Tamed Agility
Tamed Agility

Pragmatic Contracting and Collaboration in Agile Software Projects
Industrial software development is one of the major success stories of the twentieth century. Otherwise, software would not have been able to pervade other areas of life and business, established business models of entire industries would not have been swept away by digitalization, and the global success of Apple, Amazon, Google, Facebook, and eBay would not have been possible.

Software engineering, i.e., the design of larger and larger software systems based on engineering principles, enabled the development of software systems that seemed impossible just a couple of years ago. Therefore, any kind of fundamental denial of this success story is downright absurd (Osterweil et al. 2008). This fact cannot be changed, not even by numerous studies on the alleged state of the software industry, which were in some cases prepared under the flimsiest of conditions, as exposed, e.g., by Eveleens and Verhoef (2010), Glass (2006), or Jørgensen and Moløkken-Østfold (2006).

Yet, time and again, evidence is provided of projects that encounter difficulties—sometimes because the established software development practices have not been followed, sometimes because the individuals involved are too optimistic in their announcements and promises, and in some instances because the numerous individuals involved in a software development project do not have a uniform picture of the actual aim of the project.

It is astonishing that this happens relatively often and is not regarded as a rare exception. Obviously, problems can arise in other projects, not just in software development—airports are finished after serious delays or not at all, public construction projects become more expensive than planned, and trains cannot stop at all platforms. However, genuine project disasters, in the form of a multiplication of the project duration or cost, or in the form of canceled or rolled-back projects, seem to arise more frequently in software development than in other sectors.

Perhaps this is because the immaterial nature of software makes it more difficult to estimate the project state and makes the loss associated with a cancelled project less tangible. Perhaps it is also because software development projects (in which the relevant investment is “only” human resource cost) are often too ambitious and not overly concerned with lean solutions.

Perhaps it is also because the question on the nature of the software process can still not be answered definitively. Is it primarily a production process? Then, it can
be structured from a Taylorist perspective, where detailed specifications are provided, such as in the car production process on an assembly line. Or is it a purely creative process, which is solely driven by the engineer’s design talent? In this case, procedural specifications make little sense, in the same manner that the idea of a precise process to create a painting makes no sense. Software engineering seems to lie between these two poles. There are sections that must be clearly regulated and standardized, such as certain testing activities or configuration management. Others cannot be described using algorithms and cannot be supported by a heuristic process method, such as the approach to identify features to be developed at an early stage.

And then there is the phenomenon of uncertainty. Lehman (1989) provided a convincing argument that software projects are exposed to uncertainties; i.e., that during the course of development, situations could arise that were previously unforeseen (or at least uncertain to occur) and for which appropriate support was unknown. Lehman also noticed that, in most cases, these situations could not be identified in advance. Other authors also made this observation early on:

- “Uncertainty is inherent and inevitable in software development processes and products.”—Ziv et al.’s uncertainty principle in software engineering (1996)
- “For a new software system, the requirements will not be completely known until after you have a working product.”—Humphrey’s requirements uncertainty principle (1995)
- It is impossible to fully specify or test an interactive system.—Wegner’s lemma (1997)

In light of this finding, which is confirmed in practically every software project, terms such as “software factory” (Cusumano 1989) and titles of scientific articles such as “Software Processes are Software too” (Osterweil 1987) seem misleading or at least ambiguous. Software processes (at least for developing socio-technical systems) are insight-driven processes, they are comprised of more creative than algorithmic parts, and it is certainly the case that they are not precisely foreseeable (Gruhn and Urbainczyk 1998).

This in no way denies the existence of types of software that can be fully described. For example, embedded systems without human interfaces can be completely specified and created in line with the production paradigm.

However, this does not apply for socio-technical systems, for the simple reason that these kinds of systems do not end at the screen, but rather extend into the mind of the user. This does not just mean that software must be prepared for unforeseen user behavior. Rather, in socio-technical systems, the software is only a small part of a system comprised of human and mechanical participants that work together to perform complex processes. This interaction, into which software must seamlessly integrate, cannot be fully described and is also subject to constant change. In particular, when dealing with innovation, with the establishment of new business processes and services, and with the implementation of new automations, the design, implementation, and adaptation of software is a creative process, whose purpose requires continuous calibration. The development of these kinds of software
solutions is not a production process, but rather a cognitive process, which is most likely to succeed when all stakeholders keep an eye on the common goal and pay attention to lean solutions.

Even if these solutions are of a technical nature, the goal they must support is anchored in the application domain and not in information technology (IT). Close communication between enterprise IT\(^1\) and operating departments is unavoidable and essential for success in companies that develop software. However, it is often also characterized by different terminology and, especially, by different types of abstraction (and abstraction capacity).

However, the constant realignment of the project idea, the continuous consultation between enterprise IT and the application domain, and the rejection of the idea of a “software factory” (which suggests a completely predictable software production) also result in a few unpleasant conclusions. For example, the fact that the provision of a complete advance specification is not possible (and that the quest for this is doomed to failure), that there will be late requirements (which only arise during development or even after), that budget allocations and cost estimates are provisional, and that at the start of a project, it is impossible to know precisely what can be obtained and at what cost.

But is this really still necessary? Almost 50 years after the term “software engineering” was coined? After almost 50 years in which the “engineering” in “software engineering” defines a claim, namely the claim of reproducibility, reliability, and calculability? It appears to be so, as software development is still risky, projects still encounter difficulties and, when searching for the causes, the same reasons are constantly identified: a lack of understanding of the application domain, incorrect prioritization, and a lack of communication between the stakeholders (Curtis et al. 1988). Software processes are and will always be cognitive processes, but they must satisfy the expectations of production processes.

---

**Structure and Audience of This Book**

This is the challenge that this book deals with—the cognitive nature of software development, the necessity for a unified purpose, the concentration on lean software, the focus on added value, and the omission of the irrelevant. It describes specific instruments and methods enabling all stakeholders to develop a uniform understanding of the software to be created, to determine their genuinely essential requirements, and to deal with changes to this understanding and the requirements.

---

\(^1\)By “enterprise IT,” we refer to a company’s enterprise IT department or to external contractors that perform this function.
The Interaction Room described in Part II brings all stakeholders together for this purpose—not to a table, but in a room where digitalization and mobilization strategies are jointly developed, where technology potentials are evaluated and where software projects are planned and managed. Why does this require a dedicated room? Because stakeholders can then communicate face to face rather than through e-mails. Because the room can be used to outline complex relationships in a comprehensible manner instead of having to laboriously write them up in great detail. Because there is only room for the most important issues. And because insights are not lost in short-term memory or huge documents, but concisely noted and constantly present. In short, because the Interaction Room makes projects visible and tangible.

The adVANTAGE contract model described in Part III ensures that the insight-driven and imprecise process of software development does not just function, but that it is allowed to flourish in a commercial environment, i.e., in a client and contractor relationship. In this model, changes to the project flow are not a reason for stress, but considered normal project events. The contract model ensures that stakeholders focus on generating maximum benefits, creating lean software, and distributing risk fairly despite (or with the aid of) all the changes.

How this can work during the day-to-day running of a project is shown in the practical example of the development of an inventory management system for a private health insurance company in Part IV. This is a complex system with, at first glance, an almost unmanageable number of business requirements, statutory conditions, stakeholders, and processes for general and special cases, embedded in the organically developed IT landscape of an insurance company from North Rhine-Westphalia. The example of the project kickoff and the first sprint shows how employees of the company and the IT contractor developed an overview of the project using the Interaction Room, how the design and development was managed, and how efforts were billed.

Ultimately, the success of every single software project, independently of the application domain and the technology used, depends on the skills of the stakeholders. Only if the stakeholders are prepared to talk to each other, interact with each other, respect different perceptions of value and effort drivers, reach compromises, pursue innovative solutions, and refrain from political maneuvers, can instruments such as the Interaction Room and adVANTAGE fully unfold their potential. Part V therefore finally describes the requirements profile that software engineers as well as domain experts must satisfy today.

Even though contracting and collaboration may be grounded in two different academic disciplines, they are inseparable in practice where all theory boils down to enabling people to work effectively with each other toward a successful product in a sustainable business relationship.

This book is therefore geared toward CIOs, project managers, and software engineers in industrial software development practice who want to learn how to deal effectively with the inevitable uncertainty of complex projects, who want to
achieve higher levels of understanding and cooperation in their relationships with customers and suppliers, and who want to run their software projects at lower risk despite their inherent uncertainty.

Acknowledgments

The authors would like to thank Simon Grapenthin for sharing his extensive hands-on experience in facilitating Interaction Room workshops and training Interaction Room coaches in a wide range of business domains. We would also like to thank Sandra Delvos for countless hours of designing and revising the book’s illustrations, and Alexander Lohberg and Anja Wintermeyer for their background research.

Reykjavik, Iceland
Essen, Germany
Berlin, Germany

References


Contents

Part I  Introduction

1  The Need for Tamed Agility ............................. 3
   1.1  A New School of IT .................................. 3
       1.1.1  Mobility ........................................ 4
       1.1.2  Agility ......................................... 5
       1.1.3  Elasticity ....................................... 5
       1.1.4  Resulting Challenges ........................... 6
   1.2  Agile or Plan-Driven? ................................. 7
   1.3  A Pragmatic Middle Ground ........................... 11
   1.4  Tamed Agility in Practice ............................ 13

References .............................................. 14

Part II  The Interaction Room

2  A Room for Ideas ....................................... 17
   2.1  Key Interaction Room Principles ...................... 18
   2.2  Involve Domain Experts ............................... 20
   2.3  Refine the Scope Continuously ........................ 21
   2.4  Favor Relevance Over Completeness .................. 23
   2.5  Favor Clarity Over Syntactic and Semantic Precision 25
   2.6  Define Value and Effort Drivers ..................... 26
   2.7  Manage Late Requirements ............................ 27
   2.8  Manage Early Requirements ........................... 29
   2.9  Reveal Uncertainties Early ........................... 30
   2.10 Make Cost Changes Transparent ....................... 32
   2.11 Analyze the Risk of Disasters ....................... 33
   2.12 Build Trust Between Stakeholders ................... 34
   2.13 Visualize the Project’s Progress ..................... 35

References .............................................. 36

3  Interaction Room Basics ............................... 39
   3.1  Method Overview .................................... 40
   3.2  Canvases .......................................... 41
   3.3  Annotations ....................................... 43
3.4 Variants ................................................. 48
3.5 Stakeholders ............................................ 51
  3.5.1 Interaction Room Method Coach ................. 52
  3.5.2 Interaction Room Domain Coach ................. 53
  3.5.3 Process Owner .................................... 54
  3.5.4 Additional Roles .................................. 54
3.6 Workshop Preparation ................................. 55
3.7 Results and Follow-up Activities ..................... 56

4 Using an Interaction Room for Digitalization Strategy
   (IR:digital) .............................................. 59
  4.1 Relevant Stakeholders ................................. 64
    4.1.1 Digital Business Expert ....................... 65
    4.1.2 Digital Technology Expert .................... 67
    4.1.3 Interaction Engineer ......................... 68
  4.2 Partner Canvas ....................................... 69
    4.2.1 Methodology and Notation ................... 69
    4.2.2 Annotations and Analysis .................... 72
  4.3 Physical Object Canvas ............................... 73
    4.3.1 Methodology and Notation ................... 74
    4.3.2 Annotations and Analysis .................... 79
  4.4 Touchpoint Canvas ................................... 81
    4.4.1 Methodology and Notation ................... 81
    4.4.2 Annotations and Analysis .................... 83
  4.5 Cross-Canvas Analyses ............................... 84
  4.6 Workshop Structure and Follow-up Activities ....... 86
References ................................................. 89

5 Using an Interaction Room for Software Project Scoping
   (IR:scope) .............................................. 91
  5.1 Relevant Stakeholders ................................. 92
    5.1.1 Application Developer ....................... 92
    5.1.2 Operations Expert ............................ 92
    5.1.3 User ........................................ 93
  5.2 Feature Canvas ....................................... 93
    5.2.1 Methodology and Notation ................... 93
    5.2.2 Annotations and Analysis .................... 94
  5.3 Process Canvas ....................................... 95
    5.3.1 Methodology and Notation ................... 96
    5.3.2 Annotations and Analysis .................... 99
  5.4 Object Canvas ....................................... 103
    5.4.1 Methodology and Notation ................... 103
    5.4.2 Annotations and Analysis .................... 106
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>Integration Canvas</td>
<td>109</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Methodology and Notation</td>
<td>110</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Annotations and Analysis</td>
<td>112</td>
</tr>
<tr>
<td>5.6</td>
<td>Cross-Canvas Analyses</td>
<td>114</td>
</tr>
<tr>
<td>5.7</td>
<td>Workshop Structure and Follow-up Activities</td>
<td>116</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
<td>118</td>
</tr>
<tr>
<td>6</td>
<td>Using an Interaction Room for Mobile Application Development (IR:mobile)</td>
<td>119</td>
</tr>
<tr>
<td>6.1</td>
<td>Relevant Stakeholders</td>
<td>120</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Mobility Expert</td>
<td>121</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Business Developer</td>
<td>121</td>
</tr>
<tr>
<td>6.2</td>
<td>Persona Canvas</td>
<td>121</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Methodology and Visualization</td>
<td>122</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Annotations and Analysis</td>
<td>123</td>
</tr>
<tr>
<td>6.3</td>
<td>Portfolio Canvas</td>
<td>124</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Methodology and Visualization</td>
<td>124</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Annotations and Analysis</td>
<td>125</td>
</tr>
<tr>
<td>6.4</td>
<td>Touchpoint Canvas</td>
<td>127</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Methodology and Notation</td>
<td>127</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Annotations and Analysis</td>
<td>129</td>
</tr>
<tr>
<td>6.5</td>
<td>Interaction Canvas</td>
<td>132</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Methodology and Notation</td>
<td>132</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Annotations and Analysis</td>
<td>135</td>
</tr>
<tr>
<td>6.6</td>
<td>Cross-Canvas Analyses</td>
<td>137</td>
</tr>
<tr>
<td>6.7</td>
<td>Workshop Structure and Follow-up Activities</td>
<td>138</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>140</td>
</tr>
<tr>
<td>7</td>
<td>Using an Interaction Room for Technology Evaluation (IR:tech)</td>
<td>141</td>
</tr>
<tr>
<td>7.1</td>
<td>Relevant Stakeholders</td>
<td>142</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Technology Expert</td>
<td>142</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Enterprise Architect</td>
<td>142</td>
</tr>
<tr>
<td>7.2</td>
<td>Feature Canvas</td>
<td>142</td>
</tr>
<tr>
<td>7.3</td>
<td>Process, Object, and Integration Canvases</td>
<td>145</td>
</tr>
<tr>
<td>7.4</td>
<td>Cross-Canvas Analyses</td>
<td>145</td>
</tr>
<tr>
<td>7.5</td>
<td>Workshop Structure and Follow-up Activities</td>
<td>146</td>
</tr>
<tr>
<td>8</td>
<td>Using an Interaction Room for Agile Project Monitoring (IR:agile)</td>
<td>149</td>
</tr>
<tr>
<td>8.1</td>
<td>From Feature Canvas to Product Backlog</td>
<td>150</td>
</tr>
<tr>
<td>8.2</td>
<td>Sprint Planning Workshops</td>
<td>151</td>
</tr>
<tr>
<td>8.3</td>
<td>Requirements Exchange</td>
<td>152</td>
</tr>
<tr>
<td>8.4</td>
<td>Risk Map</td>
<td>153</td>
</tr>
</tbody>
</table>
8.5  Progress Control ................................................. 158
8.6  Cost Forward Progressing ................................. 159
References .......................................................... 164

9  Using Interaction Rooms Under Difficult Conditions .......... 165
9.1  Temporary Interaction Rooms ............................... 165
9.2  Distributed Interaction Rooms ............................. 167
9.3  Augmented Interaction Rooms ............................... 168
References .......................................................... 169

10  Summary .......................................................... 171
Reference .......................................................... 174

Part III  The adVANTAGE Contract Model

11  Framing Software Projects in Commercial Terms .............. 177
Reference .......................................................... 180

12  Traditional Contract Models in an Agile World .............. 181
12.1  Fixed Price ..................................................... 184
12.2  Time and Materials .......................................... 187
12.3  Pay Per Use .................................................... 188
12.4  Summary ........................................................ 191
References .......................................................... 193

13  Agile Contract Models ......................................... 195
13.1  Fixed Price per Iteration .................................... 195
13.2  Fixed Price per (Whatever) Point .......................... 196
13.3  Money for Nothing, Change for Free ...................... 197
13.4  Shared Pain/Shared Gain .................................... 198
13.5  Multi-stage Contract Models ............................... 200
13.6  Summary ........................................................ 201
References .......................................................... 202

14  Key adVANTAGE Principles ..................................... 205
14.1  Commitment to Agility ....................................... 206
14.2  Mutual Trust .................................................... 207
14.3  Contractor’s Willingness to Assume Risk ................. 209
14.4  Budget Security ................................................. 210
14.5  Shared Pain ...................................................... 210
14.6  Efficiency Incentives ......................................... 211
Reference .......................................................... 212

15  adVANTAGE Procedures ........................................... 213
15.1  Initial Requirements Collection and Budget Estimate ... 213
15.2  Feature Prioritization and Sprint Definition .............. 215
15.3  Sprint Implementation and Controlling .................... 217
15.4 Sprint Inspection and Billing ........................................... 221
   15.4.1 Full Completion of Sprint ........................................ 221
   15.4.2 Partial Completion of Sprint ................................. 224
15.5 Planning the Next Sprint ............................................... 224
15.6 Project Termination ...................................................... 226
15.7 Summary ...................................................................... 227
Reference ........................................................................... 228

16 adVANTAGE in Practice ...................................................... 229
   16.1 Case Study: The BERGFÜRST Crowd Investing Platform .... 229
   16.2 Fine-Tuning adVANTAGE Parameters ............................ 233
References ......................................................................... 234

17 Summary .......................................................................... 235

Part IV A Sample Project

18 Case Study: The Cura Health Insurance Benefit System .......... 241

19 Initial Project Scoping with the IR:scope ................................ 243
   19.1 Project Vision ............................................................ 243
   19.2 Identification of Stakeholders and Objectives ................... 244
   19.3 Feature Canvas .......................................................... 245
      19.3.1 Feature Identification and Canvas Population .......... 245
      19.3.2 Annotation and Analysis ...................................... 245
   19.4 Process Canvas .......................................................... 248
      19.4.1 Identification and Prioritization of Business Processes .. 248
      19.4.2 Canvas Population .............................................. 250
      19.4.3 Annotation and Analysis ...................................... 252
   19.5 Object Canvas ............................................................ 255
      19.5.1 Canvas Population .............................................. 255
      19.5.2 Annotation and Analysis ...................................... 255
   19.6 Integration Canvas ...................................................... 258
      19.6.1 Canvas Population .............................................. 258
      19.6.2 Annotation and Analysis ...................................... 259
   19.7 Cross-Canvas Annotation Analysis ................................... 260
   19.8 Documentation and Follow-up Activities .......................... 261

20 Project Monitoring with the IR:agile ...................................... 263
   20.1 From Feature Canvas to Product Backlog ......................... 263
   20.2 Risk Map .................................................................. 265
   20.3 The First Sprint ......................................................... 267
      20.3.1 Planning the First Sprint ..................................... 267
      20.3.2 Results of the First Sprint ................................... 267
20.4 Settlement Using adVANTAGE ................................. 269
20.5 Cost Forward Progressing ................................. 270
20.6 Using the Requirements Exchange ......................... 270

21 Lessons Learned ............................................... 273

Part V Conclusion

22 The Big Picture ................................................ 281
References ......................................................... 282

23 A New Skill Set ................................................ 283
23.1 General Software Technology and Methodology Skills .... 283
23.2 New School of IT Skills: Mobility ......................... 284
23.3 New School of IT Skills: Agility ......................... 287
23.4 New School of IT Skills: Flexibility ....................... 288
23.5 Business Development and Domain Knowledge ........ 289
23.6 Knowledge of Business Processes, Business Models, 
and Partnerships ................................................. 290
23.7 Insights and Experiences .................................. 291
References ......................................................... 292

24 Outlook: Twelve Hypotheses ................................. 293

Appendix A: Interaction Room Workshop Agendas .......... 295

Appendix B: Interaction Room Annotations .................. 299

Appendix C: adVANTAGE Contract Template ................. 313

Index ................................................................. 329