Agility Across Time and Space
Implementing Agile Methods in Global Software Projects
Agility Across Time and Space
Agility Across Time and Space

Implementing Agile Methods in Global Software Projects
Drawing from earlier definitions from Jim Highsmith or Steve Adolph and the OODA loop, I like to define agility as “the ability of an organization to react to change in its environment faster than the rate of these changes.” This definition uses the ultimate purpose or function of being agile for a business, rather than defining agility by a labeled set of practices (e.g., you’re agile when you do XP, Lean, or Scrum) or by a set of properties defined in opposition to another set (the agile manifesto approach).

An analogy could be the definition of a road. Would you define a road as something made of crushed rocks and tar, or define it as a surface that is black rather than white, flat rather than undulated, and with painted lines rather than monochrome? Or as a component of a transportation system, allowing people and goods to be moved on the ground surface from point A to point B? And let the properties or components be derived from this, allowing some novel approach in road design.

It is quite possible to adopt a labeled set of agile practices, or a set of practices that perfectly conform to the agile manifesto and not become agile. You then “do Agile” but are not agile.

Agile software development methods do succeed in contexts which are identical or very similar to the contexts in which they have been created. As these contexts—the “agile sweet spot”—are very frequent in software development, representing more than 50% of all software being developed, this may have led sometimes their proponents to a certain complacency: thinking that the method has universal value, that its represents some ultimate recipe, the holy grail.

Agile methods may fail in various ways when they are applied “out of the box”, i.e., with no or little adaptation, in contexts that are very far, at least on one dimension, from the context in which they have been originally created. Rather than an analysis of the root cause, this usually triggers screams of “you must have not done it right” by its proponents. And this again leads to discussion of “purity”, “scrumbutts”, etc.

Agile methods can be stretched with variable success outside of the context in which they have been created; for example, scaling them up to larger projects, or across distributed teams. In my experience, the contextual factors that have the greatest risks of derailing agile projects are:
• size
• large systems with a lack of architectural focus
• software development not driven by customer demand
• lack of support from surrounding stakeholders, traditional governance
• novice team
• very high constraint on some quality attribute (safety-critical system, real-time constraints)

As noted by many authors in the last few years, we cannot just rely on acts of faith by eloquent process gurus to help us define the adequate process, or set of practices outside of the agile sweet spot. Cold-headed, impartial investigation is required. Such research is generally not very easy to conduct; it is often qualitative, rather than quantitative, it draws more from social sciences than computer science, not easy to publish, not easy to carve down to masters’ thesis bite size.

This is the reason why I welcome this volume on *agility across time and space*. Looking at how agile practices performed once stretched outside of the agile sweet spot, for large projects and distributed projects, the non-trivial ones. The researchers and practitioners who collectively wrote this volume have been examining without prejudice what works and what does not, and trying to get at the root cause, giving us another and better perspective on this fascinating wave: the agile software development movement. They confront some of the factors I mentioned earlier: size, distribution, role of architecture, culture.

Because all things considered, our stakeholders do not care whether you did or not your daily stand-up meetings, whether pairing was followed religiously, how many columns in your kanban, or if you played poker for estimations. They only care about quality software hitting the market as fast as we possibly can. The software developer should only be concerned by what will allow her to achieve this in her specific context. And in a turbulent environment, can the organization react to change in its environment faster than the rate of these changes?

Vancouver, BC, Canada

*Philippe Kruchten*
Preface

Despite the progress in the field of software engineering, software projects are still being late, are over budget, and do not deliver the expected quality. Two major trends have emerged in response to these: global sourcing and the application of agile methods. The new paradigms soon became anecdotally popular for their benefits of cheaper and faster development of high quality software. Many companies recently started to look into merging these two promising approaches into one strategy.

Globally Distributed Development

Global sourcing promises organizations the benefits of reaching mobility in resources, obtaining extra knowledge through deploying the most talented people around the world, accelerating time-to-market, increasing operational efficiency, improving quality, expanding through acquisitions, reaching proximity to market and many more. However, these benefits are neither clear-cut nor can their realization be taken for granted, as the literature may lead one to believe [1]. In fact, there are many challenges related to communication, coordination and control when developing software with global software teams [2].

Agile Development

Agile development has recently attracted huge interest from software industry [3]. It is being recognized for its potential to improve communication and, as a result, reduce coordination and control overhead in software projects. Methods for agile software development constitute a set of practices for software development that have been created by experienced practitioners [4]. The “agile manifesto” was published in 2001 by the key people behind the early agile development methods. The manifesto states that agile development should focus on four core values [5]:

- Individuals and interactions over processes and tools,
• Working software over comprehensive documentation,
• Customer collaboration over contract negotiation,
• Responding to change over following a plan.

Agile methods can be seen as a reaction to plan-based or traditional methods, which emphasize “a rationalized, engineering-based approach” [6] in which it is claimed that problems are fully specifiable and that optimal and predictable solutions exist for every problem. The “traditionalists” are said to advocate extensive upfront planning, codified processes, and rigorous reuse to make development an efficient and predictable activity [7]. By contrast, agile processes address the challenges of the increasingly complex nature of software development by relying on people and their creativity rather than on formalized processes [6]. The goal of optimization is being replaced by those of flexibility and responsiveness [8]. Ericksson et al. [9] define agility as follows: *agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like.* (p. 89)

**The Role of Agility in Distributed Development**

Global software development has matured considerably since its inception and has become an integral part of the information technology landscape. Now, rather than deciding whether or not to get involved in global sourcing, many companies are facing decisions about whether or not to apply agile methods in their distributed projects. These companies are often motivated by the opportunities of solving the coordination and communication difficulties [4] associated with global software development.

Empirical evidence from case studies conducted by Paasivaara and Lassenius [10], and Holmström, Fitzgerald et al. [11] show successful implementation of agile values and principles in different globally distributed projects. This motivates assessing the viability of agile practices for distributed software development teams. The interest in becoming agile and distributed is also illustrated by the increasing number of research publications and seminars devoted to the topic.

**Implementing Agility Across Time and Space**

Despite the increased attention, merging the two strategies is no easy task due to significant differences in fundamental principles of agile and distributed development approaches. In particular, while agile principles prescribe close interaction and co-location, the very nature of distributed software development does not support these prerequisites. Taylor, Greer et al. [12] claim that distributed agile software development suffers substantial difficulties because of its complex development environment and there is little empirical evidence describing actual development experiences. The lack of clear understanding of who, what, when, why and how in agile
distributed development motivated us to collect experiences from various companies that had started, and also benefitted from, becoming agile and distributed.

**Aims of the Book**

The idea to write a book on agile and distributed software development gradually evolved as the critical mass of questions related to merging seemingly incompatible approaches emerged. The questions that the authors aimed to answer with this book include:

- What shall companies expect from merging agile and distributed strategies?
- What are the stumbling blocks that prevent companies from reaching the agile benefits in distributed environment, and how to recognize unfeasible strategies and unfavorable circumstances?
- What helps managers cope with the challenges of implementing agile approaches in distributed software development projects?
- How can distributed teams survive the decisions taken by the management and become efficient through the application of agile approaches?

**Book Overview**

This book consists of five parts.

1. In the **Motivation** part the editors introduce the fundamentals of agile distributed software development and explain the rationale behind the application of agile practices in globally distributed software projects.

![Fig. 1 Book layout](image)

2. The second part of the book is called **Transition**. Here we have gathered seven chapters that discuss the transition to being agile and distributed. The chapters describe implementation strategies, adoption of particular agile practices for distributed projects, and general concepts of agility.
3. The third part of the book, **Management**, focuses on managerial aspects and decisions in agile distributed software projects. Practical implications for project planning, time management, customer and sub-contractor interaction, tool support and architecture-centric development are presented in eight chapters.

4. The fourth part is devoted to agile and distributed **Teams**. Here we have collected six chapters that provide in-depth hands-on advice for the team members and their managers. Topics discussed include agile distributed team configuration, effective communication and knowledge transfer, the role of architecture in task division, and allocation of roles and responsibilities.

5. Finally, in the **Epilogue** we summarize the contributions of the different chapters and present results from a Delphi-inspired study that highlights the major areas of concern and future trends for research and practice in agile distributed development.

Most of the chapters in this book offer practical advice based on experiences obtained in and from the industry. These experiences are collected through personal observations of practitioners, empirical research in particular studied contexts or extensive continuous observations gained from various sources.

**Target Audience**

This book is primarily targeted at practitioners (managers and team members) involved in globally distributed software projects - those who are practicing agile methods and those who are not. We believe that it will serve as a useful source of practical advice, which are based on the real life examples of application of agile practices in distributed development, and will hopefully motivate companies to try improving their sourcing strategies by adopting best practices and benefits that agile promises.

Many book chapters are based on the sound empirical research and identify gaps and commonalities in the existing state-of-the-art and state-of-the-practice. We thus believe that our book can be also of relevance and interest for the academic audience, in particular, researchers working in the field, as well as lecturers and students of global agile software development.

**References**


*Darja Šmite*
*Nils Brede Moe*
*Pär J. Ägerfalk*
Acknowledgements

We are thankful to all the authors for their valuable contributions and effort in creation of this book. We are also thankful to the external reviewers for their timely responses and valuable feedback. Springer Computer Science Editorial and especially Ralf Gerstner deserves a special gratitude for initiating the creation of this book and supporting our ideas on its way. A special thanks goes to Likoebe M. Maruping for the book title idea and to Claes Wohlin who has brought us together. Last but not the least we are thankful to our families and friends for their immeasurable support.

This book has been supported by

- the Software Engineering Research Lab in the School of Computing, Blekinge Institute of Technology,
- the Research Project “Agile”, funded by the Research Council of Norway under grant 179851/I40, and
- the Department of Informatics and Media, Uppsala University
Contents

Part I  Motivation

1  Fundamentals of Agile Distributed Software Development ................. 3
   Darja Šmite, Nils Brede Moe, and Pär J. Ågerfalk
   1.1  Introduction .................................................................... 3
   1.1.1  Distributed Software Development ............................... 3
   1.1.2  Agile Software Development ........................................ 4
   1.2  Merging Agility with Distribution ..................................... 4
   1.2.1  Potential Issues .......................................................... 5
   1.2.2  All or Nothing versus À la carte .................................... 6
   1.3  Current Practice ............................................................. 6
   1.4  Conclusions ................................................................. 7
   References ............................................................................ 7

Part II  Transition

2  Implementing Extreme Programming in Distributed Software
   Project Teams: Strategies and Challenges ................................. 11
   Likoebe M. Maruping
   2.1  Introduction .................................................................. 11
   2.2  Implementing XP Practices: Where Is an Organization to Start? 12
       2.2.1  The Promise of XP .................................................... 12
       2.2.2  Understanding How Your Software Project Team Is
              Structured and Why It Matters ................................. 13
   2.3  Case Overview ............................................................. 14
   2.4  XP in Distributed Software Project Teams: Implementation
       Strategies and Pitfalls to Avoid .......................................... 16
       2.4.1  The Planning Game .................................................... 16
       2.4.2  Collective Ownership ................................................. 18
       2.4.3  Coding Standards ...................................................... 19
       2.4.4  Use of a Metaphor ..................................................... 20
2.4.5 Simplicity of Design ........................................ 21
2.4.6 Sustainable Pacing ........................................ 22
2.4.7 Pair Programming ......................................... 23
2.4.8 Continuous Integration and Unit Testing ............... 24
2.4.9 Refactoring ............................................... 25
2.4.10 Customer Involvement .................................. 26
2.4.11 Small Functional Releases ............................. 27
2.5 Conclusions ................................................. 28
References ....................................................... 29
Further Reading ............................................... 30

3 Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report ........................................ 31
Daniel Wildt and Rafael Prikladnicki

3.1 Introduction .................................................. 31
3.2 Case Overview .............................................. 32
3.3 Transitioning to Agile in a Distributed Environment .......... 34
3.3.1 Don’t Tell What Agile Is and Be Successful .......... 35
3.3.2 A Fully Cultural Transition from Traditional to Agile Development ........................................... 37
3.3.3 Benefits of Using Agile Methods in Distributed Environment ........................................... 40
3.4 Practical Recommendations ................................ 41
3.5 Conclusions .................................................. 45
References ....................................................... 45

4 Tailoring Agility: Promiscuous Pair Story Authoring and Value Calculation ........................................... 47
Steve Tendon

4.1 Introduction .................................................. 47
4.2 The Case ..................................................... 48
4.2.1 Background .............................................. 48
4.2.2 Management Support and Sponsorship ............... 50
4.2.3 The Pilot Project ........................................ 51
4.2.4 The Journey of Implementing Agility ................ 53
4.2.5 The Final: Project Approval ................................ 61
4.3 Benefits from Implementing Agility over Traditional Approaches ........................................... 62
4.3.1 More Commonality ........................................ 62
4.3.2 Smaller Scope ............................................ 63
4.3.3 ROI Anticipation ........................................ 63
4.3.4 Smaller Country-Specific Dependencies ............... 63
4.3.5 Avoiding Waste Upfront ................................ 63
4.4 Why Agile Succeeded? ....................................... 64
4.4.1 Induction .................................................. 64
4.4.2 Co-location and Alternating On- and Off-Site Activities ........................................... 65
4.4.3 Promiscuous Pair Story Authoring .................... 66
5 Scrum and Global Delivery: Pitfalls and Lessons Learned ...... 71
Cristiano Sadun
5.1 Introduction ........................................ 71
5.2 Cases Overview .................................... 72
  5.2.1 Background .................................... 72
  5.2.2 Project NOR1 .................................. 74
  5.2.3 Project NOR2 .................................. 76
5.3 The Experiences .................................... 77
  5.3.1 Signing Agreements ............................ 77
  5.3.2 Establishing Remote Access .................... 79
  5.3.3 Overcoming Communication Barriers ........... 80
  5.3.4 Actively Managing Distributed Agile Projects ... 82
  5.3.5 Dealing with Idle Time ....................... 84
  5.3.6 Achieving Motivation and Peer Feeling .......... 86
  5.3.7 Adapting Governance and Steering ............. 87
5.4 Conclusions ........................................ 88
References ............................................ 89

6 Onshore and Offshore Outsourcing with Agility: Lessons Learned . 91
Clifton Kussmaul
6.1 Introduction ........................................ 91
6.2 Case Overview ..................................... 92
  6.2.1 Background .................................... 92
  6.2.2 Project Organization .......................... 93
  6.2.3 Introduction of Agility ....................... 95
  6.2.4 Overview of Project Activities ................ 95
  6.2.5 Cross-border Relationship Dynamics .......... 97
6.3 Lessons Learned .................................. 98
  6.3.1 People ........................................ 99
  6.3.2 Processes .................................... 100
  6.3.3 Coordination ................................ 102
6.4 Conclusions ....................................... 104
References ............................................ 105
Further Reading ..................................... 105

7 Contribution of Agility to Successful Distributed Software
  Development .......................................... 107
Saonee Sarker, Charles L. Munson, Suprateek Sarker, and Suranjan
  Chakraborty
7.1 Introduction ....................................... 107
7.2 Distributed Project Success .......................... 108
7.3 Types of Agility ................................... 109