Agile Software Development Deserves An Open-Minded Approach

This month's CROSSTALK focuses on one of the potentially hot topics of debate in the software industry today – agile software development vs. disciplined, process-focused software development. There seems to be much confusion about what agile software development is and is not, and what agile implies. We hope that our selection of articles will provide some insight into agile software development methodologies and how they compare in application and use to the process improvement strategies embodied in capability maturity models that we have championed for many years. Perhaps these articles may even widen your understanding and perceptions of where agile development fits within today's software development challenges.

Our lead article by Jim Highsmith, *What Is Agile Software Development?* sets the stage for understanding agile software development. He discusses what he terms the *sweet spot problem* domain for agile approaches and then gives greater detail on the three dimensions he refers to as agile ecosystems: chaordic perspective, collaborative values, and barely sufficient methodology.

We next provide part one of a two-part article by Alistair Cockburn, *Learning From Agile Software Development – Part One*. Part one describes how agile and plan-driven teams make different trade-offs of money for information or for flexibility, and presents the first seven of 10 principles for tuning a project to meet various priorities, including cost, correctness, predictability, speed, and agility. Part two will be featured in the November issue of CROSSTALK.

In *Agile Methodologies and Process Discipline*, Mark C. Paulk addresses issues surrounding agile development as compared to rigorous software process improvement models such as the Capability Maturity Model® for Software (SW-CMM®). He summarizes and critiques the compatibility of agile methodologies with plan-driven methodologies as described by the SW-CMM, and concludes by making the point, "Perhaps the biggest challenge in dealing effectively with both agile and plan-driven methodologies is dealing with extremists in both camps who refuse to keep an open mind."

In his article, *Odyssey and Other Code Science Success Stories*, John Manzo tells of the successful delivery of an actual complex industrial automation application using an agile development methodology based on eXtreme Programming (XP). He includes some real-world insights from a developer's experience applying the development method, including a quantitative measure of XP's effectiveness since its inception.

Our supporting articles this month begin with *Integrating Systems and Software Engineering: What Can Large Organizations Learn From Small Start-Ups?* Author Paul E. McMahon explores variations in large and small engineering organizations and presents an alternative view of large projects that he claims may aid companies in their quest for more effective systems and software integration. He believes that using adaptive techniques and small informal teams inside large organizations can complement a formal organizational process focus that may already exist within a company.

Next, in *Highpoints From the Agile Software Development Forum*, Pamela Bowers summarizes the keynote talks from "Creating Competitive Advantage Through Agile Development Practices," a technology forum held at Westminster College in Salt Lake City in March. In *Agile Before Agile Was Cool,* Gordon Sleve describes a specific example of choice of an agile methodology for the development of a needed application, with quick turnaround to meet a customer's need. He notes that this was done long before agile methods were well known or popular and concludes by saying, "I see both methodologies coexisting and filling an important purpose: *to make our customers successful*."

We conclude this issue with two online articles: *Should You Be More Agile?* by Rich McCabe and Michael Polen, and *Agile Development: Weed or Wildflower?* by David Kane and Steve Ornburn. These authors believe that software projects, even in the defense community, can benefit from the techniques of agile development.

As noted in the lengthy references at the end of many of these articles, much is being written about agile software development at this time. We hope that this selection of thought-provoking articles will enable you to become more enlightened about the many perspectives of these new software development alternatives and will have a positive influence on opening your minds to new possibilities.

H. Bruce Allgood
Deputy Director, Computer Resources Support Improvement Program