INTRODUCTION

Gradually, there is a global shift in the paradigm of managerial thinking from following heavy and lengthy strategies to situational and tactical behavior. It is this type of organization and management of work that will successfully maneuver in the modern dynamic realities of the world.

The development of a long and solid plan may not end. In the process of its creation, there will be such a number of changes, accounting and reflection of which will take a long period of time.

It is to overcome crises associated with a shortage of thoughts, ideas, and time that we need to use flexible processes, the essence of which is to adapt to current conditions, focus on working with people and perceiving change as an inevitable good, not a terrible problem.

The managers and specialists who have studied Agile and use it in their professional activities are certainly more competent and valuable personnel for any organization.

The purpose of graduate work is to determine the impact and effectiveness of a flexible management system for international trade enterprises, as exemplified by Smart Trading Group.

To achieve this goal we must complete the following ***tasks***:

- to consider fundamentals of the concept of flexible project management methodologies for enterprise management;
- to analyze characteristics of the main requirements and limitations for clustering methodologies;
- to provide analysis of the transition of the trading enterprise management to the flexible management methodologies;
- to determine characteristics of trading enterprise management processes of Smart Trading Group;

- to consider experience defining flexible management methodologies for enterprises;
- to determine the impact of the Smart Trading Group transition on flexible management methodologies;
- to assess ways to effectively implement flexible methodologies;
- to analyxe pilot project of forming a flexible management system in a trading enterprise.

The object of the graduate work is flexible methodologies for managing product creation processes in a trading enterprise.

The subject of research - the means of analysis of management processes in a trading company.

The main research methods are analysis, synthesis, comparison, classification, formalization and system analysis. Also used method of sociological questioning, process modeling, method of constructing management algorithms.

The practical significance of the results obtained the results of the work summarize the creation of the pilot project and its results for Smart Trading Group in 5 weeks. The conducted research makes it possible to effectively change management processes at the enterprise, increasing the number of processes and accelerating their implementation.

Support for project coordination was studied in publications of Ukrainian and foreign scientists such as Anderson A., Biloshchytsky A., Bushuyev S., Bushuyva N., Cohn M., Ehlrich D., DeMarco Tom, Detchman A., Gogunskii V., Griffin Em., Highsmith G., Sachenko A., Milosevic D., Morozov V., Osherove M., Scanlan N., Sutherland J., Surowiecki H., Turner R., Twed S. and others. In particular, the field of project management, changes, as well as the synthesis of design product configuration in various subject areas, in particular, in distributed projects is deeply studied.

Regulatory acts, Textbooks, Smart Trading Group reporting electronic documentation of the pilot project served as ***information sources for the study*.**

Structure of the work*: the work consists of 3 sections, introduction and conclusion.* The first section is devoted to the theoretical foundations of the

implementation of the formation of management of flexible methodologies for technological processes: the essence of flexible methodologies is revealed and their main types are identified, the theoretical foundations of the implementation of management operations are identified, the features of the organization of process management in the enterprise are analyzed and the most effective ways to achieve flexibility in production are highlighted.

The second section provides a general description of the enterprise under study, an analysis of its managerial and information technology activities, as well as an analysis of the interaction between performers.

In the third section, the main ways to improve the use of flexible methodologies are indicated, directions for the introduction of a pilot project with agile technology for managing enterprise processes are developed, and the effectiveness of these proposals is evaluated.

PART 1. THEORETICAL BASIS OF DEFINITION OF FLEXIBLE METHODOLOGIES OF ENTERPRISE MANAGEMENT

1.1. Fundamentals of the concept of flexible project management methodologies for enterprise management

In Scrum, we acknowledge that we can't get all of the requirements or the plans right up front. In fact, we believe that trying to do so could be dangerous because we are likely missing important knowledge, leading to the creation of a large quantity of low-quality requirements. This figure illustrates that when using a plan-driven, sequential process, a large number of requirements are produced early on when we have the least cumulative knowledge about the product. This approach is risky, because there is an illusion that we have eliminated end uncertainty. It is also potentially very wasteful when our understanding improves or things change. Plan-driven, sequential processes focus on using (or exploiting) what is currently known and predicting what isn't known. Scrum favors a more adaptive, trial-anderror approach based on appropriate use of exploration. Exploration refers to times when we choose to gain knowledge by doing some activity, such as building a prototype, creating a proof of concept, performing a study, or conducting an experiment. In other words, when faced with uncertainty, we buy information by exploring. Our tools and technologies significantly influence the cost of exploration. Historically software product development exploration has been expensive, a fact that favored a more predictive, try-to-get-it-right-up-front approach (see Figure 1.1).

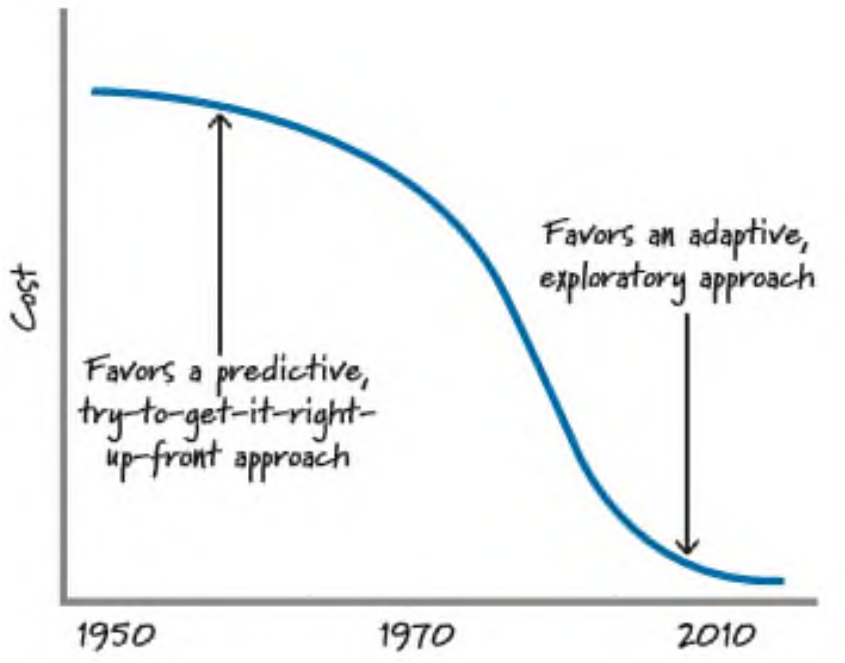


Figure 1.1. Historical cost of exploration [9]

A waterfall-style process that allowed for careful consideration of current knowledge and prediction in the presence of uncertainty in an attempt to arrive at a good solution just made economic sense. Fortunately, tools and technologies have gotten better and the cost of exploring has come way down. There is no longer an economic disincentive to explore. In fact, nowadays, it's often cheaper to adapt to user feedback based on building something fast than it is to invest in trying to get everything right up front. Good thing, too, because the context (the surrounding technologies) in which our solutions must exist is getting increasingly more complex. In Scrum, if we have enough knowledge to make an informed, reasonable step forward with our solution, we advance. However, when faced with uncertainty, rather than trying to predict it away, we use low-cost exploration to buy relevant information that we can then use to make an informed, reasonable step forward with our solution. The feedback from our action will help us determine if and when we need further exploration.

When using sequential development, change, as we have all learned, is substantially more expensive late than it is early on. As an example, a change made during analysis might cost $1; that same change made late during testing might cost $1,000. Why is this so? If we make a mistake during analysis and find it during

analysis, it is an inexpensive fix [19]. If that same error is not found until design, we have to fix not only the incorrect requirement, but potentially parts of our design based on the wrong requirement. This compounding of the error continues through each subsequent phase, making what might have been a small error to correct during analysis into a much larger error to correct in testing or operations. To avoid late changes, sequential processes seek to carefully control and minimize any changing requirements or designs by improving the accuracy of the predictions about what the system needs to do or how it is supposed to do it. Unfortunately, being excessively predictive in early-activity phases often has the opposite effect. It not only fails to eliminate change; it actually contributes to deliveries that are late and over budget as well [25, 145p.].

Why this paradoxical truth? First, the desire to eliminate expensive change forces us to overinvest in each phase—doing more work than is necessary and practical. Second, we're forced to make decisions based on important assumptions early in the process, before we have validated these assumptions with feedback from our stakeholders based on our working assets. As a result, we produce a large inventory of work products based on these assumptions. Later, this inventory will likely have to be corrected or discarded as we validate (or invalidate) our assumptions, or change happens (for example, requirements emerge or evolve), as it always will.

In Scrum, we assume that change is the norm. We believe that we can't predict away the inherent uncertainty that exists during product development by working longer and harder up front. Thus, we must be prepared to embrace change. And when that change occurs, we want the economics to be more appealing than with traditional development, even when the change happens later in the product development effort. Our goal, therefore, is to keep the cost-of-change curve flat for as long as possible—making it economically sensible to embrace even late change. Figure 1.2 illustrates this idea. We can achieve that goal by managing the amount of work in process and the flow of that work so that the cost of change when using Scrum is less affected by time than it is with sequential projects. Regardless of which product development approach we use, we want the following relationship to be true: a small change in

requirements should yield a proportionally small change in implementation and therefore in cost (obviously we would expect a larger change to cost more). Another desirable property of this relationship is that we want it to be true regardless of when the change request is made. With Scrum, we produce many work products (such as detailed requirements, designs, and test cases) in a just-in-time fashion, avoiding the creation of potentially unnecessary artifacts. As a result, when a change is made, there are typically far fewer artifacts or constraining decisions based on assumptions that might be discarded or reworked, thus keeping the cost more proportional to the size of the requested change[47, 69p.].

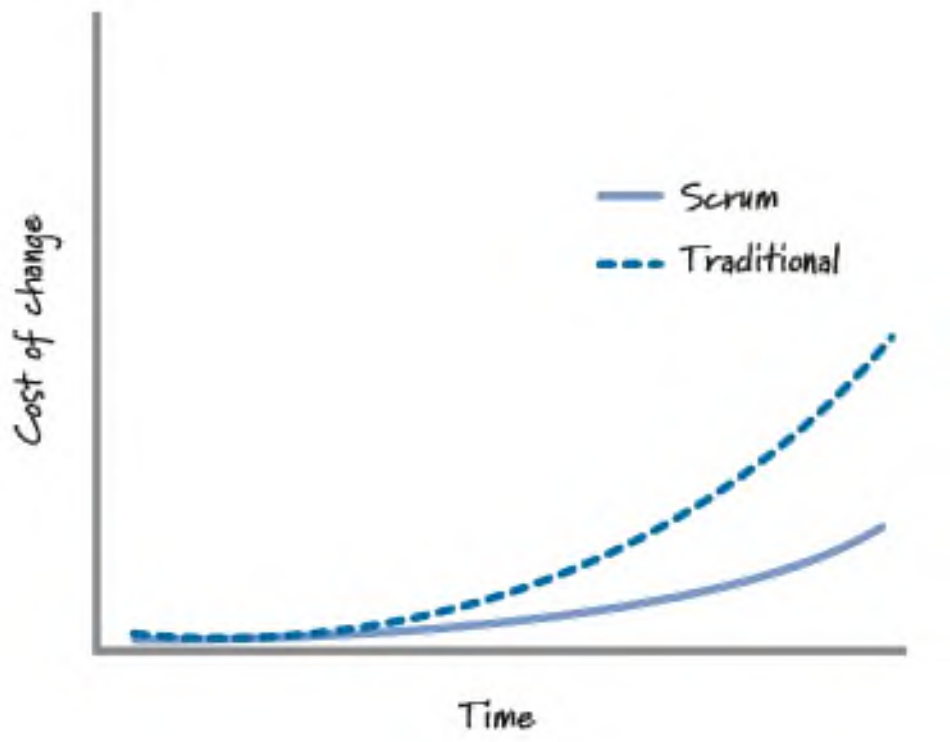


Figure 1.2 Flattening the cost-of-change curve **[34]**

Being tolerant of long-lived assumptions also makes plan-driven processes tolerant of late learning, so fast feedback is not a focus. With Scrum, we strive for fast feedback, because it is critical for helping truncate wrong paths sooner and is vital for quickly uncovering and exploiting time-sensitive, emergent opportunities. In a plan-driven development effort, every activity is planned to occur at an appointed time based on the well-defined phase sequence. This approach assumes that earlier activities can be completed without the feedback generated by later activities. As a result, there might

be a long period of time between doing something and getting feedback on what we did (hence closing the learning loop). Let's use component integration and testing as an example. Say we are developing three components in parallel. At some time these components have to be integrated and tested before we have a shippable product. Until we try to do the integration, we really don't know whether we have developed the components correctly. Attempting the integration will provide critical feedback on the component development work. Using sequential development, integration and testing wouldn't happen until the predetermined downstream phase, where many or all components would be integrated. Unfortunately, the idea that we can develop a bunch of components in parallel and then later, in an integration phase, smoothly bring them together into a cohesive whole is unlikely to work out. In fact, even with well-conceived interfaces defined before we develop the components, it's likely that something will go wrong when we integrate them (see Figure 1.3).

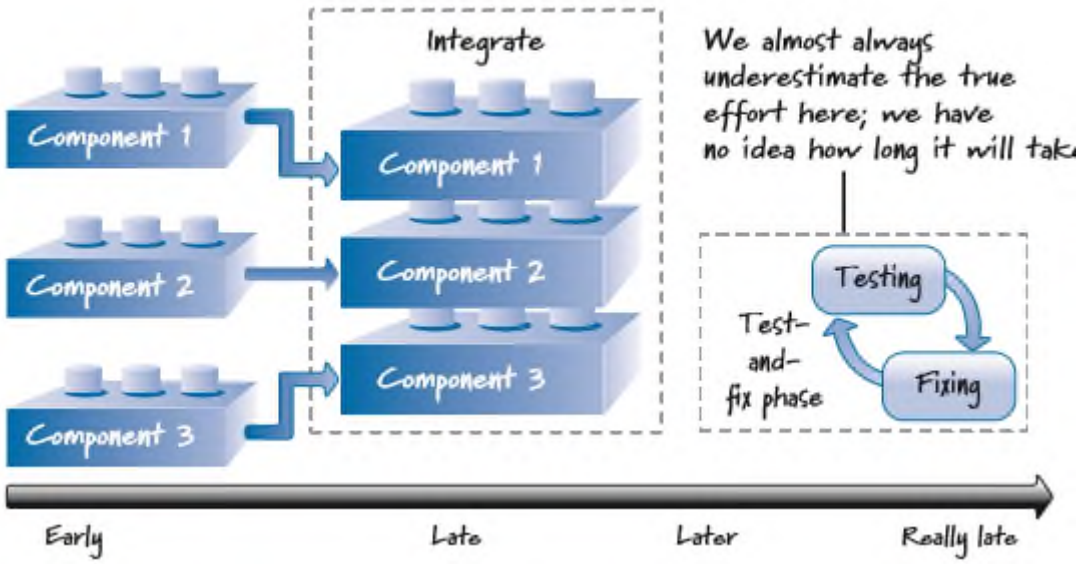


Figure 1.3 Component integration [19]

Feedback-generating activities that occur a long time after development have unfortunate side effects, such as turning integration into a large test-and-fix phase, because components developed disjointedly from each other frequently don't integrate smoothly. How long it will take and how much it will cost to fix the problem can only be guessed at this point. In Scrum, we organize the flow of work to move through the

learning loop in Figure 1.3 and get to feedback as quickly as possible. In doing so, we ensure that feedback-generating activities occur in close time proximity to the original work. Fast feedback provides superior economic benefits because errors compound when we delay feedback, resulting in exponentially larger failures.

Cost of delay is the financial cost associated with delaying work or delaying achievement of a milestone. Figure 3.15 illustrates that as capacity utilization increases, queue size and delay also increase. Therefore, by reducing the waste of idle workers (by increasing their utilization), we simultaneously increase the waste associated with idle work (work sitting in queues waiting to be serviced). Using cost of delay, we can calculate which waste is more economically damaging [51].

Unfortunately, 85% of organizations don't quantify cost of delay (Reinertsen 2009b). Combine that with the fact that most development organizations don't realize they have accumulated work (inventory) sitting in queues, and it is easy to see why their default behavior is to focus on eliminating the visible waste of idle workers. Here is a simple example to illustrate why the cost of idle work is typically much greater than the cost of idle workers. Consider this question: Should we assign a documenter to the team on the first day of development or at the end of development? Table 3.3 illustrates a comparison of these two options (there are other options we could use). Assume that we assign the documenter full-time for 12 months to work on this product, even if he is not needed 100% of the time.

Doing so costs an incremental $75K (think of this as idle worker waste) above what it would cost if we brought him on for two months at the end once the product reaches the state of "all but documented." If we assign the documenter to do all of the documentation at the end, we will need him full-time for only two months, but we will also delay the delivery of the product by the same two months. If we delay shipping the product by two months, the calculated cost of delay in terms of lifecycle profits is $500K (lifecycle profits are the total profit potential of a product over its lifetime; in this example, that potential decreases by $500K).

Table 1.1

Example Cost-of-Delay Calculation Parameter [28]

Parameter	Value
Duration with full-time documenter	12 months
Duration with documenter assigned at the end (when we reach the state of “all but documented”)	14 months
Cycle-time cost for doing documentation at the end	2 months
Cost of delay, per month	$250K
Total cost of delay	$500K
Annual fully burdened cost of documenter	$90K
Monthly fully burdened cost of documenter	$7.5K
Cost for full-time documenter	$90K
Cost for documenter if assigned at end	$15K
Incremental cost for full-time documenter	$75K

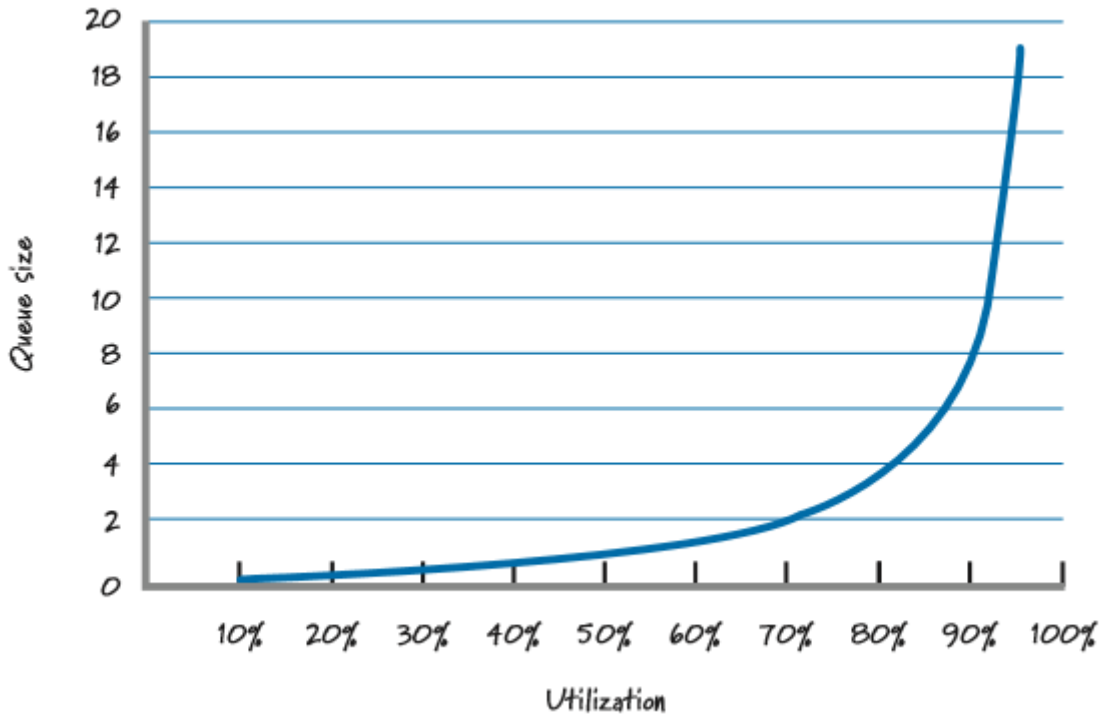


Figure 1.4. How utilization affects queue size (delay) **[32]**

In this example, the cost of the idle worker is $75K and the cost of the idle work is $500K. If we focus on optimizing the utilization of the documenter, we will substantially suboptimize the economics of the overall product. During product development we are presented with these types of trade-offs on a continuous basis; cost of delay will be one of the most important variables to consider when making economically sensible decisions.

Frequently the Scrum focus on minimally sufficient ceremony is misinterpreted to mean things like “Scrum is anti-documentation.” Scrum isn’t anti-documentation. Rather, when using Scrum, we adopt an economic perspective and carefully review which documents we create. If we write a document that is shelfware and adds no value, we have wasted our time and money creating a dead document. However, not all documents are dead [45].

It is a deliverable as part of the product (for example, installation instructions, user’s guide, and so on). Our goal is to capture an important discussion, decision, or agreement so that in the future we will have a clear recollection of what was discussed, decided, or agreed to

It is the high-value way of helping new team members come up to speed quickly. There is a regulatory requirement that a certain document be written (a cost of doing business in a regulated industry)

What we are trying to avoid is work that adds no short-term or long-term economic value. In Scrum, we believe that time and money are better spent delivering customer value.

Table 1.2

Comparison Summary of Plan-Driven and Agile Principles [17]

Topic	Plan-Driven Principle	Agile Principle
Similarity between development and manufacturing	Both follow a defined process.	Development isn’t manufacturing; development creates the recipe for the product.
Process structure	Development is phase-based and sequential.	Development should be iterative and incremental.
Degree of process and product variability	Try to eliminate process and product variability.	Leverage variability through inspection, adaptation, and

		transparency.
Uncertainty management	Eliminate end uncertainty first, and then means uncertainty.	Reduce uncertainties simultaneously
Decision making	Make each decision in its proper phase.	Keep options open
Getting it right the first time	Assumes we have all of the correct information up front to create the requirements and plans.	We can't get it right up front.
Exploration versus exploitation	Exploit what is currently known and predict what isn't known.	Favor an adaptive, exploratory approach
Change/emergence	Change is disruptive to plans and expensive, so it should be avoided.	Embrace change in an economically sensible way.
Predictive versus adaptive	The process is highly predictive.	Balance predictive up-front work with adaptive just-in-time work.
Assumptions (unvalidated knowledge)	The process is tolerant of long-lived assumptions	Validate important assumptions fast.
Feedback	Critical learning occurs on one major analyze-design-codetest loop.	Leverage multiple concurrent learning loops.
Fast feedback	The process is tolerant of late learning.	Organize workflow for fast feedback.
Batch size (how much work is completed before the next activity can start)	Batches are large, frequently 100%—all before any. Economies of scale should apply.	Use smaller, economically sensible batch sizes.
Inventory/work in process (WIP)	Inventory isn't part of the belief system so is not a focus.	Recognize inventory and manage it to achieve good flow.
People versus work waste	Allocate people to achieve high levels of utilization.	Focus on idle work, not idle workers.
Cost of delay	Cost of delay is rarely considered.	Always consider cost of delay.
Conformance to plan	Conformance is considered a primary means of achieving a good result.	Adapt and replan rather than conform to a plan
Progress.	Demonstrate progress by progressing through stages or phases	Measure progress by validating working assets
Centricity.	Process-centric—follow the process	Value-centric—deliver the value
Speed	Follow the process; do things right the first time and go fast.	Go fast but never hurry.
When we get high quality	Quality comes at the end, after an extensive test-and-fix phase	Build quality in from the beginning
Formality (ceremony)	Formality (well-defined procedures and checkpoints) is	Employ minimally sufficient ceremony

	important to effective execution.	

1.2. Characteristics of the main requirements and limitations for clustering methodologies

Scrum is an agile approach for developing innovative products and services. Figure 1.1 shows a simple, generic, agile development approach. With an agile approach, you begin by creating a product backlog—a prioritized list of the features and other capabilities needed to develop a successful product. Guided by the product backlog, you always work on the most important or highestpriority items first. When you run out of resources (such as time), any work that didn't get completed will be of lower priority than the completed work [23,27].

The work itself is performed in short, timeboxed iterations, which usually range from a week to a calendar month in length. During each iteration, a self-organizing, cross-functional team does all of the work—such as designing, building, and testing—required to produce completed, working features that could be put into production. Typically the amount of work in the product backlog is much greater than can be completed by a team in one short-duration iteration. So, at the start of each iteration, the team plans which high-priority subset of the product backlog to create in the upcoming iteration. In Figure 1.1, for example, the team has agreed that it can create features A, B, and C. At the end of the iteration, the team reviews the completed features with the stakeholders to get their feedback. Based on the feedback, the product owner and team can alter both what they plan to work on next and how the team plans to do the work. For example, if the stakeholders see a completed feature and then realize that another feature that they never considered must also be included in the product, the product owner can simply create a new item representing that feature and insert it into the product backlog in the correct order to be worked on in a future iteration. At the end of each iteration, the team should have a potentially

shippable product (or increment of the product), one that can be released if appropriate. If releasing after each iteration isn't appropriate, a set of features from multiple iterations can be released together.

As each iteration ends, the whole process is begun anew with the planning of the next iteration.

Scrum Origins Scrum's rich history can be traced back to a 1986 Harvard Business Review article, "The New New Product Development Game" (Takeuchi and Nonaka 1986). This article describes how companies such as Honda, Canon, and Fuji-Xerox produced world-class results using a scalable, team-based approach to all-at-once product development. It also emphasizes the importance of empowered, self-organizing teams and outlines management's role in the development process. The 1986 article was influential in weaving together many of the concepts that gave rise to what today we call Scrum. Scrum is not an acronym, but rather a term borrowed from the sport of rugby, where it refers to a way of restarting a game after an accidental infringement or when the ball has gone out of play. Even if you are not a rugby aficionado, you have probably seen a scrum where the two sets of forwards mass together around the ball with locked arms and, with their heads down, struggle to gain possession of the ball.

In 1993, Jeff Sutherland and his team at Easel Corporation created the Scrum process for use on a software development effort by combining concepts from the 1986 article with concepts from object-oriented development, empirical process control, iterative and incremental development, software process and productivity research, and complex adaptive systems. In 1995, Ken Schwaber published the first paper on Scrum at OOPSLA 1995 [39]. Since then, Schwaber and Sutherland, together and separately, have produced several Scrum-specific publications, including Agile Software Development with Scrum [41], Agile Project Management with Scrum (Schwaber 2004), and "The Scrum Guide" [39]. Though Scrum is most commonly used to develop software products, the core values and principles of Scrum can and are being used to develop different types of products or to organize the flow of various types of work. For example, I have worked with organizations that have successfully

used Scrum for organizing and managing the work associated with hardware development, marketing programs, and sales initiatives.

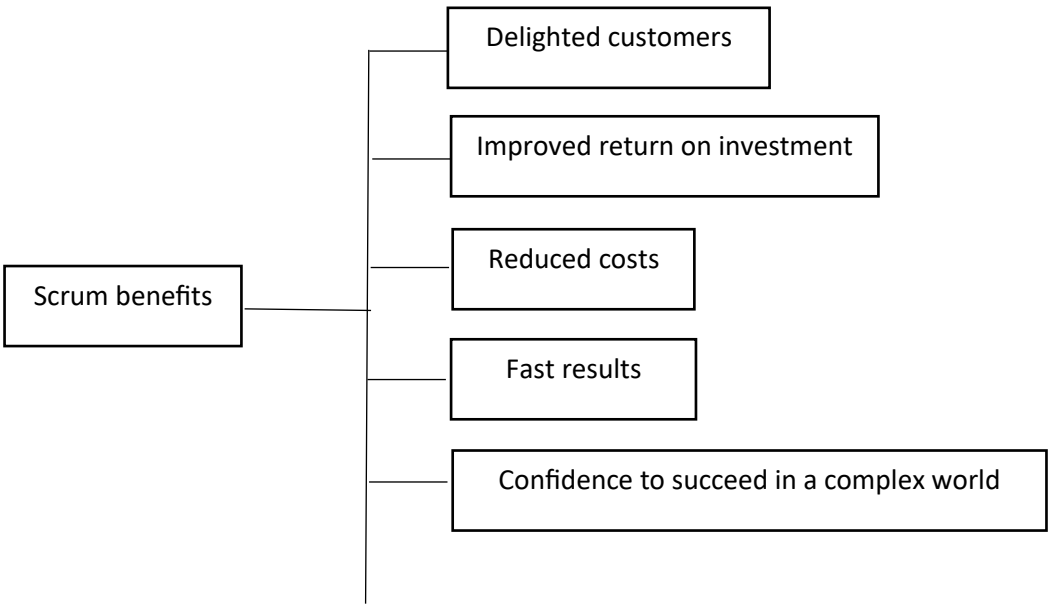
Why Scrum? That was the bad news; the good news was that most everyone agreed. Genomica operated in a complex domain where more was unknown than known. We built products that had never been built before. Our focus was on bleeding-edge, continuously evolving, state-of-the-art, discovery informatics platforms that research scientists would use to help discover the next blockbuster molecule. We needed a way of developing that would allow us to quickly explore new ideas and approaches and learn fast which solutions were viable and which were not. We had a strategic corporate partner to whom we needed to show working results every few weeks or so to get feedback, because our product had to integrate with its core line of DNA sequencers.

This need for rapid exploration and feedback did not mesh well with the detailed, upfront planning we had been doing. We also wanted to avoid big up-front architecture design. A previous attempt to create a next generation of Genomica's core product had seen the organization spend almost one year doing architecture-only work to create a grand, unified bioinformatics platform. When the first real scientist-facing application was put on top of that architecture, and we finally validated design decisions made many months earlier, it took 42 seconds to tab from one field on the screen to the next field.

If you think a typical user is impatient, imagine a molecular biologist with a Ph.D. having to wait 42 seconds! It was a disaster. We needed a different, more balanced approach to design, which included some design up front combined with a healthy dose of emergent, justin-time design. We also wanted our teams to be more cross-functional. Historically Genomica operated like most organizations. Development would hand off work to the test teams only after it was fully completed. We now had a desire for all team members to synchronize frequently—daily was the goal. In the past, errors were compounded because important issues were being discussed too late in the development effort. People in different areas weren't

communicating frequently enough. For these reasons, and others, we determined that Scrum would be a good fit for Genomica [28].

The Genomica pre-Scrum experience of building features that nobody wanted and delivering those features late and with poor quality is not uncommon. Genomica, like many other organizations, had survived by being no worse than its competitors. I saw the same problems when I first started working in commercial software development in the mid-1980s. And for many, after nearly 30 years, the situation hasn't improved. Today, if you gathered together your business people and developers and asked them, "Are you happy with the results of our software development efforts?" or "Do you think we deliver good customer value in a timely, economical, and quality manner?" what would they say? More often than not, the people I meet during my worldwide training and coaching answer both questions with a resounding "No." This is followed by a chorus of "Project failure rate is unacceptably high"; "Deliverables are late"; "Return on investment frequently falls short of expectations"; "Software quality is poor"; "Productivity is embarrassing"; "No one is accountable for outcomes"; "Employee morale is low"; "Employee turnover is too high." Then there's the under-the-breath snicker that accompanies the tongue-in-cheek "There must be a better way." Yet even with all this discontent, most people seem resigned to the fact that dissatisfaction is just part of the reality of software development. It doesn't have to be. Organizations that have diligently applied Scrum are experiencing a different reality (see Figure 1.5).



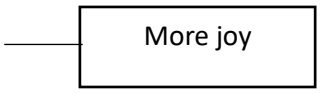


Figure 1.5. Scrum benefits **[78]**

These organizations are repeatedly delighting their customers by giving them what they really want, not just the features they might have specified on the first day when they knew the least about their true needs. They are also seeing an improved return on investment by delivering smaller, more frequent releases [19]. And, by relentlessly exposing organizational dysfunction and waste, these organizations are able to reduce costs. Scrum's focus on delivering working, integrated, tested, business-valuable features each iteration leads to results being delivered fast. Scrum is also well suited to help organizations succeed in a complex world where they must quickly adapt based on the interconnected actions of competitors, customers, users, regulatory bodies, and other stakeholders. And Scrum provides more joy for all participants. Not only are customers delighted, but also the people doing the work actually enjoy it! They enjoy frequent and meaningful collaboration, leading to improved interpersonal relationships and greater mutual trust among team members. Don't get me wrong. Though Scrum is an excellent solution for many situations, it is not the proper solution in all circumstances. The Cynefin framework (Snowden and Boone 2007) is a sense-making framework that helps us understand the situation in which we have to operate and decide on a situation-appropriate approach. It defines and compares the characteristics of five different domains: simple, complicated, chaotic, complex, and a fifth domain, disorder, which occurs when you don't know which other domain you are in (see Figure 1.6). We will use the Cynefin framework to discuss situations in which Scrum is and is not a good fit. First, it is important to realize that the many facets of software development and support will not fit nicely into just one Cynefin domain. Software development is a rich endeavor, with aspects that overlap and activities that fall into all of the different domains (Pelrine 2011). So, while most software development work falls in the domains of complicated or complex, to boldly

claim that software development is a complex domain would be naive, especially if we define software development to include the spectrum of work ranging from innovative new-product development, ongoing software product maintenance, and operations and support.

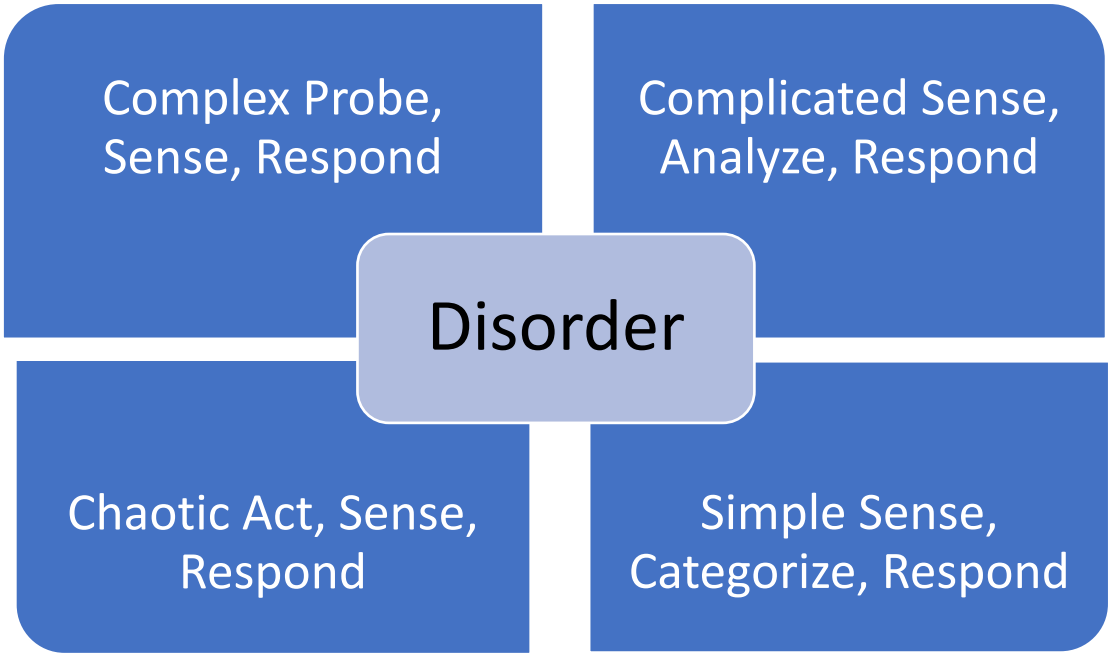


Figure 1.6. Cynefin framework [49]

Complex Domain. When dealing with complex problems, things are more unpredictable than they are predictable. If there is a right answer, we will know it only with hindsight. This is the domain of emergence. We need to explore to learn about the problem, then inspect and adapt based on our learning. Working in complex domains requires creative and innovative approaches. Routine, cookie-cutter solutions simply don't apply. We need to create a safe-fail environment for experimentation so that we can discover important information. In this environment high levels of interaction and communication are essential. Innovative new-product development falls into this category as does enhancing existing products with innovative new features. Scrum is particularly well suited for operating in a complex domain. In such situations our ability to probe (explore), sense (inspect), and respond (adapt) is critical.

Complicated problems are the domain of good practices dominated by experts. There might be multiple right answers, but expert diagnosis is required to figure them out. Although Scrum can certainly work with these problems, it might not be the best solution. For example, a performance optimization effort that calls for adjusting parameters to find the best overall system performance would be better served by assembling experts and letting them assess the situation, investigate several options, and base their response on good practice. Much of day-to-day software maintenance (dealing with a flow of product support or defect issues) falls into this category. This is also where many of the tactical, quantitative approaches like Six Sigma are particularly well suited, although these tactical approaches can also apply with simple domains.

Simple Domain When dealing with simple problems, everyone can see cause and effect. Often the right answer is obvious and undisputed. This is the domain of legitimate best practices. There are known solutions. Once we assess the facts of our situation, we can determine the proper predefined solution to use. Scrum can be used for simple problems, but it may not be the most efficient tool for this type of problem. Using a process with a well-defined, repeatable set of steps that are known to solve the problem would be a better fit. For example, if we want to reproduce the same product over and over again, a well-defined assembly-line process would be a better fit than Scrum. Or deploying the same commercial-off-the-shelf (COTS) product into the 100th customer environment might best be completed by repeating a well-defined and proven set of steps for installing and configuring the product [61].

Chaotic Domain Chaotic problems require a rapid response. We are in a crisis and need to act immediately to prevent further harm and reestablish at least some order. For example, suppose a university published an article stating that our product has a flawed algorithm that is producing erroneous results. Our customers have made substantial business investments based on the results from our product, and they are filing lawsuits against us for large damages. Our lead algorithm designer is on holiday in the jungles of Borneo and can’t be reached for two more weeks. Scrum is not the best solution here. We are not interested in prioritizing a backlog of work and

determining what work to perform in the next iteration. We need the ability to act immediately and decisively to stem the bleeding. With chaotic problems, someone needs to take charge of the situation and act.

Disorder You are in the disorder domain when you don't know which of the other domains you are in. This is a dangerous place to be because you don't know how to make sense of your situation. In such cases, people tend to interpret and act according to their personal preference for action. In software development, many people are familiar with and therefore have a personal preference for phase-based, sequential approaches that work well in simple domains. Unfortunately, these tend to be a rather poor fit for much of software development. When you are in the disorder domain, the way out is to break down the situation into constituent parts and assign each to one of the other four domains. You are not trying to apply Scrum in the disorder domain; you are trying to get out of this domain [37].

Interrupt-Driven Work Scrum is not well suited to highly interrupt-driven work. Say you run a customer support organization and you want to use Scrum to organize and manage your support activities. Your product backlog is populated on a continuous basis as you receive support requests via phone or email. At no point in time do you have a product backlog that extends very far into the future, and the content and order of your backlog could change frequently (perhaps hourly or every few minutes). In this situation, you will not be able to reliably plan iterations of a week or more because you won't know what the work will be that far into the future. And, even if you think you know the work, there is a very good likelihood that a high-priority support request will arrive and preempt any such forward-looking plans. In interrupt-driven environments you would be better off considering an alternative agile approach called Kanban. Kanban is not a stand-alone process solution, but instead an approach that is overlaid on an existing process. In particular, Kanban advocates that you.

Visualize how the work flows through the system (for example, the steps that the support organization takes to resolve a support request) Limit the work in process (WIP) at each step to ensure that you are not doing more work than you have the

capacity to do Measure and optimize the flow of the work through the system to make continuous improvements [9].

The sweet spots for Kanban are the software maintenance and support areas. Some Kanban practitioners point out that Kanban's focus on eliminating overburden (by aligning WIP with capacity) and reducing variability in flow while encouraging an evolutionary approach to change makes it appropriate to use in complex domains as well. Scrum and Kanban are both agile approaches to development, and each has strengths and weaknesses that should be considered once you make sense of the domain in which you are operating. In some organizations both Scrum and Kanban can be used to address the different system needs that coexist. For example, Scrum can be used for new-product development and Kanban for interrupt-driven support and maintenance.

Closing Scrum is not a silver bullet or a magic cure. Scrum can, however, enable you to embrace the changes that accompany all complex product development efforts. And it can, and has, worked for Genomica and many other companies that decided to employ an approach to software development that better matched their circumstances. Although the Scrum framework is simple, it would be a mistake to assume that Scrum is easy and painless to apply [7].

1.3. Flexible metodology's tranfomation and pilot project's management

The cost of migrating. What will you say when the executives ask about cost? In the model proposed, your main expense will be obtaining a knowledgeable agile coach or consulting company to come in and train and mentor your team. This usually involves 2 to 10 days of training, with several phone calls and one-off consulting sessions post training. These services can run from US$2,000 to $50,000, depending on the length of the engagement and the level of agile expertise already present in your company. The other expenses are less tangible. They're frequently labeled *soft* expenses because they don't add to company outlay but reallocate existing employees. This will be true of your core team. Over a 3-month period, the core team may spend

10 percent of their time working on the new agile methodology. Other costs are relatively minor, such as printing out materials to support training. The last expense of note is slower delivery. You can expect the first few projects to be slower as the team gets comfortable with the new processes and each other. After an acclimation period, the team will gel around the process and you will start to deliver high-priority features sooner than before; but patience is required during the first few projects.

An analogy that comes to mind is automobile reviews. We subscribe to magazines

such as *Motor Trend,* and we frequently read the road-test reviews for new vehicles. Almost every auto review laments the position of the shifter, the strange angle of the seats, or the lack of cup holders. The test driver may not feel comfortable in the car and may even prefer the previous model. If you buy the same magazine six months later, it will contain the long-term roadtest results for the same vehicle. Frequently, the extended review will say something like, "Although initially quirky, the position of the shifter becomes intuitive with longterm use and simplifies the shifting process. We also found the seating position to be excellent for long-distance road trips." Your migration to agile will be similar. After you get comfortable with how agile works, you'll find it "becomes intuitive with longterm use and simplifies the development process."

While we're discussing the cost of migrating to agile, we should also consider the cost of *not* migrating. Reflect on the reasons for pursuing agile listed in the previous section, and imagine what will happen if you *don't* address those issues:

- Declining customer satisfaction
- Loss of key employees
- Missed deadlines for compliance-related projects
- Lost sales

Lost product opportunities Dr. Phil frequently asks his guests a question that relates to migrating to agile: "How is your current process working for you?" This is Dr. Phil's subtle way of saying that if what you're doing today isn't working for you, you need to make a change.

The risks in migrating. If you migrate to agile correctly, we believe the risks are minimal. But we'll list some that can occur with poor management of the migration process:

■ You can fail on a critical project if you use it to pilot your agile methodology.

The first few agile projects shouldn't be mission critical. Begin with projects that have medium priority, and work your way up to critical ones.

■ The migration can fail if it's executive driven and there is disregard for pursuing employee buy-in.

■ Projects may be affected if employees hear about the work the core team is doing and decide to experiment without guidance. We've seen teams take one agile practice and try it with disregard for how it needs to dovetail into the upstream and downstream processes.

■ The migration can lead to cowboy coding and insufficient documentation with improper training and coaching.

■ The migration can fail if you obtain too much coaching. You can end up rolling out a process that the consultant likes versus one that provides value in your environment.

■ The migration can fail if you obtain too little coaching. Many teams have labeled their lifecycle as agile after adding one or two agile practices. In these instances, the improvements are marginal, and a true migration doesn't occur. You can also comfort your executives and mitigate risk by following the process outlined

■ Completing an assessment to see your potential for adding agility

■ Identifying people within your organization who are passionate about improving your process and involving them in the migration

■ Reviewing your existing process to identify logical places to add agility

■ Performing a pilot on a non-mission-critical project to identify potential issues before attempting to scale agile across the entire company

It's fair to ask "What will agile do for me?" on a personal level.

The answer to this question is usually unique. More than likely, the executives will never tell you the answer to this question directly. You must deduce the best way

to make them look good. Here are a few ways we've seen an agile migration satisfy the personal needs of executives:

■ Agile allows executives to acquire new skills and knowledge that enhance their value to the company and increase their chances for promotion.

■ The move to agile provides an opportunity to demonstrate leadership skills by leading a major organizational change. The executive sponsor reaps this reward.

■ The migration to agile leads to more wealth. Like most of us, executives care about their compensation. Migrating to agile lowers costs and increases revenues, which should also lead to an increase in stock value or, if you're a small company, survival.

■ All managers dislike dealing with people issues. The agile work environment is more satisfying, and the executives will find themselves dealing with fewer employee issues. They will also be pleased to see employee retention rates increase.

■ As we mentioned earlier, customer satisfaction increases with agile. A happier customer leads to more pleasant discussions with the executives.

Many companies try to "shotgun" agile into their organization. They think, "Let's get through the migration pain quickly and start obtaining the benefits as soon as possible." We've seen a few cases where this approach makes sense: for example, a project team that has become so dysfunctional that they're delivering practically no functionality or business value. This approach also works well for a start-up company that hasn't yet established its development process. But for most companies, you should allow time for the process to "bake."

This is why we suggest an iterative approach for bringing agile into an organization. An iterative approach allows you to see how well your employees are adapting to the change. It also lets you learn what works and what doesn't in your environment. In effect, it allows you to reach the right level of agility for your organization.

Part of your iterative approach will include a process for maintaining the methodology. We suggest establishing a core team to support this maintenance. A core team is composed of employees from all aspects of the development process. They

play a huge part in establishing your custom methodology and then settle into a maintenance mode with the goal of constantly adapting to your environment. Next, you need to choose the best way to iteratively create a methodology at your company. Should you select a packaged method, such as Scrum or XP? Or should you create a custom or hybrid process?

In order to be successful, you should customize your agile process. For many years, consultants and others have said that you must embrace agile completely or not at all. In 2006, we witnessed a shift in this attitude. Highly respected folks such as Kent Beck (the founder of XP) and Steve McConnell (the writer of *Code Complete*) now endorse customization. Kent Beck noted the following in an interview with InfoQ (InfoQ.com is an independent online community focused on change and innovation in enterprise software development) in 2006:

Failure at an organizational level seems to come from the inability to customize processes and make them their own. Trying to apply someone else's template to your organization directly doesn't work well. It leaves out too many important details of the previous successes and ignores your company's specific situation. Rubber-stamping agile processes isn't agile. The value of having a principle-based process is that you can apply the principles for an individualized process for your situation and, as an extra bonus, one that has been designed to adapt from your learning as you adopt changes into your organization. It's always "custom." Kent's quote is comforting to us because it supports our personal experiences. Custom means picking and choosing the agile practices that best support your environment. Custom means you shouldn't use a pure packaged methodology off the shelf, such as Scrum or XP. You can start with one of these methods as a basis for your process, but you should modify it to obtain the best results for your company.

To be specific, here are the steps we'll walk:

1 Assess your organization to determine where you should begin adding agility.

2 Obtain executive support for the move to a more agile process. You can use the readiness assessment to quantify the value of bringing in agile and identify the risks you must manage during migration.

3 Get the development team involved in the migration process to ensure buy-in. You do this by establishing a core team.

4 Identify a coach or consultant to help you with your migration. They will train the core team on agile and help you with other adoption aspects.

5 Develop a clear understanding of your current processes by documenting them.

6 Review your current process, and look for areas that can be shifted to more agile methods. Focus on areas with the most potential for improvement and the most value to the customer and your organization. The readiness assessment will also help with this task.

7 Outline a custom process based on the findings in step 6.

8 Try the new process on a pilot project.

9 Review the findings after the pilot, make changes, and continue to scale out your new methodology.

A large project requires training many people—and possible several third parties—on the new methodology. This will delay your ability to gather feedback about the new process and adjust it, which is the ultimate objective of the pilot.

Let us give you an example of a project that we consider large. We were upgrading a company's intranet platform, and the project was scheduled to run for 8 months.

The project went through several gateways to obtain funding. It required high involvement from the software provider, and consultants were needed to train the team on the new software. Because the application affected the enterprise, we engaged several internal teams to help us with training, communication, and security. A project of this length and scope doesn't allow time for testing a new process.

You also don't want to work on a project that is too small. Such a project limits the areas and phases in which you can test the new process.

The question you need to ask is, "What do you do during a project that can be completed within 8 weeks?" You need to identify a group of features to serve as a mini-project within the larger project. For example, if you're building a website similar to eBay, the project may take 6 months to a year. To test your new

methodology, you can grab a few features and test the new process on them. For example, you could do a mini-pilot project around seller feedback and its related features.

If you go the subset route you may have limitations on how far you can test the new process. Your subset of features may need to rejoin the other features and go through the legacy testing and deployment processes. If you experience this issue, you can try to pull the features completely out of the project in subsequent tests, running them all the way through to deployment.

pilot project needs to have some level of urgency to test the new process under pressure; but if the project is deemed critical, failure isn't a viable option. You may panic, abort, and revert to methods you're more comfortable with to complete the project. As we mentioned in section 9.1, the pilot project is a marketing effort as well as a test of the new methodology—you don't want to send a message to your company that the pilot was aborted.

A project is usually critical if your company or the customer can't survive without it. Here are some example projects that a business would consider critical:

■ A project to ensure a revenue stream

■ A project that supports meeting a regulatory or compliance deadline ■ A project with expiring funds (budget tied to a time frame)

■ A project that delivers functionality that is a foundation for the organization (such as service-oriented architecture [SOA]) From a customer perspective, these projects could be considered critical:

■ A project tied to a fixed bid

■ A project that the customer depends on for a marketing campaign

■ A project that relates to a regulatory or compliance issue on the customer end

Your objective should be to find a medium-priority project. Such a project allows some flexibility as you feel out your new process and also provides a level of urgency. You're moving to agile to better support urgent projects, so you need to simulate this with your pilot.

Here are some examples of medium-priority projects:

- Adding the ability to book hotels on an existing travel site
- Delivering a maintenance release onto an existing platform
- Adding customizable stock quotes to a portal page
- Modifying your HR application to let employees view their vacation balance
- Adding advanced search capabilities to your existing simple search

Now that you understand the pilot's desired size and priority, let's look at the breadth of the project.

Your pilot project needs to touch all major areas related to projects at your company. You don't need to go deep, but you should go wide. It may be difficult to select a test project that utilizes all possible processes and departments, but you should select one that hits the majority of them. Although your pilot will go wide, you don't want to test outside of your company.

The reason is that you'll be busy watching the process within your company. Adding a third party into the mix may diminish your ability to record feedback and learn from the pilot. You can involve third parties in subsequent projects, when you have more bandwidth for their feedback.

It's tempting to try to test agile area by area, but doing so usually leads to poor results. Your new agile process will be designed to have practices integrate with each other. This integration is where a good portion of the value comes from. A segmented test may mask these benefits and also minimize your ability to identify issues with the process. Here's an example. A few years ago, a prominent company implemented an agile development process. The company had created a core team to create the custom methodology. The core team was composed of employees from various departments such as Program Management, Development, QA, User Interface, Implementation, and Analytics. On occasion, core-team members demonstrated a proposed process on a live project. In one such instance, a program manager decided to test the agile feature-card process on a live project. The project was already following a traditional lifecycle.

The value of the feature-card meeting is that it lets you gather enough information to plan the project: it's a precursor to gathering detailed requirements. The

project that was already in flight had complete functional specifications and a robust project plan. To perform the feature-card meeting, the program manager had to pretend that all the information in the functional specifications didn't exist. She also had to determine what to do with the output from the feature-card meeting.

In an agile process, the output feeds an iteration or sprint plan. The project in progress already had a detailed development plan, so no value was gained from the feature-card exercise. This demo had a *negative* effect on the company's agile deployment. Because the feature-card meeting was used at the wrong time, it added no value. The employees who tested the process quickly spread their experience throughout the company, and other employees who were already against an agile methodology now had the ammo they needed to try to stop it: they had proof that agile didn't add any value to the development process.

In summary, you can use agile and traditional processes together, but you have to design interface points so the process flows logically.

Every company has a different way of populating their request/potential projects backlog (a.k.a. *project backlog*). Sometimes backlog evaluation is driven by an executive review process, sometimes it's driven by product marketing, and sometimes it's driven by a customer.

PART 2. ANALYSIS OF THE TRANSITION OF THE TRADING ENTERPRISE MANAGEMENT TO THE FLEXIBLE MANAGEMENT METHODOLOGIES

2.1. Characteristics of trading enterprise management processes of Smart Trading Group

Scrum doesn't prescriptively answer Smart Trading Group process questions; instead, it empowers teams to ask and answer their own great questions. Scrum doesn't give individuals a cookbook solution to all of their organizational maladies; instead, Scrum makes visible the dysfunctions and waste that prevent organizations from reaching their true potential. These realizations can be painful for many organizations. However, if they move past the initial discomfort and work to solve the problems Scrum unearths, organizations can take great strides in terms of both their software development process and products and their levels of employee and customer satisfaction. We will begin with a description of the entire Scrum framework, including its roles, activities, artifacts, and rules [54].

Scrum development efforts consist of one or more Scrum teams, each made up of three Scrum roles: product owner, ScrumMaster, and the development team (see Figure 2.1). There can be other roles when using Scrum, but the Scrum framework requires only the three listed here.

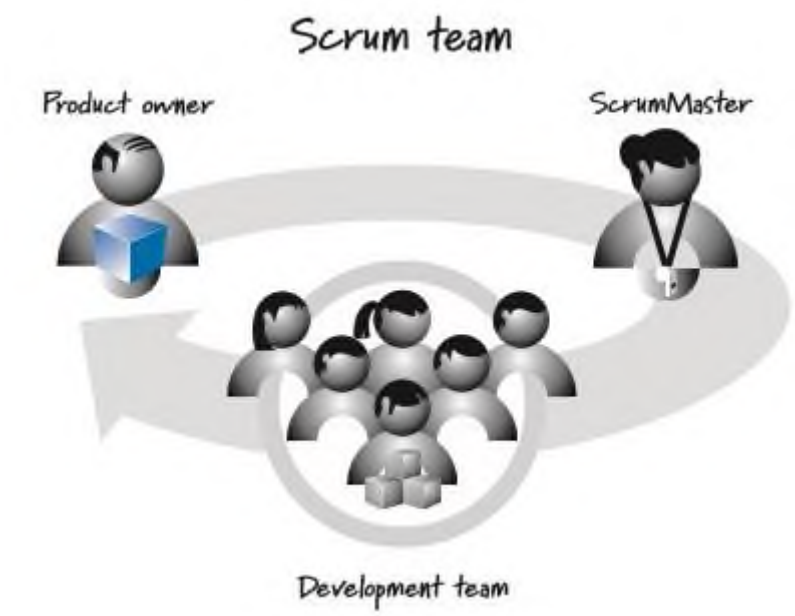


Figure 2.1. Scrum roles of Smart Trading Group **[74]**

The product owner is responsible for what will be developed and in what order. The ScrumMaster is responsible for guiding the team in creating and following its own process based on the broader Scrum framework. The development team is responsible for determining how to deliver what the product owner has asked for. If you are a manager, don't be concerned that "manager" doesn't appear as a role in Figure 2.1; managers still have an important role in organizations that use Scrum Smart Trading Group. The Scrum framework defines just the roles that are specific to Scrum, not all of the roles that can and should exist within an organization that uses Scrum [41].

Product Owner The product owner is the empowered central point of product leadership. He1 is the single authority responsible for deciding which features and functionality to build and the order in which to build them. The product owner maintains and communicates to all other participants a clear vision of what the Scrum team is trying to achieve. As such, the product owner is responsible for the overall success of the solution being developed or maintained. It doesn't matter if the focus is on an external product or an internal application; the product owner still has the obligation to make sure that the most valuable work possible, which can include technically focused work, is always performed. To ensure that the team rapidly builds what the product owner wants, the product owner actively collaborates with the ScrumMaster and development team and must be available to answer questions soon after they are posed.

The ScrumMaster helps everyone involved understand and embrace the Scrum values, principles, and practices. She acts as a coach, providing process leadership and helping the Scrum team and the rest of the organization develop their own highperformance, organization-specific Scrum approach. At the same time, the ScrumMaster helps the organization through the challenging change management process that can occur during a Scrum adoption. As a facilitator, the ScrumMaster helps the team resolve issues and make improvements to its use of Scrum. She is also responsible for protecting the team from outside interference and takes a leadership

role in removing impediments that inhibit team productivity (when the individuals themselves cannot reasonably resolve them). The ScrumMaster has no authority to exert control over the team, so this role is not the same as the traditional role of project manager or development manager. The ScrumMaster functions as a leader, not a manager [53].

Development Team Traditional software development approaches discuss various job types, such as architect, programmer, tester, database administrator, UI designer, and so on. Scrum defines the role of a development team, which is simply a diverse, cross-functional collection of these types of people who are responsible for designing, building, and testing the desired product. The development team self-organizes to determine the best way to accomplish the goal set out by the product owner. The development team is typically five to nine people in size; its members must collectively have all of the skills needed to produce goodquality, working software. Of course, Scrum can be used on development efforts that require much larger teams. However, rather than having one Scrum team with 35 people, there would more likely be four or more Scrum teams, each with a development team of nine or fewer people.

An alternative approach would be for the product owner and team to select all of the target product backlog items at one time. The development team alone does the task breakdowns to confirm that it really can deliver all of the selected product backlog items.

Sprint Execution in **Smart Trading Group.** Once the Scrum team finishes sprint planning and agrees on the content of the next sprint, the development team, guided by the ScrumMaster's coaching, performs all of the task-level work necessary to get the features done (see Figure 2.10), where "done" means there is a high degree of confidence that all of the work necessary for producing good-quality features has been completed. Exactly what tasks the team performs depends of course on the nature of the work (for example, are we building software and what type of software, or are we building hardware, or is this marketing work?). Nobody tells the development team in what order or how to do the task-level work in the sprint backlog. Instead, team

members define their own task-level work and then self-organize in any manner they feel is best for achieving the sprint goal.

Daily Scrum Smart Trading Group**.** Each day of the sprint, ideally at the same time, the development team members hold a timeboxed (15 minutes or less) daily scrum (see Figure 2.2). This inspect-andadapt activity is sometimes referred to as the daily stand-up because of the common practice of everyone standing up during the meeting to help promote brevity.

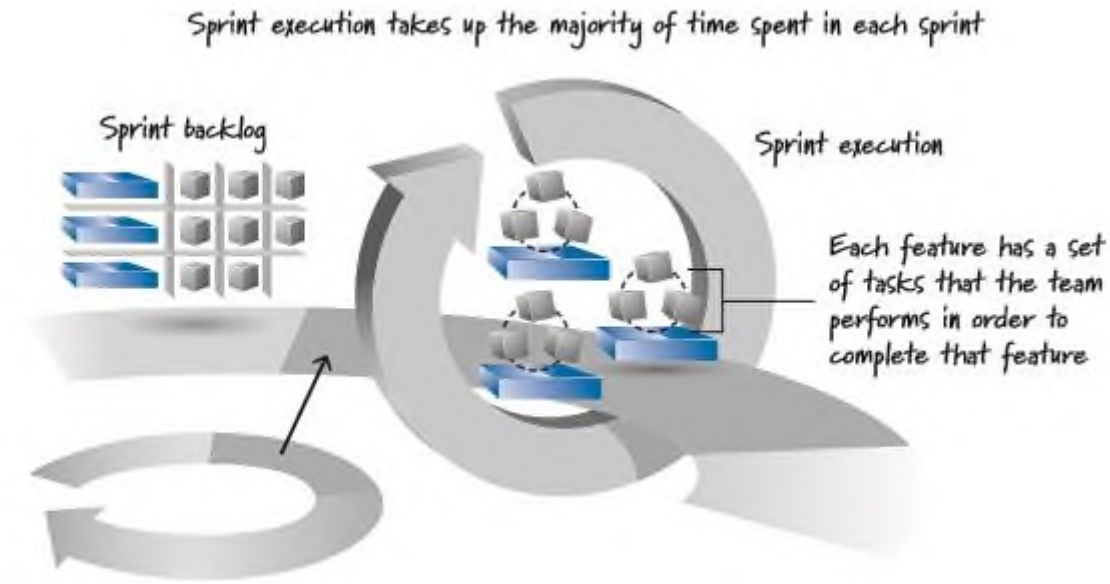


Figure 2.2 Sprint execution Smart Trading Group **[41]**

A common approach to performing the daily scrum has the ScrumMaster **Smart Trading Group** facilitating and each team member taking turns answering three questions for the benefit of the other team members:

What did manager accomplish since the last daily scrum? What does manager plan to work on by the next daily scrum? What are the obstacles or impediments that are preventing me from making progress? [38]

By answering these questions, everyone understands the big picture of what is occurring, how they are progressing toward the sprint goal, any modifications they want to make to their plans for the upcoming day’s work, and what issues need to be

addressed. The daily scrum is essential for helping the development team manage the fast, flexible flow of work within a sprint. The daily scrum is not a problem-solving activity. Rather, many teams decide to talk about problems after the daily scrum and do so with a small group of interested people. The daily scrum also is not a traditional status meeting, especially the kind historically called by project managers so that they can get an update on the project's status. A daily scrum, however, can be useful to communicate the status of sprint backlog items among the development team members. Mainly, the daily scrum is an inspection, synchronization, and adaptive daily planning activity that helps a selforganizing team do its job better.

Although Smart Trading Group use has fallen out of favor, Scrum has used the terms "pigs" and "chickens" to distinguish who should participate during the daily scrum versus who simply observes. The farm animals were borrowed from an old joke (which has several variants): "In a ham-and-eggs breakfast, the chicken is involved, but the pig is committed." Obviously the intent of using these terms in Scrum is to distinguish between those who are involved (the chickens) and those who are committed to meeting the sprint goal (the pigs). At the daily scrum, only the pigs should talk; the chickens, if any, should attend as observers. I have found it most useful to consider everyone on the Scrum team a pig and anyone who isn't, a chicken. Not everyone agrees. For example, the product owner is not required to be at the daily scrum, so some consider him to be a chicken (the logic being, how can you be "committed" if you aren't required to attend?). This seems wrong to me, because I can't imagine how the product owner, as a member of the Scrum team, is any less committed to the outcome of a sprint than the development team. The metaphor of pigs and chickens breaks down if you try to apply it within a Scrum team.

Done In Scrum, we refer to the sprint results as a potentially shippable product increment (see Figure 2.3), meaning that whatever the Scrum team agreed to do is really done according to its agreed-upon definition of done. This definition specifies the degree of confidence that the work completed is of good quality and is potentially shippable. For example, when developing software, a bare-minimum definition of done should yield a complete slice of product functionality that is designed, built,

integrated, tested, and documented. An aggressive definition of done enables the business to decide each sprint if it wants to ship (or deploy or release) what got built to internal or external customers. To be clear, “potentially shippable” does not mean that what got built must actually be shipped. Shipping is a business decision, which is frequently influenced by things such as “Do we have enough features or enough of a customer workflow to justify a customer deployment?” or “Can our customers absorb another change given that we just gave them a release two weeks ago?” Potentially shippable is better thought of as a state of confidence that what got built in the sprint is actually done, meaning that there isn’t materially important undone work (such as important testing or integration and so on) that needs to be completed before we can ship the results from the sprint, if shipping is our business desire [45]. As a practical matter, over time some teams may vary the definition of done. For example, in the early stages of game development, having features that are potentially shippable might not be economically feasible or desirable (given the exploratory nature of early game development). In these situations, an appropriate definition of done might be a slice of product functionality that is sufficiently functional and usable to generate feedback that enables the team to decide what work should be done next or how to do it.

Figure 2.3. Work automation **Smart Trading Group [32]**

Artificial intelligence, automation, industrial robotics and the similar emerging technologies are growing at an alarming speed, but there has been inconsiderable concentration to their effect on employment and other organizational practices. Despite the fact that these technologies can upgrade the speed, cost, and quality of the products and services, the fact is they may also expel a huge numbers of employees. A recent user survey revealed that around 85% of respondents believe that AI and automation will transform their job in the coming three years. 83% of those inspected said they were excited, but 84% also stated feeling suspicious.The purpose of this post is to explore today's fast evolving world of AI and automation and its impact on the field of project management.What is Automation?From transportation to manufacturing, and utilities to defense, automation can be found in almost every industry. With innumerable projects being labor-intensive, the production of automated machinery has revamped performance and also augmented better quality control [39].

FMS—Flexible Manufacturing Systems integrate numerous industrial automation tools like robotics, control systems and the like to develop one practical system CAM—CAM makes use of computer software to run machinery. This type is

usually used to mechanize the manufacturing procedures and scheduling tasksNumerical Control (NC)—It is basically a type of programmable automation, in which the entire system is managed by the numbers, signs or symbols. These tools are primarily responsible for carrying out recurring tasks such as 3D printing, drills, etc.I ndustrial Robotics—Industrial robots can be mechanized as well as programmed in two or more axesUse of Automation in Real Life and at WorkThe automation of work was an outcome of economic theories that surrounds around output optimization.

2.2. Experience defining flexible management methodologies for enterprises

The goal of comparing agile principles Smart Trading Group with traditional development principles Smart Trading Group is not to make the case that plan-driven, sequential development is bad and that Scrum is good. Both are tools in the professional developer's toolkit; there is no such thing as a bad tool, rather just inappropriate times to use that tool. Scrum and traditional, plan-driven, sequential development are appropriate to use on different classes of problems. In making the comparison between the two approaches, we are using the pure or "textbook" description of plan-driven, sequential development. By taking this perspective when describing traditional development, I am better able to draw out the distinctions and more clearly illustrate the principles that underlie Scrum-based development. One pure form of traditional, plan-driven development frequently goes by the term waterfall (see Figure 2.4).

However, that is just one example of a broader class of plan-driven processes (also known as traditional, sequential, anticipatory, predictive, or prescriptive development processes). Plan-driven processes are so named because they attempt to plan for and anticipate up front all of the features a user might want in the end product, and to determine how best to build those features. The idea here is that the better the planning, the better the understanding, and therefore the better the execution. Plan-driven processes are often called sequential processes because practitioners perform, in

sequence, a complete requirements analysis followed by a complete design followed in turn by coding/building and then testing [24].

Plan-driven development works well if you are applying it to problems that are well defined, predictable, and unlikely to undergo any significant change. The problem is that most product development efforts are anything but predictable, especially at the beginning. So, while a plan-driven process gives the impression of an orderly, accountable, and measurable approach, that impression can lead to a false sense of security. After all, developing a product rarely goes as planned. For many, a plan-driven, sequential process just makes sense, understand it, design it, code it, test it, and deploy it, all according to a well-defined, prescribed plan. There is a belief that it should work. If applying a plan-driven approach doesn't work, the prevailing attitude is that we must have done something wrong. Even if a plan-driven process repeatedly produces disappointing results, many organizations continue to apply the same approach, sure that if they just do it better, their results will improve.

The problem, however, is not with the execution. It's that plan-driven approaches are based on a set of beliefs that do not match the uncertainty inherent in most product development efforts. Scrum, on the other hand, is based on a different set of beliefs—ones that do map well to problems with enough uncertainty to make high levels of predictability difficult. The principles are drawn from a number of sources, including the Agile Manifesto [40], lean product development (Reinertsten 2009b; Poppendieck and Poppendieck 2003), and "The Scrum Guide" [60].

Variability and Uncertainty Scrum leverages the variability and uncertainty in product development to create innovative solutions. I describe four principles related to this topic:

Embrace helpful variability. Employ iterative and incremental development. Leverage variability through inspection, adaptation, and transparency. Reduce all forms of uncertainty simultaneously.

Embrace Helpful Variability Plan-driven processes treat product development like manufacturing—they shun variability and encourage conformance to a defined process. The problem is that product development is not at all like product

manufacturing. In manufacturing our goal is to take a fixed set of requirements and follow a sequential set of well-understood steps to manufacture a finished product that is the same (within a defined variance range) every time.

In product development the goal is to create the unique single instance of the product, not to manufacture the product. This single instance is analogous to a unique recipe. We don't want to create the same recipe twice; if we do, we have wasted our money. Instead, we want to create a unique recipe for a new product. Some amount of variability is necessary to produce a different product each time. In fact, every feature we build within a product is different from every other feature within that product, so we need variability even at this level. Only once we have the recipe do we manufacture the product—in the case of software products, as easily as copying bits. That being said, some manufacturing concepts do apply to product development and can and should be leveraged. For example, recognizing and managing inventory (or work in process), which is essential to manufacturing, is also essential in product development. By the very nature of the work involved, however, product development and product manufacturing are not at all the same thing and as such require very different processes [56].

Employ Iterative and Incremental Development Plan-driven, sequential development assumes that we will get things right up front and that most or all of the product pieces will come together late in the effort. Scrum, on the other hand, is based on iterative and incremental development. Although these two terms are frequently used as if they were a single concept, iterative development is actually distinct from incremental development. Iterative development acknowledges that we will probably get things wrong before we get them right and that we will do things poorly before we do them well [23]. As such, iterative development is a planned rework strategy. We use multiple passes to improve what we are building so that we can converge on a good solution. For example, we might start by creating a prototype to acquire important knowledge about a poorly known piece of the product. Then we might create a revised version that is somewhat better, which might in turn be followed by a pretty good version. Iterative development is an excellent way to improve the product

as it is being developed. The biggest downside to iterative development is that in the presence of uncertainty it can be difficult up front to determine (plan) how many improvement passes will be necessary. Incremental development is based on the age-old principle of "Build some of it before you build all of it." We avoid having one large, big-bang-style event at the end of development where all the pieces come together and the entire product is delivered. Instead, we break the product into smaller pieces so that we can build some of it, learn how each piece is to survive in the environment in which it must exist, adapt based on what we learn, and then build more of it.

The biggest drawback to incremental development is that by building in pieces, we risk missing the big picture (we see the trees but not the forest). Scrum leverages the benefits of both iterative and incremental development, while negating the disadvantages of using them individually. Scrum does this by using both ideas in an adaptive series of timeboxed iterations called sprints (see Figure 2.4).

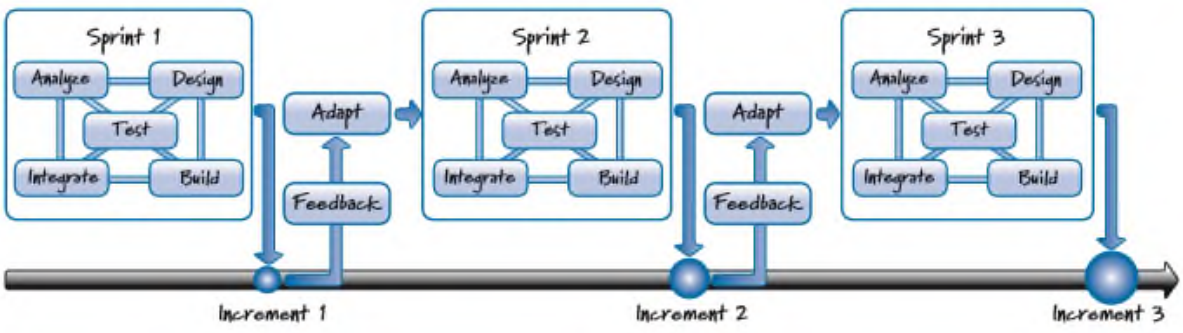


Figure 2.4 Scrum uses iterative and incremental development in Smart Trading Group **[81]**

During each sprint we perform all of the activities necessary to create a working product increment (some of the product, not all of it). This is illustrated in Figure 2.4 by showing that some analysis, design, build, integration, and test work is completed in each sprint. This all-at-once approach has the benefit of quickly validating the assumptions that are made when developing product features. For example, we make some design decisions, create some code based on those decisions, and then test the

design and code—all in the same sprint. By doing all of the related work within one sprint, we are able to quickly rework features, thus achieving the benefits of iterative development, without having to specifically plan for additional iterations.

A misuse of the sprint concept is to focus each sprint on just one type of work—for example, sprint 1 (analysis), sprint 2 (design), sprint 3 (coding), and sprint 4 (testing). Such an approach attempts to overlay Scrum with a waterfall-style work breakdown structure. I often refer to this misguided approach as WaterScrum, and I have heard others refer to it as Scrummerfall [46].

In Scrum, Smart Trading Group doesn't work on a phase at a time; we work on a feature at a time. So, by the end of a sprint we have created a valuable product increment (some but not all of the product features). That increment includes or is integrated and tested with any previously developed features; otherwise, it is not considered done. For example, increment 2 in Figure 2.4 includes the features of increment 1. At the end of the sprint, we can get feedback on the newly completed features within the context of already completed features. This helps us view the product from more of a big-picture perspective than we might otherwise have. We receive feedback on the sprint results, which allows us to adapt. We can choose different features to work on in the next sprint or alter the process we will use to build the next set of features. In some cases, we might learn that the increment, though it technically fits the bill, isn't as good as it could be. When that happens, we can schedule rework for a future sprint as part of our commitment to iterative development and continuous improvement.

This helps overcome the issue of not knowing up front exactly how many improvement passes we will need. Scrum does not require that we predetermine a set number of iterations. The continuous stream of feedback will guide us to do the appropriate and economically sensible number of iterations while developing the product incrementally.

Leverage Variability through Inspection, Adaptation, and Transparency Plan-driven processes and Scrum are fundamentally different along several dimensions (see Table 3.1, based on dimensions suggested by Reinertsen 2009a). A plan-driven,

sequential development process assumes little or no output variability. It follows a well-defined set of steps and uses only small amounts of feedback late in the process. In contrast, Scrum embraces the fact that in product development, some level of variability is required in order to build something new. Scrum also assumes that the process necessary to create the product is complex and therefore would defy a complete up-front definition. Furthermore, it generates early and frequent feedback to ensure that the right product is built and that the product is built right. At the heart of Scrum are the principles of inspection, adaptation, and transparency (referred to collectively by Schwaber and Beedle 2001 as empirical process control). In Scrum, we inspect and adapt not only what we are building but also how we are building it (see Figure 3.5). To do this well, we rely on transparency: all of the information that is important to producing a product must be available to the people involved in creating the product. Transparency makes inspection possible, which is needed for adaptation. Transparency also allows everyone concerned to observe and understand what is happening. It leads to more communication and it establishes trust (both in the process and among team members).

Table 2.1

Comparison of Plan-Driven and Scrum Processes [31]

Dimension	**Plan-Driven**	**Scrum**
Degree of process definition	Well-defined set of sequential steps	Complex process that would defy a complete up-front definition
Randomness of output	Little or no output variability	Expect variability because we are not trying to build the same thing over and over
Amount of feedback used	Little and late	Frequent and early

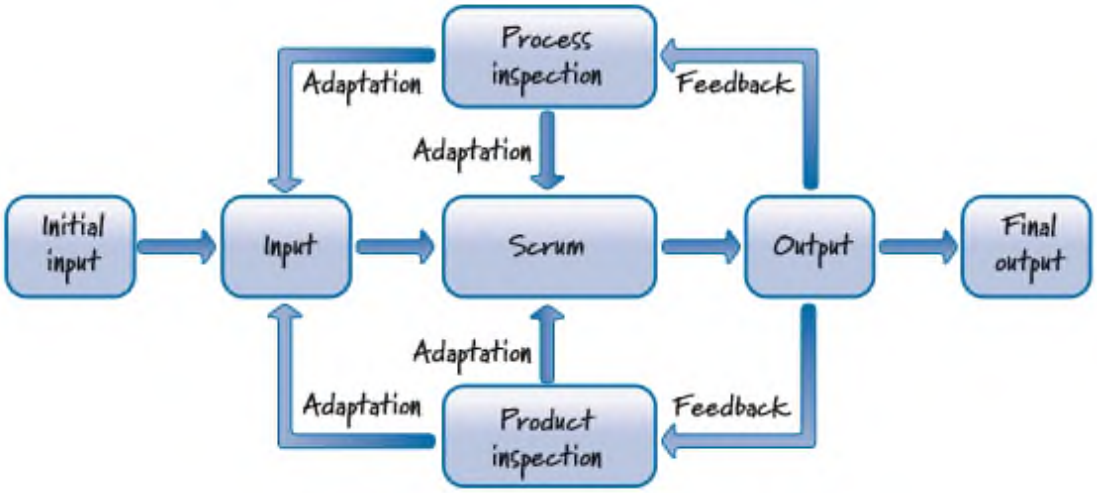


Figure 2.5 Scrum process model in Smart Trading Group **[14]**

Developing new products is a complex endeavor with a high degree of uncertainty. That uncertainty can be divided into two broad categories (Laufer 1996):

End uncertainty (what uncertainty)—uncertainty surrounding the features of the final product Means uncertainty (how uncertainty)—uncertainty surrounding the process and technologies used to develop a product

In particular environments or with particular products there might also be customer uncertainty (who uncertainty). For example, start-up organizations (including large organizations that focus on novel products) may only have assumptions as to who the actual customers of their products will be. This uncertainty must be addressed or they might build brilliant products for the wrong markets. Traditional, sequential development processes focus first on eliminating all end uncertainty by fully defining up front what is to be built, and only then addressing means uncertainty. This simplistic, linear approach to uncertainty reduction is ill suited to the complex domain of product development, where our actions and the environment in which we operate mutually constrain one another. For example:

Smart Trading Group decides to build a feature (our action). We then show that feature to a customer, who, once he sees it, changes his mind about what he really wants, or realizes that he did not adequately convey the details of the feature (our action elicits a response from the environment).

We make design changes based on the feedback (the environment's reaction influences us to take another unforeseen action).

In Scrum, Smart Trading Group does not constrain ourselves by fully addressing one type of uncertainty before we address the next type. Instead, we take a more holistic approach and focus on simultaneously reducing all uncertainties (end, means, customer, and so on). Of course, at any point in time we might focus more on one type of uncertainty than another. Simultaneously addressing multiple types of uncertainty is facilitated by iterative and incremental development and guided by constant inspection, adaptation, and transparency. Such an approach allows us to opportunistically probe and explore our environment to identify and learn about the unknown unknowns (the things that we don't yet know that we don't know) as they emerge.

Prediction and Adaptation When using Scrum, we are constantly balancing the desire for prediction with the need for adaptation [49].

Keep options open. Accept that you can't get it right up front. Favor an adaptive, exploratory approach. Embrace change in an economically sensible way. Balance predictive up-front work with adaptive just-in-time work.

Scrum contends that we should never make a premature decision just because a generic process would dictate that now is the appointed time to make one. Instead, when using Scrum, we favor a strategy of keeping our options open. Often this principle is referred to as the last responsible moment (LRM) (Poppendieck and Poppendieck 2003), meaning that we delay commitment and do not make important and irreversible decisions until the last responsible moment. And when is that? When the cost of not making a decision becomes greater than the cost of making a decision (see Figure 3.6). At that moment, we make the decision. To appreciate this principle, consider this. On the first day of a product development effort we have the least information about what we are doing.

On each subsequent day of the development effort, we learn a little more. Why, then, would we ever choose to make all of the most critical, and perhaps irreversible, decisions on the first day or very early on? Most of us would prefer to wait until we

have more information so that we can make a more informed decision. When dealing with important or irreversible decisions, if we decide too early and are wrong, we will be on the exponential part of the cost-of-deciding curve in Figure 2.6.

As we acquire a better understanding regarding the decision, the cost of deciding declines (the likelihood of making a bad decision declines because of increasing market or technical certainty). That's why we should wait until we have better information before committing to a decision.

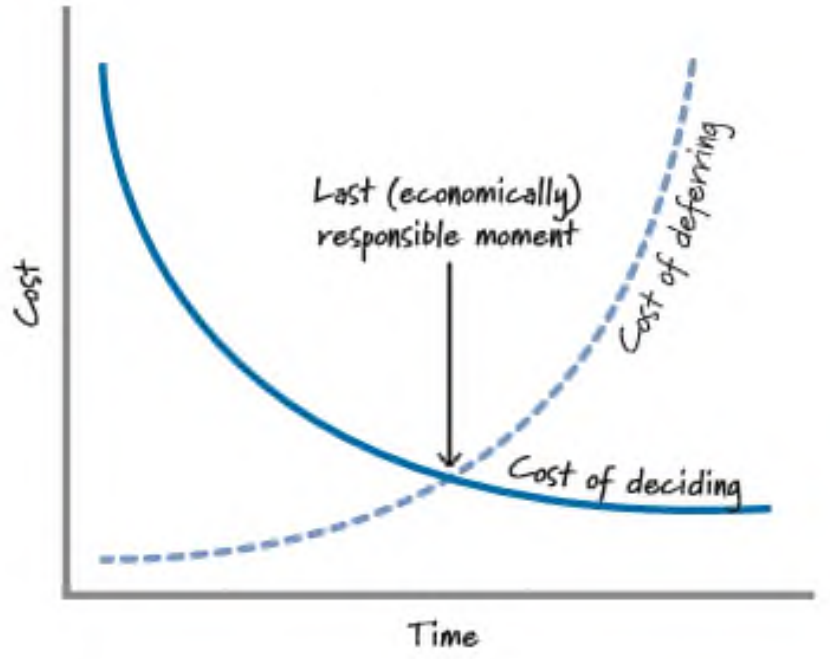


Figure 2.6 Make decisions at the last responsible moment

2.3. Determining the impact of the Smart Trading Group transition on flexible management methodologies

Smart Trading Group is a media company that engages in television broadcasting andinteractive media operations. The interactive media operations consist of three product websites. Our case study is tied to the interactive Smart Trading Group and its following three sites:

■ The first website focuses on delivering the news. If the television station is covering a breaking story, the news site needs to have the story online during the TV

coverage or a few minutes afterward. The news website also provides enriched coverage about news articles, such as blogs, opinion surveys, and deeper analysis.

■ The second website is focused on classified advertising for the local metropolitan area. The classifieds are for real estate, autos, and merchandise. The case study's pilot project will deliver an application for the classifieds site.

■ The third website is responsible for travel and outdoors. This site contains content related to tourism, hiking, getaways, and lodging.

All three websites sell online advertising space to national and local businesses. The three sites are supported by a group of 20 people. The skill sets in this group include product management, development, design, database analysis, business analysis, architecture, implementation, support, testing, and project management. Several team members wear various hats depending on the state of the project. For example, developers may do their own DBA work, or a product manager may end up doing the requirements documentation. The development group is also the maintenance and support group for the production environment [7].

The Smart Trading Group websites have always been a secondary priority for the company. They don't make much money and exist only as a supplementary presence to the television station. Until about a year ago, working in the web group was laid-back and easy; there was rarely any pressure, and projects were completed on a loose schedule. All of that has changed now. With the popularity of online advertising on the rise, Smart Trading Group's web division found itself overwhelmed with advertising requests and a significant increase in site traffic. In addition, advertisers were asking Smart Trading Groupto publish more television content on the websites to attract a younger crowd. Suddenly, web projects had urgency, and revenue was on the line. The heat was on the three product teams. The lack of reliable schedules and the teams' frequent need to push out promise dates were motivating advertisers and readers to go to competitive websites. Projects needed to be delivered on time to retain the customers.

Smart Trading Group started down its road to more agility when the company's project manager began investigating ways to deliver products sooner. PM conducted

research and discussed her issue with a number of friends and colleagues. One friend Wendy spoke with, Jim Moore, happened to be an agile coach. Jim told Wendy about several companies he had helped move to agile and the benefits these companies had achieved. After Wendy was convinced, she set about selling Smart Trading Group's CIO on the idea that Smart Trading Group's development team needed to give agile a try. Wendy will get approval from the executive team to pilot agile.

Smart Trading Group will create two teams as the company completes its pilot project and evaluates how agile it can become. First, the company will create a *core team*. The core team will be trained and mentored by the agile coach, Jim Moore. The core team will be in charge of reviewing the existing development process at Smart Trading GroupMedia to see where agile practices can be injected. This team will consist of actual project team members [5].

After the core team outlines a new process to test, a *pilot project team* will be selected to actually complete the pilot project. The pilot project team will include a few core team members, but most of the pilot team members selected will be getting their first look at the new process. The pilot team will receive training on agile principles and basic practices, and then they will perform the pilot and provide feedback to the core team. The core team will use the feedback to refine the process and then continue to scale the new process throughout the company.

Smart Trading Group's development team has various experiences, backgrounds, and opinions about how software development should be completed (see figure 2.7). Some team members have experience in formal, plan-driven environments; many team members have worked in environments where the development process was homegrown; and a few team members have experience in agile environments. As Smart Trading Group pursues its new process, some team members will be energetic, some will be neutral, and some will be skeptics. In our experience, a variety of responses is common in most companies. Smart Trading Group will use all three perspectives to help roll out the best possible new process.

Some of the energetic team members will help in documenting the existing process. The team members on the fence will be part of the core team and the pilot

team. Skeptics will also be included on both teams, and their input will be welcome as the new process is created and critiqued.

When people think of becoming agile, they often envision the practices and not the goals of an agile process.

A traditional process has the customer involved mainly at the beginning and the end of the project. In agile, you seek customer feedback and input throughout the project. The customer or product owner is involved in planning, tradeoff decisions, prioritization, and demonstrations. Increased customer involvement leads to several benefits such as quicker feedback, accurate delivery, increased customer satisfaction, and rapid decisions. A great indirect benefit of customer involvement is the customer's newfound appreciation for the work needed to deliver on requests [11].

Agile processes improve prioritization and deliver higher-value features first. This is accomplished by creating feature cards or user stories and evaluating features before requirements are detailed. You'll evaluate features for their customer value, level of risk, frequency of use, and dependencies. This allows you to do the following:

- Estimate work and evaluate risks early in the process.
- Prioritize features in terms of customer value early in the process.
- Deliver features in usable subsets.

In effect, the agile prioritization process lets your team run leaner and create deep requirements only for work that passes the prioritization test.

The majority of people on an agile project team are involved in planning, estimating, and sequencing. The team is also involved in adapting to discoveries between iterations. Over time, the team begins suggesting features for the product or platform. Increasing team involvement ensures that everyone understands the value of the project before work begins and also increases team satisfaction.

A more agile and iterative methodology provides an opportunity to reassess and redirect the project while it's in motion. You perform development in iterations and offer demonstrations at the end of each. The customer has an opportunity to request changes based on the demonstrations, even though this may affect other features or potentially the project timeline. Team members learn to expect and embrace change

any companies try to plan all of a project's details at the start. The planning may be at a detailed level even though the amount of uncertainty at this point is extremely high. An agile team performs a level of planning that correlates to the current level of uncertainty in the project.

As you learn more about desired features you'll do more detailed planning, but you won't waste time trying to guess intricate details early in the project. Figure 3.1 illustrates this point.

A secondary definition of *agile* could be *continuous risk management*. The processes are all intended to make the team alert and responsive to new information and changes as the project progresses. The following are a few examples of how agile manages risk:

■ Features are evaluated for requirements uncertainty and technical uncertainty. These attributes help determine whether a feature goes into an iteration and what iteration it should go into, to mitigate risk. For example, a feature with high business value and high technical risk, such as an interface, would go into an early iteration to allow more time for uncertainty. On the other hand, a feature with low business value and high technical uncertainty might be moved to the last iteration or removed from the project all together.

■ Risk is managed via demonstrations throughout the project. The customer gets a feel for how requirements are translated into an application before the project is complete. This provides a window for adapting and hitting the final target.

■ Risk is managed on a daily basis by building and integrating the latest code. This process allows the team and the customer to validate the status of the latest build.

■ Deployment risk is also managed by gathering maintenance and deployment concerns as early as possible. This starts early in the planning phase and continues throughout development.

■ Risk is managed via team review of potential features. During the feature-card exercise, representatives from all areas can raise risks and concerns with proposed features. These concerns are noted with the feature information and sometimes can lead to a feature not being pursued.

The agile process described in this work approaches the issue differently. We suggest a standardized methodology, but the required processes are minimal and are of value to every project. Your team chooses the majority of the processes to use at the start of the project. The team also revisits their process and documentation options as the project proceeds, to see if they need to add or remove a process or document. To illustrate this idea, let's look at an example from Smart Trading GroupMedia *after* the company has outlined a new, more agile process (see table 2.2). Smart Trading GroupMedia has projects that last from 1 week to 6 months. The company doesn't require the teams for one-week projects to create iteration plans or to do a cost-benefit analysis every time.

Table 2.2

Required and optional processes and documentation [61]

Required for all projects	Optional processes and documents
Project worksheet	Elevator statement
Operational worksheet	Documented answers to feasibility discussion guide questions
Feature-card exercise (cards optional)	Feature-card document (possibly created using only index cards)
Retrospective discussion	User scenarios Prototypes and/or mockup Iteration plan Maintenance plan Evolutionary requirements Additional documentation as required by the team/project Test plan Detailed schedule Launch plan Action items from project retrospective Test Driven Development (TDD) Agile estimating Daily stand-up meeting Demonstrations

These one-week projects are frequently driven by a need to increase readership or to provide support in the aftermath of a major news event such as an election.

Executive approval is almost immediate, and the projects use team members already assigned to the website. These teams only need the processes and documents outlined in the first column of table 2.2.

Conversely, Smart Trading Group pursues some major projects that require funding, synchronization with third parties, and identification of milestones. In these instances, the project teams review the items in the second column of table 3.1 and decide which ones to use in addition to the required ones in the first column.

In this way, agile provides the correct amount of structure for the project. Time isn't wasted on processes that don't add value, and teams can scale their processes mid-project if needed. Now that you understand the goals of an agile process, you need to know the best way to obtain them. You can do this by selecting a prepackaged agile process, creating a process from scratch, or a combination of the two agile principles can be applied in any environment, but some environmental characteristics influence how easy the principles are to adopt. Let's look at these characteristics.

Urgency to deliver in Smart Trading Group. Agile works best in an urgent environment. It provides tools to prioritize features quickly and determine how much scope to pursue within the constraints of a critical timeline. If you have urgency due to a competitive market, compliance deadlines, or a large backlog of project requests, agile provides methods for quicker delivery.

Evolving or volatile requirements in Smart Trading Group. One descriptor of agile could be *just enough*. "Give me just enough requirements to start a design." "Give me just enough design to start my code." "Give me just enough code to demonstrate some level of value to the customer." If you don't have all the requirements, you can still get started with an agile project. If you complete an iteration and the customer wants to change the requirements, you can adapt and still meet the objectives. Managing changing requirements still takes effort in an agile environment, but you don't have to fight the project framework. The framework is designed to support uncertainty.

Customer availability in Smart Trading Group. One Agile Manifesto principle states, "Business people and developers must work together daily throughout the

project.” In our experience, these groups don’t have to work together every day throughout a project cycle, but there are definite times when the customer must be available. In theory, a project must not be urgent if the customer can’t make time to clarify requirements or review functionality. The customer can have a proxy, such as a product manager; but someone needs to be available every day to represent the customer’s vision.

CONSISTENT RESOURCES in Smart Trading Group. Part of the power of agile is a level of familiarity within the team and a consistent understanding of the processes they use. Agile teams and processes get better over time. If project team members are new to each other, they must learn processes together while at the same time trying to complete the project. Agile works best with a core group of people who work together on continuous projects. Agile isn’t a good methodology to use with a team that has never worked together before, unless you have long-term plans to keep them together [13].

CO-LOCATED RESOURCES in Smart Trading Group. Agile promotes face-to-face communication and common understanding. One of the best ways to support this principle is to put your team members face to face. Co-location is an amazing tool. Team can get out of *email hell*, and their mutual understanding of the project will increase.

One of the best setups we have seen is at a Fortune 500 company we visited. All 10 of the project team members are in an area approximately 25 feet by 25 feet. The cubicles have half-walls that provided a level of privacy when people are sitting but let them easily see the rest of the team and communicate when they stand up. This setup provides the privacy the developers enjoy when they’re deep into a coding session but also lets team members stand up to converse with each other at any time without having to go to each others’ cubicles. Team members can also walk a few feet and reach common areas where they can whiteboard a design or have a quick caucus.

In larger companies than Smart Trading Group, a project team may be constructed of team members from a shared resource pool. For example, the QA (Quality Assurance) lead for a project may be from the QA shared resources pool. If

such team members view themselves as resources on loan, and not as team members dedicated to the project, the result can be functional silos.

When silos exist, team members are more concerned about the welfare of their team or area than they are with the livelihood of the project. This mentality doesn't bode well for agile development and leads to customer neglect. The team needs to bond as a unified group toward the goals of the project. Roles are assigned, but one of the objectives of agile is for the team to working collectively.

Working collectively can also be applied to team member roles. A tester can point out a possible code improvement. A developer can suggest a feature enhancement. In general, team members *speak out*—they don't limit their roles to their titles. Management should ensure that individual goals include how well employees support the common good of the project [19].

Now that you know the characteristics that make agile easier to implement, let's look at a few that make agile more difficult to move to.

LACK OF AGILE KNOWLEDGE in Smart Trading Group. First challenge will be finding expertise to help you with your migration. If you're fortunate, you'll have some level of agile experience within your company; but this probably won't be true to the point that you can coach yourself through an agile migration. We'll help you with this issue by showing you how often Smart Trading GroupMedia requested assistance, from initial training to issues encountered along the way.

LARGE PROJECT TEAMS in Smart Trading Group. Agile is compromised as team size increases. Major principles such as face-to-face communication and common understanding require additional effort to maintain their effectiveness as a team grows.

Larger teams require additional overhead to ensure that information is shared consistently across all groups. Scrum teams frequently use the term *scrum of scrums*, meaning a representative from each team Scrum attends a master Scrum meeting to share information with other groups. Jeff Bezos of Amazon.com believes that the most productive and innovative teams can be "fed with two pizzas." Jeff shared this thought with his senior managers at an offsite retreat. He envisioned a company culture of

small teams that could work independently, which would lead to more innovative products. Since that time, the Amazon “pizza teams” have created some of the most popular features on the site (Fast Company, 2004).

If your team has an average appetite, you can convert Jeff’s concept into a team of five to seven people. This is a nice-size group for communication and agility. If five to seven is perfect, then what is the maximum size for a team to remain agile? On the high side, we believe you can have a team of 15 people without major impact on your agility. When you have more than 15, communication needs to become more formal, which slows the team.

There are ways to make agile work with larger or distributed teams, but you’ll sacrifice some level of agility.

Related to large teams, many companies use distributed development. Frequently, the distributed development is performed by offshore resources. Distributed development implies that the team is large in size and that communication methods must be scaled to get information to all involved. In addition, you may have issues with time zone differences, language, and code integration into a common environment. Some offshore companies support and advertise the use of agile methodologies, but their location may make it challenging to support the core principles.

We’ve seen agile teams successfully use offshore resources for commodity or repeatable-type work, such as regression testing, smoke testing, and cookie-cutter development (for example, providing an offshore group with standardized tools to create automated workflows) [40].

FIXED-BID CONTRACT WORK in Smart Trading Group. Fixed-bid contract work goes against most of the agile principles. The customer isn’t a partner, evolving requirements are a no-no, and adapting is usually called *scope creep*.

We used to believe that fixed-bid work couldn’t be performed using an agile process, but recently we’ve met several managers who have customized their process to allow the inner workings to be agile while customer interaction remained contract oriented.

AN IMMATURE OR ONE-TIME TEAM in Smart Trading Group. If you have a team that will work together for only one project, they're usually better served by using a plan-driven methodology unless they have previous exposure to agile.

If the team will work through multiple projects or releases, you can introduce agile techniques, and the team can migrate to a full agile methodology as their knowledge matures.

It's agile. We don't need to do any planning to convert to it, just start thinking agile! A lot of folks take this approach when migrating to agile. But if you go too fast, you don't give your company enough time to digest the concepts. When this happens, you may experience issues with common understanding and terminology.

Don't let this happen to you. You need to plan before migrating to agile, and this project will show you how to do it with an *awareness, buy-in, ownership* approach. If you take your time, the methodology will stick, and you'll minimize the risk of failure.

TEAM WITH SPECIALIZED SKILL SETS An organization's structure can create artificial barriers between teams, and so can skill sets. If your team has specialized skill sets, it's hard to be agile when the work mix doesn't correlate well to the available resource types. Some tasks always have to be done by certain individuals, which doesn't help the team bond or unite when pursuing the completion of a feature.

Specialized skill sets also place an additional constraint on team capacity. Imagine that your team has only one person who can perform user-interface design, and the work assigned to an iteration is 80 percent user-interface work. Other team members can look for work to do outside of the iteration, but delivery will be slow due to the one-person constraint [62].

Teams that are just becoming agile usually have members with specialized roles. You can overcome this constraint by cross training over time and rewarding employees for obtaining and using additional skills.

AVOIDING CUSTOMIZATION in Smart Trading Group. Many people get hung up on the questions, "Are we doing it right? Are we doing it in an agile fashion? Are we following a pure agile process?" When teams ask us these questions, we tell them

the answers aren’t important. All we want to know is this: Have you created a development process that provides the most benefit to your company?

This same mentality has managers trying to find a perfect agile methodology and insert it directly into their company. As we discussed earlier, you can start with a packaged agile process, but you need to look at the realities of your company and adjust accordingly. Smart Trading GroupMedia will look at a generic agile process and see how it applies to their realities; then, they’ll modify the process to fit their environment.

PART 3. FORMATION OF A FLEXIBLE MANAGEMENT SYSTEM FOR INTERNATIONAL TRADE ENTERPRISES

3.1. Agile adoption using assessments in Smart Trading Group

Many managers hesitate to begin major change initiatives like becoming agile because they must objectively identify the risks involved with the transition. By conducting a comprehensive readiness assessment, you can prove to management that the organization possesses the necessary characteristics for a successful transition to agile.

Conducting readiness assessments can help identify and reduce the risks associated with the adoption process, because you have better insight into whether adopting the practice will succeed or fail before you begin the transition phase.

For example, collaborative planning is a commonly adopted agile practice. It calls for all stakeholders to be involved with the planning process, not only the project manager. This seems like a pretty simple practice, so it’s common for organizations to mandate it without any readiness assessment. But in reality, successfully adopting this practice relies on four organizational characteristics:

■ *Management style* —Before you begin using collaborative planning, you need to find out whether a collaborative or a command-control relationship exists between managers and the employees. The management style is an indication of whether management trusts the developers and vice versa. If management doesn’t trust the employees’ opinions, then collaborative planning may result in many arguments.

Manager buy-in —It’s great to know whether management supports collaborative planning. Many managers prefer to maintain control over the planning process and hence are apprehensive during collaborative planning sessions.

■ *Power distance* —Power distance is a characteristic related to organizational and national cultures. By measuring power distance, you can find out whether people are intimidated by their managers and afraid to participate and be honest in their

presence. If a big power distance exists between managers and employees, then having everyone sit around a table to plan together may not be as effective as you'd wish.

■ *Developer buy-in* —To reap the benefits of collaborative planning, the entire team (including the developers) should be willing to be part of a collaborative planning environment. As obvious as this may seem, many developers don't see any benefit in being part of the planning process, and therefore even if collaborative planning is mandated, they won't be active participants.

These are some of the organizational characteristics that you should assess before you attempt to adopt an agile practice like collaborative planning. The absence of some or all of these characteristics may result in a failed attempt to adopt the practice.

The problem is that when a change initiative fails for any reason, a number of dangerous, negative consequences may result:

■ Decreased team productivity

■ Unmotivated team

■ Increased team resistance to future change initiatives

■ Jeopardizing of management's credibility (if the change was mandated from management)

Table 2.4

The results of a sample agile-readiness assessment in Smart Trading Group. The results show the suitability of each of the characteristics needed to successfully adopt collaborative planning [36]

Characteristic	Suitability result
Management style Whether a collaborative or a command-control relation exists between managers and subordinates. The management style indicates whether management trusts the developers and vice versa.	Partially suitable (30.5%)
Manager buy-in Whether management supports or resists having a collaborative environment	Largely suitable (72.5%)
Power distance Whether people are intimidated by /	Largely suitable (60.5%)

afraid to participate and be honest in the presence of their managers	
Developer buy-in Whether the developers are willing to plan in a collaborative environment	Fully suitable (92.5%)

Let's dissect the readiness-assessment table and understand its layout. The assessment uses two main components: organizational characteristics and assessment indicators. *Organizational characteristics* are the various attributes you need to assess to determine whether a team or organization is ready to adopt a certain agile practice. These characteristics may be related to a number of different aspects of the organization,

most commonly the following:

- *Customers*—The project's customers and clients
- *Builders*—The technical staff involved with the development of the project
- *Managers*—The managers or executives overseeing the project and involved with decision making

APN	Adaptive planning involves delaying the detail planning of the next iteration until immediately before the start of the iteration. By delaying the planning to the last minute, the plan can incorporate the latest feedback obtained about the product so far, including what was learned from the previous iteration. Adaptive planning helps teams embrace change because the focus shifts from adhering to a plan (which makes people less embracing of change) to continuously planning based on the latest feedback obtained (which inherently promotes the culture of welcoming change).				
Various characteristics to be assessed to determine the team's readiness for this practice					**Indicators**
Management buy-in Whether the team's management is willing to base the planning for the next iteration on the client's feedback from the current (previous) iteration					APN_M1
Management buy-in Whether the team's management is willing to plan as late as possible for an iteration (immediately before the iteration)					APN_M2
Indicators (questions) to be answered by the manager(s)					
APN_M1	**The plan for upcoming iteration may change based on customer feedback from the previous or current iteration.**				
	Strongly Disagree	Tend to Disagree	Neither Agree nor Disagree	Tend to Agree	Strongly Agree
APN_M2	**You agree with developing the detailed plan for an iteration only after the conclusion of the previous iteration.**				
	Strongly Disagree	Tend to Disagree	Neither Agree nor Disagree	Tend to Agree	Strongly Agree

Figure 2.7 The readiness-assessment table for the agile practice of collaborative planning [8]

■ *Tools*—The software tools used within the organization or for a certain project

■ *Culture*—The overall culture of the people within an organization or the project team

■ *Project management*—The procedures and practices related to managing projects in the organization

■ *Software process*—The activities and artifacts related to the software-development process in the organization

■ *Physical environment*—The physical layout of the organization and the geographical and spatial distribution of its employees

Indicators are the questions you use to assess each organizational characteristic. Indicators can be targeted at four different groups in the organizations:

■ *Developers*—Team members who are involved in building the actual system. They usually include developers/coders, architects, and testers.

■ *Managers*—Any team members involved with management of the project. This role is suitable for project managers, team leaders, and any other management positions in direct relation with the project.

Product owners—Team members who are involved with the product's business direction. This role is suitable for any team member who is in direct contact with the customer. Some of the common positions that fall under this role are business analyst, product manager, product leader, engagement manager, and project manager (if they're in contact with the client).

■ *Assessors*—People outside the team. Their main role is to observe whether certain process activities and artifacts exist. Common positions that fall under this role are agile coaches, quality-assurance personnel, process-improvement personnel, and independent observers outside the team.

Smart Trading Group needs to complete four calculations assessment from employers to determine the final results.

STEP 1: COMPUTE A WEIGHT FOR EACH INDICATOR The first step is to assign a weight to each indicator. A *weight* is a fractional value between 0 and 1 that expresses the indicator's level of influence on the characteristic being assessed. The weights of all the indicators belonging to the same characteristic must sum to 1 At this point, it doesn't care which indicators are answered by whom—all he is interested in is the total number of indicators required to assess the team's management style. Jay computes the weights as follows (assuming all indicators have an equal influence on the parent factor):

1 (sum of all weights) / 4 (number of indicators, including developers and managers) = 0.25 (weight per indicator)

STEP 2: COMPUTE WEIGHED INTERVALS After Jay computes the weight for each indicator, the next step is to compute the *weighted intervals* for each of the indicators. To achieve more accurate assessment results, each answer represents a range of values, not a fixed number. Table 2.3 shows the lists of ranges assigned to each answer in the readiness assessment

Table 2.5

The range of values assigned to each answer option in the readinessassessment survey [41]

Answer	Value range
Strongly Disagree	0–15%
Tend to Disagree	15–40%
Neither Agree nor Disagree	40–60%
Tend to Agree	60–85%
Strongly Agree	85–100%

These ranges can change, depending on the threshold and the answer values the assessor uses. Table 2.6 shows another set of answers with different value ranges

Table 2.6

A set of numeric values assigned to different answer options in the readinessassessment survey [68]

Answer	Value range
Never	0–20%
Rare	20–50%
Seldom	50–80%
Frequently / Usually	80–100%
Always	85–100%

What's important is to ensure that you have a suitable range for each of the answer values in the assessment. Jay starts to compute the weighted intervals for management style. Table 2.7 shows the answers given to the sample indicators.

Table 2.7

Answers provided during the readiness assessment for the Management style characteristic [29]

	Strongly Disagree	Tend to Disagree	Neither Agree nor Disagree	Tend to Agree	Strongly Agree
Indicator	0-15%	15%-40%	40%-60%	60%-85%	85%-100%
COP_M1		1			
COP_M2		1			
COP_D1		1		1	
COP_D2	1	1			

Once you have the answers from the sample indicators, the next step is to multiply the weight of the indicator by the high and low end of the interval range selected for the indicator.

Table 2.5

All the answers provided during the readiness assessment are converted into number ranges and then multiplied by the weight of each assessment indicator

Reference number	Computed weight	Interval low end	Interval high end	Interval low end × weight	Interval low end × weight
COP_M1	0.25	15	40	15 x 0.25 = 3.75	40 x 0.25 = 10
COP_M2	0.25	15	40	15 x 0.25 = 3.75	40 x 0.25 = 10
COP_D1	0.25	(15 + 60)/2 = 37.5	(40 + 85)/2 = 62.5	15 x 0.25 = 3.75	62.5 x 0.25 = 15.6
COP_D2	0.25	(0 + 15)/2 = 7.5	(15 + 40)/2 = 27.5	7.25 x 0.25 = 1.8	27.5 x 0.25 = 6.8

STEP 3: CALCULATE THE RESULT RANGE

The next step is to compute the result range by calculating the *optimistic* and *pessimistic range* for each characteristic. You do this by summing up all the weighed intervals you obtained from the previous step. The following shows some of calculations:

Pessimistic result = Sum of all the weighted low-end results from step 2

Pessimistic result: 3.7 + 3.7 + 9.4 + 1.8 = 18.6

Optimistic result = Sum of all the weighted high-end results from step 2

Optimistic result: 10 + 10 + 15.6 + 6.8 = 42.4 Result in terms of an interval = 18.6–42.4 Result as a single number = (18.6 + 42.4) / 2 = 30.5

STEP 4: TRANSLATE TO A NOMINAL SCORE

Table 2.9 shows a list of the nominal values used for Smart Trading Group's readiness assessment. Although this step is optional, people will be able to read your report more easily if the results are translated to a nominal value. These nominal

values are used to evaluate the suitability of the characteristic to support the successful adoption of the agile practice. If the result range from step 3 fits within one of these nominal value intervals, then that suffices; if it doesn't, then you need to obtain an average and see where that number lies in the nominal-value intervals.

Manager's calculated result for management style falls between the *Not suitable* and *Partially suitable* values. So he uses the single-number result and finds that management style barely makes it into the *Partially suitable* range.

If you want your migration to agile to last beyond a few projects, you need the change to be driven from within by key players throughout the company. You need to establish a team based on the people who build and deliver your software today.

The role of this group, which we call the *core team*, is to learn as much as they can about agile and to use this knowledge to add agility to your existing process with the help of an agile coach. The team collaborates and reaches consensus on new processes; then they mentor project teams as they use agile techniques. This core team is powerful and influential for three reasons:

■ *They aren't a part of line management.* A few members may come from the management ranks, but the majority of the team are *doers*: people who design, build, create, and test code. This adds to the team's credibility as you roll out the methodology to the company. Agile isn't a management initiative being forced on everyone; it's coming from real people who will be a part of the project teams.

■ *Because the team is composed of doers, they know the ins and outs of developing in your environment.* This is different than when consultants come in, suggest standard practices, and disregard the realities of a specific company. The core team has experience with your company, and as they develop a methodology they know what to keep and what to discard from existing practices.

■ *Having team members from all areas initializes awareness across the company.* Imagine a tester going back to the testing team and excitedly telling them what is going on with the new methodology, or a developer doing the same with the development team.

Many companies use outside consulting to get their methodology going. We've seen several companies choose to go with agile methods such as Scrum and then have a third party come in to train employees and design and deploy the methodology. In our opinion, this approach isn't as effective as growing the methodology from within.

Creating it from within the organization addresses all the issues with ownership. It's hard to get a team to buy into a process that was forced on them. (Note that in rare cases, an organization is so dysfunctional that it needs to have a methodology forced on it—but this should be the exception, not the norm.)

We support using an agile coach along the way, but we prefer coaches who use a Socratic approach. This type of coach asks you questions that lead you to your own answers.

The managers will probably want you to provide a time estimate. You can take two approaches to the work the core team performs:

■ *Get the work done as quickly as possible.* This is the preferred approach. You make process work the number-one priority for the group for 1 to 3 weeks.

■ *Have team members work part-time on the core team.* Many teams can't pull several team members away from their daily work for a solid 1 to 3 weeks. Greg experienced this constraint at the *Seattle Times*. To compensate, Greg's team worked on the new process three times a week for 2 hours at a time. Using this process, the duration for establishing a new process was 6 weeks. Greg's team enjoyed the slower process, though, because it gave them more time to think about what they were designing.

Table 2.8

Smart Trading GroupMedia's core team. Core teams are composed of cross-functional team members withvarious levels of agile knowledge. The diversity of the team works well for scrutinizing the new process [18]

Functional area/Role	Background
Development	Familiar with Extreme Programming (XP) development tech niques but comfortable with the waterfall/homegrown pro cess that Smart Trading

	GroupMedia has used for the last few years
Quality assurance	Concerned that agile will bypass or minimize the need for testing. Experience working in an ISO 9000 environment. Frequently says “document what you do, do what you document
Operations	Exists in a stressful world of managing production issues and deploying new functionality. Worried that he won’t have enough time to work with the core team.
Requirements	An agile zealot. Has been looking forward to this day for a long time. Dedicated to making agile work at Smart Trading Group. Works with Product Management to refine feature design for customers.
Architecture	Wants to make sure that an agile methodology doesn’t bypass good architectural practices and that there is enough time to build the infrastructure needed for projects.
Architecture	Unfamiliar with agile but excited about the promise to embrace the customer and changing requirements. Identifies target markets and strategic needs for Acme Media’s products.

Smart Trading Group need to get manager approval for the employees you select for your team. The managers will probably want you to provide a time estimate. You can take two approaches to the work the core team performs:

■ *Get the work done as quickly as possible.* This is the preferred approach. You make process work the number-one priority for the group for 1 to 3 weeks.

■ *Have team members work part-time on the core team.* Many teams can’t pull several team members away from their daily work for a solid 1 to 3 weeks. Greg experienced this constraint at the *Seattle Times*. To compensate, team worked on the

new process three times a week for 2 hours at a time. Using this process, the duration for establishing a new process was 6 weeks. Greg's team enjoyed the slower process, though, because it gave them more time to think about what they were designing.

After you select your team, you need to meet with the core-team members and set expectations.

An agile team comes across as poised and ready for wherever the project may lead them. Agile team members don't fear uncertainty; they look forward to the challenge and know they will succeed. Where does this air of self assurance come from? Does this attitude reflect the type of people who were hired? Or does it reflect the processes that are being used? Is the attitude a byproduct of executive support? Does confidence come from a history of successful deliveries?

The answer to all of these questions is *Yes*. Each of these items supports the effectiveness and self-reliance that is inherent in an agile team. In some ways, creating an agile team is like baking a cake. You can obtain the ingredients exactly as the recipe requests, bake at the suggested temperature, and let the cake cool the specified time before applying the icing. But what happens if you're at high altitude and you forget to make the necessary adjustments? The cake rises too quickly and then turns out too dry. Or what if someone jumps up and down in the kitchen while the cake is baking? The cake collapses and never rises. In this section, we'll give you the ingredients for creating your agile team. We'll walk you through "high-altitude baking" and how you should adjust your recipe accordingly

3.2. Ways to effectively implement flexible methodologies

Smart Trading Group has three product groups: the news site, the classifieds site, and a travel/outdoors site. Each group has its own processes for development. To minimize complexity and expedite the migration, the core team has decided to document only the classified's development process for now. The focus of this site is advertisements for real estate, autos, and merchandise. Smart Trading Grouphas

assigned core-team members to various areas to document the phases. The assignments are based on experience.

The Smart Trading Groupteam members have chosen to use butcher paper and index cards to document their methodology. They find this approach useful because they can see the progress that each mini-research team is making as they document their respective areas. It also lends itself to questions when the team meets daily to review progress on the documentation exercise. Smart Trading Grouphas time-boxed its reverse engineering work to ensure the process doesn't go on for months. One week is allotted for documenting the existing methodology. Most of the work will be performed offline from the core team meeting; then, the group will meet to review the findings gathered by subteams and individuals.

Our case study, Smart Trading Group, will represent your company. We'll take Smart Trading Groupthrough these nine steps and show you how the company iteratively creates and tests a custom process. We'll also show you how Smart Trading GroupMedia takes its own constraints into account with the new methodology. Before we jump into the case study, let's spend a moment looking at the characteristics that make it easier to adopt agile and the characteristics that make agile adoption more challenging.

Characteristics that make agile easier to adopt in Smart Trading Group Fgile principles can be applied in any environment, but some environmental characteristics influence how easy the principles are to adopt. Let's look at these characteristics.

Urgency to deliver in Smart Trading Group. Agile works best in an urgent environment. It provides tools to prioritize features quickly and determine how much scope to pursue within the constraints of a critical timeline. If you have urgency due to a competitive market, compliance deadlines, or a large backlog of project requests, agile provides methods for quicker delivery.

EVOLVING OR VOLATILE REQUIREMENTS in Smart Trading Group. One descriptor of agile could be *just enough*. "Give me just enough requirements to start a design." "Give me just enough design to start my code." "Give me just enough code to demonstrate some level of value to the customer." If you don't have all the

requirements, you can still get started with an agile project. If you complete an iteration and the customer wants to change the requirements, you can adapt and still meet the objectives. Managing changing requirements still takes effort in an agile environment, but you don't have to fight the project framework. The framework is designed to support uncertainty.

CUSTOMER AVAILABILITY in Smart Trading Group. One Agile Manifesto principle states, "Business people and developers must work together daily throughout the project." In our experience, these groups don't have to work together every day throughout a project cycle, but there are definite times when the customer must be available. In theory, a project must not be urgent if the customer can't make time to clarify requirements or review functionality. The customer can have a proxy, such as a product manager; but someone needs to be available every day to represent the customer's vision.

CONSISTENT RESOURCES in Smart Trading Group. Part of the power of agile is a level of familiarity within the team and a consistent understanding of the processes they use. Agile teams and processes get better over time. If project team members are new to each other, they must learn processes together while at the same time trying to complete the project. Agile works best with a core group of people who work together on continuous projects. Agile isn't a good methodology to use with a team that has never worked together before, unless you have long-term plans to keep them together.

CO-LOCATED RESOURCES in Smart Trading Group. Agile promotes face-to-face communication and common understanding. One of the best ways to support this principle is to put your team members face to face. Co-location is an amazing tool. Your team can get out of *email hell*, and their mutual understanding of the project will increase.

One of the best setups we have seen is at a Fortune 500 company we visited. All 10 of the project team members are in an area approximately 25 feet by 25 feet. The cubicles have half-walls that provided a level of privacy when people are sitting but let them easily see the rest of the team and communicate when they stand up. This setup provides the privacy the developers enjoy when they're deep into a coding session but

also lets team members stand up to converse with each other at any time without having to go to each others' cubicles. Team members can also walk a few feet and reach common areas where they can whiteboard a design or have a quick caucus.

In larger companies, a project team may be constructed of team members from a shared resource pool. For example, the QA (Quality Assurance) lead for a project may be from the QA shared resources pool. If such team members view themselves as resources on loan, and not as team members dedicated to the project, the result can be functional silos.

When silos exist, team members are more concerned about the welfare of their team or area than they are with the livelihood of the project. This mentality doesn't bode well for agile development and leads to customer neglect. The team needs to bond as a unified group toward the goals of the project. Roles are assigned, but one of the objectives of agile is for the team to working collectively. Working collectively can also be applied to team member roles. A tester can point out a possible code improvement. A developer can suggest a feature enhancement. In general, team members *speak out*—they don't limit their roles to their titles. Management should ensure that individual goals include how well employees support the common good of the project.

Now that you know the characteristics that make agile easier to implement, let's look at a few that make agile more difficult to move to. First challenge will be finding expertise to help you with your migration. If you're fortunate, you'll have some level of agile experience within your company; but this probably won't be true to the point that you can coach yourself through an agile migration. We'll help you with this issue by showing you how often Smart Trading Group requested assistance, from initial training to issues encountered along the way.

Agile is compromised as team size increases. Major principles such as face-to-face communication and common understanding require additional effort to maintain their effectiveness as a team grows.

Larger teams require additional overhead to ensure that information is shared consistently across all groups. Scrum teams frequently use the term *scrum of scrums*,

meaning a representative from each team Scrum attends a master Scrum meeting to share information with other groups.

Jeff Bezos of Amazon.com believes that the most productive and innovative teams can be “fed with two pizzas.” Jeff shared this thought with his senior managers at an offsite retreat. He envisioned a company culture of small teams that could work independently, which would lead to more innovative products. Since that time, the Amazon “pizza teams” have created some of the most popular features on the site (Fast Company, 2004).

If your team has an average appetite, you can convert Jeff’s concept into a team of five to seven people. This is a nice-size group for communication and agility. If five to seven is perfect, then what is the maximum size for a team to remain agile? On the high side, we believe you can have a team of 15 people without major impact on your agility. When you have more than 15, communication needs to become more formal, which slows the team.

There are ways to make agile work with larger or distributed teams, but you’ll sacrifice some level of agility.

DISTRIBUTED DEVELOPMENT in Smart Trading Group. Related to large teams, many companies use distributed development. Frequently, the distributed development is performed by offshore resources. Distributed development implies that the team is large in size and that communication methods must be scaled to get information to all involved. In addition, you may have issues with time zone differences, language, and code integration into a common environment. Some offshore companies support and advertise the use of agile methodologies, but their location may make it challenging to support the core principles. We’ve seen agile teams successfully use offshore resources for commodity or repeatable-type work, such as regression testing, smoke testing, and cookie-cutter development (for example, providing an offshore group with standardized tools to create automated workflows).

FIXED-BID CONTRACT WORK in Smart Trading Groupю Fixed-bid contract work goes against most of the agile principles. The customer isn’t a partner, evolving requirements are a no-no, and adapting is usually called *scope creep*.

We used to believe that fixed-bid work couldn't be performed using an agile process, but recently we've met several managers who have customized their process to allow the inner workings to be agile while customer interaction remained contract oriented.

Team that will work together for only one project, they're usually better served by using a plan-driven methodology unless they have previous exposure to agile.

If the team will work through multiple projects or releases, you can introduce agile techniques, and the team can migrate to a full agile methodology as their knowledge matures.

An organization's structure Smart Trading Group can create artificial barriers between teams, and so can skill sets. If your team has specialized skill sets, it's hard to be agile when the work mix doesn't correlate well to the available resource types. Some tasks always have to be done by certain individuals, which doesn't help the team bond or unite when pursuing the completion of a feature.

Specialized skill sets also place an additional constraint on team capacity. Imagine that your team has only one person who can perform user-interface design, and the work assigned to an iteration is 80 percent user-interface work. Other team members can look for work to do outside of the iteration, but delivery will be slow due to the one-person constraint.

Teams that are just becoming agile usually have members with specialized roles. You can overcome this constraint by cross training over time and rewarding employees for obtaining and using additional skills.

Many people get hung up on the questions, "Are we doing it right? Are we doing it in an agile fashion? Are we following a pure agile process?" When teams ask us these questions, we tell them the answers aren't important. All we want to know is this: Have you created a development process that provides the most benefit to your company This same mentality has managers trying to find a perfect agile methodology and insert it directly into their company. As we discussed earlier, you can start with a packaged agile process, but you need to look at the realities of your company and adjust accordingly. Smart Trading GroupMedia will look at a generic agile process and

see how it applies to their realities; then, they'll modify the process to fit their environment.

Many managers hesitate to begin major change initiatives like becoming agile because they must objectively identify the risks involved with the transition. By conducting a comprehensive readiness assessment, you can prove to management that the organization possesses the necessary characteristics for a successful transition to agile. If your organization isn't ready for certain agile practices, the assessment will help pinpoint exactly which characteristics need to be enhanced. This information will help management make a more informed decision about whether the organization should start the agile initiative.

Figure 3.1 shows part of a complete readiness assessment report from an online readiness assessment tool, created by Ahmed, named Dr. Agile (www.dragile.com). As you can see, the first column lists different agile practices. To the right of each practice are the organizational characteristics that are assessed; the final columns show the assessment results. A report like this is beneficial and insightful to executives because it shows them the amount of effort needed to adopt agile in the organization. Many times, executives are reluctant to start an agile adoption initiative because there are too many unknowns. One of these important unknowns is whether the organization is ready. Readiness assessments give executives visibility into the amount of effort (which they translate into cost) required for the adoption process.

When executives looks at this report and sees that their organization has all the necessary characteristics for a successful adoption, they're more inclined to support and even champion the initiative. If the organization needs to enhance some of the characteristics required for the adoption of agile practices, then executives have the option to either undertake the necessary steps to improve these characteristics or go ahead with the adoption of those practices the organization is currently ready for.

We believe that readiness assessments provide executives with the right amount of information, visibility, and insights to make them support the agile transition initiative.

Agile Practice	Assessment Area	Assessment Characteristics	Not Achieved 0% - 30%	Partially Achieved 30% - 60%	Largely Achieved 60%-85%	Fully Achieved 85% - 100%
Retrospectives	Managers	Willingness to participate in retrospectives				92
	Adaptability of Process	Ability to change and improve the software process in the middle of a project				92
Collaborative Planning	Project Planning	Willingness to plan with others	29			
	Management Style	Collaboration		50		
	Management Transparency	Openness with developers		48		
	Power Distance	Intimidation to give honest feedback			79	
Collaborative Teams	Interaction	Existence of interaction between developers		33		
	Culture	Willingness to work in collaborative teams				92
		Belief in group work			71	
Coding Standards	Standards	Existence of coding standards				100
	Discipline	Willingness to abide to coding standards			71	
...	...	...	...	...	...	...

Figure 3.1. Part of a readiness-assessment report generated by Dr. Agile (www.dragile.com). The report shows each agile practice (far-left column) and the degree to which its supporting characteristics are achieved in the organization (far-right columns) [52]

Smart Trading Group identified a set of 20 or so common agile practices. Then we created a readiness-assessment table for each of these practices. By creating a separate readinessassessment table for each practice, we've given people the flexibility to assess their team/organization for one particular practice without having to go through the assessment questions for all the other practices; and we've made the assessment extensible, because new practices can easily be added to the assessment by creating readiness-assessment tables for them. The next section will show you the readinessassessment table for an agile practice and take you through a step-by-step description of how to use the table to determine whether your team or organization is ready to adopt this practice [17].

The assessment uses two main components: organizational characteristics and assessment indicators. *Organizational characteristics* are the various attributes you need to assess to determine whether a team or organization is ready to adopt a certain agile practice. These characteristics may be related to a number of different aspects of the organization, most commonly the following:

■ *Customers*—The project's customers and clients

■ *Builders*—The technical staff involved with the development of the project

■ *Managers*—The managers or executives overseeing the project and involved with decision making

COP	Collaborative Planning All stakeholders of the project, developers, managers and business people come together during the planning activities of the project				
Various characteristics to be assessed to determine the team's readiness for this practices					**Indicators**
Management Style: Whether or not a collaborative or a command-control relation exists between managers and subordinates. The management style is an indication of whether or not management trusts the developers and vice-versa.					COP_M1, COP_M2 COP_D1, COP_D2
Manager's Buy-In Whether or not management is supportive of or resistive to having a collaborative environment					COP_M3
Power Distance: Whether or not people are intimidated/afraid to participate and be honest in the presence of their managers					COP_D3, COP_D4
Developer's Buy-in Whether or not the developers are willing to plan in a collaborative environment					COP_D5
Indicators (questions) to be answered by the project manager(s)					
COP_M1	**Irrelevant of your personal preferences, as a manager you actively encourage team work over individual work**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_M2	**You frequently brainstorm with the people you are managing**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_M3	**It is beneficial to have developers and business people take part in creating the project plan**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
Indicators (questions) to be answered by the developers					
COP_D1	**Your manager encourages you to be creative and does not dictate what to do exactly**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_D2	**Your manager gives you the authority to make some decisions without referring back to him/her**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_D3	**It is acceptable for you to express disagreement with your manager(s) without fear of their retribution**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_D4	**People's titles and positions are not a great cause of intimidation in the organization**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree
COP_D5	**You would like to participate in the planning process of the project you work on**				
	Strongly Disagree	Tend to Disagree	Neither Agree or Disagree	Tend to Agree	Strongly Agree

Figure 3.2 The readiness-assessment table for the agile practice of collaborative planning **[65]**

■ *Tools*—The software tools used within the organization or for a certain project

■ *Culture*—The overall culture of the people within an organization or the project team

■ *Project management*—The procedures and practices related to managing projects in the organization

■ *Software process*—The activities and artifacts related to the software-development process in the organization

■ *Physical environment*—The physical layout of the organization and the geographical and spatial distribution of its employees [56]

Indicators are the questions you use to assess each organizational characteristic. Indicators can be targeted at four different groups in the organizations:

■ *Developers*—Team members who are involved in building the actual system. They usually include developers/coders, architects, and testers.

■ *Managers*—Any team members involved with management of the project. This role is suitable for project managers, team leaders, and any other management positions in direct relation with the project.

■ *Product owners*—Team members who are involved with the product's business direction. This role is suitable for any team member who is in direct contact with the customer. Some of the common positions that fall under this role are business analyst, product manager, product leader, engagement manager, and project manager (if they're in contact with the client).

■ *Assessors*—People outside the team. Their main role is to observe whether certain process activities and artifacts exist. Common positions that fall under this role are agile coaches, quality-assurance personnel, process-improvement personnel, and independent observers outside the team. Figure 4.5 shows where the organizational characteristics and indicators are laid out in the readiness-assessment table for collaborative planning.

The top part of the table names the agile practice and briefly describes it. Section A lists the organizational characteristics. In this example, four characteristics should

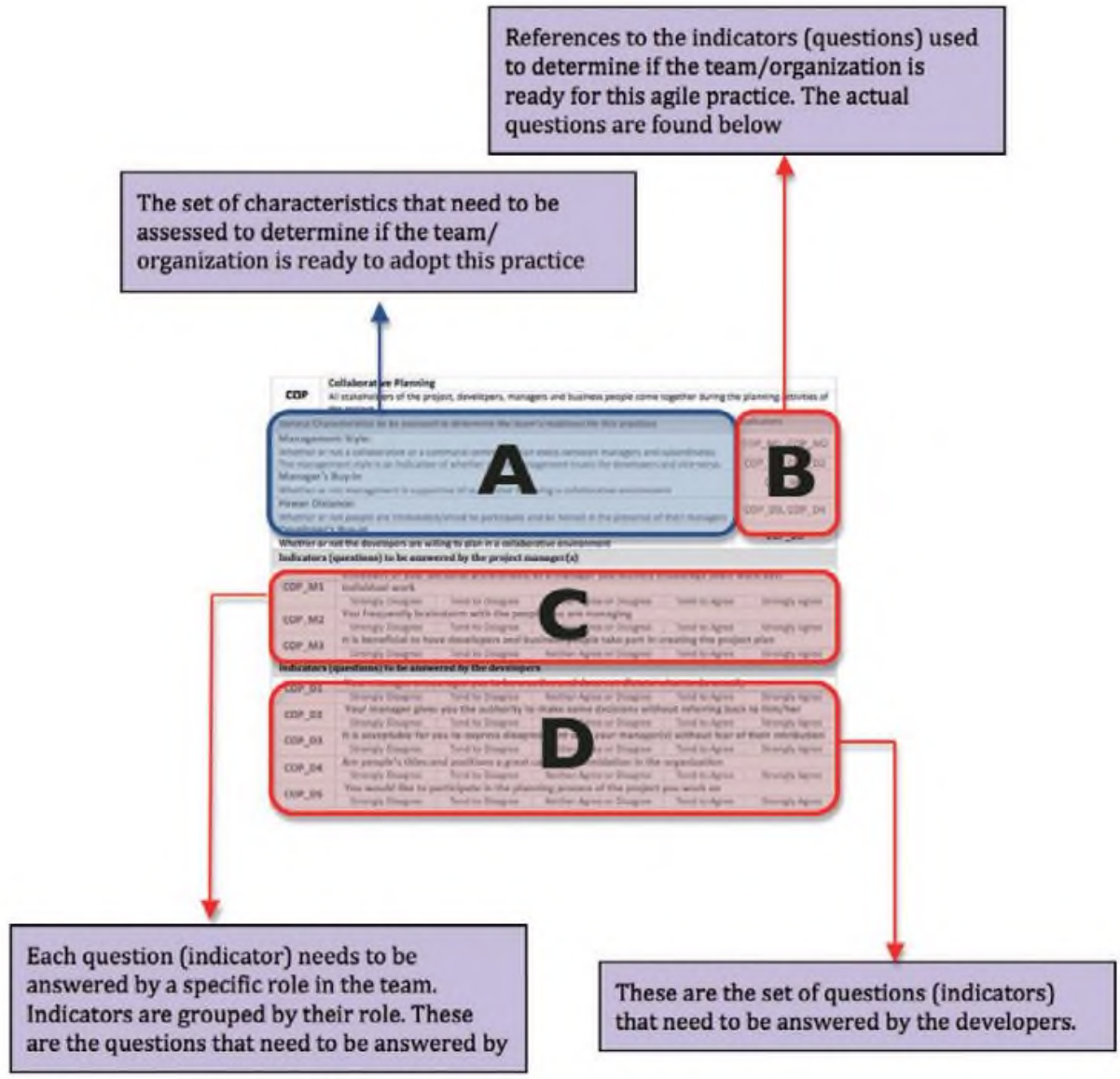


Figure 3.3 Dissecting the layout of a readiness-assessment table

be assessed for collaborative planning (management style, manager buy-in, power distance, and developer buy-in). Each characteristic is accompanied by a brief explanation of what you want to discover by assessing it. For example, as mentioned earlier, you assess power distance to determine whether people are intimidated by their managers and afraid to participate and be honest in the presence of those managers.

In front of each characteristic (section B) are reference codes. These codes refer to the indicators that are used to assess the organizational characteristic. The first letter

after the underscore in the code denotes whom the indicator is targeted at. If the letter is an *M*, then a manager should answer the indicator. The letter *D* refers to a developer, *C* to a customer or product owner, and *A* to an assessor. The indicators are grouped depending on this letter. In this example, we have only three indicators for managers and five for developers; indicators targeted at managers are grouped together (section C), and indicators targeted at developers are also grouped together (section D). If we were analyzing an agile practice that had more indicators targeted at assessors, then we would have a section E (beneath D) that contained those indicators.

The main change in the development area is that Smart Trading Group will no longer use the *waterfall model* for development. In the past, the developer received a functional specification and started building from it. In the new model, the developer has the feature card, and all the information is recorded on it. The developer works with the requirements team and the product manager to build to the minimum specification so a demonstration and validation can be obtained quickly. Another significant change is the daily stand-up meeting.

The development team used to meet daily to discuss status during development; now, most of the project team (not just developers) is there to synchronize on information and quickly resolve roadblocks to keep development rolling. Table 3.1 outlines the Development phase

Table 3.1

Smart Trading Group's new Development phase [81]

ID	Development phase	Group(s)	Change notes
1	Perform iteration 0: development initiation work	Project team with potential executive assistance	In the past, Smart Trading Groupdevelopers received a func tional specification to build from at this point. This step is used to put foundation pieces in place, such as architecture, vendor contracts, and environment preparation.
2	Perform development	Project team	The team swarms on the features to clarify the design and build the

	iterations 1–n.		functionality. Unit testing occurs at a minimum. Demonstrations are scheduled for the end of the iteration, but impromptu customer demonstrations can happen within the iteration.
3	Hold a daily stand-up meeting (part of the development iterations)	Project team	Smart Trading GroupMedia had weekly status meetings in the past, but they were only attended by managers who reported status for their teams. The new daily stand-up meeting is limited to 15 minutes, and it's "standing." The team discusses what has been done, what will be done, and any roadblocks or issues. The team discusses the status of features and whether they're ready to be integrated.
4	Integrate-build.	Development implementation	In the past, Smart Trading Groupwaited until the end of devel opment to integrate and build. Optimally, they would like to integrate every day, but for now they will settle for integrating three times a week.
5	Test.	Quality assurance	Smart Trading Grouphasn't made many changes to the testing process. The team considered test-driven development, but the assessment they completed indicated they aren't mature enough to pursue it at this time.
6	Repair bugs.	Development	Previously, Smart Trading Groupreserved a few weeks at the end of the project to do bug cleanup. In the new model, bugs identified during an iteration affect capacity for subsequent iterations. Smart Trading Group reserves some time during the deployment phase to clean up bugs, but major bugs are treated as features.
7	Update maintenance	Development	In the past, the team waited until deployment to create maintenance and

	and support plans.		support plans. Now they consider them part of feature delivery
8	Complete the iteration.	Project team	This step indicates the end of the time allotted for the iteration.

Development phase is focused around delivering value early and early validation of customer requirements. Testing also occurs sooner so that issues are easier to trace and repair.

Smart Trading Group also needs to consider a new phase that was previously limited to the end of the project: the Adapt phase.

In the past, *change* was a bad word at Smart Trading Group. If you needed to change requirements, the schedule, the scope, or some other project attribute, you had to create a change request. Now, change is expected, and the Adapt phase is dedicated to reacting to change. See table 8.7. Smart Trading GroupMedia still requires change requests for some items, specifically those that require incremental cost; but most changes are embraced, and the team works to deliver what is needed at the end, not what was requested at the beginning

Table 3.2

Smart Trading GroupMedia's Adapt phase embraces change [65]

Step	Adapt phase	Adapt phase	Change notes
1	Perform customer acceptance.	Requirements and guests	In the past, Smart Trading Groupperformed customer acceptance at the very end of development. The new process allows several reviews. The deliverables for the iteration are presented to the customer for review, testing, and ultimately acceptance. When the review is kicked off, stakeholders are invited to see the overall status of the iteration. In the past, the review was only for customers.

			This step helps prevent surprises. If features aren't accepted, the team reviews the issues and makes a decision on whether to continue work on the feature into the next iteration. In the past, rework time was limited to the bug-fix window.
2	Undertake discovery	Project team	The team reviews new information that materializes during development: business-climate changes, com petitor product changes, priority changes, and so on.
3	Evaluate the iteration pace.	Project manager	The project manager reviews development's actual velocity versus the forecast velocity and adjusts the features assigned to the next iteration accordingly.
4	Re-plan.	Project manager and team	The team modifies the plan for the next iteration based on all the information gleaned during the Adapt phase.

The team focuses on learning as the project progresses and delivering what is needed at the end, not necessarily what was requested at the beginning.

Adaptation occurs throughout Smart Trading Group's development lifecycle, but it's stressed during the Adapt phase. Customer demonstrations provide the ultimate opportunity to validate whether the product is on track and, if not, what needs to be done to redirect. This phase more than any other reveals what agile is all about. Change is unavoidable in a project, so the methodology should embrace change. The team will be busy enough reacting to the change; they don't need additional hassles from the process.

Smart Trading Group hasn't identified many functional changes for the Deployment phase, but they have found some cultural areas to work on; see table 3.3

Table 3.3

Smart Trading GroupMedia's Deployment phase

Step	Deployment phase	Group(s)	Change notes
1	Train support groups	Implementation	This step is unchanged.
2	Finalize maintenance plans.	Development	In the past, the entire plan was created days before deployment; now it's tweaked and final ized. Work began back in the Planning phase.
3	Finalize operation and support plans.	Implementation	The team finalizes these plans versus doing all the work at this point.
4	The team finalizes these plans versus doing all the work at this point.	Project Team	The team finalizes the documentation versus doing all the work at this point
5	Deploy the code to disaster recovery	Implementation	Smart Trading Grouphas had issues deploying in

			the past. To mitigate risk, they will deploy to the disaster recovery environment first to identify potential production deployment issues.
6	Deploy the code to production		This step is now performed after disaster-recov ery deployment.
7	Hold a lessons-learned (retrospective) meeting	Project team	Smart Trading Grouphas never stopped to review its processes between releases.
8	Celebrate	Project team and stakeholders	Things have been chaotic lately, and Smart Trading Grouphas stopped celebrating. The company needs to return a sense of accomplishment to the team.

Smart Trading GroupMedia's Deployment phase finalizes work that has been in progress since the Feasibility phase. Items such as maintenance plans have been discussed and worked on throughout the project. The team has also added a step to stop and reflect on how well the process is working—a retrospective.

The team also notes that they've quit celebrating at the end of projects. After many change requests, schedule slips, and chastisement from the executives, the team

hasn’t been in a celebratory mood lately. The team believes in the new process, and they believe that the future will warrant success, so celebrations have been added as an anticipated part of the delivery process.

3.3. Pilot project of forming a flexible management system in a trading enterprise

The pilot will be the first time the new lifecycle is exposed on a real project with a real team. In effect, it’s a marketing event for the new process. If you choose the wrong type of pilot, you may end up aborting, which will be a poor advertisement for the new methodology. With that thought in mind, you want to select a project that will push you through the test but not *shove* you. You want time to test the process in all areas such as requirements, design, development, testing, and implementation. You also want to give your pilot team time to acclimate to their new level of ownership. Agile is about methodology and culture. The team should understand the literal process, but they should also begin to understand *what it means to be agile*. You want them to start envisioning what it’s like to own a project and be highly involved in decisions. Let’s look at the traits of a good pilot project.

Pilot project should have an overall completion estimate somewhere between a couple of weeks and a maximum of 8 weeks. One of the easiest ways to complicate your migration is to test your new methodology on a large project.

Smart Trading Group populates its request backlog via a quarterly planning process. The executives review all known project requests once every three months, prioritize them, and loosely assign them to the quarter for completion. Smart Trading Group supports a website that includes news, classifieds, and travel/outdoors content. Executives from these three areas attend the quarterly planning process along with managers from support areas such as online advertising, user registration, and engineering. Smart Trading Group’s project backlog is pictured in table 3.4.

Table 3.4

Smart Trading Group's project backlog [54]

Area	Potential projects	Priority H-M-L	Priority H-M-L	Request type	Detailed description
News	My News" personalization	M	9 weeks	Customer	User can build their own page with content from AcmeNews. com along with other sites.
Classifieds	Free merchan dise advertising	M	7 weeks	Customer	Compete with eBay and Craig slist for lost classified adver tisements.
All	Free merchan dise advertising	M	2 weeks	Infrastructure	Many readers' videos are being blocked by firewalls. Investigate Flash as a solution.
Classifieds	Autos email	L	1 weeks	Customer	Many readers'

	alerts				videos are being blocked by firewalls. Investigate Flash as a solution.
News	Wireless weather alerts	M	3 weeks	Customer	Many readers' videos are being blocked by firewalls. Investigate Flash as a solution.
News	Behavioral targeting	M	10 weeks	Advertiser	Serve tailored ads to the reader based on their browsing habits.

When the Smart Trading Group quarterly planning team sits down to identify their pilot project, they invite the project manager, Wendy Johnson, to assist. Wendy recently trained on the agile principles, and she has worked with the core team to outline the new methodology. Wendy also has a good feel for what a test project should do to exercise all the new processes. The team sits down and compares the projects in the backlog to the selection criteria.

Smart Trading GroupMedia begins by screening projects by size. The team filters out the behavioral targeting and "My News" projects because they're estimated to run longer than 8 weeks. Next, they filter by project breadth. The Macromedia investigation is only an investigation—it will hit few areas and won't have a code deliverable. It's also a high-priority project, which means there will be little patience for testing a new process.

The autos email alert item is closer to a feature than a project. It's a small enhancement to the existing autos site. This project doesn't require a feature-card meeting and will have only one iteration. Wireless weather alerts is a niche project that requires a limited and specialized project team. It doesn't require team members from most of the departments.

That leaves Smart Trading Group with the free merchandise advertising project. It's estimated to last less than 8 weeks, which will allow for timely feedback. It will hit all the phases of the new process and involve all the major functional areas of a project: requirements, design, development, implementation, quality, and operations. It's a medium-priority project, so it will push the team along. It's also a good fit because third parties won't be involved.

The project is a product-management initiative, which means an internal product manager can play the role of the customer. This supports a pilot objective of not involving a real customer in the first test of the agile methodology. Based on all these findings, Smart Trading Group chooses the free merchandise advertising project as its pilot. Smart Trading Group's next step will be to identify the team members needed to perform the pilot.

Many companies struggle when trying to validate a project's value. Some companies initialize a project without knowing if it's viable; other companies scrutinize the value of a project for months before making a decision. There are issues with both approaches.

If you perform minimal validation, you'll frequently deliver projects that provide marginal value. You may also find that you're aborting on projects because you overlooked major risks at the outset. In both instances, you waste company time and resources and potentially lose the opportunity to deliver valuable projects.

Companies that perform too much validation have a different set of issues. These companies create so many hurdles and gateways that a considerable expense is associated with project justification. They also minimize their ability to achieve benefits from projects that need to deliver value early: time that could be spent performing the project is frequently lost to the justification cycle.

This process works for two reasons:

■ The feasibility effort is time-boxed.

■ The team is empowered to question the viability of the project after the Feasibility phase.

Time-boxing the effort prevents a runaway train. A time limit adds urgency to the effort and prevents waste. Smart Trading GroupMedia has a 3-day limit for feasibility work. We suggest you create a time limit for feasibility work within your company, too. A good rule of thumb is 2 to 5 days. Some employees won't be happy with this time limit: they will say that each project is different and that larger projects require more time for feasibility work.

They will also say that setting a fixed time for an activity is anti-agile. We agree with all these points. This is where our second point comes into play: the team can cancel the project at any time.

The agile process that Smart Trading Grouphas created is represented by five virtual phases. We use the term *virtual* because in reality you may perform feasibility, planning, development, or adapting at any point in an agile project. The work is not performed in a serial fashion.

First is the Feasibility phase. You use this phase to determine if an idea has enough merit to justify going forward with more detailed requirements, planning, funding, and staffing. Why are you doing this project? What is the value of this request? What are the risks in pursuing this project? The Feasibility phase provides answers to these questions quickly. You can see typical feasibility activities in figure 10.2.

The Planning phase gets started by reviewing the output of the Feasibility phase and going deeper into the information provided. You use the Planning phase to break the idea into discrete pieces of functionality called *features* or *user stories*. You then prioritize the features and loosely assign them to development iterations.

The release plan provides a first pass at the work that will be created, tested, and demonstrated during the Development phase. This work is completed in iterations and

queued for later deployment. Each iteration is a deliverable subset of features; these features are demonstrated at the end of development iterations.

When the team reviews the features that are delivered, they adapt. The team gathers feedback from the customer during the Adapt phase to ensure their needs were satisfied by the features delivered. The team also reviews their velocity (pace) to see if their capacity estimates are correct. The team uses this information to adjust the plan for the forthcoming iteration.

The last phase is Deployment, which begins after the last iteration is complete. You use this phase to deliver code to the production environment. You also use the

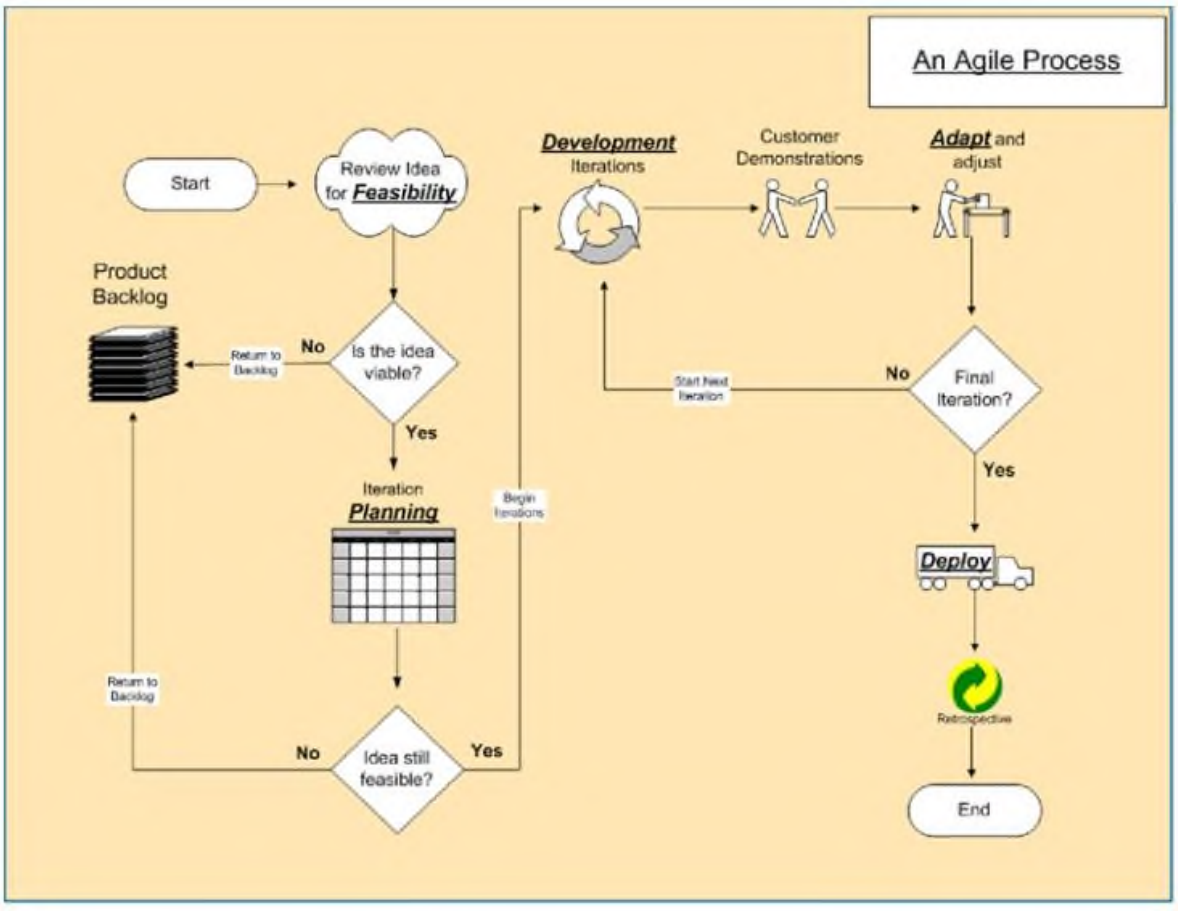


Figure 3.3 Overview of Smart Trading GroupMedia’s phases

Deployment phase to prepare all of those affected by delivery of the project: you train customers, prepare support organizations, turn on marketing plans, and complete the phase with a project retrospective. Smart Trading Group’s pilot project, free merchandise advertising (FMA), has been approved for feasibility research.

The FMA project met the conditions of the criteria: it's medium priority, can be completed within 8 weeks, and involves the majority of departments and areas within Acme. Now Smart Trading GroupMedia needs to identify a group of employees to look at the idea in more detail and make sure it's feasible to fund the project and identify a full project team.

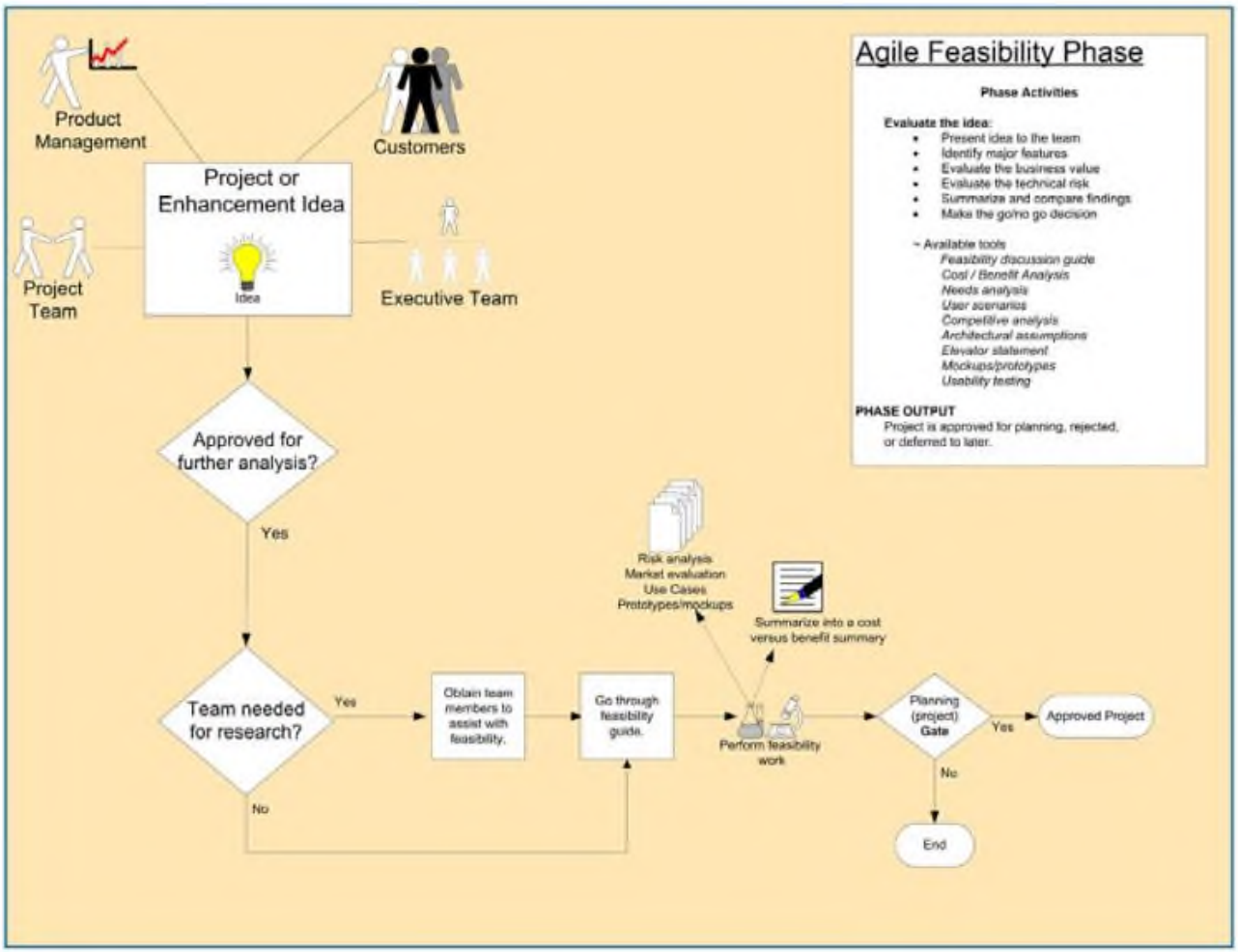


Figure 3.4 Smart Trading Group's Feasibility phase. The team gathers just enough information to validate the value of the project and make the call about whether to go forward

Deployment phase to prepare all of those affected by delivery of the project: you train customers, prepare support organizations, turn on marketing plans, and complete

the phase with a project retrospective. Smart Trading GroupMedia's pilot project, free merchandise advertising (FMA), has been approved for feasibility research.

The FMA project met the conditions of the criteria: it's medium priority, can be completed within 8 weeks, and involves the majority of departments and areas within Acme. Now Smart Trading GroupMedia needs to identify a group of employees to look at the idea in more detail and make sure it's feasible to fund the project and identify a full project team.

After a project has been endorsed for feasibility analysis, you assign an individual or a team to perform the work. The work can be performed by an individual if the idea is simple, such as a slight enhancement to existing functionality. More complicated ideas should be reviewed from various perspectives, and a team will be required to provide the analysis

The feasibility guide focuses the team on the areas they should be researching during the phase. When the team feels they're reaching diminishing returns, or their time limit expires, they will summarize their findings and present them to an approval body. The team will make a recommendation on "go/no go" at the conclusion of the exercise, and the approval body will make the ultimate decision about whether to proceed.

Note that an approval body can be as small as a product manager or as large as a steering committee or executive team. In our experience the feasibility team's suggestion is usually followed. Let's follow Smart Trading GroupMedia as it goes through the feasibility phase with its pilot project, beginning with the creation of a feasibility team.

Smart Trading Group's agile process allows for different teams to be used between the Feasibility phase and the Planning/Development phases. For example, a light team or even one person can do the feasibility analysis. When an idea is through the Feasibility phase, Smart Trading Groupcan identify the team for planning and development—the actual project team.

Smart Trading Group chose this design to minimize the impact on the organization until the project has been deemed viable. In the case of the pilot project,

Smart Trading Grouphas decided to name the team for planning and development during the Feasibility phase. Only a few team members receive feasibility assignments; the remaining team members sit in on feasibility activities to learn how the process works. This is the beginning of Acme's agile education process.

On projects after the pilot, Smart Trading Group will perform the Feasibility phase as designed, having only needed team members participate. When you create your lifecycle, you can go either way. If you can free up the entire team for the Feasibility phase, and the entire group is about eight people or fewer, then we recommend having the entire project team involved in feasibility activities. In contrast, you may choose to limit involvement to the minimum amount of employees needed to explore the project for value. Many companies do this because they have a limited number of team members.

If you're working on an agile project, one with volatile requirements or a tight deadline, you won't have deep requirements when you begin the Feasibility phase. A request or idea has been approved for a feasibility investigation, and no one has documented detailed requirements to this point. Your idea is relatively new, and you're working rapidly to either start the project or dismiss it. The information that is available is presented to the feasibility team. The idea usually has a champion or author who can meet with the team and go over the concept. This person brings in all the information they have at this point, whether it's a diagram on a cocktail napkin, a detailed flowchart, or a mini functional specification. The following items can be analyzed during the Feasibility phase: If your idea comes from within, you may have a white paper created by someone to outline the idea. In Smart Trading GroupMedia's case, the idea came from product management, which created a proposal for FMA.

Smart Trading Group starts its feasibility work with a meeting between Jay (the product manager) and the feasibility team.

When you create your own custom methodology, you'll determine the approval level required to pass from the Feasibility phase to the Planning phase. Final approval may come from the project team, or you may decide to have it come from a management group. It's logical to have different approval levels depending on the

scope of a project. For example, a project that can be completed in a few days with existing team members may be blessed by a product manager or project team. Conversely, a project that will take months to complete, with incremental expense, probably needs approval from a management group

Smart Trading Group has a team that is dedicated to their project, but it isn't unusual to have team members working on different projects at the same time. In many environments, executives want to know what types of skill sets are needed and how long you'll need those skill sets. In these instances, the output from your feasibility work should include a high-level estimate of the people you'll need for the project and approximately how long you'll use them. This is especially true if you believe you'll need help from outside the company; outside assistance may require additional funds that aren't budgeted for your team or the platform

After a project makes it through feasibility analysis, you select the team that will deliver the project. Ideally, you want to begin with your feasibility team and expand that group as needed. Feasibility-team members can share the information from the Feasibility phase with the additional team members; this will kick-start the project team. Assignments are based on the current estimated size of the project, the type of team members needed, and team member availability. If your company is small, you may have the same people work on every project. You may also have employees who are dedicated to a website or product within your company.

Smart Trading Group uses a resource pool for its projects. This provides flexibility and also increases the tribal knowledge throughout the team. There are specialists for each area, but team members may be assigned to any product or project depending on company need and the employee's desire to learn about a new area or technology.

When the pilot project is chosen, chances are that some core-team members will be on it due to their functional jobs. But the majority of pilot-team members probably won't be from the core team. You need to review your pilot-team roster and determine whether the team has enough core-team members to support mentoring and hand-holding during the pilot.

If the pilot team does not have enough mentoring, you should assign a few coreteam members to assist them. These core-team members must be present for all major meetings and check in daily with the team.

Table 3.6

The pilot project team for the Auctionator/Online Auction System. (OAS)

Pilot project role	Name	Background	Pilot project role
Project manager	*Wendy Johnson	Wendy is part of the core team. She looks forward to seeing how the project-manager role works in an agile environment.	
Developer	*Roy Williams	Roy is part of the core team.	Developer
Developer	Matt Lee	Matt usually supports development of applications for the Classifieds group.	Developer
UI	Ryan Getty	Ryan has a lot experience doing prototype work, which should bode well for the agile process.	UI
QA	Gina Wallace	Gina is curious to see how much she'll be involved in planning the pilot project. Her peer, Vijay Kumar, is on the core team, and	QA

		she'll ask Vijay for mentoring during the pilot	
Operations	Tom Klein	Tom works for Matt Shiler, who is on the core team. Tom has no agile experience.	Operations
Requirements	Rich Jenkins	Rich has spent a lot of time learning about agile from his peer on the core team, Wes Hunter.	Requirements
Architecture	Keith Gastaneau	Keith is part of the core team.	Architecture
Customer	Jay Fosberg	Jay is a product manager for the Classifieds group and proposed the concept of Free Merchandise Advertising. Jay will play the role of the customer for the pilot.	Customer

Smart Trading Group has three core-team members working on the pilot project. A good rule of thumb is to ensure that the pilot team includes two or more core-team members. Smart Trading Group also has a bonus in that one of the core-team members on the pilot is the project manager. She will be involved in almost every aspect of the project and available to provide mentoring to the pilot team. Jay Fosberg will be the proxy for the customer during the project, which isn't a stretch for him. Jay has always been a strong supporter of the customer and an advocate for application usability. After you identify the pilot project team, you need to train them and orient them on the process you're going to assess

The untrained pilot-team members go through a process similar to that followed by the core team. They need an overview of why the company is pursuing agile,

training on the agile principles, and an understanding of the process they're to use on the pilot. If possible, have your agile coach provide the pilot team's training, with core-team members contributing to the discussions. This process training takes one or two days and is slightly different for the pilot team. In addition to the fundamental principles and practices of agile, they also need training on how the core team has represented those principles in the custom methodology.

For example, they need to see how the core team's new process supports the agile principle of *customers and developers working together daily*. If core-team members are attending the fundamentals training with the pilot team, they can show the pilot team how the principles are reflected in the custom methodology.

You should expect the untrained team members to have some cynicism and negativity related to using the new process. Their concerns usually relate to the following:

■ *A misunderstanding of agile principles*—Common misconceptions

■ *A belief that the current process works fine*—Project team members may be unaware of the issues that the new methodology addresses.

■ *Lack of detail in the agile process*—In the past, you may have prescribed every step in the development process. Now, the team is asked to participate in selecting the processes that add the most value, and that can be a shock.

If your environment has been controlling in the past, the last item will take time to resolve. Agile doesn't assign employee A to do step B; it tells the employee to deliver value to the customer as quickly as possible and provides tools and processes to reach that goal. The team works together to determine logical steps and assignments during the project.

No matter what the feedback is, listen to the pilot team with an open mind. Where applicable, show them how the new design takes their concerns into account.

If team members appreciate the value of a project, it will increase the likelihood of their buy-in and support. If they don't believe in the value, they may not apply themselves, and they may undermine the project. You can provide clarity around the value by having the team work together to create an elevator statement.

An *elevator statement* allows you to condense the project concept into a short, compelling paragraph. The idea is that if you're asked about your project in an elevator on the bottom floor of a building, you should be able to describe the project in a concise, intriguing way before you reach the top. This is a great tool for communicating the value of the project to those outside the team, and it centralizes everyone on the benefits that should be delivered [9].

You begin the exercise by having the team review the known requirements and the desired deliverables. Then the team works together to answer the following questions:

- Who is the customer?
- What do they need?
- What is the category of the product or service?
- What are the most compelling benefits to the customer?
- Can you quantify the benefits?
- What differentiates your product from existing alternatives?

We also find it helpful to show the team a few examples of elevator statements to get them started.

User stories in Smart Trading Group. Some of the people we've worked with refer to feature cards as user stories on steroids. We believe this is an accurate description. Feature cards share the same goals as user stories. Smart Trading Group aren't looking for requirements; you want information to help you plan. Smart Trading Group aren't looking for formal requirements to review; you want to interact with the customer *verbally* to better understand their needs. Smart Trading Group also want to gather just enough information to understand the scope of the system.

User stories and feature cards collect conversations with customers. Feature cards also aim to represent a piece of work that can be completed within an iteration. And feature cards and user stories both collect the tests needed to verify a feature is complete.

The main thing that makes feature cards different is the additional fields for uncertainty, dependencies, and frequency of use. By adding these fields, you make it

easier for the team to prioritize and sequence features after the initial customer conversations.

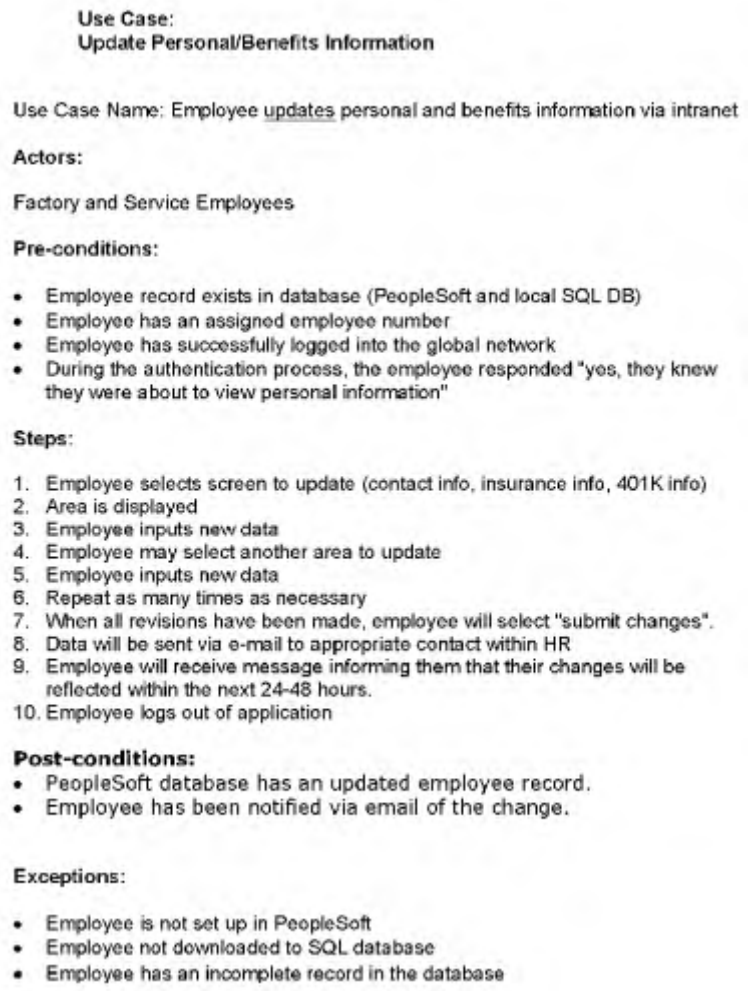
Use cases in Smart Trading Group. Use cases have been an essential requirements-gathering tool for many years. We've used them on many projects, and they can provide a good process for documenting a system's detailed requirements. But if use cases are used too early, they can create issues for a project:

■ A use case can imply technical-design details and bias the team toward implementation. You can easily lose sight of what the customer needs and begin rushing down the path to building the application.

■ A use case can define a large scope of functionality. You want to see all your features defined as pieces of work that can be completed within an iteration. Many times, a use case exceeds this timeframe.

The use case didn't have an extreme bias toward design details, but hints were starting to surface. Greg's team was already thinking about using PeopleSoft and a SQL database. This team also assumed there would be separate screens for each benefits area and that the customer would receive a change confirmation via email. Such assumptions may end up being correct, but they steer the customer away from a conversation about their needs and into a discussion of implementation details. It's important to note that there are two types of use cases: *essential* and *real*. An *essential* use case is closer to a feature card; the interaction listed is at a high level and isn't implementation specific [45].

A *real* use case describes the detailed interation with the system, naming screens, databases, triggers, and other system artifacts. Returning to the second issue with use cases, let's see how scope could become large for an Auctionator feature (see figure 3.5).



Use Case:
Update Personal/Benefits Information

Use Case Name: Employee updates personal and benefits information via intranet

Actors:

Factory and Service Employees

Pre-conditions:

- Employee record exists in database (PeopleSoft and local SQL DB)
- Employee has an assigned employee number
- Employee has successfully logged into the global network
- During the authentication process, the employee responded "yes, they knew they were about to view personal information"

Steps:

1. Employee selects screen to update (contact info, insurance info, 401K info)
2. Area is displayed
3. Employee inputs new data
4. Employee may select another area to update
5. Employee inputs new data
6. Repeat as many times as necessary
7. When all revisions have been made, employee will select "submit changes".
8. Data will be sent via e-mail to appropriate contact within HR
9. Employee will receive message informing them that their changes will be reflected within the next 24-48 hours.
10. Employee logs out of application

Post-conditions:

- PeopleSoft database has an updated employee record.
- Employee has been notified via email of the change.

Exceptions:

- Employee is not set up in PeopleSoft
- Employee not downloaded to SQL database
- Employee has an incomplete record in the database

Figure 3.5 A Smart Trading Group use case **[46]**

A Smart Trading Group use case can create issues for a project when it's used too early. The format can bias the team toward implementation planning versus trying to understand the true user or business need

The Smart Trading Group team could probably complete the main use case for *place a bid* within 10 days, but other use cases that could come from the exceptions probably wouldn't be completed within the same 10 days. Frequently, it takes more time to create an exception feature than it does to support the main, *perfect world* flow.

In conclusion, we think use cases are a great requirements-gathering tool, but they should be used in conjunction with feature cards or after feature cards are complete.

In our experience, a functional specification is a deep, detailed document that speaks to how a requirement will be delivered. Functional specifications frequently

include use cases, wireframes, interaction diagrams, formal business requirements, and entityrelationship diagrams.

Functional specifications are common in a waterfall environment. We usually see the following flow around a functional specification:

1 A business-requirements or marketing-requirement document is created.

2 A functional specification is created from the business-requirement document.

3 A technical design document is created from the functional specification.

4 A test plan is created from the functional specification.

The process can be formal, and each document is created in series. In some cases, the customer may not be consulted as the documents are being created; this process is common with fixed-bid work. The team is trying to deliver to a requirement document and to use statements such as "The system shall…." If the customer did not detail their requirements correctly, you don't care—you get paid as long as you deliver to their specifications. This approach is different from the agile mentality of learning as you go and engaging in frequent customer interaction.

If you contrast a feature card to a functional specification (FSP), you'll notice the following differences:

■ A feature card starts a requirements conversation. An FSP tries to cover all requirements immediately.

■ A feature card is used to record conversations. An FSP doesn't include conversations but focuses on documenting how a documented business requirement will be met.

■ A feature card focuses on verbal communication, common understanding, and synchronizing the team on the customer goals. An FSP focuses on documenting the functional details and having team members read the FSP to understand what they should do.

■ A feature card focuses on gathering just enough information to prioritize, sequence, and estimate the work. Note that we don't see an issue with creating functional specifications, but we *do* see issues with the process that usually surrounds

them. In our experience, we've witnessed four weaknesses with the processes typically used with a formal FSP:

■ A functional specification focuses on delivering what was requested at the beginning. In our experience, what the customer wants changes as they see the product demonstrated. Feature cards begin the process of identifying what is needed at the end, not the beginning.

■ A functional specification can position the customer as an enemy, with definitive statements such as *the system shall*. The customer is also somewhat inhuman when their needs are presented on paper versus via a face-to-face conversation.

■ The process around functional specifications can delay the ability to get early estimates for the project. An FSP may take weeks or months to complete, and then it's passed to developers for technical design and ultimately development estimates. It may take months to get an estimate for project duration.

■ When estimates do come in, you may realize that you've completed functional specifications for features that you won't have time to complete. This FSP work will be wasted effort. There is value in FSPs when you work with offshore resources, or to help you support traceability requirements. It's also great to have a document that holds all the information about a feature in one place, especially if you don't have a dedicated team room or a place to hold your whiteboard diagrams and flow. We've seen some teams take pictures from their whiteboard-modeling discussions and store them in the FSP.

This is a great idea because team members usually understand diagrams better than requirements statements. The main point is that you don't want to start your initial planning process by creating detailed functional specifications.

Estimating software in A Smart Trading Group is a mystery for most teams. Teams can spend huge amounts of time breaking down features to create their estimates, but the actual time needed is usually a vastly different number. The issue lies in two areas: techniques and expectations.

Most teams use traditional estimation and capacity-planning techniques. Traditional techniques are dependent on constants and repetitive work. A traditional

planning process wants to know how much time it takes to build a widget, how many machines are available to build the widgets, and how many hours a day the machines can be used for building the widgets [15].

As you probably know, each piece of software is unique, and it's difficult to estimate something that is being built for the first time. We never build the same widget twice. It's also hard to treat a developer like a machine and predict their output on a daily basis. Communicating this to sponsors and stakeholders is also challenging; many experienced software professionals still believe incorrect estimates are more closely tied to incompetence than to the realities of software development. Agile estimation techniques won't remove uncertainty from your early estimates, but they will improve your accuracy as the project proceeds. This is true because agile estimation methods take actual work into account as the project progresses. Your work mix may be diverse, but if you measure at an aggregate level you can still identify an average that you can use for estimating your capacity. We'll demonstrate this process as we follow the Auctionator through its development iterations.

In the early stages of a project, someone guesses how long it will take to deliver. This person may be a salesperson, project manager, or development manager. They may make a guess based on their experience, or they may have some quick chats with seasoned employees and solicit their opinions. When the timeline guess is in place, the project begins. If the project is related to a product, there may be marketing requirements to reference. If the project is for a customer, there may be a statement of work to reference. In either case, it's common for an analyst team to convert the information into functional specifications. After the functional specifications are completed, a conversation begins with the development team, designs begin to evolve, and some teams may document a technical design and architectural plan. When this work is complete, the development team provides estimates based on the anticipated approach. The team also estimates their capacity by resource type. Then the estimates, capacity, and known dependencies are entered into a project plan. At this point, the team has a schedule that they feel confident in, and they share it with the stakeholders. This exercise may take several weeks or months to complete. If a project is timeboxed,

the team may find that there isn't enough time to deliver all the features forwhich they created functional specifications, designs, and estimates. The team then has to scope back the features for the project to meet the timeline, realizing they've wasted valuable time in estimating features that won't be pursued [46].

Agile estimation techniques address the shortcomings of this method. You don't design and estimate all your features until there has been a level of prioritization and you're sure the features are needed. You used a phased approach to estimation, recognizing that you can be more certain as the project progresses and you learn more about the features.

At a high level, the phased process looks like this:

1 Estimate the features in a short, time-boxed exercise during which you estimate feature size, not duration.

2 Use feature size to assign features to iterations and create a release plan.

3 Break down the features you assigned to the first iteration. *Breaking down* means identifying the specific tasks needed to build the features and estimating the hours required. 4 Re-estimate on a daily basis during an iteration, estimating the time remaining on open tasks. Agile estimating is also different in that you involve the entire team in the estimation process. Let's take a moment to look at the value of whole-team estimation.

The first thing the Smart Trading Group team needs to do is establish two reference points for all features. They do this by identifying a feature that is 2 story points in size and a feature that is 5 story points in size. After a review of the features, the Smart Trading Group team concludes that *Search by category* is 2 story points and *Receive help online* is 5 story points; see table 3.6.

Smart Trading Group's team then reviews all the features against *Search by category* and *Receive help online* to determine if the other features are the same size, smaller, or larger. As additional features are estimated, they're also used as reference points to compare the nonestimated features.

After the Smart Trading Group team completes the planning-poker exercise, they have a prioritized, estimated product backlog. Now the question becomes, how many

features can they complete within the project timeline? We'll rejoin Smart Trading GroupMedia to see how they answer this question.

Table 3.7

Story points let you evaluate capacity and throughput without performing detailed task analysis in advance [78]

ID	Feature name (ability to)	Story points
5	Register on the site	3
4	Place an item up for bid	3
10	Bid on an item	3
17	Auction engine	8
13	**Search by category**	**2**
16	Purchase an item immediately	2
1	Flag problem postings	2
6	Contact the seller	3
2	Create alerts for item type	3
9	**Receive help online**	**5**
11	Record seller feedback	5
12	View seller information	2
14	Perform advanced search	8
3	Email a friend	2

7	Customize my view	8
15	Retract a bid	2

Extreme Programming (XP) environments frequently have iterations of 1 to 2 weeks in length. Scrum teams like to do a sprint/ iteration every 30 days. More than likely, a number between 1 and 4 weeks will work for your environment.

We suggest that you start with 2-week iterations and see how that timeframe works for you. It may be good to use this 2-week iteration length for several projects before making the call on whether it's successful. If you find that your features are too large to complete in 2 weeks, you can examine your features to see if you've broken them down to their true, essential requirements; alternatively, you can try a longer iteration length. Smart Trading GroupMedia has followed this advice and created its release plan assuming 2- week iterations.

Smart Trading Group has only two iterations in its pilot project, which is a minor change from the company's previous development process. In theory, you want more iterations so you can demonstrate and react to new information sooner. But in the case of a pilot project, two iterations are fine. The team is just learning agile techniques, and they can increase the number of iterations on subsequent projects [15].

When Smart Trading Group completes detailed planning for iteration 1, the team finds that they've assigned 19 story points into the iteration. For now, 20 points will be used as the capacity number, so they plan iteration 2 to hold 20 story points. Smart Trading Group's features were prioritized, grouped, and estimated remaining is to load up each iteration with 20 story points.

Auctionator Release Plan						
Iteration number:	**Iteration 0**	**Iteration 1**	**Adapt week**	**Iteration 2**	**Adapt week and deployment preparation**	**Go live**
Iteration dates:	4/23–4/27	4/30–5/11	5/14–5/18	5/21–6/1	6/4–6/14	6/15
Iteration theme:	Initial architecture modeling Prepare development environment Test with vendors	Core, critical functionality coded	Iteration retrospective Technical reviews Usability testing Acceptance testing	Create and integrate secondary features	Iteration retrospective Technical reviews Usability testing Acceptance testing	Go live; deploy to productio n
Features						
API setup with eLertz	X					
Register on the site		X(3)				
Place an item up for bid		X(3)				
Bid on an item		X(3)				
Auction engine		X(8)				
Search by category		X(2)				
Purchase an item immediately				X(2)		
Flag problem postings				X(2)		
Contact the seller				X(3)		
Create alerts for item type				X(3)		
Receive help online				X(5)		
Record seller feedback				X(3)		
View seller information				X(2)		
Sideline (features that can come in if others go out, currently at the top of the backlog)						
Perform Advanced search						
Email a friend						
Customize my view						

Figure 3.7 The completed release plan for the Auctionator project **[38]**

The numbers in parentheses represent story-point estimates. Smart Trading GroupMedia has estimated its story-point capacity at 20 points per iteration. After loading, iteration 1 holds 19 points, and iteration 2 contains 20 points.

Smart Trading Group has involved the entire project team in creation of the release plan, so the team is up to speed on why the project is being pursued, the overall timeline, and the features assigned to each iteration. Smart Trading Groupstill needs to

bring other groups on board to make sure the project is supported and to create awareness about support that may be needed.

The first objective of the kickoff meeting is to bring stakeholders and sponsors up to speed. The project team shares the information gathered during the feasibility and chartering exercises, including scope, benefits, key dependencies, constraints, risks, and the release schedule. In a traditional environment, the presentation may be performed by a project manager or development manager. In an agile environment, you should try to get as many team members to present as feel comfortable doing so. At Smart Trading GroupMedia's kickoff, four team members present: Wendy, the project manager, Jay, the customer, Gina, the tester, and Roy, the developer. A second objective of the kickoff meeting, and perhaps the most important, is to bring support groups up to speed so they can see when their help may be needed. Some of the support areas typically discussed during a kickoff meeting are as follows:

■ *Operations*—These teams will support your application once it's deployed, and they need to know what type of maintenance will be required to keep the application working correctly.

■ *Security*—In larger companies, your application may need to be reviewed to make sure it complies with corporate standards.

■ *Load testing*—In larger companies, you may need to reserve load-testing equipment.

■ *Load balancing*—You may have specialized groups that manage load-balancing environments, and you'll need their support for your project.

■ *Hardware and storage*—If you're doing a project that requires new equipment, you may need help from hardware teams.

■ *Documentation*—If your project will require supporting documentation, you may want to invite documentation teams to your kickoff.

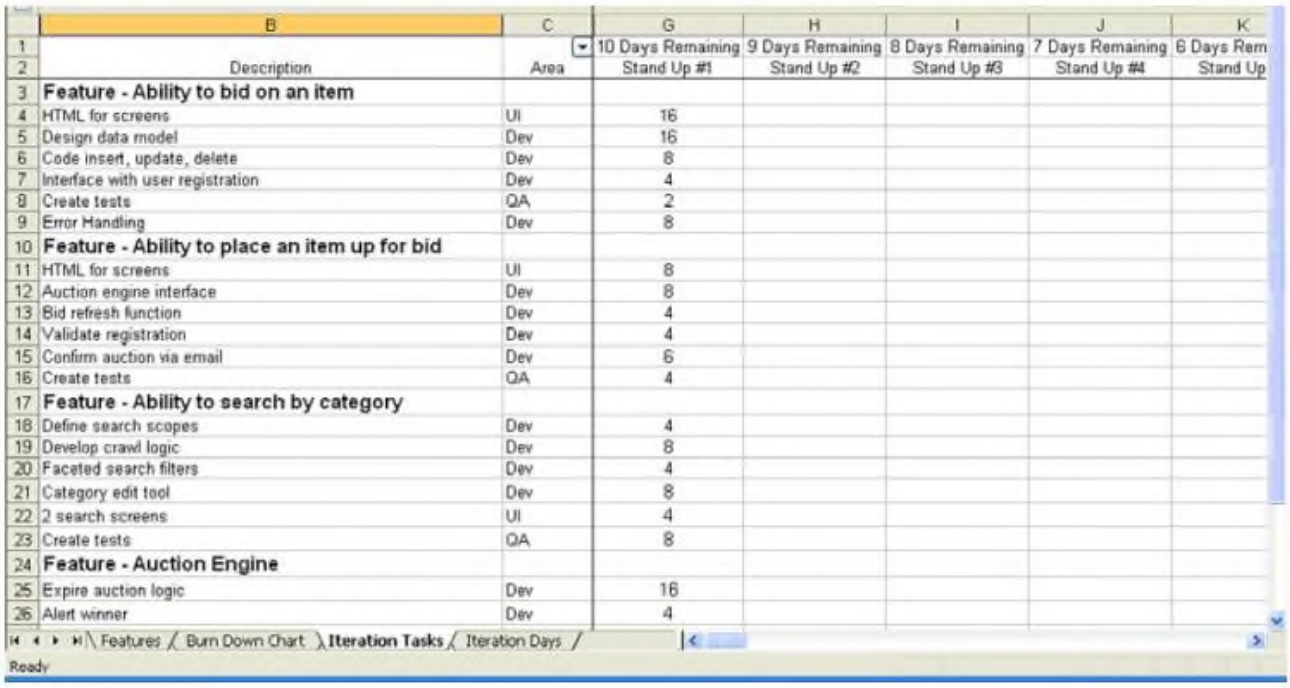
■ *Marketing*—If you need to do public announcements or advertising, you must bring this team up to speed with your release plan.

■ *Training*—If your project will require training for employees or customers, you should invite this team to the kickoff.

The Smart Trading GroupMedia team loads the iteration plan into a tool called a *burn down chart* that lets them view the iteration plan and present it to stakeholders and other parties who may not be on-site or have easy access to the team work area (see figure 16.9).

You may also find that your iteration becomes complex due to dependencies and that it's hard to keep track of all the work using an iteration wall or a burn down chart alone. In those instances, you can use tools such as Microsoft Project. Some people believe that if you're using a tool like Microsoft Project, you aren't agile.

Many people believe that tools like Project imply formality and overhead, and they're good only for traditional projects. We can tell you that this definitely isn't true. It isn't the tool that takes away agility but the way the tool is used. Let's look at an example. As we've mentioned, Greg is a project manager, and his team releases new software every 8 weeks. Greg typically uses tools such as an iteration wall and burn down charts. But his team noted that on some iterations, it was hard to keep track of dependencies.



	B	C	G	H	I	J	K
1			10 Days Remaining	9 Days Remaining	8 Days Remaining	7 Days Remaining	6 Days Rem
2	Description	Area	Stand Up #1	Stand Up #2	Stand Up #3	Stand Up #4	Stand Up
3	**Feature - Ability to bid on an item**						
4	HTML for screens	UI	16				
5	Design data model	Dev	16				
6	Code insert, update, delete	Dev	8				
7	Interface with user registration	Dev	4				
8	Create tests	QA	2				
9	Error Handling	Dev	8				
10	**Feature - Ability to place an item up for bid**						
11	HTML for screens	UI	8				
12	Auction engine interface	Dev	8				
13	Bid refresh function	Dev	4				
14	Validate registration	Dev	4				
15	Confirm auction via email	Dev	6				
16	Create tests	QA	4				
17	**Feature - Ability to search by category**						
18	Define search scopes	Dev	4				
19	Develop crawl logic	Dev	8				
20	Faceted search filters	Dev	4				
21	Category edit tool	Dev	8				
22	2 search screens	UI	4				
23	Create tests	QA	8				
24	**Feature - Auction Engine**						
25	Expire auction logic	Dev	16				
26	Alert winner	Dev	4				

Features / Burn Down Chart / Iteration Tasks / Iteration Days /

Ready

Figure 3.8 Smart Trading Group enters its iteration plan into a burndown chart **[37]**

A Smart Trading Group lets them track the work remaining as the iteration progresses. An electronic tool makes it easier to share status with the rest of the company or the customer.

Project teams have struggled with these issues for years. They frequently prepare to deploy code and at the last second realize something is missing. Let's delve into these two issues and see how Smart Trading Group simplifies status measurement and makes status transparent to the team

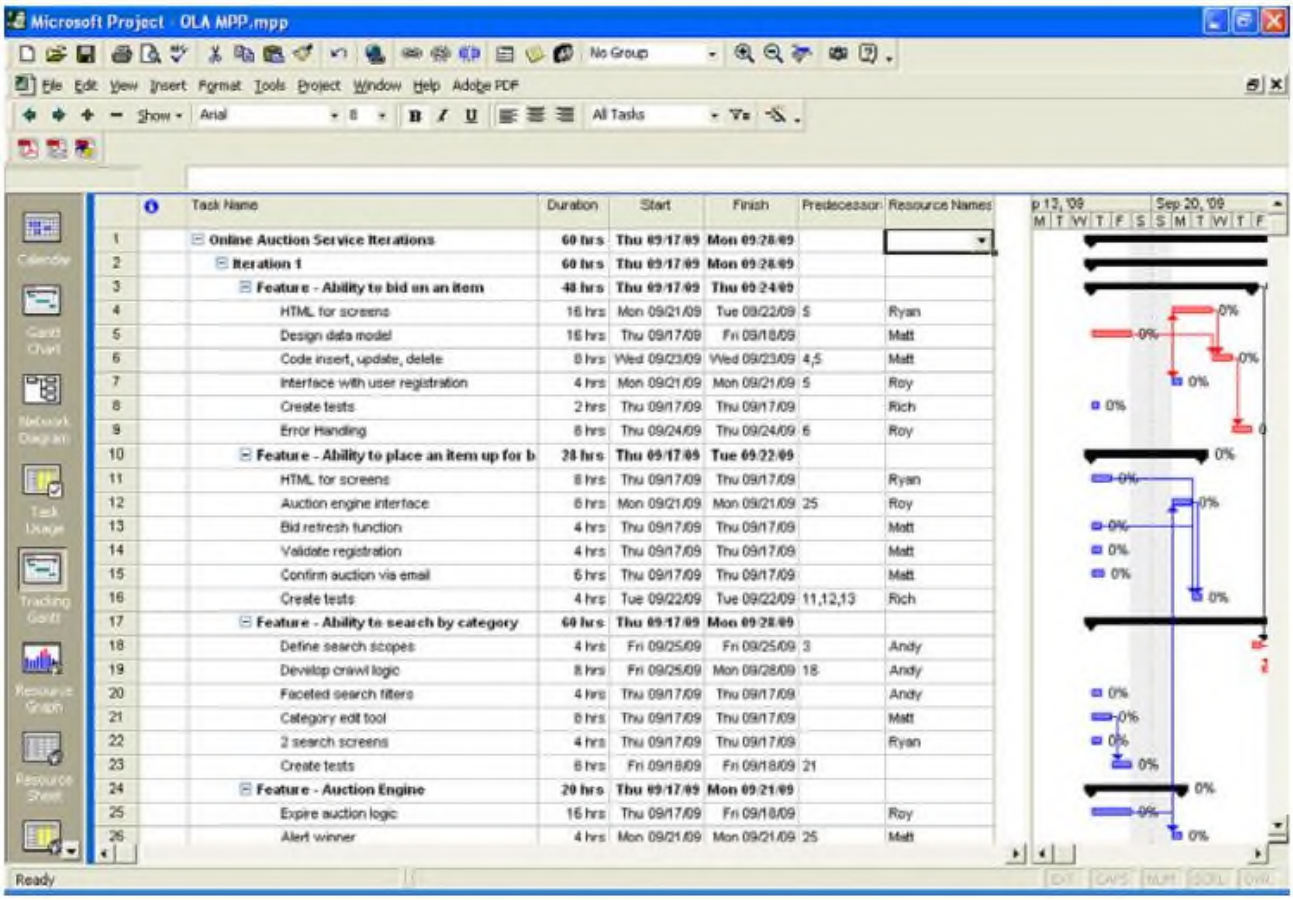


Figure 3.9 Microsoft Project relation **[49]**

A Smart Trading Group use case isn't usually viewed as a tool used by agile teams, but it can help with agility when an iteration becomes complex or team members need to anticipate how much work they must deliver. The key to using a tool is to make sure everyone knows that assignments are tentative and to ensure that the plan is highly visible and updated daily.

As we mentioned, Smart Trading Group has entered its iteration tasks into a burn down chart, which projects how much work should be completed as the iteration progresses. On a day-by-day basis, the team can see if they're on track, running behind, or running ahead of schedule.

Smart Trading Group estimates 152 hours of tasks to complete for the iteration. The burn down chart shows that this number needs to be down to 138 hours of work after day 1, 120 hours of work after day 2, and so on.

On the last day, the team should have no hours of work remaining. Software development doesn't care if it's being tracked in a nice linear chart. In reality, the work comes in surges, with the team sometimes stuck on an issue or problem and not making any progress. Figure 3.10 shows Smart Trading Group's burn down chart after 7 days of work. Many teams collect the estimates for remaining hours of work at their daily stand-up meetings.

This is where the *days remaining* line comes from in the chart. Day 1 represents the task estimates before work has begun, and day 2 represents the estimates after the first stand-up meeting. The Smart Trading Group team has started work, and now they believe the work is larger than originally estimated. What was originally estimated at 152 hours of work is now estimated to be 160 hours of work.

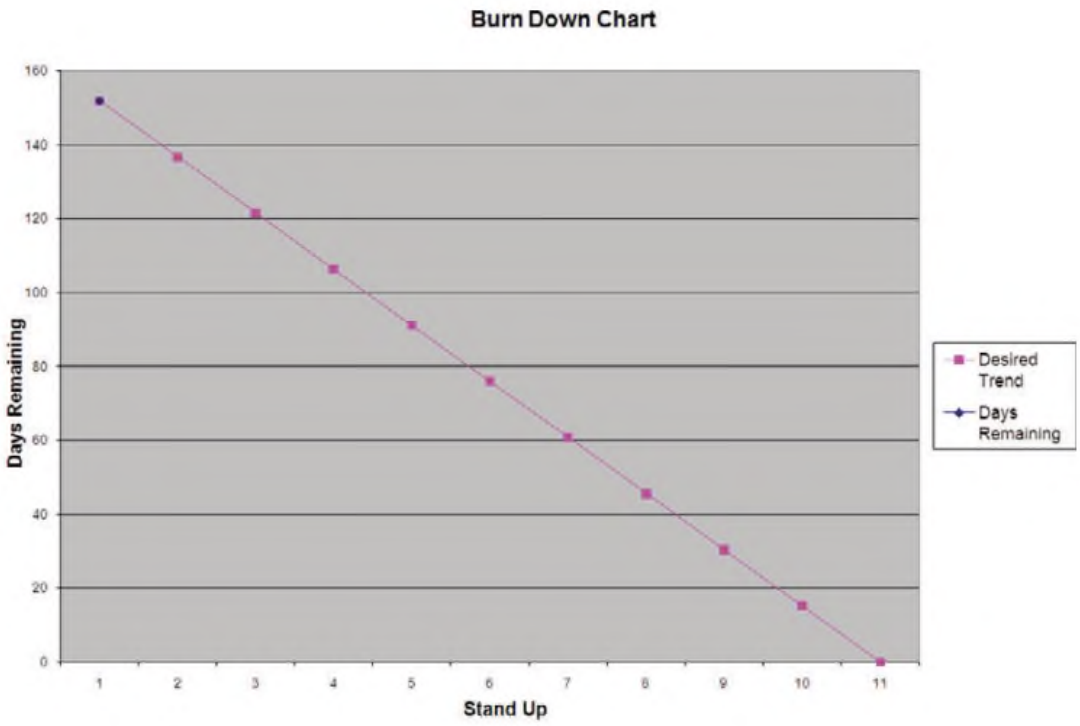


Figure 3.10 A burndown chart **[45]**

A burndown chart tells you how many hours of work you should have remaining as the iteration progresses.

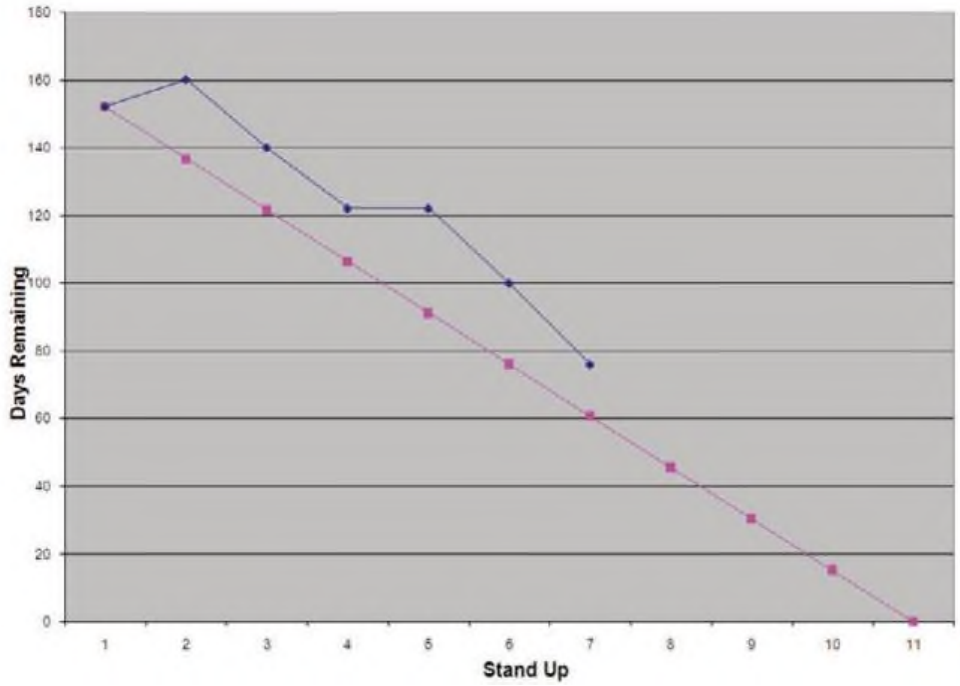


Figure 3.11 As the iteration progresses Smart Trading Group **[87]**

Iteration progresses sees status on a daily basis. In this example they're running behind after seven days of work, but they're trending to get back on schedule.

This is a common occurrence: after your team gets deep into the code, they may encounter surprises and need to increase their estimates for hours of work remaining for some tasks. They will also discover new tasks that you'll need to record into your burn down chart. The positive here is that some tasks will also be easier than expected, which will help your team stay on track toward completing the work within the iteration. Your team will also appreciate knowing the status of the iteration on a daily basis.

Smart Trading Group's situation also illustrates another common occurrence: as the team progresses from day 4 to 5, the amount of work remaining doesn't change. This happens when a team gets stuck on an issue. The team may be investigating options or doing technical research, so no code is created during this time.

A pilot project basically maps to the first stage (the innovators; see figure 23.3). The majority has gone through the process of conducting a pilot project at Smart Trading GroupMedia and at your organization.

CONCLUSIONS

The numbers in parentheses represent story-point estimates. Smart Trading Group has estimated its story-point capacity at 20 points per iteration. After loading, iteration 1 holds 19 points, and iteration 2 contains 20 points.

Smart Trading Group has involved the entire project team in creation of the release plan, so the team is up to speed on why the project is being pursued, the overall timeline, and the features assigned to each iteration. Smart Trading Groupstill needs to bring other groups on board to make sure the project is supported and to create awareness about support that may be needed.

Almost every agile pilot has these issues: the process may be slower, the team struggles to normalize around the process, and agile cynics find reasons to stop using agile.

Almost every agile pilot provides the benefit of letting the team experience agile and start understanding what the principles are all about.

If you work for a small company, the pilot may be your last test step before going live with agile across your company. Your goals will focus on maturing and on adding more agile practices as you mature. You can do this with the help of an agile coach.

Smart Trading Group’s development team has never enjoyed change requests. Whenever the customer asks for a change, team members go back to their desks and mumble about “scope creep.” After their agile training, team members have a new perspective. Let’s look at an example. If a customer asks for an alarm clock, and during a demonstration they notice that they forgot to ask for a snooze button, then they have experienced a natural occurrence in software development. Sometimes you miss a feature detail during a requirements discussion, and a demonstration exposes what was overlooked. This type of change should be embraced.

On the other hand, if a customer asks for an alarm clock, and after a demonstration they request that the alarm clock also be usable as a coffeemaker, then you have a radical change in requirements and potentially a new project. Some might call this scope creep, but we think the issue ties to the need to ask “why?” during the

featurecard exercise. If the root business need is known, it's doubtful that a major functionality shift like this will take place during development.

Smart Trading Group's new development model has a formal adapt phase between iterations, during which the team demonstrates to the customer. These demonstrations will occur every 2 weeks. The team has also adopted an approach of surfacing work in advance of the adapt window if pieces are ready early or if the feedback will help them make design decisions.

During development iterations, the team needs to be 100 percent focused on the work at hand. Everyone must be available to clarify requirements and to collaborate on and solve solutions.

Smart Trading Group's development team works on projects and maintains the three existing websites. In addition, they have daily maintenance activities, and they frequently work on mini-enhancements that don't require a project team. These supporting activities have contributed to Smart Trading Group's failure to deliver in the past. The development team must continue to perform their support activities, but in their new development model they will try to defer maintenance and support activities until the adapt week. During development, they will be interrupted only for critical production issues. Smart Trading Group also realized that the company had become people dependent in several areas. Experts existed for every area, and developers didn't cross-train with each other.

When a production problem occurred, frequently only one person had the skills to work the issue. Smart Trading Group identified this problem when the company outlined its new agile process and created a plan to cross-train the developers. The cross-training will spread skills throughout the team and let the developer with the most free time address the issue, which will minimize project interruptions

Smart Trading Group estimates 152 hours of tasks to complete for the iteration. The burn down chart shows that this number needs to be down to 138 hours of work after day 1, 120 hours of work after day 2, and so on.

Smart Trading Group's situation also illustrates another common occurrence: as the team progresses from day 4 to 5, the amount of work remaining doesn't change.

This happens when a team gets stuck on an issue. The team may be investigating options or doing technical research, so no code is created during this time.

A pilot project basically maps to the first stage. The majority of this has gone through the process of conducting a pilot project at Smart Trading GroupMedia and at your organization.

In the past, Smart Trading Group sought feedback from the customer after development was complete. Using this approach, Smart Trading Group's team frequently found the customer had functionality issues with the code that was developed. The issues went beyond bugs: they frequently tied to the customer saying the code didn't meet their needs. Smart Trading Group tried to address as many issues as possible before going live, and if major issues couldn't be fixed quickly, they delayed shipping the product.

In Smart Trading Group's previous process, the business analysts worked with the customer to document requirements. Frequently, an internal product manager communicated the needs. After the requirements were documented, the analyst passed them to the developer and determined whether there were any questions.

In Smart Trading Group's new process, the developers have already had direct interaction with the customer. The entire team participated in the feature-card exercise, and the developers gained clarity on some customer needs by asking questions directly. The customer (in this case, the product manager) will also be available during the development iterations and can clarify their needs and be consulted if issues are encountered.

If the Smart Trading Group team decided to track their pilot project in a project tool, they would say that the project is around 50 percent complete.

REFERENCES

1. 31. Palmer S. R., Felsing J. M. Practical Guide to Feature-Driven Development // InformIT. URL: http://www.informit.com/articles/article.aspx?p=26059 (30.05.15).
2. A Guide to the project management body of knowledge (PMBoK guide). Sixth Edition – USA: PMI Inc., 537 p. (2017).
3. A Guide to the project management body of knowledge (PMBoK guide). Sixth Edition – USA :PMI Inc., 2017. – 537 p.
4. Abbas N., Gravell A.M., Wills G.B. Historical roots of agile methods: Where did «Agile thinking» come from? : Springer Verlag, 2008. 94-103 c.
5. Anup Hande, Sunita Suralkar, P.M.Chawan Distributed Software Project Development. RL: http://www.ijera.com/papers/Vol2_issue3/FT239981003.pdf (30.05.15).
6. Archibald R. High-tech management/ Russell D. Archibald; trans. with English. Mamontov EV; under. ed. Bazhenova AD, Arefyeva A.O. - 3rd ed., Remaking. and ext. - M .: ITI Company; DMK Press, 2004. - 472 s,
7. Arzamastsev, A., Fabrikantov, O., Zenkova, N., Belousov, N.: Optimizatsiya formul dlya rascheta IOL [Optimization of formulae FOR intraocular lenses calculating]. Vestnik Tambovskogo universiteta. Seriya Estestvennye i tekhnicheskie nauki – Tambov University Reports. Series: Natural and Technical Sciences, vol. 21, no. 1, pp. 208-213. (2016).
8. Baird A., Riggins F.J. Planning and sprinting: Use of a hybrid project use of a hybrid project use of a hybrid project management methodology within a CIS capstone course // J. Inf. Syst. Educ. 2012. T. 23. № 3. C. 243–258.
9. Biloshchytskyi A.., Kuchansky A., Andrashko Yu., Biloshchytska S.. “A method for the identification of scientists research areas based on a cluster analysis of scientific publications,” Eastern-European Journal of Enterprise Technologies. No.5.,Vol. 2., Issue 89, pp. 4-10, 2017.

10. Boehm B., Turner R. Using risk to balance agile and plan-driven methods // Computer. 2003. T. 36. № 6. C. 57–66.
11. Bushev S.D. Creative technologies for project and program management: Monograph. / Bushueva NS, Babaev IA, Yakovenko VB, Grisha EV, Dzyuba SV, Voitenko AS / - K .: Summit-Book, 2010. - 768 with.
12. Bushev S.D. Morozov V.V. Dynamic leadership in project management. Tutorial. K .: Business Ukraine 2002, 310s
13. Bushev SD, Morozov VV Project Procurement Management. K .: Ukr. INTEI, 1999. Vol.1,2– 188s., 195s. 24. Shapoval M.I. Quality Management Kiev: Knowledge, 2006. - 472s.
14. Bushev SD, Project management: the basics of prof. knowledge and competence assessment system project. managers (National Competence Baseline, NCB UA Version 3.1) / S.D. Bushev, NS Busheva. - Ed. 2nd. - K .: IRIDIUM, 2010. - 208 p.
15. Bushueva N., Models and Methods for Proactive Management of Organizational Development Programs. Monograph. Scientific World, 2007, 199p.
16. Bushuyev, S., Burkov, V.: Resources Management for distributed projects and programs. Monograph, Nikolayev: Publisher Torubara V.V., 338 p. (2015).
17. Bushuyeva, N.: Models and Methods for Proactive Management of Organizational
18. Charles G. Cobb. Making Sense of Agile Project Management: Balancing Control and Agility, by Wiley, 2011. – 265 p.
19. Chatfild C., Johnson T., Microsoft Project 2016 Step by Step. Amazon Digital Services LLC, 2016, 577 p.
20. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Available: http://www.sciencedirect.com/science/article/pii/S0167739X08001957, last accessed 2019/02/21.

21. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, [Online]. Available: http://www.sciencedirect.com/ science/article/pii/S0167739X08001957
22. Cloud Terminology – Key Definitions, Available: https://www.getfilecloud.com/cloud-terminology-glossary, last accessed 2019/02/21.
23. Cockburn A. Agile Software Development. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. 5. Cohen D., Lindvall M., Costa P. An Introduction to Agile Methods. , 2004. 341 p.
24. Cohn M. Succeeding with Agile: Software Development Using Scrum. 2008. P. 53–81.
25. Coplien J.O., Harrison N.B. Organisational patterns of agile software development. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2004. 432 p.
26. Daniel D.R. Management information crisis // Harvard Bus Rev. 1961. № September-October. C. 111–121.
27. Demarco T. Deadline. Project Management Novel [Text] / Tom DeMarco. - M.: Top, 2006. - 288 p.
28. DeMarco, T., Lister, T.: Waltzing with bears. Risk management in software development projects. - M.: Izd. "Company p.m.Office", 196 p. (2005).
29. Development Programs. Monograph. Scientific World, 199 p. (2007).
30. Doug Carlo. Extreme project management. - M .: Company p.m. office, 2007. - 588 p.
31. Faulkner W., Badurdeen F. Sustainable Value Stream Mapping (Sus-VSM): Methodology to visualize and assess manufacturing sustainability performance // Journal of Cleaner Production. 2014. T. 85. p. 8–18.
32. Fernandez D.J., Fernandez J.D. Agile project management - Agilism versus traditional approaches // J. Comput. Inf. Syst. 2008. T. 49. № 2. C. 10–17.

33. Free ITIL, v.3, Available: http://www.wikiitil.ru/books/2015_Free_ITIL.pdf, last accessed 2019/03/10.
34. Furht, B., Escalante, A.: Handbook of Cloud Computing. Available: https://link.springer.com/book/10.1007/978-1-4419-6524-0, last accessed 2019/02/21.
35. Gabrielle Benefield. Rolling out Agile in a Large Enterprise, HICSS '08 Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences, 2008.
36. Garaedagi, D.: System thinking. How to manage chaos and complex processes. Platform for modeling business architecture, Grevtsov Buks (Grevtsov Publicher). 480 p. (2011).
37. Gary Chin. Agile Project Management: How to Succeed in the Face of Changing Project Requirements, by AMACOM, 2004. – 229 p.
38. Gogunskii V., Kolesnikova K., Lukianov D. “Lifelong learning is a new paradigm of personnel training in enterprises”, Eastern-European Journal of Enterprise Technologies. № 4/2 (82). – pp. 4–10. 2016.
39. Gontoryova IV Project Management Kharkiv: KNEU, 2007. - 348p. 10.Tarasyuk G.M. Project Management Kiev: Caravel, 2009. - 320p.
40. Grashina M., Duncan V. Fundamentals of project management / M. Grashina, V. Duncan. - St. Petersburg: Peter, 2006. - 208 p.
41. Hofstede G. Cultural dimensions for project management // International Journal of Project Management. 1983. Т. 1. № 1. С. 41–48. 16. Livermore J.A. Factors that significantly impact the implementation of an agile software development methodology // Journal of Software. 2008. Т. 3. № 4. С. 31–36.
42. ICB CompetenceBaseline, IPMA, 2006 - 112p.
43. International Project Management Association. Individual Competence Baseline Version 4.0. International Project Management Association, 432 p. (2015).
44. International Project Management Association. Individual Competence Baseline Version 4.0.International Project Management Association,2015, 432 p.

45. James P. Lewis. Fundamentals of Project Management (3rd Edition) by AMACOM Books, 2006. - 176 p.
46. Kalnichenko OV, Morozov VV Development of project concept. Methodical instructions for the course work. - K .: Taras Shevchenko National University, 2015. - 42 p. (Electronic access: file: /// C: / Users / HOME / Downloads /% D0% 9C% D0% B5% D1% 82_% D0% 9A% D0% A 0_% D0% BA% D0% BE% D0% BD% D1% 86% D0% B5% D0% BF% D1% 86% D1% 96% D1% 8F_% D0% 9A% D0% 9D% D0% A3_2210% 20 (4) .pdf).
47. Keith Locker, James Gordon. Project management. Degrees of higher skill. - M .: Grevtsov Publisher, 2008. - 352 p.
48. Ken Schwaber, Jeff Sutherland, “The Scrum Guide”, 2011. – 16 p.
49. Kosyakov, M.: Introduction to distributed computing. SPb: Public Research Institute IST, 155 p. (in Russian) (2014).
50. Ladas C. Scrumban - Essays on Kanban Systems for Lean Software Development // Lean Software Engineering: сообщество специалистов по программной инженерии, практикующих методологию Lean. URL: http://leansoftwareengineering.com/ksse/scrum-ban/ (30.05.15).
51. Louise Ledbrook, “Waterfall Project Management: An Overview”, 2012, http://projectcommunityonline.com/waterfall-project-management-anoverview.html
52. Maimon O., Rokach L., Data Mining and Knowledge Discovery Handbook, 2005, [Online]. Available: http://www .bookmetrix.com/detail/book/ae1ad394-f821-4df2-9cc4-cbf8b93edf40
53. Maimon, O., Rokach, L. : Data Mining and Knowledge Discovery Handbook, 2005, Available: http://www.bookmetrix.com/detail/book/ae1ad394-f821-4df2-9cc4-cbf8b93edf40, last accessed 2019/03/11.
54. Management of projects, programs and project-oriented business: Collective monograph. Volume 3 / V.V. Morozov, E.D. Kuznetsov, OB Danchenko and others; for science. ed. VV Morozov - K .: University of Economics and Law "KROK", 2012. - 238 p.

55. Mannaro K. Adopting Agile Methodologies in Distributed Software Development. URL: http://veprints.unica.it/53/1/mannaro_katiuscia.pdf (30.05.15).
56. Mazur II Project management: uch. manual / Mazur II, Shapiro VD , Olderogge NG, Polkovnikov AV - M.: Omega-L, 2012. - 959 p.
57. Michael Cohn. Succeeding with Agile: Software Development Using Scrum, by Addison-Wesley Professional, 2009. – 504 p.
58. Miles Downey. Effective coaching. Good Book, M. 2007. - 288
59. Milosevic D. A set of tools for project management / Dragan Z. Milosevic: Per. with English. Mamontov EV; Ed. Unknown SI - M.: ITI Company; DMK Press, 2006. - 729 p.
60. Misra S.C., Kumar V., Kumar U. Identifying some important success factors in adopting agile software development practices // J Syst Software. 2009. T. 82. № 11. C. 1869–1890.
61. Morozov V. Formation, management and development of the project team / VV Morozov, AM Cherednichenko, TI Shpileva - Kiev: Taxon, 2009. - 461p.
62. Morozov V., Kalnichenko O., Bronin S. ”Development Of The Model Of The Proactive Approach in Creation Of Distributed Information Systems”, Eastern-European Journal of Enterprise Technologies, № 43/2 (94), pp. 6-15, 2018.
63. Morozov, V., Kalnichenko, O.: Construction of an integrated model of management processes of IT projects on the basis of a proactive approach. In Monograph “Moden management: Economy and Administration”. Opole (Poland): The Academy of Management and Administration in Opole, pp. 82-89 (2018).
64. Mostenska TL Management. Textbook. - 2nd edition. / T.L. Mostenska, V.O. Novak, M.G. Lutsky, OV Ilyenko // - K .: Condor Publishing House, 2012.- 758p.
65. Novak V.O. Organizational behavior. Textbook. / V.O. Novak, T.L. Mostenska, OV Ilyenko // - K .: Condor Publishing House, 2012.- 498p.

66. Nozdrina LV, Yashchuk VI, Polotai OI Project Management Kiev: Training Center. l-ri, 2010. - 430s.
67. Oracle Primavera P6 Enterprise Project Portfolio Management, Available: https://www.oracle.com/applications/primavera/products/project-portfolio-management/, last accessed 2019/03/25.
68. Orr Alan D. Project Management: A Guide to Key Process Models and Methods / Ed. TVGerasimova. - Dnepropetrovsk: Business Book Balance, 2006. - 224 p. - Per. with English.
69. Owen R., Koaskela L., Henrich G. Is agile project maangement applicable to construction? Santiago Chile: , 2006.
70. Palmer S. R. An Introduction to Feature-Driven Development // DZone. URL: http://agile.dzone.com/articles/introduction-feature-driven (30.05.15).
71. Pankaj Jalote, Aveejeet Palit, Priya Kurien, V.T. Peethamber, “Timeboxing: a process model for iterative software development”, The Journal of System and Software (2004), pp. 117-127
72. Pete Deemer and Gabrielle Benefield. THE SCRUM PRIMER An Introduction to Agile Project Management with Scrum Version 1.04, goodagile from www.goodagile.com, 2007
73. Portny. E. Project Management For Dummies; 2nd Edition, by Willey Publishing, Inc., 2007. – 384 p.
74. Prigogine, I., Nikolis, G.: Knowledge of the complex. Introduction Per. from English, M.: Lenar, 360 p. (2017).
75. Primak VM Project Management Kiev: University of Kiev, 2011. - 320p.
76. Proactive Project Management. Available: http://www. itexpert.ru/rus/ITEMS/200810062247/, last accessed 2019/03/12.
77. Project management. Fundamentals of project management: pupil / ML Razu, TM Bronnikov, BM Razu, etc .; Under. ed. prof. M. L. Razu. - M .: KNORUS, 2006. - 768 p.
78. Project management: a directory for professionals / Ed. VD Shapiro. - M .: Omega-L, 2011. - 960 p.

79. Project Management: Project Planning Processes [Text]: Textbook / IV. Chumachenko, V.V. Morozov, NV Dotsenko, A.M. Cherednichenko. - K .: KROK University of Economics and Law, 2014. - 670 p.
80. Project Risk Management Guidelines: Managing Risk in Large Projects and Complex Procurements / Dale F. Cooper, Stephen Gray, Geoffrey Raymond 14, and Phil Walker. John Wiley & Sons Ltd, Chichester, West Sussex, England, 2005. - 384p.
81. Rach V.A. Project Management: Practical Aspects of Implementing Regional Development Strategies: Tutorial. tool. / VA Rach, OV Rossoshanska, OM Medvedev; in a row. V.A. Racha. - K.: KI.S., 2010. - 276 p.
82. Roberts Paul. Guide to Project Management, by Profile Books/The Economist, 2007. – 319 p.
83. Sachenko A., G. Hladiy, I. Dobrotvor "Multi-agent System of IT Project Planning", in Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Vol. 2, 21-23 September, 2017, Bucharest, pp. 548-553.
84. Samokhvalov, Yu.: Development of the forecast graphethod in conditions of incompleteness and inaccuracy of expert assessments. The magazine Cybernetics and Systems Analysis;, T. 54, №1, pp. 84-91 (2018).
85. Sandeep J. Working with Agile in a Distributed Team Environment // MSDN Magazine: журнал компании Microsoft для разработчиков. URL: https://msdn.microsoft.com/en-us/magazine/hh771057.aspx (дата обращения: 30.05.15).
86. Schuh Peter. Integrating Agile Development in the Real World, by Charles River Media, 2004. – 364 p.
87. Shenhar A.J. One size does not fit all projects: Exploring classical contingency domains // Manage. Sci. 2001. T. 47. № 3. C. 394–414.
88. Sutherland J., Viktorov A., Blount J. Distributed Scrum: Agile Project Management with Outsourced Development Teams. Систем. требования:

Adobe Reader. URL: http://141.138.141.68/~vandenho/Arjan/www/SutherlandDistributedScrumAgile2006_v4_28_Feb_2006.pdf (30.05.15).

89. Sutherland, J.: Scrum: The Art of Doing Twice the Work in Half the Time, 2017, 272 p.
90. Tarasyuk G.M. Project Management Kiev: Caravel, 2009. - 320p. 15. Project Management: Monograph / Ed. VD Shapiro. - St. Petersburg: TwoThree, 1996. - 610 p.
91. Tesla Yu.M. Project Management Information Technologies Kiev, KNUBA. - 2013. - 120 p.
92. Teslia Yu.,. Khlevnyi A, Khlevna I., “Control of informational Impacts on project management”, in Proceedings of the 1th IEEE International Conference on Data Stream Mining & Processing, 23-27 August, Lviv, Ukraine, 2016.
93. Thomas D., Hansson D. H. Agile Web Development with Rails. 2007. P. 17–64. 2.
94. Turner J. Turner J. A Guide to Design-Oriented Management [Text] / J. Rodney Turner. - M.: Grebennikov Publishing House, 2007. - 552 p.
95. Turner, R.: Guide to project-based management, tran. from English, Moskow, Grebennikov Publishing House, 552 p (in Russian) (2007).
96. Tyan RB, Kholod BI, Tkachenko VA Project management. Textbook.- Kyiv: Center for Educational Literature, 2004.- 224 p.
97. Vakulenko AV Quality Management Kiev: KNEU, 2004. - 167 p. 26. Ostankova L.A. Analysis, modeling and management of economic risks Kiev: Training Center. L-ri., 2011. - 256 p.
98. Vocabulary - Handbook on Project Management / Edited by SD Bushueva. - Kyiv: Delovaya Ukraina Publishing House, 2001. - 640 p.
99. Wysocki R.K. Effective Project Management?: Traditional, Agile, Extreme. Indianapolis, IN: Wiley, 2012.
100. Xavier Amatriain Rubio, Gemma Hornos Cirera. Agile Methods in Research (presentation), by Telefonica, 2008. – 42 s.

101. Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham and Soo Fun Tan. “Software Development Life Cycle AGILE vs Traditional Approaches”, International Conference on Information and Network Technology (ICINT 2012).