Gildan Media

Companion PDF

AGILE PROJECT MANAGEMENT FOR DUMMIES

by

Mark C. Layton

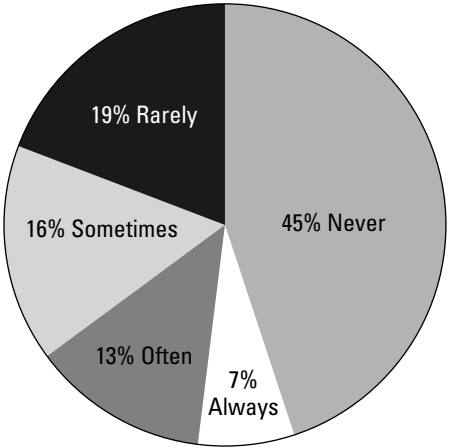


Figure 1-1: Actual use of software features.

Figure 1-2: Agile project management timeline.

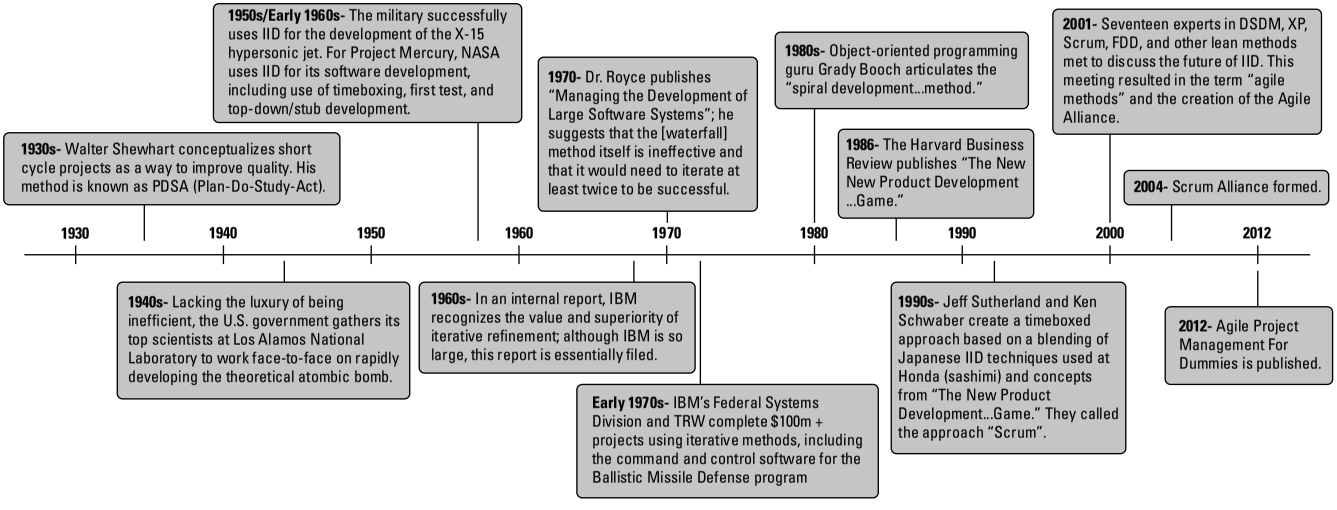


Figure 1-3: Waterfall versus agile project.

Table 2-1 Individuals and Interactions Versus Processes and Tools

	Individuals and Interactions Have High Value	***Processes and Tools Have High Value***
Pros	Communication is clear and effective. Communication is quick and efficient. Teamwork becomes strong as people work together. Development teams can self-organize. Development teams have more chances to innovate. Development teams can customize processes as necessary. Development team members can take personal ownership of the project. Development team members can have deeper job satisfaction.	Processes are clear and can be easy to follow. Written records of communication exist.
Cons	Development team members must have the *capacity* to be involved, responsible, and innovative. People may need to let go of ego to work well as members of a team.	People may over-rely on processes instead of finding the best ways to create good products. One process doesn't fit all teams — different people have different work styles. One process doesn't fit all projects. Communication can be ambiguous and time-consuming.

Table 2-2 Identifying Documentation That's Useful

Document	*Does the Document Support Product Development?*	*Is the Document Barely Sufficient or Gold-Plated?*
Project schedule created with expensive project management software, complete with Gantt Chart.	No. Start-to-finish schedules with detailed tasks and dates tend to provide more than what is necessary for product development. Also, many of these details change before you develop future features.	Gold-plated. Although project managers may spend a lot of time creating and updating project schedules, the truth is project team members tend to want to know only key deliverable dates. Management often wants to know only whether the project is on time, ahead of schedule, or behind.
Requirements documentation.	Yes. All projects have requirements — details about product features and needs. Development teams need to know those needs to create a product.	Possibly gold-plated. Requirements documents can easily grow to include unnecessary details. Agile approaches provide simple ways to describe product requirements.
Product technical specifications.	Yes. Documenting how you created a product can make future changes easier.	Possibly gold-plated; usually barely sufficient. Technical documentation usually includes just what it needs — development teams often don't have time for extra flourishes and are keen to minimize documentation.
Weekly status report.	No. Weekly status reports are for management purposes, but do not assist product creation.	Gold-plated. Knowing project status is helpful, but traditional status reports contain outdated information and are much more burdensome than necessary.
Detailed project communication plan.	No. While a contact list can be helpful, the details in many communication plans are useless to product development teams.	Gold-plated. Communication plans often end up being documents about documentation — an egregious example of busywork.

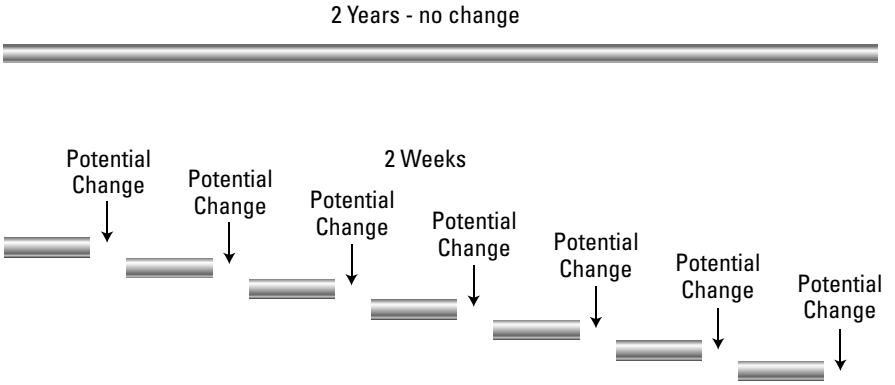


Figure 2-1: Traditional project opportunity for change.

Table 2-3 Customer Dissatisfaction and How Agile Might Help

Examples of Customer Dissatisfaction with Projects	*How Agile Approaches Can Increase Customer Satisfaction*
The product requirements were misunderstood by the development team.	Product owners work closely with the customer to define and refine product requirements and provide clarity to the development team. Agile project teams demonstrate and deliver working product features at regular intervals. If a product doesn't work the way the customer thinks it should work, the customer is able to provide feedback at the end of the sprint, not the end of the project.
The product wasn't delivered when customer needed it.	Working in sprints allows agile project teams to deliver high-priority product features early and often.
The customers can't request changes without additional cost and time.	Agile processes are built for change. Development teams can accommodate new requirements, requirement updates, and shifting priorities with each sprint — offsetting the cost of these changes by removing the lowest priority requirements.

Table 2-4 Contrasting Historical Project Management with Agile Project Management

Traditional Project Management Tasks	*Agile Approach to the Project Management Task*
Create a fully detailed project requirement document at the beginning of the project. Try to control requirement changes throughout the project.	Create a product backlog — a simple list of requirements by priority. Quickly update the product backlog as requirements and priorities change throughout the project.
Conduct weekly status meetings with all project stakeholders and developers. Send out detailed meeting notes and status reports after each meeting.	The development team meets quickly, for no longer than 15 minutes, at the start of each day to discuss that day's work and any roadblocks. They can update the centrally visible burndown chart in under a minute at the end of each day.
Create a detailed project schedule with all tasks at the beginning of the project. Try to keep the project tasks on schedule. Update the schedule on a regular basis.	Work within sprints and identify only specific tasks for the active sprint.
Assign tasks to the development team.	Support the development team by helping remove impediments and distractions. On agile projects, development teams define their own tasks.

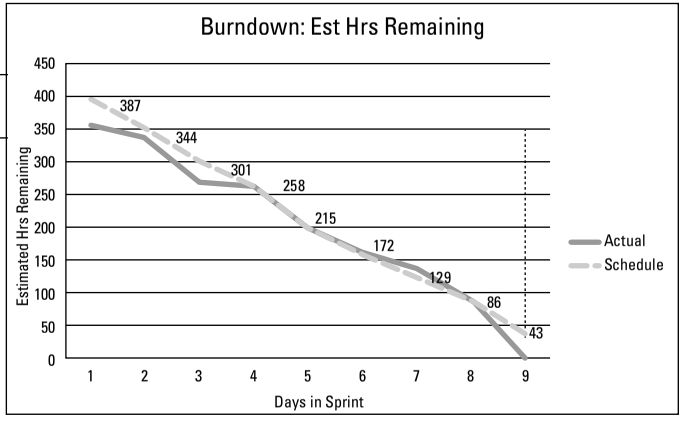
Figure 2-2: Charts, graphs, and dashboards for reporting project status.

My XYZ Mobile Banking - Sprint 1

Sprint dates: February 4 - February 15

Sprint Goal

As a <mobile banking customer>,
I want to <log in to my account>
So I can <view my account balances and pending transactions>.



Burndown - Based on Est Hours Remaining

Number of working days	9
Leona (35 hrs wk)	63
Joey (35 hrs wk)	63
Bob (35 hrs wk)	63
Marie (20 hrs wk)	36
Pablo (35 hrs wk)	63
Madison (35 hrs wk)	63
Total:	360
Total per day:	40

Feature Burndown - Based on Est Hours Remaining

Task	Priority	Status	Responsible	PO Approved?	Mo 4	Tu 5	W 6	Th 7	F 8	M 11	T 12	W 13	Th 14	F 15
User Story #1: Authenticate and Access My Accounts														
Create authentication screen for username & password with submit button	1	Completed	Madison		8	8	8	0	0	0	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Marie		4	4	4	4	4	4	0	0	0	0
Create logged in screen	1	In progress	Pablo		16	16	16	16	16	16	16	16	12	12
Using authentication code from online banking application, develop login code for iPhone / iPad application	2	In progress	Leona		24	24	24	8	4	0	0	0	0	0
Create calls to database to verify username & password	1	Completed	Bob		24	24	24	8	4	0	0	0	0	0
Create authentication screen for username & password with submit button	3	Completed	Leona		8	8	8	8	8	0	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Leona		8	8	8	8	8	0	0	0	0	0
Create logged in screen	2	Completed	Leona		16	16	16	16	4	0	0	0	0	0
Using authentication code from online banking application, develop login code for iPhone/iPad application	2	Completed	Leona		4	4	4	4	4	0	0	0	0	0
Create calls to database to verify username & password	2	Completed	Leona		4	4	4	4	4	0	0	0	0	0

world on the right.

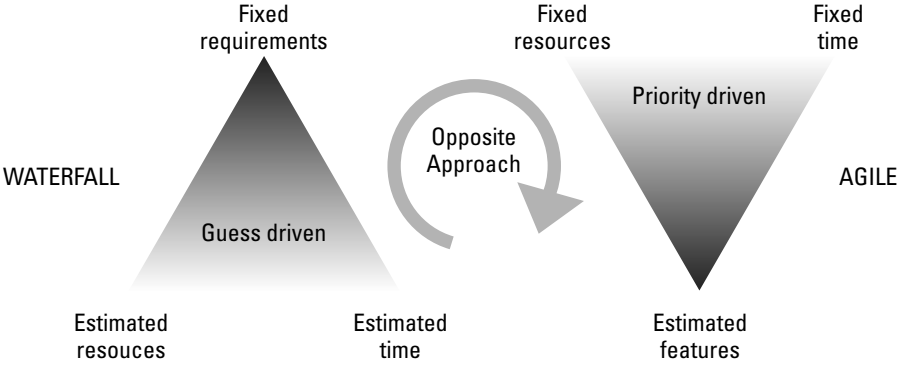


Figure 3-1: A comparison of historical project management and agile concepts.

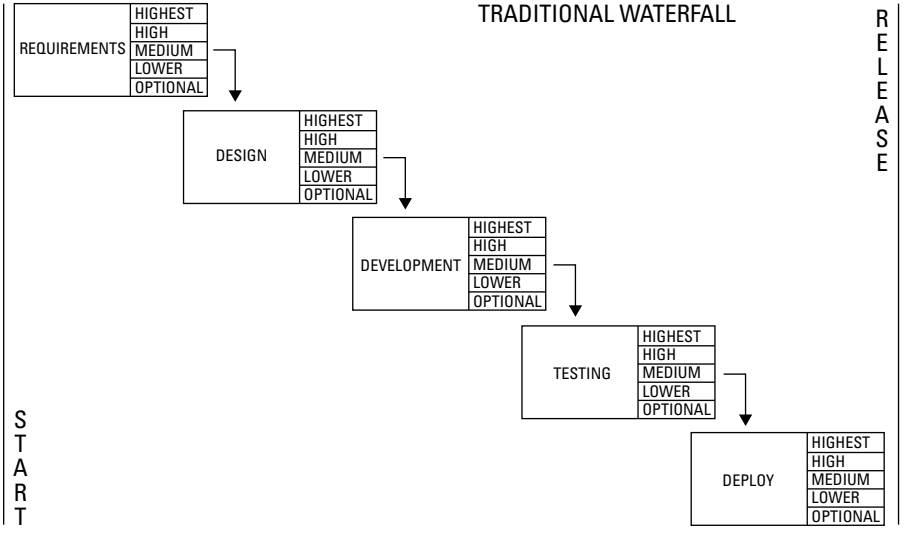


Figure 3-2: The waterfall project cycle is a linear methodology.

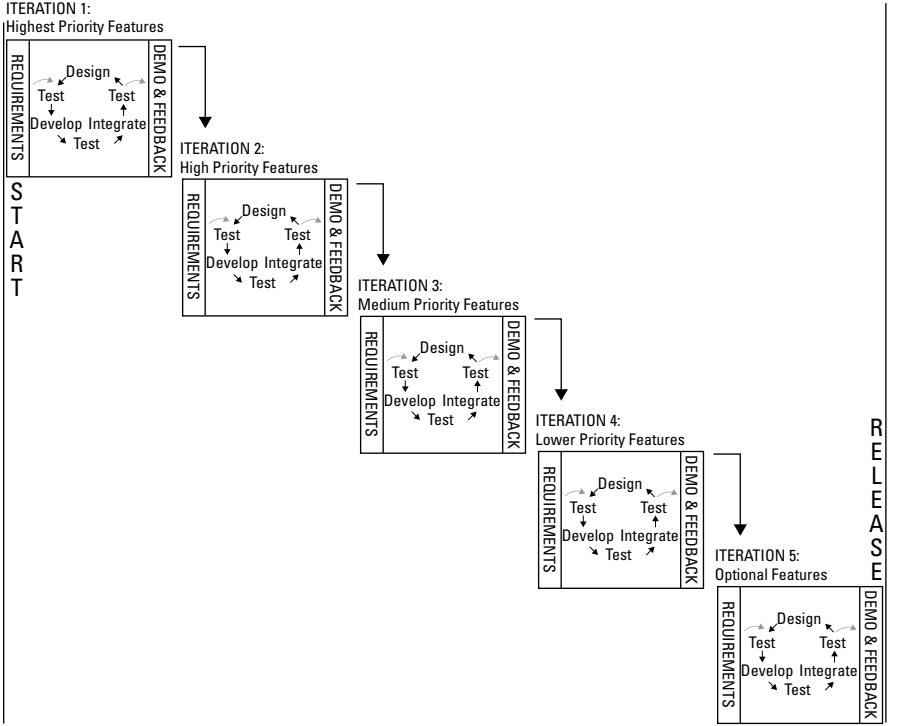


Figure 3-3: Agile approaches have an iterative project cycle.

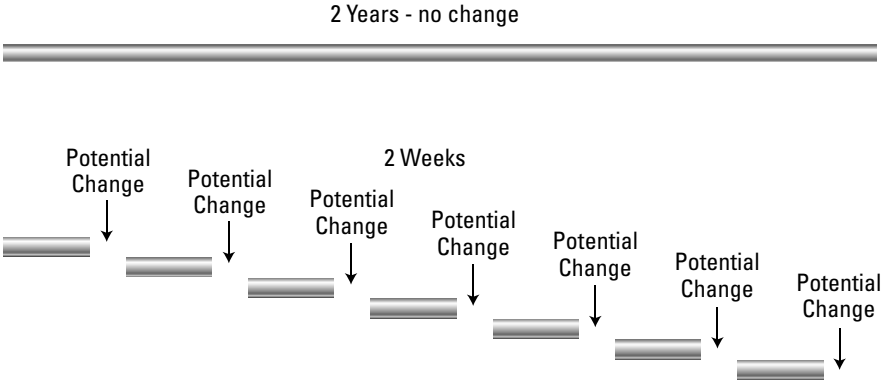


Figure 3-4: Stability in flexibility on agile projects.

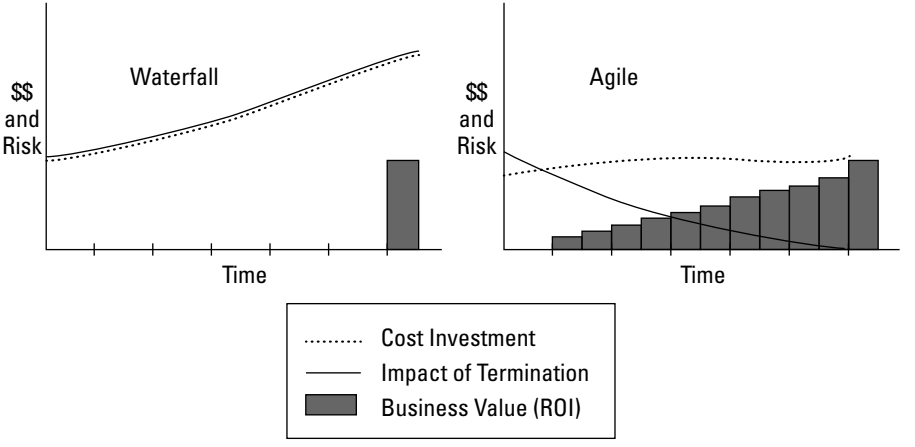
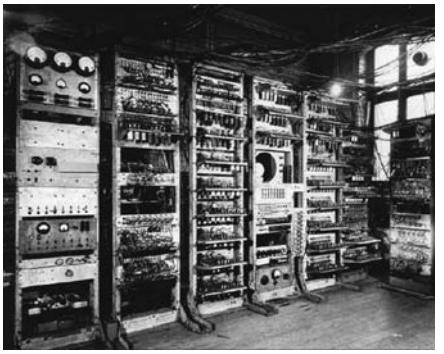


Figure 3-5: A risk and investment chart comparing waterfall and agile methodologies.



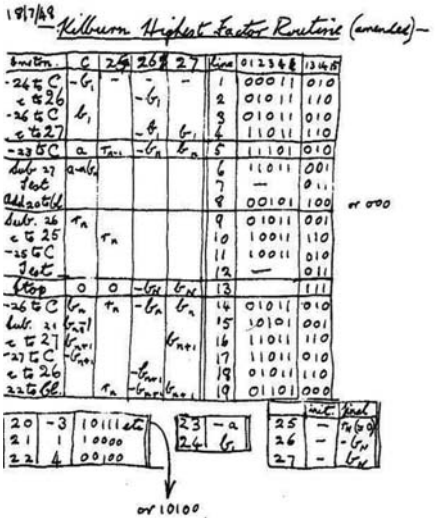


Figure 4-1: Early hardware and software.

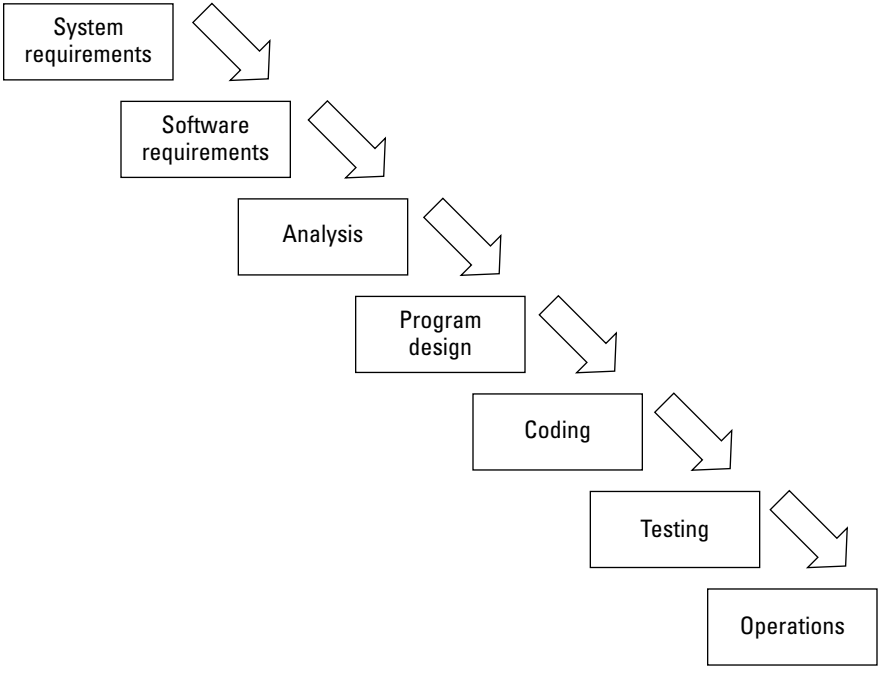


Figure 4-2: The origins of waterfall.

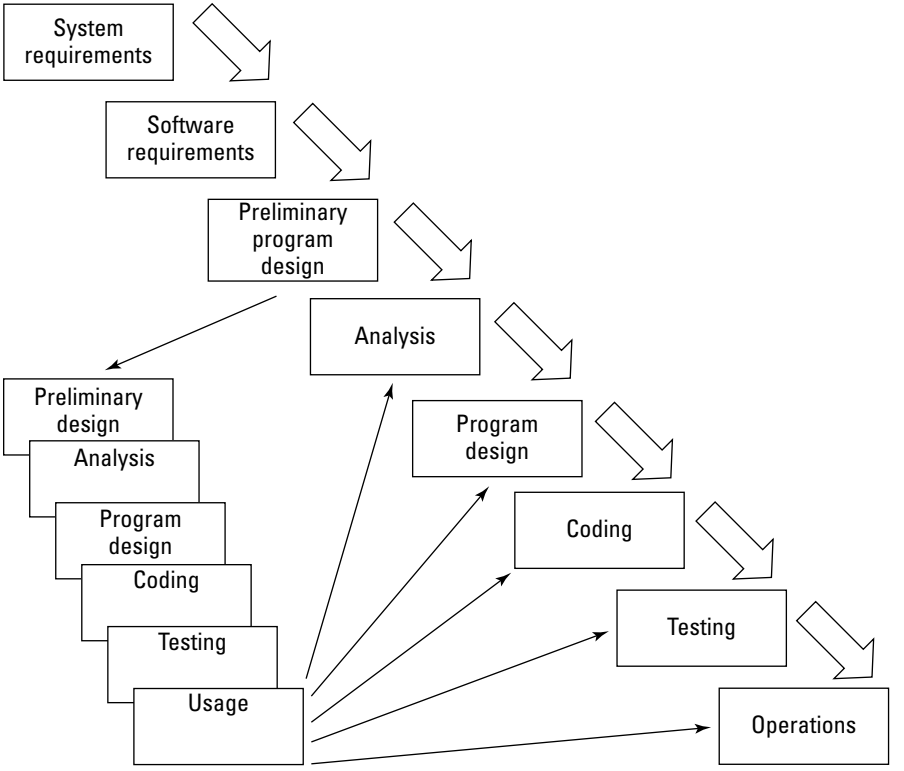


Figure 4-3: Iteration in waterfall.

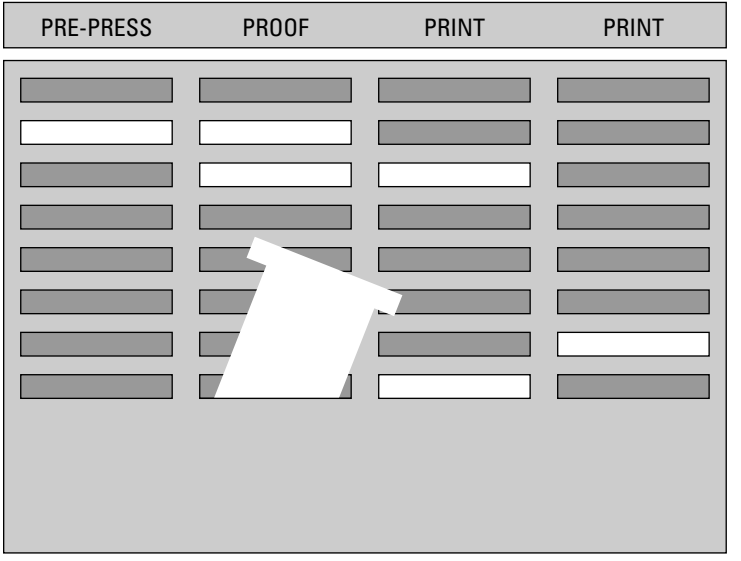


Figure 4-4: A modern kanban board.

Table 4-1 Key Practices of Extreme Programming

XP Practice	*Underpinning Assumption*
Planning game	All members of the team should participate in planning. No disconnect exists between business and technical people.
Whole team	The customer needs to be collocated (physically located together) with the development team and be available. This accessibility enables the team to ask more minor questions, quickly get answers, and ultimately deliver a product more aligned with customer expectations.
Coding standards	Use coding standards for consistency; don't constantly reinvent the basics of how to develop products within your organization. Standard code identifiers and naming conventions are two examples of having coding standards.
System metaphor	When describing how the system works, use an implied comparison, a simple story that is easily understood (for instance, "the system is like cooking a meal").
Collective code ownership	The entire team is responsible for the quality of code. Any engineer can modify another engineer's code to enable progress to continue.
Sustainable pace	Overworked people are not effective. Too much work leads to mistakes, which leads to more work, which leads to more mistakes. Avoid working more than 40 hours per week for an extended period of time.
Pair programming	Two people work together on a programming task. One person is strategic, and one person is tactical. They explain their approach to each other. No piece of code is understood by only one person.
Design improvement	Continuously improve design by refactoring code — removing duplications within the code.
Simple design	The simpler the design, the lower the cost to change the software code.
Test-driven development (TTD)	Write automated customer acceptance and unit tests before you code anything. Test your success before you claim progress.
Continuous integration	Team members should be working from the latest code. Integrate code components across the development team as often as possible to identify issues and take corrective action before problems build on each other.
Refactoring	Expect to improve code constantly. The fewer dependencies, the better.
Small releases	Release value to the customer often. Avoid going more than three to four months without a customer release. Some organizations release daily.

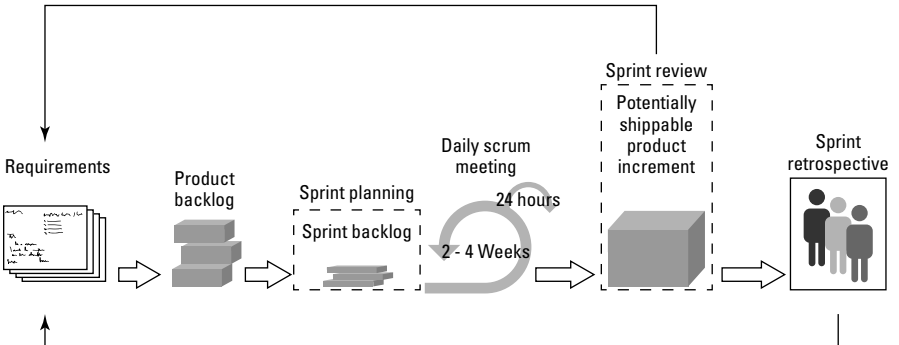


Figure 4-5: The scrum approach.

Table 4-2 Similarities Between Lean, Extreme Programming, and Scrum

Lean	*Extreme Programming*	*Scrum*
Engaging everyone	Entire team Collective ownership	Cross-functional development team
Optimizing the whole	Test-driven development Continuous integration	Product increment
Delivering fast	Small release	One- to four-week sprints

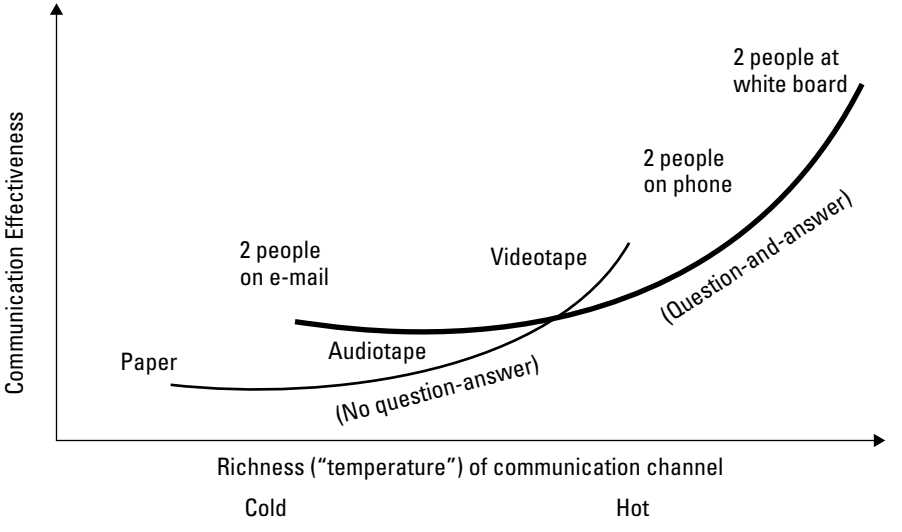


Figure 5-1: Better communication through collocation.

Table 5-1 Common Distractions

Distraction	*Do*	*Don't*
Multiple projects	Do make sure that the development team is dedicated 100 percent to a single project — this one!	Don't fragment the development team between multiple projects, operations support, and special duties.
Multitasking	Do keep the development team focused on a single task, ideally coding one piece of functionality at a time. A task board can help keep track of the tasks in progress and quickly identify whether someone is working on multiple tasks at once.	Don't let the development team switch requirements. Switching tasks creates a huge overhead in lost productivity.

Distraction	*Do*	*Don't*
Over-supervising	Do leave development team members alone after you collaborate on iteration goals; they can organize themselves. Watch their productivity skyrocket.	Don't interfere with the development team or allow others to do so. The daily scrum meeting provides ample opportunity to assess progress.
Outside influences	Do redirect any distracters. If another task surfaces, ask the product owner to decide whether the task's priority is worth sacrificing sprint functionality.	Don't mess with the development team members and their work. They're pursuing the sprint goal, which is the top priority during an active sprint. Even a seemingly quick task can throw off work for an entire day.
Management	Do shield the development team from direct requests from management (unless management wants to give team members a bonus for their excellent performance).	Don't allow management to negatively affect the productivity of the development team. Make interrupting the development team the path of greatest resistance.

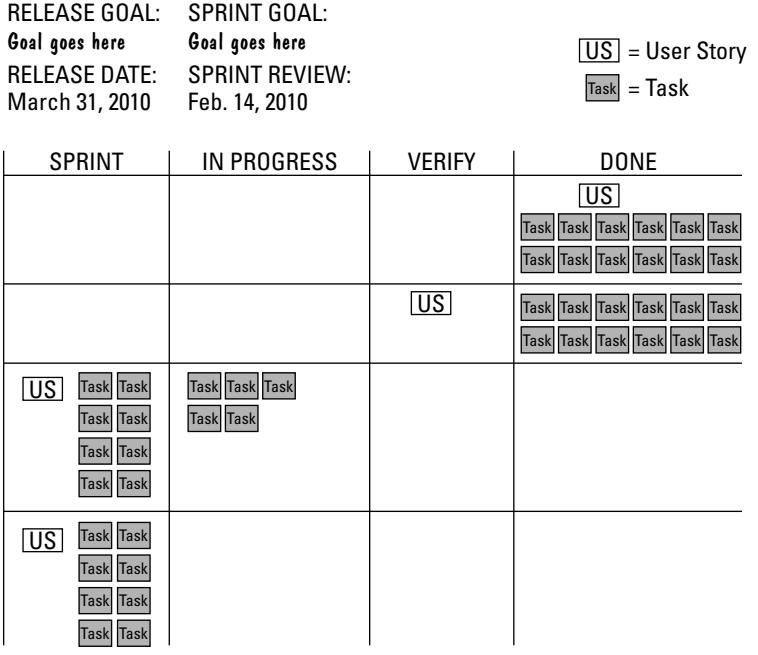


Figure 5-2: A white board and kanban board used in agile.

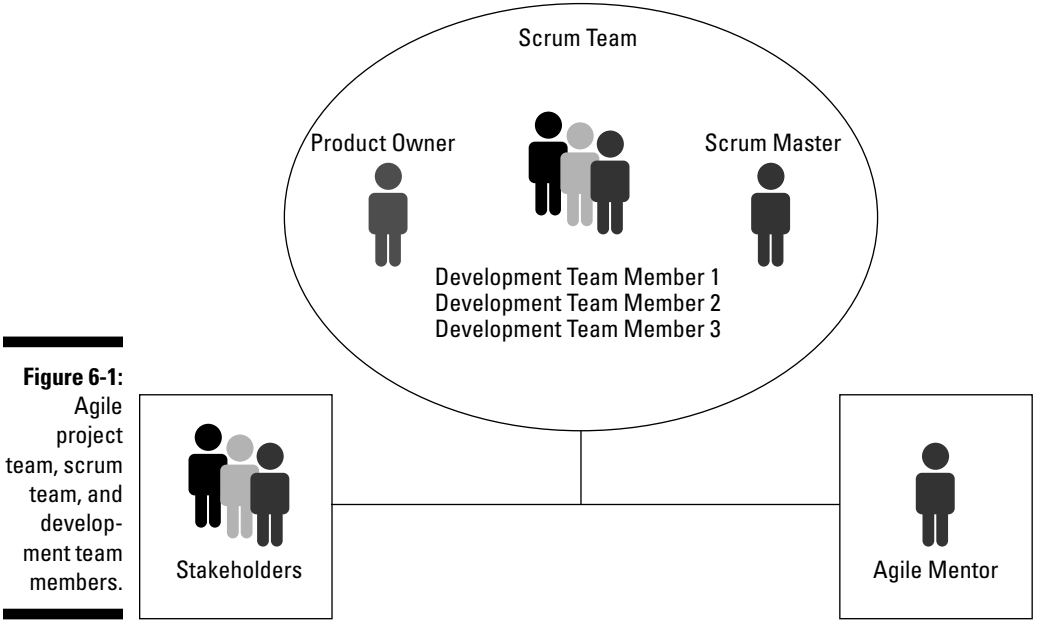


Figure 6-1: Agile project team, scrum team, and development team members.

Table 6-1 Characteristics of a Good Agile Team Member

Responsibility	*A Good Agile Team Member . . .*
Creates the product.	Enjoys creating products. Is skilled in at least one of the jobs necessary to create the product.
Is self-organizing and self-managing.	Exudes initiative and independence. Understands how to work through impediments to achieve goals.
Is cross-functional.	Has curiosity. Willingly contributes to areas outside his or her mastery. Enjoys learning new skills. Enthusiastically shares knowledge.
Is dedicated and collocated.	Is part of an organization that understands the gains in efficiency and effectiveness associated with focused, collocated teams.

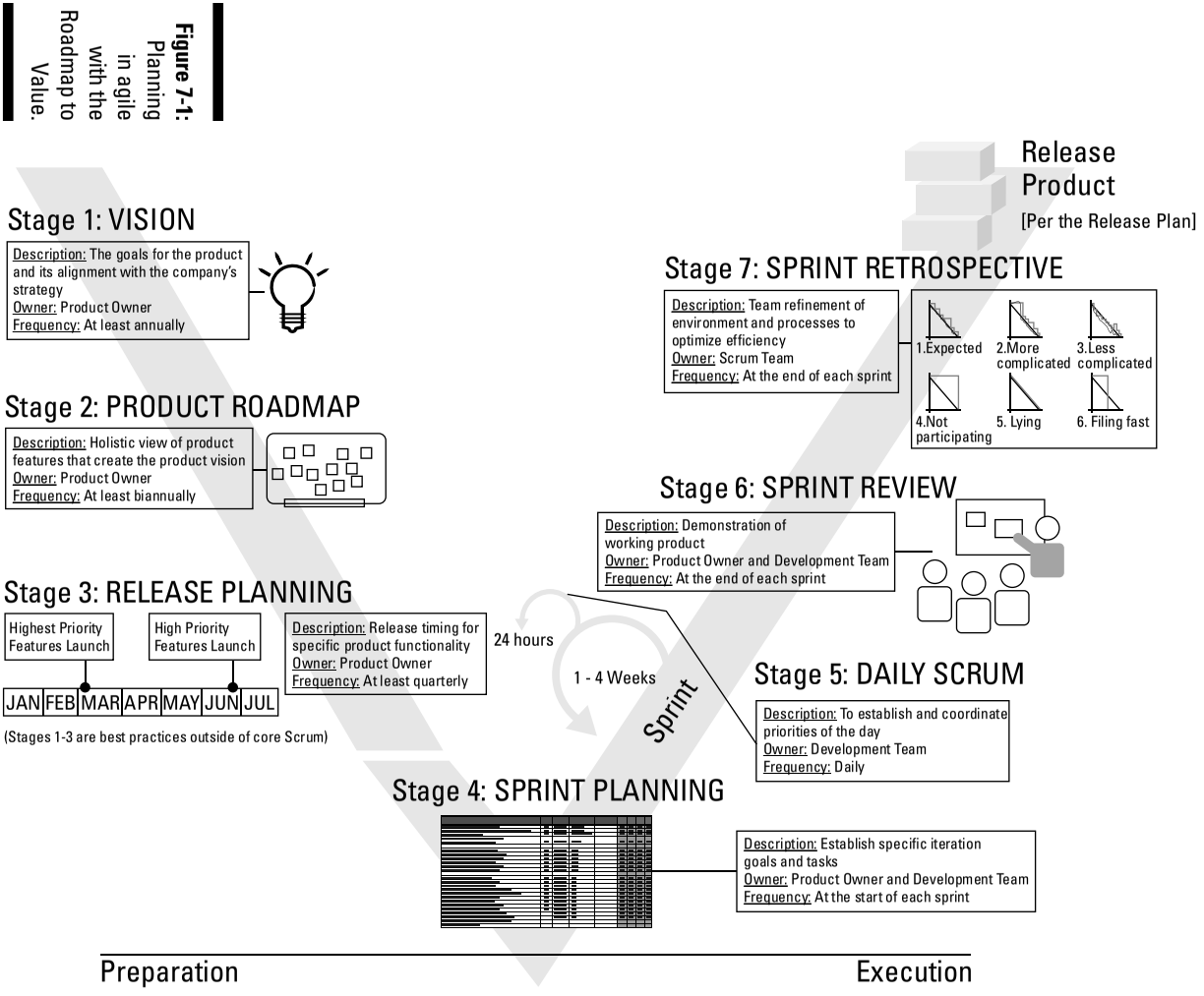
Table 6-2 Characteristics of a Good Product Owner

Responsibility	*A Good Product Owner . . .*
Supplies project strategy and direction.	Envisions the completed product. Firmly understands company strategy.
Provides product expertise.	Has worked with similar products in the past. Understands needs of the people who will use the product.
Understands customer and other stakeholder needs.	Understands relevant business processes. Creates a solid customer input and feedback channel. Works well with business stakeholders.
Manages and prioritizes product requirements.	Focuses on efficiency. Remains flexible. Is decisive. Turns stakeholder feedback into valuable, customer-focused features. Is practical about prioritizing financially valuable features, high-risk features, and strategic system improvements.
Is responsible for budget and profitability.	Understands which product features can deliver the best return on investment. Manages budgets effectively.
Decides on release dates.	Understands business needs regarding timelines.
Works with development team.	Works with the development team to understand capabilities. Works well with developers. Adeptly describes product features. Avails himself or herself for questions and clarification every day.
Accepts or rejects work.	Understands requirements and ensures that completed features work correctly.
Presents completed work at the end of each sprint.	Clearly introduces the accomplishments of the sprint before the development team demonstrates the sprint's working functionality.

Table 6-3 Characteristics of a Good Scrum Master

Responsibility	*A Good Scrum Master . . .*
Upholds scrum values and practices.	Is an expert on scrum processes. Is passionate about agile techniques.
Removes roadblocks and prevents disruptions.	Has organizational clout and can resolve problems quickly. Is articulate, diplomatic, and professional. Is a good communicator and a good listener. Is firm about the development team's need to focus only on the project and the current sprint.
Fosters close cooperation between external stakeholders and the scrum team.	Looks at the needs of the project as a whole. Avoids cliques and helps break down group silos.
Facilitates consensus building.	Understands techniques to help groups reach agreements.
Is a servant-leader.	Does not need or want to be in charge or be the boss. Ensures that all members of the development team have the information they need to do the job, use their tools, and track progress. Truly desires to help the scrum team.

Figure 7-1: Planning in agile with the Roadmap to Value.



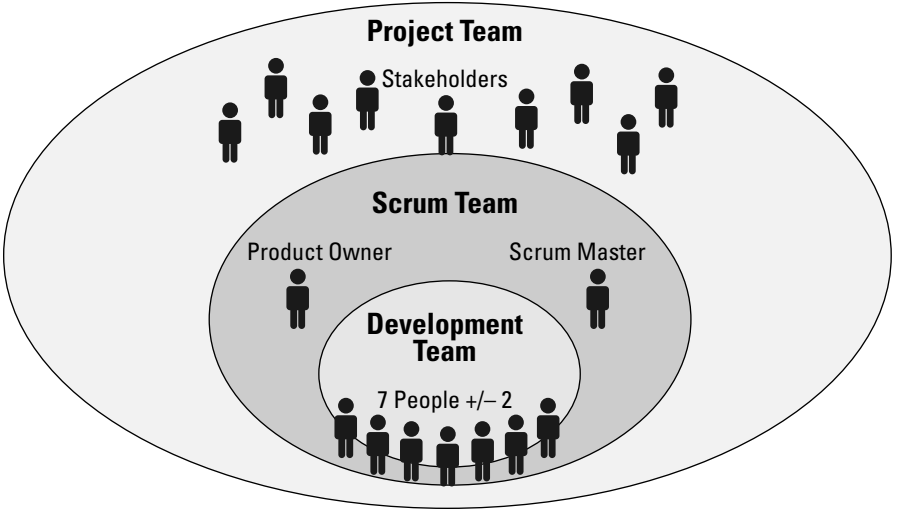


Figure 7-2: Teams in agile.

Figure 7-3 shows how the vision statement — Stage 1 on the Roadmap to Value — fits with the rest of the stages and activities in an agile project.

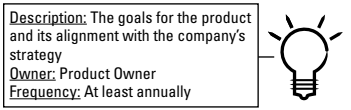


Figure 7-3: The product vision statement as part of the Roadmap to Value.

The product owner is responsible for knowing about the product, its goals, and its requirements throughout the project. For those reasons, the product owner creates the vision statement, although other people may have input. After the vision statement is complete, it becomes a guiding light, the "what we are trying to achieve" statement that the development team, scrum master, and stakeholders refer to throughout the project.

When creating a product vision statement, follow these four steps:

1. **Develop the product objective.**
2. **Create a draft vision statement.**
3. **Validate the vision statement with product and project stakeholders. Revise the vision statement based on feedback.**
4. **Finalize the vision statement.**

The look of a vision statement follows no hard-and-fast rules. However, anyone involved with the project, from the development team to the CEO, should be able to understand the statement. The vision statement should be internally focused, clear, nontechnical, and as brief as possible. The vision statement should also be explicit and avoid marketing fluff.

Vision Statement for Product:

For: _____ (Target Customer)

who: _____ (needs)

the: _____ (product name)

is a: _____ (product category)

that: _____ (product benefit, reason to buy)

Unlike: _____ (competitors)

our product: _____ (differentiation/value proposition)

Figure 7-4: Expansion of Moore's template for a vision statement.

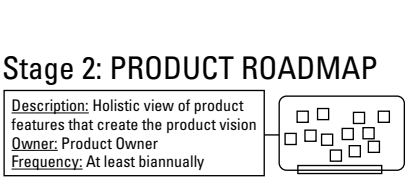


Figure 7-5: The product roadmap as part of the Roadmap to Value.

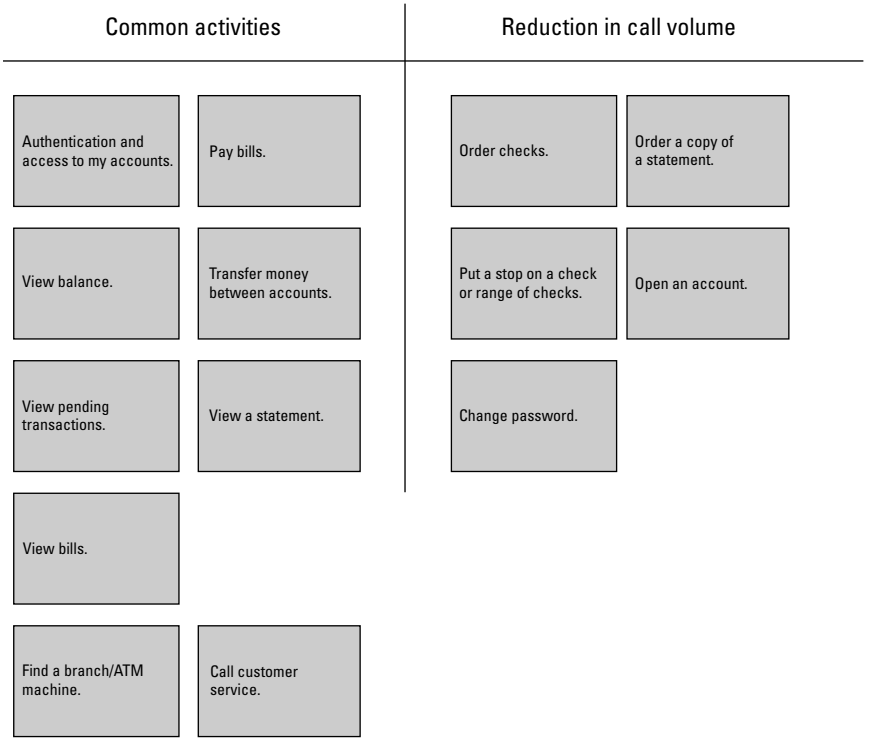


Figure 7-6: Features grouped by themes.

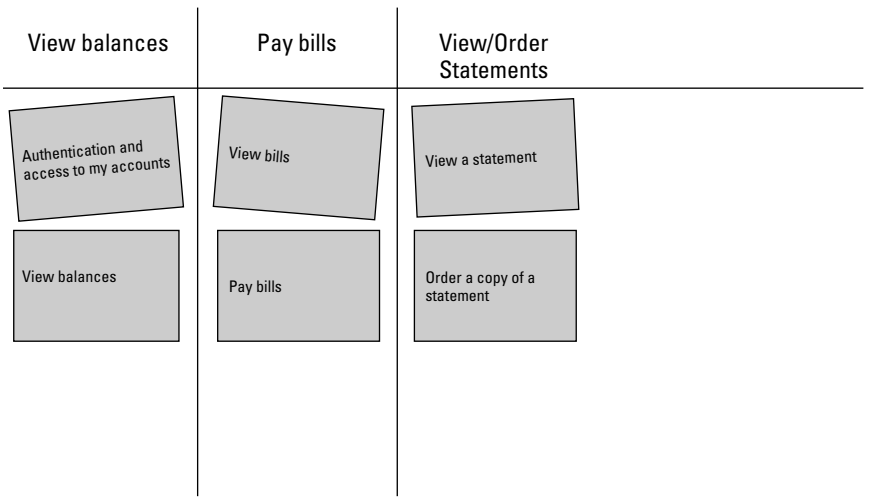


Figure 7-7: Requirement categories on a white board.

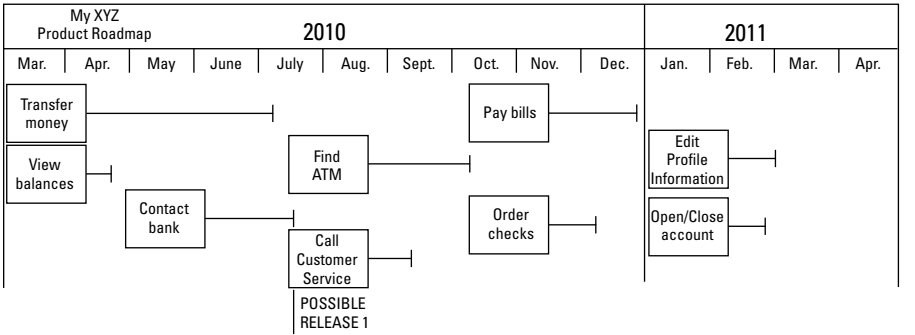


Figure 7-8: Product roadmap with prioritized require-ments.

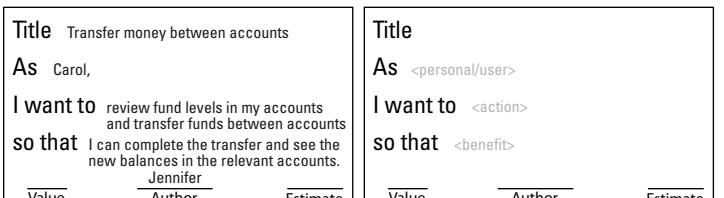


Figure 8-1: Card-based user story example.

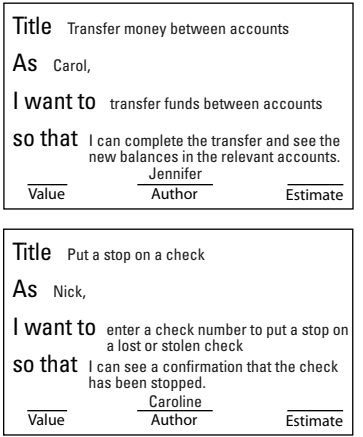


Figure 8-2: Sample user stories.

Table 8-1	Decomposing a Requirement
Requirement Level	***Requirement***
Theme	See account data with a mobile application.
Features	See account balances.
	See a list of recent withdrawals or purchases.
	See a list of recent deposits.
	See my upcoming automatic bill payments.
	See my account alerts.

Table 8-1 ***(continued)***

Requirement Level	*Requirement*
Epic User Stories — decomposed from "see account balances"	See checking account balance.
	See savings account balance.
	See investment account balance.
	See retirement account balance.
User Stories — decomposed from "see checking account balance"	Log into mobile account.
	Securely log into mobile account.
	See a list of my accounts.
	Select and view my checking account.
	See account balance changes after withdrawals.
	See account balance changes after purchases.
	See day's end account balance.
	See available account balance.
	See mobile application navigation items.
	Change account view.
	Log out of mobile application.

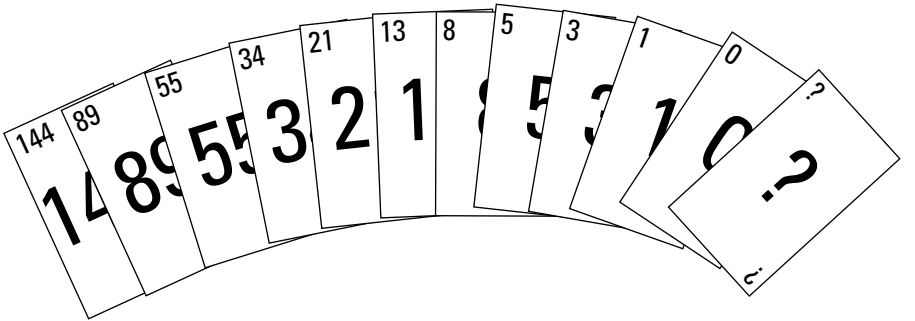


Figure 8-3: A deck of estimation poker cards.

SIZE	POINTS
XtraSmall (XS)	1 pt
Small (S)	2 pts
Medium (M)	3 pts
Large (L)	5 pts
XtraLarge (XL)	8 pts

Figure 8-4: Story sizes as T-shirt sizes and their Fibonacci numbers.

ning fits into an agile project.

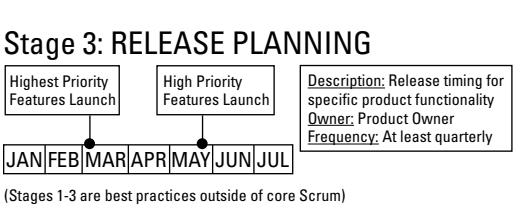


Figure 8-5: Release planning as part of the Roadmap to Value.

ID	Story	Type	Status	Value
121	As an Administrator, I want to link accounts to profiles, so that customers can access new accounts.	Feature	Not started	5
113	As a Customer, I want to view my account balances, so that I know how much money is currently in each account.	Feature	Not started	3
403	As a Customer, I want to transfer money between my active accounts, so that I can adjust each account's balance.	Feature	Not started	1
97	As a Site Visitor, I want to contact the bank, so that I can ask questions and raise issues.	Feature	Not started	2
68	As a Site Visitor, I want to find locations, so that I can use bank services.	Feature	Not started	8

Figure 8-6: Product backlog sample.

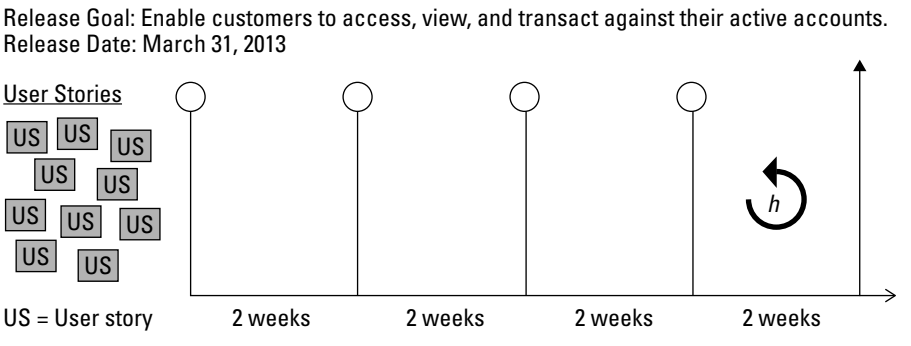


Figure 8-7: Sample release plan.

Figure 8-8: Sprint planning as part of the Roadmap to Value.

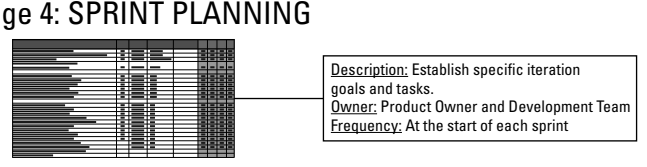


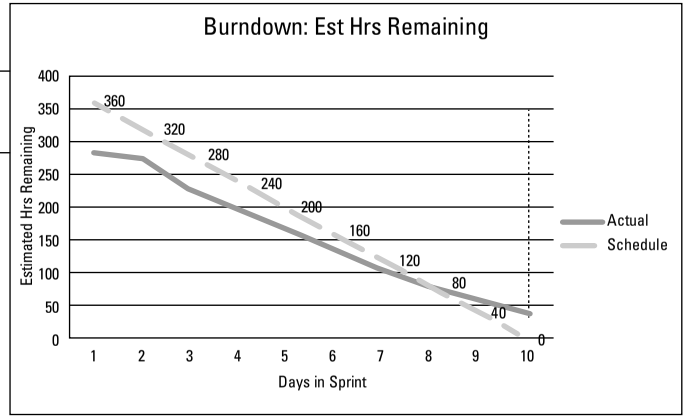
Figure 8-9: Sprint back-log example.

My XYZ Mobile Banking - Sprint 1

Sprint dates: February 4 - February 15

Sprint goal

As a <mobile banking customer>,
I want to <log in to my account>
So I can <view my account balances and pending transactions>.



Burndown - Based on Est Hours Remaining	
Number of working days	9
Leona (35 hrs wk)	63
Joey (35 hrs wk)	63
Bob (35 hrs wk)	63
Marie (20 hrs wk)	63
Pablo (35 hrs wk)	63
Madison (35 hrs wk)	63
Total:	387
Total per day:	43

Feature Burndown - Based on Est Hours Remaining

Task	Priority	Status	Responsible	PO Approved?	Mo 4	Tu 5	W 6	Th 7	F 8	M 11	T 12	W 13	Th 14	F 15
User Story #1: Authenticate and Access My Accounts														
Create authentication screen for username & password with submit button	1	Completed	Madison		3	8	8	0	0	0	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Marie		4	4	4	4	4	4	0	0	0	0
Create logged in screen	1	In progress	Pablo		16	16	16	16	16	16	16	16	12	12
Using authentication code from online banking application, develop login code for iPhone / iPad application	2	In progress	Leona		24	24	24	8	4	0	0	0	0	0
Create calls to database to verify username & password	1	Completed	Bob		24	24	24	8	4	0	0	0	0	0
Create authentication screen for username & password with submit button	3	Completed	Leona		8	8	8	8	8	0	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Leona		8	8	8	8	8	0	0	0	0	0
Create logged in screen	2	Completed	Leona		16	16	16	16	4	0	0	0	0	0
Using authentication code from online banking application, develop login code for iPhone / iPad application	2	Completed	Leona		4	4	4	4	4	0	0	0	0	0
Create calls to database to verify username & password	2	Completed	Leona		4	4	4	4	4	0	0	0	0	0

If my sprint is this long...	My sprint planning meeting should last no more than...
One week	Two hours
Two weeks	Four hours
Three weeks	Six hours
Four weeks	Eight hours

Figure 8-10: Sprint planning meeting to sprint length ratio.

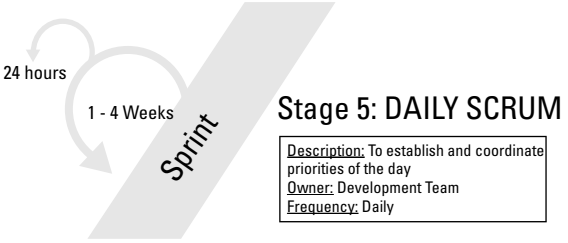


Figure 9-1: The sprint and the daily scrum in the Roadmap to Value.

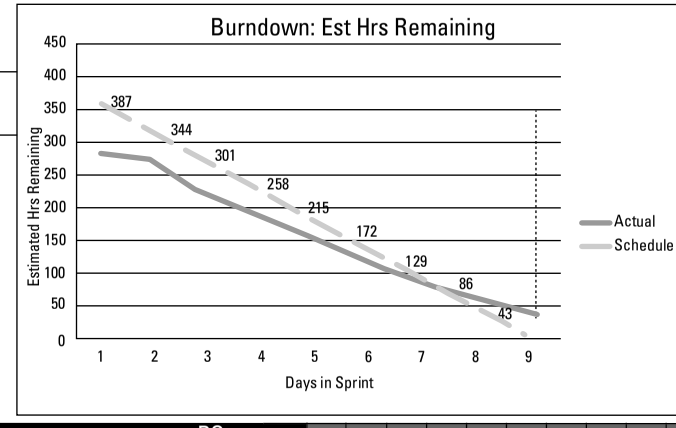
Figure 9-2: Sample sprint backlog.

My XYZ Mobile Banking - Sprint 1

Sprint dates: February 4 - February 15

Sprint goal

As a <mobile banking customer>,
I want to <log in to my account>
So I can <view my account balances and pending transactions>.



Burndown - Based on Est Hours Remaining

Number of working days	9
Leona (35 hrs wk)	63
Joey (35 hrs wk)	63
Bob (35 hrs wk)	63
Marie (20 hrs wk)	63
Pablo (35 hrs wk)	63
Madison (35 hrs wk)	63
Total:	387
Total per day:	43

Feature Burndown - Based on Est Hours Remaining

Task	Priority	Status	Responsible	PO Approved?	M 4	Tu 5	W 6	Th 7	F 8	M 11	T 12	W 13	Th 14	F 15
User Story #1: Authenticate and Access My Accounts														
Create authentication screen for username & password with submit button	1	Completed	Madison		8	3	8	0	0	0	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Marie		4	0	4	0	0	0	0	0	0	0
Create logged in screen	1	In progress	Pablo		7	2	16	0	0	0	0	0	0	0
Using authentication code from online banking application, develop login code for iPhone / iPad application	2	In progress	Leona		6	2	2	0	0	0	0	0	0	0
Create calls to database to verify user name & password	1	Completed	Bob		8	0	0	0	0	0	0	0	0	0
Create authentication screen for username & password with submit button	3	Completed	Leona		8	8	0	1	1	1	0	0	0	0
Create error screen for user to re-enter credentials	3	Completed	Bob		8	8	0	0	0	0	0	0	0	0
Create logged in screen	2	Completed	Madison		6	2	0	0	0	0	0	0	0	0
Using authentication code from online banking application, develop login code for iPhone/iPad application	2	Completed	Leona		4	4	4	0	0	0	0	0	0	0
Create calls to database to verify username & password	2	Completed	Marie		4	4	0	0	0	0	0	0	0	0

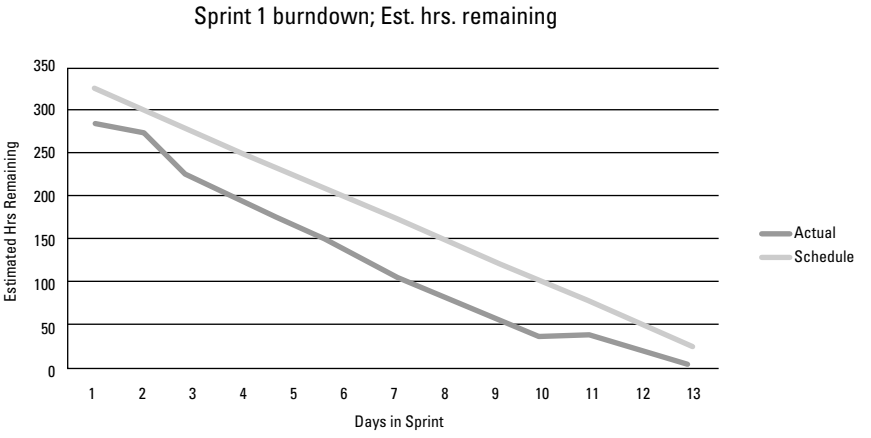


Figure 9-3: Burndown chart.

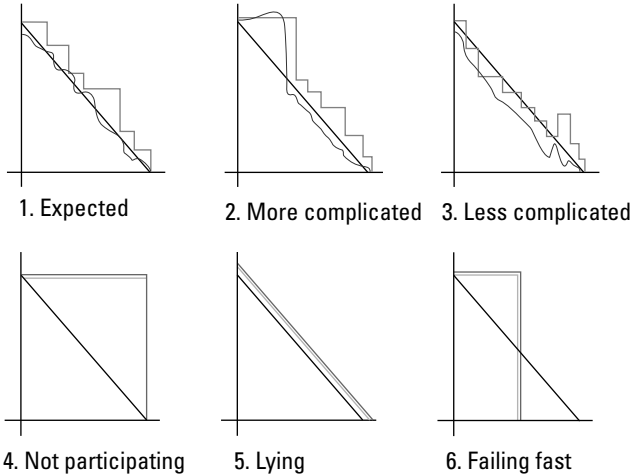


Figure 9-4: Profiles of burndown charts.

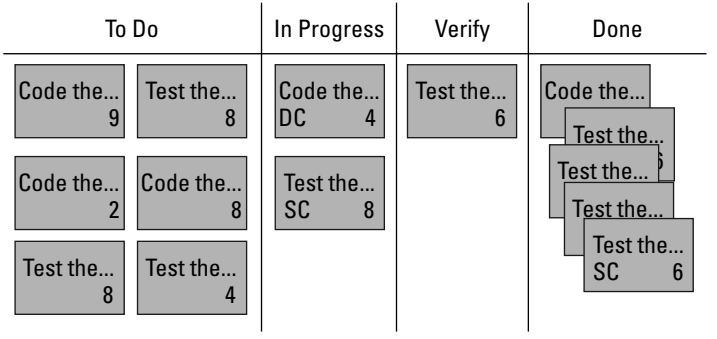


Figure 9-5: Sample task board.

The task board is a lot like a kanban board. *Kanban* is a Japanese term that means *visual signal.* (For more on kanban boards, see Chapter 4.) Toyota created these boards as part of its lean manufacturing process.

Day-to-day work on an agile project involves more than just planning and tracking progress. In the next section, you see what most of your day's work will include, whether you are a member of the development team, a product owner, or a scrum master.

Some development teams report status only with a task board and ask the scrum master to convert status into the sprint backlog. This process helps the scrum master see trends and potential issues.

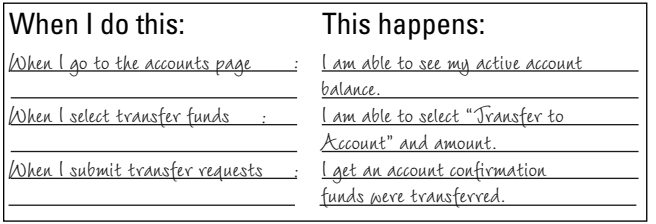


Figure 9-6: User story verification.

Table 9-1 Common Roadblocks and Solutions

Roadblock	*Action*
The development team needs simulation software for a range of mobile devices so that it can test the user interface and code.	Do some research to estimate the cost of the software, prepare a summary of that for the product owner, and have a discussion about funding. Process the purchase through procurement, and deliver the software to the development team.
Management wants to borrow a development team member to write a couple of reports. All your development team members are fully occupied.	Tell the requesting manager that person is not available, and is not likely to be for the duration of the project. As you're likely a problem solver, you may want to suggest alternative ways in which the manager could get what he or she needs. You may also have to justify why you cannot pull the person off the project, even for half a day.
A development team member cannot move forward on a user story because he or she does not fully understand the story. The product owner is out of the office for the day on a personal emergency.	Work with the development team member to determine if any work can happen around this user story while waiting on an answer. Help locate another person who could answer the question. Failing that, ask the development team to review upcoming tasks (not related to this stopped one) and move things around to keep productivity up.
A user story has grown in complexity and now appears to be too large for the sprint length.	Have the development team work with the product owner to break the user story down so that some demonstrable value can be completed in the current sprint and the rest can be put back into the product backlog. The goal is to ensure this sprint ends with completion, even if that is a smaller user story, rather than an incomplete user story.

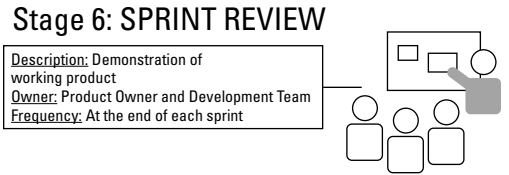


Figure 10-1: The sprint review in the Roadmap to Value.

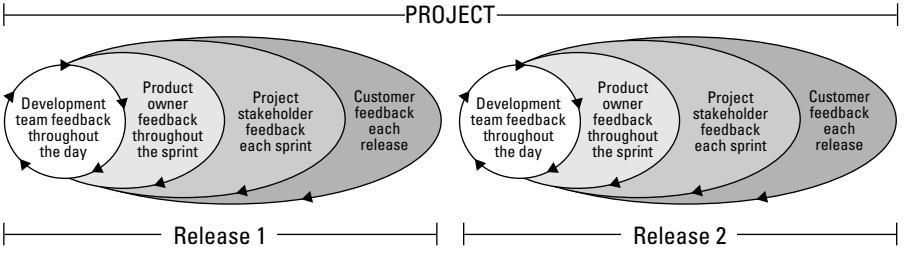


Figure 10-2: Agile project feedback loops.

The sprint review usually takes place later in the day on the last day of the sprint, often a Friday. One of the rules of scrum is to spend no more than one hour in a sprint review meeting for every week of the sprint — Figure 10-3 shows a quick reference.

If my sprint is this long...	My sprint review meeting should last no more than...
One week	One hour
Two weeks	Two hours
Three weeks	Three hours
Four weeks	Four hours

Figure 10-3: Sprint review meeting to sprint length ratio.

If my sprint is this long...	My sprint review meeting should last no more than...
One week	45 minutes
Two weeks	1.5 hours
Three weeks	2.25 hours
Four weeks	Three hours

Figure 10-4: Sprint retrospective meeting to sprint length ratio.

Table 11-1 Development Sprint Elements Versus Release Sprint Elements

Element	*Used in Development Sprint*	*Used in Release Sprint*
Sprint planning	Yes	Yes
Product backlog	Yes	No
Sprint backlog	Yes For a development sprint, your sprint backlog contains user stories and the tasks needed to create each user story. You estimate user stories relatively, with story points. (See Chapters 7 and 8.)	Yes In a release sprint, you no longer need to put your requirements in the user story format. Instead, you will create only a list of tasks needed for the release. You will not use story points, either; just add the estimated hours each task will take.
Burndown chart	Yes	Yes
Daily scrum	Yes	Yes Involve stakeholders from outside the scrum team who have tasks associated with releasing the product, like enterprise build managers or other configuration managers.
Daily activities	In a development sprint, your daily activities focus on creating shippable code.	In a release sprint, your daily activities focus on preparing your working software for external release.

(continued)

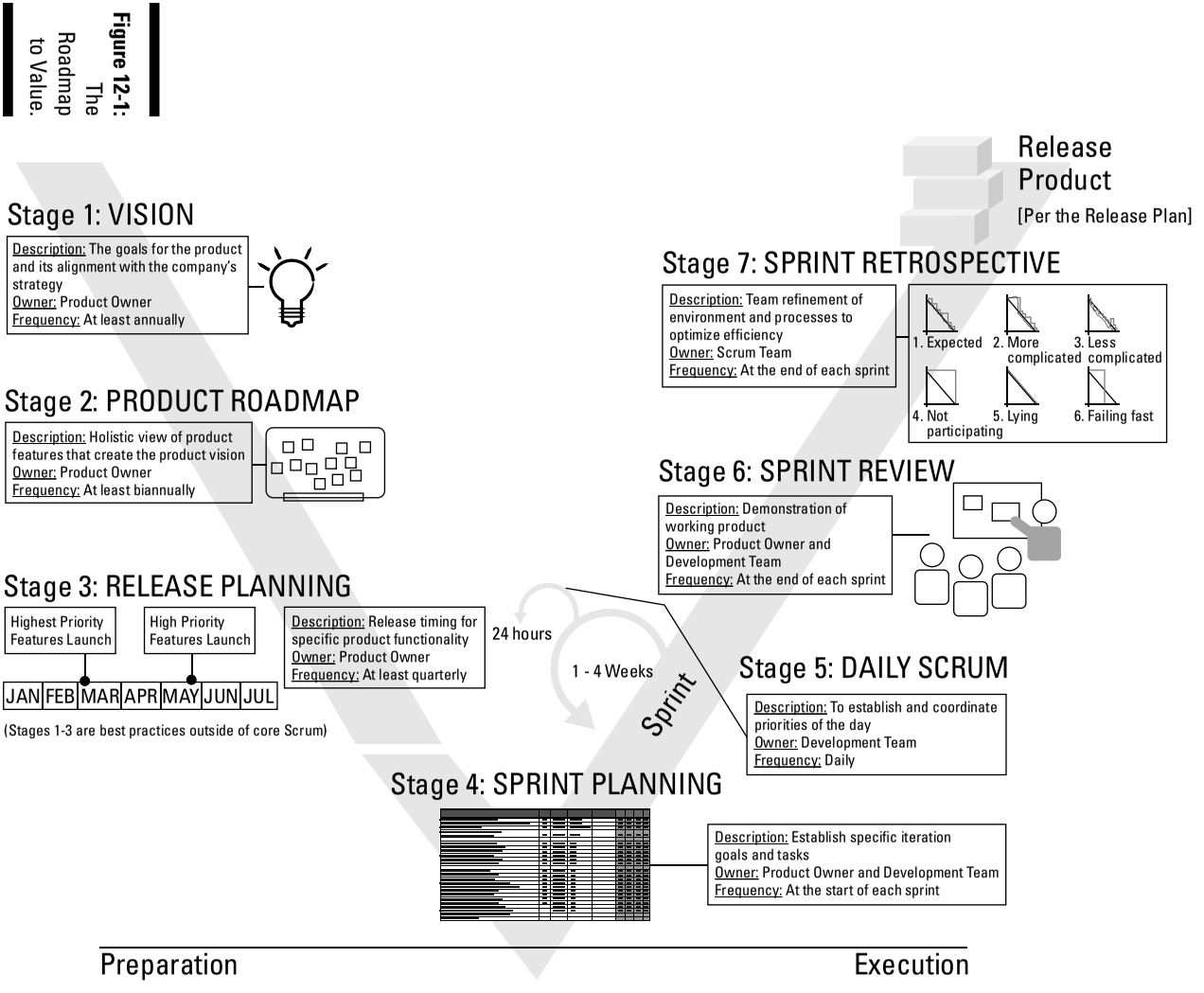
Table 11-1 *(continued)*

Element	*Used in Development Sprint*	*Used in Release Sprint*
End-of-day reporting	Yes	Yes
Sprint review	Yes	Yes Some organizations use a release sprint review as a *go or no-go meeting* to authorize launching the product.
Sprint retrospective	Yes	Yes

Table 12-1 Historical Versus Agile Scope Management

Scope Management with Traditional Approaches	*Scope Management with Agile Approaches*
Project teams attempt to identify and document complete scope at the beginning of the project, when the teams are the least informed about the product.	You gather high level requirements at the beginning of your project, breaking down and further detailing requirements that are going to be implemented in the immediate future. Requirements are gathered and refined throughout the project as the team's knowledge of customer needs and project realities grows.
Organizations view scope change after the requirements phase is complete as negative.	Organizations view change as a positive way to improve a product as the project progresses. Changes late in the project, when you know the most about the product, are often the most valuable changes.
Project managers rigidly control and discourage changes once stakeholders sign off on requirements.	Change management is an inherent part of agile processes. You assess scope and have an opportunity to include new requirements with every sprint. The product owner determines the value and priority of new requirements and adds those requirements to the product backlog.
The cost of change increases over time, while the ability to make changes decreases.	You fix resources and schedule initially. New features with high priority don't necessarily cause budget or schedule slip; they simply push out the lowest-priority features. Iterative development allows for changes with each new sprint.
Projects often include *scope bloat,* unnecessary product features included out of fear of mid-project change.	You determine scope by considering which features directly support the project vision, the release goal, and the sprint goal. The development team creates the most valuable features first to guarantee their inclusion and to ship those features as soon as possible. Less valuable features might never be created.

Figure 12-1: The Roadmap to Value.



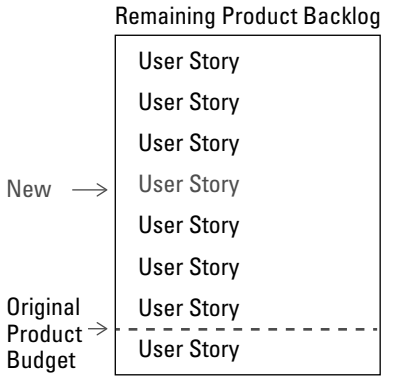


Figure 12-2: Adding a new requirement to the product backlog.

Keeping the product backlog up to date will allow you to quickly prioritize and add new requirements. With a current product backlog, you always understand the scope left in a project. Chapter 7 has more information about prioritizing requirements.

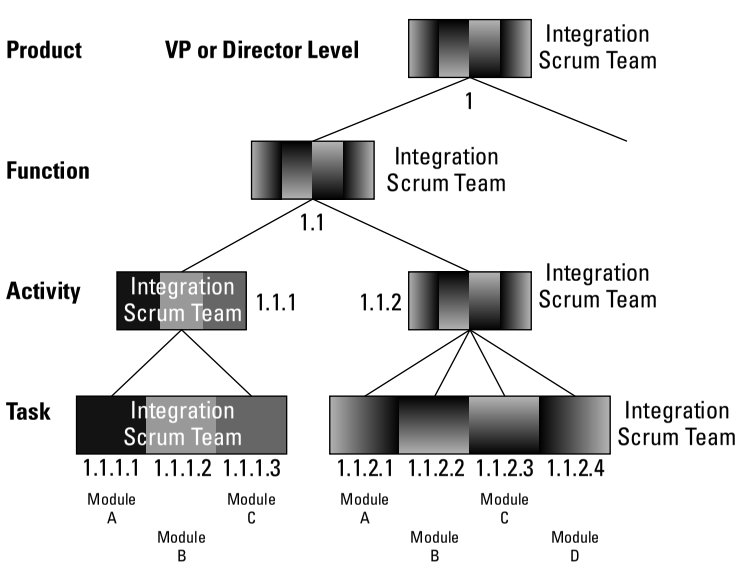
Table 12-2 Agile Artifacts and Scope Management Roles

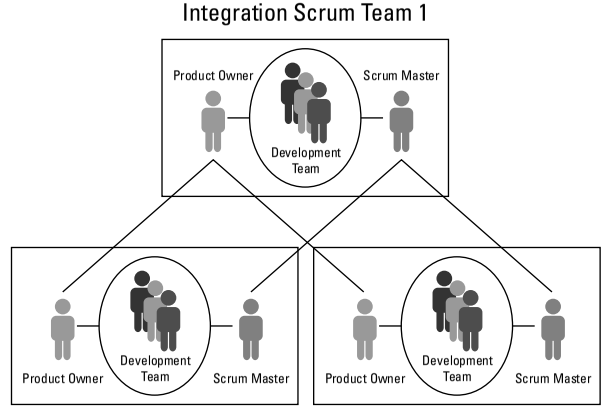
Artifact	*Role in Establishing Scope*	*Role in Scope Change*
Vision Statement: A definition of the product's end goal. Chapter 7 has more about the vision statement.	Use the vision statement as a benchmark to judge whether features belong in scope for the current project.	When someone introduces new requirements, those requirements must support the project vision statement.
Product Roadmap: A holistic view of product features that create the product vision. Chapter 7 has more about the product roadmap.	Product scope is part of the product roadmap. Requirements at a feature level are good for business conversations about what it means to realize the product vision.	Update the product roadmap as new requirements arise. The product roadmap provides visual communication of the new feature's inclusion in the project.
Release Plan: A digestible mid-term target focused around a minimum set of marketable features. Chapter 8 has more about the release plan.	The release plan shows the scope of the current release. You may want to plan your releases by themes — logical groups of requirements.	Add new features that belong in the current release to the release plan. If the new user story doesn't belong in the current release, leave it on the product backlog for a future release.
Product Backlog: A complete list of all known scope for the product. Chapters 7 and 8 offer more about the product backlog.	If a requirement is in scope, it is part of the product backlog.	The product backlog contains all scope changes. New, high-priority features push lower-priority features down on the product backlog.
Sprint Backlog: The user stories and tasks within the scope of the current sprint. Chapter 8 has more about the sprint backlog.	The sprint backlog contains the user stories that are in scope for the current sprint.	The sprint backlog establishes what is allowed in the sprint. Once the development team commits to the sprint goal in the sprint planning meeting, only the development team can modify the sprint backlog.

Table 13-1 Historical Versus Agile Time Management

Time Management with Traditional Approaches	*Time Management with Agile Approaches*
Fixed scope directly drives the schedule.	Scope is not fixed on agile projects. Time can be fixed, and development teams can create the requirements that will fit into a specific time frame.
Project managers determine time based on the requirements gathered at the beginning of the project.	During the project, scrum teams assess and reassess how much work they can complete in a given time frame.
Teams work on all project requirements at one time in phases, like requirements-gathering, design, development, testing, and deployment. There is no schedule difference between critical requirements and optional requirements.	Scrum teams work in sprints and complete all the work on the highest-priority, highest-value requirements first.
Teams do not start actual product development until later in the project, after the requirements-gathering and design phases are complete.	Scrum teams start product development in the very first sprint.
Time is more variable on traditional projects.	Time-boxed sprints on agile projects stay stable.
Project managers try to predict schedules at the project start, when they know little about the product.	Scrum teams determine long-range schedules on actual development performance in sprints. Scrum teams adjust time estimates throughout the project as they learn more about the product and the development team's speed, or *velocity.* You find more about velocity later in this chapter.

Figure 13-1: Multiple scrum teams on a project.





Using agile artifacts for time management

The product roadmap, release plan, product backlog, and sprint backlog all play a part in time management. Table 13-2 shows how each artifact contributes to time management.

Table 13-2 Agile Artifacts and Time Management Roles

Artifact	*Role in Time Management*
Product roadmap: The product roadmap is a prioritized, holistic view of the high-level requirements that support the product's vision. Find more about the product roadmap in Chapter 7.	The product roadmap is a strategic look at the overall project priorities. While the product roadmap likely will not have specific dates, it will have general date ranges for groups of functionality and will allow an initial framing for bringing the product to market.
Product backlog: The product backlog is a complete list of all currently known product requirements. Find more about the product backlog in Chapters 7 and 8.	The user stories in your product backlog will have estimated story points. Once you know your development team's velocity, you can use the total number of story points in the product backlog to determine a realistic project end date.
Release plan: The release plan contains a release schedule for a minimum set of requirements. Find more about the release plan in Chapter 8.	The release plan will have a target release date for a specific goal that is supported by a minimal set of marketable functionalities. Scrum teams only plan and work on one release at a time.
Sprint backlog: The sprint backlog contains the requirements and tasks for the current sprint. Find more about the sprint backlog in Chapter 8.	During your sprint planning meeting, you estimate individual tasks in the backlog in hours. At the end of each sprint, you take the total completed story points from the sprint backlog to calculate your development team's velocity for that sprint.

Table 13-3 Historical Versus Agile Cost Management

Cost Management with Traditional Approaches	*Cost Management with Agile Approaches*
Cost, like time, is based on fixed scope.	Project schedule, not scope, has the biggest impact on cost. You can start with a fixed cost and fixed amount of time, and complete requirements that fit into your budget and schedule.
Organizations estimate project costs and fund projects before the project starts.	Product owners often secure project funding after the product roadmap stage is complete. Some organizations even fund agile projects one release at a time; product owners will secure funding after completing release planning for each release.
New requirements mean higher costs. Because project managers estimate costs based on what they know at the project start, which is very little, cost overruns are common.	Project teams can replace lower-priority requirements with new, equivalently-sized high-priority requirements with no impact on time or cost.

Cost Management with Traditional Approaches	*Cost Management with Agile Approaches*
Scope bloat (see Chapter 12) wastes large amounts of money on features that people simply do not use.	Because agile development teams complete requirements by priority, they concentrate on creating only the product features that users really need, whether those features are added on day one or day 100 of the project.
Projects cannot generate revenue until the project is complete.	Project teams can release working, revenue-generating functionality early, creating a self-funding project.

Table 13-4 Sample Scrum Team Budget for a Two-Week Sprint

Team Member	*Hourly Rate*	*Weekly Hours*	*Weekly Cost*	*Sprint Cost (2 Weeks)*
Don	$80	40	$3,200	$6,400
Peggy	$70	40	$2,800	$5,600
Bob	$70	40	$2,800	$5,600
Mike	$65	40	$2,600	$5,200
Joan	$85	40	$3,400	$6,800
Tommy	$75	40	$3,000	$6,000
Pete	$55	40	$2,200	$4,400
Total		280	$20,000	$40,000

Table 13-5 Income from a Traditional Project with a Final Release After Six Months

Month	*Income Generated*	*Total Project Income*
January	$0	$0
February	$0	$0
March	$0	$0
April	$0	$0
May	$0	$0
June	$100,000	$100,000

Table 13-6 Income from a Project with Monthly Releases and a Final Release after Six Months

Month/Release	*Income Generated*	*Total Project Income*
January	$15,000	$15,000
February	$25,000	$40,000
March	$40,000	$80,000
April	$70,000	$150,000
May	$80,000	$230,000
June	$100,000	$330,000

Table 14-1 Historical Versus Agile Team Dynamics

Team Management with Traditional Approaches	*Team Dynamics with Agile Approaches*
Project teams rely on *command and control* — a top-down approach to project management, where the project manager is responsible for assigning tasks to team members and attempting to control what the team does.	Scrum teams are self-managing, self-organizing, and benefit from *servant leadership.* Instead of top-down management, a servant-leader coaches, removes obstacles, and prevents distractions to help the scrum team thrive.
Companies evaluate individual employee performance.	Agile organizations evaluate scrum team performance; every member of the scrum team receives the same review. Scrum teams, like any sports team, succeed or fail as a whole team.
Team members often find themselves working on more than one project at a time, switching their attention back and forth.	Development teams are dedicated to one project at a time, and reap the benefits of focus.
Development team members have distinct roles, like "programmer" or "tester."	Development teams work *cross*-functionally, doing different jobs within the team to ensure they complete priority requirements quickly.
Development teams have no specific size limits.	Development teams are intentionally limited in size. Ideally, development teams have seven, plus or minus two people.
People are commonly referred to as "resources," a shortened term for "human resources."	People are called "people" or "team members." On an agile project, you probably will not hear the term "resource" used to refer to people.

Table 14-2 Project Management and Self-Managing Teams

Area of Project Management	*How Development Teams Self-Manage*	*How Product Owners Self-Manage*	*How Scrum Masters Self-Manage*
Scope	May suggest features based on technical affinity. Work directly with the product owner to clarify requirements. Identify how much work they can commit to completing in a sprint. Identify the tasks to complete scope in the sprint backlog. Determine the best way to create specific features.	Use the product vision, the release goal, and each sprint goal to determine if and where scope items belong. Use product backlog prioritization to determine which requirements are developed.	Remove impediments that limit the amount of scope the development team can create. Through coaching, help development teams become more productive with each successive sprint.
Procurement	Identify the tools they need to create the product. Work with the product owner to get those tools.	Secure necessary funding for tools and equipment for development teams.	Help procure tools and equipment that accelerate development team velocity.
Time	Provide effort estimates for product features. Identify what features they can create in a given time frame — the sprint. Often provide time estimates for tasks in each sprint. Choose their own schedules and manage their own time.	Ensure that the development team correctly understands product features so that development teams can correctly estimate the effort to create those features. Use velocity — development speed — to forecast long-term timelines.	Facilitate estimation poker games. Help development teams increase velocity, which affects time. Protect team from organizational time-wasters and distractions.

(continued)

Table 14-2 *(continued)*

Area of Project Management	*How Development Teams Self-Manage*	*How Product Owners Self-Manage*	*How Scrum Masters Self-Manage*
Cost	Provide effort estimates for product features.	Ultimately responsible for the budget and return on investment on an agile project. Use velocity to forecast long-term costs, based on timelines.	Facilitate estimation poker games. Help development teams increase velocity, which affects cost.
Team dynamics	Prevent bottlenecks by working cross-functionally, and are willing to take on different types of tasks. Continuously learn and teach one another. Commit, both individually and as part of the scrum team, to their projects and to one another. Strive to build consensus when making important decisions.	Commit to their projects and are integrated members of the scrum team.	Facilitate scrum team collocation. Help remove impediments to scrum team self-management. Commit to their projects and are integrated members of the scrum team. Strive to build consensus within the scrum team when making important decisions. Facilitate relationships between the scrum team and stakeholders.
Communication	Report on progress, upcoming tasks, and identify roadblocks in their daily scrum meetings. Keep the sprint backlog up-to-date daily, providing accurate, immediate information about a project's status. Present working functionality to project stakeholders at the sprint review meetings at the end of each sprint.	Communicate information about the product and the business needs to development teams on an ongoing basis. Communicate information about the project progress to product stakeholders. Help present working functionality to stakeholders at the sprint review meetings at the end of each sprint.	Encourage face-to-face communication between all scrum team members. Foster close cooperation between the scrum team and other departments within the company or organization.

Area of Project Management	*How Development Teams Self-Manage*	*How Product Owners Self-Manage*	*How Scrum Masters Self-Manage*
Quality	Commit to providing technical excellence and good design. Test their work throughout the day and comprehensively test all development each day. Inspect their work and adapt for improvements at sprint retrospective meetings at the end of each sprint.	Add acceptance criteria to requirements. Ensure that the development team correctly understands and interprets requirements. Provide development teams with feedback about the product from the organization and from the marketplace. Accept features as Done during each sprint.	Help facilitate the sprint retrospective. Help ensure face-to-face communication between scrum team members, which in turn helps ensure quality work. Help create a sustainable development environment so that the development team can perform at its best.
Risk	Identify and develop the risk mitigation approach for each sprint. Alert the scrum master to roadblocks and distractions. Use information from each sprint retrospective to reduce risk in future sprints. Embrace cross-functionality to reduce risk if one member unexpectedly leaves the team. Commit to delivering shippable functionality at the end of each sprint, reducing risk in the overall project.	Look at overall project risks as well as risks to their ROI commitment.	Help prevent roadblocks and distractions. Help remove roadblocks and identified risks. Facilitate development team conversations about possible risks.

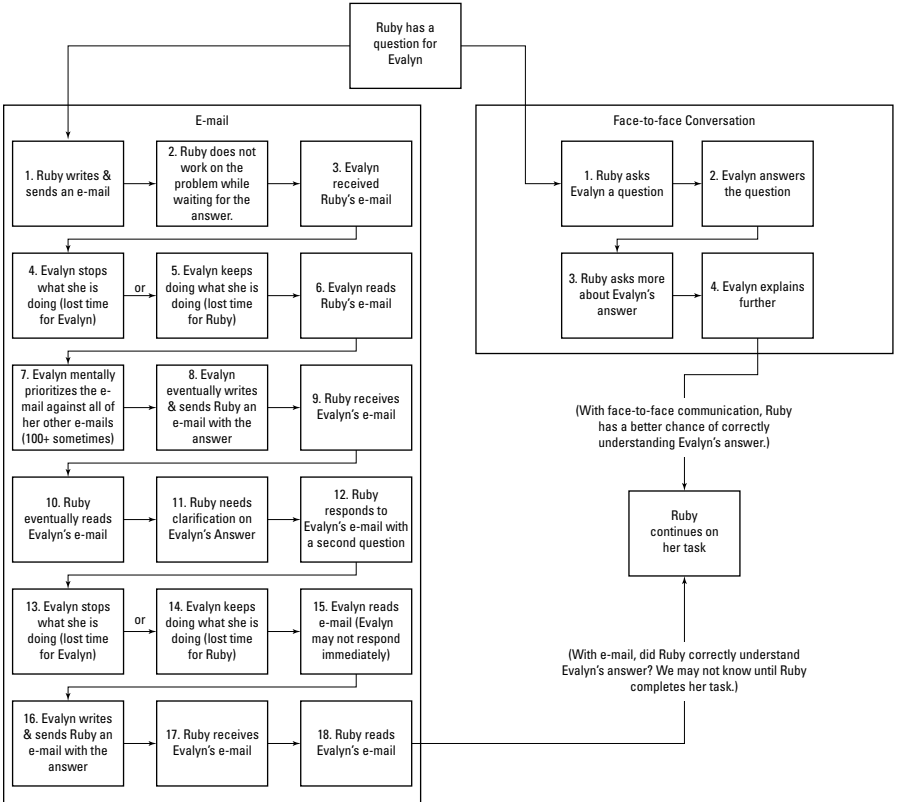


Figure 14-1: E-mail versus face-to-face conversation.

Table 14-3 Success of Collocated and Dislocated Scrum Teams

Team Location	*Success Percentage*
Collocated scrum team	83%
Dislocated but physically reachable	72%
Distributed across geographies	60%

Agile Adoption Rate Survey Results (Scott W. Ambler, Ambysoft, Copyright© 2008)

Table 14-4 Historical Versus Agile Communication

Communication Management with Traditional Approaches	*Communication Management with Agile Approaches*
Team members might make no special effort for in-person conversations.	Agile project management approaches value face-to-face communication as the best way to convey information.
Traditional approaches place high value on documentation. Teams may create a large number of complex documents and status reports based on process, rather than considering actual need.	Agile documents, or *artifacts*, are intentionally simple and provide information that is barely sufficient. Agile artifacts only contain essential information and can often convey project status at a glance. Project teams use the *show, don't tell* concept, showing working software to communicate progress on a regular basis in the sprint review.
Team members may be required to attend a large number of meetings, whether or not those meetings are useful or necessary.	Meetings on agile projects are, by design, as quick as possible and will include only people who will truly add to the meeting and benefit from the meeting. Agile meetings provide all the benefits of face-to-face communication without wasting time. The structure of agile meetings is to enhance, not reduce, productivity.

Figure 14-2, from Alistair Cockburn's presentation *Software Development as a Cooperative Game* (Copyright Humans and Technology, Inc.), shows the effectiveness of face-to-face communication versus other types of communication.

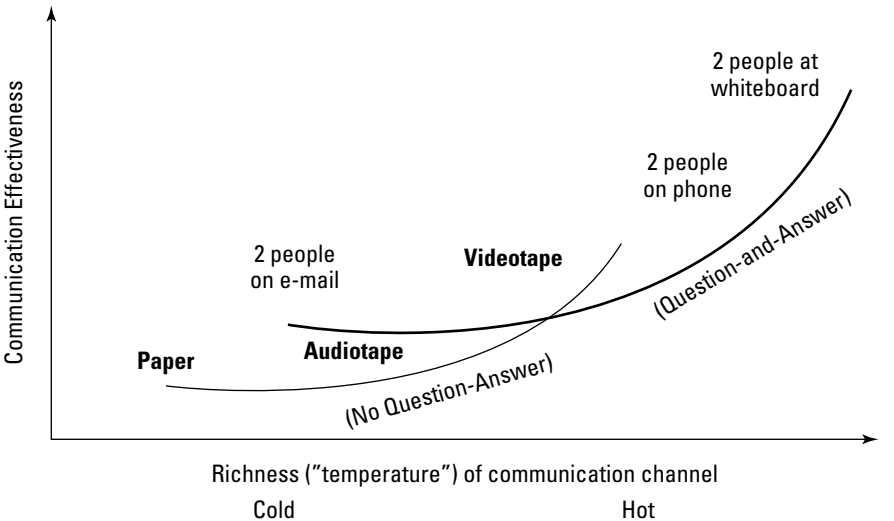


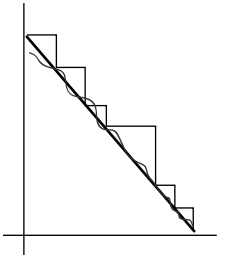
Figure 14-2: Comparison of communication types.

Table 14-5 provides a view of the different communication channels on an agile project.

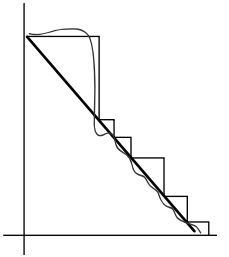
Table 14-5 Agile Project Communication Channels

Channel	*Type*	*Role in Communication*
Project planning, release planning, and sprint planning	Meetings	Planning meetings communicate the details of the project, the release, and the sprint to the scrum team. Learn more about planning meetings in Chapters 7 and 8.
Product vision statement	Artifact	The product vision statement communicates the end goal of the project to the project team and the organization. Find out more about the product vision in Chapter 7.
Product roadmap	Artifact	The product roadmap communicates a long-term view of the features that support the product vision and are likely to be part of the project. Find out more about the product roadmap in Chapter 7.
Product backlog	Artifact	The product backlog communicates the scope of the project as a whole to the project team. Find out more about the product backlog in Chapters 7 and 8.
Release plan	Artifact	The release plan communicates the goals for a specific release. Find out more about the release plan in Chapter 8.
Sprint backlog	Artifact	When updated daily, the sprint backlog provides immediate sprint and project status to anyone who needs that information. The burndown chart on the sprint backlog provides a quick visual of the sprint status. Find out more about the sprint backlog in Chapters 8 and 9.

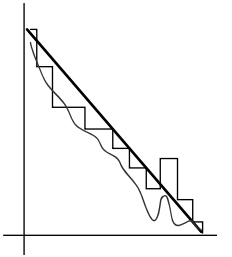
Channel	*Type*	*Role in Communication*
Task board	Artifact	Using a task board visually radiates out status of the current sprint or release to anyone who walks by the scrum team's work area. Find out more about the task board in Chapter 9.
Daily scrum	Meeting	The daily scrum provides the scrum team with a verbal, face-to-face opportunity to coordinate the priorities of the day and identify any challenges. Find out more about daily scrum meetings in Chapter 9.
Face-to-face conversations	Informal	Face-to-face conversations are the most important mode of communication on an agile project.
Sprint review	Meeting	The sprint review is the embodiment of the show, don't tell philosophy. Displaying working software to the entire project team conveys project progress in a more meaningful way than a report ever could. Find out more about sprint reviews in Chapter 10.
Sprint retrospective	Meeting	The sprint retrospective allows the scrum team to communicate with one another specifically for improvement. Find out more about sprint retrospectives in Chapter 10.
Meeting notes	Informal	Meeting notes are an optional, informal communication method on an agile project. Meeting notes can capture action items from a meeting to ensure people on the scrum team remember them for later. Notes from a sprint review can record new features for the product backlog. Notes from a sprint retrospective can remind the scrum team of commitments for improvement.
Collaborative solutions	Informal	White boards, sticky notes, and electronic collaboration tools all help the scrum team communicate. Ensure that these tools augment, rather than replace, face-to-face conversations.



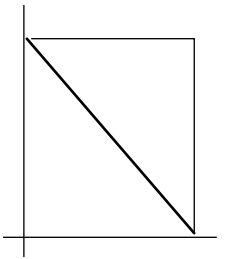
1. Expected



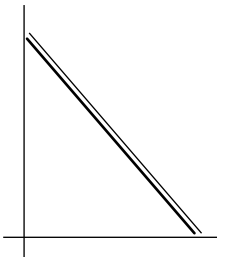
2.More Complicated



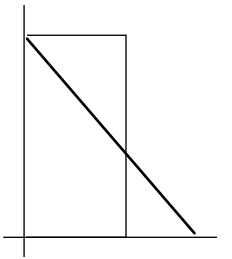
3. Less Complicated



4. Not Participating



5. Lying



6. Failing Fast

Figure 14-3: Profiles of burndown charts.

Table 15-1 shows some differences between quality management on traditional projects and on agile projects.

Table 15-1 Historical Versus Agile Quality

Quality Management with Traditional Approaches	*Quality Dynamics with Agile Approaches*
Testing is the last phase of a project before product deployment. Some features are tested months after they were created.	Testing is a daily part of each sprint and is included in each requirement's definition of done. You use automated testing, allowing quick and robust testing every day.
Quality is often a reactive practice, with focus mostly on product testing and issue resolution.	You address quality both reactively, through testing, and proactively, encouraging practices to set the stage for quality work. Examples of proactive quality approaches include face-to-face communication, pair programming, and established coding standards.
Problems are riskier when found at the end of a project. Sunk costs are high by the time teams reach testing.	You can create and test riskier features in early sprints, when sunk costs are still low.
Problems, sometimes called *bugs,* are hard to find at the end of a project, and fixes for problems at the end of a project are costly.	Problems are easy to find when you test a smaller amount of work. Fixes are easier when you fix something you just created, rather than something you created months earlier.
Sometimes, in order to meet a deadline or save money, teams cut the testing phase short.	Testing is assured on agile projects, because it is part of every sprint.

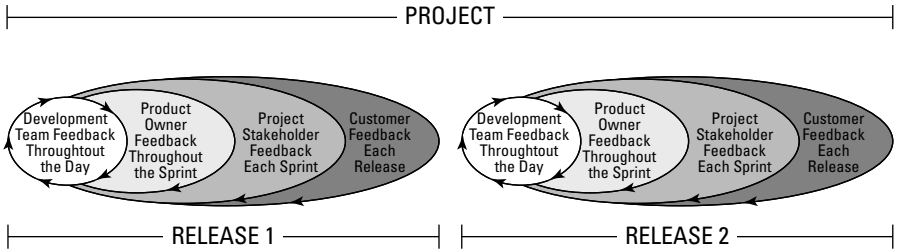


Figure 15-1: Quality feedback in an agile project.

Figure 15-2: Testing within sprints.

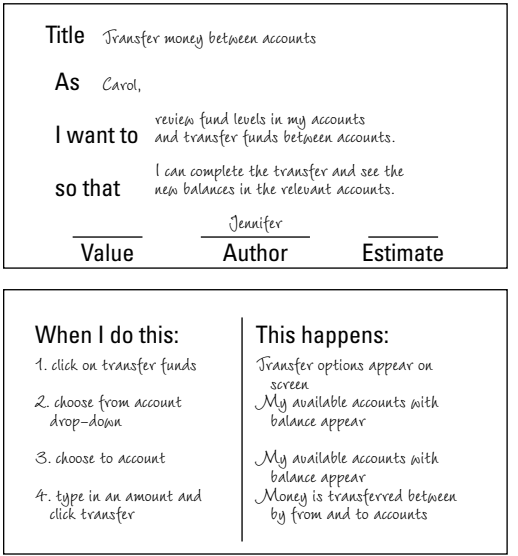


Figure 15-3: A user story and acceptance criteria.

Table 15-2 Historical Versus Agile Risk

Risk Management with Traditional Approaches	*Risk Dynamics with Agile Approaches*
Large numbers of projects fail or are challenged.	Risk of catastrophic failure — spending large amounts of money with nothing to show — is almost eliminated.
The bigger, longer, and more complex the project, the more risky it is. Risk is highest at the end of a project.	You gain product value immediately, rather than sinking costs into a project for months or even years with the growing chance of failure.
Conducting all the testing at the end of a project means that finding serious problems can put the entire project at risk.	You test at the same time you develop. If a technical approach, a requirement, or even an entire product is not feasible, the development team discovers this in a short time, and you have more time to course correct. If correction is not possible, stakeholders spend less money on a failed project.
Projects are unable to accommodate new requirements mid-project without increased time and cost because there is extensive sunk cost in even the lowest-priority requirements.	You welcome change for the benefit of the product. Agile projects accommodate new high-priority requirements without increasing time or cost by removing a low-priority requirement of equal time and cost.
Traditional projects require time and cost estimates at the project start, when teams know the least about the project. Estimates are often inaccurate, creating a gap between expected and actual project schedules and budgets.	You can estimate project time and cost using the scrum team's actual performance, or velocity. You refine estimates throughout the project, because the longer you work on a project, the more you learn about the project, the requirements, and the scrum team.
When stakeholders do not have a unified goal, they can end up confusing the project team with conflicting information about what the product should achieve.	You have a single product owner, who is responsible for creating a vision for the product and represents the stakeholders to the project team.
Unresponsive or absent stakeholders can cause project delays and result in products that do not achieve the right goals.	The product owner is responsible for providing information about the product immediately. You also have a scrum master, who helps remove impediments on a daily basis.

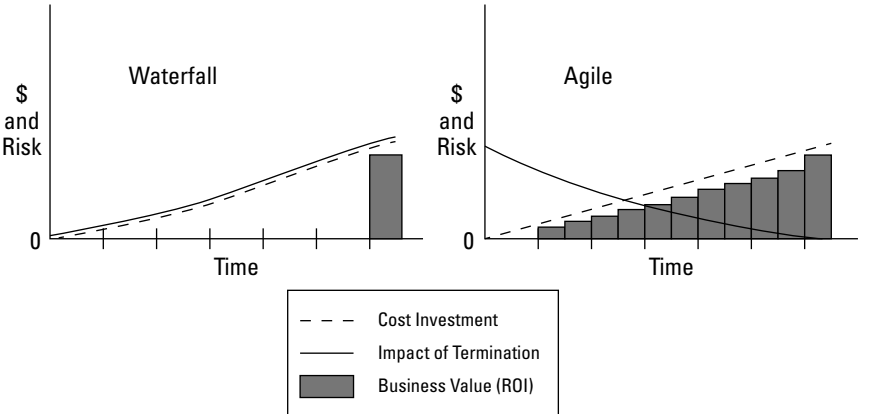


Figure 15-4: Agile projects' declining risk model.

Figure 15-5 shows a sample definition of done, with details.

Definition of Done

Sprint	Release	Risks Accepted
QA Environment	Staging Environment	Load Testing
Unit Tested	Performance Tested	
Functional Tested	Security Tested	
Integration Tested	Enterprise System Integrated	
User Acceptance Tested	Focus Group Tested	
Regression Tested	User Documentation	
XDocs	Training Documentation	

Figure 15-5: Sample definition of done.

Table 15-3 Income from a Traditional Project with a Final Release after Six Months

Month	*Income Generated*	*Total Project Income*
January	$0	$0
February	$0	$0
March	$0	$0
April	$0	$0
May	$0	$0
June	$100,000	$100,000

In Table 15-3, the project created $100,000 in income after six months of development. Now compare the ROI in Table 15-3 to the ROI in Table 15-4.

Table 15-4 Income from an Agile Project with Monthly Releases and a Final Release After Six Months

Month/Release	*Income Generated*	*Total Project Income*
January	$15,000	$15,000
February	$25,000	$40,000
March	$40,000	$80,000
April	$70,000	$150,000
May	$80,000	$230,000
June	$100,000	$330,000

In Table 15-4, the project generated income with the very first release. By the end of six months, the project had generated $330,000 — $230,000 more than the project in Table 15-3.

Table 15-5	Cost of Failure on a Waterfall Project		
Month	***Phase and Issues***	***Sunk Project Cost***	***Total Sunk Project Cost***
January	Requirements Phase	$80,000	$80,000
February	Requirements Phase	$80,000	$160,000
March	Design Phase	$80,000	$240,000
April	Design Phase	$80,000	$320,000
May	Design Phase	$80,000	$400,000
June	Development Phase	$80,000	$480,000

(continued)

Table 15-5 *(continued)*

Month	*Phase and Issues*	*Sunk Project Cost*	*Total Sunk Project Cost*
July	Development Phase	$80,000	$560,000
August	Development Phase	$80,000	$640,000
September	Development Phase	$80,000	$720,000
October	QA Phase: Large-scale problem uncovered during testing.	$80,000	$800,000
November	QA Phase: Development team attempted to resolve problem to continue development.	$80,000	$880,000
December	Project cancelled; product not viable.	0	$880,000

In Table 15-5, the project stakeholders spent six months and close to a million dollars to find out that a product idea would not work. Compare the sunk cost in Table 15-5 to that in Table 15-6.

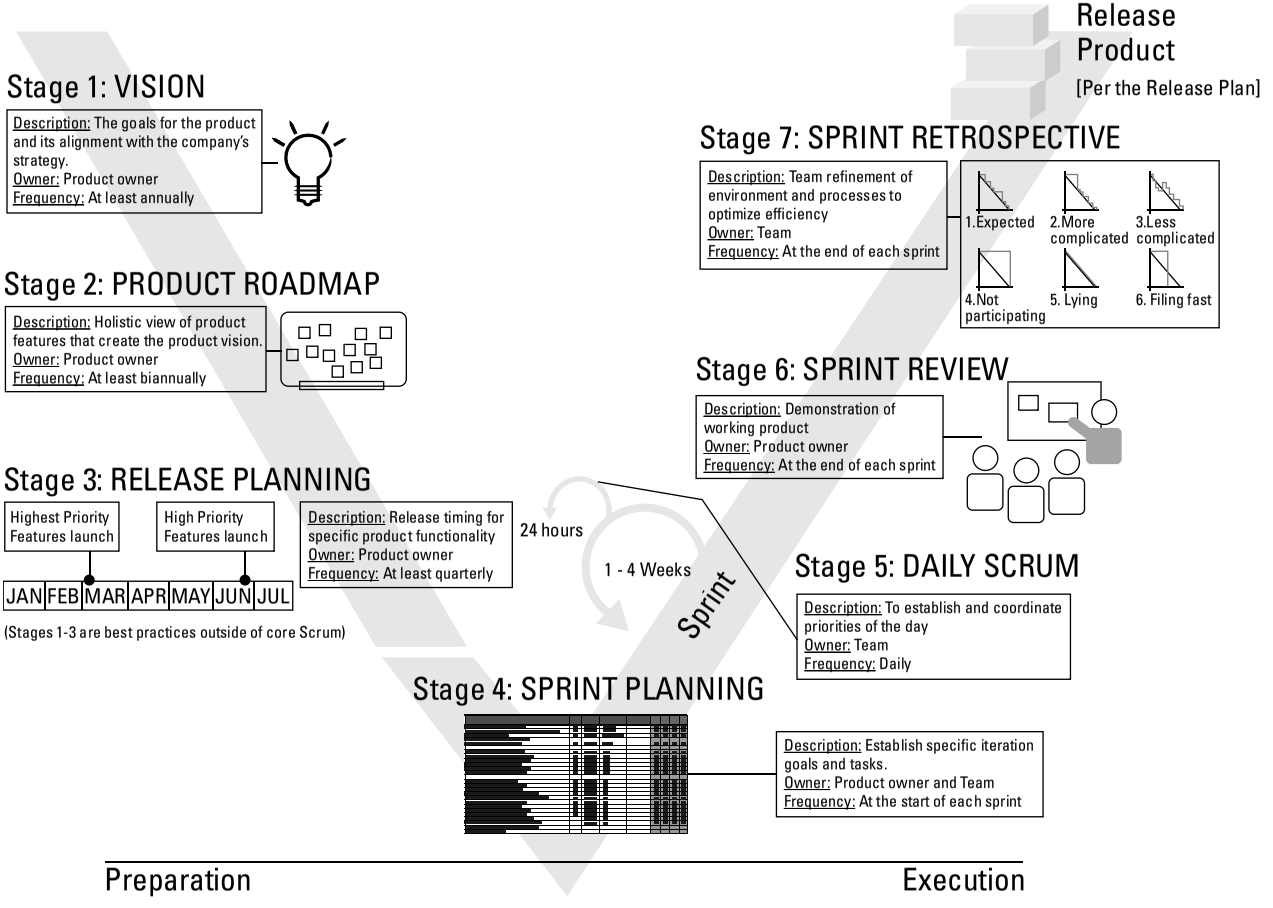
Table 15-6 Cost of Failure on an Agile Project

Month	*Sprint and Issues*	*Sunk Project Cost*	*Total Sunk Project Cost*
January	Sprint 1: No issues. Sprint 2: No issues.	$80,000	$80,000
February	Sprint 3: Large-scale problem uncovered during testing resulted in failed sprint; sprint still failed. Sprint 4: Development team attempted to resolve problem to continue development; sprint ultimately failed.	$80,000	$160,000
Final	Project cancelled; product not viable.	0	$160,000

Table 15-7 Agile Project Risk Management Tools

Artifact or Meeting	*Role in Risk Management*
Product vision	The product vision statement helps unify the project team's definition of product goals, mitigating the risk of misunderstandings about what the product will need to accomplish. While creating the product vision, the project team might think of risks on a very high level, in conjunction with the marketplace, customers, and organizational strategy. Find out more about the product vision in Chapter 7.
Product roadmap	The product roadmap provides a visual overview of the project's requirements and priorities. This visual overview allows the project team to quickly identify gaps in requirements and incorrectly prioritized requirements. Find out more about the product roadmap in Chapter 7.
Product backlog	The product backlog is a tool for accommodating change within the project. Being able to add changes to the product backlog and reprioritize requirements regularly helps turn the traditional risk associated with scope changes into a way to create a better product. Keeping the requirements and the priorities on the product backlog current helps ensure the development team can work on the most important requirements at the right time. Find out more about the product backlog in Chapters 7 and 8.
Release planning	During release planning, the scrum team discusses risks to the release and how to mitigate those risks. Risk discussions in the release planning meeting should be high-level and relate to the release as a whole. Save risks to individual requirements for the sprint planning meetings. Find out more about release planning in Chapter 8.
Sprint planning	During each sprint planning meeting, the scrum team discusses risks to the specific requirements and tasks in the sprint and how to mitigate those risks. Risk discussions during sprint planning can be done in depth, but should only relate to the current sprint. Find out more about sprint planning in Chapter 8.
Sprint backlog	The burndown chart on the sprint backlog provides a quick view of the sprint status. This quick view helps the scrum team manage risks to the sprint just as they arise and minimize impact by addressing problems immediately. Find out more about sprint backlogs and how burndown charts show project status in Chapter 9.

Figure 16-1: The Agile Roadmap to Value.



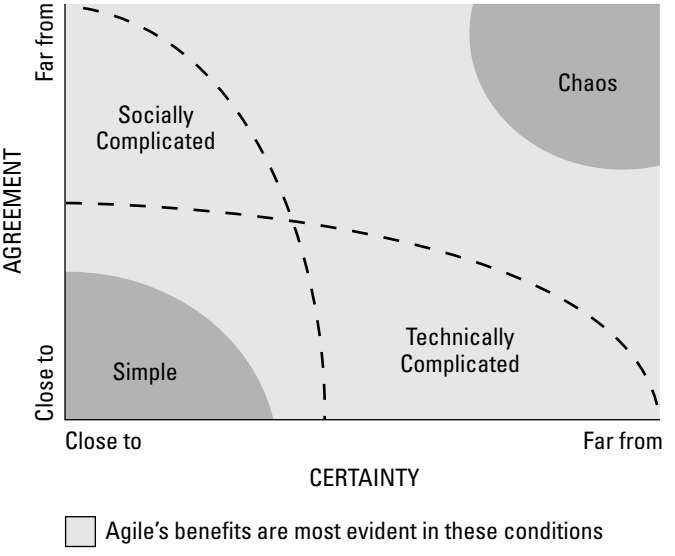


Figure 17-1: Projects that can benefit from agile techniques.

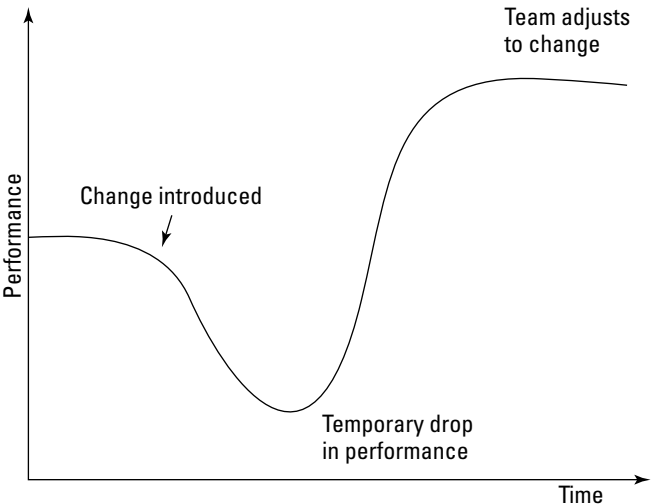


Figure 17-2: Satir's Curve.

Table 17-1 Common Agile Transition Problems and Solutions

Problem	*Description*	*Potential Solution*
Faux agile: Cargo cult agile and double work agile	Sometimes organizations will say that they are "doing agile." They may go through some of the practices used on agile projects, but they haven't embraced the principles of agile and are ultimately creating waterfall deliverables and products. This is sometimes called *cargo cult agile* and is a sure path to avoiding the benefits of agile techniques. Trying to complete agile processes in addition to waterfall processes, documents, and meetings is another faux agile approach. *Double work agile* results in quick project team burnout. If you're doing twice the work, you aren't adhering to Agile Principles.	Insist on following one process — an agile process. Garner support from management to avoid non-agile principles and practices.
Lack of training	Investment in a hands-on training class will provide a quicker, better learning environment than even the best book, blog, or white paper. Lack of training often indicates an overall lack of organizational commitment to agile practices. Keep in mind that training can help scrum teams avoid many of the mistakes on this list.	Build training into your implementation strategy. Giving teams the right foundation of skills is critical to success and necessary at the start of your agile transition.

Problem	*Description*	*Potential Solution*
Ineffective product owner	No role is more different than traditional roles than that of the product owner. Agile project teams need a product owner who is an expert on business needs and priorities and can work well with the rest of the scrum team on a daily basis. An absent or indecisive product owner will quickly sink an agile project.	Start the project with a person who has the time, expertise, and temperament to be a good product owner. Ensure the product owner has proper training. The scrum master can help coach the product owner and may try to clear roadblocks preventing the product owner from being effective. If removing impediments doesn't work, the scrum team should insist on replacing the ineffective product owner with a product owner — or at least an agent — who can make product decisions and help the scrum team be successful.
Lack of automated testing	Without automated testing, it may be impossible to fully complete and test work within a sprint. Manual testing is a waste of time that fast-moving scrum teams don't have.	You can find many low-cost, open-source testing tools on the market today. Look into the right tools and make a commitment as a development team to using those tools.
Lack of transition support	Making the transition successfully is difficult and far from guaranteed. It pays to do it right the first time with people who know what they are doing.	When you decide to move to agile project management, enlist the help of an agile mentor — either internally from your organization or externally from a consulting firm — who can support your transition. Process is easy, but people are hard. It pays to invest in professional transition support with an experienced partner who understands behavioral science and organizational change.

(continued)

Table 17-1 ***(continued)***

Problem	*Description*	*Potential Solution*
Inappropriate physical environment	When scrum teams are not collocated, they lose the advantage of face-to-face communication. Being in the same building isn't enough; scrum teams need to sit together in the same area.	If your scrum team is in the same building but not sitting in the same area, move the team together. Consider creating a room or annex for the scrum team. Try to keep the scrum team area away from distracters, such as the guy who can talk forever or the manager who needs just one small favor. Before starting a project with a dislocated scrum team, do what you can to enlist local talent. If you must work with a dislocated scrum team, take a look at Chapter 14 to see how to manage dislocated teams.
Poor team selection	Scrum team members who don't support agile processes, who don't work well with others, or who don't have capacity for self-management will sabotage a new agile project from within.	When creating a scrum team, consider how well potential team members will enact the Agile Principles. The keys are versatility and a willingness to learn.
Discipline slips	Remember that agile projects still need requirements, design, development, testing, and releases. Doing that work in sprints requires discipline.	You need more, not less, discipline to deliver working products in a short iteration. Progress needs to be consistent and constant. The daily scrum helps ensure progress is occurring throughout the sprint. Use the sprint retrospective as an opportunity to reset approaches to discipline.

Problem	*Description*	*Potential Solution*
Lack of support for learning	Scrum teams succeed as teams and fail as teams; calling out one person's mistakes (known as the *blame game*) destroys the learning environment and destroys innovation.	The scrum team can make a commitment at the project start to leaving room for learning and to accepting success and failures as a group.
Diluting until dead	Watering down agile processes with old waterfall habits erodes the benefits of agile processes until those benefits no longer exist.	When making process changes, stop and consider whether those changes support the Agile Manifesto and the Agile Principles. Resist changes that don't work with the manifesto and principles. Remember to maximize work not done.

Table 19-1 ROI on a Traditional Project

Month	*Monthly Income*	*Monthly Costs*	*Monthly ROI*	*Total Income*	*Total Costs*	*Total ROI*
January	$0	$80,000	–$80,000	$0	$80,000	–$80,000
February	$0	$80,000	–$80,000	$0	$160,000	–$160,000
March	$0	$80,000	–$80,000	$0	$240,000	–$240,000
April	$0	$80,000	–$80,000	$0	$320,000	–$320,000
May	$0	$80,000	–$80,000	$0	$400,000	–$400,000
June (project launch)	$100,000	$80,000	$20,000	$100,000	$480,000	–$380,000
July	$100,000	$0	$100,000	$200,000	$480,000	–$280,000
August	$100,000	$0	$100,000	$300,000	$480,000	–$180,000
September	$100,000	$0	$100,000	$400,000	$480,000	–$80,000
October (break-even)	$100,000	$0	$100,000	$500,000	$480,000	$20,000
November	$100,000	$0	$100,000	$600,000	$480,000	$120,000
December	$100,000	$0	$100,000	$700,000	$480,000	$220,000