulevičius, 2005) is considered in Chapter 8 and shown in Figure 7.10. It is implemented according to the evaluation framework for RE-tool functional requirements described in Chapter 5. The framework prototype supports the R-TEA method and guides evaluation though five steps. They are:

1. *Feature selection* – frameworks features and activities are selected for evaluation.
2. *Feature prioritisation* – the activities are prioritised according to their importance in the scale 0-to-10. 0 means non-important and 10 means very important activities.
3. *RE-tool selection*. The prototype users select the targeted tools for the evaluation from the list, which is entered to the prototype by the prototype administrator according to tool surveys (e.g., Table 3.3).
4. *RE-tool evaluation* – RE-tool functions are mapped with the selected framework activities. The tools evaluators have to consider question how well the activity is supported in the tool. The evaluation scale is 0-to-10, where 0 means that a tool feature is not supported and 10 means a very strong support of the feature.

[19] See Appendix E.
[20] See Appendix F.

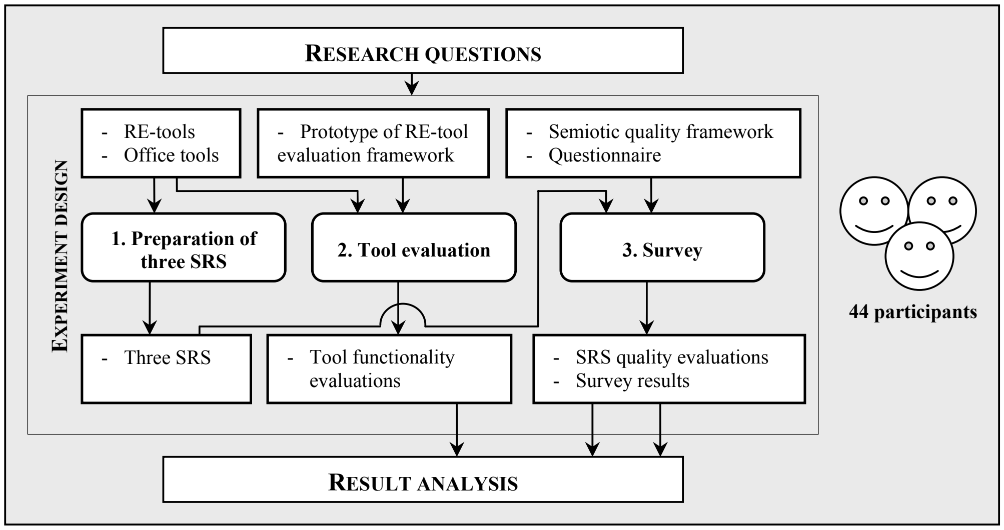
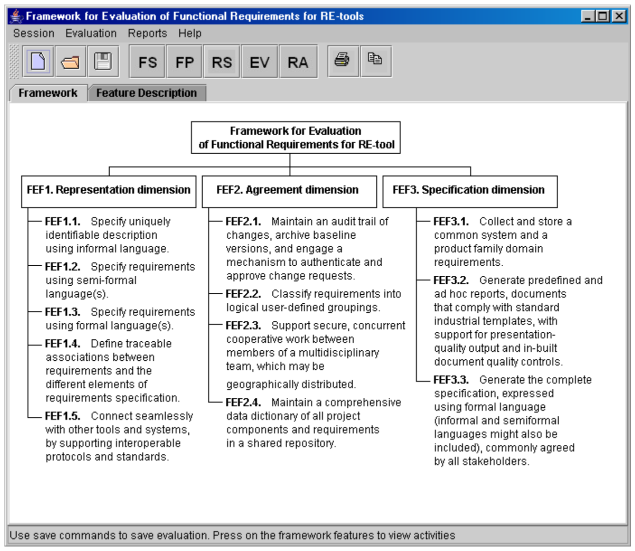


Figure 7.9 Cases study *D* Design



Framework for Evaluation of Functional Requirements for RE-tool

FEF1. Representation dimension

- **FEF1.1.** Specify uniquely identifiable description using informal language.
- **FEF1.2.** Specify requirements using semi-formal language(s).
- **FEF1.3.** Specify requirements using formal language(s).
- **FEF1.4.** Define traceable associations between requirements and the different elements of requirements specification.
- **FEF1.5.** Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.

FEF2. Agreement dimension

- **FEF2.1.** Maintain an audit trail of changes, archive baseline versions, and engage a mechanism to authenticate and approve change requests.
- **FEF2.2.** Classify requirements into logical user-defined groupings.
- **FEF2.3.** Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.
- **FEF2.4.** Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.

FEF3. Specification dimension

- **FEF3.1.** Collect and store a common system and a product family domain requirements.
- **FEF3.2.** Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.
- **FEF3.3.** Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.

Use save commands to save evaluation. Press on the framework features to view activities

Figure 7.10 Prototype of the Framework for Evaluation of Functional RE-tool Requirements

5. *Result analysis* – the final score for each tool is calculated as the sum of multiplications between framework activity priorities (defined in phase 2) and RE-tool evaluation score (defined in phase 4).

In addition the framework prototype can maintain rationale by commenting any evaluation issue during framework activity selection and prioritisation, tool selection and evaluation, and result analysis.

The ***quality of the requirements specifications*** is considered according to the semiotic quality framework (Krogstie 1998; Krogstie 2001a; Krogstie and Jørgensen, 2003) described in Chapter 4.

Experiment tasks. The experiment comprised three tasks (Figure 7.9). The first task uses RE-tools and office tools and applies evaluation techniques in order to prepare three requirements specifications for the same problem (Figure 7.11). The first task is designed for the whole group. In order to prepare the first requirements specification (a), the group uses RE-tool #1 and the corresponding evaluation scenario. The evaluation scenario helps to understand the problem and to learn the RE-tool functionality. The second requirements specification is prepared for the same problem, but, first, the group get familiar with RE-tool #2 functionality according to the RE-tool tutorial (b). The third requirements specification for the same problem is prepared using office tools chosen by the group (c).

In the second task the participants used the prototype of the evaluation framework in order to evaluate the tool functionality and ability to support the RE process. The second task was performed individually by each participant. However, the framework prototype maintains evaluation rationale, and a group could communicate by commenting and discussing the evaluation issues.

The third task comprised the survey, which consisted of three parts. First, the semiotic quality framework (Krogstie 1998; Krogstie 2001a; Krogstie and Jørgensen, 2003) is applied to evaluate the quality of requirements specifications. Next, the

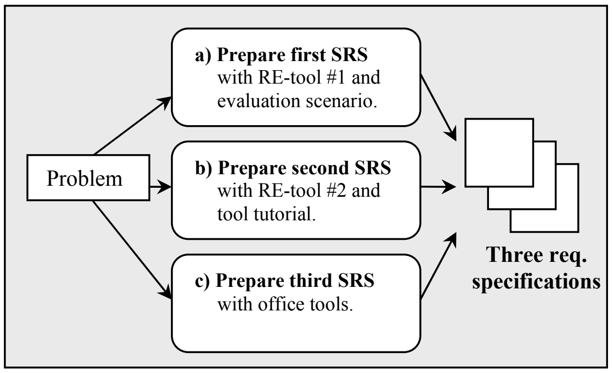

Problem
a) Prepare first SRS with RE-tool #1 and evaluation scenario.
b) Prepare second SRS with RE-tool #2 and tool tutorial.
c) Prepare third SRS with office tools.
Three req. specifications

Figure 7.11 First task: Preparation of Three Requirements Specifications

participants compared the evaluation techniques and the automated RE process support by different tools[21].

Post-evaluation is performed in order to reduce the subjectivity of the requirements specification quality evaluation. Four independent teaching assistants were evaluating the performance of each group and the quality of requirements specifications. First, the teaching assistants inspected the requirements specifications individually; next they had a discussion after which the "best-quality" requirements specification or requirements specifications were selected in each group.

7.5.2 Result Analysis Method

Result analysis method (Figure 7.12) comprises correlation analysis, descriptive statistics and hypothesis testing (Wohlin *et. al*, 2002). Table 7.12 shows the correspondence between the tool functionality and requirements specification quality evaluations[22]. The tool functionality evaluation is calculated as a sum of all framework activity evaluations:

$$\textit{Tool functionality} = \sum_{i}^{n}\sum_{j}^{m} p_{i,j}e_{i,j} ,$$

where n – group size (4 or 5 participants), m - number of framework activities, $p_{i,j}$ – the priority of activity j, set by group member i, and $e_{i,j}$ – evaluation of activity j, evaluated by group member i.

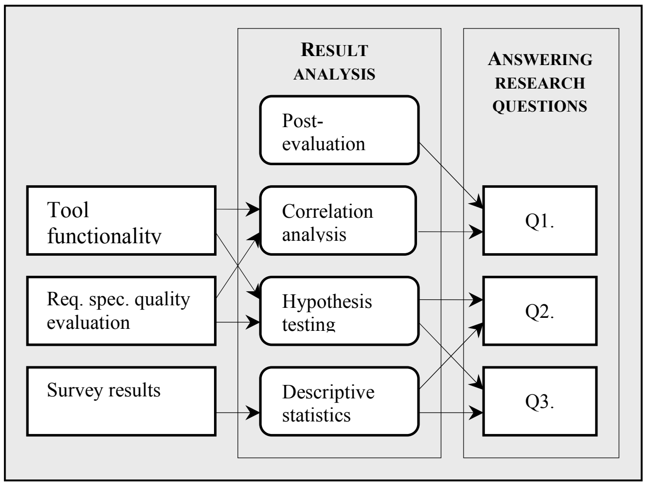


Figure 7.12 Result Analysis Method

[21] See Appendix I.
[22] See Appendix J.

The requirements specification quality evaluation is calculated as the *sum* of all quality type evaluations[23]:

$$SRS\ quality = \sum_{i}^{n}\sum_{j}^{t} q_{i,j}\ ,$$

where t – the number of quality types (e.g., syntactic, semantic, and pragmatic), and $q_{i,j}$– the evaluation of the requirements specification quality type j, evaluated by group member i. *Correlation coefficients* are calculated between tool functionality and requirements specification quality evaluations.

Hypothesis testing. In order to characterise the tool and requirements specification quality assessment, three null hypotheses are formulated:

- **H_{10}:** Evaluation scenarios and RE-tool tutorials contributes to the same RE-tool functionality evaluation.
- **H_{20}:** Evaluation scenarios and RE-tool tutorials contributes to the same quality of requirements specifications.
- **H_{30}:** The requirements specifications prepared with the RE-tools and office tools are of the same quality.

The hypothesis follows the results in Table 7.12, and the *pair t-test* is applied. H_{10} and H_{20} compare performance of the evaluation techniques to assess the tools and the requirements specification quality (Tables 7.13 and 7.14). H_{30} compares the requirements specification quality, where requirements specifications are prepared using different tools (Tables 7.17). If the null hypothesis is not rejected, nothing can be said about the outcome (Wohlin *et. al*, 2002). Therefore, the survey performed in task 3, is helpful here.

Descriptive statistics. The survey defines a number of subjective measures. Measures X1,…,X7 analyse participants' opinion about the evaluation techniques:

– **X1** – understandability of the RE-tool functionality;
– **X2** – learning of the RE-tool functionality;
– **X3** – understandability of the problem for which requirements specifications were prepared;
– **X4** – preparation of the requirements specification;
– **X5** – satisfaction of the RE-tool usage;
– **X6** – quality of the requirements specification;
– **X7** – evaluation of the RE-tools.

Measures Y1,…,Y7 characterise the participants' evaluation of the tool support for the RE process:

[23] See Appendix J.

– **Y1** – understandability of the RE process;
– **Y2** – support for RE activities;
– **Y3** – functionality to prepare the requirements specification;
– **Y4** – means to ensure the quality of the requirements specification;
– **Y5** – satisfaction of the tool usage during RE;
– **Y6** – usability to execute different RE activities;
– **Y7** – changeability and traceability of the requirements and requirements specification.

The *mode*, *median*, *mean*, *variance* and *standard deviation* of subjective measures are calculated in Tables 7.15 and 7.18. The *correlation coefficient* shows the dependencies between the subjective measures in Tables 7.16 and 7.19.

7.5.3 Case Study Results

Q1. In Table 7.12 the correlation coefficient between the tool functionality and requirements specification quality shows a direct dependency in seven groups (1, 4, 5, 7, 8, 9, and 10). Groups 1, 5, 8, 9, and 10 prepared the "highest-quality" requirements specification with the highest-scored functionality tool. The result is supported by the post-evaluation findings, where the teaching assistants selected the same 7 requirements specifications which were prepared with the high-evaluated functionality tools (Table 7.12, column 'post-evaluation).

However, in groups 2, 3, and 6 the correlation coefficient is negative. These results show that the high-evaluated tool functionality does not yield a high quality requirements specification. Negative or low correlation does not mean that there is no dependency. It rather defines that there are some other factors; e.g. low tool usability or participants' inexperience might have caused a low requirements specification quality.

In general, the experiment findings in Table 7.12 show tendency that the evaluation framework helps to select the tools which contribute with a high-quality requirements specification. However it is also important to evaluate tool non-functional characteristics, like tool usability, performance and evaluator's experience.

Q2. Table 7.13 shows summary statistics for the RE-tool functionality assessment using evaluation techniques. The t-test is found higher than critical value of t-test (2.2621). This means that H_{10} have to be rejected (α=0.05).

Table 7.13 shows summary statistics for the requirements specification quality evaluation using evaluation techniques. The t-test is found lower than critical value of t-test (2.2621). This means that H_{20} cannot be rejected (α=0.05).

Table 7.12 Evaluation of Tool Functionality and Requirements Specification Quality

Groups	Tools	Evaluation technique	Tool functionality	Req. spec. quality	Correlation coefficient	Post-evaluation (req. spec. preferred by teaching assistants)
Group1	RDT	Evaluation scenario	3603	119	0,9674	RequisitePro
	RequisitePro	RE-tool tutorial	3672	128		
	Office tools	-	2370	98		
Group2	CORE	Evaluation scenario	4844	101	-0,7450	CORE and MS Office
	CaliberRM	RE-tool tutorial	5583	102		
	Office tools	-	4448	127		
Group3	CORE	Evaluation scenario	4155	96	-0,6549	CORE
	CaliberRM	RE-tool tutorial	5378	81		
	Office tools	-	2933	91		
Group4	CORE	Evaluation scenario	1396	105	0,8811	CORE
	RequisitePro	RE-tool tutorial	1466	98		
	Office tools	-	805	87		
Group5	CORE	Evaluation scenario	2165	85	0,7060	Office tools
	CaliberRM	RE-tool tutorial	1850	83		
	Office tools	-	2238	105		
Group6	RequisitePro	Evaluation scenario	3571	110	-0,7626	RequisitePro
	CaliberRM	RE-tool tutorial	4536	109		
	Office tools	-	3167	126		
Group7	RDT	Evaluation scenario	1338	99	0,8765	CORE and Office tools
	CORE	RE-tool tutorial	1496	136		
	Office tools	-	1620	134		
Group8	RDT	Evaluation scenario	1626	70	0,2849	RDT, CORE and Office tools
	CORE	RE-tool tutorial	1656	87		
	Office tools	-	1173	75		
Group9	RDT	Evaluation scenario	3168	114	0,4881	RequisitePro
	RequisitePro	RE-tool tutorial	4255	148		
	Office tools	-	2460	135		
Group10	RDT	Evaluation scenario	4208	90	0,9926	CaliberRM and RDT
	CaliberRM	RE-tool tutorial	5108	93		
	Office tools	-	2133	77		

Table 7.13 t-test of the RE-tool Functionality Assessment Using Evaluation Techniques

	Mean	Standard deviation	*t-test*, $|t_0|$
Evaluation scenario	3007.4	1283.706	**2.858**
RE-tool tutorial	3500	1712.709	

Table 7.14 t-test of the Requirements Specification Quality Evaluation Using Evaluation Techniques

	Mean	Standard deviation	*t-test*, $|t_0|$
Evaluation scenario	98.9	14.579	**1.051**
RE-tool tutorial	106.5	23.377	

In Table 7.15 measures of central tendency (mode, median and mean) highlight better understandability of the problem to which requirements specifications were prepared (X3), preparation of the requirements specification (X4), satisfaction of the RE-tool usage (X5), quality of the requirements specification (X6) and the evaluation of the RE-tools (X7) while using evaluation scenarios than RE-tool tutorials.

The understandability of the RE-tool functionality (X1) and learning of tool functionality (X2) is higher using tool tutorials. The results are confirmed by H_{10}, too. The groups also commented that, first, they studied RE-tool tutorials to evaluate functionality, and later on they used evaluation scenarios to prepare requirements specifications.

In Table 7.16, correlation coefficients show a relatively strong dependency between RE-tool functionality (X1) and learning of functionality (X2), satisfaction of RE-tool usage (X5), and quality of the requirements specification (X6). The coefficient also indicates, that in order to evaluate an RE-tool (X7), it is important to learn tool functional features (X2).

Generally, the experiment findings indicate that the *evaluation scenarios* provide better or at least equal means to evaluate and compare RE-tools than RE-tool tutorials.

Q3. Table 7.17 shows summary statistics of the requirements specification quality evaluation using different tools to prepare requirements specifications. The t-test is found lower than critical value of t-test (2.2621) in both cases when RE-tools were used with different evaluation techniques. This means that H_{30} cannot be rejected (α=0.05).

In Table 7.18 only the mean of Y5 shows better satisfaction of the office tools. Other measures indicate higher performance of the RE-tools.

In Table 7.19 correlation coefficients suggest that satisfaction of tool usage (Y5) depends on understandability of the RE process (Y1), functionality to prepare requirements specification (Y3), and usability to execute different RE activities. The correlation coefficients also indicate a dependency between functionality to prepare the requirements specification (Y3) and usability to execute different RE activities (Y6).

Table 7.15 Descriptive Analysis of Evaluation Techniques (evaluation scale 1-to-6)

Measure	Mode	Median	Mean	Variance	Standard deviation
X1	5	4	3.66	2.46	1.57
X2	4	4	3.50	1.84	1.36
X3	3	3	2.89	1.56	1.26
X4	2	3	2.91	1.53	1.24
X5	2	3	3.20	1.93	1.39
X6	2	3	3.11	2.09	1.43
X7	2	3	3.43	2.02	1.42

The understandability of the RE process (Y1) correlates with the changeability and traceability of the requirements and requirements specification (Y7), which characterises the complexity of the requirements model.

The experiment results suggests to use the RE-tool to support the RE process and maintain the requirements specification rather than office tools.

Table 7.16 Correlations Coefficient between Evaluation Technique Measures

	X1	X2	X3	X4	X5	X6	X7
X1	**1,00**						
X2	0,49	**1,00**					
X3	0,17	0,06	**1,00**				
X4	0,38	0,40	0,31	**1,00**			
X5	0,45	0,41	0,09	0,24	**1,00**		
X6	0,53	0,01	0,35	0,28	0,30	**1,00**	
X7	0,34	0,60	0,07	0,21	0,35	0,25	**1,00**

Table 7.17 t-test of the Requirements Specification Quality, where Requirements Specifications are Prepared with RE-tools and Office Tools

	Mean	Standard deviation	*t-test, |t₀|*
RE-tools and evaluation scenario	98.9	14.579	**1.051**
Office tools	105.5	23.372	
RE-tools and tool tutorials	106.5	23.377	**0.171**
Office tools	105.5	23.372	

Table 7.18 Descriptive Analysis of Tool Usage (evaluation scale 1-to-6)

Measure	Mode	Median	Mean	Variance	Standard deviation
Y1	2	2	2.91	2.5	1.58
Y2	2	2	2.50	1.65	1.28
Y3	2	2	2.80	1.47	1.21
Y4	2	2	2.43	1.23	1.11
Y5	3	3	3.53	1.92	1.39
Y6	2	3	3.00	1.63	1.28
Y7	1	2	2.25	1.73	1.31

Table 7.19 Correlation Coefficient between Tool Measures

	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Y1	**1,00**						
Y2	0,35	**1,00**					
Y3	0,32	0,13	**1,00**				
Y4	0,13	0,48	0,00	**1,00**			
Y5	0,40	0,26	0,48	0,01	**1,00**		
Y6	0,15	0,34	0,47	0,16	0,50	**1,00**	
Y7	0,45	0,14	0,19	0,05	0,41	0,10	**1,00**

7.5.4 Threats to Validity

The case study design involves some possible threats to validity. The internal validity regards whether the treatment actually causes the outcome. In this case study, the participants were given the necessary material to prepare the requirements specifications and to evaluate their quality. Therefore it may be difficult to determine which treatment that actually causes the results.

The case study involves 44 students rather than practitioners who would be more relevant since the ultimate goal of the framework is to help industry to evaluate tools. The participants had basic knowledge but limited experience with RE in practice. However, they all were following the same course (and essentially been following the same study program for 3,5 years), i.e., participants were quite homogeneous regarding age and background. The use of students is a quite common research approach to evaluate the applicability of software engineering techniques (Sinha and Vessey, 1999; Benediktsson, Dalcher, Thorbergsson, 2004; Karlsson *et al.*, 2004). Since the participants were in their 4th year, they had only 1 year left of their studies, their knowledge and experience therefore being quite close at least to practitioners who just graduated.

The evaluation is based on subjective judgment. The individuals interpret the quality of requirements specifications and functionality of RE-tools according to their experience.

The quality of the evaluation scenarios which are prepared by one of the teaching assistants could influence both the RE-tool assessment and the quality of the requirements specifications. In order to mitigate the threat the scenarios were executed with the RE-tools by other two teaching assistants before the case study.

Time limits and fatigue may influence the results. After preparation of two requirements specifications using RE-tool the participants were tired. However, the learning effect could be noticed here as well, as the participants were already familiar with the problem to be specified, so the higher-quality requirements specification could be prepared using office tools.

External validity is influenced by the number of requirements, which was relatively small. Therefore many participants preferred office tools instead of RE-tools. The situation could be different if dealing with a large number of requirements changing over time, where the usefulness of advanced tools would be more evident.

7.5.5 Lessons Learnt

Case study D focuses on the performance and correctness of the evaluation framework for the functional RE-tool requirements, and uses the R-TEA method to guide through tool assessment. The case study compares two evaluation techniques (evaluation scenarios and RE-tool tutorials) and investigates the RE-tool and office tool performance to support the RE process. The findings indicate:

- The evaluation framework guides to the selection of the tools which help to prepare a qualitative requirements specification. However the evaluation process is subjective, in particular, it is much affected by the user experience with tools.
- Combination of both evaluation techniques allows better assessment of the RE-tools than using the techniques separately. Experiment findings recommend studying of the RE-tool tutorials first. Tutorials introduce the basic functionality of the RE-tools. Next, the evaluation scenarios help to investigate the similar RE-tool performance in the applied study according to the same problem domain.
- RE-tools do provide better support for the RE process than office tools. RE-tools help to understand and guide the RE process. RE-tools also provide the means to maintain the requirements specification changeability and quality; when using office tools more manual work is needed to maintain the requirements specification.
- Tool selection depends not only on tool functionality evaluation but also on tool usability characteristics (Goowin, 1987). The experiment indicated a poor usability of the RE-tools, although the RE-tool functionality is higher than office tools. The RE-tools are designed for use of the skilled specialists (Kotonya and Sommerville, 1998; Kaindl *et al.*, 2002; Matulevičius, 2004b) proficient both in engineering methods and the tool functionality. But this is not always the case when evaluating the RE-tools as the users could have different working experience and understanding of the RE process.

7.6 Summary of Case Studies

Framework for the functional RE-tool requirements. All the case studies analyses the evaluation framework for the *functional* RE-tool requirements and involves the investigation of the RE-tools. In Case study B the framework is applied to compare two RE processes. Although the framework is specifically designed for the RE-tools, but its origin allows its application to consider the RE process, too. The evaluation framework decomposes RE into activities and tasks, and can thus help to evaluate the RE process. The investigation of the RE practice shows how the features and activities of the evaluation framework characterise the RE process. But the importance of framework activities differs, because framework activities depend on the degree of the RE activities performed in the certain environment and the degree needed to support these activities.

In the following three characteristics which contribute to the validity of the evaluation framework for the functional RE-tool requirements are considered using hypothesis analysis method. The characteristics are:

- *agreement* about the framework features and activities;

– *importance* of the framework features and activities;

Both characteristics contribute are important during preparation of the requirements specification for the RE-tool acquisition. The statistical analysis show that the framework yield agreement about the most important environment needs.

– *rate of not tested* features and activities;

The statistical evaluation of the rate of not tested features indicate the degree of the tools evaluation in an environment and the degree of the RE-tool support for the RE process.

Agreement. Case studies A, C and D analyses agreement about the framework features (Table 7.20). Feature importance is calculated as average of the framework activities. The variance measure is applied in order to find out about the feature and activity agreement. However the question remains whether the evaluation framework helps to reach the agreement about the RE-tool features. In order to answer this question the null hypothesis is formulated:

H_{01}: Agreement about the framework features is the same in all case studies.

The *paired t-test* and a *one-way analysis of variance* (ANOVA) is used to test the H_{01} hypothesis. Tables 7.21 and 7.22 show the summary statistics, and they mean that it not possible to reject the null hypothesis.

Importance. The range of the framework feature importance indicates that the requirements representation features are the most important among other two. Next comes the requirements specification and requirements agreement.

Table 7.20 Comparison of Three Case Studies
Importance is calculated as average of framework activity evaluations, agreement is calculated as the variance of framework activity evaluations)

Framework feature	Case study A Evaluation scale 0-to-10		Case study C Evaluation scale 0-to-5		Case study D Evaluation scale 0-to-10	
	Importance	Agreement	Importance	Agreement	Importance	Agreement
FEF1.1	8.10	2.80	4.25	1.24	7.38	5.61
FEF1.2	8.30	1.40	2.39	4.62	6.23	4.58
FEF1.3	6.50	8.80	1.09	4.78	4.76	3.47
FEF1.4	7.50	3.60	2.91	3.65	6.01	4.10
FEF1.5	7.90	3.00	2.68	4.74	7.00	5.32
FEF2.1	8.20	2.60	1.94	4.60	5.18	4.24
FEF2.2	7.40	4.30	1.70	3.40	5.74	4.48
FEF2.3	8.30	1.90	2.00	5.91	4.64	6.03
FEF2.4	7.60	3.60	1.54	4.52	5.63	6.97
FEF3.1	7.30	4.60	1.86	4.76	5.09	3.83
FEF3.2	8.30	1.90	2.66	5.27	6.23	6.04
FEF3.3	7.90	3.40	1.92	4.69	6.93	3.45

Table 7.21 The Paired t-test for Agreement about Framework Features (t_{crit} = 2.20098, α=0.05)

	Mean	*Standard deviation*	*t-test, $\lvert t_0 \rvert$*
Case Study A	3.49	1.931	1.2927
Case Study C	4.35	1.172	
Case Study A	3.49	1.931	1.7472
Case Study D	4.84	1.134	
Case Study C	4.35	1.172	1.0612
Case Study D	4.84	1.134	

Table 7.22 ANOVA test for Agreement about Framework Features

	Mean	*Standard deviation*	*F*	*p-value*	*F crit*
Case Study A	3.49	1.931	2.6332	0.086894	3.2849
Case Study C	4.35	1.172			
Case Study D	4.84	1.134			

On the framework feature level (Table 7.23) high feature importance is indicated for requirements informal (FEF1.1) and semiformal (FEF1.2) representation, requirements model import/export from other tools (FEF1.5) traceability (FEF1.4) and report printing (FEF3.2). The medium range of feature importance is indicated for cooperative work support (FEF2.1 and FEF2.3), requirements sorting and prioritisation features (FEF2.2) and specification preparation (FEF3.3). Weak importance is indicated for user support at different level (FEF2.4), requirements repository activities (FEF3.1) and requirements formal representation (FEF1.3). The weak features support is also influenced by the RE-tool functionality as many RE-tools do not support them.

Not-tested Feature Rate. Case studies C and D analyses the rate of not tested features (Table 7.23). The null hypothesis is formulated to test whether the rate of not tested features is the same.

H_{01}: Rate of not-evaluated frameworks activities is the same in both cases C and D.

The paired *t-test* is used to test the H_{01} hypothesis on the rate measures from Table 7.24. Table 7.25 shows the summary statistics. It is possible to reject the null hypothesis, as the absolute value is higher than critical value of the *t*-test.

There are several factors, which influence rate of not-tested features. First, it is the understandability of the framework features and activities. The RE-tool evaluators could not comprehend all the features and activities easily and therefore leave them as not-evaluated. The difference between both cases could be influenced by the setting of the cases. In Case study D participants could choose eight framework features instead of twelve, evaluated in Case study C.

Finally the rate of not tested features indicates the RE process support by the RE-tools. As already discussed in Chapter 3 and as the results of the case studies show

Table 7.23 Range of Feature Importance

Features	Range			Mean	Variance
	Case A	Case C	Case D		
FEF1.1	5	1	1	2.33	5.33
FEF1.2	1	5	4	3.33	4.33
FEF1.5	6	3	2	3.67	4.33
FEF3.2	3	4	5	4.00	1.00
FEF1.4	9	2	6	5.67	12.33
FEF3.3	7	8	3	6.00	7.00
FEF2.1	4	7	9	6.67	6.33
FEF2.3	2	6	12	6.67	25.33
FEF2.2	10	10	7	9.00	3.00
FEF2.4	8	11	8	9.00	3.00
FEF3.1	11	9	10	10.00	1.00
FEF1.3	12	12	11	11.67	0.33

Table 7.24 Rate of Not-tested Framework Features

Feature	Case C	Case D
FEF1.1	0.0	0.6
FEF1.2	37.5	31.6
FEF1.3	75.0	35.2
FEF1.4	27.5	19.5
FEF1.5	29.5	20.6
FEF2.1	50.0	18.2
FEF2.2	50.3	14.0
FEF2.3	58.3	35.4
FEF2.4	62.5	17.0
FEF3.1	56.3	22.5
FEF3.2	40.6	24.7
FEF3.3	54.2	11.3

Table 7.25 The Paired t-test for the Rate of Not-tested Framework Features (t_{crit} = 2.20098, α=0.05)

	Mean	*Standard deviation*	*t-test, $\lvert t_0 \rvert$*
Case Study C	45.14	19.76	5.2115
Case Study D	20.88	10.10	

the RE-tools have different weaknesses to support the RE process and maintain the quality of the requirements specifications.

Framework for Non-functional RE-tool requirements. The case studies did not have the purpose to acquire the RE-tool(s) to the organisation, but rather to investigate the performance of the techniques used while evaluating the RE-tools. Therefore the validation of the evaluation framework for the *non-functional* RE-tool requirements is limited. However, the following observation could be made: all the case studies required definition of the environmental settings which corresponded to the external organisational requirements, such as decision making approach, investigation of the participant experience in software development and usage, and definition of social norms of the environment. The case studies were dealing with the RE-tool product requirements such as RE-tool usability, performance, reliability and maintainability. The findings report about poor usability and understandability of the RE-tools. The RE-tool maintainability and reliability features were not particularly evaluated in the cases; however, a number of RE-tools' misbehaviour (e.g., slow performance and the RE-tool crash) were noticed through all the cases.

The R-TEA method. All the case studies contribute to the R-TEA validity which is analysed using performance, correctness and usability tests. The case studies address the elicitation of the environmental needs and preparation of the requirements specification for the tool acquisition. In all the case studies the RE-tools are selected according to their availability and the information provided by the tool vendors in the Internet. Next the tool assessment is performed using the evaluation frameworks and applying different evaluation techniques.

None of the cases was dealing with the industrial environment were the validation of the frameworks and the R-TEA method would be much stronger, in particular if to apply the frameworks in the company which has decided to adapt the RE-tool for their RE process. However the case studies executed in an academic environment contribute positively towards validity of the R-TEA method.

7.7 Chapter Summary

In this chapter four case studies are presented. Case study A describes how to evaluate the organisational needs and to highlight the most important and agreed requirements for the RE-tool organisation. A number of limitations of the RE-tools to support the RE process, is highlighted. Case study B continues with the investigation of the RE process and the maintenance of the requirements specification using different office and RE-tools. The case study concludes that the RE activities are better supported if to execute them by the means of the RE-tool instead of the office tools. In the case study it is also observed that the requirements specification produced with the RE-tools is of higher quality. Case study C analyses two evaluation processes where different evaluation techniques are applied. The case study concludes that combination of different techniques could result with a more detail RE-tool evaluation results, but this is time consuming activity. The organisation should properly define the evaluation process requirements before stating the evaluation sessions. Finally, in Case study D correctness and performance of the evaluation framework is considered. The case study investigates whether the framework helps to select an RE-tool(s) which yield a high quality requirements specifications. The findings indicate a positive tendency of the framework application. However the organisation should also consider many organisational and social factors which influence the RE-tool acceptance and usability.

In the next chapter the tool prototypes constructed according to the proposed frameworks and the R-TEA method, are presented. The first prototype is applied in the Case study D and it facilitates the evaluation of the RE-tool. The second prototype describes a design-implementation environment for research of the weak features of the RE-tools.

CHAPTER 8

Prototypes

In the previous chapter four case studies were presented in order to validate the proposed evaluation frameworks and the RE-tool evaluation approach (R-TEA). This chapter presents two prototype tools. The first prototype used in Case study D, facilitates the application of the evaluation framework for the functional RE-tool requirements. It also supports the guidelines of the R-TEA method.

Throughout the work both in a literature study and the case studies, a number of RE-tool weaknesses are discovered and highlighted. The second prototype FB-RET (fragment-based RE-tool) presents an environment which suggests possibilities to consider the RE-tool weaknesses in design-implementation settings.

8.1 Prototype: Framework for Functional RE-tool Requirements

Most of the COTS product evaluation approaches analysed in Chapter 4, manages the tool selections process suggesting manual activities (in the best case separate tools for requirements prioritisation or/and rationale maintenance). Only some of the approaches are followed with the software tools to help the tool selection and decision support process. Maiden *et al.* (2003) suggest selecting components against requirements (SCARLET) Process Advisor. The tool is based on the previously discussed PORE method (Maiden and Ncube, 1998), and it integrates workflow and internet technologies with situated process models to offer the right advise at the right time during a tool or component selection process. Elsewhere a tool prototype to construct a quality model is suggested by Carvallo *et al.* (2004). The prototype supports the quality-based evaluation approach; however it is based only on non-functional requirements as it follows the ISO 9126 standard (1991).

In Chapter 5 the evaluation framework for functional RE-tool requirements is described. The *framework prototype* (Matulevičius, 2005) is a client of a repository-based application. Its purpose is to guide the RE-tool evaluators through the RE-tool selection according to the R-TEA method described in Chapter 6. The prototype consists of three main components (Figure 8.1): 1) *repository* where the evaluation and management data are stored; 2) *administrator application* where user accounts and sessions are created and administered (Figure 8.2); and 3) *evaluator application* where the prototype users perform evaluation activities (Figure 8.3) following the guidelines of the R-TEA method.

The prototype is a *JAVA* application and uses the *MySQL* database management system to support repository activities. The repository is installed in the server computer. Both the administrator and evaluator applications connect the server computer through the MySQL Connector/J interface which allows access from any geographical location.

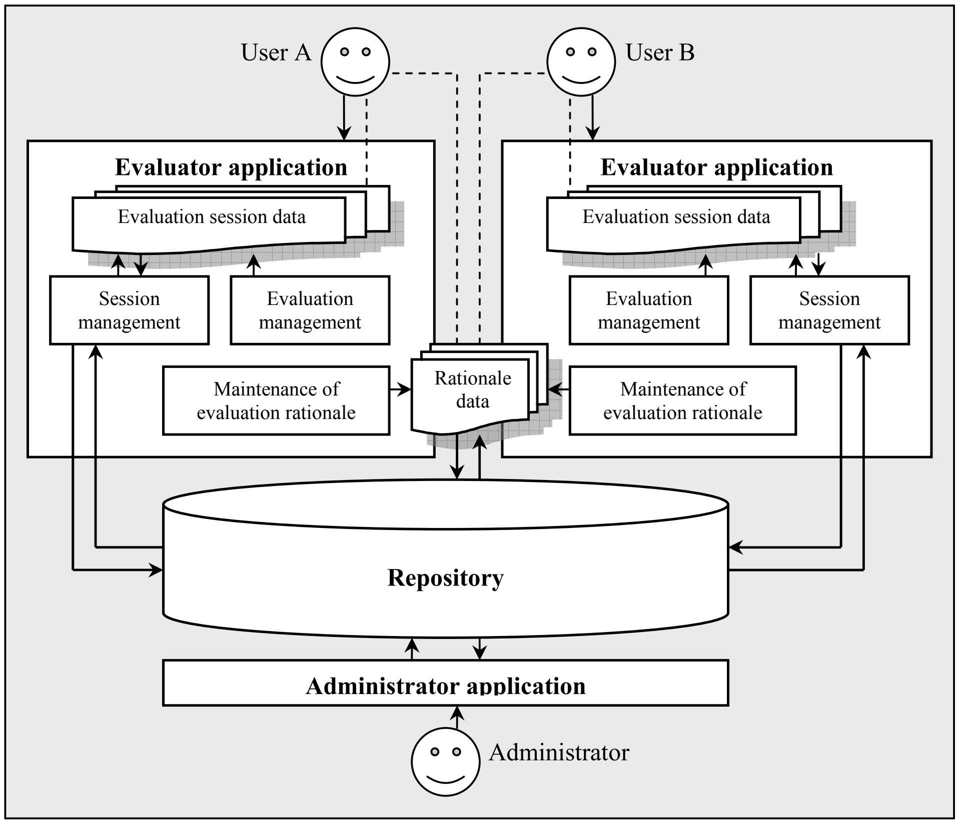
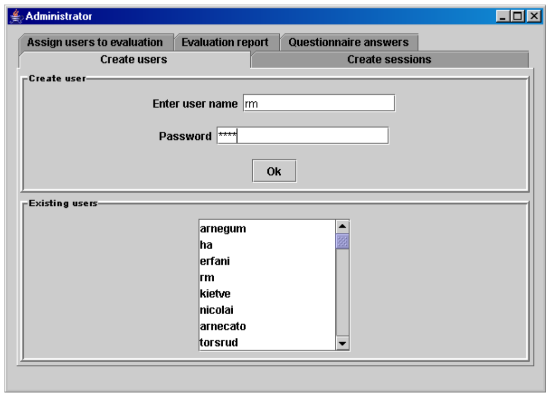


Figure 8.1 Overview of the Prototype Components
(the dotted line indicate the data, which are visible to the prototype users)

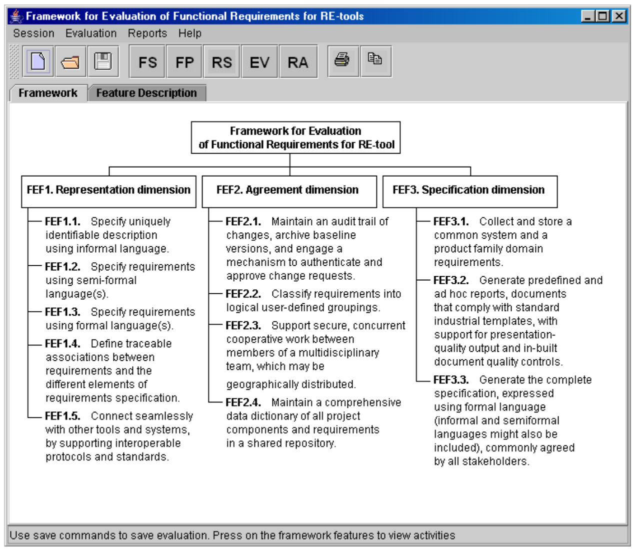


Enter user name rm

Password ****

arnegum
ha
erfani
rm
kietve
nicolai
arnecato
torsrud

Figure 8.2 Administrator Application in the Framework for Evaluation of Functional RE-tool Requirements



Framework for Evaluation of Functional Requirements for RE-tool

FEF1. Representation dimension

- **FEF1.1.** Specify uniquely identifiable description using informal language.
- **FEF1.2.** Specify requirements using semi-formal language(s).
- **FEF1.3.** Specify requirements using formal language(s).
- **FEF1.4.** Define traceable associations between requirements and the different elements of requirements specification.
- **FEF1.5.** Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.

FEF2. Agreement dimension

- **FEF2.1.** Maintain an audit trail of changes, archive baseline versions, and engage a mechanism to authenticate and approve change requests.
- **FEF2.2.** Classify requirements into logical user-defined groupings.
- **FEF2.3.** Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.
- **FEF2.4.** Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.

FEF3. Specification dimension

- **FEF3.1.** Collect and store a common system and a product family domain requirements.
- **FEF3.2.** Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.
- **FEF3.3.** Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.

Use save commands to save evaluation. Press on the framework features to view activities

Figure 8.3 Framework Application in the Framework for Evaluation of Functional RE-tool Requirements

The rest of this section focuses on the evaluator application, which supports the evaluation framework and the R-TEA method. The prototype has the following main modules: *session management* where the evaluation sessions are created, saved, and opened; *evaluation management* where RE-tool evaluation is performed; and *maintenance of evaluation rationale* where prototype users can comment any evaluation issue.

8.1.1 Session Management

The prototype for the evaluation framework of functional RE-tool requirements is based on the *evaluation session* where session is defined as evaluation of one selected set of RE-tools at a time. The session could be created in two different ways:

– The session could be created by the prototype administrator. After session creation, the administrator assigns the users who will be evaluating the selected RE-tools. The same data and comments on the evaluation issues will be visible to all the assigned users of the session.
– The session could be created by the individual user (let’s say, user A) after he or she logs in to the system. In this case the session data will be visible only to user A. User A could give a different name to the common session where he or she is working with other users. In this case the new session is also created and it will be accessible only to user A.

After login to the prototype (Figure 8.4), the user gets default session without name or other parameters. The user needs to save the default session (Figure 8.5) by giving a session name, or she or he can open an existing (Figure 8.6). The session is working with the uploaded data from the repository. In order to update the repository data the user should use “save commands” when he or she exits the prototype.

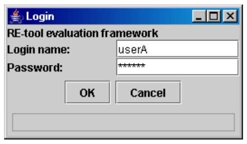

Login
RE-tool evaluation framework
Login name: userA
Password: ******

Figure 8.4 Login to the Prototype

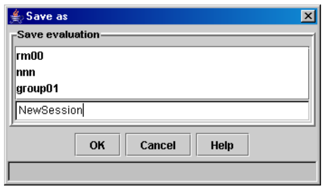

Save as
Save evaluation
rm00
nnn
group01
NewSession

Figure 8.5 Saving Evaluation Session

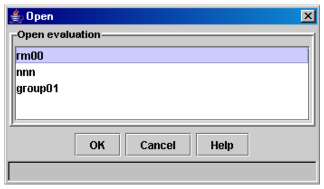

Open
Open evaluation
rm00
nnn
group01

Figure 8.6 Open Evaluation Session

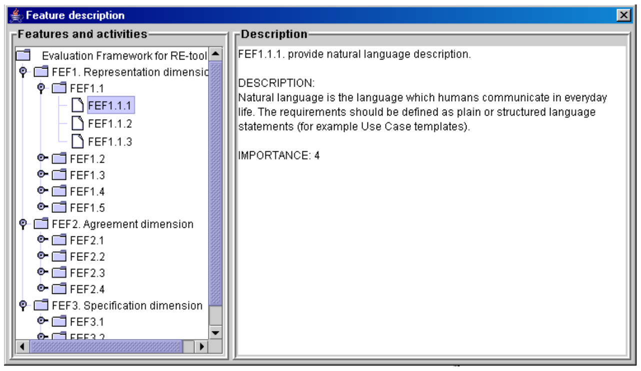
8.1.2 Evaluation Management

Evaluation of the RE-tools is performed following the R-TEA (RE-tool evaluation approach) method, defined in Chapter 6. The user can explore the framework using the "Feature description" window (Figure 8.7) whish lists all the framework activities and features and provides a comprehensive description of their meaning.

The tool evaluation process starts with the preparation of the requirements specification which is used during the tool comparison. In the prototype it means two functions 1) selection of the features and activities, and 2) prioritisation of the selected features and activities.

Figure 8.8 presents a feature selection (FS) window. The user can select/unselect the individual activities, or he or she can mark/unmark the whole feature (and its activities). The navigation between framework features is performed by choosing the appropriate tab at the top of the window.

The feature prioritisation (FP) window is shown in Figure 8.9. The prototype uses a simple prioritisation according to the activity importance. Each selected activity is given an importance value in a scale 0-to-10 where 0 means not important activity and 10 the most important activity. Later on the priorities are used to calculate the overall evaluation of the RE-tools.

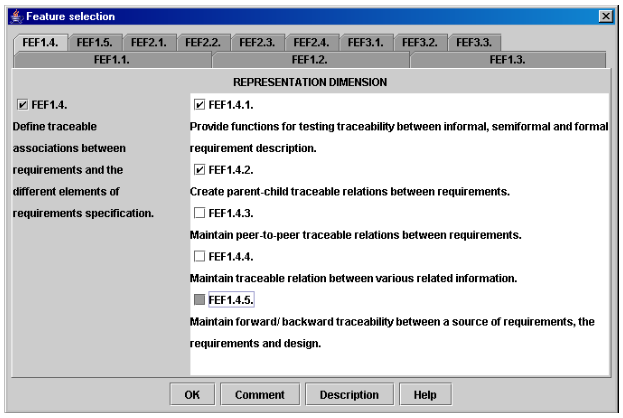


FEF1.1.1. provide natural language description.

DESCRIPTION:
Natural language is the language which humans communicate in everyday life. The requirements should be defined as plain or structured language statements (for example Use Case templates).

IMPORTANCE: 4

Figure 8.7 Feature Description Window

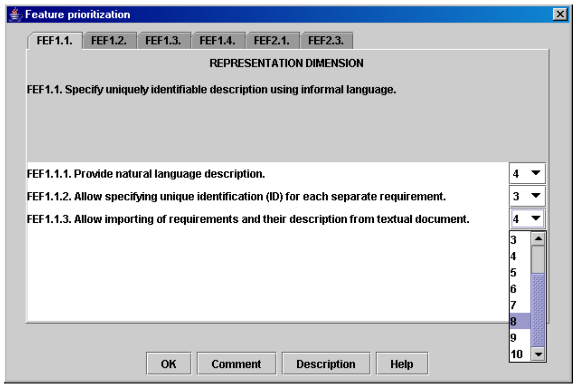

Feature selection
FEF1.4. FEF1.5. FEF2.1. FEF2.2. FEF2.3. FEF2.4. FEF3.1. FEF3.2. FEF3.3.
FEF1.1. FEF1.2. FEF1.3.
OK Comment Description Help

REPRESENTATION DIMENSION

☑ FEF1.4.
Define traceable associations between requirements and the different elements of requirements specification.

☑ FEF1.4.1.
Provide functions for testing traceability between informal, semiformal and formal requirement description.
☑ FEF1.4.2.
Create parent-child traceable relations between requirements.
☐ FEF1.4.3.
Maintain peer-to-peer traceable relations between requirements.
☐ FEF1.4.4.
Maintain traceable relation between various related information.
■ FEF1.4.5.
Maintain forward/ backward traceability between a source of requirements, the requirements and design.

Figure 8.8 Framework Feature Selection


Feature prioritization
FEF1.1. FEF1.2. FEF1.3. FEF1.4. FEF2.1. FEF2.3.
3
4
5
6
7
8
9
10
OK Comment Description Help

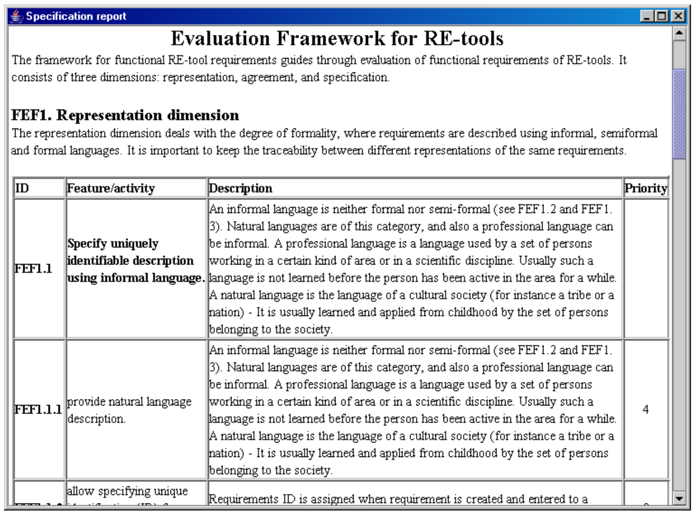
REPRESENTATION DIMENSION

FEF1.1. Specify uniquely identifiable description using informal language.

FEF1.1.1. Provide natural language description.	4
FEF1.1.2. Allow specifying unique identification (ID) for each separate requirement.	3
FEF1.1.3. Allow importing of requirements and their description from textual document.	4

Figure 8.9 Framework Feature Selection

After performing the feature selection and feature prioritisation operations it is possible to print out the requirements specification for the RE-tool acquisition (Figure 8.10). This operation is for the user convenience, if he or she wants to obtain a hard copy of the specification.



Evaluation Framework for RE-tools

The framework for functional RE-tool requirements guides through evaluation of functional requirements of RE-tools. It consists of three dimensions: representation, agreement, and specification.

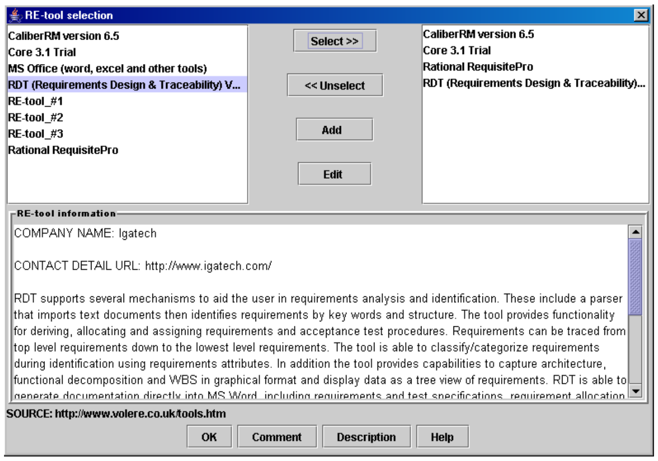
FEF1. Representation dimension

The representation dimension deals with the degree of formality, where requirements are described using informal, semiformal and formal languages. It is important to keep the traceability between different representations of the same requirements.

ID	Feature/activity	Description	Priority
FEF1.1	**Specify uniquely identifiable description using informal language.**	An informal language is neither formal nor semi-formal (see FEF1.2 and FEF1.3). Natural languages are of this category, and also a professional language can be informal. A professional language is a language used by a set of persons working in a certain kind of area or in a scientific discipline. Usually such a language is not learned before the person has been active in the area for a while. A natural language is the language of a cultural society (for instance a tribe or a nation) - It is usually learned and applied from childhood by the set of persons belonging to the society.	
FEF1.1.1	provide natural language description.	An informal language is neither formal nor semi-formal (see FEF1.2 and FEF1.3). Natural languages are of this category, and also a professional language can be informal. A professional language is a language used by a set of persons working in a certain kind of area or in a scientific discipline. Usually such a language is not learned before the person has been active in the area for a while. A natural language is the language of a cultural society (for instance a tribe or a nation) - It is usually learned and applied from childhood by the set of persons belonging to the society.	4
	allow specifying unique	Requirements ID is assigned when requirement is created and entered to a	

Figure 8.10 Requirements Specification for RE-tool Acquisition

The third step in the prototype is the RE-tool selection (RS) for evaluation (Figure 8.11). The RE-tool candidates are selected from the initial list of RE-tools. This initial list could be entered by both the prototype administrator and users according to the RE-tool surveys and reviews (e.g., Table 3.3).



COMPANY NAME: Igatech

CONTACT DETAIL URL: http://www.igatech.com/

RDT supports several mechanisms to aid the user in requirements analysis and identification. These include a parser that imports text documents then identifies requirements by key words and structure. The tool provides functionality for deriving, allocating and assigning requirements and acceptance test procedures. Requirements can be traced from top level requirements down to the lowest level requirements. The tool is able to classify/categorize requirements during identification using requirements attributes. In addition the tool provides capabilities to capture architecture, functional decomposition and WBS in graphical format and display data as a tree view of requirements. RDT is able to generate documentation directly into MS Word, including requirements and test specifications, requirement allocation

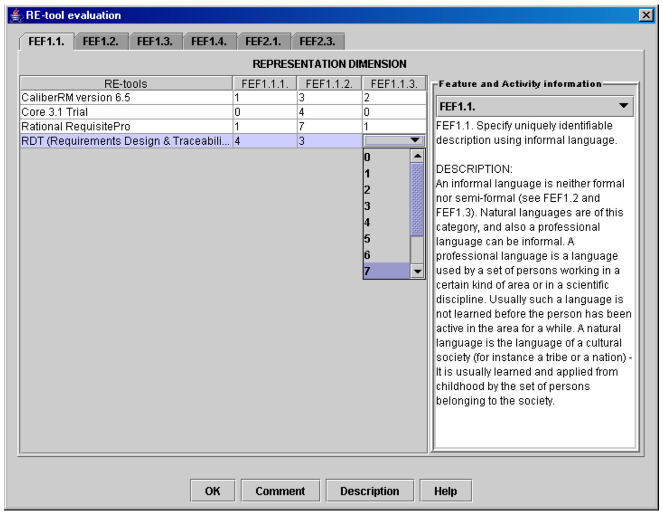
SOURCE: http://www.volere.co.uk/tools.htm

Figure 8.11 RE-tool Selection for Evaluation

RE-tool evaluation (EV) comprises the functionality assessment of the selected RE-tools candidates according to the selected framework features and activities (Figure 8.12). The tool functionality is evaluated in the scale from 0-to-10 where 0 means that the tool does not support the activity at all; 1 means that tools supports the activity very poorly, and 10 means that activity is excellently supported in the RE-tool.

The final step of the RE-tool assessment is result analysis (RA). The overall evaluation score is calculated as the sum of multiplications between feature priority (defined in phase FP) and RE-tool evaluation (defined in phase EV). The prototype also computes the not-tested feature rate, which is calculated as the percentage of activities which were evaluated as 0, and all the evaluated activities. The prototype suggests the result analysis in two levels: according to framework dimensions where the feature scores are calculated (Figure 8.13); and according to the framework features where the framework activity scores are calculated (Figure 8.14).

The final evaluation report is printed when all the users in the session finished and saved their individual evaluations (Figure 8.15). The final report is printed from the data stored in the repository and includes all evaluations performed by all users of the same session. This means that if three users were assigned to the same evaluation session by the administrator, the final report would contain all three evaluations.



REPRESENTATION DIMENSION

RE-tools	FEF1.1.1.	FEF1.1.2.	FEF1.1.3.
CaliberRM version 6.5	1	3	2
Core 3.1 Trial	0	4	0
Rational RequisitePro	1	7	1
RDT (Requirements Design & Traceabili...	4	3	

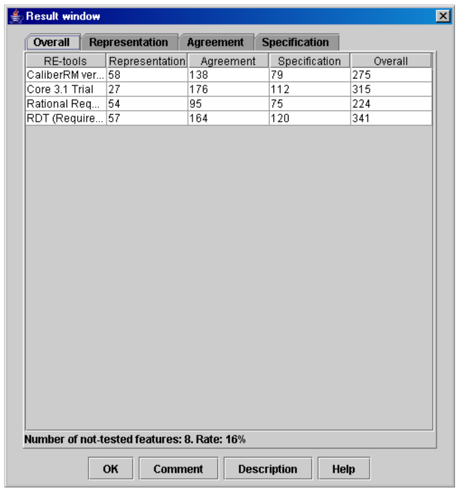
Feature and Activity information

FEF1.1.

FEF1.1. Specify uniquely identifiable description using informal language.

DESCRIPTION:
An informal language is neither formal nor semi-formal (see FEF1.2 and FEF1.3). Natural languages are of this category, and also a professional language can be informal. A professional language is a language used by a set of persons working in a certain kind of area or in a scientific discipline. Usually such a language is not learned before the person has been active in the area for a while. A natural language is the language of a cultural society (for instance a tribe or a nation) - It is usually learned and applied from childhood by the set of persons belonging to the society.

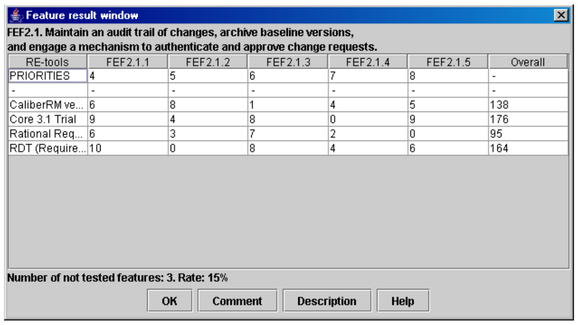
Figure 8.12 RE-tool Evaluation



RE-tools	Representation	Agreement	Specification	Overall
CaliberRM ver...	58	138	79	275
Core 3.1 Trial	27	176	112	315
Rational Req...	54	95	75	224
RDT (Require...	57	164	120	341

Number of not-tested features: 8. Rate: 16%

Figure 8.13 Result Analysis Window for Framework Dimensions



FEF2.1. Maintain an audit trail of changes, archive baseline versions, and engage a mechanism to authenticate and approve change requests.

RE-tools	FEF2.1.1	FEF2.1.2	FEF2.1.3	FEF2.1.4	FEF2.1.5	Overall
PRIORITIES	4	5	6	7	8	-
-	-	-	-	-	-	-
CaliberRM ve...	6	8	1	4	5	138
Core 3.1 Trial	9	4	8	0	9	176
Rational Req...	6	3	7	2	0	95
RDT (Require...	10	0	8	4	6	164

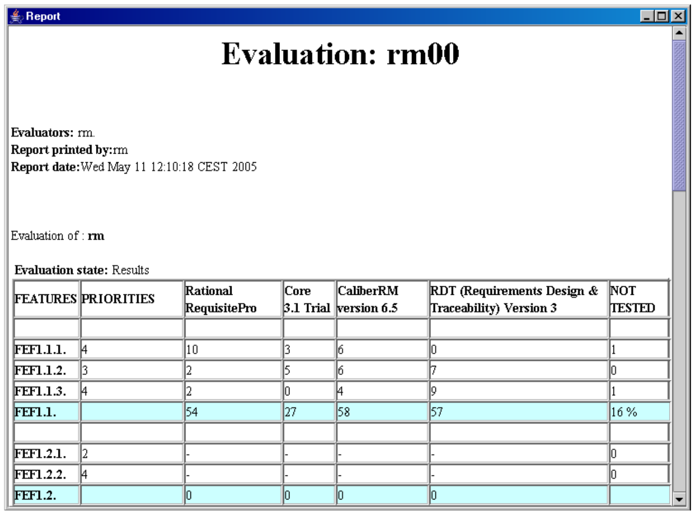
Number of not tested features: 3. Rate: 15%

Figure 8.14 Result Analysis Window for Framework Feature (FEF1.1) and Activities

8.1.3 Maintenance of Evaluation Rationale

Maintenance of evaluation rationale is ensured by the commenting functionality (Figure 8.16). The user can write comments at any evaluation moment. Each comment has two properties – “Comment type” and “Feature”. When the user calls the comment function, the prototype sets the appropriate comment type and feature according to the session step, the user is performing.

The commenting functionality gives the possibility to cooperate with other users working in the same evaluation session. All the comments written here are stored in the repository and the comments are automatically updated for all the session users. The prototype helps to discuss the evaluation issues. In this way the prototype helps to evaluate RE-tools subjectively and maintains an evaluation rationale.



Evaluation: rm00

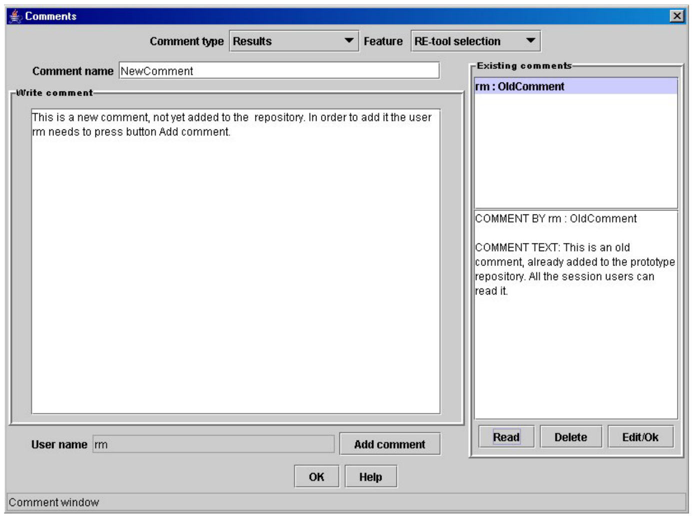
Evaluators: rm.
Report printed by:rm
Report date: Wed May 11 12:10:18 CEST 2005

Evaluation of : **rm**

Evaluation state: Results

FEATURES	PRIORITIES	Rational RequisitePro	Core 3.1 Trial	CaliberRM version 6.5	RDT (Requirements Design & Traceability) Version 3	NOT TESTED
FEF1.1.1.	4	10	3	6	0	1
FEF1.1.2.	3	2	5	6	7	0
FEF1.1.3.	4	2	0	4	9	1
FEF1.1.		54	27	58	57	16 %
FEF1.2.1.	2	-	-	-	-	0
FEF1.2.2.	4	-	-	-	-	0
FEF1.2.		0	0	0	0	

Figure 8.15 Final Evaluation Report



Comment type: Results
Feature: RE-tool selection
Comment name: NewComment

This is a new comment, not yet added to the repository. In order to add it the user rm needs to press button Add comment.

rm : OldComment

COMMENT BY rm : OldComment

COMMENT TEXT: This is an old comment, already added to the prototype repository. All the session users can read it.

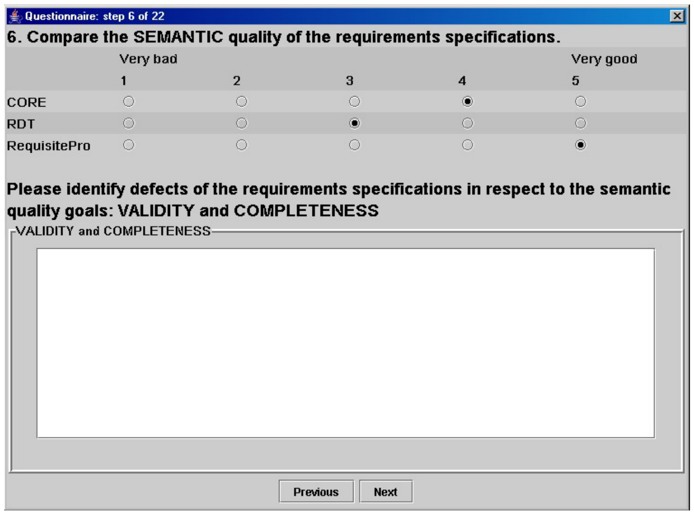
User name: rm

Comment window

Figure 8.16 Comment Window (Rationale Maintenance)

8.1.4 Questionnaire about Quality of Requirements Specification

The prototype includes a questionnaire (Figure 8.17) to evaluate the quality of requirements specifications prepared while testing different RE-tools. The specification quality evaluation is performed according to the semiotic quality framework (Krogstie 2001a). The questionnaire suggests a numerical evaluation of each quality type and a textual commenting of quality goals. The questionnaire also addresses a number of issues related to the RE-tool usage, performance of evaluation techniques, and usability, reliability, performance and maintainability of the RE-tools. The questionnaire was used in the Case study D described in Chapter 7, in order to analyse how the RE-tool evaluation influence the quality of the requirements specifications.


Questionnaire: step 6 of 22

6. Compare the SEMANTIC quality of the requirements specifications.

	Very bad 1	2	3	4	Very good 5
CORE	○	○	○	●	○
RDT	○	○	●	○	○
RequisitePro	○	○	○	○	●

Please identify defects of the requirements specifications in respect to the semantic quality goals: VALIDITY and COMPLETENESS

VALIDITY and COMPLETENESS

Figure 8.17 Questionnaire for Requirements Specification Quality

8.1.5 Prototype Improvements

The framework prototype does not support the evaluation of the non-functional RE-tool features. The possible extension is to implement a similar questionnaire as discussed in section 8.1.4. Such a questionnaire should allow selection and prioritisation of the required non-functional RE-tool features and representative questions in order to collect the user opinion about the evaluated RE-tools.

A possible improvement for the prototype is development of a knowledge base where the experiences from the RE-tool evaluations would be collected. The prototype could also be expanded with the evaluation technique used in R-TEA, description and guidelines of the technique application while assessing the selected RE-tools.

The framework prototype was applied in Case study D in Chapter 7. Further investigation of the prototype is needed in order to establish its usefulness, usability and performance. The prototype validation would be stronger if the framework was applied in industrial settings, especially in a company planning to acquire an RE-tool. The future work also involves the comparison of various frameworks and performance of similar experiments involving practitioners as research subjects.

8.2 Environment for RE-tool Weakness Consideration

The literature study in Chapter 3 and case studies in Chapter 7 identify a list of weaknesses for the commercial RE-tools to support the RE process. The major limitations are weak representation of the requirements model, weak maintenance of traceability, lack of user involvement, lack of collaboration means, weak support of requirements reuse, and lack of support for requirements specification standards. As discussed by Firesmith (2003b), a modern RE-tool should be a repository-based application (Figure 8.18). It should include the following basic properties:

User Interface. The way to enter requirements and their metadata is by using a user-friendly interface that allows one to easily enter individual requirements, their metadata, and requirements including text, diagrams, and tables. The user interface should not only support requirements input but also requirements specification and report generation including template creation, querying, and actual production. However the literature (Duitoid and Paech, 2001; Lang and Duggan, 2001; Urquhart, 2001) and a Case study A identify that RE-tools are complex for user to comprehend tool functionality. RE-tools are has too many functions which are not because of their complexity.

*Requirements Engineering Support***.** The RE-tool should be complete in that it should support all tasks of requirements engineering including business analysis, cost/benefit analysis, application visioning, and requirements elicitation, analysis, specification, and management. RE-tools should also include requirements traceability to architecture, design, implementation, and testing work products and forward and backward requirements traceability. However the commercial RE-tools do not ensure the appropriate traceability mechanism. In the best case they allow requirements representation using only informal (natural) language (but not semiformal or formal ones).

Support for Related Activities. The RE-tool should include scope control and configuration management. This includes interoperability with management, configuration management, quality engineering, modelling, and testing tools for forward and reverse engineering. Thus, an RE-tool should not be stand-alone, but a critical component of an integrated development environment. Only some of the commercial RE-tools has interface with modelling tools; however this is limited to the tools developed by the same vendor.

Team Development. RE is best performed by a cross-functional requirements team that provide an adequate experience base to capture all of the requirements and to iterate them in a timely fashion. All requirements team members should be able to work on the requirements simultaneously. Similarly, other members of the team need to have simultaneous access to the requirements for purposes of learning, evaluation, and approval. Most of the RE-tools are standalone application and do not provide any means for collaboration, cooperation and agreement. Tools also lack definition of user responsibilities or do not consider providing scenarios and guidelines for the RE activities. In the best case tools provide means for data and information sharing at the database level in the organisation's intranet.

Distributed Development. A RE-tool should support RE including specification by distributed users.

Security. Requirements for many applications involve proprietary information, trade secrets, or even national secrets. Any RE-tool should support the security of the requirements including the identification, authentication, and authorization to perform role-specific tasks of its users. It should also include privacy, integrity, and non-repudiation of requirements and their updates.

Quality Factors. An RE-tool is an application in many ways like any other. Thus, RE-tools should also have the appropriate level of non-functional RE-tool requirements (quality factors) besides security and interoperability. They include completeness, internationalization, performance, scalability, usability, and user friendliness.

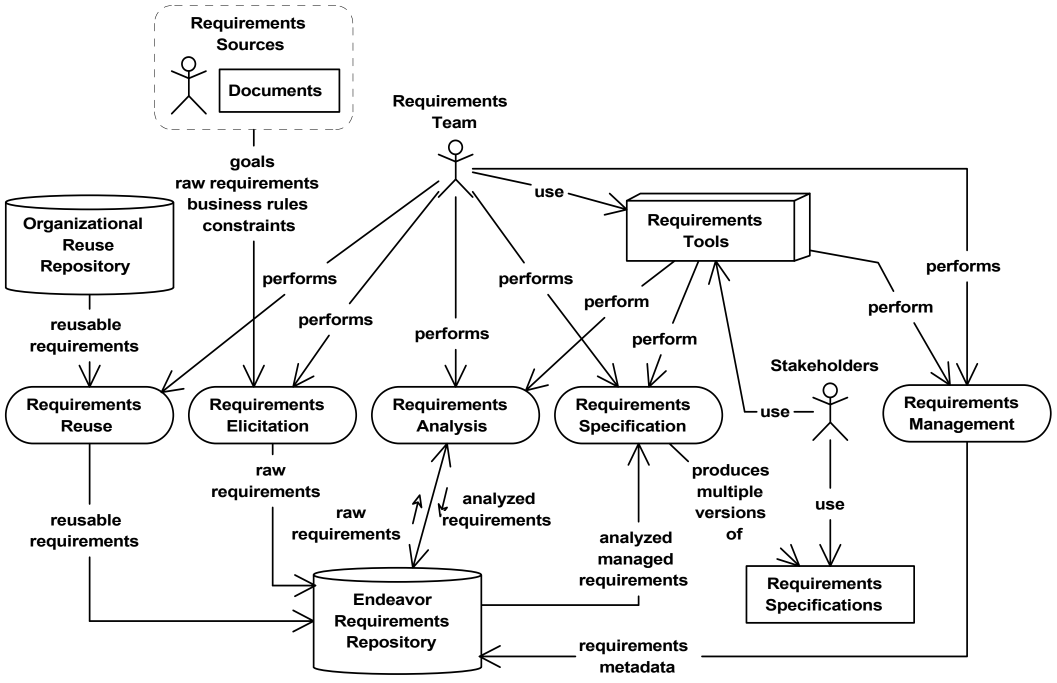


Figure 8.18 Repository-based Requirements Specification adapted from (Firesmith, 2003b)

Requirements Reuse. The RE- tools should enable the easy incorporation of existing requirements into its repository. The requirements tool needs to be able to parse the old requirements specifications, recognise potential requirements and incorporate them in a manner that will make it easy for them to be reviewed and either accepted as is, accepted with modification, or rejected. The recommendation is that all requirements would be stored as individual objects at a high-level of detail. However the market surveys and the case studies indicate that RE tools mainly support reuse using "copy-paste" functionality. RE-tools are not repository-based applications. In the best case they have databases for information storage but not for reuse activity.

This section presents a fragment-based RE-tool (FB-RET). It is a prototype (Munkelien, 2003) which provides environment to consider the weaknesses of the automated RE process support in a predefined architectural kernel. The prototype targets a requirements specification which is constructed from information fragments. A fragment is defined as an information unit here. The purpose of FB-RET is not to create a new RE-tool, but rather to provide an experimental environment to consider weaknesses of RE-tool by implementing them in this environment.

The main elements of the prototype are shown in Figure 8.19. They are management level, repository level, user identification level and user interface level. Management level provides means, tools and toolsets for managing artefacts, specifying new fragments, and modifying existing ones. Repository level is responsible of storing and organising fragments in a proper way, User identification level manages access of different user groups to the prototype. Finally user interface provides a graphical environment to the user in order to login to the prototype and use different functions.

In the following the main modules of the FB-RET prototype are analysed.

8.2.1 Management Level

The purpose of the *management level* is to provide tools, toolsets and environment to define, store, select, and combine fragments. The main requirements for the functionality of FB-RET include:

Req.1 The user should be able to create a new master product.

- ***Req.1.1*** The user should be able to enter the master product name.
- ***Req.1.2*** The user should be able to confirm the master product name.
- ***Req.1.3*** The new master product should be created in the database.
- ***Req.1.4*** The user should be able to enter new master product information in a window.

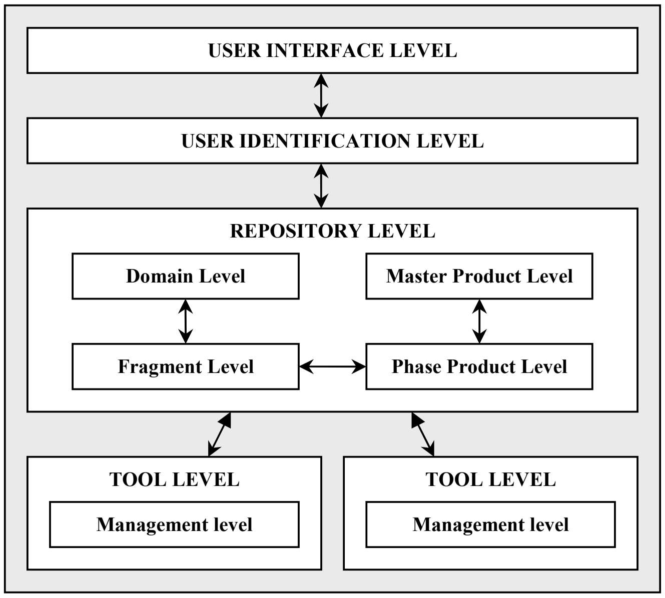


Figure 8.19 FB-RET architecture

Req.2 The user should be able to open a master product.

- ***Req.2.1*** The user should be able to choose a master product.
- ***Req.2.2*** The user should be able to choose new master product information in a window.

Req.3 The user should be able to create a new phase product.

- ***Req.3.1*** The user should be able to choose a template.
- ***Req.3.2*** The user should be able to enter phase product name.
- ***Req.3.3*** The new phase product should be created in the database.
- ***Req.3.4*** The user should be able to enter new phase product information in a window.

Req.4 The user should be able to open a phase product.

- ***Req.4.1*** The user should be able to choose a phase product.
- ***Req.4.2*** The user should be able to enter new phase product information in a window.

Req.5 The user should be able to close a phase product

- ***Req.5.1*** The user should be able to confirm saving phase product.

Req.6 The user should be able to create a new fragment.

- ***Req.5.1*** The user should be able to enter fragment name.
- ***Req.5.2*** The new fragment should be created in the database.
- ***Req.5.3*** A reference to the fragment should be created in database.
- ***Req.5.4*** The user should be able to enter new fragment information in a window.

Req.7 The user should be able to upload a fragment.

- ***Req.7.1*** The user should be able to select a domain.
- ***Req.7.2*** The user should be able to select a fragment.
- ***Req.7.3*** The system should display the reference to the fragment.
- ***Req.7.4*** The user should be able to get references of the related fragments.

Req.8 The user should be able to edit a fragment.

- ***Req.8.1*** The user should be able to choose a fragment.
- ***Req.8.2*** The user should be able to enter new fragment information in a window.

Req.9 The user should be able to remove a fragment name.

- ***Req.9.1*** The user should be able to choose a fragment.
- ***Req.9.2*** The user should be able to confirm deleting fragment.
- ***Req.9.3*** The reference of the removed fragment should be deleted from the database.

Req.10 The user should be able to create a new domain.

- ***Req.10.1*** The user should be able to enter domain name.
- ***Req.10.2*** The new domain should be created in the database.
- ***Req.10.3*** The user should be able to enter new domain information in a window.

Req.11 The user should be able to generate a report.

- ***Req.11.1*** The user should be able to choose a master product.
- ***Req.11.2*** The system should generate the report according to the template and display it.

Req.12 The user should be able to get help from the system.

Figure 8.20 depicts the main design class diagram. *MainWindow* takes care of the input from the user and distributes information and tasks to other classes.

The output to the user will also be taken care of in MainWindow class. The window which gives representation of the user interface level, is created when the application starts. In addition the class creates singular instances of *DatabaseManager* and *MasterProduct*. *DatabaseManager* manages interfaces between user interface level and repository level, and between repository level and tool (management) level. A singular object is created in class MainWindow, and the class connects the application to the database. *MasterProduct* keeps track of the master product and phase product which are opened at the time. A singular object of this class is created in class MainWindow when the application starts. *FragmentArray* contains an array that manages all fragments created in the application. *AbstractFragment* is an abstract class of a fragment. All classes that inherit from *AbstractFragment* are elements in the *FragmentArray*. The subclasses of *AbstractFragment* include all possible fragments created in the application (*AbstractUpperClass, DomainFragment,* *PhaseProductFragment, MasterProductFragment, TemplateFragment, AbstractFragmentData,* and *UrlFragment*).

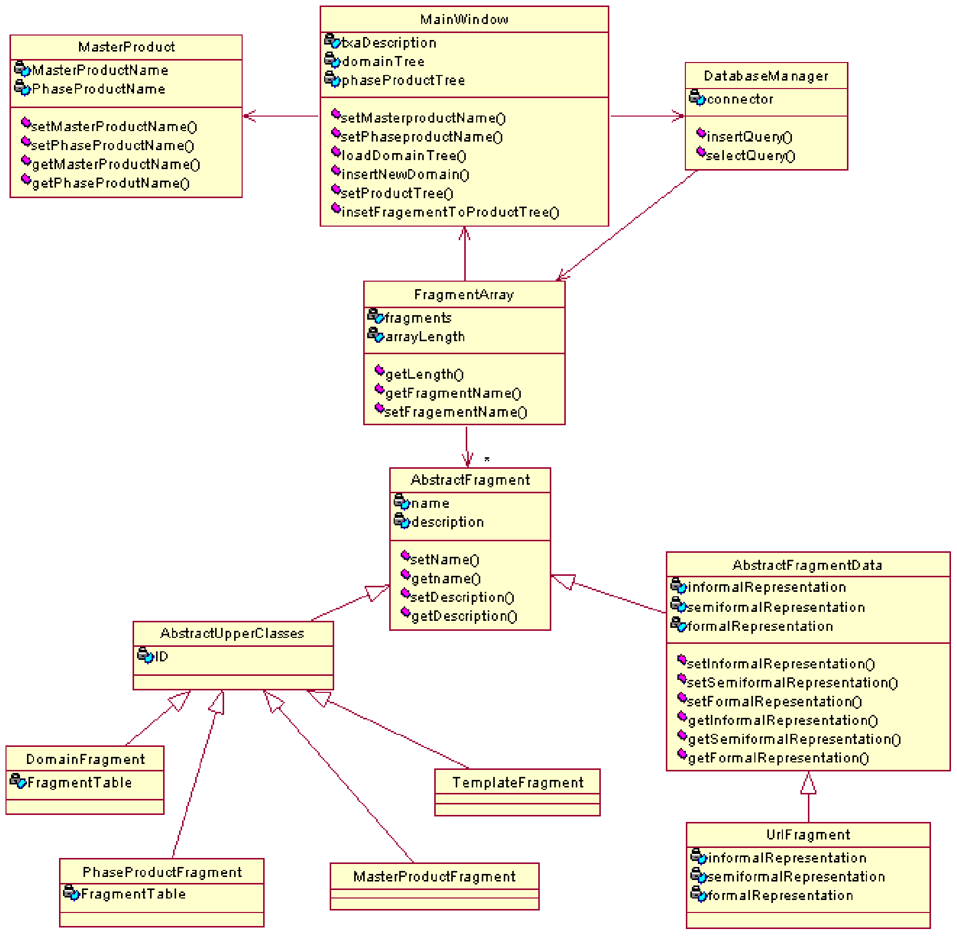


Figure 8.20 FB-RET Class Diagram

8.2.2 Repository Level

The *repository level* consists of the domain level, fragment level, phase product level, and master product level (Figure 8.21). In the repository level the information about the requirements model is stored.

In the *domain level* the predefined fragments which correspond to the different application domains and groups, are stored (tables DomainName and DomainTable). Examples of application domains are: financial, insurance, electricity, non-functional, and functional. The domain level contains reusable fragments for the particular domain. The domain level could also be used for separate predefined requirements groups (for example security requirements, non-functional requirements, and marketing requirements).

Fragment level. Fragment could be represented as triple of three representation forms: informal representation, semi-formal representation, and formal representation (table FragmentStructure). On another hand the fragment should be possible to represent using any modelling language or to link it to the model created with other tools (e.g. table FragmentStructureTextField for textual requirements representation).

Phase product level is the level, where the final fragment as the product is constructed from existing fragments in the domain and from new prepared fragments, using other specifying and modelling tools (table PhaseProductTable). The purpose of the prototype is to support construction of the requirements specification. As the template for the phase product the IEEE standard 830-1998 could be used (tables PhaseProductTemplate and PhaseProductTemplateStructure).

Master product level has double characteristics. First, it could server as the repository for finished phase products and it could provide reusable fragments for new fragment construction (tables MasterProductTable and MasterProductName). Second, it could be use for different phase products (for example pre-study documentation, requirements specification, and design specification) during the same project.

8.2.3 User Identification Level

The *user identification level* provides functionality for creating user accounts and user groups. User groups describe different stakeholders of the system, and user accounts describe individual participants of the software project. Stakeholder groups should represent different stakeholders. The user identification level should also provide functionality to define the different information access functionality and allow definition of information access viewpoint. For example, it is preferable to maintain end users access to informal and semiformal system model representation but to keep him away from formal representation.
For users with financial interest it would be preferable to access information and reports, where system budget is negotiated. For users responsible for system development it would be preferable to access information related to system analysis, modelling (design) and implementation.

Finally, the system should have one special group of users, called administrators, which should be able to create new user logins, user groups, manage user access to information and coordinate the information flows in a database. The user identification level is not implemented in the FB-BRET prototype.

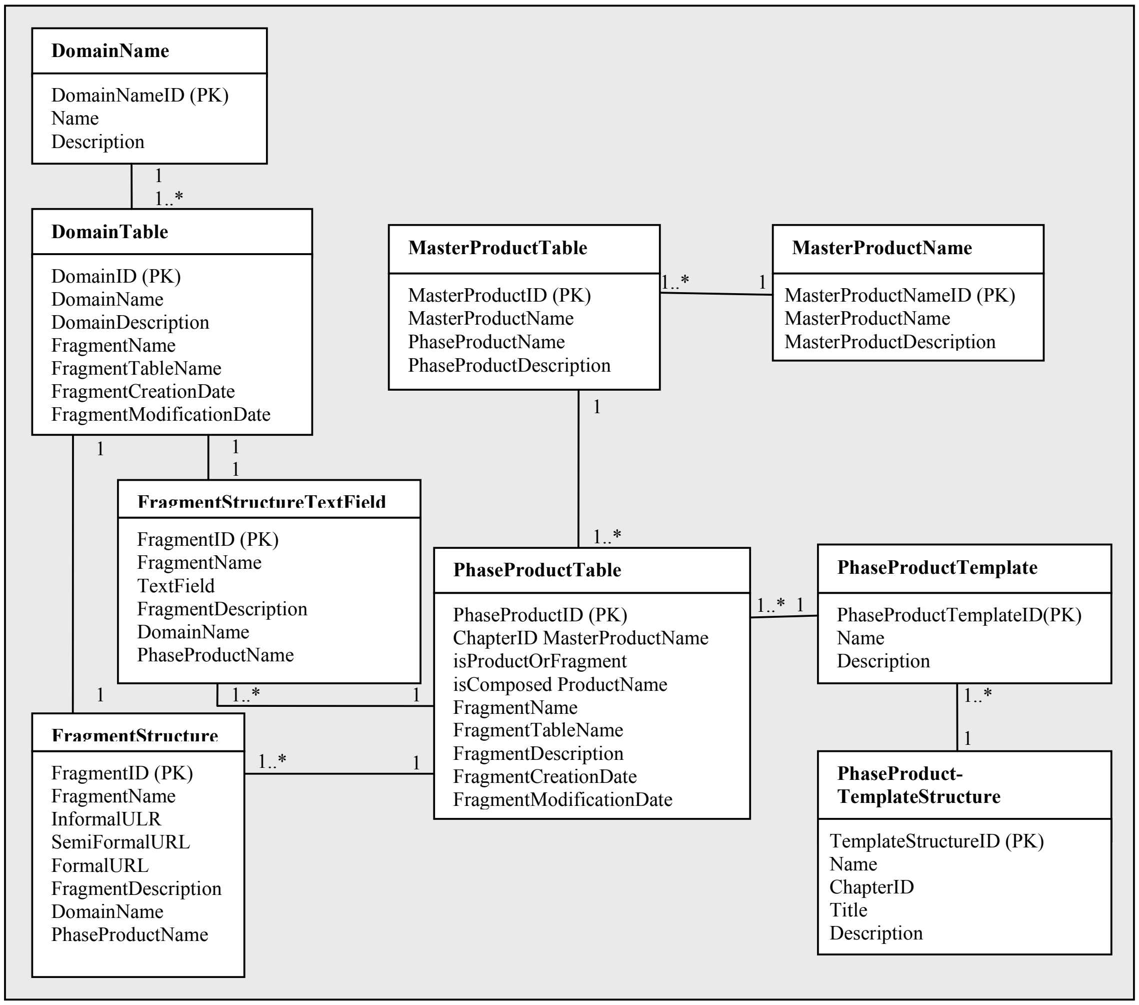


Figure 8.21 FB-RET Repository

8.2.4 Representative Example

In this section the representative example of FB-RET usage is demonstrated. The example follows the evaluation frameworks and the specification exemplar, analysed in Chapters 5 and 6. The same problem is used here as in Chapter 6 example to show the application of the specification exemplar. Let's say, it is important that an RE-tool would maintain the object oriented and structural modelling perspectives and allow preparation of the semi-formal requirements model.

After starting the FB-RET application (Figure 8.22), first, the master product (Figure 8.23) and phase product (Figure 8.24) should be defined. In this example the user-defined template for the requirements specification in chosen. This means that the FR-RET user has to specify the chapters and subchapters of the requirements specification.

All the elements in the FB-RET are considered as fragments, but the application separates between fragments which specify the chapter titles, subtitles and the fragment which contain description of the certain developed system elements. The separation is done by the attribute 'Representation' during the definition of the fragment. In Figure 8.25 new fragments are created for the chapters functional and non-functional requirements (chapterID: FEF0 and NF0), and for the semiformal requirements representation and representation using the modelling perspective

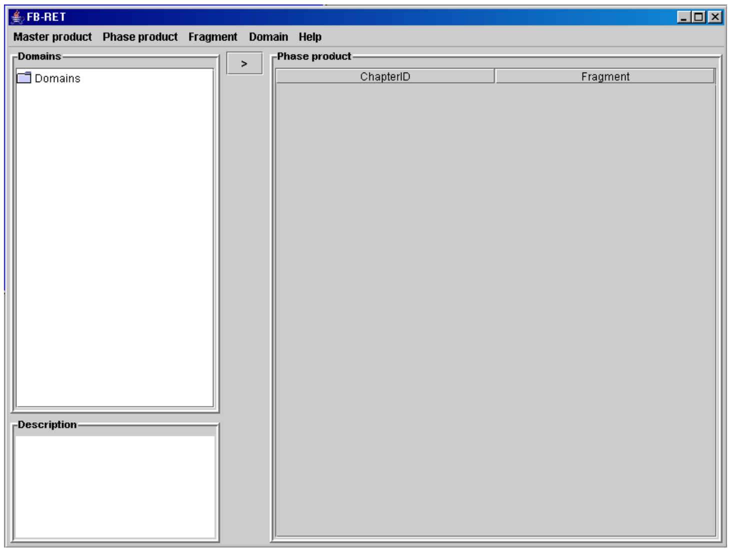
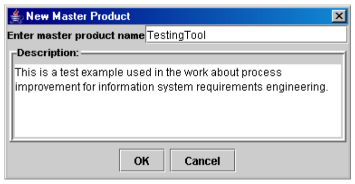


Figure 8.22 FB-RET Main Window



Enter master product name TestingTool

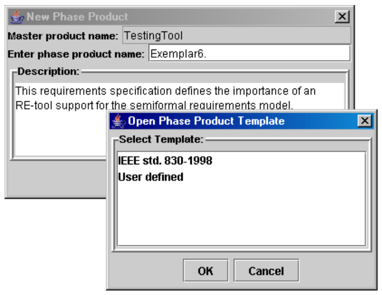
Description:

This is a test example used in the work about process improvement for information system requirements engineering.

Figure 8.23 Creation of New Master Product in FB-RET

(chapterID: FEF1.2 and NF1.2). ChapterID here specifies the fragment (or requirements) identification number.

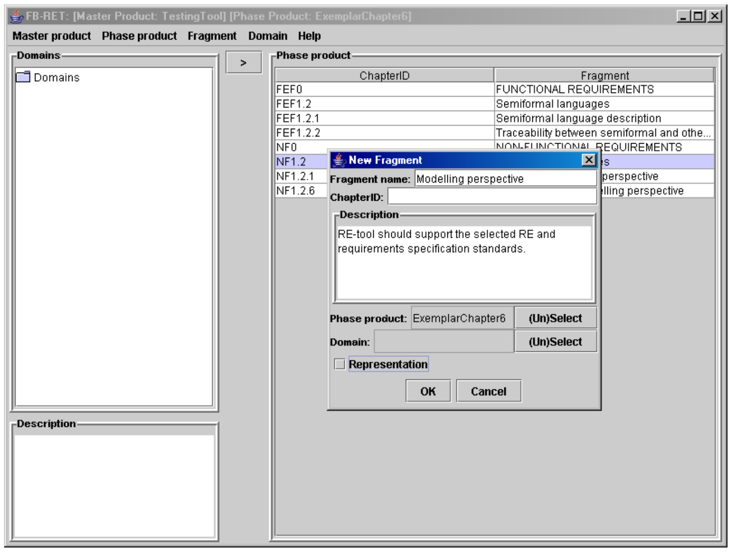
Next, the fragments which carry information about the concrete requirements are defined from the domain levels, which group the requirements to certain categories. The requirements are uploaded from the domains to the requirements specification by defining the appropriate chapterID number and the information (Figure 8.26). The uploaded fragments are FEF1.2.1 and FEF1.2.2 (to specify RE-tool functionality) and NF1.2.1 and NF1.2.6 (to describe working process requirements). Finally, after the creation of the requirements model the requirements specification is printed as shown in Figure 8.27.



Master product name: TestingTool
Enter phase product name: Exemplar6.
Description:
This requirements specification defines the importance of an RE-tool support for the semiformal requirements model.

Select Template:
IEEE std. 830-1998
User defined

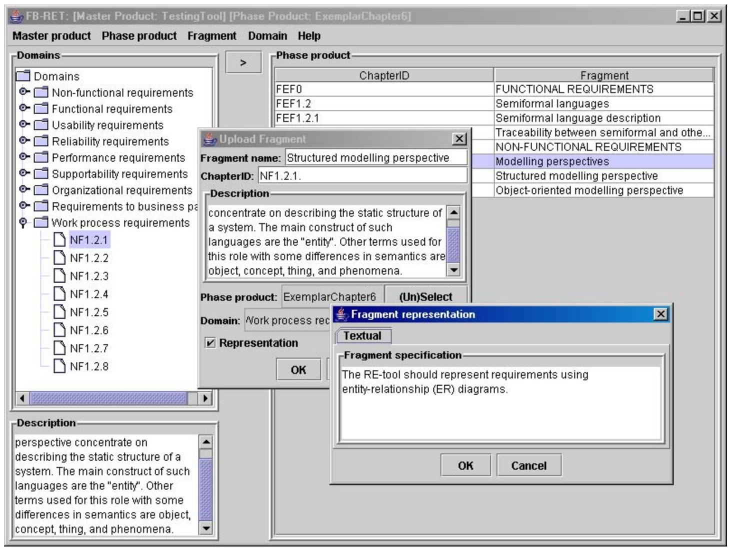
Figure 8.24 Creation of New Phase Product in FB-RET



ChapterID	Fragment
FEF0	FUNCTIONAL REQUIREMENTS
FEF1.2	Semiformal languages
FEF1.2.1	Semiformal language description
FEF1.2.2	Traceability between semiformal and othe...
NF0	NON-FUNCTIONAL REQUIREMENTS
NF1.2	s
NF1.2.1	perspective
NF1.2.6	lling perspective

Fragment name: Modelling perspective
ChapterID:
Description:
RE-tool should support the selected RE and requirements specification standards.
Phase product: ExemplarChapter6
Domain:

Figure 8.25 Definition of New Fragments in FB-RET



Domains
- Domains
  - Non-functional requirements
  - Functional requirements
  - Usability requirements
  - Reliability requirements
  - Performance requirements
  - Supportability requirements
  - Organizational requirements
  - Requirements to business pa
  - Work process requirements
    - NF1.2.1
    - NF1.2.2
    - NF1.2.3
    - NF1.2.4
    - NF1.2.5
    - NF1.2.6
    - NF1.2.7
    - NF1.2.8

ChapterID	Fragment
FEF0	FUNCTIONAL REQUIREMENTS
FEF1.2	Semiformal languages
FEF1.2.1	Semiformal language description
	Traceability between semiformal and othe...
	NON-FUNCTIONAL REQUIREMENTS
	Modelling perspectives
	Structured modelling perspective
	Object-oriented modelling perspective

Fragment name: Structured modelling perspective

ChapterID: NF1.2.1.

Description: concentrate on describing the static structure of a system. The main construct of such languages are the "entity". Other terms used for this role with some differences in semantics are object, concept, thing, and phenomena.

Phase product: ExemplarChapter6

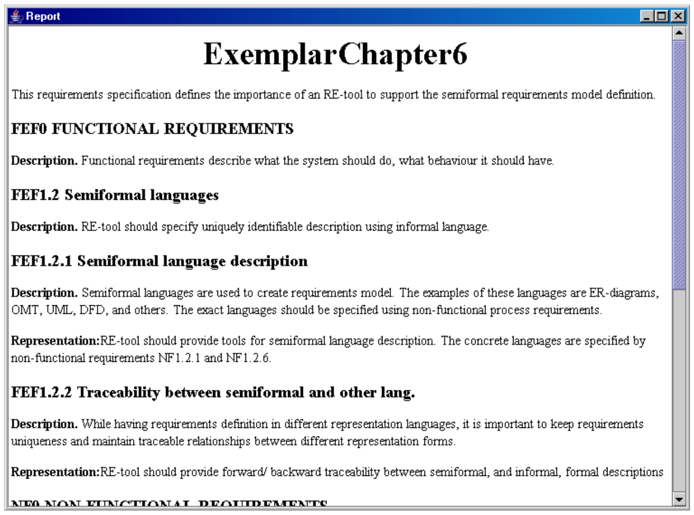
Domain: Work process rec

☑ Representation

Fragment specification: The RE-tool should represent requirements using entity-relationship (ER) diagrams.

Description: perspective concentrate on describing the static structure of a system. The main construct of such languages are the "entity". Other terms used for this role with some differences in semantics are object, concept, thing, and phenomena.

Figure 8.26 Fragment Upload from Domain in FB-RET



ExemplarChapter6

This requirements specification defines the importance of an RE-tool to support the semiformal requirements model definition.

FEF0 FUNCTIONAL REQUIREMENTS

Description. Functional requirements describe what the system should do, what behaviour it should have.

FEF1.2 Semiformal languages

Description. RE-tool should specify uniquely identifiable description using informal language.

FEF1.2.1 Semiformal language description

Description. Semiformal languages are used to create requirements model. The examples of these languages are ER-diagrams, OMT, UML, DFD, and others. The exact languages should be specified using non-functional process requirements.

Representation:RE-tool should provide tools for semiformal language description. The concrete languages are specified by non-functional requirements NF1.2.1 and NF1.2.6.

FEF1.2.2 Traceability between semiformal and other lang.

Description. While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceable relationships between different representation forms.

Representation:RE-tool should provide forward/ backward traceability between semiformal, and informal, formal descriptions

NF0 NON-FUNCTIONAL REQUIREMENTS

Figure 8.27 Requirements Specification in FB-RET

8.2.5 Evaluation and Extensibility

The purpose of the FB-RET prototype is not a complete RE-tool, but rather a testable-programmable environment which would emphasise the investigation of the RE-tool weaknesses by implementing new functionality, adapting RE methods and facilitating analysis of existing RE techniques. Currently the FB-RET system is only a prototype containing the main functionality to prepare a textual report about the requirements model. In this section the FB-RET prototype is compared against the evaluation framework for the functional RE-tool requirements described in the Chapter 5. The comparison highlights the FB-RET extensibility possibilities.

FEF1.1: The FB-RET prototype represents requirements only in natural languages. However it gives the possibility to define certain requirements properties (like chapterID, fragment name, description and others). The possible extensibility could be the implementation of the broader requirement (fragment) properties lists which could define the requirements using for example use case techniques. The FB-RET prototype does not maintain the import of requirements from the textual (or any other) documents.

FEF1.2 and **FEF1.3:** The FB-RET prototype does not have any possibilities to represent requirements model in semiformal or formal languages. On another hand the prototype shows the possibility to extend the requirements model creation using any modelling languages. Each element in FB-RET is described as the fragment hierarchy (Figure 8.20). In order to create new a representation element, the new class for the fragment should be implemented and placed in the fragment hierarchy. In order to support the newly implemented fragment class, the interface user level should be extended with additional window(s) and the repository level should be extended with additional database tables (e.g., FragmentStructure and FragmentStructureTextField in Figure 8.21).

FEF1.4: Traceability between requirements and fragments is a research field in FB-RET. Currently the prototype allows limited functionality to define parent-child traceable relationships by setting chapterID in a hierarchical order. However this is a manual activity. The prototype does not have any other possibilities to maintain any traceability. The fragment specification should be extended with functionality to define peer-to-peer, parent-child relationships, to trace among different requirements models and across the phase products of the same master product.

FEF1.5: The FB-RET prototype does not have any import/export possibilities of the requirements model. The possible extension is the engineering of wizards supporting data import/export from/to textual and graphical documents (of various formats).

FEF2.1: This feature is not supported in the FB-RET prototype although the Figure 8.19 proposes the user identification level. The implementation of feature in the prototype requires extension of the database structure, requirements definition of different user groups their functionality and usability. Another extension point is functionality definition for tracking of requirements and requirements model history.

FEF2.2: The FB-RET prototype does not have any means for requirements filtering, sorting classifying or prioritising. On another hand it includes a wide range of improvement possibilities by expanding the requirement (fragment) property and attribute list and implementing various requirements prioritisation methods.

FEF2.3: The prototype is a repository-based JAVA application. It is possible to configure the application so that a number of users from different geographical locations would access the database. On another hand the FB-RET prototype does not have any cooperation or collaboration means, it does not maintain requirements rationale. The changes performed on the requirements model does not reflect on the interface views of the other users. In addition the possible extensions include the repository management activities (e.g., data replication).

FEF2.4: The current version of FB-RET includes the list of documents (Munkelien, 2003), which help to understand its background (pre-study), requirements (requirement specification), design and implementation (design and implementation documents) and main functionality (user manual). The documentation is useful while developing and expanding the prototype, but not guiding the different groups of users through the RE process. The possible extensions include preparation of help facilities both for the skilled developers and non-skilled users. The user support also expands when implementing the prototype extensions.

FEF3.1: The FB-RET prototype is based on the repository which helps to store fragments. It facilitates the requirements reuse, as fragments could be reused two conceptual sources: from different domains and from already approved phase and master products. The defined phase product templates facilitate creation of the phase documents, correspond to the standards (e.g. IEEE 830-1998 standard) and allow reuse of phase document structure. The extensions of FEF3.1 include the improvement of already implemented functionality for repository management, functionality to spread requirements changes through the domains and products, as well as selection and extraction of common phase product requirements, and comparison of requirements feasibility.

FEF3.2 and **FEF3.3:** Both these features are supported poorly in the FB-RET prototype. The prototype creates only one report as the phase product (requirements specification). The phase product could be created according to predefined (organisational and international) standards. However the extensions of these features include implementation of report generation wizards, functionality to print out a number of process reports and documents. In addition all the other above described prototype extensions should be followed with the appropriate reports, too.

The described evaluation and extensions are summarised in Table 8.1. In addition, the FB-RET prototype and all its improvements should be validated with respect to usability, correctness and performance.

Table 8.1 Evaluation and Extensions of the FB-RET Prototype

Feature	Evaluation	Extensions
FEF1.1	Represent requirements (fragment) using natural language; Manually maintains requirements (fragment) ID (chapterID).	Check ID for uniqueness; Requirements representation using different informal techniques; Requirement import from textual documents.
FEF1.2 and FEF1.3	No means to define semiformal or formal requirements representations.	Requirements representation means to create semiformal and formal requirements views.
FEF1.4	Manual maintenance of requirements hierarchy; Mental relationships between requirements representations.	All possible types of requirements traceability (e.g., peer-to-peer, parent-child, traceability between requirements representations, traceability between phase products, etc)
FEF1.5	No import/export functionality.	Semi-automated import/export wizards.
FEF2.1	Design issues for user identification level; No means to support requirements history.	User identification level; Different information access for various user groups Requirements and requirements history maintenance. Registration of requirements discussions, rationale, elicitation information.

FEF2.2	No requirements classification, sort, filter or prioritisation functionality.	Requirements property lists;
		Sorting, filtering functionality;
		Prioritisation methods.

FEF2.3	Repository-based application; Data access from different locations.	Repository management functions; Collaboration, cooperation, rationale registration functionality.
FEF2.4	Documentation for prototype requirements, design and implementation.	Guidance of different groups of users (technical and non-technical); Help system(s), tutorials, and demonstrations.
FEF3.1	Repository-based prototype; Reuse of predefined fragments from domain and phase/master products; Reuse of phase product templates.	Improvement of reuse functionality; Functionality for spreading requirements across products; Means to adapt requirements changes to domain; Means to compare requirements feasibility.
FEF3.2 and FEF3.3	One report about the phase product.	Process reports; Reports on additional functionality; Wizards to generate reports.

8.3 Chapter Summary

In this chapter two prototype tools are presented. The first prototype supports the guidelines of the RE-tool evaluation approach and facilitates the application of the evaluation framework for the functional RE-tool requirements. The main functionality of the framework prototype and the improvement possibilities which include the extension of non-functional RE-tool feature evaluation, implementation and maintenance of the RE-tool assessment knowledge base for reuse, are surveyed.

Although the prototype is applied in Case study D, its usability, performance and reliability should be investigated in future case studies.

The second prototype describes the fragment-based RE-tool (FB-RET). The prototype requirements, architecture, and design, and a representative example of prototype usage, are described. Finally the functionality of the FB-RET prototype is evaluated by the means of the evaluation framework for the RE-tool requirements and highlights the extensibility points. The FB-RET prototype stands as the kernel environment for the investigation of the RE-tool weaknesses.

PART IV

CONCLUSIONS AND FUTURE WORK

In the previous part of this work the RE-tool selection methods are considered. The general belief is that a qualitative methodology would yield a selection of the proper RE-tool(s) which could improve quality of the RE process and product. In order to facilitate the evaluation of the RE-tools two evaluation frameworks are developed. Next they are adapted to the specification exemplar which initiates an RE-tool evaluation approach (R-TEA). The R-TEA method is analytically compared against the other existing general tool selection approaches (Matulevičius and Sindre, 2005). The comparison concludes that R-TEA could be faster applied and executed because it directly targets the RE-tool domain.

Several case studies examine the performance, usability and reliability of the proposed evaluation frameworks and approach in Chapter 7. In Chapter 8 two prototype tools are presented. The first prototype facilitates the application of the evaluation framework for the functional RE-tool requirements and maintains guidelines of the R-TEA method. The second prototype presents a design-implementation environment for analysis of the RE-tool weaknesses.

The four part of this work consists of one chapter called *Conclusions and Future Work*. The summary of the work highlights the conclusions to the research questions and research problem. Next, the theoretical, methodological and practical contributions of this work are discussed. Finally the chapter concludes with the limitations and proposals for the future work.

CHAPTER 9

Conclusions and Future Work

In this final chapter, first, the conclusions to the research problem and the research questions are discussed. Further the main contributions of the work are presented Next the chapter states the contribution. Finally the limitations of this study and recommendations for further research conclude the chapter.

9.1 Conclusions on the Research Questions

Requirements engineering (RE) is considered to be one of the most important stages in the process of system and software development process. Errors occurring during the RE process could be very expensive in the later development stages as well as during the actual use and maintenance of the system. Here the RE process support and improvement while developing information systems is analysed. Kotonya and Sommerville (1998) identify three possible objectives of process improvement, e.g., quality improvement, schedule reduction and resource reduction. Potential areas for improvement of RE processes are, for example, developing of new RE methods, searching for better methods and techniques for training users, supporting maintenance of the software products, reusing of methodology and techniques across different systems, and working out techniques for interviewing and negotiation. As the projects grow during the development life cycle it gets very important to search for the means to control the complexity and the size of the developed system. Here the emphasis is on the acquisition of targeted RE-tools. RE-tools facilitate the RE process support and improvement by enacting means to perform RE activities faster, to apply new RE methods, and to reuse the RE artefacts. They suggest means to improve the quality of requirements specifications and to reduce the amount of resources and time needed for the RE process. The main research problem that is addressed in this work, as stated in Chapter 1 is as follows:

Research Problem: *How to improve the RE process by acquiring RE-tools?*

The research problem is specialised into two research questions. In order to provide answer to the research problem first the conclusions to each of these two questions are presented.

RQ.1: *What software tools provide better means to support the RE process and to maintain high-quality requirements artefacts?*

The first research question considers what the automated support possibilities are for the RE process. The RE process could be executed using different software tools. Office and modelling tools could be applied to produce and maintain requirements specifications. On the other hand the market offers a number of targeted RE-tools which support not only requirements representation activities, but also allow requirements traceability, emphasise requirements reuse and provide means to produce more complete requirements specifications.

Two evaluation frameworks which are based on existing theory as found in the literature are proposed in this work. The frameworks are presented as a skeleton structure which suggests RE-tool features in order to initiate the evaluation activities before selecting the RE-tools. The frameworks are applied in an empirical survey and several case studies.

The findings of the survey report that the industrial practice relies on office and modelling tools. However the RE process support is not adequate and has many problems. The requirements are usually represented informally and only in some cases semi-formally. But formal requirements representation is not considered. The requirements traceability is difficult to maintain, as the relationships must be defined manually. The RE process involves many different tools, causing interoperability problems. It is difficult to maintain the requirements history, rationale and collaboration means. Finally the practice in best cases relies on organisational requirements specification standards. The reuse of requirements is complicated in the best case as “copy-paste” operations.

Case study B compares two RE processes and the quality of two requirements specifications. The findings of the case study report that RE-tools support the RE process better, and yield requirements specifications of substantially higher quality.

Case study D considers the subjective aspects of the automated RE process support. The participants of the case study indicated that RE-tools provide better understandability and support of the RE process, the RE-tools contain better means to manage and ensure quality of the requirements specification. The satisfaction of the tool usage is slightly higher for the office tools, but satisfaction directly depends on the tool understandability, usability and functionality, and it could also be explained by previous familiarity with office tools.

Based on the presented results it is possible to answer the research questions that the targeted RE-tool provide better means to ensure the quality of both RE process and requirements specifications. However the investigation also revealed a number of RE-tool weaknesses. In order to suggest environment for the analysis of the RE-tool limitations, a prototype tool is presented. The prototype facilitates the analysis of RE-tool features and the RE-tool requirements by using implementation activities in proposed environment. Hopefully, the prototype will contribute to a better automated support for the RE process methods developed in a future research.

RQ.2: *How to evaluate and acquire software tools for the RE process support according to organisational needs so that they could lead to improvement of the RE process?*

The second research question considers what the methodology is applicable to the RE-tools assessment. The literature suggests a number of methodological guidelines on how to evaluate software tools before starting using them. The well known methods include PORE (Maiden and Ncube, 1998) OTSO (Kontio, 1996), CAP (Ochs *et al.*, 2000), scenario-based COTS selection (Feblowitz and Greenspan, 1998), STACE (Kunda and Brooks, 1999; Kunda, 2003), and quality-based method (Franch and Carvallo, 2002). All these methods emphasise analysis of COTS (commercial-off-the-shelf) products, but none of them directly focus on the RE-tool selection domain. Therefore the evaluation criteria, measures, and application are time consuming and demand domain knowledge.

The general quality evaluation frameworks (e.g. NATURE framework (Pohl, 1994; Pohl, 1996), semiotic quality framework (Krogstie 1998; Krogstie 2001a; Krogstie and Jørgensen, 2003) are either too abstract for the RE-tool assessment. Others are created for other tool domains (Post and Kagan, 2000; Sedigh-Ali, Ghafoor and Paul, 2001; Nikiforova and Sukovskis, 2002).

The number of RE-tool requirements frameworks (INCOSE framework; Haywood and Dart, 1997; Hoffmann *et al.*, 2004) are, however, development-oriented and they do not contain guidelines for their application and usage. Therefore the assessment of the RE-tools is again complex and time consuming.

This work has proposed an RE-tool evaluation approach (R-TEA) which uses an specification exemplar created according to two RE-tool evaluation frameworks. The specification exemplar enables the evaluation process by involving the tool users in the tool selection. The users have to externalise their needs by prioritising RE-tool requirements and producing a requirements specification for the RE-tool acquisition. The evaluation process comprises phases for the RE-tool candidate selection, evaluation using different techniques. At the end the users have to make a decision:

- to select the best evaluated RE-tool(s) without changes of work processes;
- to select the best evaluated RE-tool(s) and to adapt work process to the RE-tool(s);
- to repeat the evaluation itself if the RE-tool(s) is not found.

In order to facilitate usage of the method a prototype tool is presented. The prototype supports the evaluation framework for functional RE-tool requirements and provides guidelines of framework application according to the R-TEA method.

The R-TEA method is analytically compared with other evaluation approaches. The comparison shows that the R-TEA method could reduce evaluation time and this provides cheaper assessment of the RE-tools. The R-TEA method targets the RE-tool domain and guides the users through RE-tool assessment by combining both evaluation frameworks and their application guidelines.

The R-TEA method is applied in several case studies. Case study A presents the framework application to the environment. Case study C applies different evaluation techniques in two RE-tool evaluation sessions. The results report on a time consuming application of the evaluation framework in comparison to questionnaire-based evaluation. But the framework provided a detailed evaluation which would be more preferred when making a real case RE-tool acquisition. The findings are supported by Lauesen (2002) who states that while less detail domain-level requirements might be appropriate for the selection of business COTS, more detailed product-level requirements are appropriate for technical COTS of which RE-tools are an example. Case studies C and D consider the usefulness of different evaluation techniques. The case studies compared performance of the evaluation scenarios and RE-tool tutorials and concluded that combination of both techniques contributes to a more mature assessment of the RE-tools.

Finally, Case study D investigates performance and correctness of the evaluation framework for functional RE-tool requirements and the R-TEA method. The case study addresses the question whether the method contributes with an RE-tool which helps to prepare and maintain a high-quality requirements specification. The case study concludes with a positive intension to towards performance of the R-TEA method to select the proper RE-tool(s).

Based on the presented findings the answer to the research question is to apply the R-TEA method which yield the selection of the targeted RE-tools to support the RE process and to maintain the quality of the RE artefacts.

Taken together the answers to these two research questions constitute the main contributions of this work. So, returning to the original question on how to improve the RE process, the work proposes to acquire and to use the targeted RE-tools to support and maintain the quality of the RE process and requirement specifications. In many cases some initial work has to be done before acquiring an RE-tool. For example the organisation should assess its maturity in order to determine the capability to execute the RE process. RE-tool acquisition could be a logical step for the RE process improvement. A successful acquisition and appropriate usage of the RE-tool(s) could reduce resources and time to produce requirements specifications. RE-tool(s) could also improve the quality of both the RE process and requirements specification.

An RE-tool assessment and acquisition is a social activity which should be based on a constructivistic world view. The process should involve both potential tool users and organisation's management and the evaluation process should include the learning activities. The conclusion on the research problem is that the targeted RE-tool could lead to RE process improvement when the organisation is ready for the changes of the work processes and have a fairly long time perspective for their investment.

9.2 Contribution

The objective of this work was to develop an RE-tool acquisition method, which would help to elicit the environment needs and RE-tool requirements and to adapt the RE-tool(s) in an inexpensive way and short amount of time. Based on this objective and answers provided to the research questions, the work contributes with theoretical, methodological and practical knowledge.

Theoretical contribution: The work presents several theoretical contributions:

– *Survey and comparison of the RE practice in different geographical areas*. Based on the literature study and the empirical survey in Chapter 3 the work describes an existing situation of the RE practice and in such a way provides theoretical background for improving the RE process and its activities. Besides providing a theoretical basis for answering the second research question *RQ.2*, the contribution addresses the first research question RQ.1.
– *Comparison of the RE-tool evaluation frameworks*. The analytical overview and comparison of the RE-tool frameworks in section 4.2 contributes to the understanding of the RE-tool requirements and criteria needed to assess these tools.
– *Comparison of the general tool evaluation approaches*. The survey of the general tool evaluation approaches in section 4.3 highlights the basic guidelines and steps of the tool acquisition. They stand as the theoretical contribution of the tool selection domain.

Analysis and comparison of both RE-tool evaluation frameworks and general tool evaluation approaches could also be useful for practitioners to have an overview of existing tool evaluation approaches. Such an overview could also be of use to researchers working on improved tools, or improved ways to evaluate tools.

Methodological contribution: The main methodological contributions are:

– *Framework for comparison of the tool evaluation methodologies*. Based on the literature study the work contributes with a framework for evaluation of different evaluation approaches for the RE-tool assessment. The framework in section 6.1 suggests methodological guidelines which are applied while comparing the

proposed approaches in section 6.3. The contribution leads to the answer of the second research question *RQ.2*.
– *Guidelines for combining evaluation frameworks and approaches for the RE-tool assessment*. The work proposes a combination between both RE-tool evaluation frameworks and the general evaluation approaches. Such a composition could contribute with a qualitative methodology which would save time, highlight the evaluation criteria and guide through the RE-tool evaluation process.

Both methodological contributions suggest means to perform similar comparisons, surveys and investigation of new evaluation approaches and the evaluation frameworks when they are created and engineered.

Practical contribution: Based on the results the work has made several contributions for practice. They are:
– *Two frameworks for comparison and evaluation of RE-tools*. Two evaluation frameworks for RE-tools are suggested in Chapter 5. Both frameworks and relationships between them provide evaluation criteria for assessing RE-tools and contribute to the answer of research question *RQ.1*.
– *A specification exemplar for initiating the RE-tool evaluation*. The specification exemplar is constructed according to the evaluation frameworks. It contributes to the initial stage of the RE-tool evaluation process. It could also serve as an evaluation technique which allows comparison of the tools according to the same guidelines and criteria.
– *RE-tool evaluation approach*. Chapter 6 contributes with an RE-tool evaluation approach (R-TEA) which is a proposed answer to the second research question *RQ.2*. The R-TEA method directly targets the RE-tool selection domain and allows saving time in comparison to the other approaches which describe the general tool evaluation.
– *Prototype tool for facilitating the RE-tool evaluation* contributes with support for the evaluation activities according to the functional framework for the RE-tool requirements and for the R-TEA method guidelines. The additional questionnaire and functionality includes extension possibilities for the non-functional framework, as well as for improving the prototype usability and performance.
– *Prototype tool for investigation of the automated support for the RE process*. Throughout the work many weak aspects of the commercially available RE-tools are discovered and discussed. The prototype described in section 8.2 provides a design and implementation environment to analyse these weaknesses.

The major contribution of this work includes *the increased understanding for the RE process support and improvement objectives, targets and means showing how they* *could be achieved by the means of the successfully acquired RE-tool(s) to the organisational environment.*

9.3 Limitations

The investigation reported here is not without limitations. All the investigation is carried out in an academic environment, but not in the industrial organisation which would have a purpose to acquire an RE-tool to support the RE process. The survey results (Matulevičius, 2004a) and the literature analysis in Chapter 3 indicate that many organisations are not mature enough to adapt the RE-tools. On the other hand a multiple attempts were made to obtain an industry case study. There were several discussions with different organisations in Lithuania, Norway, and Sweden suggesting them to use the proposed evaluation frameworks and the R-TEA approach to assess the RE-tools. The organisations responded with the scepticism towards the costs of RE-tool acquisition and usage. It is possible to state that the major problem to conduct a case study in industrial organisation included the accessibility to willing organisations.

The case studies included the testing of the performance and correctness of the proposed contributions (frameworks, R-TEA and prototypes). Some work is also performed to assess the usability and application of the RE-tool evaluation framework and the prototype tools. However, none of them had a goal to select the RE-tool for usage. This made a limitation to evaluate all the features and activities of the evaluation frameworks, especially for the non-functional RE-tool requirements, in the empirical studies.

The work has not performed empirical investigations comparing several different RE-tool evaluation frameworks or evaluation approaches. In order to mitigate this limitation the extensive analytical survey and comparison are provided in Chapters 4 and 6. The analytical comparison also provides an overview of the case studies performed with different general evaluation approaches. In such a way the work provides the design settings for the future case studies comparing different tool evaluation approaches.

Finally, the work needs further investigation of usability, performance, and reliability of the prototypes presented in Chapter 8. Some testing of the framework prototype is performed in Case study D. However the validity of both prototypes is limited and none of them have been applied outside the experimental settings.

9.4 Future Work

The results of this study, its contributions and limitations point to several possibilities for further research:

- *Industrial RE-tool acquisition.* In order to further validate the proposed evaluation frameworks one should carry out the RE-tool evaluation and acquisition process in an industrial organisation. Such an activity should also be supported by the framework prototype application where prototype usability, performance and reliability would be tested by the industry persons.

- *Specific environmental needs*. This work surveys the existing development models, such as waterfall, spiral, transformational and others. The R-TEA method does not target any particular development model but suggests the process requirements which describe the RE process used in an organisation. The future investigation could involve the theoretical and practical analysis of the specific RE processes and how they could be automated using the RE-tools.
- *Framework comparison*. The future work suggests a case study where different evaluation frameworks would be compared against the functional and non-functional frameworks for the RE-tool evaluation proposed in this work. The case study should highlight when to use different frameworks, which frameworks provide better usability, reliability and performance, which frameworks better separate between the RE-tool features and make better selection according to the environment needs.
- *R-TEA comparison with the general evaluation approaches*. The case study settings presented in this work suggests an experiment where general tool selection approaches would be compared against the RE-tool evaluation approach. Such an experiment would clarify the advantages and disadvantages of various tool selection approaches and would provide knowledge when, where and under which organisational requirements it is recommended to use a certain evaluation approach.
- *Extension of the framework prototype*. The further investigation involves the research on the framework prototype usability, performance and reliability. It includes the framework prototype extension with additional functionality. First, the framework prototype should be expanded with the possibilities to maintain the non-functional feature evaluation. Second, extensions include the implementation of the repository for knowledge reuse both for the same organisation and for the multi-organisational purposes. The future work could also involve the adaptation of the prototype to the organisational maturity assessment model.
- *Investigate new methods for tool-based RE process improvement.* The platform for such investigation is suggested in the fragment-based RE-tool (FB-RET) prototype. The possible extensions are considered in section 8.2.5 and they include
  - Semiformal and formal requirements representation;
  - Maintenance of requirements traceability;
  - Means for requirements cooperation, communication, and agreement;
  - Requirements prioritisation;
  - Users support at different level of abstraction;
  - Requirements repositories;
  - Templates for requirements specification;

There are certainly other directions for the future research. However, the value of any such future work depends on the specific targets and goals of each particular investigation.

9.5 Final Remarks

There is a number of RE process support and improvement methods, such as better communication maintenance, development of new methods, and user training methodology. The RE-tool evaluation approach presented in this work helps to evaluate the RE-tools before acquisition for the organisation's needs to support the RE process. The method reduces the tool assessment costs as it targets the RE-tool domain and makes focus on the RE-tool compatibility with user requirements.

Finally, in support to the constructivist standpoint it is not possible to claim "Here is the way for the successful RE-tool selection". Each organisation has to assess their own capability of tool usage, maturity of process, social and financial factors. Nevertheless the results of this investigation can provide valuable inputs to organisation specific processes while selecting the RE-tool and evaluating the maturity of the RE-tool usage.

Bibliography

Achour C. B., & Ncube C. (2000). Engineering the PORE Method for COTS Selection and Implementation with the MAP Process Meta-Model. *Proceeding of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (*REFSQ 2000*). Stockholm, Sweden.

Alexander I. F., & Stevens R. (2002). *Writing Better Requirements*. Addison-Wesley.

Basili V. R. (1993), The Experience Factory and its Relationship to Other Improvement Paradigms. Sommerville I., & Paul M. (eds.). *Software Engineering - ESEC '93, 4th European Software Engineering Conference*, Garmisch-Partenkirchen, Germany, Springer-Verlag 68-83.

Basili V. R. & Boehm B. (2001). COTS-based Systems Top 10. *IEEE Computer*, 34 (5) 91-93.

Basili V. R. & Caldiera G. (1995). Improve Software Quality by Reusing Knowledge and Experience. *Sloan Management Review*, 1 (37), 55-64.

Basili V. R. & Rombach H. D. (1991). Support for Comprehensive Reuse. *IEEE Software Engineering Journal*, 5 (6), 303-316.

Bate *et al.*, (1995). *A System Engineering Capability Maturity Model, version 1.1*. Technical Report CMU/SEI-95-MM-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Batini C., Ceri S., & Navathe S. B. (1992). *Conceptual Database Design. An Entity-Relationship Approach*. Benjamin Cummings, Redwood City, California.

Becker J., Rosemann M., & von Uthmann C. (2000). Guidelines of Business Modelling. *Business Process Management: Models, Techniques and Empirical Studies*, Springer-Verlag, 30-49.

Beecham S., Hall T., & Rainer A., (2004). Developing a RE Process Improvement Model. *Proceedings of the industrial experience track of the European Software Process Improvement Conference (EuroSPI 2004)*, Trondheim, Norway.

Benediktsson O., Dalcher D., & Thorbergsson H. (2004). Choosing a Development Life Cycle: Comparing Project and Product Measures. *Proceedings of the 17th International Conference Software and Systems Engineering and their Applications* (*ICSSEA 2004*), Paris, France.

Bertrand P., Darimont R., Delor E., Massonet P., & van Lamsweerde A. (1998). GRAIL/KAOS: an Environment for Goal Driven Requirements Engineering. *Proceedings of the 20th International Conference on Software Engineering* (*ICSE'98*). IEEE-ACM, Kyoto.

Boehm B. (1999). Managing Software Productivity and Reuse. *IEEE Computer*, 9 (32), 111-113.

Boehm B., Brown J. R., Kaspar J. R., Lipow M., MacLoed G. J., & Merritt M. J. (1978). Characteristics of Software Quality. *TRW Series of Software Technology*, Amsterdam.

Botella P., Burgués X., Carvallo J. P., Franch X., & Quer, C. (2002). Using Quality Models for Assessing COTS Selection. Lopez O. P., & Diaz J. S. (eds.). *Proceedings of the 5th Workshop on Requirements Engineering (WER 2002)*. Valencia, Spain, 263-278.

Carvallo J. P., Franch X., & Quer C. (2004a). A Quality Model for Requirements Management Tools. *Requirements Engineering for Sociotechnical Systems*, Idea Group Publishing.

Carvallo J. P., Franch X., Grau G., & Quer C. (2004b). QM: A Tool Building Software Quality Models. *Proceedings of the 12th IEEE International Requirements Engineering Conference* (*RE'04*). Kyoto, Japan, 358-359.

Chatzoglou P. (1997). Factors Affecting Completion of the Requirements Capture Stage of Projects with Different Characteristics. *Information and Software Technology*, 39, 627-640.

Chen P. P. (1976). The Entity-Relationship Model: Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 9-36.

Chung L., Nixon B. A., Yu E., & Mylopoulos J. (2000). *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers.

Cooper J., & Fisher M. (2002). *Software Acquisition Capability Maturity Model (SA-CMM), version 1.03.* Technical Report CMU/SEI-2002-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Costello R. J., & Liu D. B. (1995). Metrics for Requirements Engineering. *System Software*, 29, 39-63.

Curtis B., Krasner H., & Iscoe N., (1998). A Field Study of the Software Design Process for Large Systems. *Communication of the ACM*, 31 (11), 1268-1287.

Damian D. E., Eberlein A., Shaw M. L. G., & Gaines B. R. (2003). An Exploratory Study of facilitation in Distributed Requirements Engineering. *Requirements Engineering*, 8, 23-41.

Davis A. M. (1988). A Comparison of Techniques for the Specification of External System Behaviour. *Communication of ACM*, 31 (9), 1098-1115.

Davis A. M. (1993). *Software Requirements. Objects, Functions and States*. Prentice Hall.

Davies I., Green P., & Rosemann, M. (2003). Modelling in the Australian Practice – Preliminary Insights. *Information Age - Feb/March 2003 issue,* IDG Communications, St Leonards.

Davis A. M, Overmeyer S., Jordan K., Caruso, Dandashi F., Dinh A., Kincaid A., Ledeboer G., Reynolds G., Sitaram P., Ta P., & Theofanos A. (1993). Identifying and Measuring Quality in a Software Requirements Specification. *Proceedings of the First International Software Metrics Symposium*, 141-152.

Deming W. E. (1986). *Out of the Crisis*. The MIT Press.

Denning P. J. (2000). Computer Science: The Discipline. Ralston A., Reilly E. D. & Hemmendinger D (eds.) *Encyclopedia of Computer Science*, the 4th edition, Wiley.

Dietz J. L. G., & Widdershoven G. A. M. (1992). A Comparison of the Linguistic Theories of Searle and Habermas as a Basis for Communication Support Systems. van Riet R. P., & Meersman R. A. (eds.) *Linguistic Instruments in Knowledge Engineering*, Elsevier, 121-130.

Dillman D. A. (2000). *Mail and Internet Surveys: The Tailored Design Method*. John Wiley & Sons.

Dubois E., Du Bois P., & Petit M. (1993). Eliciting and Formalizing Requirements for C.I.M. Information Systems. Rolland C., Bodart F. & Cauvet C. (eds.) *Proceedings of the 5th International Conference on Advanced Information Systems Engineering* (*CAiSE'93*), 253-274.

Duitoit A. H, & Paech B. (2001). Developing Guidance and Tool Support for Rationale-based Use Case Specification. *Proceedings of the International Workshop on Requirements Engineering: Foundation for Software Quality* (*REFSQ'2001*). Interlaken, Switzerland.

El Emam K., & Madhavji N. H. (1995). A Field Study of Requirements Engineering Practice in Information Systems Development. *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering* (*RE'95*), 68-80.

El Emam K., Drouin J-N. & Melo W. (1998). *SPICE. The Theory and Practice and Software Process Improvement and Capability Determination*. IEEE Computer Society.

Ekremsvik S., & Tiset E. M., (2003). *Kravspek-verktøy, marketstudiet.* TDT4730 Information System Specification Report, IDI-NTNU, Trondheim, Norway.

Feather M. S., Fickas S., Finkelstein A., & van Lamsweerde (1997). Requirements and Specification Exemplars. *Automated Software Engineering*, 4, 419-438.

Feblowitz M. D., & Greenspan S. J. (1998). Scenario-Based Analysis of COTS Acquisition Impacts. *Requirements Engineering*, 3, 182-201.

Fenton N. E., & Pfleeger S. L. (1997). *Software Metric. A Rigorous and Practical Approach*, PWS Publishing Company.

Ferdinandi, P. L. (2002). *A Requirements Pattern. Succeeding in the Internet Economy.* Addison-Wesley.

Finkelstein A., Spanoudakis G., & Ryan M. (1996). Software Package Requirements and Procurement. *Proceedings of the 8th International Workshop Software Specification and Design*. IEEE Comp. Soc. Press, USA, 141-145.

Firesmith D. (2003a). Common Concepts Underlying Safety, Security, and Survivability Engineering. *Technical Note CMU/SEI-2003-TN-033*.

Firesmith D. (2003b). Modern Requirement Specification. *Journal of Object Technology*, 2(1), 53-64.

Franch X., & Carvallo I. (2002). A Quality-model-based Approach for Describing and Evaluating Software Packages. *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02).* Essen, Germany, 104-111.

Gane C., & Sarson T. (1979). *Structured System Analysis: Tools and Techniques.* Prentice-Hall.

Goowin N. C. (1987). Functionality and Usability. *Communications of the ACM*, 30 (3), 229-233.

Grau G., Carvallo J. P., Franch X., & Quer C. (2004). DesCOTS: A Software System for Selecting COTS Components. *Proceedings of the 30th EUROMICRO Conference*, IEEE**,** 118-126.

Halvorsen C. P., & Conradi R. (2001). A Taxonomy to Compare SPI Frameworks. *Software Quality Professional*, 4(3), Journal of American Association for Quality, 46-59.

Hammer M., & Champy J. (1995). *Reengineering the Corporation: a Manifesto for Business Revolution*. Nicholas Brealey, London.

Hands K., Peiris D. R., & Gregor P. (2004). Development of a Computer-Based Interviewing Tool to Enhance the Requirements Gathering Process, *Requirements Engineering,* 9, 204-216.

Harrison W., Ossher H., & Tarr P. (2000). Software Engineering Tools and Environments: A Roadmap. A. Finkelstein (Ed.) *The Future of Software Engineering*. ACM Press.

Hatley D. J., & Pirbhai I. A. (1998). *Strategies for Real-Time System Specification*. Dorset House, New York.

Haywood E., & Dart P. (1997). Towards Requirements for Requirements Modelling Tools. *Proceedings of the 2nd Australian Workshop on Requirements Engineering, Australia*, 61-69.

Hirschheim R. A., & Klein H. K. (1989). Four Paradigms of Information Systems Development. *Communications of the ACM*, 32 (10), 1199-1219.

Hofmann H. F., & Lehner F. (2001). Requirements Engineering as a Success Factor in Software Projects. *IEEE Software*, 58-66.

Hoffmann M., Kuhn N., Weber M., & Bittner M. (2004). Requirements for Requirements Management Tools. *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'04).* Kyoto, Japan, 301-308.

Humphrey W. S. (1989). *Managing the Software Process*. Addison-Wesley.

IEEE (1998) *IEEE Recommended Practice for Software Requirements Specification*, IEEE Std 830-1993 (Revision of IEEE Std 830-1998).

INCOSE (2002). INCOSE: Tools Survey: Requirements Management (RM) Tools by International Council on Systems Engineering (INCOSE) [online]. Available: http://www.incose.org/.

ISO/IEC (1991). Information Technology – Software Product Evaluation– Quality Characteristics and Guide Lines for their Use. *ISO/IEC IS 9126*, Switzerland.

James L. (1996). What's Wrong with Requirements Management Tools? *Requirements Engineering*, 1, 190-194.

Kaindl H., Brinkkemper S., Bubenko jr. J., Farbey B., Greenspan S. J., Heitmeyer C. L., do Prado Leite J. C. S., Mead N. R., Mylopoulos J., & Siddiqi J. (2002). Requirements Engineering and Technology Transfer: Obstacles, Incentives, and Improvement Agenda. *Requirements Engineering*, 7 (3), 113-123.

Kaindl H. (2004). Active Tool Support for Requirements Engineering Through RETH. *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE 2004)*, 262-263.

Kaindl H. & Mannion M. (2005). Verification of Selection from Product Line Requirements. *Proceedings of the INCOSE 2005 International Symposium*.

Karlsson L., Dahlstedt A. G., Natt och Dag, J., Regnell B., & Persson, A. (2002). Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study. *Proceedings of the 8th International Workshop on Requirements Engineering – Foundation for Software Quality* (*REFSQ'02*), German.

Karlsson L., Berander P., Regnell B., & Wohlin C. (2004). Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparison versus Planning Game Partitioning, *Proceedings of the Empirical Assessment in Software Engineering* (*EASE 2004*), Scotland.

Kelly S., Lyytinen K., & Rossi M. (1996). MetaEdit+ a Fully Configurable Multi-user and Multi-tool CASE and CAME Environment. Constantopoulos P., Mylopoulos J., & Vassiliou Y., (eds.). *Advances in Information System Engineering, 8th International Conference* (*CAiSE'96*), Heraklion, Crete, Greece. Springer LNCS, 1-21.

Khwaja A. R., & Urban J. E. (2002). A Synthesis of Evaluation Criteria for Software Specifications and Specification Techniques. *Journal of Software Engineering and Knowledge Engineering*, 15 (5), 581-599.

Kontio J. (1996). A Case Study in Applying a Systematic Method for COTS Selection. *Proceedings of the 18th International Conference on Software Engineering* (*ICSE'96*), IEEE.

Kotonya G. & Sommerville I. (1998). *Requirements Engineering. Processes and Techniques*. John Wiley & Sons.

Krogstie J. (1998). Integrating the understanding of Quality in Requirements Specification and Conceptual Modeling. *Software Engineering Notes*, 23 (1), 89-91.

Krogstie J. (2001a). A Semiotic Approach to Quality in Requirements Specifications. *Proceedings IFIP 8.1 Working Conference on Organisational Semiotics*.

Krogstie J. (2001b). Using a Semiotic Framework to Evaluate UML for the Development for Models of High Quality. Siau K., & Halpin, T. (eds.). *Unified Modelling Language: System Analysis, Design and Development Issues*, IDEA Group Publishing, 89-106.

Krogstie J., & Jørgensen H. D. (2003). Quality of Interactive Models. Genero M., Grandi F., van den Heuvel W.-J., Krogstie J., Lyytinen K., Mayr H.C., Nelson J., Olivé A., Piattini M., Poels G., Roddick J. F, Siau K., Yoshikawa M., & Yu E. S. K. (eds.). *Advanced Conceptual Modeling Techniques, ER 2002 Workshops: ECDM, MobIMod, IWCMQ, and eCOMO*, Tampere, Finland, October, 2002, Revised Papers, Springer-Verlag Berlin Heidelberg, 251-263.

Krogstie J., & Sølvberg, A. (1996). A Classification of Methodology Frameworks for Computerised Information Systems Support in Organisations. *Proceedings of the IFIP8.1/8.2 Conference on Method Engineering: Principles of Method Construction and Tool Support*, USA.

Kulak D., & Guiney E. (1998). *Use Cases: Requirements in Context*. Addison-Weslay (1998).

Kunda D., & Brooks L. (1999). Applying Social-Technical Approach for COTS Selection. *Proceedings of the 4th UKAIS conference*.

Kunda D., (2003). STACE: Social Technical Approach to COTS Software Selection. Cechich A., Piattini M., & Vallecillo A. (eds.). *Component-Based Software Quality - Methods and Techniques*. Lecture Notes in Computer Science, Springer-Verlag, 85-98.

LaBudde E. V. (1997). *Finding the Right Off-the-Shelf Requirements Management Tool*. Available online at: http://www.devicelink.com/mddi/archive/97/10/013.html.

Lang M., & Duggan J. (2001). A Tool to Support Collaborative Software requirements Management. *Requirement Engineering*, 6(3), 161-172.

Lauesen S. (2002). *Software Requirements. Styles and Techniques*. Addison-Wesley.

Leffingwell D., & Widrig D. (2000). *Managing Software Requirements. A Unified Approach*, Addison-Wesley.

Lending D., & Chervany N. L., (1998). CASE Tools: Understanding the Reasons for Non-Use. *Computer Personnel*, 19(2), 13-26.

Lindland O. I., Sindre G., & Sølvberg A. (1994). Understanding quality in conceptual modelling. *IEEE Software*, 11(2), 42-49.

Loucopoulos P., & Karakostas V. (1995). *System Requirements Engineering*. McGraw-Hill Book Company.

Lubars M., Potts C., & Richter C. (1993). A Review of the State of the Practice in Requirements Modelling. *Proceeding of the First IEEE International Symposium in Requirements Engineering* (*RE'93*), 2-14.

Maiden N. A., Croce V., Kim H., Sajeva G., & Topuzidou S. (2003). SCARLET: Integrated Process and Tool Support for Selecting Software Components. Cechich A., Piattini M., & Vallecillo A. (eds.). *Component-Based Software Quality - Methods and Techniques*. Lecture Notes in Computer Science, Springer-Verlag, 85-98.

Maiden N. A., & Kim H. (2002). SCARELT: Light-Weight Component Selection in BANKSEC. Barbier F. (eds.). *Business Computer-based Software Engineering*. vol. 705. Kluwer Academic Publishers, Boston, 49-63.

Maiden N. A., Kim H., & Ncube C. (2002). Rethinking Process Guidelines for Selecting Software Components. In: *Proceedings of the First International Conference on COTS-Based Software Systems*. Lecture Notes in Computer Science, Springer-Verlag, 151-164.

Maiden N. A., & Ncube C. (1998). Acquiring COTS Software Selection Requirements. *IEEE Software*, 46-56.

Mannion M., Kaindl H., Wheadon J., & Keepence B. (1999). Reusing Single System Requirements from Application Family Requirements. *Proceedings of the 21st international conference on Software engineering*, US, 453-462.

Matulevičius R., Karlsson L., & Sindre G. (2004). How Evaluation Techniques Influence the RE-Tool Evaluation: An Experiment. *Proceedings of the Industrial Experience Track of the European Software Process Improvement Conference* (*EuroSPI 2004*), Trondheim, Norway.

Matulevičius R., & Sindre G. (2005a). Requirements Engineering Tool Evaluation Approach. *Accepted for the Proceedings of the 14th International Conference on Information Systems Development* (*ISD 2005*), Karlstad, Sweden, August.

Matulevičius R. & Sindre G. (2005b). Overview of the Evaluation Approaches and Frameworks for Requirements Engineering Tools. *Accepted for the Proceedings of the 14th International Conference on Information Systems Development* (*ISD 2005*), Karlstad, Sweden, August.

Matulevičius R., & Sindre G. (2004). Requirements Specification for RE-tool Evaluation: Towards a Specification Exemplar. *Proceedings of the 17th International Conference Software and Systems Engineering and their Applications (ICSSEA 2004)*, Paris, France, November-December.

Matulevičius R., & Strašunskas D. (2003). Evaluation Framework of Requirements Engineering Tools for Verification and Validation. Genero M., Grandi F., van den Heuvel W.-J., Krogstie J., Lyytinen K., Mayr H.C., Nelson J., Olivé A., Piattini M., Poels G., Roddick J. F, Siau K., Yoshikawa M., Yu E. S. K. (eds.). *Advanced Conceptual Modeling Techniques, ER 2002 Workshops: ECDM, MobIMod, IWCMQ, and eCOMO*, Tampere, Finland, Revised Papers, Springer-Verlag Berlin Heidelberg, 251-263.

Matulevičius R. (2005). Prototype of the Evaluation Framework for Functional Requirements of RE-tools. *Accepted for the Proceedings of the 13th IEEE International Requirements Engineering Conference* (*RE'05*), Paris, France, August-September.

Matulevičius R. (2004a). Survey of Requirements Engineering Practice in Lithuanian Software Development Companies. Vasilecas O., Čaplinskas A., Wojtkowski W., Wojtkowski W. G., Zupancic J. Wrycza S. (eds.). *Proceedings of the 13th International Conference on Information*

Systems Development (*ISD 2004*). *Advances in Theory, Practice and Education.* To be published by Kluwer Academic/Plenum Publishers. Vilnius, Lithuania, September, 327-339.

Matulevičius R. (2004b). Validating an Evaluation Framework for Requirement Engineering Tools. Krogstie J., Halpin T., & Siau K., (eds.). *Information Modeling Methods and Methodologies* (*Adv. Topics of Database Research*), Idea Group Publishing, 148-174.

Matulevičius R. (2004c). How Requirements Specification Quality Depends on Tools: A Case Study. Persson A., Stirna J. (eds.). *Proceedings of the 16th International Conference on Advanced Information System Engineering* (*CAiSE'2004*), Riga, Latvia, (2004), Springer-Verlag Berlin Heidelberg, 353-367.

Matulevičius R. (2004d). Process Improvement in Requirements Engineering by Acquisition of RE Tools. Glinz M. (eds.): *Proceedings of the Doctorial Consortium at the 12th IEEE International RE Conference (RE'04)*, Kyoto, Japan, September, 37-40.

Matulevičius R. (2003a). Validating an Evaluation Framework for Requirement Engineering Tools. *Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design* (*EMMSAD'03*), Klagenfurt/Velden, Austria, June 16-20.

Matulevičius R. (2003b). Usability and Adaptability of Evaluation Framework for Requirements Engineering Tools. *Proceedings of the 2003 International MultiConference in Computer Science & Engineering* (*SERP'03*), Monte Carlo Resort, Las Vegas, Nevada, USA, June.

Matulevičius R. (2002). Process Improvement in Information System Requirements Engineering. *Scientific Proceedings of Riga Technical University*, *Information System Development* (*ISD 2002*) *Doctoral Consortium*, Riga, Latvia.

McCall J. A., Richards P. K., & Walters G. F. (1997). *Factors in Software Quality*, RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055.

Moody D. L., & Shanks G. G. (2003). Improving the Quality of Data Models Empirical Validation of a Quality Management Framework. *Information Systems*, 28 (6), 619-650.

Mora M. M., & Denger, D. (2003). Requirements Metrics. An Initial Literature Survey on Measurement Approaches for Requirement Specifications. *IESE-Report No. 096.03/E*, version 1.0, Kaiserslautern, 2003.

Munkelien K. (2003). *Kravspek-verktøy, prototype*, TDT4730 Information System Specification Report. IDI-NTNU. Trondheim.

NASA, (1991). NASA-STD-2001-91, *NASA Software Documentation Standard*. NASA Headquarters Software Engineering Program.

Nguyen L., & Swatman P. A. (2003). Managing the Requirements Engineering Process, *Requirements Engineering*, 8, 55-68.

Nikiforova O., & Sukovskis U. (2002). Framework for Comparison of System Modeling Tool. *Proceedings of the 5th IEEE International Baltic Workshop on DB and IS (BalticDB&IS'2002),* Tallinn, Estonia.

Nikula U., Sajaniemi J., & Kälviäinen H. (2000). A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. TBRC Research Report 1, *Telecom Business Research Center Lappeenranta*, Lappeenranta University of Technology.

Nuseibeh B. & Easterbrook, S. (2000). Requirements Engineering: a Roadmap. In: A. Finkelstein (eds.). *The future of software engineering*. ACM Press.

Oberndorf T. (1997). *COTS and Open Systems – an Overview*. [Online]. Available: http://www.sei.cmu.edu/str/descriptions/cots.html#ndi

Ochs M. A., Pfahl D., Chrobok-Diening G., & Nothhelfer-Kolb B. (2000). A COTS Acquisition Process: Definition and Application Experience. *Proceedings of the 11th ESCOM Conference*, Shaker, Maastricht, 335-343.

Orlikowski W. J. (1993). CASE Tools as Organisational Change: Investing Incremental and Radical Changes in System Development. *Management Information Systems Quarterly*, 17 (3), 309-340.

Paulk M. C., Curtis B., Chrissis M., & Weber, C. (1993). *Capability Maturity Model for Software: Version 1.1.* Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Paulk M. C., (1998). Models and Standards for Software Process Assessment and Improvement. *Advances in Computers*, 46, Academic Press, 1-33.

Pohl K. (1994). The Three Dimensions of Requirements Engineering: a Framework and its Applications, *Information systems*, *19*(3), 243-258.

Pohl K. (1996). *Process Centred Requirements Engineering*. Research Press LtD, John Wiley & Sons Inc.

Post G., & Kagan A. (2000). OO-CASE tools: an Evaluation of Rose. *Information and Software Technology*, 42, 383-388.

Regnell B., Host M., Natt och Dag J., & Hjelm T. (2000). Visualization of Agreement and Satisfaction in Distributed Prioritisation of Market Requirements. *Proceedings of the 6th International Workshop on Requirements Engineering – Foundation for Software Quality* (*REFSQ2000*), Sweden.

Rybinski H. (1987). A First-Order-Logic Databases. *ACM Transaction on Database Systems*, September 1987.

Robertson S., & Robertson J. (1999). Mastering the Requirements Process. Addison-Wesley.

Rout T. (2001). The SPICE Approach to Software Process Improvement. Hunter R. B., Thayer R. H., & Paulk M. C. (eds.). *Software Process Improvement*, IEEE Computer Society.333-350.

Rumbaugh J., Jacobson I., & Booch G. (1999). *The Unified Modeling Language Reference Manual.* Addison Wesley Professional.

Saaty T. L. (1980). *The Analytic Hierarchy Process*. McGraw-Hill, New York.

Sankaran S. (2001). Methodology for an Organisational Action Research Thesis. *Action Research International* [Online]. Available: http://www.scu.edu.au/schools/gcm/ar/ari/p-ssankaran01.html

Seddon R. B., Graeser V., & Willcocks L. P. (2002). Measuring Organisational IS Effectiveness: An Overview and Update of Senior Management Perspectives. *ACM SIGMIS Database*, 33 (2), ACM Press, 11-28.

Sedigh-Ali S., Ghafoor A., & Paul R. (2001). Software Engineering Metrics for COTS-Based Systems. *IEEE Computer*, 34 (5), 44-50.

SEI (2002). *Capability Maturity Model Integration (CMMISM): Version 1.1.* Technical Report SEI, CMU/SEI-2002-TR-029.

Shewhart W. A. (1939). *Statistical Method from the Viewpoint of Quality Control*. Graduate School of the Department of Agriculture, Washington. Re-published in 1986, Dover Publications, Inc.

Shipman F. M., & McCall R. (1994). Supporting Knowledge-Base Evolution with Incremental Formalization. Human Factors in Computing Systems. April, 285-291.

Simsion G. C. (2001). *Data modeling essentials. Analysis, Design, and Innovation*. Scottsdale, AZ : Coriolis Group Books.

Sindre G., & Krogstie J. (1995). Process Heuristics to Achieve Requirement Specifications of Feasible Quality. *Proceedings of the 2nd International Workshop on Requirements*

Engineering: Foundation for Software Quality (*REFSQ'95*). Jyväskylä, Finland.

Sinha A. P., & Vessey I. (1999). An Empirical Investigation of Entity-based and Object-oriented Data Modeling: a Development Life Cycle Approach. *Proceedings of the 20th International Conference on Information Systems*, USA, 229-244.

Snaprud M., & Kaindl H. (1994). Types and Inheritance in Hypertext. International Journal of Human-Computer Studies, 41, 223-241.

Sommerville I. (1997). *Software Engineering*. Fifth edition. Addison-Wesley.

Sommerville I., & Sawyer P. (1997). *Requirements Engineering. A Good Practice*. John Wiley and Sons.

Sommerville I., & Ransom J. (2005). An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement. *ACM Transactions on Software Engineering and Methodology*, 14 (1), 85-117.

Sølvberg A. (1999). Data and What They Refer to. Chen P., Akoka J., Kangassalo H., & Thalheim B. (eds.) *Conceptual Modeling: Current Issues and Future Trends*. LNCS 1565, Springer Verlag.

Sørumgård S., & Sindre S. (1995). Aspects of Software Process Quality. *Proceedings of the 4th Software Quality Conference,* Scotland.

Sørumgård S. (1997). *Verification of Process Conformance in Empirical Studies of Software Development*. PhD thesis, Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway.

Thunem S. (1997). *Process Modeling for Process Improvement – A Process Conformance Approach*. PhD thesis, Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway.

Urquhart C. (2001). Analysts and Client in Organisational Contexts: a Conversational Perspective. *Journal of strategic information Systems*, 10, 243-262.

Vigder M. R., & Dean J. (1997). An Architectural Approach to Building Systems from COTS Software Components. *Proceedings of the 1997 Center for Advanced Studies Conference (CASCON 97)*. Toronto.

Walker A. *et al.* (eds.) (2003). *The New International Webster's Comprehensive Dictionary of the English Language*. Encyclopaedic edition, Triden Press International.

Wiegers K. (1999). Automating Requirements Management. *Software Development*, 7 (7), July.

Wilson W. M., Rosenberg L. H., & Hyatt L. (1996). Automated Quality Analysis of Natural Language Requirement Specification. *Proceedings of the 14th Annual Pacific Northwest Software Quality Conference*.

Wohlin C., Runeson P., Høst M., Ohlsson M. C., Regnell B., & Wesslen A. (2002). *Experimentation in Software Engineering*, Kluwer Academic Publishers.

Yourdon E. (2000). *Modern Structured Analysis*. Prentice Hall.

Yu E., & Cysneiros L. M. (2002). Agent-Oriented Methodologies – Towards a Challenge Exemplar. Giorgini P., Lespérance Y., Wagner G., & Yu E. (eds.). Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002 at CAiSE'02).

Yu E. (1995). Modelling Strategic Relationships for Process Reengineering, PhD. thesis, University of Toronto.

Zave P. (1997). Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys, 29(4), 315-321.

Appendix A

This appendix presents a questionnaire used in the empirical study of the RE processes in Lithuanian software development organisations (Matulevičius, 2004a) as described in Chapter 3. Similar questionnaire is applied to investigate Norwegian software development organisations (Ekremsvik and Tiset, 2003).

Requirements Engineering: A Questionnaire

We invite you to take part in the project for process improvement in requirements engineering of information system. The goal of the research is to develop improved methods for requirements engineering in information systems development and to investigate how industrial requirements engineering processes could be improved by means of these methods. The research is focused on various aspects of requirements engineering, such as elicitation, analysis, documentation, validation, management. We are working on validating and identifying guidelines for RE processes from the executer's perspective and viewpoint. The results of this study are important for designers and developers because they will provide a guideline on how to build successful information systems.

If you like to see the results of this questionnaire or the study, please e-mail me. The project is performed, maintained and administered by information system (IS) group of Norwegian University of Science and Technology (NTNU).

This questionnaire is designed to discover what aspects of requirements engineering have been useful to you in your career. The results of the survey will be used to improve the requirements engineering activities and processes. All the collected data will be highly confidential and will be used only for study and research purposes. Your input is valuable and would be of great importance in helping us to create successful requirements engineering methods. In particular we have no intention of judging you as an organisation or a person – we are merely interested in learning about the current practice of requirements engineering.

The questionnaire consists of 22 questions, which are listed in 8 steps. It will take you about 15-20 minutes to fill it in. Please enter your user ID to the field below and carefully answers the following questions.

1.Name of your organisation. (optional)

2.URL to organisation's web page (optional)

3.Which of the following statements best describe your organisation? (Select all the answers that fit)

- Consulting company (we work in consulting for system integration, project management, support, and related services);
- Software development company (we develop packaged and/or in-house created software products and support our customers after product release);
- Software users (We buy software products and employ them to our organisational needs);
- Others (or additional comments). Please specify:

4. How many employees are working in the company?

5. How many employees are working with software development?

6. What is the role of software development projects in your company? (Mark all the answers that fit)

- Development/sale of off-the-shelf products;
- Development for external customers;
- Development for internal customers;

7. What is your position in the organisation? (Select ONE of them)

- Managing director;
- Project manager;
- Product leader;
- System developer;
- Communication expert;
- Technical author;
- Training and user support staff;
- User interface designer/usability expert;
- Others (or additional comments). Please specify:

REQUIREMENTS ENGINEERING

Requirements engineering (RE) is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families.
Please evaluate the IMPORTANCE of different requirements engineering FEATURES, which you use in your organisation in the following scale:

5 - **very important and useful** activities and features, which are widely used in our organisation work.
4 - **important and useful** features and activities, which are good and sometimes used in our organisation work.
3 - **useful but not important** features and activities, but still sometimes used in organisation work.
2 - features and activities, which **don't strongly affect** the organisation work.
1 - **not important neither useful** features and activities, which are not used at all.

8. REQUIREMENTS DISCOVERY
What is important during process of requirements discovering?

☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Meetings and conversations with (potential) software stakeholders.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Performance of surveys (market, stakeholders).
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Analysing existing (similar) software.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Analysing competitive documentations, which was provided by software vendor.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Reuse of domain specific requirements from predefined repositories.
Other(or additional comments). Please specify:

9. IMPORTANCE OF STAKEHOLDERS
What roles of stakeholders are important in RE process?
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Project managers
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Software developers.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Project/product sponsors.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Executive committee.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Communication experts.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Technical authors.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Training and user support staff.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Domain experts.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – End-users.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – Indirect stakeholders and/or their managers.
☐ 1, ☐ 2, ☐ 3, ☐ 4, ☐ 5 – User interface designer/ usability expert.
Other(or additional comments). Please specify:

10. REQUIREMENTS TRACEABILITY
What traceable relationships are important during RE?

□ 1, □ 2, □ 3, □ 4, □ 5 – We keep information about requirements source (from where requirement was discovered).
□ 1, □ 2, □ 3, □ 4, □ 5 – We keep different versions of requirements specification.
□ 1, □ 2, □ 3, □ 4, □ 5 – We keep traceable relationships with requirements specification and design phrases.
Other(or additional comments). Please specify:

11. REQUIREMENTS REPRESENTATION
Please evaluate importance of different requirements descriptions.
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirements description, using informal language and conversations
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirements descriptions, using semi-formal definitions.
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirements descriptions, using formal definitions.

12. REPRESENTATION LANGUAGES and TECHNIQUES
What requirements representation languages and techniques are important for you during RE?
□ 1, □ 2, □ 3, □ 4, □ 5 – Natural language.
□ 1, □ 2, □ 3, □ 4, □ 5 – Use Case templates.
□ 1, □ 2, □ 3, □ 4, □ 5 – Use Case diagrams.
□ 1, □ 2, □ 3, □ 4, □ 5 – ER diagrams.
□ 1, □ 2, □ 3, □ 4, □ 5 – UML.
□ 1, □ 2, □ 3, □ 4, □ 5 – State charts.
□ 1, □ 2, □ 3, □ 4, □ 5 – DFD
□ 1, □ 2, □ 3, □ 4, □ 5 – OMT.
□ 1, □ 2, □ 3, □ 4, □ 5 – Z-schemas
□ 1, □ 2, □ 3, □ 4, □ 5 – Algebraic specifications.
□ 1, □ 2, □ 3, □ 4, □ 5 – Action semantic.
□ 1, □ 2, □ 3, □ 4, □ 5 – Aggregate approach.
Other(or additional comments). Please specify:

13. REQUIREMENTS GROUPS
Please evaluate importance of requirements groups.
□ 1, □ 2, □ 3, □ 4, □ 5 – Non-functional requirements
□ 1, □ 2, □ 3, □ 4, □ 5 – Functional requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Domain/ user/ organisational requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Architectural requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Adaptability/ reliability/ security/ usability requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – End-users requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Developers requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Financial requirements.
Other(or additional comments). Please specify:

14. REQUIREMENTS ATTRIBUTES (e.g.: priority, originality, etc)
Please evaluate importance of requirements attributes.
□ 1, □ 2, □ 3, □ 4, □ 5 – Definition of requirements attributes.
□ 1, □ 2, □ 3, □ 4, □ 5 – Definition of views according to attributes.
□ 1, □ 2, □ 3, □ 4, □ 5 – Sorting according to attributes.
□ 1, □ 2, □ 3, □ 4, □ 5 – Filtering according to attributes.
Other(or additional comments). Please specify:

15. NEGOTIATION and COLLABORATION FACILITIES
What negotiation and collaboration facilities are important during RE?
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirements message boards.
□ 1, □ 2, □ 3, □ 4, □ 5 – Means of brainstorm.
□ 1, □ 2, □ 3, □ 4, □ 5 – Means of discussion/negotiation.
□ 1, □ 2, □ 3, □ 4, □ 5 – Maintenance of rationale behind requirements.
□ 1, □ 2, □ 3, □ 4, □ 5 – Media/video/meeting facilitators.
□ 1, □ 2, □ 3, □ 4, □ 5 – Access data from geographically distributed work places.
□ 1, □ 2, □ 3, □ 4, □ 5 – Editing the same elements (requirements) synchronously.
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirement change notification and propagation.
Other(or additional comments). Please specify:

16. REUSE of REQUIREMENTS
Please evaluate how important is reuse of requirements.

□ 1, □ 2, □ 3, □ 4, □ 5 – Maintenance of data repository.
□ 1, □ 2, □ 3, □ 4, □ 5 – Selection and extraction of domain specific requirements from other projects.
□ 1, □ 2, □ 3, □ 4, □ 5 – Reverse engineering of requirements from code/ design.
Other(or additional comments). Please specify:

17. REPORTS and DOCUMENTATION
Please evaluate importance of reports and documentation during RE.

□ 1, □ 2, □ 3, □ 4, □ 5 – Reports and documentation of representations (informal, semi-formal, formal).
□ 1, □ 2, □ 3, □ 4, □ 5 – Reports and documentation, which help to understand requirements (rationale, brainstorm review, negotiation accounts, etc).
□ 1, □ 2, □ 3, □ 4, □ 5 – Reports and documentation about requirements agreement.
Other(or additional comments). Please specify:

18. FINAL REQUIREMENTS SPECIFICATION
What is importance of final requirements specification?

□ 1, □ 2, □ 3, □ 4, □ 5 – Standards requirements specification (i.e.: IEEE std 830-1998).
□ 1, □ 2, □ 3, □ 4, □ 5 – Standard for requirements specification, defined by your organisation.
□ 1, □ 2, □ 3, □ 4, □ 5 – Requirements specification, agreed between all stakeholders, defined in formal language.
Other(or additional comments). Please specify:

19. On the average, how much of software development is requirements engineering?

- up to 25% of project development time;
- 26 - 50% of project development time;
- 51 – 75% of project development time;
- more than 75% of a project development time;
- difficult to say.

20. What software tools do you use for requirements engineering? (Mark all the answers that fit)

- Standard office tools (for example: word processors, spreadsheets, etc);
- Graphical packages (for example: Visio, ABC Flowcharter, Adobe Photoshop, etc);
- Requirements engineering/management tools (for example: RequisitePro, DOORS, RDT, Cradle, etc);
- Other tools

21. Could you write requirements engineering activities, which you perform with requirements engineering/management tool.

21. Why did you decide not to use a requirement engineering tool? (Mark all the answers that fit)

- The usage of requirements engineering tool was not considered;
- The functionality in none of the existing tools didn't fit our development approach;
- The usage/functionality of the tool was too complicated;
- We did not find reliable vendor;
- Prices of a requirements engineering tool(s) were too high;
- Return on investment for a requirements engineering tool(s) was too low;
- Other(or additional comments). Please specify:

22. Can we contact you with follow-up questionnaire or interview?

- YES
- NO

Your assistance in providing this information is very appreciated. If there is anything else you would like to tell us about the RE processes in your organisation and usability of the requirements engineering tool(s), please do so in the space provided below.

THANK YOU for the time, spent to complete this questionnaire.

Appendix B

This appendix presents results of the empirical study of the RE processes in Lithuanian software development organisations (Matulevičius, 2004a) described in Chapter 3.

1.Name of your organisation. (optional)
2.URL to organisation's web page (optional)

3.Which of the following statements best describe your organisation? (Select all the answers that fit)

- Consulting company (we work in consulting for system integration, project management, support, and related services);
- Software development company (we develop packaged and/or in-house created software products and support our customers after product release);
- Software users (We buy software products and employ them to our organisational needs);
- Others (or additional comments). Please specify:

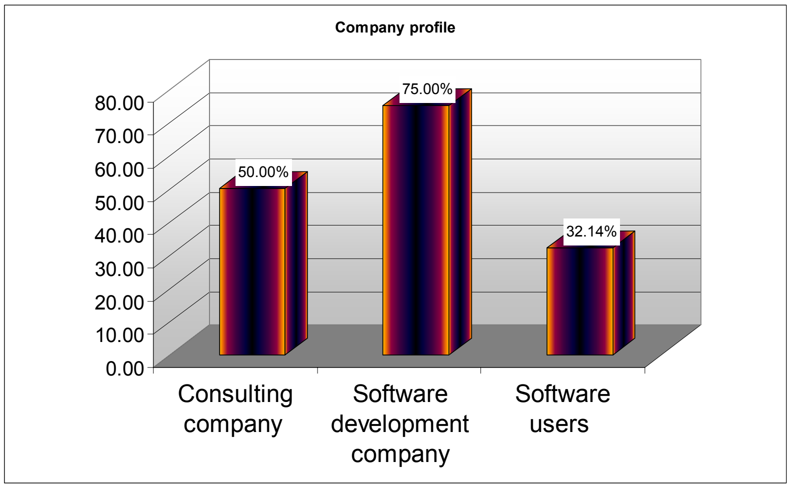


Figure B.1 Responses to Question 3

4. How many employees are working in the company?
5. How many employees are working with software development?

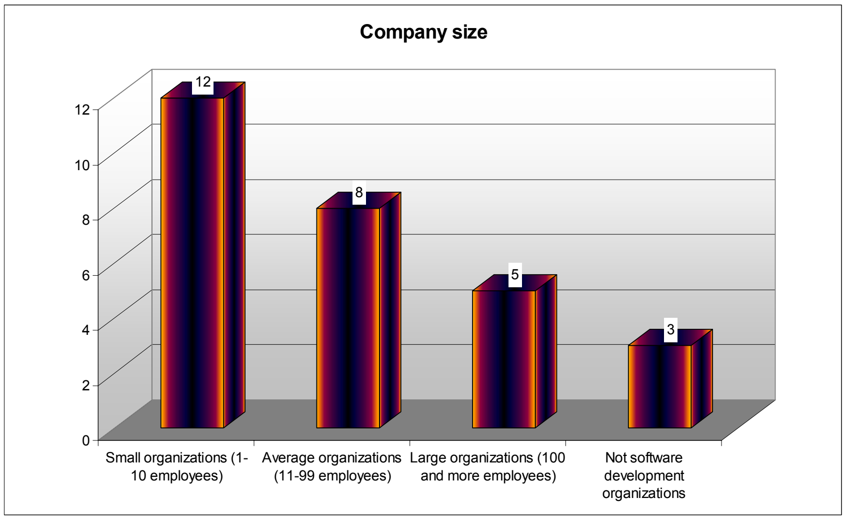


Figure B.2 Responses to Question 5

6. What is the role of software development projects in your company? (Mark all the answers that fit)

- Development/sale of off-the-shelf products;
- Development for external customers;
- Development for internal customers;

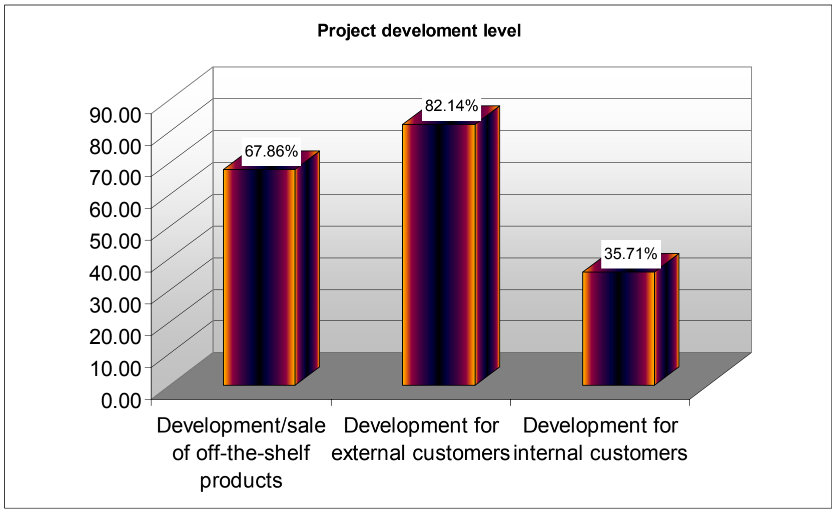


Figure B.3 Responses to Question 6

7. What is your position in the organisation? (Select ONE of them)

- Managing director;
- Project manager;
- Product leader;
- System developer;
- Communication expert;
- Technical author;
- Training and user support staff;
- User interface designer/usability expert;
- Others (or additional comments). Please specify:

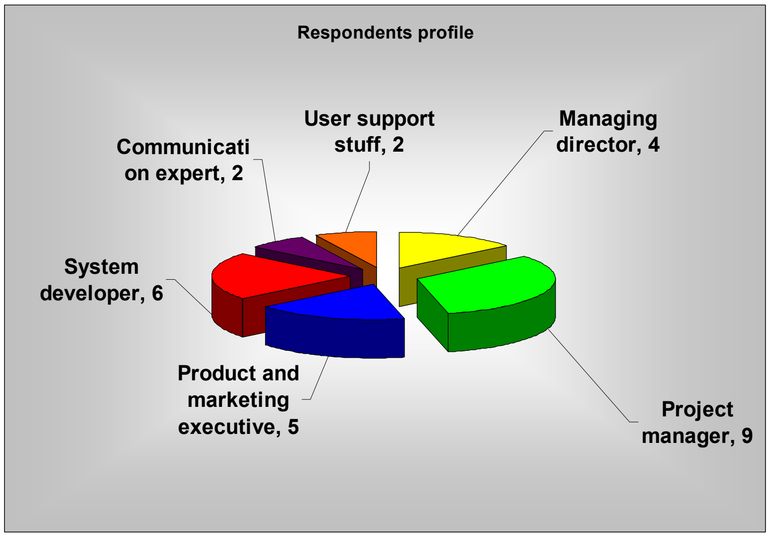


Figure B.4 Responses to Question 7

The tables presents the mean (*M*), variance (V), and chi-square (χ^2) calculations for the RE activities. χ^2 is calculated like $\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$, where *O* is the observed pattern, *E* is the expected pattern. The table value of χ^2 for five degree of freedom and α=0.05 is 11.07. Two importance levels - high and low - are selected according to the *E* mean threshold, which is equal to 3.79. High importance is if the *E* mean is higher than threshold - {18, 6, 1, 1, 1, 1} and {12, 8, 4, 2, 1, 1}. Low importance is if the *E* mean is equal to threshold – {9, 10, 6, 1, 1, 1} and {10, 9, 5, 2, 1, 1}. If the calculation is higher than χ^2 value (with all *E*), then the activity is considered not important.

8. REQUIREMENTS DISCOVERY
What is important during process of requirements discovering?

Table B.1 Findings of Question 8

ACTIVITIES, METHODS, AND TECHNIQUES OF REQUIREMENTS DISCOVERY.	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Analysing existing (similar) software.	10	11	6	1	0	0	4,07	0,74	4,96	**High**
Performance of surveys (market,	6	11	5	4	2	0	3,54	1,44	8,38	**High**

stakeholders).										
Meetings and conversations with (potential) software stakeholders.	26	2	0	0	0	0	4,93	0,07	10,22	**High**
Reuse of domain specific requirements from predefined repositories.	6	6	8	5	2	1	3,33	1,54	9,90	**Low**
Analysing competitive documentations, which were provided by software vendors.	6	8	6	5	3	0	3,32	3,33	11,41	Not imp.

9. IMPORTANCE OF STAKEHOLDERS
What roles of stakeholders are important in RE process?

Table B.2 Findings of Question 9

STAKEHOLDERS	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Project managers.	10	7	6	2	2	1	3,78	1,56	2,46	**High**
Software developers.	10	8	5	2	2	1	3,86	0,87	1,58	**High**
Marketing personnel.	19	6	2	1	0	0	4,54	0,63	3,06	**High**
Domain experts.	18	6	2	2	0	0	4,43	0,85	4,00	**High**
End-users.	8	14	5	1	0	0	4,04	0,63	8,58	**High**
Training and user support staff.	7	6	7	6	1	1	3,44	1,49	10,70	**Low**
Communication experts.	5	8	3	5	5	2	3,12	2,11	24,91	Not imp.
Indirect stakeholders and/or their managers.	2	8	12	4	2	0	3,14	1,02	20,31	Not imp.
Technical authors.	4	6	9	7	2	0	3,11	1,36	22,30	Not imp.
User interface designer/ usability expert.	2	8	7	4	5	2	2,92	1,59	26,31	Not imp.
Project/product sponsors.	7	4	4	6	3	4	3,25	2,11	24,88	Not imp.

10. REQUIREMENTS TRACEABILITY
What traceable relationships are important during RE?

Table B.3 Findings of Question 10

ACTIVITIES, METHODS, AND TECHNIQUES OF REQUIREMENTS TRACEABILITY	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
We keep different versions of requirements specification.	14	9	4	1	0	0	3,96	1,07	2,96	**High**
We keep information about requirements source (from where requirement was discovered).	14	9	3	2	0	0	4,25	0,86	2,71	**High**
We keep traceable relationships with requirements specification and design phases.	8	10	3	6	1	0	3,64	1,50	10,31	**Low**

11. REQUIREMENTS REPRESENTATION
Please evaluate importance of different requirements descriptions.

Table B.4 Findings of Question 11

REQUIREMENTS REPRESENTATION	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Requirements description, using informal language and conversations.	9	10	6	2	0	1	3,96	0,88	3,25	**High**
Requirements descriptions, using semi-formal definitions.	9	12	4	3	0	0	3,96	0,92	5,25	**High**
Requirements descriptions, using formal definitions.	6	7	9	2	3	1	3,41	1,56	8,40	**Low**

12. REPRESENTATION LANGUAGES and TECHNIQUES
What requirements representation languages and techniques are important for you during RE?

Table B.5 Findings of Question 12

REQUIREMENTS REPRESENTATION LANGUAGES AND TECHNIQUES	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Natural language.	16	6	3	1	0	2	4,42	0,73	4,58	**High**
State charts.	5	9	3	2	2	7	3,62	1,55	40,30	Not imp.
DFD.	4	10	3	2	2	7	3,75	1,25	41,51	Not imp.
Use Case diagrams.	8	5	5	2	1	7	3,81	1,46	38,80	Not imp.
Use Case templates.	2	14	2	1	2	7	3,62	1,15	46,71	Not imp.
UML.	6	5	5	2	2	8	3,55	1,73	53,38	Not imp.
ER diagrams.	3	10	3	1	2	9	3,58	1,37	70,50	Not imp.
OMT.	1	4	4	4	3	12	2,75	1,53	138,08	Not imp.
Action semantic.	3	2	2	5	3	13	2,80	2,17	164,64	Not imp.
Algebraic specifications.	0	2	5	3	5	13	2,19	1,23	186,61	Not imp.
Z-schemas.	0	2	2	3	8	13	1,87	1,27	210,74	Not imp.
Aggregate approach.	0	1	5	3	6	13	2,07	1,07	186,61	Not imp.

13. REQUIREMENTS GROUPS
Please evaluate importance of requirements groups.

Table B.6 Findings of Question 13

REQUIREMENTS GROUPS	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Functional requirements.	16	6	3	1	1	1	4,30	1,14	2,58	**High**
End-users requirements.	11	10	3	1	0	3	4,24	0,69	6,33	**High**
Architectural requirements.	6	14	4	0	1	3	3,96	0,79	8,27	**Low**
Adaptability/ reliability/ security/ usability requirements.	9	10	3	0	1	5	4,13	0,94	18,50	Not. imp.
Domain/ user/ organisational requirements.	10	8	3	0	2	5	4.04	1,41	19,58	Not imp.
Developers' requirements.	5	10	6	2	1	4	3,67	1,10	11,78	Not imp.
Non-functional requirements.	6	6	6	3	3	4	3,38	1,81	18,00	Not imp.
Financial requirements.	5	8	5	5	0	5	3,57	1,17	24,11	Not imp.

14. REQUIREMENTS ATTRIBUTES (e.g.: priority, originality, etc)
Please evaluate importance of requirements attributes.

Table B.7 Findings of Question 14

ACTIVITIES, METHODS, AND TECHNIQUES RELATED TO REQUIREMENTS ATTRIBUTES	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Sorting according to attributes.	8	9	5	1	1	4	3,92	3,82	9,38	**Low**
Definition of views according to attributes.	7	10	4	2	1	4	3,83	1,19	10,21	**Low**
Definition of requirements attributes.	9	8	6	0	1	4	4,00	1,04	10,40	**Low**
Filtering according to attributes.	7	7	6	1	1	6	1,12	1,20	26,34	Not imp.

15. NEGOTIATION and COLLABORATION FACILITIES
What negotiation and collaboration facilities are important during RE?

Table B.8 Findings of Question 15

ACTIVITIES, METHODS, AND TECHNIQUES FOR NEGOTIATION AND COLLABORATION	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Means of brainstorm.	8	11	4	1	2	2	3,83	1,34	4,96	**High**

Means of discussion/negotiation.	16	7	3	1	0	1	4,41	0,71	3,21	**High**
Maintenance of rationale behind requirements.	7	12	5	0	1	3	3,96	0,87	10,33	**High**
Requirement change notification and propagation.	9	9	4	0	2	4	3,96	1,35	11,77	Not imp.
Requirements message boards.	6	10	3	3	2	4	3,63	1,55	13,01	Not imp.
Media/video/meeting facilitators.	3	10	7	5	1	2	3,35	1,12	11,31	Not imp.
Access data from geographically distributed work places.	7	7	3	5	2	4	3,50	1,70	16,64	Not imp.
Editing the same elements (requirements) synchronously.	5	7	6	2	3	5	3,96	1,35	23,14	Not imp.

16. REUSE of REQUIREMENTS
Please evaluate how important is reuse of requirements.

Table B.9 Findings of Question 16

ACTIVITIES, METHODS, AND TECHNIQUES FOR REQUIREMENTS REUSE	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Selection and extraction of domain specific requirements from other projects.	8	6	8	1	3	2	3,64	1,91	7,38	**Low**
Maintenance of data repository.	10	3	8	1	3	3	3,58	1,69	13,68	Not imp.
Reverse engineering of requirements from code/ design.	4	5	8	4	5	2	2,96	1,80	26,18	Not imp.

17. REPORTS and DOCUMENTATION
Please evaluate importance of reports and documentation during RE.

Table B.10 Findings of Question 17

ACTIVITIES, METHODS, AND TECHNIQUES FOR REQUIREMENTS DOCUMENTATION	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Reports and documentation, which help to understand requirements (rationale, brainstorm, negotiation, etc).	11	9	5	1	1	1	4,04	1,11	0,96	**High**
Reports and documentation of representations (informal, semi-formal, formal).	13	10	4	0	0	1	4,33	0,54	3,58	**High**
Reports and documentation about requirements agreement.	14	9	3	1	1	0	4,21	1,06	2,21	**High**

18. FINAL REQUIREMENTS SPECIFICATION
What is importance of final requirements specification?

Table B.11 Findings of Question 18

ACTIVITIES, METHODS, AND TECHNIQUES FOR FINAL REQUIREMENTS SPECIFICATION	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Standard for requirements specification, defined by your organisation.	16	8	1	1	1	1	3,45	1,97	0,89	**High**
Standards requirements specification (e.g., IEEE std 830-1998).	7	4	6	2	3	6	4,37	1.01	34,04	Not imp.
Requirements specification, agreed between all stakeholders, defined in formal language.	13	4	3	4	0	4	4,08	1,38	14,33	Not imp.

19. On the average, how much of software development is requirements engineering?

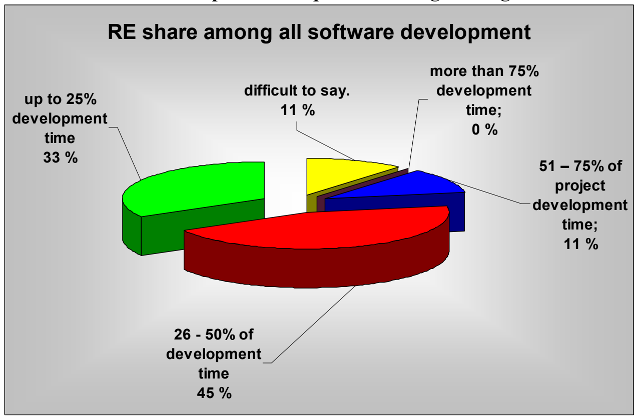


Figure B.5 Responses to Question 19

20. What software tools do you use for requirements engineering? (Mark all the answers that fit)

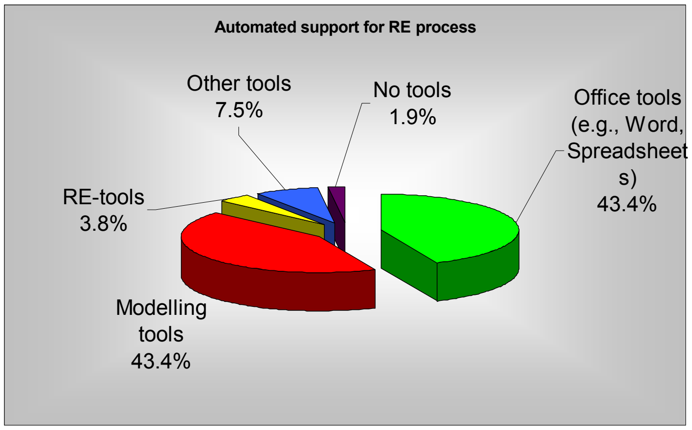


Figure B.6 Responses to Question 20 (tool categories)

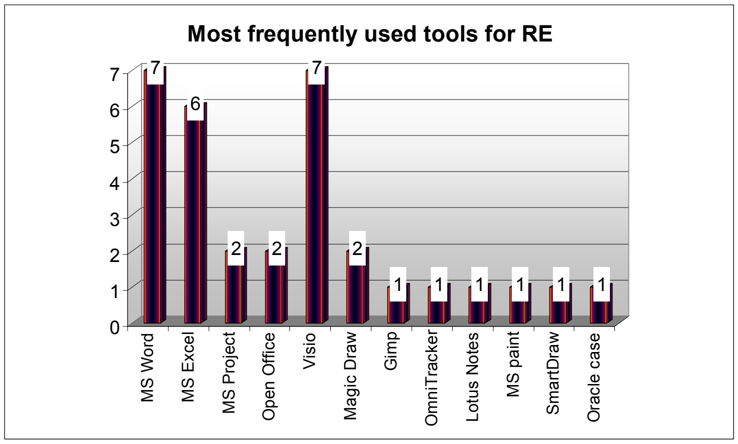


Figure B.7 Responses to Question 20 (individual tools)

21. Why did you decide not to use a requirement engineering tool? (Mark all the answers that fit)

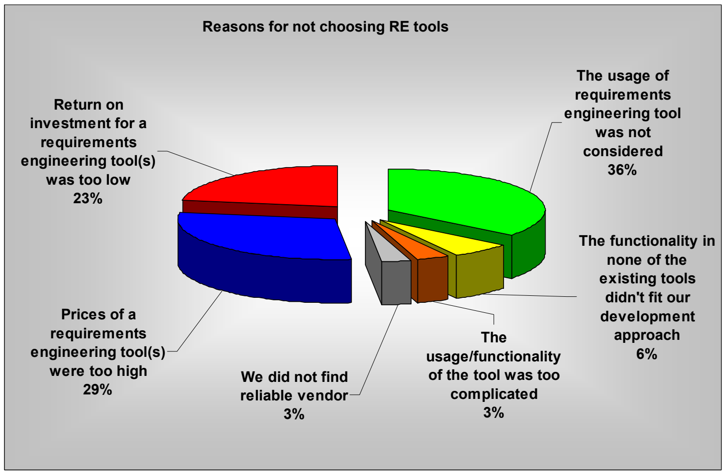


Figure B.8 Responses to Question 21

Appendix C

Case studies A and C described in Chapter 7, are using an evaluation framework for functional RE-tool requirements. The framework is adapted to the evaluation form provided in this appendix.

EVALUATION OF FRAMEWORK FOR COMPARISON OF REQUIREMENTS ENGINEERING TOOLS

Different tools are developed for system modelling. Some tools are widely used, some rarely. The reason for selecting the tool can be different. Sometimes it is the name of tool's vendor, sometimes it is a recommendation from people working in similar area, sometimes it is conservatism in people thinking, when people get to use a modelling tool and at the same time feel unsatisfied with their selection. So we need a proper way for evaluation and comparison modelling tools before acquisition.

Table C.1 Evaluation Form for the Requirements Representation Dimension

Representation dimension

	Feature	**Importance of the feature**	**Future tendency**	**Activities** THE TOOL SHOULD:	**Importance of the activity**	**Future tendency**
Representation dimension	**FEF1.1.** Specify uniquely identifiable description using informal language.	________	______	provide natural language description.	__________	______
				allow specifying unique identification (ID) for each separate requirement.	__________	______
				allow importing of requirements and their description from textual document.	__________	______
				________________________________	__________	______
				________________________________	__________	______
	FEF1.2. Specify requirements using semi-formal language(s).	________	______	provide tools for semiformal language description (i.e. ER-diagrams, UML diagrams, DFD, OMT).	__________	______
				provide forward/ backward traceability between informal, semiformal, formal descriptions.	__________	______
				________________________________	__________	______
				________________________________	__________	______
	FEF1.3. Specify requirements using formal language(s).	________	______	provide tools for formal language description (i.e. Z-schemas, algebraic specifications, action semantics, B-notations).	__________	______
				provide forward/ backward traceability between informal, semiformal, formal descriptions.	__________	______
				________________________________	__________	______
				________________________________	__________	______
	FEF1.4. Define traceable associations between requirements and the different elements of requirements specification.	________	______	provide functions for testing traceability between informal, semiformal and formal requirement description.	__________	______
				create parent-child traceable relations between requirements.	__________	______
				maintain peer-to-peer traceable relations between requirements.	__________	______
				maintain traceable relation between different related information.	__________	______
				maintain forward/ backward traceability between source of requirements, requirements and design.	__________	______
				________________________________	__________	______
				________________________________	__________	______
	FEF1.5. Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.	________	______	allow importing/exporting requirements description from/to textual documents.	__________	______
				allow importing/exporting requirements description from/to graphical documents.	__________	______
				allow import/export of requirements structure.	__________	______
				________________________________	__________	______
				________________________________	__________	______
				________________________________	__________	______

What important features do you think need to be added to representation dimension?	**What important features do you think need to be added to representation dimension?**
________________________________	________________________________

Table C.2 Evaluation Form for the Requirements Agreement Dimension

Agreement dimension

	Feature	**Importance of the feature**	**Future tendency**	**Activities** **THE TOOL SHOULD:**	**Importance of the activity**	**Future tendency**
Agreement dimension	**FEF2.1.** Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	________	______	maintain user authentication to the system (i.e. user name, password).	__________	______
				allow grouping users into different groups.	__________	______
				allow creating different views (according to documents, requirements, attributes) for different groups of stakeholders.	__________	______
				register agreement/ rationale/ discussion/ negotiation/ changes/ history of requirements and by how it was achieved.	__________	______
				call the earlier requirement description/ versions and register them into history context.	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
	FEF2.2. Classify requirements into logical user- defined groupings.	________	______	allow specifying attributes/ properties of the requirement.	__________	______
				provide sorting according to different attributes/ properties.	__________	______
				provide filtering according to different attributes/ properties.	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
	FEF2.3. Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	________	______	provide www-based interface for geographically distributed users.	__________	______
				allow making copy for modification of already approved version of requirements description in different abstract levels (document, requirement).	__________	______
				provide change approval cycle for multiple change negotiation and approval before posting into common repository.	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
	FEF2.4. Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.	________	______	provide the single repository or data dictionary.	__________	______
				provide separate data dictionaries for non-technical users and technical users.	__________	______
				provide the help system to the users.	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
				____________________________________	__________	______
				____________________________________	__________	______

What important features do you think need to be added to agreement dimension?	**What important features do you think need to be added to agreement dimension?**
____________________________________	____________________________________

Table C.3 Evaluation Form for the Requirements Specification Dimension

Specification dimension

	Feature	**Importance of the feature**	**Future tendency**	**Activities** **THE TOOL SHOULD:**	**Importance of the activity**	**Future tendency**
Specification dimension	**FEF3.1.** Collect and store common system's and product family's domain requirements.	________	______	enable selection and extraction of common domain requirements.	__________	______
				incorporate common requirements to concrete project.	__________	______
				adapt/ spread changes in domain requirements to concrete projects within domain.	__________	______
				provide comparison of domain requirements feasibility.	__________	______
				______________________________	__________	______
				______________________________	__________	______
	FEF3.2. Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	________	______	provide wizards for report generation.	__________	______
				provide possibility to print report according views and sorting.	__________	______
				provide possibility to print results of rationale, brainstorm and etc.	__________	______
				provide techniques for error checking.	__________	______
				______________________________	__________	______
				______________________________	__________	______
				______________________________	__________	______
				______________________________	__________	______
				______________________________	__________	______
				______________________________	__________	______
				______________________________	__________	______
	FEF3.3. Generate the complete specification, expressed using formal language commonly agreed by all stakeholders.	________	______	correspond to standards of software documentation.	__________	______
				correspond to standards, which were deployed in a organisation.	__________	______
				support formal languages for complete, commonly agreed requirements specification.	__________	______
				______________________________	__________	______
				______________________________	__________	______

What important features do you think need to be added to specification dimension?	**What important features do you think need to be added to specification dimension?**
______________________________	______________________________

Appendix D

This appendix presents a raw material of Case study A described in Chapter 7. The table shows how the framework features are evaluated by the participants.

Table D.1 Raw Material of Case Study A

Partici-pants / Features	1	2	3	4	5	6	7	8	9	10	11	12	13	Mean	Variance
FEF1.1	**10**	**10**	**7**	**10**	**5**	**7**	**7**	**5**	**10**	**8**	**8**	**8**	**10**	**8.08**	**3.4**
FEF1.1.1	10	10	7	10	10	10	7	10	10	7	8	8	10	**9.00**	**1.8**
FEF1.1.2	9	7	7	10	7	5	9	8	10	9	5	8	10	**8.00**	**3.0**
FEF1.1.3	10	10	9	2	3	7	7	5	8	7	5	8	9	**6.92**	**6.4**
FEF1.2	**10**	**5**	**9**	**3**	**7**	**10**	**9**	**10**	**8**	**9**	**10**	**9**	**9**	**8.31**	**4.6**
FEF1.2.1	10	5	9	5	7	10	9	10	8	9	10	9	8	**8.38**	**3.1**
FEF1.2.2	10	8	5	5	7	9	9	10	8	9	10	8	10	**8.31**	**3.1**
FEF1.3	**8**	**2**	**7**	**1**	**10**	**8**	**6**	**8**	**6**	**8**	**2**	**9**	**9**	**6.46**	**8.8**
FEF1.3.1	7	2	7	1	4	7	6	8	6	7	2	9	8	**5.69**	**6.7**
FEF1.3.2	7	2	5	1	10	8	6	5	6	9	2	9	10	**6.15**	**9.5**
FEF1.4	**9**	**10**	**9**	**5**	**5**	**8**	**8**	**8**	**8**	**8**	**5**	**6**	**8**	**7.46**	**2.8**
FEF1.4.1	8	5	5	1	3	10	6	10	8	4	5	6	9	**6.15**	**7.5**
FEF1.4.2	9	5	7	5	5	8	9	8	6	7	5	6	10	**6.92**	**3.1**
FEF1.4.3	9	5	5	5	7	8	8	8	8	9	5	6	5	**6.77**	**2.7**
FEF1.4.4	10	5	5	5	7	9	6	10	7	10	5	6	5	**6.92**	**4.4**
FEF1.4.5	10	5	9	5	3	8	9	6	9	10	5	6	10	**7.31**	**5.7**
FEF1.5	**10**	**8**	**6**	**10**	**7**	**7**	**7**	**6**	**6**	**8**	**10**	**8**	**10**	**7.92**	**2.6**
FEF1.5.1	10	10	8	2	3	7	7	5	8	8	10	8	10	**7.38**	**6.9**
FEF1.5.2	10	5	8	2	7	7	7	8	8	8	10	8	10	**7.54**	**4.8**
FEF2.1	**10**	**8**	**8**	**10**	**4**	**7**	**9**	**8**	**9**	**6**	**10**	**8**	**9**	**8.15**	**3.0**
FEF2.1.1	10	6	6	10	3	8	9	8	9	7	10	10	10	**8.15**	**4.6**
FEF2.1.2	9	6	6	10	3	8	5	6	7	6	8	8	10	**7.08**	**4.1**
FEF2.1.3	7	10	9	8	6	10	8	8	9	7	6	8	10	**8.15**	**2.0**
FEF2.1.4	10	5	8	10	5	6	9	6	7	7	8	8	8	**7.46**	**2.8**
FEF2.1.5	9	9	6	10	4	6	7	5	7	4	6	7	7	**6.69**	**3.4**
FEF2.2	**9**	**8**	**8**	**5**	**7**	**8**	**7**	**7**	**8**	**7**	**5**	**7**	**10**	**7.38**	**1.9**
FEF2.2.1	9	6	8	10	7	10	7	8	8	9	5	7	10	**8.00**	**2.5**

FEF2.2.2	8	6	8	5	3	7	8	7	8	2	5	7	10	**6.46**	**4.9**
FEF2.2.3	9	8	8	5	5	6	8	7	8	9	5	7	10	**7.31**	**2.7**
FEF2.3	**10**	**10**	**8**	**8**	**7**	**7**	**6**	**10**	**7**	**9**	**8**	**8**	**10**	**8.31**	**1.9**
FEF2.3.1	8	10	8	9	3	9	7	10	9	9	10	9	9	**8.46**	**3.4**
FEF2.3.2	9	5	8	5	7	7	6	9	7	7	8	8	10	**7.38**	**2.3**
FEF2.3.3	9	6	8	10	6	8	6	9	7	10	8	7	10	**8.00**	**2.3**
FEF2.4	**10**	**7**	**7**	**10**	**3**	**7**	**7**	**9**	**10**	**5**	**8**	**7**	**9**	**7.62**	**4.3**
FEF2.4.1	8	5	7	2	4	9	6	3	10	0	8	6	6	**5.69**	**8.2**
FEF2.4.2	9	5	5	10	5	7	7	8	10	9	8	7	10	**7.69**	**3.6**
FEF2.4.3	8	10	8	5	7	0	7	10	10	7	10	8	10	**7.69**	**7.9**
FEF3.1	**9**	**7**	**6**	**3**	**5**	**9**	**8**	**6**	**8**	**8**	**8**	**8**	**10**	**7.31**	**3.6**
FEF3.1.1	10	5	6	3	3	9	8	5	8	7	8	6	10	**6.77**	**5.5**
FEF3.1.2	9	7	6	3	3	10	7	7	8	7	8	8	10	**7.15**	**4.8**
FEF3.1.3	9	7	6	0	3	7	7	5	7	10	8	8	9	**6.62**	**7.3**
FEF3.1.4	8	8	5	3	7	10	6	7	9	7	6	8	9	**7.15**	**3.5**
FEF3.2	**9**	**10**	**7**	**10**	**8**	**7**	**7**	**9**	**8**	**7**	**8**	**8**	**10**	**8.31**	**1.4**
FEF3.2.1	10	8	7	0	7	7	6	10	7	7	8	9	10	**7.38**	**6.8**
FEF3.2.2	9	7	7	10	8	9	9	8	8	7	8	8	10	**8.31**	**1.1**
FEF3.2.3	10	6	7	10	8	9	9	7	8	7	8	8	8	**8.08**	**1.4**
FEF3.2.4	10	9	7	10	9	7	7	10	9	8	10	8	10	**8.77**	**1.5**
FEF3.3	**10**	**5**	**6**	**10**	**8**	**9**	**6**	**8**	**10**	**8**	**5**	**8**	**10**	**7.92**	**3.6**
FEF3.3.1	9	5	6	0	7	10	6	8	10	8	5	8	10	**7.08**	**7.7**
FEF3.3.2	8	3	6	10	8	8	6	9	10	8	5	8	10	**7.62**	**4.4**

Appendix E

This appendix presents a problem used in Case studies C and D in order to test the RE-tools. The problem depends to the electricity domain and it describes the process of electrical fault handling. In the following a small set of requirements, captured for this problem is listed.

The electricity utility company monitors their network and restores electricity faults by sending the emergency repair truck to the accident place. The company want to computerize this process, that it would be possible to get guidelines how to manage communication between emergency repair trucks and the service centre, how to organize man power and provide support for decision making when electricity fault happens in the network.

The typical sequence of actions during electricity fault: when an electric power interruption occurs, customers of the electricity utility company call a service centre to report service unavailability; based on the customer call, a work order (ticket) is created; the ticket is assigned to an emergency repair truck for further investigation and repair; after the emergency repair truck restores the power supply, it informs the service centre about the done work.

1.3 DEFINITIONS

1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.

1.3.2 Service centre - the organization, which provides electricity power supply to the customer.

1.3.3 Executive - a manager, who works in the service centre and coordinates the fault handling process.

1.3.4 System - Support Electricity Fault Restore system - the information system, used by the service centre to manage the faults and the ERTs

1.3.5 ERT - emergency repair truck - the team which handles the electricity faults and restore power supply to the customer in the place of accident.

3. ENGINEERING REQUIREMENTS

3.1 TICKET PRINTING

3.1.1 System should print the ticket. When the electric fault happen in the network, customer phones to the service centre. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket.

3.1.2 System should handle the customer call. When the electric fault happens the customer calls to the service centre. System should recall to customer phone and accept the information.

3.1.3 System should register the call to the database. After receiving the information system should evaluate if it is an emergency call. If yes, the call should be registered to the system database.

3.1.4 System should inform executive in the service centre. When the call is registered in the database, system should send an e-mail to the responsible executive within the service centre.

3.1.5 Executive should print the ticket about electric fault in the network. When the executive receive the ticket about the fault, he or she prints it out for further procedures.

3.2 SEND TICKET TO THE ERT

3.2.1 Executive should send ticket to ERT. Executive should send the ticket to the ERT which is the closest to the accident place and it is not busy with other fault handlings.

3.2.2 The system should monitor the geographical places of ERT. The system gathers the information about the ERT automatically. ERT depends on to the information, posed by service centre.

3.2.3 Executive should select the ERT. Executive evaluates the information about the ERTs, the service centre posses, and selects the most suitable (the closest to the accident place and not busy at that time moment) to handle the new electric fault.

3.2.4 Executive should send the ticket to the ERT. When the appropriate ERT is selected the ticket about electric fault is sent to the ERT by the executive.

3.3 THE ERT ARRIVES TO ACCIDENT PLACE

3.3.1 The ERT should arrive to the accident place. When the ERT receives the ticket, it should use the information from the ticket, fault history, and the geographical data about the area, kept in the system, in order to get to the accident place.

3.3.2 The ERT should request the information about fault history from the system. When the ERT gets the ticket, it investigates information and requests the information about related or similar faults from the system.

3.3.3 The ERT should request the information about geographical area from the system. When the ERT gets the ticket, it could request the system about the geographical area in order to get to the accident place faster.

3.3.4 The system should send all the requested information to the ERT. When system receives requests about the needed information (fault history, geographical area) it should submit it to the ERT.

3.4 THE ERT RESTORES THE POWER SUPPLY

3.4.1 The ERT should inform the service centre about power restore. The ERT register information about the fault in the system database. Also it should provide the information about the work done, about the restoration time.

3.4.2 The ERT should register the fault in the system database. ERT sends the information to system about the fault, its type, exact place and estimated time to repair.

3.4.3 The ERT should inform the service centre about moment the power supply is restore. The ERT sends the note that the fault is restored and the customer is getting the power supply again.

4. REQUIREMENTS TRACEABILITY

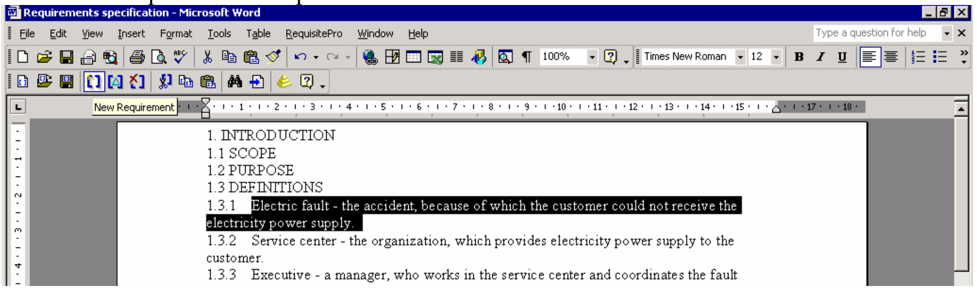
Appendix F

The appendix shows the fragment of the evaluation scenarios used in Case studies C and D. The scenarios are created to test three RE-tools (RDT, RequisitePro and CORE) on the predefined problem described in Appendix E.

RequisitePro

CAPTURING DEFINITIONS and REQUIREMENTS

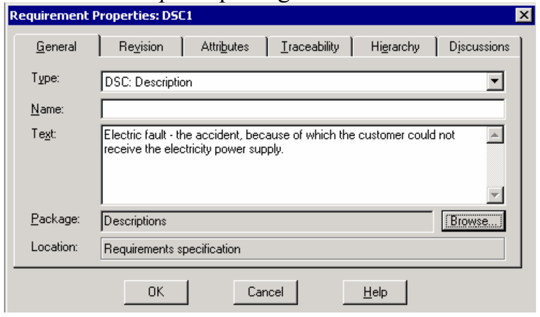
Definitions and requirements are provided in the lower case in the document.


Requirements specification - Microsoft Word
File Edit View Insert Format Tools Table RequisitePro Window Help
Type a question for help
100%
Times New Roman
12
New Requirement

1. INTRODUCTION
1.1 SCOPE
1.2 PURPOSE
1.3 DEFINITIONS
1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.
1.3.2 Service center - the organization, which provides electricity power supply to the customer.
1.3.3 Executive - a manager, who works in the service center and coordinates the fault

Figure F.1 Example of Requirements Document

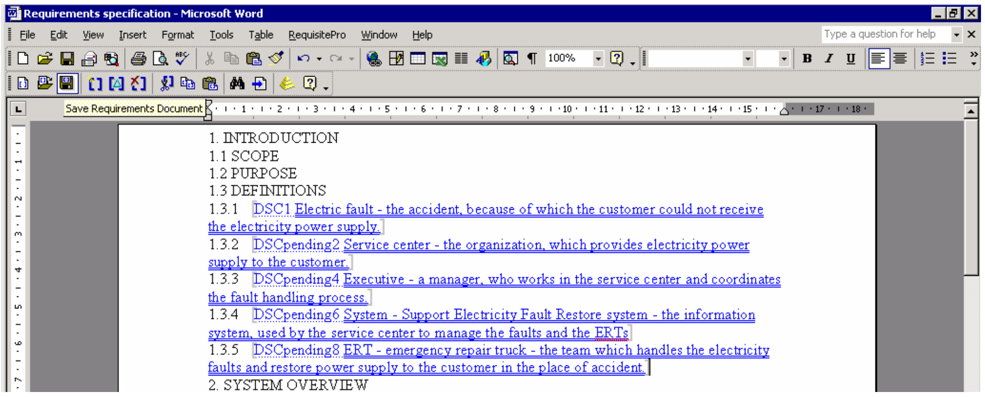
Select the 1.3.1 definition "*Electric fault -…*", press ***New*** requirement button in ***RequisitePro*** toolbar.
In ***Requirement Property*** window select:
Type: *DSC: Description*
Press ***Browse*** button to choose the *Description* package.



Type: DSC: Description
Name:
Text: Electric fault - the accident, because of which the customer could not receive the electricity power supply.
Package: Descriptions
Location: Requirements specification

Figure F.2 RequisitePro. Requirements Properties (1)

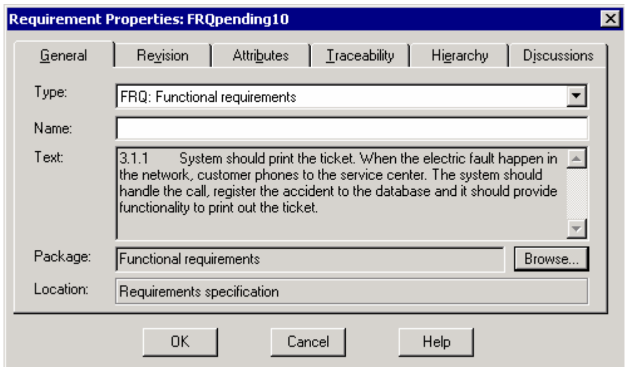
Press *OK* button to close ***Requirement Properties*** window.
Repeat the same actions to create *1.3.2., 1.3.3., 1.3.4., 1.3.5.* definitions.
Press button "***Save Requirements Document***" in RequisitePro toolbar.


Requirements specification - Microsoft Word
File Edit View Insert Format Tools Table RequisitePro Window Help
Type a question for help
100%
Save Requirements Document

1. INTRODUCTION
1.1 SCOPE
1.2 PURPOSE
1.3 DEFINITIONS
1.3.1 DSC1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.
1.3.2 DSCpending2 Service center - the organization, which provides electricity power supply to the customer.
1.3.3 DSCpending4 Executive - a manager, who works in the service center and coordinates the fault handling process.
1.3.4 DSCpending6 System - Support Electricity Fault Restore system - the information system, used by the service center to manage the faults and the ERTs.
1.3.5 DSCpending8 ERT - emergency repair truck - the team which handles the electricity faults and restore power supply to the customer in the place of accident.
2. SYSTEM OVERVIEW

Figure F.3 Selected Requirements in the Requirements Document

Use the same procedures to create functional requirements, defined in section
3. ENGINEERING REQUIREMENTS.



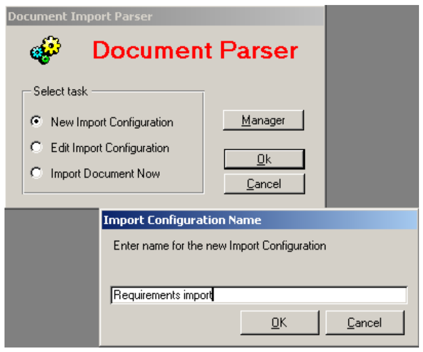
Type: FRQ: Functional requirements
Name:
Text: 3.1.1 System should print the ticket. When the electric fault happen in the network, customer phones to the service center. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket.
Package: Functional requirements
Location: Requirements specification

Figure F.4 RequisitePro. Requirements Properties (2)

Press button "Save *Requirements Document*" in RequisitePro toolbar.

RDT
CAPTURING REQUIREMENTS

Utility->Import document
In ***Document Import Parser***:
 New Import Configuration -> OK
 Enter import configuration name: '*Requirements import*", press ***OK***.



Select task: [✓ New Import Configuration] Edit Import Configuration Import Document Now

Enter name for the new Import Configuration: Requirements import

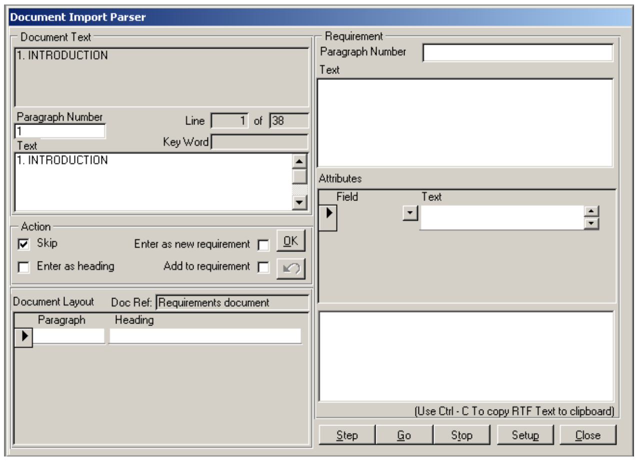
Figure F.5 RDT. Initiation of Requirements Import

Close the window ***Document Import Parser – Setup***.

Utility->Import document
In ***Document Import Parser***:

- ***Import Document Now -> OK***
- Select the requirements document '*requirements.txt*'
- In '***Document Parser***' window select "*Requirements import*" configuration, press *OK*.
- Enter requirements document reference - "*Requirements document*" in "***Document Reference***" window.
- Enter name for requirements database – "*RDTrequirementsModel*", press ***"Save***".

'***Document Import Parser'*** window

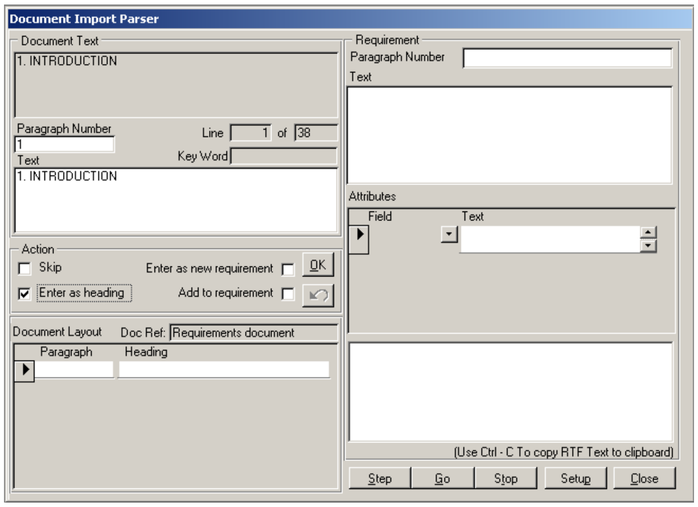


Document Text: 1. INTRODUCTION
Paragraph Number: 1 Line 1 of 38
Key Word:
Text: 1. INTRODUCTION
Action: [✓ Skip] Enter as heading Enter as new requirement Add to requirement
Doc Ref: Requirements document
Requirement Paragraph Number:
(Use Ctrl - C To copy RTF Text to clipboard)

Figure F.6 RDT. Document Import Parse Window (1)

All the text in upper case letters consider as the document heading and "*Enter as heading*" with appropriate paragraph number.
All the text in lower case letters consider as the requirements and "*Enter as new requirement*" or "*Add to requirement*", if it is the additional information about the requirement.
For example:

Paragraph Number: *1*
Text: *1.INTRODUCTION*
Action: *Enter as heading*
Press ***OK***



Document Import Parser

Document Text: 1. INTRODUCTION
Paragraph Number: 1
Line 1 of 38
Key Word:
Text: 1. INTRODUCTION

Action
☐ Skip
☑ Enter as heading
Enter as new requirement ☐
Add to requirement ☐

Document Layout
Doc Ref: Requirements document

Paragraph	Heading

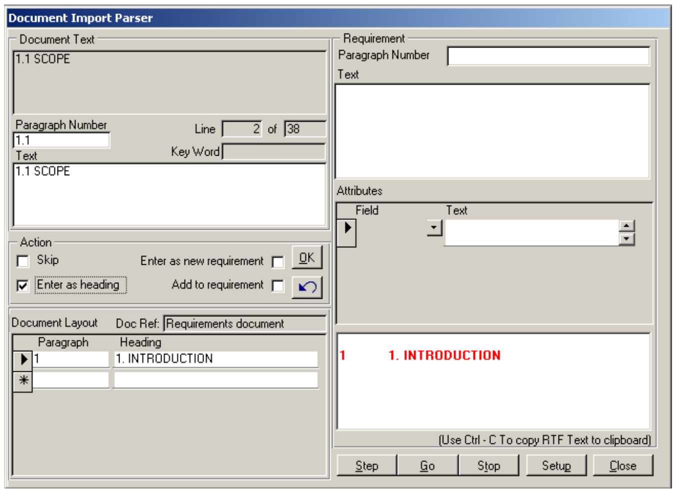
Requirement
Paragraph Number:
Text:
Attributes

Field	Text

(Use Ctrl - C To copy RTF Text to clipboard)

Figure F.7 RDT. Document Import Parse Window (2)

Paragraph Number: *1.1*
Text: *1.1 SCOPE*
Action: *Enter as heading*
Press ***OK***



Document Import Parser

Document Text: 1.1 SCOPE
Paragraph Number: 1.1
Line 2 of 38
Key Word:
Text: 1.1 SCOPE

Action
☐ Skip
☑ Enter as heading
Enter as new requirement ☐
Add to requirement ☐

Document Layout
Doc Ref: Requirements document

Paragraph	Heading
1	1. INTRODUCTION

Requirement
Paragraph Number:
Text:
Attributes

Field	Text

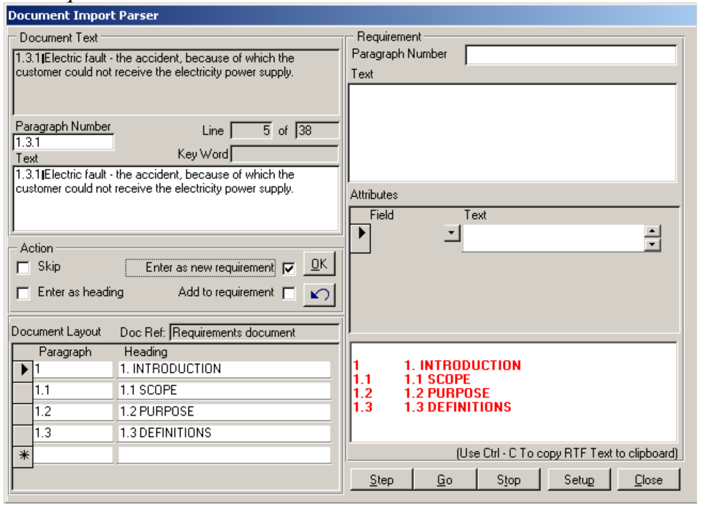
1 1. INTRODUCTION

(Use Ctrl - C To copy RTF Text to clipboard)

Figure F.8 RDT. Document Import Parse Window (3)

Paragraph: *1.3.1*
Text: *1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity*

power supply.
Action: *Enter as new requirement* Press ***OK***



Document Import Parser

Document Text: 1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.

Paragraph Number: 1.3.1 Line 5 of 38
Key Word:
Text: 1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.

Action: Skip [ ] Enter as new requirement [✓] Enter as heading [ ] Add to requirement [ ]

Document Layout Doc Ref: Requirements document

Paragraph	Heading
1	1. INTRODUCTION
1.1	1.1 SCOPE
1.2	1.2 PURPOSE
1.3	1.3 DEFINITIONS

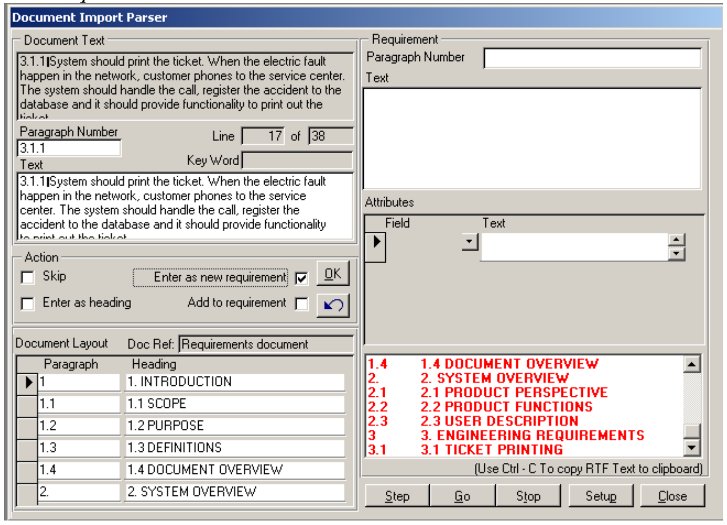
Requirement Paragraph Number:
Text:
Attributes Field Text

```
1       1. INTRODUCTION
1.1     1.1 SCOPE
1.2     1.2 PURPOSE
1.3     1.3 DEFINITIONS
```

(Use Ctrl - C To copy RTF Text to clipboard)

Figure F.9 RDT. Document Import Parse Window (4)

Paragraph: *3.1.1*
Text: *3.1.1 System should print the ticket. When the electric fault happen in the network, customer phones to the service centre. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket.*
Action: *Enter as new requirement* Press ***OK***



Document Import Parser

Document Text: 3.1.1 System should print the ticket. When the electric fault happen in the network, customer phones to the service center. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket

Paragraph Number: 3.1.1 Line 17 of 38
Key Word:
Text: 3.1.1 System should print the ticket. When the electric fault happen in the network, customer phones to the service center. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket

Action: Skip [ ] Enter as new requirement [✓] Enter as heading [ ] Add to requirement [ ]

Document Layout Doc Ref: Requirements document

Paragraph	Heading
1	1. INTRODUCTION
1.1	1.1 SCOPE
1.2	1.2 PURPOSE
1.3	1.3 DEFINITIONS
1.4	1.4 DOCUMENT OVERVIEW
2.	2. SYSTEM OVERVIEW

Requirement Paragraph Number:
Text:
Attributes Field Text

```
1.4     1.4 DOCUMENT OVERVIEW
2.      2. SYSTEM OVERVIEW
2.1     2.1 PRODUCT PERSPECTIVE
2.2     2.2 PRODUCT FUNCTIONS
2.3     2.3 USER DESCRIPTION
3       3. ENGINEERING REQUIREMENTS
3.1     3.1 TICKET PRINTING
```

(Use Ctrl - C To copy RTF Text to clipboard)

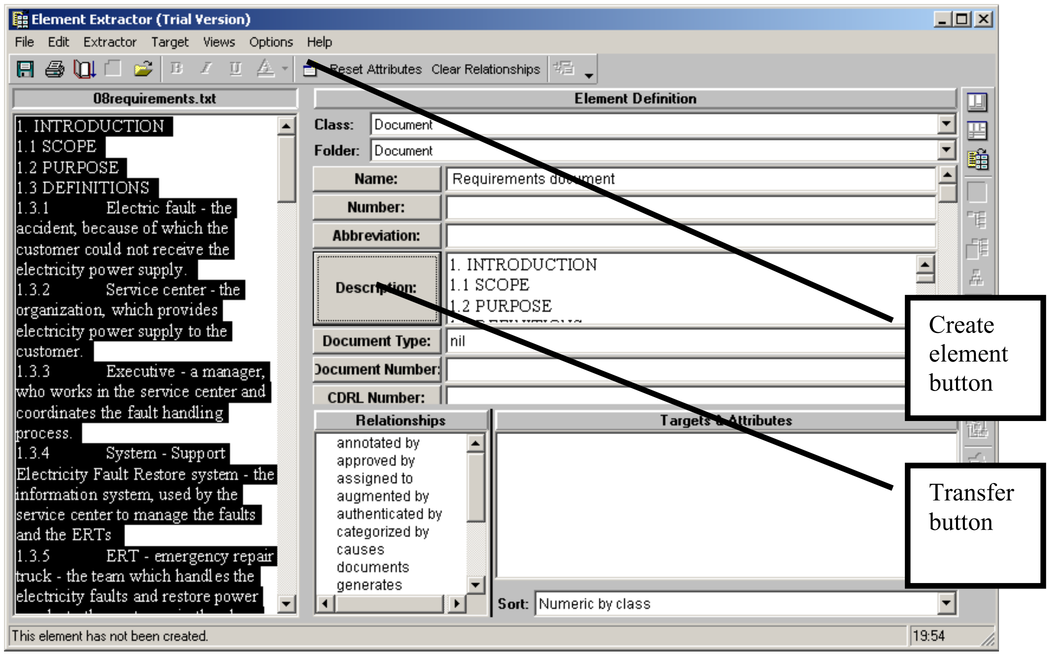
Figure F.10 RDT. Document Import Parse Window (5)

Press "***Close***" after import of the whole document.

CORE
IMPORTING REQUIREMENTS

From ***CORE Control Panel*** -> ***General*** click ***ELEMENT EXTRACTOR*** button.
Select ***File -> Load Document***
Open *requirements.txt* document

Select the ***Document*** class from the ***Class*** pull-down selection list.
Write a name for the requirements document – *Requirements document*.
Highlight the entire contents of the requirements.txt (crtl+A)
Click the ***Description*** transfer button in the right pane to transfer the selected text to the ***Description*** attribute field.


Element Extractor (Trial Version)
File Edit Extractor Target Views Options Help
Reset Attributes Clear Relationships
08requirements.txt
Element Definition
Relationships
Targets & Attributes
Create element button
Transfer button

1. INTRODUCTION
1.1 SCOPE
1.2 PURPOSE
1.3 DEFINITIONS
1.3.1 Electric fault - the accident, because of which the customer could not receive the electricity power supply.
1.3.2 Service center - the organization, which provides electricity power supply to the customer.
1.3.3 Executive - a manager, who works in the service center and coordinates the fault handling process.
1.3.4 System - Support Electricity Fault Restore system - the information system, used by the service center to manage the faults and the ERTs
1.3.5 ERT - emergency repair truck - the team which handles the electricity faults and restore power

Class: Document
Folder: Document
Name: Requirements document
Number:
Abbreviation:
Description: 1. INTRODUCTION 1.1 SCOPE 1.2 PURPOSE
Document Type: nil
Document Number:
CDRL Number:

annotated by
approved by
assigned to
augmented by
authenticated by
categorized by
causes
documents
generates

Sort: Numeric by class

This element has not been created. 19:54

Figure F.11 CORE. Window of Element Extractor

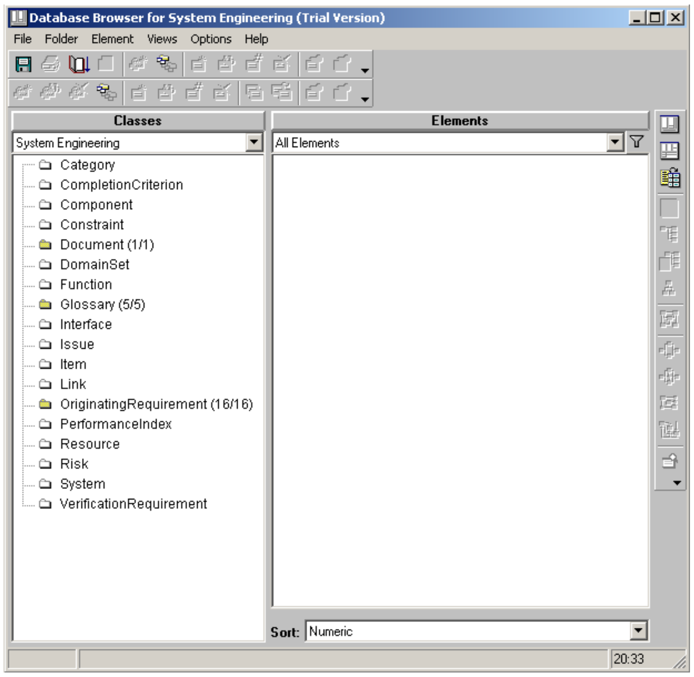
Click on the down arrow next to the ***Document Type*** field and choose ***Originating Requirements*** to reflect the source of this information.
Press ***Create*** element button.

Click ***Reset Attributes*** command on the toolbar

Change the ***Class*** to ***Originating Requirements***

Select the appropriate information for the requirements and repeat the same operation described above to capture requirements from the section 3.

Close ***Element Extractor*** window.
From ***CORE Control Panel*** -> ***General*** click ***DATABASE BROWSER*** button.


Database Browser for System Engineering (Trial Version)

File Folder Element Views Options Help

Classes: System Engineering
- Category
- CompletionCriterion
- Component
- Constraint
- Document (1/1)
- DomainSet
- Function
- Glossary (5/5)
- Interface
- Issue
- Item
- Link
- OriginatingRequirement (16/16)
- PerformanceIndex
- Resource
- Risk
- System
- VerificationRequirement

Elements: All Elements

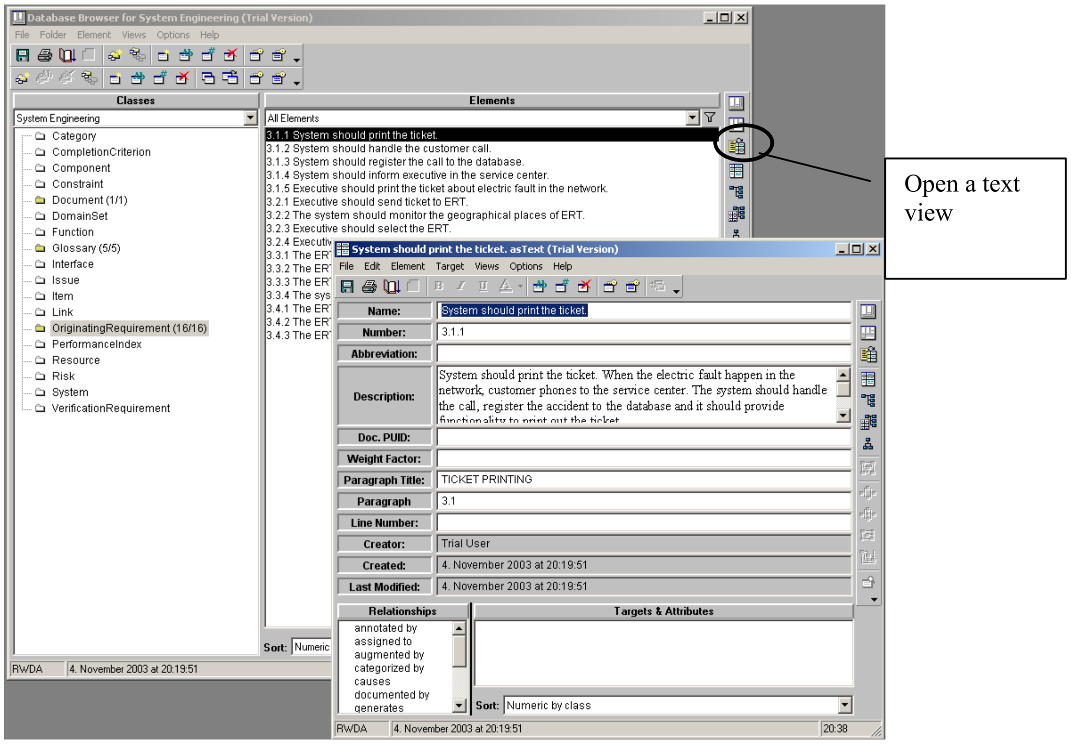
Sort: Numeric

20:33

Figure F.12 CORE. Database Browser Window

Select ***Originating requirements*** in ***Classes*** panel.

Select requirement *3.1.1*
Press a button ***Open a text view.***



3.1.1 System should print the ticket.
3.1.2 System should handle the customer call.
3.1.3 System should register the call to the database.
3.1.4 System should inform executive in the service center.
3.1.5 Executive should print the ticket about electric fault in the network.
3.2.1 Executive should send ticket to ERT.
3.2.2 The system should monitor the geographical places of ERT.
3.2.3 Executive should select the ERT.
3.2.4 Executiv
3.3.1 The ER
3.3.2 The ER
3.3.3 The ER
3.3.4 The sys
3.4.1 The ER
3.4.2 The ER
3.4.3 The ER

Name:	System should print the ticket.
Number:	3.1.1
Abbreviation:	
Description:	System should print the ticket. When the electric fault happen in the network, customer phones to the service center. The system should handle the call, register the accident to the database and it should provide functionality to print out the ticket
Doc. PUID:	
Weight Factor:	
Paragraph Title:	TICKET PRINTING
Paragraph	3.1
Line Number:	
Creator:	Trial User
Created:	4. November 2003 at 20:19:51
Last Modified:	4. November 2003 at 20:19:51

Figure F.13 CORE. Window of Element Properties

Appendix G

This appendix presents a raw material of Case study C described in Chapter 7. The table provides data on how different groups (group number in parenthesis) evaluated the RE-tool functionality at the Norwegian University of Science and Technology.

Table G.1 Raw Material of Case Study C

	RDT (1)	Requisite Pro (1)	CORE (1)	Requisite Pro (2)	CORE (2)	Requisite Pro (3)	RDT (3)	CORE (3)	Mean
FEF1.1.1	4	5	5	5	5	5	2	3	**4.25**
FEF1.1.2	5	5	1	5	4	5	3	3	**3.88**
FEF1.1.3	5	4	5	5	5	4	5	4	**4.63**
FEF1.1									**4.25**
FEF1.2.1	0	4	5	5	3	1.5	0	0.66	**2.40**
FEF1.2.2	4	5	5	5	0	0	0	0	**2.38**
FEF1.2									**2.39**
FEF1.3.1	0	4	0	0	0	4	0	0	**1.00**
FEF1.3.2	0	4	0	5	0	0	0	0	**1.13**
FEF1.3									**1.06**
FEF1.4.1	3	4	5	0	0	4.5	3	4	**2.94**
FEF1.4.2	4	5	5	5	5	4	4	4	**4.50**
FEF1.4.3	4	5	4	5	4	0	0	0	**2.75**
FEF1.4.4	3	3	0	4	0	4	3	4	**2.63**
FEF1.4.5	3	0	0	0	0	4	3.5	3.5	**1.75**
FEF1.4									**2.91**
FEF1.5.1	5	3	5	5	5	3.67	4	4	**4.33**
FEF1.5.2	0	5	5	5	5	3	1	0.75	**3.09**
FEF1.5									**2.68**
FEF2.1.1	5	5	0	4	0	0	0	0	**1.75**
FEF2.1.2	5	0	0	0	0	0	0	0	**0.63**
FEF2.1.3	0	0	0	0	0	4	1	4	**1.13**
FEF2.1.4	5	5	0	5	3	3.5	1.5	3.5	**3.31**
FEF2.1.5	5	5	0	4	3	4	2	0	**2.88**
FEF2.1									**1.94**
FEF2.2.1	4	4	5	4	0	2.4	2.4	3	**3.10**
FEF2.2.2	0	0	0	0	0	0	3	4	**0.88**
FEF2.2.3	0	0	0	0	0	4	2	3	**1.13**
FEF2.2									**1.70**
FEF2.3.1	0	0	0	5	5	0	0	0	**1.25**
FEF2.3.2	5	5	5	0	0	0	0	0	**1.88**
FEF2.3.3	4	5	5	0	5	0	4	0	**2.88**

FEF2.3									**2.00**
FEF2.4.1	4	0	5	0	0	0	0	0	**1.13**
FEF2.4.2	3	0	0	0	0	0	0	0	**0.38**
FEF2.4.3	2	0	0	5	5	5	4	4	**3.13**
FEF2.4									**1.54**
FEF3.1.1	4	5	5	0	0	4	3	4	**3.13**
FEF3.1.2	5	4	5	0	0	4	4	4	**3.25**
FEF3.1.3	0	0	5	0	0	3.5	0	0	**1.06**
FEF3.1.4	0	0	0	0	0	0	0	0	**0.00**
FEF3.1									**1.86**
FEF3.2.1	3	5	4	5	5	0	0	0	**2.75**
FEF3.2.2	5	5	5	5	5	5	4	4	**4.75**
FEF3.2.3	0	5	0	0	0	0	0	0	**0.63**
FEF3.2.4	0	0	0	5	4	3	4	4	**2.50**
FEF3.2									**2.66**
FEF3.3.1	3	5	5	4	5	0	0	0	**2.75**
FEF3.3.2	0	0	0	0	0	0	0	0	**0.00**
FEF3.3.3	4	0	4	4	4	0	4	4	**3.00**
FEF3.3									**1.92**

Appendix H

This appendix presents a raw material of Case study C described in Chapter 7. The appendix displays the questionnaire and the responses' frequencies. The questionnaire was used and the results were gathered at the Lund University.

1. Do you have any industrial requirements engineering experience?

YES	3
NO	27
	30

2. Have you used any of the tools before?

	NO	Yes, a little				Yes, a lot
DOORS	25	5				
CaliberRM	27	3				
RequisitePro	27	3				

3. Have you found the needed information in the Internet pages in order to answer the questions, provided to you in the "preparation assignments"?

	Not tested	Not sufficient				Sufficient	Mean
	0	**1**	**2**	**3**	**4**	**5**	
DOORS	1		3	11	7	8	3.57
CaliberRM	1		3	9	9	8	3.63
RequisitePro	1	1	6	11	7	4	3.13
OVERALL:	**3**	**1**	**12**	**31**	**23**	**20**	**3.44**

4. Please evaluate the functionality to represent/specify requirements.

	Not tested	Bad				Good	Mean
	0	**1**	**2**	**3**	**4**	**5**	
DOORS		1	11	10	6	2	2.90
CaliberRM	1		3	4	15	7	3.77
RequisitePro	1		3	15	11		3.17
OVERALL:	**2**	**1**	**17**	**29**	**32**	**9**	**3.28**

5. Do the RE-tools support requirements traceability?

	Not tested	Bad				Good	Mean
	0	**1**	**2**	**3**	**4**	**5**	

DOORS		2	10	6	9	3	3.03
CaliberRM				6	12	12	4.20
RequisitePro	1		2	9	14	4	3.57
OVERALL:	**1**	**2**	**12**	**21**	**35**	**19**	**3.60**

6. Do the RE-tools provide requirements prioritisation functionality?

	Not tested	**Bad**				**Good**	**Mean**
	0	**1**	**2**	**3**	**4**	**5**	
DOORS	5	7	12	1	4	1	1.83
CaliberRM	6	8	6	9	1		1.70
RequisitePro	7	7	3	10	3		1.83
OVERALL:	**18**	**23**	**23**	**23**	**12**	**6**	**2.40**

7. Do the RE-tools support collaboration activities between several users?

	Not tested	**Bad**				**Good**	**Mean**
	0	**1**	**2**	**3**	**4**	**5**	
DOORS	7		1	11	6	5	2.80
CaliberRM	12			6	7	5	2.37
RequisitePro	15	3	4	4	4		1.30
OVERALL:	**34**	**3**	**5**	**21**	**17**	**10**	**2.16**

8. Do the RE-tools support printing reports (requirements discussions, requirements specifications, wizards to generate reports, etc.)?

	Not tested	**Bad**				**Good**	**Mean**
	0	**1**	**2**	**3**	**4**	**5**	
DOORS	14		1	7	7	1	1.87
CaliberRM	12			11	7		2.03
RequisitePro	14		1	10	3	2	1.80
OVERALL:	**40**	**0**	**2**	**28**	**17**	**3**	**1.90**

9. How difficult was it to understand the functionality of the RE-tool?

	Not tested	**Difficult**				**Easy**	**Mean**
	0	**1**	**2**	**3**	**4**	**5**	
DOORS		6	9	6	6	3	2.70
CaliberRM			1	6	18	5	3.90
RequisitePro			6	13	11		3.17
OVERALL:	**0**	**6**	**16**	**25**	**35**	**8**	**3.26**

10. Please evaluate the usability of the RE-tools. (Usability considers how easy it is to learn and use the RE-tool)

	Not tested	**Difficult**				**Easy**	**Mean**
	0	**1**	**2**	**3**	**4**	**5**	
DOORS	1	4	15	3	5	2	2.43
CaliberRM			1	7	17	5	3.87
RequisitePro		1	5	16	8		3.03
OVERALL:	**1**	**5**	**21**	**26**	**30**	**7**	**3.11**

11. Do the RE-tools have interfaces with other tools?

	Office tools	Model-ling tools	Databas es	Web report
DOORS	19	2	10	11
CaliberRM	21	5	14	4
RequisitePro	28	13	14	5
None of the listed	1	1	1	2

12. Please evaluate which RE-tool that suits different types of organisations

	Large company	New immature company	Company, which produce many products	Company, which implies object oriented development paradigm
DOORS	17	5	8	6
CaliberRM	10	14	16	14
RequisitePro	16	10	10	8
None of the listed	1	2	1	2

13. Which of the RE-tools would you prefer to use, for example in your "KRAM-project"?

DOORS		7	
CaliberRM		16	3
RequisitePro		5	2
None		2	

Appendix I

The appendix presents the questionnaire used in Case study D. The questionnaire is implemented in a framework prototype described in Chapter 8. Questions 1 and 2 address the case study design. Questions 3-9 consider the quality types of the requirements specifications which was prepared for the problem provided in Appendix E. Question 10 analyses the usability and usefulness of evaluation techniques. Question 11 considers the usability and usefulness of different tools applied in the case study to support the RE process.

1. In what order were RE-tools used in the exercise? (Check first tool at 1, second tool at 2 and third tool at 3)

1.	◘ CORE	**2.**	◘ CORE	**3.**	◘ CORE
	◘ RDT		◘ RDT		◘ RDT
	◘ RequisitePro		◘ RequisitePro		◘ RequisitePro
	◘ CaliberRM		◘ CaliberRM		◘ CaliberRM
	◘ MS Office		◘ MS Office		◘ MS Office

2. Please check the evaluation techniques you were using together with the RE-tools.

	Evaluation scenario	**RE-tool tutorial**
CORE	◘	◘
RDT	◘	◘
RequisitePro	◘	◘
CaliberRM	◘	◘

REQUIREMENTS SPECIFICATION

The quality of the requirements specification is compared using the semiotic quality evaluation framework.

3. Compare the physical quality of the requirements specification.

	Very bad				**Very good**
	1	**2**	**3**	**4**	**5**
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the physical quality goals:

Externalisation ____

____

Internalisability____

____

4. Compare the empirical quality of the requirements specification.

	Very bad				**Very good**
	1	**2**	**3**	**4**	**5**
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the empirical quality goal:

Minimal error frequency ____

____

5. Compare the syntactic quality of the requirements specification.

	Very bad			Very good	
	1	2	3	4	5
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the syntactic quality goal:
Syntactic correctness ____

____

6. Compare the semantic quality of the requirements specification.

	Very bad			Very good	
	1	2	3	4	5
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the semantic quality goals:
Validity ____

____

Completeness ____

____

7. Compare the pragmatic quality of the requirements specification.

	Very bad			Very good	
	1	2	3	4	5
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the pragmatic quality goal:
Comprehension ____

____

8. Compare the perceived semantic quality of the requirements specification.

	Very bad			Very good	
	1	2	3	4	5
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the perceived semantic quality goals:
Perceived validity ____

____

Perceived completeness ____

____

9. Compare the social quality of the requirements specification.

	Very bad			Very good	
	1	2	3	4	5
RE-tool_1	◘	◘	◘	◘	◘
RE-tool_2	◘	◘	◘	◘	◘
MS Office	◘	◘	◘	◘	◘

Please identify defects of the requirements specifications in respect to the social quality goal:
Agreement ____

____

EVALUATION TECHNIQUES

Two evaluation techniques are used in this exercise. They are evaluation scenario and RE-tool tutorials.

10. Which evaluation technique contributed to better:

Evaluation scenario		RE-tool tutorial
	understandability of the RE-tool functionality?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	learning of the RE-tool functionality?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	understandability of the problem, to which requirements specifications were prepared?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
Evaluation scenario	**preparation of the requirements specification?**	**RE-tool tutorial**
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	satisfaction of the RE-tool usage?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	quality of the requirements specification?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	Evaluation of the RE-tools?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	

1 – evaluation scenario much better, 2 – evaluation scenario better, 3 – evaluation scenario slightly better
4 – tool tutorial slightly better, 5 – tool tutorial better, 6 – tool tutorial much better

RE-TOOLS

11. Which tools do provide better:

	understandability of the RE process?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	support for the RE activities?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	functionality to prepare the requirements specification?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
RE-tools	**means to ensure the quality of the requirements specification?**	**MS Office**
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	satisfaction of the tool usage in RE?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	usability to execute different RE activities?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	
	changeability and traceability of the requirements and requirements specification?	
	1◘ 2◘ 3◘ 4◘ 5◘ 6◘	

1 – RE-tools much better, 2 – RE-tools better, 3 – RE-tools slightly better
4 – MS Office slightly better, 5 – MS Office better, 6 – MS Office much better

Appendix J

The appendix presents the raw material of the Case study D described in Chapter 7.

Table J.1 provided the tool evaluation results. The results are calculated by the framework prototype applied in the case study. Table J.2 shows the results of the requirements specification quality evaluation. Table J.3 displays the raw material of the evaluation techniques assessment according to the questionnaire in Appendix I (see question 10). Table J.4 displays the raw material of the RE process automated support according to the questionnaire in Appendix I (see question 11).

Table J.1 Evaluation of the RE-tool Functionality
The score is calculated as the sum of the RE-tool function evaluations according to the evaluation framework.

Group	RE-tools	Group members					Total tool evaluation
		1	2	3	4	5	
Group 1	RequisitePro	916	740	805	510	798	3769
	RDT	915	719	846	374	854	3708
	MS Office	884	330	774	123	330	2441
Group 2	CaliberRM	1623	1188	1406	1366		5583
	CORE	1437	1073	1236	1098		4844
	MS Office	1300	908	1194	1046		4448
Group 3	CaliberRM	1370	751	1337	952	968	5378
	CORE	818	631	818	1057	*831*	4155
	MS Office	*587*	626	*587*	378	756	2933
Group 4	RequisitePro	444	309	340	373		1466
	CORE	404	281	340	371		1396
	MS Office	275	251	122	157		805
Group 5	CORE	600	451	563	551		2165
	CaliberRM	582	365	416	487		1850
	MS Office	442	781	400	615		2238
Group 6	RequisitePro	822	723	660	635	731	3571
	CaliberRM	995	929	1053	731	828	4536
	Latex	853	896	934	299	185	3167
Group 7	RDT	274	256	267	294	247	1338
	CORE	373	268	329	275	251	1496
	MS Office	326	287	447	279	281	1620
Group 8	RDT	479	345	559	243		1626
	CORE	420	259	546	431		1656
	MS Office	187	370	290	326		1173

Group 9	RDT	995	335	667	712	459	3168
	RequisitePro	1035	410	1041	1253	516	4255
	MS Office	628	470	628	427	307	2460
Group 10	RDT	1023	1107	1042	1036		4208
	CaliberRM	1274	1308	1310	1216		5108
	MS Office	580	476	506	571		2133

Table J.2 Quality Evaluation of the Requirements Specification
The score is calculated as the sum of the quality types (see questions 3-9 in Appendix I).

Group	RE-tools	Quality type							Overall quality
		Physical	Empirical	Syntactic	Semantic	Pragmatic	Perceived semantic	Social	
Group 1	RequisitePro	17	18	22	19	21	17	14	128
	RDT	18	15	21	17	17	17	14	119
	MS Office	19	16	13	13	13	14	10	98
Group 2	CaliberRM	14	12	18	18	12	9	19	102
	CORE	12	16	20	16	15	9	13	101
	MS Office	20	20	18	15	20	19	15	127
Group 3	CaliberRM	9	10	12	12	11	13	14	81
	CORE	12	14	16	13	15	13	13	96
	MS Office	11	15	13	13	15	12	12	91
Group 4	RequisitePro	14	13	16	14	13	13	15	98
	CORE	17	14	16	17	13	13	15	105
	MS Office	14	12	13	11	12	11	14	87
Group 5	CORE	13	13	13	13	13	11	9	85
	CaliberRM	12	14	12	13	12	11	9	83
	MS Office	19	10	16	13	13	18	16	105
Group 6	RequisitePro	14	16	22	15	14	15	14	110
	CaliberRM	15	16	21	18	13	13	13	109
	Latex	19	22	17	16	20	16	16	126
Group 7	RDT	14	14	12	17	15	15	12	99
	CORE	22	23	16	21	19	17	18	136
	MS Office	20	19	13	22	21	19	20	134
Group 8	RDT	10	10	10	11	12	8	9	70
	CORE	13	17	10	13	16	8	10	87
	MS Office	13	13	7	12	14	7	9	75
Group 9	RDT	20	17	16	17	16	13	15	114
	RequisitePro	24	22	20	20	21	21	20	148
	MS Office	23	22	20	19	19	19	13	135
Group 10	RDT	12	13	14	14	12	12	13	90
	CaliberRM	13	14	12	15	14	11	14	93
	MS Office	11	15	11	9	12	11	8	77

Table J.3 Raw Material of the Evaluation Technique Assessment (according to question 10 in Appendix I)

Group No.	1. Understandability of the RE-tool functionality?	2. Learning of the RE-tool functionality?	3. Understandability of the problem to which requirements specification were prepared?	4. Preparation of the requirements specification?	5. Satisfaction of the RE-usage?	6. Quality of the requirements specification?	7. Evaluation of the RE-tools?
Group1	1	2	3	3	3	3	2
	1	3	2	2	3	1	4
	5	5	3	3	4	5	5
	4	5	5	5	4	5	4
	6	6	2	5	6	3	6
Group2	5	5	3	4	2	2	3
	2	3	4	3	3	2	3
	5	1	2	1	3	4	2
	4	2	1	1	2	4	1
Group3	1	3	2	1	1	1	2
	6	5	3	4	3	2	5
	1	4	3	2	2	2	5
	5	5	3	2	4	2	2
Group4	2	2	3	3	2	2	3
	5	2	5	2	4	5	4
	2	3	4	1	2	1	3
	5	4	5	3	4	6	4
Group5	2	2	2	2	2	2	2
	3	3	2	2	3	1	1
	2	2	2	2	2	2	2
	5	6	2	4	5	2	5
Group6	5	2	2	2	4	4	4
	3	2	3	3	2	3	3
	4	5	1	2	5	4	6
	5	4	5	4	4	4	5
	5	3	5	3	6	4	4
Group7	1	2	4	4	2	2	1
	2	2	3	2	2	3	2
	5	4	3	4	5	5	2
	3	3	1	5	4	2	2
	5	5	5	4	2	2	2
Group8	5	6	1	2	1	2	5
	4	2	3	4	2	5	4
	5	4	4	3	5	4	3
	4	4	3	3	5	3	3
Group9	5	4	1	2	4	3	5
	5	4	3	3	5	5	3
	2	1	1	1	1	3	2
	3	4	5	2	2	4	6
	3	4	2	4	3	4	5
Group 10	6	3	4	6	1	6	3
	4	5	3	4	4	5	5
	2	4	2	2	5	2	4
	3	4	2	4	3	1	4
M	**3.66**	**3.50**	**2.89**	**2.91**	**3.20**	**3.11**	**3.43**

Table J.4 Raw Material of the RE Process Automated Support (according to question 11 in Appendix I)

Group	Understandability of the RE process	Support for the RE activities	Functionality to prepare the requirements specification	Means to ensure the quality of the requirements specification	Satisfaction of the tool usage during RE	Usability to execute different RE activities	Changeability and traceability of the requirements and requirements specification
Group1	2	2	2	1	2	2	3
	1	1	2	1	2	2	2
	2	1	3	2	3	2	3
	5	4	5	4	3	3	2
	1	1	6	1	6	6	1
Group2	6	6	3	2	6	2	2
	3	4	3	2	4	4	3
	1	2	3	4	5	3	2
	2	2	2	2	5	4	1
Group3	3	2	2	2	2	2	2
	2	3	2	3	4	3	2
	6	1	2	1	***2***	3	2
	2	2	1	2	4	3	2
Group4	3	2	2	2	2	1	1
	2	1	2	2	2	3	1
	3	2	2	3	3	4	1
	2	2	1	1	3	2	2
Group5	6	5	4	2	6	5	4
	5	2	5	2	6	6	2
	2	5	2	3	5	5	3
	2	5	4	2	4	5	2
Group6	1	1	3	2	3	2	1
	6	2	2	3	6	2	6
	6	1	5	1	6	2	6
	3	2	5	3	3	3	2
	5	4	2	5	4	3	4
Group7	3	3	2	4	2	2	1
	3	***3***	2	4	***3***	3	3
	3	2	4	1	3	3	5
	5	3	5	3	5	5	2
	3	3	3	4	4	4	4
Group8	2	4	2	5	1	3	1
	4	4	3	4	3	4	3
	2	2	3	2	3	2	3
	2	2	3	3	4	2	1
Group9	1	2	3	2	3	2	2
	4	4	4	3	3	2	1
	2	2	2	1	2	2	1
	2	1	2	2	4	1	1
	2	2	2	2	3	2	1
Group 10	1	3	2	3	4	4	3
	2	2	3	3	3	4	3
	1	1	1	1	1	1	1
	4	2	2	2	3	4	1
M	**2.91**	**2.50**	**2.80**	**2.43**	**3.53**	**3.00**	**2.25**

Term Index

3-activity model, 42
4-activity model, 43
achievable, 49, 169
actor and *role*, 23
agreement, 149, 190
annotated, 169
annotated by relative importance, 49
annotated by relative stability, 49
annotated by version, 49
application system, 18
at right level of detail, 49
at the right level of detail, 169
behavioural, 22
Capability Maturity Model (CMM), 31
commercial off-the-shelf (COTS), 88
commonly agreed, 41
communication, 23
complete, 49, 169
complete specification, 41
Comprehensibility appropriateness, 75
computer aided system engineering (CASE), 59, 61, 76
computerised organisational information system, 18
concise, 49, 169
concision, 48
consistent, 169
constructivistic, 19, 127
corporate feedback cycle, 28
correct, 49
COTS Acquisition Process (CAP), 92
coverage in process, 19
coverage in product, 19
cross-referenced, 49, 169
decision about the RE-tool selection, 134
Design, 10
design-independent, 49, 169
Development/Management model, 44
domain analysis, 22
Domain appropriateness, 75
electronically stored, 48, 169
empirical quality, 74, 108
evaluation framework, 78
evaluation scenarios, 133, 171, 180
evaluation team, 88, 131
evaluation techniques, 132, 150
executable (interpretable, prototypable), 49
experience factory, 29
Experience Factory (EF), 28
Experimentation, 10
external consistency, 49
external requirements, 120
externalise, 18
features, 101
formal language, 102, 105
fragment-based RE-tool (FB-RET), 210
framework for evaluation non-functional RE-tool requirements, 114
framework for evaluation of functional RE-tool requirements, 103
framework prototype, 180, 197
free selection, 78
functional, 22
functional requirements, 40, 112
functional RE-tool requirements, 102
goal/question/metric (GQM), 29
history, 107
importance, 149, 191

improvement, 25
improvement of something by means of the process, 25
improvement of the process, 25
informal language, 102
informal representations, 41
Information System (IS), 5, 17
internalisation, 18
internally consistent, 49
investigation of the functional requirements, 132
investigation of the process requirements, 133
investigation of the product requirements, 133
ISO/IEC 15504, 33
ISO/IEC 9126 quality-based approach, 95
knowledge-based, 22
Lang and Duggan requirements, 79
local reality, 18
maintainability, 119
maintenance of requirements specification, 151, 161
management level, 210
maturity, 20
mentalistic, 19
modelling tools, 6
modifiable, 49
natural language, 105
NATURE framework, 41, 75
non-functional requirements, 40, 112
non-functional RE-tool requirements, 102
not redundant, 49
object, 23
objectivistic, 19
office tools, 6, 150, 158, 180
Off-The-Shelf Option (OTSO), 91
operational, 20
organisational quality, 74
organisational reality, 18
organisational requirements, 120
organised, 49, 169
Participant language knowledge appropriateness, 75
perceived semantic quality, 74, 110
performance, 118
personal views, 41
perspective, 22
physical quality, 74, 108
Plan-Do-Check-Act (PDCA), 26
pragmatic quality, 74, 110
precise, 49
precision, 169
preparation of a requirements specification, 131
priority-based framework, 82
process, 24
Process improvement, 5, 24
Procurement-Oriented Requirements Engineering (PORE), 89
Product requirements, 116
professional language, 102, 105
project feedback cycle, 28
project organisation, 28
prototyping, 20
quality improvement, 59
Quality Improvement Paradigm (QIP), 28
quality of the requirements specification, 48, 167, 182
rate of not tested, 191
rationale, 107
RE process, 45
reliability, 117
repository level, 213
requirement, 39
requirements agreement dimension, 41, 106, 165
requirements analysis and negotiation, 44
requirements documentation, 44
requirements elicitation, 42, 43
Requirements engineering (RE), 5, 40
Requirements engineering tools (RE-tools), 3, 61
requirements for business parties, 120
requirements management, 44
requirements management tools, 61
requirements of the evaluation process, 6, 114

requirements representation dimension, 41, 104, 164
requirements specification, 42, 47, 167
requirements specification dimension, 41, 106, 166
requirements validation, 43, 44
resource reduction, 59
RE-tool evaluation approach (R-TEA), 129, 131
RE-tool tutorials, 133, 171, 180
Reusability, 48
reusable, 169
reuse, 19
role-based framework, 82
rule, 22
scenario-based COTS selection, 92
schedule reduction, 59
selection of business parties, 132
semantic quality, 74, 110
semiformal language, 102, 105
semiotic quality framework, 74
social quality, 74, 111
Social Technical Approach to COTS Evaluation (STACE), 94
specification exemplar, 129, 133
spiral, 20
stakeholders, 19
structural, 22
syntactic quality, 74, 108
Technical actor interpretation appropriateness, 75
Theoretical approach, 10
tool weaknesses, 61, 150, 157
Total Quality Management (TQM), 26
traceable, 49, 169
traced, 49, 169
transformational, 21
unambiguous, 49, 169
understandable, 49, 169
understantability, 48
usability, 116
user identification level, 214
users, 88, 131
validation, 111
verifiable, 49
verification, 111
waterfall, 20
Weltanschauung, 19
work avoidance, 3
working faster, 3
working smarter, 3