By Sanjiv Augustine,
Bob Payne, Fred Sencindiver,
and Susan Woodcock

Agile Project Management: STEERING FROM THE EDGES

Agile project management lets software project managers and employees alike adapt to changing circumstances, rather than try to impose rigid formal controls, as in traditional linear development methods.

Dealing with an increasingly volatile organizational environment is a serious challenge for managers of any software development project [11]. Traditional formal software development methodologies can be characterized as reflecting linear, sequential processes, and the related management approaches can be effective in developing software with stable, known, consistent requirements. Yet most real-world development efforts are much more likely to be conducted in more volatile environments, as organizations adapt to changing technology, markets, and social conditions. Requirements for systems must be able to change right along with them, often at "Internet speed" [3]. Even seemingly minor changes can produce unanticipated effects, as systems become more complex and their components more interdependent. Project management approaches based on the traditional linear development methodologies are mismatched with such dynamic systems.

Observing this tendency for software requirements to change, Meir Lehman, writing in [9], suggested that their underlying processes can be characterized as "multi-level, multi-loop, multi-agent feedback systems." Software developers have long responded to this complexity with iterative, often ad-hoc approaches. More recently, a host of "agile" development methodologies, including eXtreme Programming (XP) [4], Crystal, Scrum, Adaptive Software Development, Dynamic Systems Development Method, and Feature-Driven Development, have sought to focus on rapid iterative delivery, flexibility, and working code [1].

In our experience, project managers invariably fall back on the traditional linear approaches, seeking to reign in the increasing volatility of their projects. This can be true even when they use agile methodologies.

THE AGILE MANAGER understands the effects of the mutual interactions among a project's various parts and steers them in the direction of continuous learning and adaptation.

These efforts can lead to "stable systems drag" [11] in which organizations try to respond simultaneously to both changing environmental conditions and to their own increasingly obsolete legacy systems.

Projects that employ agile methodologies are complex adaptive systems (CAS) [8], as discussed in the sidebar. We have evolved a CAS-based Agile Project Management (APM) framework, aiming to leverage XP to steer projects to success in terms of being on schedule and within budget while satisfying their customers. The APM framework prescribes the six practices for managing agile development projects discussed here:

Organic teams of from seven to nine members. Self-organization and emergent order are due in part to complex interactions or flows among agents. Organizing a project into organic teams implies a minor interaction penalty in terms of communication and coordination overhead [6]. Allowing members to join and leave the team allows dynamic team composition and supports adaptability to changing external conditions. The team [10] maintains optimal internal channels of communication while minimizing the effect of an interaction penalty. Even when a project requires a larger team of, say, more than 15 members, it can still be organized into several smaller, organic subteams working in parallel. The agile manager is responsible for establishing clear roles and responsibilities to ensure proper team alignment and accountability.

Guiding vision. CAS agents help anticipate and adapt to changing conditions. A project vision translated into a simple statement of project purpose and communicated to all team members has a powerful effect on individual member behavior. In the U.S. Army, an example of this principle is "commander's intent." The Army knows its leaders are not omnipresent; the commander's intent is employed as a guide for soldiers' individual initiatives, actions, and decisions. Even if a mission falls on the shoulders of the lowest-ranking person, that person is still able to carry out the mission. Likewise, agile managers guide their teams by defining, disseminating, and sustaining a vision that influences the internal models of individual agents. The Agile Manifesto (www.agilemanifesto.org) created in 2001 by the proponents of these methodologies articulated a core set of values useful in steering this vision.

Simple rules. In CAS, agents follow simple rules, but their interactions result in complex behavior that emerges over time. The standard XP practices represent a set of simple rules for agile development projects. They're accepted by all members of the team at the outset, though the team can adjust or add new practices as needed. Throughout a project, the manager identifies practices that aren't being followed, seeks to understand why they're not, and removes obstacles to their implementation. XP practices provide simple generative rules without restricting the autonomy and creativity of team members.

Free and open access to information. In CAS, information about plans, progress, objectives, and organization is the catalyst for adaptation by each member of a project team. The richness of the interaction among team members depends largely on their openness to the exchange of information. For an agile team

COMPLEX ADAPTIVE SYSTEMS

The CAS concept is derived from the mathematical science of complexity. Complex systems are nonlinear, open, dynamic. In nonlinear systems the value of the whole cannot be determined by the sum of the parts. An open system interacts with its environment, receiving inputs and providing outputs, but does not control it. A dynamic system changes and evolves its behavior in response to its inputs. Order emerges through the interaction among the system's parts as they evolve (within the larger system) in response to the changing environment.

CASs are therefore composed of semiautonomous agents that seek to maximize some measure of fitness by evolving or adapting to changes as they occur. Local, often simple, rules guide the interaction among the agents and result in the system's global behavior [7].

An ant colony is an example of a CAS. Individually, ants have primitive brains yet collectively run surprisingly sophisticated and efficient operations. Using a few simple rules of logic without central direction, they find food, build and maintain their nests, tend to their young, and respond to attacks [2]. C

to adapt, information must be open and free-flowing. In the APM world, information flows freely and team members benefit from the power of knowledge no matter what its source.

Light touch management style. With traditional approaches, everything is viewed through the prism of control—of change, risk, and, most important, people. Elaborate methodologies, tools, and practices have evolved to manage an out-of-control world. But tools fail when neat linear tasks don't easily accommodate dynamic processes and when neat schedules require frequent updating to reflect changing circumstances.

Imposing more and more control, managers may forget their own original purpose—creating order. In such cases, they may come to believe that more control leads to more order. Unfortunately, this view doesn't account for the uncertainties inherent in the real world. Skilled professionals don't adapt well to micromanagement, and tools and techniques quickly reach their limits when not used appropriately. Managers realize that increased control doesn't yield increased order, accepting their own inability to know everything in advance while relinquishing some control to achieve greater order.

Adaptive leadership. An agile project team balances on the edge of chaos—a concept from complexity theory. Systems with too much structure are too rigid, while systems without enough structure spiral into chaos. Leading a team by nurturing small organic teams, establishing a guiding vision, establishing simple rules, championing open information exchange, and managing with a light touch is challenging enough. There's also the risk of a team veering into chaos. Nonlinear behavior can be either positive or negative in a project context, and controls placed on the system by well-intended managers can produce unintended outcomes.

Adaptive leadership employs "systems thinking" to understand a project's internal forces. For example, events are understood in terms of their patterns, or the common elements that recur in diverse circumstances. Systems archetypes reflecting common types of problems help identify the unintended and counterintuitive consequences of actions when cause and effect aren't closely related in time and space. The agile manager understands the effects of the mutual interactions among a project's various parts and steers them in the direction of continuous learning and adaptation. An adaptive APM-based framework includes several practices:

- The ability to manage and adapt to change;
- A view of organizations as fluid, adaptive systems composed of intelligent people;
- Recognition of the limits of external control in establishing order; and
- An overall humanistic problem-solving approach that:
  - Considers all members to be skilled and valuable stakeholders in team management;
  - Relies on the collective ability of autonomous teams as the basic problem-solving mechanism; and
  - Minimizes up-front planning, stressing instead adaptability to changing conditions.

Following them helps make managers adaptive leaders, setting direction, establishing simple rules for the system, and encouraging constant feedback, adaptation, and collaboration.

APM Case Study

In 2002, as part of an eight-member advisory implementation team in a Fortune 50 financial services company, two of the authors (Augustine and Payne)

led the recovery and stabilization of a large mission-critical product-development project involving a team of more than 120 IT and services professionals in multiple locations. Though the project began with a skilled team and clear mandate, project-delivery challenges emerged from the complexity of such a large team in what was for the organization a critical business endeavor. At the time we were consulted, the project was already several months behind schedule, along with frustrated customers and dispirited developers.

To resuscitate the project, we implemented XP nested within APM. We organized six development teams by general business functionality and used a SWAT team (concept from the Crystal Orange methodology [5]) to integrate code across teams at iteration end. To accommodate legacy code without extensive unit tests, we maintained a separate quality assurance (QA) team. We used APM practices to manage and coordinate all the teams. Following combined release planning, we conducted individual release planning for each of them. We initiated two-week iterations of software delivery, devoting the first to retrofitting unit tests for major sections of legacy code. At iteration end, the QA team and the SWAT team together integrated code and fixed minor defects. Users then conducted acceptance testing, while the QA team performed more rigorous manual testing.

A lack of shared understanding of the project's goals was a major issue in terms of delivering customer value. We thus established a guiding vision to serve as an internal model for all project team members, entrusting the task to a newly created project office (PO) that included all 15 of the company's business and technical project managers. It conducted release planning and translated an existing release document into an XP release plan representing the major requirements for each release iteration and embodying the specifics of the guiding vision. The PO presented it at iteration planning meetings, as well as at the daily standup meeting, reviewing it weekly to accommodate changes.

We replaced the existing project-delivery process, establishing the XP practices and values as simple rules for all 120 team members. We then initiated overall XP training, followed by intensive breakout training sessions tailored to each subgroup. To overcome the team's collective inertia, we began two-week iterations within a few days of completing the training. We then placed XP process mentors on each team to inculcate XP values and bolster our application of XP. We also held several bootstrap training sessions to reinforce XP practices.

Before beginning our work on the project, information was restricted to a select few senior managers. Our aim was to make information available to all. That's why we did the following:

- Collocated four of the six development teams in a single development bullpen area; despite the physical limitations, it proved invaluable in promoting information sharing;
- Used a war room dedicated to the project for both impromptu and formal meetings;
- Used a large whiteboard in the main bullpen area to radiate information [4]; design diagrams jostled for space with action items from the daily standup meetings, while important announcements also found their way there due to the board's convenience and effectiveness;
- Embraced the XP one-team concept; project members all had to recognize they were on the same team working toward the same goal;
- Employed pair programming to open up and share information;
- Employed the daily standup meeting as yet another way to disseminate information to team members; and
- Employed the weekly PO meeting as an information-sharing forum for both business and technical managers.

Before we arrived, managers responded to schedule slippage and frustrated customers by micromanaging developers. Schedule pressure dictated long hours. This and hasty integration periods contributed to low-quality code. We thus negotiated (with the company's senior management) a delicate balance, so developers would no longer be required to work sustained overtime. Many managers took on a new style; instead of creating, allocating, and micromanaging tasks, they gave their individual team members greater autonomy to determine which tasks had to be done while demanding demonstrable results at the end of each iteration. To keep the project on schedule and budget, they also had to do the following:

- Maintain close communication through weekly meetings and regular on-site interaction;
- Keep close watch on progress, implementing project tracking three times per iteration;
- Implement process reflections every three or four iterations to fine-tune processes;
- Earmark the first iteration for focusing on the new process while adapting to iterative delivery;
- Address meeting overload by introducing agendas to give structure to the meetings; and

- Adapt XP practice implementation, so when legacy code precluded continuous integration, a basic build running all unit tests was reimplemented as a nightly build.

Along with the PO, other useful innovations included:

- The XP practices, which many developers adopted enthusiastically;
- A release plan that emerged as the shared guiding vision;
- An XP "bills of rights" [4] for developers and customers alike, clarifying roles and responsibilities;
- A light-touch management style; for example, when executive management mandated a sudden, major GUI change involving several hundred pages of GUI code, a motivated developer spontaneously wrote scripts that automated changes to hundreds of files. The team finished the iteration ahead of schedule, impressing the business team and senior management while boosting developer confidence;
- A light touch among the management team. As the release date drew near, a business manager stepped forward to direct the entire team through the steps, both business and technical, of a readiness review; and
- A palpable project heartbeat reflecting the activities of the team members. Analysts buzzed and developers quickened the pace of their code writing toward iteration end when the SWAT and QA teams took over.

Some team managers also had to deal with a number of difficulties:

- Communicating the higher-level guiding vision (objectives, strategy) to everyone;
- Maintaining organic teams, even as senior management tended to add staff when dealing with schedule slippage;
- Suffering added stress (for conventional managers) in the agile environment;
- Recognizing that the light touch is ineffective with unmotivated and unproductive team members;
- Having to reinforce the simple rules; developers struggled with simple design in light of the large legacy code base;
- Dealing with the daily standup meeting, which, while useful for the exchange of information, was complicated by the size of the team and the cramped facilities; and
- Addressing the resentment of some senior developers toward the egalitarian nature of XP and APM, causing them to passively resist changes, despite the project's adaptive leadership.

Conclusion

Our previous experience managing projects taught us the difference between the assumptions of APM and the assumptions of traditional project management. By viewing an agile project as a CAS and adopting a leadership-collaboration model, we were able to develop a management framework with practices encapsulating agile methodologies (such as XP). Using the framework and scaling XP, we could thus lead the recovery and stabilization of a large mission-critical, product-development project, steering it to completion in five months in terms of schedule, budget, customer satisfaction, and business value. C

References

1. Abrahamsson, P., Warsta, J., Siponen, M., and Ronkainen, J. New directions in agile methods: Comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering* (May 3–10, 2003), 244–254.
2. Anthes, G. Ant colony IT. *Computerworld* (2001); www.computerworld.com/softwaretopics/software/appdev/story/0,10801,61394,00.html.
3. Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., and Slaughter, S. Is Internet-speed software development different? *IEEE Software 20,* 6 (Nov.–Dec. 2003), 70–77.
4. Beck, K. *eXtreme Programming Explained: Embrace Change.* Addison-Wesley, Reading, MA, 1999.
5. Cockburn, A. *Agile Software Development.* Addison-Wesley, Reading, MA, 2001.
6. DeMarco, T. *The Deadline: A Novel About Project Management.* Dorset House, New York, 1997.
7. Dooley, K. A nominal definition of complex adaptive systems. *The Chaos Network 8,* 1 (1996), 2–3.
8. Highsmith, J. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* Dorset House, New York, 2000.
9. Lehman, M. Rules and tools for software evolution planning and management. *Annals of Software Engineering 11,* 2 (2001).
10. Miller, G. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review 63* (1956), 81–97; www.well.com/user/smalin/miller.html.
11. Truex, D., Baskerville, R., and Klein, H. Growing systems in an emergent organization. *Commun. ACM 42,* 8 (Aug. 1999), 117–123.

Sanjiv Augustine (sanjiv.augustine@ccpace.com) is practice director for lean-agile consulting at CC Pace, a financial services consulting company in Fairfax, VA.
Bob Payne (bobpayne@webdc.com) is CEO and founder of Electroglide, Inc., a consulting firm in Washington, D.C.
Fred Sencindiver was an assistant professor of management science at George Washington University's Ashburn, VA campus and passed away before the final version of this article was completed.
Susan Woodcock (susan.woodcock@ccpace.com) is vice president for strategic services at CC Pace, a financial services consulting company in Fairfax, VA.