8

Light Touch

"Intelligent control appears as uncontrol or freedom.
And for that reason it is genuinely intelligent control.
Unintelligent control appears as external domination.
And for that reason it is really unintelligent control.
Intelligent control exerts influence without appearing to do so.
Unintelligent control tries to influence by making a show of force."

—Lao Tzu, Book of Ethics

Most project managers work in companies that have some form of hierarchical organization. Organizational hierarchies extend into our project teams as well, along with modern, subtle forms of command and control. For example, in many of our organizations, team members are still required to perform tasks specifically assigned to them by their project managers without advance consultation. In the more egalitarian of these organizations, team members may be consulted by the project manager; but in the end, the assignment of work still happens in a top-down fashion. In other organizations, the hierarchical control lies with someone other than the project manager—perhaps a line of business manager. In this case, the project manager's responsibilities are reduced to the administration of the project schedule and lots of coordination among multiple groups, but these responsibilities come with very little influence over the teams they are supposed to be managing. Top-down decisions are still made, but by the line of business manager, not the project manager or the team. In previous chapters, I contended that these structures are mechanistic ones that are constructed to optimize cost and control. Chapter 1, "Agile Project Management Defined," introduced the organic complex adaptive systems (CAS) model as the preferred alternative for agile teams with

highly skilled members whose primary charter is to deliver customer value. Chapters 3, "Organic Teams—Part 1," and 4, "Organic Teams—Part 2," detail how to construct Organic Teams based on the organic CAS model. But the question of control remains unanswered—how are agile managers supposed to control their teams that are organized according to the organic CAS model?

The objective of the *Light Touch* practice is to manage agile teams with a style that allows team autonomy and flexibility and a customer value focus without sacrificing control. The activities associated with this practice carry the following implications for agile managers:

- Establishing decentralized control that defers decision making for frequently occurring, less critical events to the team
- Managing the flow of customer value from one creative stage to another
- Recognizing team members as whole-persons and treating them accordingly
- Focusing on strengths rather than weaknesses to leverage people's uniqueness

The rest of this chapter lays out the activities you need to conduct to achieve this objective. The activities are grouped into two categories: *intelligent control* and *whole-person recognition*, and they are covered next.

ACTIVITIES

Table 8-1 shows the leadership and management responsibilities required to establish Light Touch management on an agile project team.

The activities shown in Table 8-1 are covered in detail in the rest of this chapter, beginning with those in the intelligent control category, covered next.

TABLE 8-1. ESTABLISHING LIGHT TOUCH: THE AGILE MANAGER'S RESPONSIBILITIES

CATEGORY	ACTIVITIES
Intelligent control	Management: • Decentralize control • Establish a pull task management system • Manage the flow • Use action sprints Leadership: • Fit your style to the situation • Support roving leadership • Learn to go with the flow
Whole-person recognition	Leadership: • Maintain quality of work life • Build on personal strengths • Manage commitments through personal interactions

INTELLIGENT CONTROL

"Hire good people and get out of the way." Most of us have heard this popular management maxim. When I first heard it years ago, it appealed to me because of its simplicity. But having tried to implement it, I now know that it is too simplistic in its outlook: Hiring good people works very well for the most part, but getting completely out of the way doesn't because it usually leaves a vacuum that affects the team's ability to deliver. As we have seen in the past several chapters, several things are the agile manager's sole responsibility. So, although command and control is not the way to manage agile teams, getting completely out of the way does not work either. So, what are some of the key things of which the agile manager needs to maintain control, while "getting out of the way" for the rest? Put another way, what is the way for agile managers to intelligently control the skilled professionals on their agile teams?

Intelligent control is the exertion of influence and direction with minimal top-down control. Intelligent control is needed to manage skilled professionals with a style that best allows them to fulfill their creative potential and to function as self-organized groups that react rapidly to change. The activities for you to practice intelligent control—decentralize control, establish a pull

task management system, manage the flow, use action sprints, fit your style to the situation, support roving leadership, and learn to go with the flow—are covered next.

Activity: Decentralize Control

The most important decision about control is deciding who will control what and when. On an agile project, the control system consists of the simple process rules and other working rules that the team commits to follow. A good way to decentralize control is to break out the control system into levels and distribute decision making among the levels. An agile project's control systems can be broken out into these three levels: the governing strategy and selected rule system, the rules, and the application of the rules.[1] For instance, if you have selected Scrum, then Scrum is your rule system. The reason you selected Scrum and what you want to accomplish with it is your governing strategy. The Scrum practices are your rules, and the application of Scrum practices is the rule application.

To decentralize control on your agile project, you can apply the project control system breakout shown in Figure 8-1. At level 1 where the rules are applied, there are many decisions to be made, and they need to be made frequently and quickly. These decisions have limited impact and cost. Decision making at level 1 should be delegated to individual team members, affording them a large degree of autonomy, flexibility, and speed. Level 2 is where the rules themselves are decided. These decisions take place less frequently and are fewer in number, but they have a much larger impact and cost. Decision making at level 2 should be handled by the team. Customers are considered to be part of the team.

Level 3 is where the choice of the rule system (XP, Scrum, Crystal, etc.) takes place and where corporate strategy is decided. These decisions are made occasionally and are very few, but they have the largest impact and cost. Decision making at level 3 should be handled by management. It has been my experience that agile managers participate mostly at level 2, and sometimes at level 3. Figure 8-1 also illustrates decision breakout between the levels. For example, a management strategy decision at level 3 to have a high quality of work life translates to team decisions at level 2 about appropriate work hours. In turn, related decisions about personal schedule are made by the individual team member at level 1. Similarly, a level 3 management decision to enhance knowledge transfer translates into decisions about pairing and collocation at level 2. At level 1, these decisions about the choice of a pairing partner are made by individual team members.

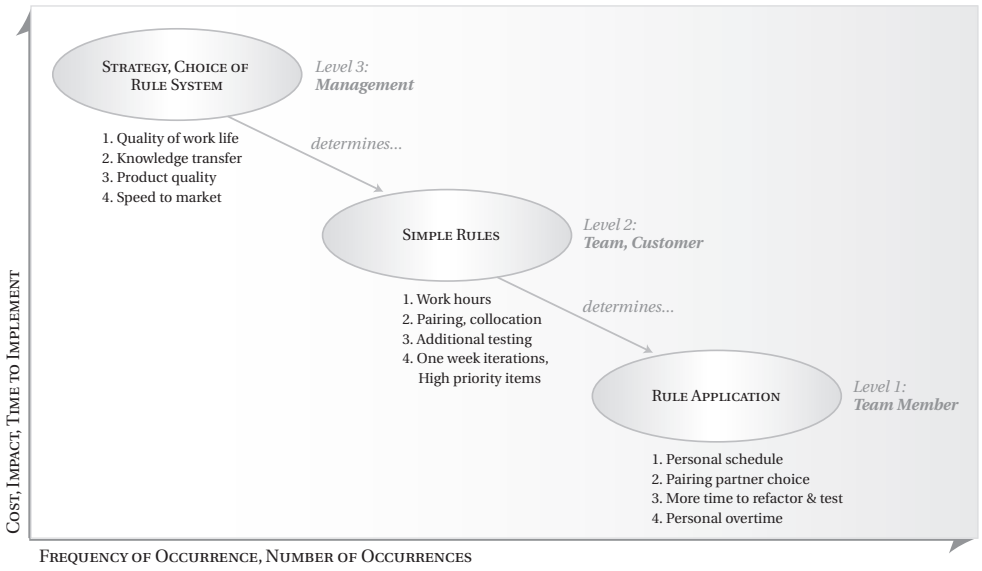


FIGURE 8-1. EXAMPLE OF DECENTRALIZED CONTROL WITH MULTIPLE CONTROL LEVELS

Activity: Establish a Pull Task Management System

A *pull* task management system is one in which tasks are "pulled" from a task queue or backlog by team members themselves, instead of "pushed" or assigned by a central coordinator, such as a project manager. Pull systems allow people to operate independently and autonomously in changing situations without wasting time waiting for work to be scheduled by someone else. On an agile team, the pull system includes prioritized *backlogs* of user stories (eXtreme Programming) or equivalent tasks (Scrum and others), as illustrated in Figure 8-2, and information radiators used as *visual controls* to indicate completion of the task to the next responsible group in the value stream.

A user story *flows* from the customer through the development value stream and back to the customer in this sequence (as shown in Figure 8-2):

1. The customer creates and prioritizes a user story representing a part of the system's functionality in iteration planning. Stories are placed along with associated tasks in an iteration plan/task backlog in order of priority. Acceptance criteria are also specified.

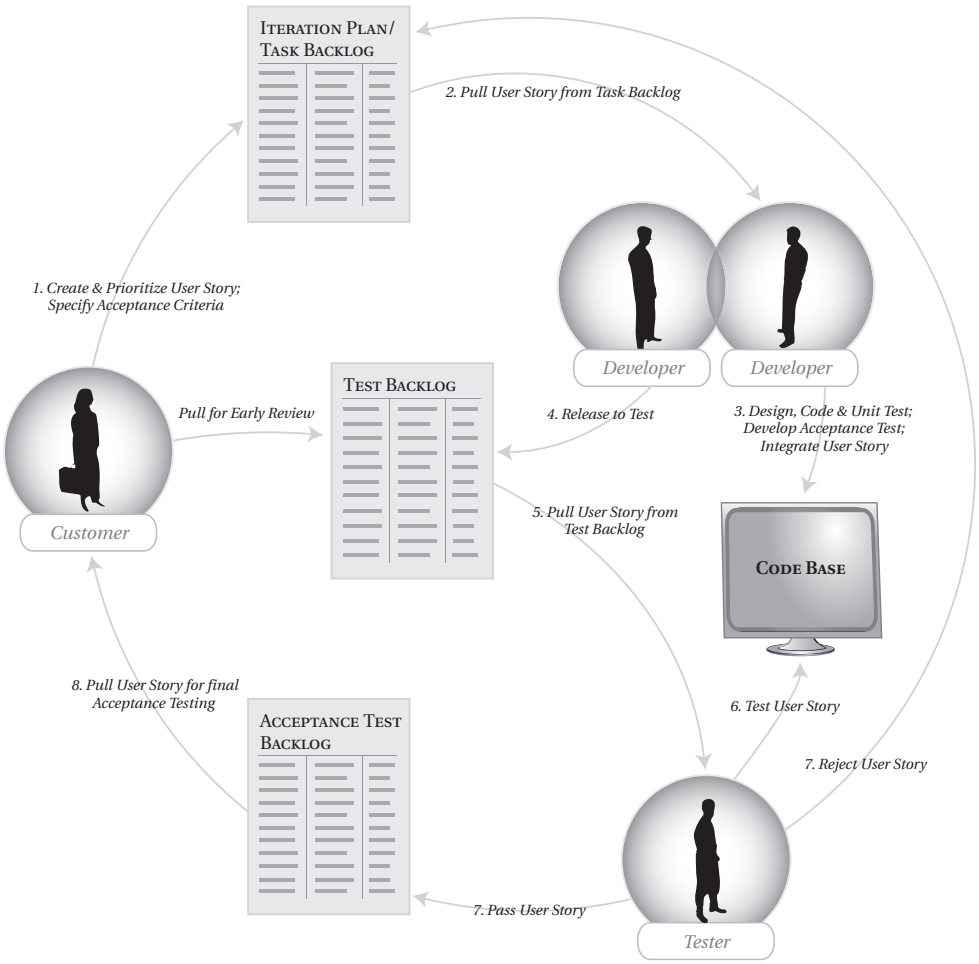


FIGURE 8-2. PULL TASK MANAGEMENT SYSTEM ON AN AGILE TEAM

2. Developers pull user stories and tasks from the iteration plan/task backlog.
3. Developers pair with other developers, business analysts, etc., to design, code, unit test, and integrate the user story into the code base.
4. When the code for the user story passes all unit and acceptance tests, developers release it to test.
5. Testers pull the user story from the test backlog for testing.
6. Testers test the user story to see whether it meets the acceptance criteria specified by the customer.

7. Testers either pass the user story and place it in the acceptance test backlog for the customers to test, or they reject it and place it once again in the iteration plan/task backlog.
8. The customer pulls user stories from the acceptance test backlog for final acceptance.

The iteration plan/task backlog is replenished and reprioritized at every iteration planning meeting. It is serviced continuously during the iteration. The test and acceptance test backlogs are replenished and serviced continuously within the iteration. You need to display visual representations of the backlogs so that team members can easily perform their work.

A Volunteer Pull Task Management System

Using a pull task management system with backlogs and visual controls is a great way to enable self-organization. This concept is not new or restricted to the software development industry. Figures 8-3 and 8-4 show a "job jar" created for a church workday by Alan Moser, a recently retired U.S. Navy captain, and junior warden at St. Barnabas Episcopal Church in Annandale, Virginia.

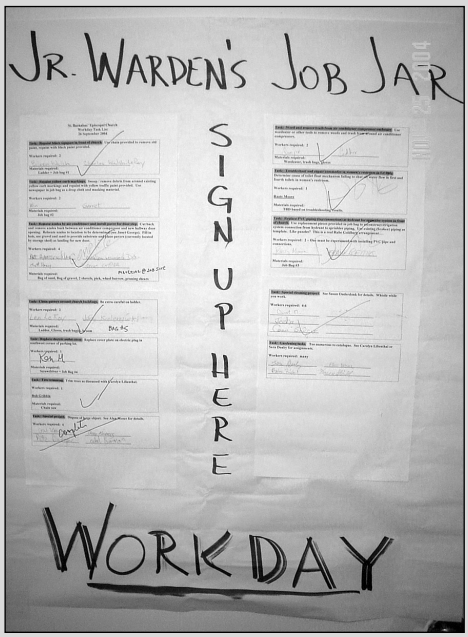
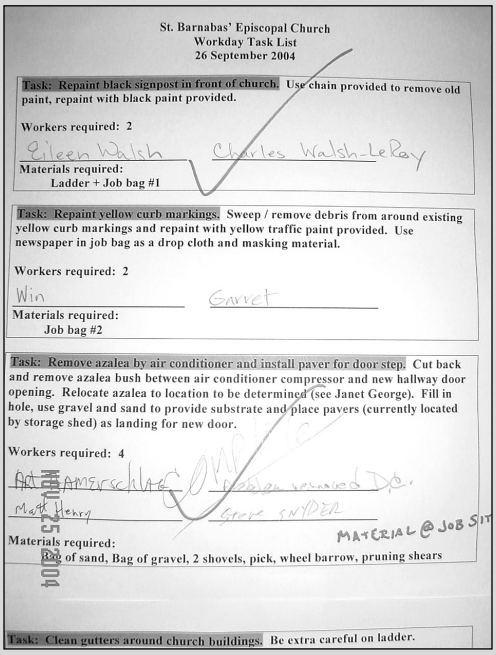


FIGURE 8-3. "JOB JAR" PULL TASK MANAGEMENT SYSTEM



St. Barnabas' Episcopal Church
Workday Task List
26 September 2004

Task: Repaint black signpost in front of church. Use chain provided to remove old paint, repaint with black paint provided.

Workers required: 2

Eileen Walsh — Charles Walsh-LeRoy

Materials required:
Ladder + Job bag #1

Task: Repaint yellow curb markings. Sweep / remove debris from around existing yellow curb markings and repaint with yellow traffic paint provided. Use newspaper in job bag as a drop cloth and masking material.

Workers required: 2

Win — Garret

Materials required:
Job bag #2

Task: Remove azalea by air conditioner and install paver for door step. Cut back and remove azalea bush between air conditioner compressor and new hallway door opening. Relocate azalea to location to be determined (see Janet George). Fill in hole, use gravel and sand to provide substrate and place pavers (currently located by storage shed) as landing for new door.

Workers required: 4

Materials required:
Bag of sand, Bag of gravel, 2 shovels, pick, wheel barrow, pruning shears

Task: Clean gutters around church buildings. Be extra careful on ladder.

FIGURE 8-4. "JOB JAR" DETAIL

On the workday, the job jar served as task backlog and visual control, and small groups of parishioners self-organized to complete the tasks, all of them working without Alan's direct supervision.

You can create charts with the user stories split into three to-do, for testing, and tested categories to serve as visual controls. These visual controls can be dynamically updated by team members as they complete their work, and serve as pull signals for the next group in the value stream to begin performing their work.

Activity: Manage the Flow

Lean Thinking has been used to reduce wastes and improve quality in many organizations for several decades with remarkable results. Besides the pull system, another key concept of Lean Thinking is continuous *flow*. Pull task management systems need to be implemented with serious thought to the flow of business value across the team. How should business value in the form of user stories be kept flowing continuously through it? In Lean organi-

zations, one-piece flow or continuous flow is employed to make one part of a system correctly and completely without interruptions and with low cycle times. Agile teams practice this concept when they define, develop, integrate, and deploy software development systems a user story at a time. The user story (in XP) or equivalent task (Scrum and others) represents the "one piece" of business value that needs to flow from the customer through development, testing, and deployment back to the customer as quickly as possible without interruptions. Pull task management helps ensure that team members are performing their work with flexibility and autonomy. So, what can the agile manager do to help the work of the team? Instead of supervising task completion, you should turn your attention to managing the flow of user stories from creation to completion.

Mary and Tom Poppendieck discuss these guidelines to avoid bottlenecks in software development queues: small batch size, steady rate of arrival and service, and slack.[2] You can apply these guidelines to manage the flow of user stories through your team's pull task management system as follows:

- *Small batch size.* Agile teams use iterative development to avoid the issues caused by large batch size—lack of early feedback, large inventory, and associated large potential waste of time and other resources. Small releases and iterative development provide two levels at which batch size can be controlled. You need to work with your customers to ensure that system functionality is being defined, created and released in small batches. At the release level, this means ensuring that feature batch size is kept small by breaking features into high-level user stories that take no longer than three weeks to implement, and that no release takes longer than three to four months, even for large projects. At the iteration level, it involves ensuring that detailed user stories that implement high-level ones represent no more than three days of work, and that iterations are kept to one, two, or three weeks in duration each.
- *Steady rate of arrival and service.* Each backlog in the agile project's task management system shown in Figure 8-2 is a queue. You need to keep an eye on all these queues to see that user stories both arrive at the respective backlog, and are serviced at a steady rate. With the iteration plan/task backlog, this is straightforward: Iteration planning is a systematic way of prioritizing and scheduling the user stories; iteration planning ensures that user stories arrive in the iteration plan/task backlog, at a steady rate. You also need to ensure that user stories are being pulled at a steady rate from the iteration plan/task backlog.

If you have an intermediate test backlog, you need to monitor it to ensure that user stories are being serviced at a steady rate by developers and arriving at the test backlog at a steady rate. Again, the user stories in the test backlog need to be serviced and passed at a steady rate by your testers to arrive at a steady rate at the acceptance test backlog. Finally, you need to monitor the acceptance test backlog to ensure that user stories are being pulled for final acceptance by your customers. Backups at any of the backlogs immediately indicate a disruption to continuous flow and, hence, a problem for you to deal with.

Take the iteration plan/task backlog, for instance. If it starts backing up within an iteration, it could either mean that your developers are having difficulties coding user stories and are not pulling new ones from it quickly enough or that testers are rejecting an inordinate share of user stories because of defects or unmet requirements. Either of these situations merits your immediate attention.

- *Slack*. Any system's performance degrades rapidly when its resources are overloaded. A software development project team is no exception. Besides, because there are humans involved, it will be even more prone to errors when utilization goes beyond 70 or 80 percent. You therefore need to ensure that you afford your team a certain amount of slack to ensure that they are consistently productive.

Use Action Sprints

Sometimes, even the best agile team will fall into a rut of creating user stories, coding them, testing them, and releasing them. People will settle into familiar roles and do what has come to be expected of them. Many members on your team may begin to get restless or bored because of the lack of variety in work and the lack of variation in method. Quality might begin to suffer and schedules might begin to slip because motivation has slipped. When this happens to me, I fall back on a technique that was introduced to me by Bob Payne, an independent XP consultant: a *sprint*. Bob came across the technique through his involvement with the Zope development community.[3]

In the Zope community, a sprint is an intense two- or three-day development session, focused on building a particular subsystem. Zope sprints differ from Scrum sprints in that they are narrowly focused and are oriented toward

technical rather than business activities. My own experience with a Zope-style sprint came on a large recovery-and-stabilization project whose managers I was responsible for coaching. Bob, who was the XP process coach, introduced the idea of a sprint as a solution for massive architectural refactoring that was needed. After consultation with all managers, we decided to devote a single iteration's worth of time to a single task—to refactor the legacy code. Everybody took part in some way or the other, just not their usual way. Six teams of more than a hundred people threw themselves into this effort. There were no formal management positions—anyone who knew the most about a particular part of the system took the lead. The pace was blistering, the pressure intense, and the goal was deliberately challenging. The entire effort was completely self-organized around a single goal. The code base developed in more than a year was refactored in a single iteration. It was a stupendous effort. That experience taught me the power of focused self-organization that a sprint can provide. Since then, I have used a variation of this technique—action sprints—on several occasions, not only to get very challenging work done in a short time, but also to identify and develop leaders on my agile teams.

An *action sprint* is a short, intensely focused activity that you can use to attack particularly difficult business- or technology-oriented problems in an unconventional way. Follow these guidelines to make the most of your action sprints:

- Focus on a single, narrow goal or action.
- Make the goal absolutely clear to everyone on the team.
- Time limit the action sprint strictly to no more than a few days.
- Dissolve all roles and responsibilities, especially management roles and responsibilities.
- Devote some time at the beginning of the action sprint for your team to come together and generate a plan.
- Participate, along with everyone else, in a hands-on fashion.

Allowing your team to conduct an action sprint requires quite a bit of trust in the team's abilities on your part, as well as the part of your organization's senior management. There is always a risk of very little resulting from it, but that is why it is time limited. On the other hand, you should seriously consider the possibility that it could yield some dramatic results for you and your organization.

Activity: Fit Your Style to the Situation

There is no "best way" to manage anything or lead everyone. Even on agile teams with their self-disciplined team members, a single leadership style simply does not exist. The reason is simple—people are complex beings. Each person's behavior springs from a lifetime of accumulated experiences, insights and values. Different people require different styles of leadership. In fact, the same people may require different styles of leadership in different situations. For instance, a software craftsman with the ability to write code without any guidance or supervision may require assistance in developing user documentation. Or an expert business analyst who deeply understands the subject behind a set of data may require help in retrieving that data from a database. An agile manager needs to be able to adapt herself to the situation to fit her team members and the situations in which they work. What is a good way for the agile manager to do this?

Paul Hersey and Ken Blanchard's Situational Leadership[4] framework categorizes a leader's necessary behavior based on the combination of direction and support needed by her follower. Accordingly, they prescribe four different styles depending on the capability and willingness of the person to perform the work, determined by asking two questions:

1. Can the person do the job?
2. Will he or she take responsibility for it?[5]

The answers to these questions determine the type of style that a leader should apply to the situation:

- The *directive* style is called for when the answers to both these questions is no—when the person both cannot do the job and will not take responsibility for it. This is the high-direction, low-support style. A leader provides high direction on the task, providing guidance on both *what* tasks are to be done and *how* to perform them. Very little support or encouragement is provided in this case.
- The *consultative* style is needed when the person cannot perform the work but is willing to take responsibility for it. This is the high-direction, high-support style. In this case, the leader still assists with the direction in both the what and how of the task, but provides a high level of support and encouragement in addition.
- The *participative* style is used when the person can perform the job but will not take responsibility for it. This is the low-direction, high-support style. There is much less direction on how to perform the task but still a high level of support and encouragement.

- The *delegative* style is applied when the answer to both questions is yes—the person can both do the job and will take responsibility for it. This is the low-direction, low-support style. Very little direction or support is provided.

Agile teams are designed to operate mainly with the delegative style. Agile team members are selected for their competence and self-discipline. However, any experienced manager knows that getting an entire team of highly competent and self-disciplined team members does not happen very often. Skill levels vary from person to person, as does the ability to self-discipline. Furthermore, skill levels for the same vary from situation to situation as well. Depending on the situation, you need to decide which one of the four styles to adopt. The picture is a little complicated, because in many cases, you will need to defer to your technical coach to provide task assistance. My personal preference is to gauge the leadership style needed for the situation and, if I cannot provide the direction necessary, I identify someone who can.

Activity: Support Roving Leadership

Roving leadership[6] is the term coined by Max DePree for unofficial leaders who rise to the occasion and take charge because of the strength of their personalities. By this definition, anyone on the team can become a leader depending on his or her response to challenging circumstances.

For instance, on one my large projects, we had a serious configuration management issue for several different reasons—legacy code integration, third-party product integration, etc. The configuration management team on this project was struggling to come up with a viable solution in time. When the release came closer and the situation became increasingly dire, one of our developers stepped up and provided the leadership and direction necessary for the configuration management team. Although he was not formally a configuration management specialist, he had recently worked for a company that develops configuration management tools. It turned out that he had just the right combination of experience necessary to perform the work, and took on the mantle of a roving leader. On another project, when I was having a difficult time answering our customer's questions, our technical coach stepped in and took charge as a roving leader to manage our response to our customer. Roving leadership like this should be common on your agile projects. What can you do to foster it?

The APM practices directly foster roving leadership. Activities such as decentralizing control and cultivating communities of practices help nurture other

leaders in the team besides you. But in the end, it is up to you to support the roving leaders as they come forth from your team to handle different situations. If you do not, roving leadership will eventually die out. What can you do to support roving leadership?

When pressure situations arise and roving leaders step forth, you need to gracefully step aside, let them handle the issue, and provide them with your full support. This is not abdicating your responsibility to lead the team. In fact, it is fulfilling your leadership responsibility in full measure and more because you are grooming the leaders of tomorrow.

Activity: Learn to Go with the Flow

There is something inherently attractive, fulfilling, and even spiritual about creative work that fulfills a vision. Creative work, including software development, seems to satisfy something very deep and primal within us. Perhaps that is why few experiences compare with working on a team that has a clear purpose and delivers clearly measurable value to its cust-omers. The experience of periods of intense concentration, close camaraderie and trust, hard work, challenge, fun, and sparks of brilliance and creativity is so fulfilling and rewarding that almost everybody wants to be a part of it. Given the right team, following the practices in this book is likely to result in this sort of intense, time-suspending, deeply rewarding experience—sometimes called *flow* (psychological flow, distinct from the value flow discussed thus far). Part of intelligent control is simply relaxing and letting this experience happen, and when it does, letting it attract team members to the work you are doing on your team. Because, after you have established the right control system and team members have assumed individual responsibility for the work that needs to be done, there will be times when you will need to do little managing. During these times, you do not need to do much besides monitor the team's progress and its value flow. Your responsibility at this point is to let your team go where it needs to go and simply immerse yourself in the experience. This activity, then, is somewhat of a nonactivity: Learn to let go and go with the flow.

WHOLE-PERSON RECOGNITION

Just like all other people, project managers have different personalities. Personality profiling tools, such as the Myers Briggs Type Indicator and the Keirsey Temperament Sorter, identify different personality types. The Myers

Briggs Type Indicator, for example, measures personal preferences on four scales: extrovert/introvert, sensate/intuitive, thinking/feeling, and judging/perceiving. It turns out that the more factual, practical, and structured personality types account for up to 44 percent of the population in general and represent many business managers, educators, and administrators.[3] Project managers with these personality types have been known to find dealing with the "soft" side of project management difficult, and may judge the material presented in this section as impractical and difficult. Project managers with other personality types—intuitive, personal, and spontaneous—will more than likely find the material here somewhat obvious and trivial. Either way, I have included the material in this section to make the point that project management is at least as much about dealing with people at a personal level as it is about tools and techniques or practices and activities.

Agile managers of all personality types need to begin to practice the softer skills of project management by recognizing a fundamental reality—your project team members are flesh-and-blood people. If you think this sounds obvious and trivial, think about the ubiquity of these terms used to refer to people: *resources*, *staff*, and *FTE*. These terms, rooted as they are in the mechanistic model, indicate a deeper problem: Our organizations are not very good at recognizing people as whole persons. At many organizations people leave important parts of their selves at the door because they are not recognized as whole persons at work.

To be strong and effective leaders of their project teams, agile managers need to recognize the wholeness of each of their team members. Each person on the team comes with a peculiar and unique mix of hopes, dreams, aspirations, philosophies, shortcomings, idiosyncrasies, personalities, moods, and emotions that go well beyond their physical selves. Now, it certainly is not up to you to manage all of these for your team members. That is primarily each individual's personal responsibility. But, to manage with a Light Touch and utilize each person's unique potential to the fullest extent, you need to begin by recognizing each one of your team members as a whole person. Activities that will help you treat your team as whole persons are *maintain quality of work life*, *build on personal strengths*, and *manage commitments through personal interactions*. These are discussed next.

Activity: Maintain Quality of Work Life

Software development is a fast-paced, demanding venture. For many professionals in today's software development world, life revolves around work. Or, at the very least, it plays a significant part in our lives. Most of us spend the

majority of our waking hours in the workplace. For instance, software development professionals in India work close to six days a week. In the United States, it is at least five days and sometime part of the weekend. Unlike our parents' generation, our work also follows us home—we remain connected to work because of the double-edged sword of modern technology. My own laptop follows me everywhere I go. There is a connection—our quality of life in general is much more dependent on the quality of our work life than ever before. How can agile managers assist their teams in maintaining a positive quality of work life, and why should they bother to do so?

Numerous studies have shown the link between quality of work life and productivity. It is also at least intuitively clear that creative activity depends on quality of work life. So, there is a strong fiscal incentive to maintain quality of work life as a means of maintaining high productivity. Besides this fiscal motivation, agile methodologies value individuals and interactions over processes and tools. So, a high quality of work life is an extension of the humanistic agile value system and an essential way of treating people as whole persons.

To maintain a high quality of work life on your team, you need to make different judgment calls based on the agile value system. Although quality of work life begins with appropriate compensation, it goes beyond that to personal growth, achievement, responsibility, and reward. Two basics that can help in this regard are sustainable pace and support for individual responsibility:

- *Sustainable pace*. XP's sustainable pace practice recommends that the team work at a pace that can be sustained over the project's long haul. XP teams do not work overtime for more than one week in a row to maintain a sustainable pace of development. You can use the sustainable pace practice to help avoid team burnout and maintain a high quality of work life.
- *Individual responsibility*. Agile teams place a premium on individual responsibility. Creating opportunities for team members to share in the responsibilities and reward of team management is an excellent way to motivate them and to enhance their quality of work life. Table 8-2 indicates some "intelligent control" ways for you to support individual responsibility and allow your team members to share in the management of the team, and thereby enhance the quality of their work lives.

Implementing XP's sustainable pace practice and allowing your team members to assume greater individual responsibility are two basic ways to enhance

quality of work life. Although circumstances will vary from team to team and from project to project, the guiding principle that you can use is to always remember that your team members are whole persons.

TABLE 8-2. CENTRALIZED RESPONSIBILITY VERSUS INDIVIDUAL RESPONSIBILITY

CENTRALIZED RESPONSIBILITY	INDIVIDUAL RESPONSIBILITY
Rigid roles with detailed job descriptions	Generalizing specialists with multiple responsibilities
Top-down control with micromanagement	Self-organization and self-discipline
Impersonal communication	Personal, face-to-face communication
Rigid specialty-focused, role-limited training	Flexible training opportunities
Sole reliance on yearly reviews for performance evaluation	Regular, "in the moment" performance evaluation and coaching
Task focus	Outcome focus

Activity: Build on Personal Strengths

Performance reviews are supposed to improve productivity by comparing employees' personal performance to some uniform "standard," and then identifying all the weaknesses to improve. I have a confession to make—I intensely dislike these annual 360-degree performance reviews. In my opinion, the whole process is tiresome, time-consuming, and marginally effective when it works. When it does not work, it turns out to be demoralizing, negatively motivating, and counterproductive. In my own performance reviews, some of my managers have complained about my difficulties in conducting these reviews. Interestingly and confusingly, some have considered me to be too lenient, whereas others have found me to be too harsh. Apparently, I am far from being alone—Marcus Buckingham and Curt Coffman's book, *First, Break All the Rules*, which is based on interviews with more than 80,000 mangers worldwide, underscores my point of view.

According to Buckingham and Coffman, the world's greatest managers recognize that trying to standardize human behavior is futile, and therefore, they do not waste their time trying to dramatically change people. Rather than focus on weaknesses, these managers build on the personal strengths of their team members and help them become more of who they already are.[6] I cannot recommend this approach enough to agile managers. For a start, it is

based on the presumption that each person is unique and has unique strengths and weaknesses—whole persons, in other words. Here is a simple example from one of my projects that illustrates how you can build on your team members' strengths.

Tom is one of our most senior and brilliant developers. A master craftsman who loves teaching almost as much as he loves programming, Tom has coached many junior developers and delivered many elegant programming solutions. He is a great learner, always researching new technologies and tools. Tom is also a strong leader of technical people because he commands their respect and affection. Despite all these gifts, Tom has a serious weakness in the eyes of the world—he can be abrasive with certain people in personal interactions. When Tom came to work on one of my projects, I was warned about a situation that he had created with a client on a previous project. Now, conventional wisdom would have had me watch for further infractions on my project, attribute them to his weakness, and write it all up on his annual review. Conventional wisdom would have him spend the rest of his tenure at our company trying to correct something that I discovered springs from his deeply rooted lack of respect for people who are not well informed.

Instead of harkening to conventional wisdom, I went with my gut feeling that Tom really could not change his attitude, at least in the time he was working with me. So, I made sure that I placed Tom in the role where he was likely to excel due to his numerous technical and analytical strengths—as technical coach. However, for all client interactions, I insisted that Tom and another team member, Linda, went as a pair. Linda is a business analyst with strong technical knowledge and great client interaction skills. Between the two of them, Tom and Linda delighted our client, delivered a great system, and the entire team had fun doing it. In short, I did not insist that Tom significantly improve his weakness, I simply worked around it and built on his many other strengths.

Activity: Manage Commitments Through Personal Interactions

In Chapter 7, "Open Information," we saw that in order to be useful, transforming exchanges between team members should result in the making, keeping, and coordination of commitments; those commitments should, in turn, result in accomplishment and action. We also saw that different types of conversations—for action, for possibility, and for disclosure—can enable action-oriented transforming exchanges. All of these—conversations, commitments, and connected action—can happen easily only when team

members on an agile project are participating regularly in close, personal interactions. To manage this network of commitments, you need to engage in close, personal interactions with team members, sponsors, and all other stakeholders.

Three main things affect all personal interactions: speaking, listening, and mood awareness. You need to attend to all three of these aspects of your personal interactions to effectively coordinate and manage the team's commitments:

- *Speaking*. When making requests of other team members, make sure your requests are clear and that they have clear conditions of satisfaction. Target your speech to generate action in others. When you make promises to your customers, ensure that your promises have clear commitments, such as completion dates. Keep your speech positive and open to develop trust.
- *Listening*. Listen carefully to your customers, sponsors, team members, and other stakeholders. Assume nothing and ask questions whenever something is even remotely unclear. Clarify conditions of satisfaction when your customer makes requests of the team. State your understanding of things regularly as an act of active listening. Listen openly and positively to give others a positive impression.
- *Mood awareness*. Pay careful attention to moods and try to shift them when necessary. Emotions and moods color how people react, speak, and listen. Positive moods generate positive thinking, speech, and listening. People are more hopeful, confident, and receptive to what you might have to say when they are in a positive mood. Negative moods generate negative thinking, speech, and listening. People are more negative and less likely to listen to what you have to say when they are in a negative mood. If you remain positive and maintain a positive mood, your presence can have a positive effect on the parties with whom you interact. If you remain aware of the moods on your project, you can even actively shift the mood in a positive direction.

By attending to your speaking, listening, and mood awareness, you can make a positive difference in the close, personal interactions you have with others on your team, and consequently, you can better coordinate commitments toward action.

SUMMARY

Most organizations have some form of hierarchical organizational structure that propagates into project teams. The organic CAS model presents a viable alternative for agile team, but questions about control remain. The objective of the *Light Touch* practice is to manage agile teams with a style that allows team autonomy and flexibility, and a customer value focus without sacrificing control. The activities for this practice fall into two categories: intelligent control and whole-person recognition.

The intelligent control activities provide agile managers with ways to intelligently control the skilled professionals on their agile teams. They include decentralize control, establish a pull task management system, manage the flow, use action sprints, fit your style to the situation, support roving leadership, and learn to go with the flow. The whole-person recognition activities help agile managers to be strong and effective leaders of their project teams by recognizing the wholeness of each of their team members. They include maintain quality of work life, build on personal strengths, and manage commitments through personal interactions.

REFERENCES

1. Reinertsen, Donald G. *Managing the Design Factory*. Simon and Schuster, 1997.
2. Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development.* Addison-Wesley, 2003.
3. From http://www.zope.org/.
4. Hersey, Paul, and Ken Blanchard. *Management of Organizational Behavior: Utilizing Human Resources*. Prentice-Hall 1981.
5. Lewis, James P. *Project Leadership*. McGraw-Hill, 2002.
6. DePree, Max. *Leadership Is an Art*. Bantam Dell, 1989.
7. Buckingham, Marcus, and Curt Coffman. *First, Break All the Rules: What the World's Greatest Managers Do Differently*. Simon and Schuster, 1999.