



Contents

<i>Foreword</i>	xiii
<i>Preface</i>	xv
Section 1 The Problem	1
Chapter 1 <i>Risk: The Basic Problem</i>	3
<i>Software development fails to deliver, and fails to deliver value. This failure has huge economic and human impact. We need to find a new way to develop software.</i>	
Chapter 2 <i>A Development Episode</i>	7
<i>Day-to-day programming proceeds from a task clearly connected to a feature the customer wants, to tests, to implementation, to design, and through to integration. A little of each of the activities of software development are packed into each episode.</i>	
Chapter 3 <i>Economics of Software Development</i>	11
<i>We need to make our software development economically more valuable by spending money more slowly, earning revenue more quickly, and increasing the probable productive lifespan of our project. But most of all we need to increase the options for business decisions.</i>	

Chapter 4	<i>Four Variables</i>	15
	<i>We will control four variables in our projects—cost, time, quality, and scope. Of these, scope provides us the most valuable form of control.</i>	
Chapter 5	<i>Cost of Change</i>	21
	<i>Under certain circumstances, the exponential rise in the cost of changing software over time can be flattened. If we can flatten the curve, old assumptions about the best way to develop software no longer hold.</i>	
Chapter 6	<i>Learning to Drive</i>	27
	<i>We need to control the development of software by making many small adjustments, not by making a few large adjustments, kind of like driving a car. This means that we will need the feedback to know when we are a little off, we will need many opportunities to make corrections, and we will have to be able to make those corrections at a reasonable cost.</i>	
Chapter 7	<i>Four Values</i>	29
	<i>We will be successful when we have a style that celebrates a consistent set of values that serve both human and commercial needs: communication, simplicity, feedback, and courage.</i>	
Chapter 8	<i>Basic Principles</i>	37
	<i>From the four values we derive a dozen or so basic principles to guide our new style. We will check proposed development practices to see how they measure up to these principles.</i>	
Chapter 9	<i>Back to Basics</i>	43
	<i>We want to do everything we must do to have stable, predictable software development. But we don't have time for anything extra. The four basic activities of development are coding, testing, listening, and designing.</i>	
Section 2	The Solution	51
Chapter 10	<i>Quick Overview</i>	53
	<i>We will rely on the synergies between simple practices, practices that often were abandoned decades ago as impractical or naïve.</i>	

Chapter 11	<i>How Could This Work?</i>	63
	<i>The practices support each other. The weakness of one is covered by the strengths of others.</i>	
Chapter 12	<i>Management Strategy</i>	71
	<i>We will manage the overall project using business basics—phased delivery, quick and concrete feedback, clear articulation of the business needs of the system, and specialists for special tasks.</i>	
Chapter 13	<i>Facilities Strategy</i>	77
	<i>We will create an open workspace for our team, with small private spaces around the periphery and a common programming area in the middle.</i>	
Chapter 14	<i>Splitting Business and Technical Responsibility</i>	81
	<i>One key to our strategy is to keep technical people focused on technical problems and business people focused on business problems. The project must be driven by business decisions, but the business decisions must be informed by technical decisions about cost and risk.</i>	
Chapter 15	<i>Planning Strategy</i>	85
	<i>We will plan by quickly making an overall plan, then refining it further and further on shorter and shorter time horizons—years, months, weeks, days. We will make the plan quickly and cheaply, so there will be little inertia when we must change it.</i>	
Chapter 16	<i>Development Strategy</i>	97
	<i>Unlike the management strategy, the development strategy is a radical departure from conventional wisdom—we will carefully craft a solution for today’s problem today, and trust that we will be able to solve tomorrow’s problem tomorrow.</i>	
Chapter 17	<i>Design Strategy</i>	103
	<i>We will continually refine the design of the system, starting from a very simple beginning. We will remove any flexibility that doesn’t prove useful.</i>	
Chapter 18	<i>Testing Strategy</i>	115
	<i>We will write tests before we code, minute by minute. We will preserve these tests forever, and run them all together frequently. We will also derive tests from the customer’s perspective.</i>	

Section 3 Implementing XP	121
Chapter 19 Adopting XP	123
<i>Adopt XP one practice at a time, always addressing the most pressing problem for your team. Once that's no longer your most pressing problem, go on to the next problem.</i>	
Chapter 20 Retrofitting XP	125
<i>Projects that want to change their existing culture are far more common than projects that can create a new culture from scratch. Adopt XP on running projects a little at a time, starting with testing or planning.</i>	
Chapter 21 Lifecycle of an Ideal XP Project	131
<i>The ideal XP project goes through a short initial development phase, followed by years of simultaneous production support and refinement, and finally graceful retirement when the project no longer makes sense.</i>	
Chapter 22 Roles for People	139
<i>Certain roles have to be filled for an extreme team to work—programmer, customer, coach, tracker.</i>	
Chapter 23 20–80 Rule	149
<i>The full value of XP will not come until all the practices are in place. Many of the practices can be adopted piecemeal, but their effects will be multiplied when they are in place together.</i>	
Chapter 24 What Makes XP Hard	151
<i>Even though the individual practices can be executed by blue-collar programmers, putting all the pieces together and keeping them together is hard. It is primarily emotions—especially fear—that make XP hard.</i>	
Chapter 25 When You Shouldn't Try XP	155
<i>The exact limits of XP aren't clear yet. But there are some absolute show-stoppers that prevent XP from working—big teams, distrustful customers, technology that doesn't support graceful change.</i>	

Chapter 26 <i>XP at Work</i>	159
<i>XP can accommodate the common forms of contract, albeit with slight modifications. Fixed price/fixed scope contracts, in particular, become fixed price/fixed date/roughly fixed scope contracts when run with the Planning Game.</i>	
Chapter 27 <i>Conclusion</i>	165
<i>Annotated Bibliography</i>	167
<i>Glossary</i>	177
<i>Index</i>	181