SOFTWARE EVOLUTION
AND FEEDBACK

Theory and Practice

Edited by

Nazim H. Madhavji
*University of Western Ontario, Canada*

Juan C. Fernández-Ramil
*The Open University, UK*

and

Dewayne E. Perry
*The University of Texas at Austin, USA*
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>xix</td>
</tr>
<tr>
<td>Preface</td>
<td>xxi</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>xxvii</td>
</tr>
<tr>
<td>Editors' Biographies</td>
<td>xxix</td>
</tr>
<tr>
<td>List of Contributors</td>
<td>xxxi</td>
</tr>
</tbody>
</table>

### PART ONE SOFTWARE EVOLUTION

#### 1 Software Evolution

*Meir Lehman and Juan C. Fernández-Ramil*

1.1 Introduction 7
  1.1.1 Evolution 7
  1.1.2 Interpretation of the Term Evolution in the Context of Software 8
1.2 The Evolution of Large Software Systems 8
  1.2.1 Early Work 8
  1.2.2 Large Programs 9
1.3 Program Classification 10
  1.3.1 The SPE Program Classification Schema 10
  1.3.2 S-type Applications and Software 10
  1.3.3 E-type Applications and Software 12
  1.3.4 P-type Situations and Software 13
1.4 The Inevitability of Evolution 13
1.5 Levels of Software-Related Evolution 14
1.6 Ab Initio Implementation or Change 16
  1.6.1 Process Steps 16
  1.6.2 The LST Paradigm 17
  1.6.3 Phenomenological Analysis of Real-World Computer Usage 18
  1.6.4 Theoretical Underpinning 18
  1.6.5 The Value of Formalisms and of Verification 19
  1.6.6Bounding 20
  1.6.7 The Consequence: Continual System Evolution 21
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.8 Summary</td>
<td>21</td>
</tr>
<tr>
<td>1.6.9 Principle of Software Uncertainty</td>
<td>22</td>
</tr>
<tr>
<td>1.7 Software Systems Evolution</td>
<td>22</td>
</tr>
<tr>
<td>1.7.1 Early Work</td>
<td>22</td>
</tr>
<tr>
<td>1.7.2 FEAST</td>
<td>23</td>
</tr>
<tr>
<td>1.7.3 The Growth Trend</td>
<td>24</td>
</tr>
<tr>
<td>1.7.4 Evolution Drivers</td>
<td>25</td>
</tr>
<tr>
<td>1.7.5 Relationship Between the Above Levels of Evolution</td>
<td>26</td>
</tr>
<tr>
<td>1.7.6 Evolutionary Development</td>
<td>26</td>
</tr>
<tr>
<td>1.8 Evolution of the Application and Its Domain</td>
<td>27</td>
</tr>
<tr>
<td>1.9 Process Evolution</td>
<td>28</td>
</tr>
<tr>
<td>1.9.1 Software Processes as Systems</td>
<td>28</td>
</tr>
<tr>
<td>1.9.2 Process Improvement</td>
<td>28</td>
</tr>
<tr>
<td>1.9.3 The Theoretical Approach</td>
<td>29</td>
</tr>
<tr>
<td>1.9.4 Evolving Specifications</td>
<td>30</td>
</tr>
<tr>
<td>1.9.5 The Empirical Approach</td>
<td>30</td>
</tr>
<tr>
<td>1.9.6 Laws of Software Evolution</td>
<td>30</td>
</tr>
<tr>
<td>1.9.7 The Ephemeral Process</td>
<td>31</td>
</tr>
<tr>
<td>1.10 Process Model Evolution</td>
<td>32</td>
</tr>
<tr>
<td>1.10.1 The Nature of the Software Process</td>
<td>32</td>
</tr>
<tr>
<td>1.10.2 Process Models</td>
<td>32</td>
</tr>
<tr>
<td>1.10.3 Software Process Models</td>
<td>33</td>
</tr>
<tr>
<td>1.10.4 Process Improvement</td>
<td>33</td>
</tr>
<tr>
<td>1.10.5 Links Between Process and Process Model Evolution</td>
<td>35</td>
</tr>
<tr>
<td>1.11 Relationships Between Levels</td>
<td>35</td>
</tr>
<tr>
<td>1.11.1 The Software/Software Process Contrast</td>
<td>35</td>
</tr>
<tr>
<td>1.11.2 The Software Process/Process Model Contrast</td>
<td>36</td>
</tr>
<tr>
<td>1.12 Conclusions</td>
<td>37</td>
</tr>
<tr>
<td>1.13 Acknowledgments</td>
<td>37</td>
</tr>
<tr>
<td>References</td>
<td>37</td>
</tr>
</tbody>
</table>

2 A Nontraditional View of the Dimensions of Software Evolution 41

Dewayne E. Perry

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction</td>
<td>41</td>
</tr>
<tr>
<td>2.2 The Domains</td>
<td>42</td>
</tr>
<tr>
<td>2.2.1 The Real World and Its Model</td>
<td>42</td>
</tr>
<tr>
<td>2.2.2 The Model and the Derived Specification</td>
<td>43</td>
</tr>
<tr>
<td>2.2.3 Theory</td>
<td>43</td>
</tr>
<tr>
<td>2.3 Experience</td>
<td>44</td>
</tr>
<tr>
<td>2.3.1 Feedback</td>
<td>45</td>
</tr>
<tr>
<td>2.3.2 Experimentation</td>
<td>46</td>
</tr>
<tr>
<td>2.3.3 Understanding</td>
<td>46</td>
</tr>
<tr>
<td>2.4 Process</td>
<td>47</td>
</tr>
<tr>
<td>2.4.1 Methods</td>
<td>47</td>
</tr>
<tr>
<td>2.4.2 Technology</td>
<td>48</td>
</tr>
<tr>
<td>2.4.3 Organization</td>
<td>48</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>49</td>
</tr>
<tr>
<td>2.6 Acknowledgments</td>
<td>50</td>
</tr>
<tr>
<td>References</td>
<td>51</td>
</tr>
</tbody>
</table>
3 IT Legacy Systems: Enabling Environments That Reduce
the Legacy Problem: A Complexity Perspective 53
Professor Eve Mitleton-Kelly

3.1 Introduction 53

3.2 The Legacy Problem 55
  3.2.1 Feedback 56
  3.2.2 Co-evolution 56
  3.2.3 The Social Ecosystem 57

3.3 The Two Case Studies 58
  3.3.1 Business and Market 60
  3.3.2 Organisation and Management 61
  3.3.3 Technology 62
  3.3.4 Interactions between the Various Elements 62

3.4 The Socio-Technical Enabling Environment 64
  3.4.1 The Bank’s Enabling Environment 64
  3.4.2 The Building Society and Some Complexity Principles 66

3.5 Summary and Conclusions 68
3.6 Acknowledgements 69

References 69

4 Facets of Software Evolution 71
Roland T. Mittermeir

4.1 Introduction 71

4.2 What is Software? 72
  4.2.1 Software: A Technical Artefact 72
  4.2.2 Software: A Utility 72
  4.2.3 Software: A Text, Reflecting Reality 73
  4.2.4 Software is Information 74

4.3 Evolution 75
  4.3.1 Principles 75
  4.3.2 Evolution Drivers 76

4.4 Strata of Software-Size and Complexity 77
  4.4.1 Module 77
  4.4.2 Design Unit 78
  4.4.3 Architecture 79
  4.4.4 System 80
  4.4.5 System-of-Systems 80
  4.4.6 Discussion 81

4.5 Approaches to (R-)evolve 82
  4.5.1 Changes in Modules 83
  4.5.2 Modifying Design Units 84
  4.5.3 Evolution on the Architectural Level 84
  4.5.4 System-Level Evolution 84
  4.5.5 Evolution of Systems-of-Systems 85

4.6 An Example 86
  4.6.1 A System-of-Systems? 86
  4.6.2 System-Level Changes 87
  4.6.3 Architectural Decisions 88
  4.6.4 Design Units 88
11 A Study of Software Evolution at Different Levels of Granularity

Elizabeth Burd

11.1 Introduction
11.2 Existing Studies of Software Evolution
11.3 Case Study Approach
11.4 Results
11.4.1 The System Level
11.4.2 Level 2, The Function Level
11.4.3 Level 3, The Data Level
11.4.4 Comparing Levels
11.5 General Recommendations
11.6 Conclusions

References

12 The Role of Ripple Effect in Software Evolution

Sue Black

12.1 Introduction
12.2 Impact Analysis
12.3 Software Maintenance and Software Maintenance Models
12.4 Background on the Ripple Effect
12.4.1 Computation of the Ripple Effect
12.4.2 The REST Software Tool
12.5 Links Between Ripple Effect and the Laws of Software Evolution
12.5.1 First Law – Continuing Change
12.5.2 Second Law – Growing Complexity
12.5.3 Third Law – Self Regulation
12.5.4 Fourth Law – Conservation of Organisational Stability
12.5.5 Fifth Law – Conservation of Familiarity
12.5.6 Sixth Law – Continuing Growth
12.5.7 Seventh Law – Declining Quality
12.5.8 Eighth Law – Feedback System
12.6 Conclusions
12.7 Further Work
12.8 Acknowledgements

References

13 The Impact of Software-Architecture Compliance on System Evolution

R. Mark Greenwood, Ken Mayes, Wykeen Seet, Brian C. Warboys, Dharini Balasubramaniam, Graham Kirby, Ron Morrison and Aled Sage

13.1 Introduction
13.2 Evolution and Compliance
13.3 A Generic Scheduling Problem
13.3.1 A ProcessWeb Example
13.3.2 Programming Around Poor Compliance
13.4 Compliance Through Configuration
13.4.1 Trade-offs in Configuration
13.5 Exploiting an Analytical Model
13.5.1 A First Analytical Model for ProcessWeb
13.6 Discussion 278
13.7 Acknowledgements References 280

14 Comparison of Three Evaluation Methods for Object-Oriented Framework Evolution 281
Michael Mattsson
14.1 Introduction 281
14.2 Object-oriented Frameworks
14.2.1 The Studied Frameworks 285
14.3 Methods and Results
14.3.1 Evolution Identification Using Historical Information 287
14.3.2 Stability Assessment 293
14.3.3 Distribution of the Development Effort 300
14.4 Method Comparison
14.4.1 Change-prone Modules 305
14.4.2 Framework Deployment 306
14.4.3 Change Impact Analysis 306
14.4.4 Benchmarking 306
14.4.5 Requirements Management 307
14.4.6 Some Comments 307
14.5 Related Work 308
14.6 Conclusion References 309 310

15 Formal Perspectives on Software Evolution: From Refinement to Retrenchment 313
Michael Poppleton and Lindsay Groves
15.1 Introduction 313
15.2 Program Refinement 314
15.3 Modifying Refinements by Adapting Derivations 320
15.4 A Compositional Approach to Program Modification 322
15.5 Retrenchment
15.5.1 Refinement – a Relational Perspective 325
15.5.2 The Need to Generalise Refinement 327
15.5.3 Retrenchment: Generalising Refinement 330
15.5.4 Retrenchment for Software Evolution 332
15.6 Conclusions References 335 336

16 Background and Approach to Development of a Theory of Software Evolution 339
Meir M. Lehman and Juan C. Fernández-Ramil
16.1 Software Evolution 339
16.2 Global Views of Evolution
16.2.1 Two Approaches 340
16.2.2 The Verbal Approach 341
16.2.3 **The Nounal Approach**
16.2.4 **Mutual Support of the Two Views**
16.2.5 **Process Improvement**

16.3 **The Case for Theory**
16.4 **Theory Development**
16.5 **A World View**
   - 16.5.1 **Real-World Program Relationship**
   - 16.5.2 **Assumptions**

16.6 **Example**
   - 16.6.1 **Introduction**
   - 16.6.2 **Preliminary Definitions**
   - 16.6.3 **Observations**
   - 16.6.4 **Inferences**
   - 16.6.5 **Guidelines**

16.7 **The Theory**
16.8 **Organisation of Theory Development**
16.9 **Goals**
16.10 **Related Work**
16.11 **Final Remarks**
16.12 **Acknowledgements**

**PART TWO FEEDBACK**

17 **Difficulties with Feedback Control in Software Processes**
   *Meir M. Lehman, Dewayne E. Perry and Wlad Turski*

17.1 **Introduction**
17.2 **Feedback and Control**
17.3 **Technology versus Sociology**
17.4 **Manifesto and Model**
17.5 **Influence versus Control**
   - 17.5.1 **Immaturity**
   - 17.5.2 **Feedback Overload**
   - 17.5.3 **Step Functions versus Regulation**
   - 17.5.4 **Design versus Production**
17.6 **Examples of Feedback Control**
17.7 **Summary**
17.8 **Acknowledgments**

18 **Governing Software Evolution through Policy-oriented Feedback**
   *Nazim H. Madhavji and José Tassé*

18.1 **Introduction**
18.2 **The Policy-Checking Mechanism**
   - 18.2.1 **Controlling System Growth**
   - 18.2.2 **Re-engineering Change-Prone Modules**
   - 18.2.3 **Discussion**
18.3 **The Contextual Framework**
   - 18.3.1 **New-Release Development**
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.3.2</td>
<td>Roles, Communication and Feedback</td>
<td>386</td>
</tr>
<tr>
<td>18.3.3</td>
<td>The Framework Architecture</td>
<td>387</td>
</tr>
<tr>
<td>18.4</td>
<td>Technological Support</td>
<td>389</td>
</tr>
<tr>
<td>18.4.1</td>
<td>Policy-Checking Mechanism</td>
<td>389</td>
</tr>
<tr>
<td>18.4.2</td>
<td>Framework</td>
<td>390</td>
</tr>
<tr>
<td>18.5</td>
<td>Evaluation</td>
<td>391</td>
</tr>
<tr>
<td>18.6</td>
<td>Related Work</td>
<td>393</td>
</tr>
<tr>
<td>18.7</td>
<td>Conclusions</td>
<td>394</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>394</td>
</tr>
<tr>
<td>19</td>
<td>Feedback in Requirements Discovery and Specification:</td>
<td>397</td>
</tr>
<tr>
<td></td>
<td>A Quality Gateway for Testing Requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Suzanne Robertson</td>
<td></td>
</tr>
<tr>
<td>19.1</td>
<td>Contents of the Requirements Specification</td>
<td>397</td>
</tr>
<tr>
<td>19.2</td>
<td>Project Drivers</td>
<td>399</td>
</tr>
<tr>
<td>19.2.1</td>
<td>Producers</td>
<td>400</td>
</tr>
<tr>
<td>19.2.2</td>
<td>Consumers</td>
<td>401</td>
</tr>
<tr>
<td>19.2.3</td>
<td>Sponsors</td>
<td>402</td>
</tr>
<tr>
<td>19.2.4</td>
<td>Subject Matter Consultants</td>
<td>403</td>
</tr>
<tr>
<td>19.2.5</td>
<td>Technical Consultants</td>
<td>404</td>
</tr>
<tr>
<td>19.2.6</td>
<td>Influencers</td>
<td>404</td>
</tr>
<tr>
<td>19.2.7</td>
<td>Project-Sociology Analysis</td>
<td>405</td>
</tr>
<tr>
<td>19.3</td>
<td>Contents of Individual Requirements</td>
<td>405</td>
</tr>
<tr>
<td>19.4</td>
<td>Keeping Track of Connections</td>
<td>407</td>
</tr>
<tr>
<td>19.5</td>
<td>The Quality Gateway</td>
<td>408</td>
</tr>
<tr>
<td>19.6</td>
<td>Lessons Learnt</td>
<td>409</td>
</tr>
<tr>
<td>19.7</td>
<td>Conclusion</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>410</td>
</tr>
<tr>
<td>20</td>
<td>Requirements Risk and Software Reliability</td>
<td>411</td>
</tr>
<tr>
<td></td>
<td>Norman F. Schneidewind</td>
<td></td>
</tr>
<tr>
<td>20.1</td>
<td>Introduction</td>
<td>411</td>
</tr>
<tr>
<td>20.1.1</td>
<td>Requirements Changes and Software Evolution</td>
<td>412</td>
</tr>
<tr>
<td>20.1.2</td>
<td>Objectives</td>
<td>412</td>
</tr>
<tr>
<td>20.1.3</td>
<td>Methods</td>
<td>413</td>
</tr>
<tr>
<td>20.2</td>
<td>Background</td>
<td>413</td>
</tr>
<tr>
<td>20.3</td>
<td>Selected Measurement Research Projects</td>
<td>414</td>
</tr>
<tr>
<td>20.4</td>
<td>Approach to Analyzing Requirements Risk</td>
<td>415</td>
</tr>
<tr>
<td>20.4.1</td>
<td>Categorical Data Analysis</td>
<td>416</td>
</tr>
<tr>
<td>20.5</td>
<td>Risk Factors</td>
<td>418</td>
</tr>
<tr>
<td>20.5.1</td>
<td>Space Shuttle Flight Software Requirements Change Risk Factors</td>
<td>418</td>
</tr>
<tr>
<td>20.6</td>
<td>Solutions to Risk Analysis Example</td>
<td>420</td>
</tr>
<tr>
<td>20.6.1</td>
<td>Categorical Data Analysis</td>
<td>420</td>
</tr>
<tr>
<td>20.6.2</td>
<td>Dependency Check on Risk Factors</td>
<td>422</td>
</tr>
<tr>
<td>20.6.3</td>
<td>Identification of Modules that Caused Failures</td>
<td>422</td>
</tr>
<tr>
<td>20.7</td>
<td>Future Trends</td>
<td>423</td>
</tr>
<tr>
<td>20.8</td>
<td>Conclusions</td>
<td>424</td>
</tr>
<tr>
<td>20.9</td>
<td>Acknowledgments</td>
<td>424</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>424</td>
</tr>
</tbody>
</table>
21 Combining Process Feedback with Discrete Event Simulation Models to Support Software Project Management

David Raffo and Joseph Vandeville

21.1 Introduction

21.2 Providing Up-to-Date Process Feedback

21.2.1 Feedback in Simulation Models

21.2.2 Metrics Repository

21.3 Discrete Event Simulation Models

21.4 Combining Process Feedback with the Discrete Model

21.4.1 Comparing Statistical Process Control with Outcome Based Control Limits

21.5 Illustrative Example

21.5.1 The Scenario Under Consideration

21.5.2 Determining the Performance of the Baseline Process

21.5.3 Use of the Model and Metrics for Quantitative Process Feedback Management

21.5.4 Assessing the Implications of Feedback and Developing an Action Plan (Is the Process in Control?)

21.5.5 Taking Corrective Action and Assessing the Impact of the Changes

21.6 Conclusions

21.7 Acknowledgements

References

22 A Feedforward Capability to Improve Software Reestimation

William W. Agresti

22.1 Introduction

22.1.1 Reestimation: State of the Practice

22.1.2 Objective

22.1.3 Related Research

22.2 A Feedforward Capability

22.2.1 Feedforward Estimation in Other Domains

22.2.2 Feedforward Estimation in Software Development

22.2.3 Operation of a Feedforward Model

22.3 Example Uses of the Feedforward Concept

22.3.1 Feedforward Capability Integrated with a Software Estimation Tool

22.3.2 The Role of a Feedforward Capability in Risk Management

22.4 Conclusion

22.5 Acknowledgements

Appendix

References

23 Modelling the Feedback Part of the Software Process in Software Resource Estimation

Juan C. Fernández-Ramil and Sarah Beecham

23.1 Introduction

23.2 The Evidence of Feedback

23.3 The Need for a Taxonomy

23.4 Feedback as a Cost Factor

23.5 Cost Estimation as a ‘System Identification’ Problem

23.6 Why do Algorithmic Cost Estimation Approaches such as COCOMO ‘Work’?
26.2 Historical Perspective
   26.2.1 Dynamic Versus Static Binding
   26.2.2 Language and Compiler Development
   26.2.3 Performance Trade-Offs
   26.2.4 The Concept of Software Application Evolution
   26.2.5 A Note about Software Ecology

26.3 Self-Adaptive Software
   26.3.1 Concepts
   26.3.2 Technology Requirements and Opportunities

26.4 Applications of Self-Adaptive Software
   26.4.1 Recent Application Work
   26.4.2 Vision Systems
   26.4.3 Face Recognition
   26.4.4 Pervasive Computing

26.5 Conclusion

References

27 Rules and Tools for Software Evolution Planning and Management
Meir M. Lehman and Juan C. Fernández-Ramil

27.1 Introduction

27.2 Laws of Software Evolution

27.3 S- and E-Type Program Classification
   27.3.1 Basic Properties
   27.3.2 Implications of the SPE Program Classification Scheme

27.4 First Law: Continuing Change

27.5 Second Law: Increasing Complexity

27.6 Third Law: Self Regulation

27.7 Fourth Law: Conservation of Organisational Stability

27.8 Fifth Law: Conservation of Familiarity

27.9 Sixth Law: Continuing Growth

27.10 Seventh Law: Declining Quality

27.11 Eighth Law: Feedback System

27.12 The FEAST Hypothesis

27.13 The Principle of Software Uncertainty

27.14 Conclusions

27.15 Acknowledgements

References

Index