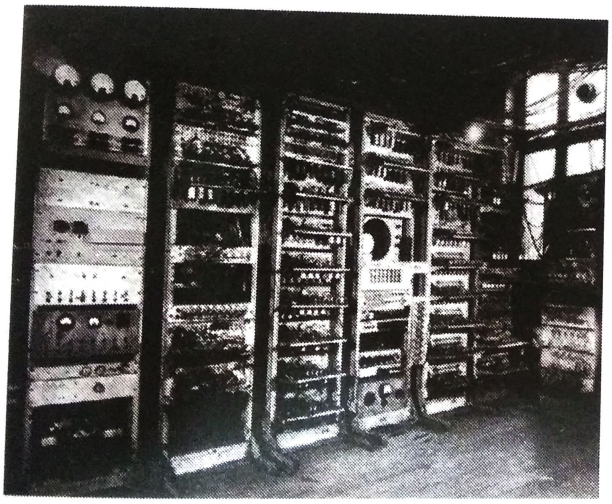
Agile project management is an empirical project management approach. In other words, you do something in practice and adjust your approach based on experience rather than theory.

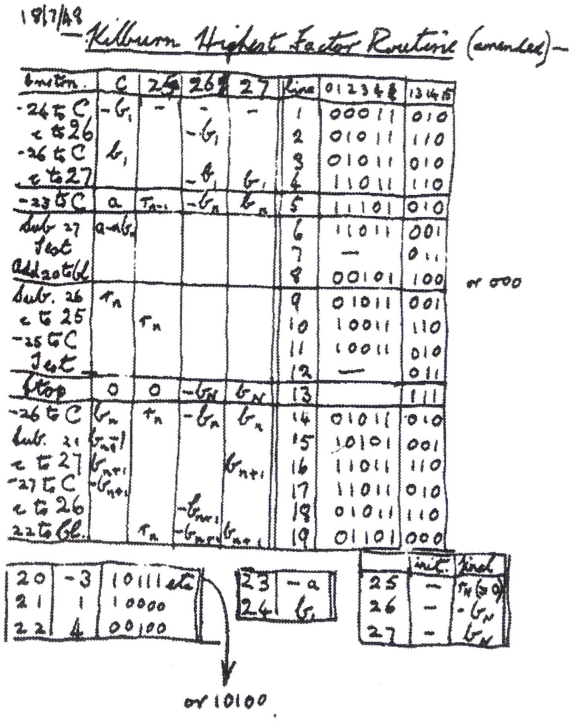
With regards to product development, the empirical approach is braced by the following pillars:

- **Unfettered transparency:** Everyone involved in the process understands and can contribute to the development of the process.
- **Frequent inspection:** The inspector must inspect the product regularly and possess the skills to identify variances from acceptance criteria.
- **Immediate adaptation:** The development team must be able to adjust quickly to minimize further product deviations.

computer systems, which were mostly hardware: 1,600 vacuum tubes but only 30 or so lines of hand-coded software. (See Figure 4-1.) An inflexible process is effective when the problems are simple and the marketplace is static, but today's product development environment is too complex for such an outdated model.



SSEM — "Baby"



Tom Kilburn's program for "Baby"

FIGURE 4-1: Early hardware and software.

Enter Dr. Winston Royce. In his article, "Managing the Development of Large Systems," published in 1970, Dr. Royce codified the step-by-step software development process known as waterfall. When you look at his original diagram in Figure 4-2, you can see where that name came from.

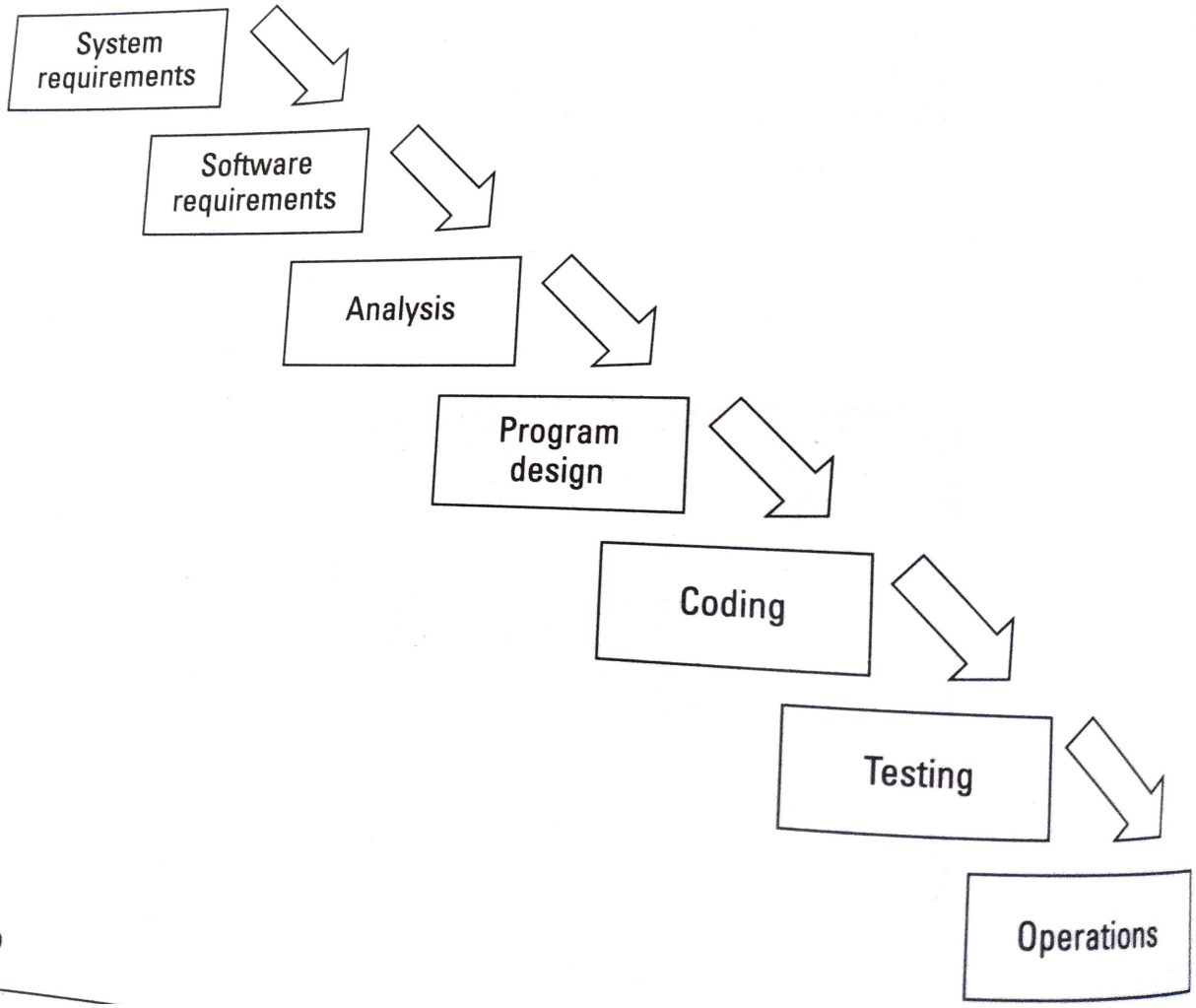


FIGURE 4-2: The origins of waterfall.

The irony here is that, even though the diagram implies that you complete tasks step by step, Dr. Royce himself added the cautionary note that you need iteration. Here's how he stated it:

> "If the computer program in question is being developed for the first time, arrange matters so that the version being delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned."

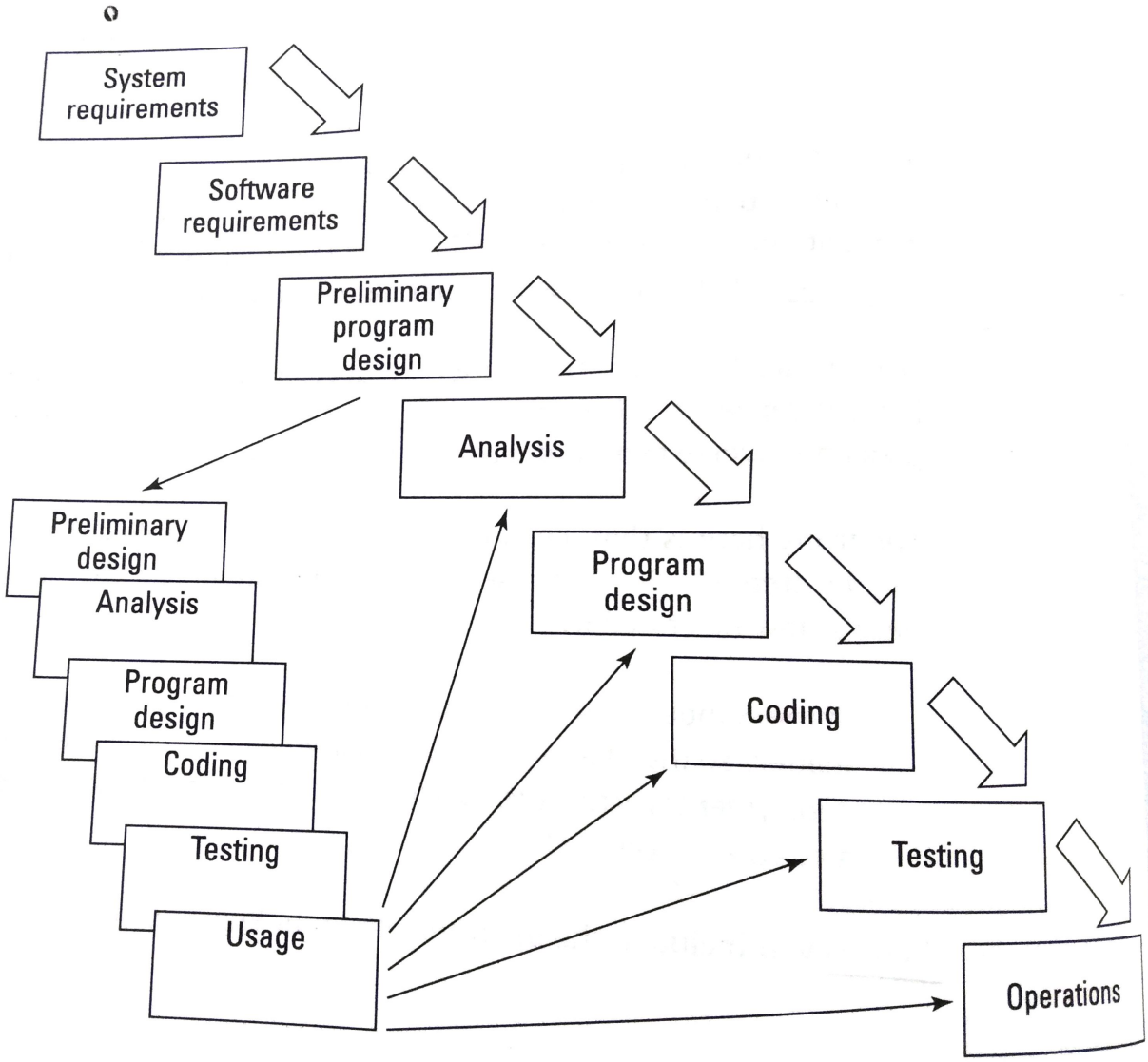


FIGURE 4-3: Iteration in waterfall.

Reviewing the Big Three: Lean, Scrum, and Extreme Programming

A host of approaches have agile characteristics. Three, however, are common to many agile projects: lean product development, scrum, and extreme programming (XP). These three approaches work perfectly together and share many common elements, although they use different terminology or have a slightly different focus. Broadly, lean and scrum focus on structure. Extreme programming does that, too, but is more prescriptive about development practices, focusing more on technical design, coding, testing, and integration. (From an approach called *extreme programming*, this type of focus is to be expected.)

An overview of lean

Lean has its origins in manufacturing. Mass production methods, which have been around for more than 100 years, were designed to simplify assembly processes (for example, putting together a Model-T Ford). These processes use complex, expensive machinery and low-skilled workers to inexpensively churn out an item of value. The idea is that if you keep the machines and people working and stockpile inventory, you generate a lot of efficiency.

Cutting the fat as lean emerges in manufacturing

In the 1940s in Japan, a small company called Toyota wanted to produce cars for the Japanese market but couldn't afford the huge investment that mass production requires. The company studied supermarkets, noting how consumers buy just what they need because they know there will always be a supply and how the stores restock shelves only as they empty. From this observation, Toyota created a just-in-time process that it could translate to the factory floor.

The result was a significant reduction in inventory of parts and finished goods and a lower investment in machines, people, and space.

Understanding lean and software development

The term *lean* was coined in the 1990s in *The Machine That Changed the World: The Story of Lean Production* (Free Press) by James P. Womack, Daniel T. Jones, and Daniel Roos. eBay was an early adopter of lean principles for software development. The company led the way with an approach that responded daily to customers' requests for changes to the website, developing high-value features in a short time.

The focus of lean is business value and minimizing activities outside product development. Mary and Tom Poppendieck discuss a group of lean principles on their blog and in their books on lean software development. Following are the lean principles from their 2003 book, *Lean Software Development* (Addison-Wesley Professional):

- **Eliminate waste.** Doing anything that is beyond barely sufficient (process steps, artifacts, meetings) slows down the flow of progress. Waste includes failing to learn from work, building the wrong thing, and thrashing (context switching between tasks or projects) — which results in only partially creating lots of product features but not completely creating any.
- **Amplify learning.** Learning drives predictability. Enable improvement through a mindset of regular and disciplined transparency, inspection, and adaptation. Encourage an organization-wide culture that allows failure for the sake of learning from it.
- **Deliver as late as possible.** Allow for late adaptation. Don't deliver late, but leave your options open long enough to make decisions at the last responsible moment based on facts rather than uncertainty — when you know the most. Learn from failure. Challenge standards. Use the scientific method — experiment with hypotheses to find solutions.
- **Deliver as fast as possible.** Speed, cost, and quality are compatible. The sooner you deliver, the sooner you receive feedback. Work on fewer things at once, limiting work in progress and optimizing flow. Manage workflow, rather than schedules. Use just-in-time planning to shorten development and release cycles.

- **Empower the team.** Working autonomously, mastering skills, and believing in the purpose of work can motivate development teams. Managers do not tell developers how to do their jobs but instead support them to self-organize around the work to be done and remove their impediments. Make sure teams and individuals have the environment and tools they need to do their job well.
- **Build quality in.** Establish mechanisms to catch and correct defects when they happen and before final verification. Quality is built in from the beginning, not at the end. Break dependencies, so you can develop and integrate functionality at any time without regressions.
- **See the whole.** An entire system is only as strong as its weakest link. Solve problems, not just symptoms. Continually pay attention to bottlenecks throughout the flow of work and remove them. Think long-term when creating solutions.

lean approaches used

An overview of scrum

Scrum, the most popular agile framework in software development, is an iterative approach that has at its core the *sprint* (the scrum term for iteration). To support this process, scrum teams use specific roles, artifacts, and events. To make sure that they meet the goals of each part of the process, scrum teams use inspection and adaptation throughout the project. The scrum approach is shown in Figure 4-4.

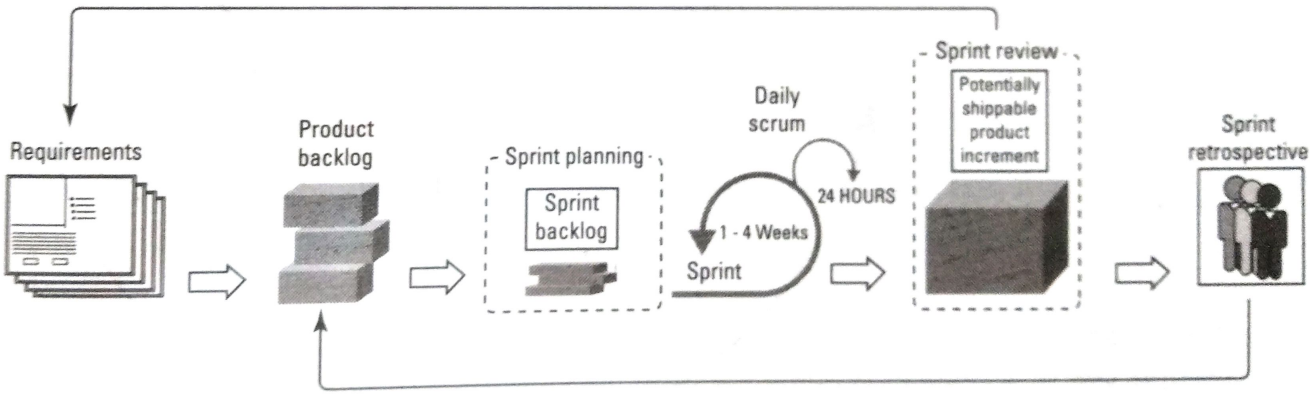


FIGURE 4-4: The scrum approach.

Going the distance with the sprint

Within each sprint, the development team develops and tests a functional part of the product until the product owner accepts it, often daily, and the functionality becomes a potentially shippable increment of the overall product. When one sprint finishes, another sprint starts. Releasing functionality to the market often occurs at the end of multiple sprints, when the product owner determines that enough value exists. However, the product owner may decide to release functionality after every sprint, or even as many times as needed during a sprint.

A core principle of the sprint is its cyclical nature: The sprint, as well as the processes within it, repeats over and over, as shown in Figure 4-5.

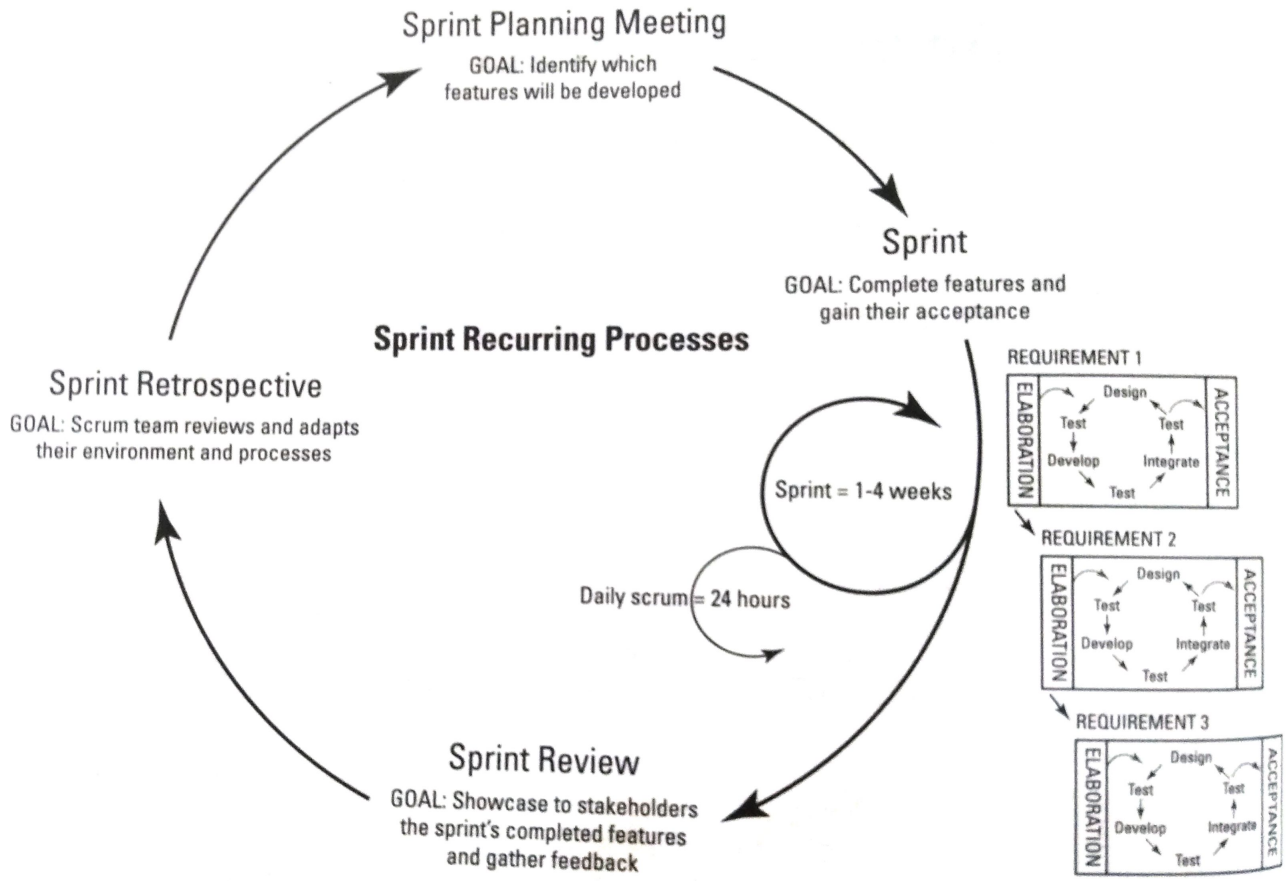


FIGURE 4-5: Sprints are recurring processes.

Understanding scrum roles, artifacts, and events

The scrum framework defines specific roles, artifacts, and events for projects.

Scrum's three *roles* — people on the project — are as follows:

- **Product owner:** Represents and speaks for the business needs of the project.
- **Development team:** Performs the day-to-day work. The development team is dedicated to the project and each team member is *multi-skilled* — that is, although team members may have certain strengths, each member is capable of doing multiple jobs on the project.
- **Scrum master:** Protects the team from organizational distractions, clears roadblocks, ensures that scrum is played properly, and continuously improves the team's environment.

work closely with two non-scrum-specific roles:

- **Stakeholders:** Anyone who is affected by or has input on the project. Although stakeholders are not official scrum roles, it is essential for scrum teams and stakeholders to work closely together throughout a project.
- **Agile mentor:** An experienced authority on agile techniques and the scrum framework. Often this person is external to the project's department or organization, so he or she can support the scrum team objectively with an outsider's point of view.

In the same way that scrum has specific roles, scrum also has three tangible deliverables, called *artifacts:*

- **Product backlog:** The full list of requirements that defines the product, often documented in terms of business value from the perspective of the end user. The product backlog can be fluid throughout the project. All scope items, regardless of level of detail, are in the product backlog. The product owner owns the product backlog, determining what goes in it and in what priority.
- **Sprint backlog:** The list of requirements and tasks in a given sprint. The product owner and the development team select the requirements for the sprint in sprint planning, with the development team breaking down these requirements into tasks. Unlike the product backlog, the sprint backlog can be changed only by the development team.
- **Product increment:** The usable, potentially shippable functionality. Whether the product is a website or a new house, the product increment should be complete enough to demonstrate its working functionality. A scrum project is complete after a product contains enough shippable functionality to meet the customer's business goals for the project.

Finally, scrum also has five events:

Finally, scrum also has five events:

- **Sprint:** Scrum's term for iteration. The *sprint* is the container for each of the other scrum events, in which the scrum team creates potentially shippable functionality. Sprints are short cycles, no longer than a month, typically between one and two weeks, and in some cases as short as one day. Consistent sprint length reduces variance; a scrum team can confidently extrapolate what it can do in each sprint based on what it has accomplished in previous sprints. Sprints give scrum teams the opportunity to make adjustments for continuous improvement immediately, rather than at the end of the project.

- **Sprint planning:** Takes place at the start of each sprint. In sprint planning meetings, scrum teams decide which goal, scope, and supporting tasks will be part of the sprint backlog.
- **Daily scrum:** Takes place daily for no more than 15 minutes. During the daily scrum, development team members make three statements:
  - What the team member completed yesterday
  - What the team member will work on today
  - A list of items impeding the team member

  The scrum master also participates in the context of impediments he or she is working to remove for the developers.
- **Sprint review:** Takes place at the end of each sprint. In this meeting, the development team demonstrates to the stakeholders and the entire organization the accepted parts of the product the team completed during the sprint. The key to the sprint review is collecting feedback from the stakeholders, which informs the product owner how to update the product backlog and consider the next sprint goal.
- **Sprint retrospective:** Takes place at the end of each sprint. The sprint retrospective is an internal team meeting in which the scrum team members (product owner, development team, and scrum master) discuss what went well during the sprint, what didn't work well, and how they can make improvements for the next sprint. This meeting is action-oriented (frustrations should be vented elsewhere) and ends with tangible improvement plans for the next sprint.

Scrum is ci...