Agile Project Management: Steering from the Edges

Managing software development projects is a risky business. Most development efforts either fail or are not delivered on time, on budget, or with all the expected capabilities[1]. While there may be many reasons for this, one issue frequently identified is changing requirements.

Traditional software development methodologies have been characterized as describing linear, sequential processes[2]. Management approaches based upon these methodologies can be effective in developing software with stable, known, and consistent requirements. Yet, most development efforts seem not to be conducted in such stable environments. Requirements for systems deployed in these more open environments continue to change over time. Even seemingly small changes can have multiple, unanticipated effects as these systems become more complex and their components more interdependent. Management approaches based on the traditional linear development methodologies are simply a mismatch for these more dynamic systems.

Complex Adaptive Systems (CAS)

Complexity science postulates that systems can be understood by looking for patterns within their complexity, and this understanding used to describe potential evolutions of the system[3].

CAS self-organize and adapt to changes in the environment without central rules governing their behaviors. This robust adaptive behavior is an emergent property of interactions among the sub-parts and/or between the environment and the system[4].

Agents are the semi-autonomous building blocks in a CAS: they seek to maximize some measure of goodness, or fitness, by evolving over time. Simple, local rules guide the interaction between agents of a system and result in global, complex behavior.

Ant colonies are examples of CAS. Individually, ants have primitive brains, yet collectively they run surprisingly sophisticated and efficient operations. Without central direction, using a few simple rules of logic, they divide responsibilities among themselves, find food, build and maintain their nests, tend to their young, and respond to attacks[5,8].

Observing this tendency for software requirements to change over time, Meir Lehman has suggested that software systems can be characterized as increasing in complexity over time and as self-regulating systems. Lehman further argues that the evolutionary processes underlying such changing systems can be characterized as "multi-level, multi-loop, multi-agent feedback systems".[6] What is required is a methodology that can respond to these complex, unpredictable, often rapid changes in requirements.

Agile software development methodologies including eXtreme Programming[13] (XP), Crystal, SCRUM, and Feature-Driven Development provide a framework for responding to these changes with rapid iterative delivery, flexibility, and a focus on working code.

Highsmith has noted that projects that employ agile development methodologies can be considered to be Complex Adaptive Systems (CAS)[7] [**see CAS sidebar**]. Our initial experiences of applying XP to software development projects have led us to a similar view, and we have evolved a CAS-based Agile Project Management (APM) framework that leverages XP in adaptively *steering* projects to success rather than trying to force success upon them.

The framework prescribes six practices for managing agile development projects as CAS – *Guiding Vision*, *Small Dynamic Teams*, *Simple Rules*, *Open Information*, *Light Touch and Agile Vigilance*. Wrapping XP projects with these APM practices addresses the mismatch with traditional management practices. It also allows XP practices to be adapted to much larger teams than normally thought possible. The practices build on the fundamentals of CAS and XP as shown in Table 1, and are explained below.

CAS Principle	XP Manifestation	Corresponding APM Practices
Autonomous Agents		
Agents maintain *internal models* that direct their behavior.	XP values serve as an internal model.	*#1: Guiding Vision.* Recognizing and nurturing a shared project vision as an internal model translates it into a powerful influence on team behavior. *#2: Small, Dynamic Teams.* Small, dynamic teams with empowered team members form the basis for rich interactions and cooperation between team members. *#3: Light Touch.* Intelligent control of teams requires a delicate mix of imposed and emergent order.
Skill *diversity* among agents contributes to innovation and self-organization.	Collaborative practices (collective ownership, etc) enable diverse skills/experience.	
Tagging enables easy identification and organization.	Limited primary roles of *customer* and *developer* allow easy tagging.	
Strategy dictates cooperation over competition.	Game theory provides optimization in iteration and release planning activities.	
Building Blocks provide necessary abstractions to organize the environment.	XP values and practices provide a simple set of concepts to tackle most projects.	
Agent Interactions		
Local, strategic rules support *aggregation* and *emergence* in a team environment.	XP values and practices form the basis for complex behavior.	*#4: Simple Rules.* Simple Rules such as XP Practices support complex, overlaying team behavior. *#5: Open Information.* Open information is an organizing force that allows teams to adapt and react to changing conditions in the environment. *#6: Agile Vigilance.* Visionary leadership requires continuous monitoring, learning and adaptation to the environment.
Emergent order is a bottom-up manifestation of order, while imposed order is a top-down manifestation.	Simple rules, disciplined coding practices and reduced hierarchy lead to *self-organization, emergent architecture/design,* and *stability in the face of change.*	
Feedback enables change and adaptation.	Constant feedback through *tracking, frequent releases co-location, paired programming,* and the *daily standup* enable change and adaptation.	
Non-linear dynamical systems are continuously adapting when they reach a state of *dynamic equilibrium* termed the *edge of chaos*.	Sweeping changes can be rapidly accomplished utilizing *unit tests, refactoring, co-location* and *continuous integration.*	

Table 1. Evolving CAS/XP to APM Practices

Practice No. 1 – Guiding Vision: *Ensure a shared guiding vision for all team members.*

CAS agents' internal models are mechanisms for anticipation and adaptation. When a project vision is translated into a statement of project purpose and communicated to all members of the team, it serves as a shared internal model that has a powerful effect on their behavior. A real example of this principle is the use of the "commander's intent" in the U.S. Army. The Army knows that its leaders can't be omnipresent. Therefore, Army leaders clearly establish the "commander's intent" to serve as

a guide on which soldiers can base their own initiatives, actions and decisions. Thus, even if the mission falls on the shoulders of the lowest ranking person, that person can carry out the mission.

Likewise, an agile manager guides the team and continuously influences team behavior by defining, disseminating and sustaining a guiding vision that influences the internal models of individual agents, and helps the team make consistent and appropriate choices. The Agile Manifesto[9] articulates a core set of values that can be used to steer this vision.

Practice No. 2 – Small, Dynamic Teams: *Enable interactions and adaptation through close relationships and clear responsibilities.*

Self-organization and emergent order are due in part to complex interactions or flows between agents. Organizing the project into small teams implies a low *interaction penalty*[10] and can trigger this interaction. Allowing members to roll on or off the team allows dynamic team composition and enables adaptability to changing external conditions. A team size of seven, plus or minus two[11] maintains optimal channels of communication on the team, and minimizes the effect of an interaction penalty. When the project requires a larger team size, organizing the project into several smaller sub-teams working in parallel is a good compromise. The agile manager establishes clear roles and responsibilities to create team alignment and ensure accountability.

Practice No. 3 – Light Touch: *Loosen stifling control.*

With traditional development approaches, everything is seen through the prism of control: change control, risk control and most importantly – people control. Elaborate methodologies, tools and practices have evolved to try and "manage" an out-of-control world. But tools fail when neat linear task breakdowns can't easily accommodate cyclical processes, and neat schedules require frequent updating to reflect the reality of changing dates and circumstances.

In the zeal of imposing more and more control, managers may forget the original purpose of control – to create order. In such cases, managers 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. Unforeseen events can ruin the best-laid plans. Skilled professionals don't adapt well to micro-management. Tools and techniques reach their limitations quickly when used inappropriately.

With "light touch" control, managers realize that increased control doesn't cause increased order; they approach management with courage by accepting that they can't know everything in advance, and relinquish some control to achieve greater order.

Practice No. 4 – Simple Rules: *Establish and refine the team's set of practices.*

In CAS, agents follow simple rules, but their interactions result in complex behavior emerging from the bottom up over time. The standard practices of XP form a good set of simple rules for agile development projects. They're stated and agreed to by all members of the team at the outset, although the team has the ability to adjust practices that aren't working or to add new practices. Throughout the project, the project manager identifies practices that aren't followed, seeks to understand why; and removes obstacles to their implementation. Used thus, the XP practices provide simple generative rules without restricting autonomy and creativity.

Practice No. 5 – Open Information: *Provide free and open access to information.*

In CAS, information is the catalyst for change and adaptation. Interactions between agents involve the exchange of information. The richness of the interactions between agents depends in large part on the openness of the information. For an agile team 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.

Practice No. 6 – Agile Vigilance: *Continuously monitor and tune process structure.*

An agile system like a project team is one that maintains balance on the *edge of chaos* – a concept from complexity theory: systems with too much structure are too rigid, and systems without enough structure descend into unorganized chaos. Leading a team by establishing a guiding vision, nurturing small, dynamic teams, setting simple rules, championing open information, and managing with a light touch is extremely challenging. With this new, powerful model of team interaction comes the risk of the team veering off the edge. Non-linear behavior can be either positive or negative in a project context; controls placed on the system can have unintended outcomes.

Agile Vigilance employs the discipline of systems thinking in understanding the project's natural forces and using them to advantage. *Events* are understood in terms of their *patterns* – common elements that recur in diverse circumstances. *Systems archetypes* that capture the common types of problems on projects help identify unintended and counter-intuitive 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 the project's parts and steers the project towards continuous learning and adaptation on the edge.

These APM practices encapsulate the XP practices, and provide a *leadership-collaboration*[7] framework for management with:

- an intrinsic ability to manage and adapt to change;
- a view of organizations as fluid, adaptive systems composed of intelligent living beings;
- a recognition of the limits of external control in establishing order and of intelligent control as a means of establishing order; and
- an overall problem solving approach that is humanistic in that:
  - It considers all members skilled and valuable stakeholders in team management;
  - It relies on the collective ability of autonomous teams as the basic problem solving mechanism; and
  - It limits up-front planning to a minimum based on an assumption of unpredictability and instead, stresses adaptability to changing conditions.

By following these practices, the manager becomes an adaptive leader – setting direction, establishing simple, generative rules for the system, and encouraging constant feedback, adaptation, and collaboration by *steering from the edges*.

APM Case Study

In early 2002, as part of an eight member advisory team, two of the authors (Augustine and Payne) led the recovery and stabilization of a large project with a project team of over one hundred and

twenty people spanning multiple locations. Though the project began with a promising start – with a skilled team and a clear mandate – it ran into issues because of the complexity involved in managing a large team involved in a critical endeavor. When we were requested to assist with its management, it was several months behind schedule with frustrated customers and dispirited developers.

We implemented XP in conjunction with APM to resuscitate the project: to provide strong management and to scale XP, we wrapped it within our APM framework. In five months, we made the first major release on time to the day, with few bugs, fewer late hours, and delighted customers. Two releases since then have built on this success. Our approach follows.

Large-Scale Iterative Delivery

We organized six development teams by approximate business functionality as shown in Figure 1, and used a SWAT team (concept from the Crystal Orange methodology[12]) for integrating 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 teams. After conducting combined release planning, we conducted further release planning for each of the six teams individually. We initiated two-week iterations, devoting the first iteration entirely to retrofitting unit tests for major sections of legacy code. At iteration end, working with the QA team, SWAT team members integrated code and fixed minor defects. Users then conducted acceptance testing, and the QA team took over for more rigorous manual testing.

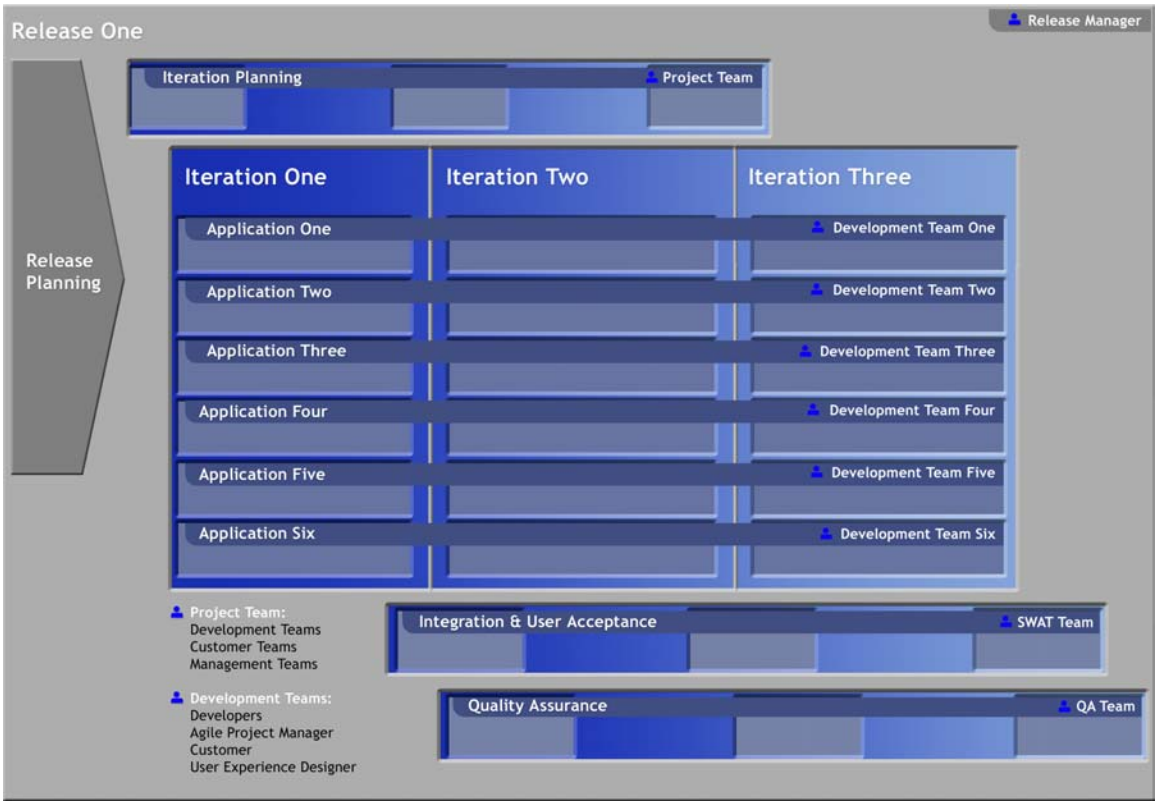

Figure 1. Iterative Delivery on a Large Project

Guiding Vision

The lack of a shared understanding of the project's end goals was one of the major factors affecting the project. Our first task was to get a shared, *Guiding Vision* in place to serve as a shared internal model for all project team members. We entrusted this task to a newly created Project Office (PO), formed with all business and technical project managers. The PO quickly conducted Release Planning and translated an existing release document into an XP Release Plan. The Release Plan represented the major requirements iteration by iteration for the release, and embodied the specifics of the *Guiding Vision*. The PO presented it at Iteration Planning meetings and at the daily standup to continuously communicate it to the whole team, and reviewed it weekly to accommodate changes.

Small, Dynamic Teams

On a large project with over a hundred and twenty people, we faced no small challenge in organizing project staff to keep them both productive and agile. Each development team contained members with diverse skills, and was organized by functionality. Developers, business analysts, testers worked together on each team to develop and deploy their respective application. As major sets of requirements were developed, members moved from team to team, reorganizing to tackle changing requirements, and spreading key knowledge across the project.

Light Touch

Previously, managers had responded to schedule slippages and frustrated customers by micromanaging developers. Schedule pressure dictated long hours. This and hasty integration periods contributed to low quality code.

To transform this situation we negotiated a delicate balance: developers were no longer required to work sustained overtime, but in exchange, they committed fully to the new approach. With difficulty, many managers moved to a different style: instead of creating, allocating and micromanaging *tasks*, they gave their teams increased autonomy to determine tasks, but now demanded demonstrable *results* at every iteration end.

Simple Rules

To replace the existing process we started from scratch. The XP practices and values needed to be established as *Simple Rules* for all members of the project team. To accomplish this, we initiated overall XP training for all team members and followed it with intensive breakout training sessions tailored to each sub-group. To overcome initial inertia, we began two-week iterations within a few days of training. We then placed XP process mentors on each team to inculcate XP values and bolster XP practice application. To reinforce XP practices over the first few months we held several bootstrap training sessions.

Open Information

Previously, information was restricted to the select few. Our challenge in this area was to make information available to all. To accomplish this:

- We co-located four of the six development teams in a single development *bullpen* area. Despite significant limitations to the physical environment, this action proved invaluable in promoting information sharing.
- A *war-room* dedicated to project use served as a convenient facility for both impromptu and formal meetings.
- A large *whiteboard* in the main bullpen served as an *information radiator*[12]: design diagrams jostled for space with action items from the daily stand-ups. Important announcements also found their way to the whiteboard because of its convenience and effectiveness.
- We embraced the XP *One Team* concept: to be successful, project members must realize that they're all part of the same team working toward the same goal.
- *Pair Programming* provided another good way to open up and share information.
- The *Daily Standup* provided another effective way of disseminating information among team members.
- For managers, both business and technical, the weekly PO meeting was a vital information-sharing forum.

Agile Vigilance

With a large team, a new process, new environment and ever-looming deadlines, there was more than enough change to handle. To handle this change, while keeping the project on track, managers:

- Maintained close communication through weekly meetings and regular on-site interaction;
- Kept a close watch on progress by implementing project tracking three times an iteration;
- Implemented process reflections every 3-4 iterations to fine-tune processes;
- Earmarked the first iteration as a clean-up iteration to focus on the new process while adapting to iterative delivery;
- Recognized and dealt with a pattern of meeting overload by optimizing the time spent in meetings. Meetings were held either early in the morning, or before close of business. Formal agendas were introduced to structure the meetings; and
- Finessed XP practice implementation: for example, when *continuous integration* couldn't be fully implemented because of legacy code and scripts, a basic build that ran all unit tests was implemented as a nightly build.

Successes

The PO proved to be a good communication forum for managers. Other successes included:

- After several iterations of successful delivery, the Release Plan emerged as the shared *Guiding Vision*, and the teams worked in alignment towards release;
- On *Small*, *Dynamic Teams*, many developers took to XP practices enthusiastically. Analysts enjoyed functioning as *on-site customer* because of the close proximity to the developers, and the satisfaction of working together to implement functionality;
- An early success demonstrated the value of *Light Touch* and contributed to its acceptance. Executive management mandated a sudden, major GUI change. Several hundred GUI

pages needed changing. Self-organization kicked in, and because of *Light Touch*, a motivated developer wrote scripts to automate changes to hundreds of files. The team finished the iteration ahead of schedule impressing the business team and senior management and giving the developers a huge confidence boost;

- On the management team, *Light Touch* was even more apparent. As the release drew nearer, a business manager stepped forward to lead it, defining and directing the entire team through all the steps, business and technical, of a readiness review. As a result, the team was ready for the release;
- The XP Bills of Rights for developer and customer served admirably in clarifying the roles and responsibilities and further reinforcing *Simple Rules*; and
- A tangible project heartbeat emerged that subsumed the activities of the team members: analysts buzzed before iteration start; developers picked up pace as iterations began, ramping up towards iteration end, and the SWAT and QA teams took over at iteration end.

Challenges

Some of the challenges we encountered were:

- Higher-level *Guiding Vision* (objectives, strategy) was very difficult to communicate to everyone, and the Release Plan had to suffice;
- *Small, Dynamic Teams* were difficult to maintain because of a tendency to add staff in the face of schedule slippage;
- Not all managers took well to the *Light Touch* practice, and the teams that had conventional managers suffered stress in the agile environment. But, with many managers in support, an increasing number of team members stepped up and completed tasks of their own volition;
- *Light Touch* proved ineffective with unmotivated and unproductive team members;
- On several occasions, the *Simple Rules* needed further reinforcement. In particular, the development team struggled with *simple design* because of the large legacy code base*;*
- The daily standup, while useful for *Open Information* was severely impacted by the large team size and poor facilities; and
- Some senior developers resented the egalitarian nature of XP and APM and passively resisted changes despite *Agile Vigilance*.

Conclusion

Through previous experience on XP projects, we understood the differences between the assumptions of agile methodologies and traditional project management. By viewing agile projects as CAS and adopting a leadership-collaboration model, we evolved an APM framework with clear practices that encapsulate the practices of agile methodologies such as XP. Using the framework and scaling XP, we led the recovery and stabilization of a large project – steering it to success from the edges in five months.

References

1. The Standish Group. The Chaos Report. http://www.standishgroup.com/sample_research/chaos_1994_1.php. 1994.
2. Pressman, R. *Software Engineering: A Practitioner's Approach*. Fifth Edition. Boston, MA: McGraw Hill. 2001.
3. Dooley, Kevin., *Complex Adaptive Systems: A Nominal Definition*, http://www.eas.asu.edu/~kdooley/casopdef.html
4. Kuschu, Ibrahim., *Adaptive Management – An Evolutionary Paradigm*, http://www.iuj.ac.jp/faculty/ik/adaptivesystems.html
5. Anthes, Gary, *Ant Colony IT*, Computerworld, http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,61394,00.html, 2001
6. Lehman, Meir, "Rules and Tools for Software Evolution Planning and Management", *Annals of Software Engineering*, 11:2, 2001.
7. Highsmith, James. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House, 2000.
8. Holland, John, *Hidden Order*, Perseus Publishing, 1996
9. The Agile Manifesto, http://www.agilemanifesto.org.
10. DeMarco, T., *The Deadline: a Novel About Project Management*, New York: Dorset House, 1997.
11. Miller, George,. *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, http://www.well.com/user/smalin/miller.html.
12. Cockburn, Alistair., *Agile Software Development*, Addison-Wesley, 2001.
13. Beck, Kent., *eXtreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.

Biographies

Sanjiv Augustine is Director, Technology at CC Pace Systems, a technology-based business solutions consulting company headquartered in Fairfax, Virginia. His technical interests include CAS, agile methodologies and organizational behavior. He received an MS in Computer Science from Virginia Commonwealth University. Sanjiv is a member of the ACM and IEEE. Contact him at sanjiv.augustine@ccpace.com.

Bob Payne is CEO and Founder of Electroglide Inc., a Washington DC based consulting firm specializing in Agile Software Development consulting, implementation, training, and technical services. He holds a MSEE in computer architecture. Bob is an Agile Alliance member and co-founder of the Washington DC XP users group. Contact him at bobpayne@webdc.com.

Fred Sencindiver is Assistant Professor of Management Science at George Washington University's Ashburn, VA Campus. He has been involved in software development and Information Systems project management for over 30 years. He holds an MS and a Ph.D. in Information Systems from George Washington University. Contact him at freds@gwu.edu.

Susan Woodcock is Director, Office of the President at CC Pace Systems. Her technical interests include knowledge and project management. Susan holds a BS in Systems Engineering from the University of Virginia. Contact her at susan.woodcock@ccpace.com.