Foreword

Sustainable Software Development encompasses what I consider the two critical concepts in agile development—building adaptable teams and building adaptable products. Using these concepts enables teams to deliver value to their customers in the short term and sustain the capability to deliver additional value over time. You can have an incredibly talented agile team—put them to work on a clunky, poorly designed, ineffectively tested, spaghetti coded, legacy system—and it will not be able to deliver sustainable value to customers. Conversely, you can have a superb product in all those dimensions—and the wrong team can turn it into a poor, unsustainable product, very quickly. *Sustainable Development* helps us think clearly about the continuing delivery of customer value, from both a team and a product perspective—iteration by iteration, product release by product release.

Three themes dominate Kevin's book—sustainability, balance, and context. Sustainability means developing the capability to deliver customer value today and tomorrow. Agility is the art of balancing—short term versus long term, anticipation versus adaptation, ceremony versus informality. And finally, because every project team and every project is different, one must understand the specific context of each project in order to adapt practices to that context.

Sustainable development delivers value to the customer today *and* keeps the cost of change low in order to deliver value in the future. In agile circles sustainability has been applied to people, as in maintaining a sustainable pace of work. But products are also sustainable—or not. I remember a company whose product had an 18-month QA cycle—they reached that "unsustainable" position a little at a time, but the cumulative effect of neglect was non-responsiveness to their market.

Kevin identifies four principles of sustainability—working product, design emphasis, continual refinement, and defect prevention. In each of these areas he then aggregates a number of practices into a coherent whole. For example, the chapter on design provides a vision of design and then shows how a number of practices (not usually found aggregated together) contribute to that design vision. Practices in this chapter include guiding

principles, design patterns, refactoring, simple design, reuse, re-architecture, and rapid design meetings.

Sustainability depends on the art of balancing. Agility can be defined as the art of balancing flexibility and structure, anticipation (planning) and adaptation. I use the term art because there is no silver bullet—no one answer. To illustrate this idea of balancing, Kevin uses a juggling analogy of keeping four balls in the air—the elements of continual refinement, working product, design emphasis, and defect prevention.

The journey from agile novice to agile artist requires first learning agile practices, and then learning how to adapt them and balance one with another. In Chapter 6, "Emphasis on Design," Kevin shows us how to balance the anticipation practices of guiding principles and design patterns with the adaptation practices of simple design and refactoring. If traditional methodologies can be accused of overemphasizing anticipation at the expense of adaptation, then agile methodologies can be accused of overemphasizing the reverse. Kevin's perspective helps practitioners find a middle ground—just enough, but not too much anticipation, combined with effective adaptation. Wouldn't it be wonderful if software development was simply one way or the other? But it's not. It's not simple. It requires artistic balance and Kevin gives us an insight into his own balancing artistry.

Finally, we need to understand the factors that help us make balancing decisions. These factors provide the context for making project decisions and adapting agile practices. In their book, *Shared Voyage: Learning and Unlearning from Remarkable Projects*, Alexander Laufer, Todd Post, and Edward Hoffman relate stories from several remarkable projects including the Advance Composition Explorer project at NASA and the Joint Air-to-Surface Standoff Missile project for the U.S. Air Force. This remarkably agile-like book shows how agile principles can be applied to aerospace projects. One of the three prime objectives of the study that resulted in this book was, "to enhance awareness to the different contexts surrounding different projects." How the NASA and Air Force teams applied practices and principles was context specific. Kevin provides us with key insights into adapting agile software development practices to specific situations, to specific contexts.

Kevin was practicing agile techniques before they acquired the *agile* moniker, and his lengthy experience in software product development shines through in this book. Aggregating practices into his four categories

provides great benefit and the material is immeasurably strengthened by Kevin's real life, practical experiences that manifest themselves through his fascinating stories, tips, and examples.

I met Kevin about five years ago when he was instrumental in bringing agile practices to his company. It turned out to be one of my favorite agile enablement experiences, due in great measure to Kevin's leadership of the change process. Chapter 7, "Continual Refinement," delves into change practices from the perspective of a practitioner who has led the way.

There are two categories of authors in the agile movement. First, there are those like Kent Beck whose groundbreaking *Extreme Programming Explained* helped launch the movement into our everyday consciousness. Second, there are those like Mike Cohn whose wonderful book *User Stories Applied* extends our understanding of key principles and practices. Kevin's book lies in this second category—taking things we thought we knew and expanding them in new ways that make us think, "Wow, I wish I'd thought of that." To paraphrase Luke Hohmann (*Journey of the Software Professional*), enjoy your journey with a unique software professional.

Jim Highsmith
Flagstaff, Arizona
May 2005