

REGENT UNIVERSITY

LEADER-FOLLOWER COMMUNICATION ARTICLE

CHANGING STYLES IN LEADERSHIP FOR AGILE SOFTWARE PROJECTS:
CONVERSATION IS YOUR MOST POWERFUL PROJECT MANAGEMENT TOOL

SCHOOL OF LEADERSHIP STUDIES

LMOL665 ORGANIZATIONAL COMMUNICATION THEORY

MARCH 6, 2005

JOSEPH C THOMAS

INTRODUCTION

This paper presents an article containing a leader-follower communications theme. This theme relates to the communication that takes place between a project manager, the project team, and the project's customer. It is written for an audience of project managers and meets the editorial guidelines for *PM Network* magazine (for details on the author's guidelines/style sheet, see http://www.pmi.org/info/PIR_PMNetGuidelines.asp).

CHANGING STYLES IN LEADERSHIP FOR AGILE SOFTWARE PROJECTS

Conversation is Your Most Powerful Project Management Tool

By Joseph C Thomas, PMP

“Agile project management” approaches have evolved to deal with projects that employ “agile software development” approaches and each has had significant impact on the traditional practices in their respective fields. To be successful managing software development projects that use agile techniques, project managers must adapt traditional project management approaches as well as develop new strategies. Adaptation of traditional project management approaches, such as described by the PMBOK® Guide (Project Management Institute, 2003), to agile software development projects has been widely addressed (Alleman, 2001; Fitzgerald, 2004; Griffiths, 2004; Koch, 2004; Koppensteiner and Udo, 2003). When examining principles, practices, and methodologies of agile project management, a common thread that emerges is an emphasis on soft skills for effective interpersonal relationships. Project managers managing agile software development projects can benefit from developing insights into the use of conversation as an enabling technique to apply in team leadership situations. This article first summarizes the agile software development and agile project management approaches, and then discusses how project managers can use conversation as a project management tool.

Agile Software Development

Changes in traditional software development processes in recent years have been motivated to a large degree by demands to successfully deliver software with ever increasing complexity in ever decreasing time frames. The amount of detail and complexity inherent in modern software has proven extremely difficult to define in terms of unambiguous

specifications. Even when stakeholders think they know what they need and can clearly articulate it, their needs often change once they see the results they have specified. Often, project teams hear “I know that’s what I said I wanted, but now that I see it, I know I need it to be different.”

Traditional software development approaches are characterized by a waterfall methodology (a progression from one phase to another) and relatively long cycle times that do not easily provide for rapid change and midcourse corrections. Agile software development approaches (see sidebar) address these problems by employing processes that discover stakeholder needs incrementally as a project progresses. Agile software development uses short delivery cycles (weeks) to deliver working software for stakeholder review, incrementally build upon baseline components, and freely change and adapt the software as needs emerge and are prioritized by stakeholders.

Jim Highsmith (2004, p. 1), noted author and Director of Cutter Consortium's Agile Software Development & Project Management Practice (Arlington, Massachusetts, USA), states that “There are three broad situations in which [agile methods] should be strongly considered over [traditional methods]: 1) High exploration factor projects, 2) Projects in which customer responsiveness is paramount, and 3) Organizations with innovative cultures.”

Agile Project Management

Traditional project management approaches are based on control – a project can be predicted with accuracy, planned deterministically, and managed to that plan. Also, traditional project management assumes optimization – “We’ve done this before, and we can do it again only better this time” (Highsmith, 2005).

Traditional project management approaches assume that the work of developing software can be fully defined up front before the software development work actually begins. Indeed, effectiveness of such traditional techniques as critical path analysis, program evaluation and review technique (PERT), and earned value analysis (EVA) are predicated upon the assumption that the plan can be foreknown, is complete, and is correct. Development of detail work breakdown structures for agile software development projects in its entirety doesn't apply because work is planned in short iterations.

Agile project management has evolved to support agile software development. Jim Highsmith (2004) describes a life cycle of agile project management that compliments agile software development. This life cycle consists of five phases: envision, speculate, deliver, monitor/adapt, and close. The *envision phase* is initiating the project by understanding the objectives of the stakeholders and defining the project team. The *speculate phase* is planning the project by describing product features, establishing project milestones, developing a release plan, and developing the plan for iterations within the release. The *deliver phase* is executing the iteration and release plan to deliver functioning and tested product containing necessary features. The *monitor and adapt phase* is assessing the delivered product, the business environment, and project team performance and adapting the release and iteration plans accordingly. The *close phase* is project close out by wrapping up project details and celebrating. Highsmith describes nine principles for agile project management:

1. Deliver something useful
2. Cultivate committed stakeholders
3. Employ a leadership-collaboration management style
4. Build competent, collaborative teams

5. Enable team decision-making
6. Use iterative, feature-driven delivery
7. Encourage adaptability
8. Champion technical excellence
9. Accelerate throughput

Given the nature of agile software development projects, traditional project management techniques employed to plan, monitor, and control software development project work are less critical for successful projects. What then are critical success factors in project management for agile software development projects?

Conversation as a Leadership Tool

Jane Lowther and Robert Marshall (1997) of Knowledge Teams International Pty Ltd (Melbourne, Australia), as a result of their research into knowledge team performance improvements, state “[T]he factors showing the greatest difference between the most effective and least effective teams are the ‘soft’ team dynamics and process factors.” Their research shows that trust, goodwill, and cooperation between team members are the most significant determinants of team success.

Trust can be viewed as a belief about the likelihood of a person to fulfill their promises. From this perspective, trust has three dimensions (Davis, 1998):

- Sincerity. Whether a person means what they say and keeps their promises.
- Competence. Whether a person has the capability to keep their promises.
- Consideration. Whether a person appreciates and cares about others.

James Kouzes and Barry Posner (1993, p. 100) state “Building trust begins by building a personal relationship through listening.” They identify four key behaviors to build trust:

predictable behavior, clear communication, keeping promises, and honesty. Research that James Kouzes and Barry Posner (1995, p. 22) report shows that honesty, truth, and ethics is the “single most important ingredient in the leader-constituent relationship” and is what people want most from their leaders.

Relationships with people is the common theme of effective knowledge teams, desirable leader characteristics, and agile project management principles. Project managers use speaking and listening to enable these soft skills. Conversation is a key leadership tool for project managers, and conversation is a critical success factor for managing agile software development projects.

The notion of conversation as a means for organizational leadership, understanding of organizational dynamics, and rationale for organizational diagnosis and intervention is consistent with research and communication theory.

James Taylor through his research established a noted communication theory stating that organizing people is a process of interaction and interpretation between individuals that takes place in conversation (Littlejohn and Foss, 2005).

Austin and Searle developed speech act theory (Davis, 1998) that describes how the context of speech creates the future and manages organizational change through a commitment process that determines the action of people.

Davis (1998) discusses the notion of commitment as a network of personal relationships that dictate organizational effectiveness and change. Davis asserts that commitment is to the Knowledge Age as material was to the Manufacturing Age and information to the Information Age.

Flores put forth the Linguistic Action Perspective (LAP) (Howell and Macomber, 2003) as a communication theory based upon interpersonal interaction through conversation (see sidebar).

Project managers understand that getting project teams to take appropriate action is how projects are successful. Kim Krisco (1997, p. 47), of Transition Technologies (Trinidad, Colorado, USA), noted speaker, trainer, and executive coach, states that “Creating and maintaining action is central to success. The way you speak and listen is the way you create and maintain action.”

Krisco (1997) identifies several distinctions for conversation that are useful for practical application by project managers: discussion, dialogue, coaching, enrollment, and making requests.

Discussion is a form of conversation where analysis takes place to achieve understanding through dissecting the whole into its components. It’s most useful for problem solving, and is the primary type of conversation in most business and professional situations. A project manager might use discussion to facilitate problem solving with the project team by assembling the facts of the situation so that the team can realize situational understanding and take appropriate action.

Dialogue is a form of conversation where new knowledge is created through group sharing of experiences, perspectives, ideas, and awareness. Because dialogue seeks to build new knowledge, it is the opposite of discussion. Effective dialogue consists of five processes: deep generative listening for feeling and passion beyond the words being spoken, suspension of assumptions in an effort to gain new perspectives, a spirit of inquiry to see all dimensions, respect for yourself and others, and internal awareness of your own feelings and reactions. A

project manager might use dialogue to facilitate collaboration with the project team to establish or adjust team processes.

Coaching is a form of conversation where one individual creates action in another so that they can pursue specific goals to which they are committed to achieving. Coaching helps to put the world in perspective by creating an environment and context for purposeful action. There are several steps to coaching: 1) make sure that both individuals know that a coaching conversation is taking place, 2) get the facts, separating fact from interpretation, 3) develop and explore new possibilities, and 4) get some action started. A project manager might use coaching to help a team member resolve conflict or determine and pursue career aspirations.

Enrollment is a form of conversation where a commitment to action is received. Selling is promoting an idea through describing benefits and handling objections, and can also result in action. Enrollment is an alternative to selling with a higher probability of enthusiastic commitment to action. Enrollment consists of making a proper request and getting a commitment to action, but the requestor precedes the request with a personal connection and commitment, and then presents an opportunity for the individual involvement. Enthusiastic commitment is much more likely with enrollment.

Making a request is essential to accomplishing a desired outcome. Making a proper request consists of saying exactly what is desired, exactly when it is desired, and from whom it is desired. Making the request is only half of the conversation. The other half is ensuring receipt of the desired reply. There are only four proper replies: accept, decline, counteroffer, and promise to reply later. Often, an improper response to the request is received: the non-response. People are experts at avoiding requests, but a non-response can easily be overcome with follow-up questions such as “Does that mean you’ll do it?”

A project manager uses enrollment and making a request constantly in the course of interaction with the project team and other stakeholders. In lieu of detail project plans and work breakdown structures, these conversations are the basis for getting a project team to take action and are crucial as the primary source of work direction in agile software development projects.

Conclusion

Project managers that manage agile software development projects need to apply corresponding agile project management practices. The literature establishes that traditional project management approaches still apply to agile software development projects, but project managers need insight to make appropriate adjustments. In addition to new agile project management practices, soft skills relating to interpersonal skills become increasingly important to success as project teams are smaller and team member interactions are frequent and collaborative. Project managers managing agile software development projects can benefit from developing insights into the use of conversation as an enabling technique in team leadership situations.

[SIDEBAR }

Agile Software Development Explained

Randy Miller (2003), of Borland Software Corporation's Agile Software Developer Network (Scotts Valley, California, USA), describes several characteristics of agile software development methods including process modularity, process parsimony, iterative, incremental, adaptive, time-bound, people-oriented, and collaborative.

Process modularity is the ability to quickly and informally adapt project team software development processes as a result of lessons learned and situational need. A project team can press into service a particular configuration of a process as they discover need during the progression of work. Project teams also periodically reflect on the processes in use and make adjustments to improve efficiency and effectiveness.

Process parsimony refers to economy of activities. The project team questions the necessity of activities, and performs only those that are absolutely necessary to deliver properly functioning code and to mitigate pertinent risks.

The ideas of iterative, incremental, and adaptive work together to quickly deliver results. *Iterative* is delivering in short cycles (weeks). *Incremental* is delivering an application in small pieces that build upon each other; project teams may deliver these pieces in parallel, at different times, and at different rates rather than a complete application or subsystem all at once. A project team can deliver multiple increments of a component over many iterations. In this way, a project team first delivers a functioning baseline of a component that is tested and integrated into the system. Subsequently, the team builds upon this foundation adding features to it as stakeholder get and priorities ideas. *Adaptive* is the notion of a project team adjusting its work as they discover new functions, change functions already delivered, and encounter unforeseen risks. These discoveries dictate unplanned activities that the project team must address. Iterative, incremental, and adaptive provide for the reality of changing requirements as needs emerge.

Time-bound means that iterations have fixed durations (usually a few weeks) and releases have fixed durations (usually several iterations). The project team does not allow the length of an iteration to vary, and manages its work such that each iteration results in delivery of functioning software.

People-oriented refers to the project team organizing themselves into small groups that are empowered to improve their own processes through bringing recommendations to the project team at large for evaluation and implementation as appropriate. The project team also implements work schedules limited to a 40 hour week; this provides a work pace that the project team can sustain over the long haul.

Collaborative refers to project team members working closely together and a project leadership style, as Jim Highsmith (1999) says, whose “primary role is to set direction, to provide guidance, and to facilitate in connecting people and teams.” This promotes the close communication between team members that is essential for success of agile approaches.

Agile software development approaches include extreme programming (XP), Scrum, Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), lean programming, and others (Agile Alliance, 2004). The pioneers of the agile software development movement captured the spirit of these developments in the Manifesto for Agile Software Development (Agile Manifesto, 2001a):

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

These overarching themes are supported by various principles that apply to all aspects of the software development life cycle (Agile Manifesto, 2001b; Extreme Programming, 1999). Application of these principles dramatically changes how software is built, and has driven corresponding changes in project management practices for these projects.

[END OF SIDBAR]

[SIDEBAR]

Linguistic Action Perspective

Flores' Linguistic Action Perspective theory puts forth the notion that work is accomplished in an organization through the making and delivering on commitments between individuals through the process of speaking and listening. The LAP theory offers a grammar of action:

Action	Example	Definition
Declaration	"We will put a man on the moon and bring him back safely in this decade."	Creating a space of action, not to be confused with a promise.
Request	"Please deliver the submittal on Thursday."	Calling for a statement of commitment.
Promise	"You can have the crane at noon."	Statement of commitment to provide something specific by a specific time.
Assessment	"We are making good progress."	Offering an opinion with or without any basis for the assessment.
Assertion	"All tasks were completed as promised."	Statement of fact. Includes an offer to provide evidence.

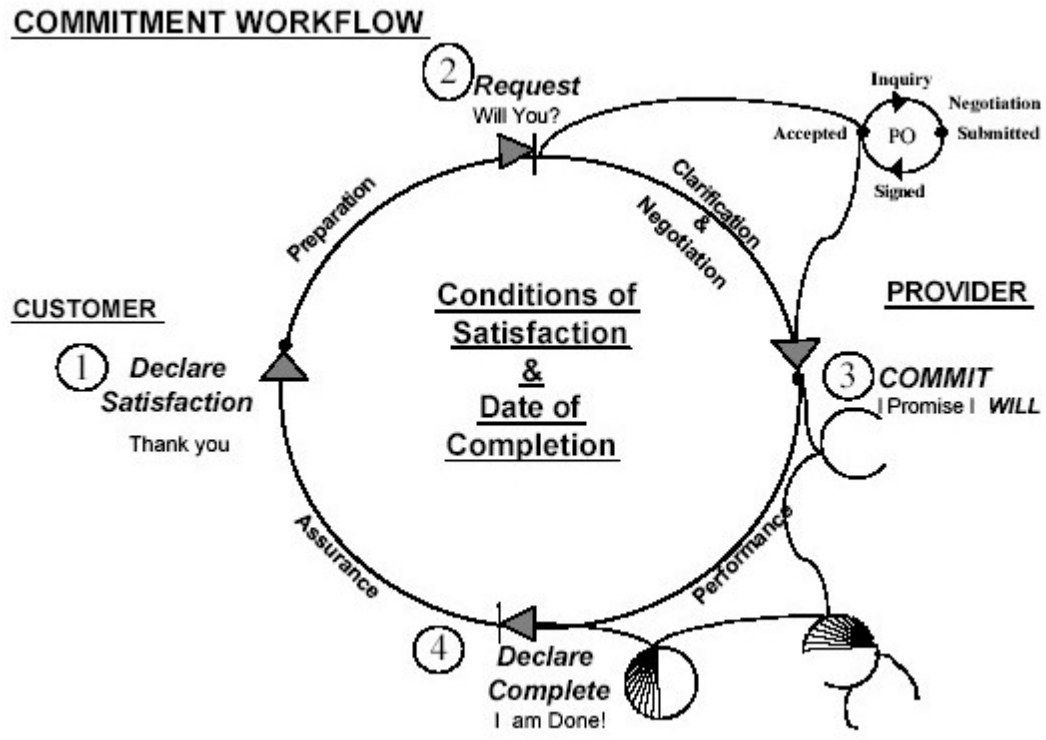


Table and graphic from Howell and Macomber (2003).

[END OF SIDBAR]

[AUTHOR BIO]

Joseph C Thomas, PMP

Mr. Thomas has over 25 years experience in software development in all roles, and a wide range of experience in computer technologies and software development methodologies. He is a Project Management Professional and holds a Bachelor of Business in Operations Research and Information Services from Eastern Michigan University (Ypsilanti, Michigan, USA) as well as a Master of Arts in Organizational Leadership from Regent University (Virginia

Beach, Virginia, USA). He is a Senior Project Manager at DTE Energy, a Fortune 250 corporation in Detroit, Michigan, USA. In this capacity, he supervises professional project managers and the project management practice in the information technology services organization. He has managed many agile software development projects over the last several years, mentored other project managers in their endeavors to manage agile software development projects, and participated in the development of agile software development methodologies, processes, tools, and project management standards and guidelines.

[END OF AUTHOR BIO]

REFERENCES

- Agile Alliance. (2004). Roadmap to Agile methods and tools: What Is Agile Software Development? Retrieved March 4, 2005 from <http://www.agilealliance.org/programs/roadmaps/Roadmap/index.htm>
- Agile Manifesto. (2001a). Manifesto for Agile Software Development. Retrieved March 4, 2005 from <http://www.agilemanifesto.org/>
- Agile Manifesto. (2001b). Principles behind the Agile Manifesto. Retrieved March 4, 2005 from <http://www.agilemanifesto.org/>
- Alleman, G. (2001). Agile Project Management and "Normative" Paradigms. Retrieved March 5, 2005 from <http://www.niwotridge.com/Essays/Editorials/PMEditorial.htm>
- Davis, C. (1998). Listening, Language and Action. Retrieved March 6, 2005 from <http://www.psych.lse.ac.uk/complexity/Seminars/1998/report98nov.htm>
- Extreme Programming. (1999). The Rules and Practices of Extreme Programming. Retrieved March 4, 2005 from <http://www.extremeprogramming.org/rules.html>
- Griffiths, M. (2004). Using Agile Alongside the PMBOK®. Retrieved March 5, 2005 from http://www.quadrus.com/resources/usingagilealongsidethepmbok_paper.pdf
- Howell, G. and Macomber, H. (2003). Linguistic Action: Contributing To the Theory of Lean Construction. Retrieved February 28, 2005 from <http://strobos.cee.vt.edu/IGLC11/PDF%20Files/01.pdf>
- Highsmith, J. (1999). Adaptive Management: Patterns for the E-Business Era. Cutter IT Journal, XII. Retrieved March 5, 2005 from <http://www.jimhighsmith.com/articles/patterns.htm>
- Highsmith, J. (2004). Agile Project Management: Principles and Tools. Retrieved March 4, 2005 from <http://www.cutter.com/research/2004/edge040309.html>
- Highsmith, J. (2005). The Changing Face of Project Management. Retrieved March 4, 2005 from <http://www.cutter.com/consultants/jhbio.html>
- Koch, A. (2004). Are Agile Methods Compatible with PMBOK? Retrieved March 5, 2005 from <http://www.pittsburghpmi.org/documents/meetings/presentations/AgileManifesto-PMBOK.pdf>
- Koppensteiner, S and Udo, N. (2003). Will Agile Development Change the Way We Manage Software Projects? Agile From a PMBOK® Guide Perspective. Retrieved March 5, 2005 from http://www.projectway.com/paper_agilevspmbok.pdf
- Kouzes, J. & Posner, B. (1993). Credibility. San Francisco: Jossey-Bass.

- Kouzes, J. & Posner, B. (1995). The Leadership Challenge. (2nd ed.). San Francisco: Jossey-Bass.
- Krisco, K. (1997). Leadership and the Art of Conversation: Conversation as a Management Tool. Rocklin CA: Prima Publishing.
- Littlejohn, S. & Foss, K.A. (2005). Theories of Human Communication. (8th ed.). Belmont, CA: Thomson Wadsworth.
- Lowther, J. and Marshall, R. (1997). Teams in the Test Tube: Building Team Performance in R and D Organizations. Retrieved March 5, 2005 from <http://www.great-teams.com/ktep/TeamsTestTube.pdf>
- Merriam-Webster. (n.d.). Retrieved March 5, 2005 from: <http://www.m-w.com/>
- Miller, R. (2003). The Dynamics of Agile Software Processes, Part I: Characteristics. Retrieved March 5, 2005 <http://bdn.borland.com/article/0,1410,29726,00.html>
- Project Management Institute. (2004). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (3rd ed.). Newtown, PA: Project Management Institute.