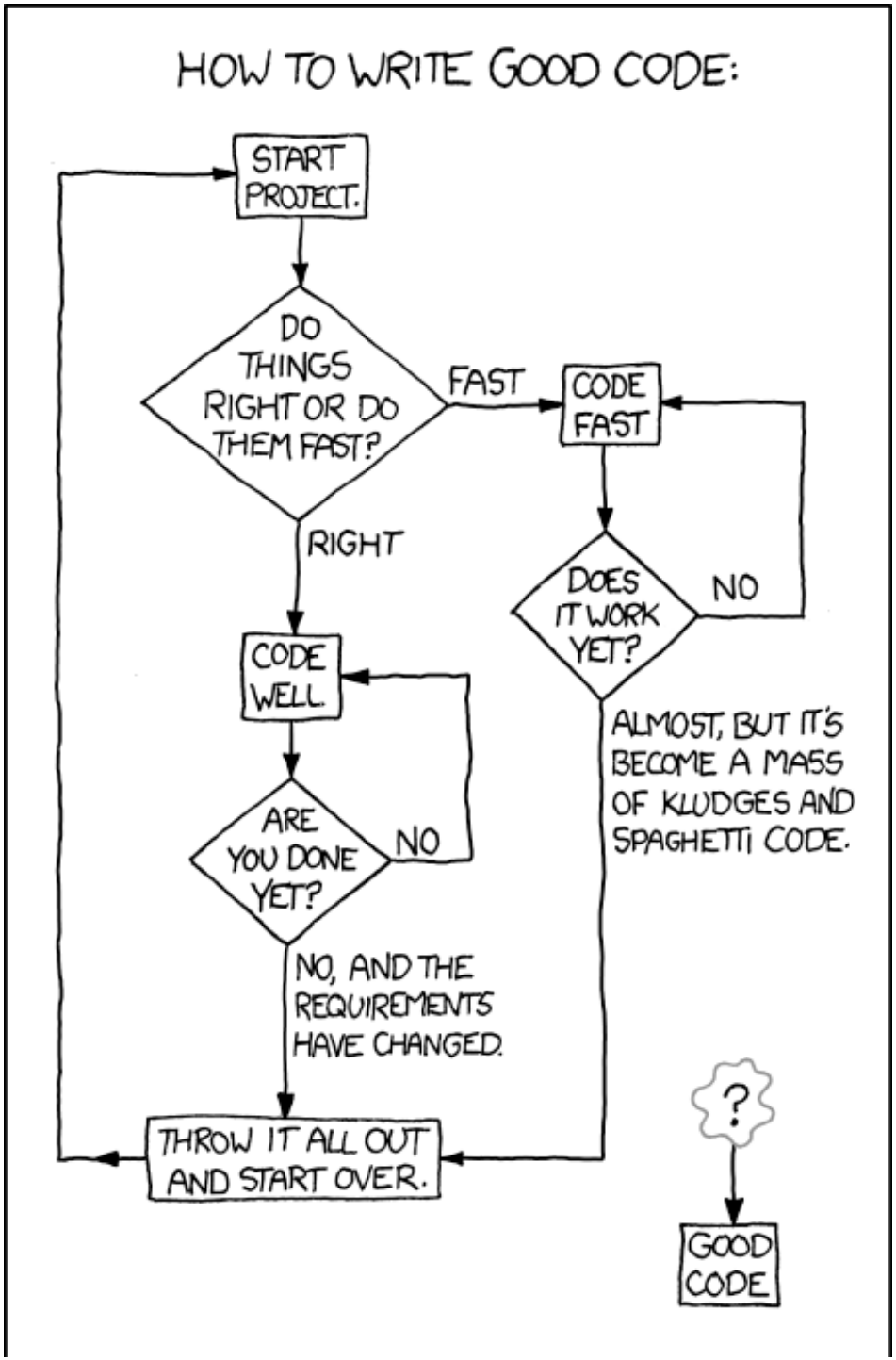


Source: xkcd 844

CPEN 321

W2 L1:
Software Lifecycle, Process

Agenda

- Some Logistics
- Software Lifecycle
- Software Development Processes

Canvas

- Is there still a problem? Reply from CIS:

  I've looked at the CPEN 321 course and I don't think the problem is on your end. Many users were experiencing access issues to Canvas yesterday. Could you ask your students if they were able to log in to Canvas, and simply didn't see the CPEN 321 course on their Dashboard? Or if they couldn't log in in general.

- Lecture slides are posted on Canvas (when it is available)

Project Pitches Next Week

- The TAs will post the list of groups and presentation order on Piazza on September 15-16

- Groups present in order during lectures on September 17 and 19

Project Presentation Structure

- Prepare a 3-slide, 5-minute pitch for each group
  - Vision and novelty
  - Must have features
  - Bonus features
  - Envisioned architecture
  - Challenges and risks

Grading (5% of your final grade)

- Graded by instructors according to:
  - Motivation: clear explanation of who needs this product and why
  - Description of mandatory and bonus features
  - Envisioned architecture and an explanation of why it is in scope for the course project, i.e., has non-trivial client and server components
  - Description risks and possible mitigation strategies
  - General presentation clarity and efficient communication of ideas

- Peer-grading
  - You will be asked to grade all presentations as well (excluding yours)
  - Grades of instructors will count as one voice each
  - Average student grades will also count as one voice

Feedback

- Instructors will give feedback on project scope, etc., which is to be taken into account for the next customer-development team meeting.

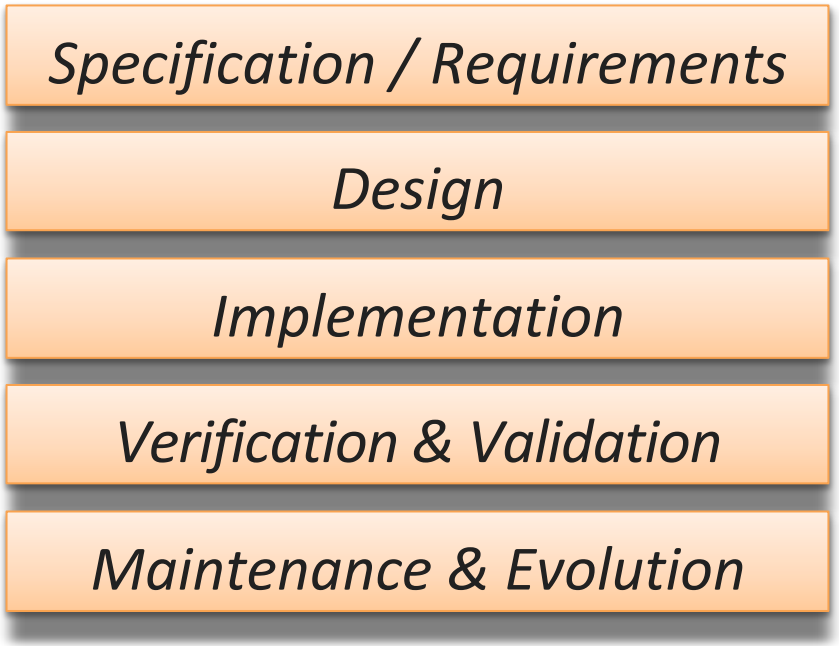
Announcement: The Myth of Multitasking

- Sort term: does not work
  - University of London: participants who multitasked during cognitive tasks experienced IQ score declines – similar to what they'd expect if they had smoked marijuana or stayed up all night.
- Long-term: damages your brain and ability to perform
  - Stanford: people who are regularly bombarded with several streams of electronic information cannot pay attention, recall information, or switch from one job to another as well as those who complete one task at a time.
  - University of Sussex: high multitaskers had less brain density in the anterior cingulate cortex, a region responsible for empathy as well as cognitive and emotional control.

http://www.talentsmart.com/articles/Multitasking-Damages-Your-Brain-and-Your-Career%2C-New-Studies-Suggest-2102500909-p-1.html

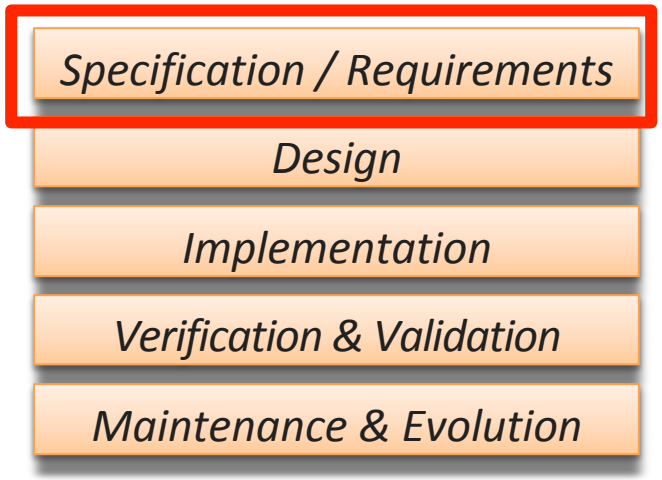
Software Lifecycle

Five Essential Tasks in Software Engineering



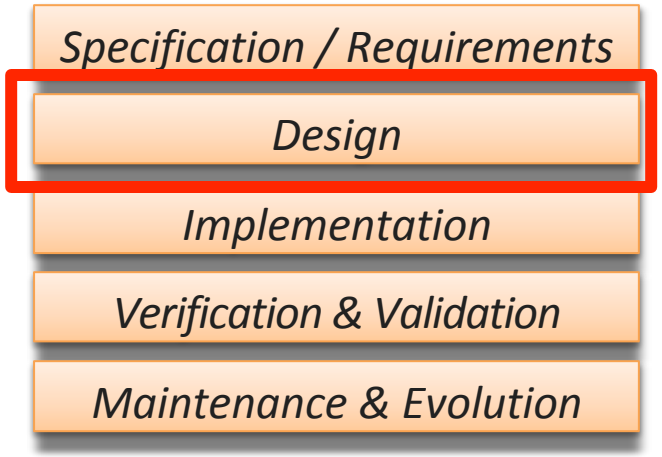
Specification / Requirements

- A **software specification** is a description of a software system to be developed
  - Lays out *functional* and *non-functional* requirements
  - May include a set of *use cases* that describe user interactions that the software must provide
- **Requirements engineering** is the process of defining, documenting and maintaining requirements
  - Requirements elicitation
  - Requirements analysis and negotiation
  - Requirements validation
  - Requirements management



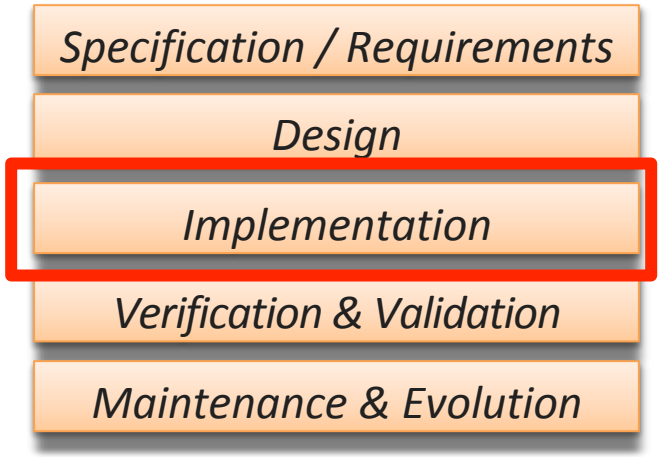
Design

- Guides the development team in building a software product
- Different aspects of design:
  - Architectural design
  - Interface design
  - Data structure design
  - Algorithm design
  - etc.
- A competent design relies on:
  - creative skill,
  - past experience,
  - known best practices
  - etc.



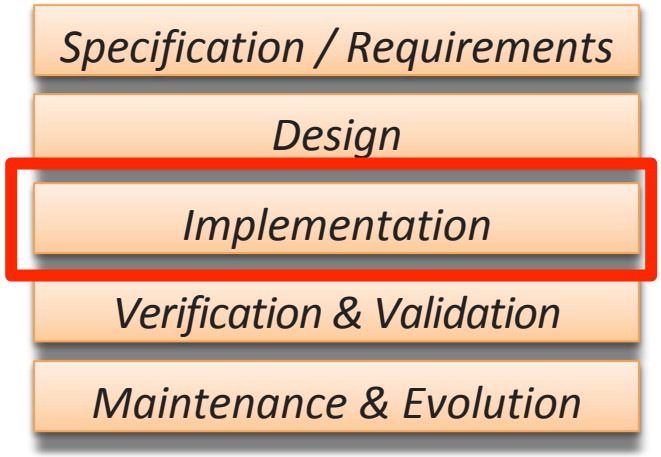
Implementation

- The process of converting the software design into an executable system
- An art, a craft, or an engineering discipline?



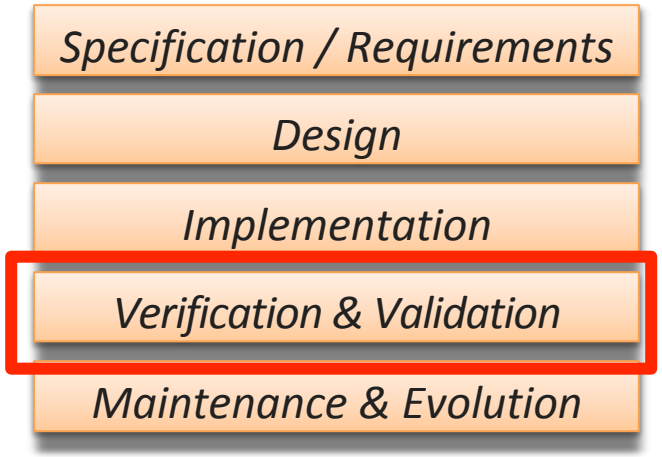
Implementation

- The process of converting the software design into an executable system
- An art, a craft, or an engineering discipline?
  - Good design helps
  - Patterns and anti-patterns
- End-product must address some fundamental principles:
  - Efficiency/performance
  - Robustness
  - Usability
  - Maintainability
  - Etc.

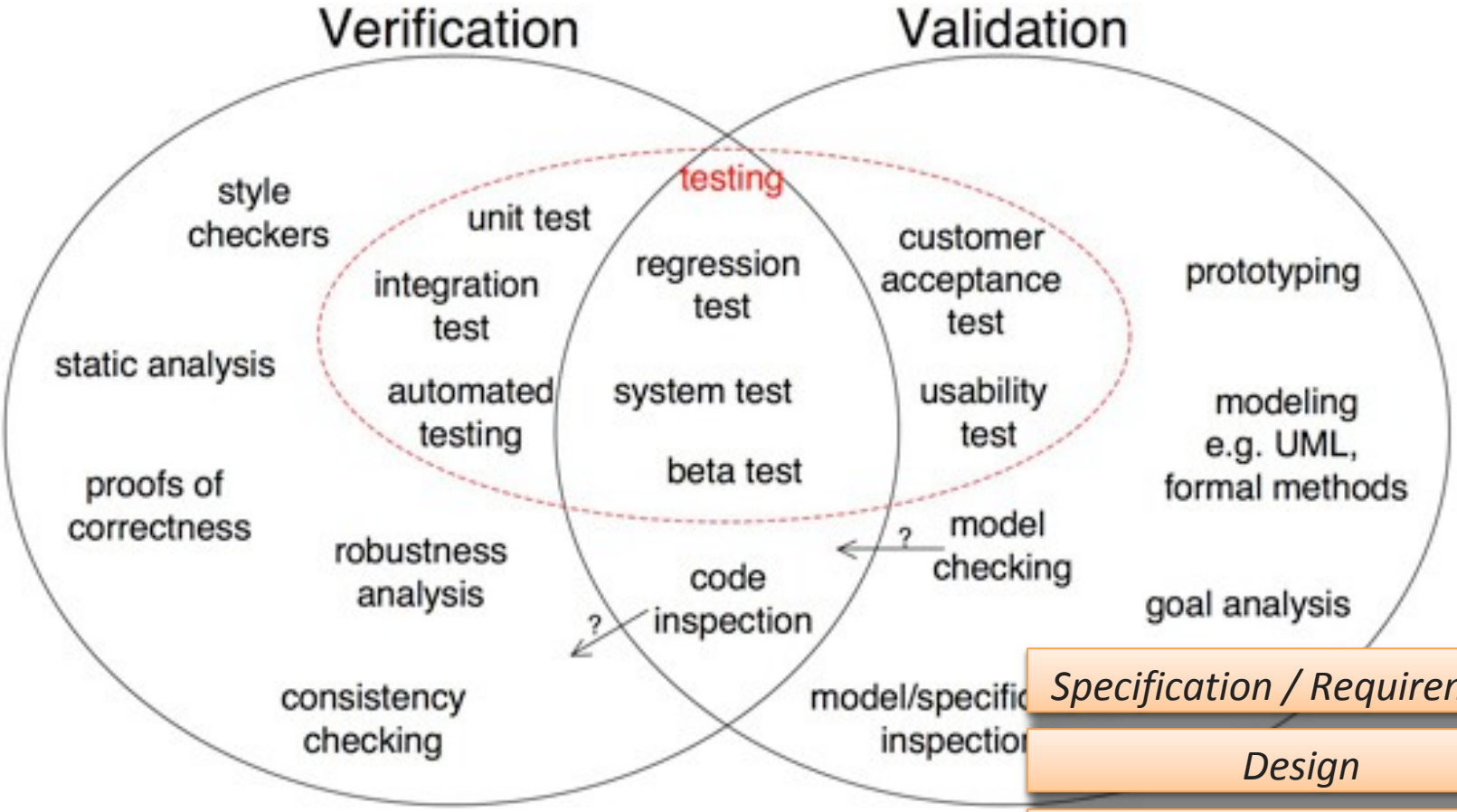


Verification & Validation

- Validation and Verification (V & V) shows that a system conforms to its specification and meets the requirements of the customer
- **Verification**: are we building the product right?
  - Does the software meet the specification?
- **Validation**: are we building the right product?
  - Does the specification capture the customer's needs?



Verification & Validation



Specification / Requirements

Design

Implementation

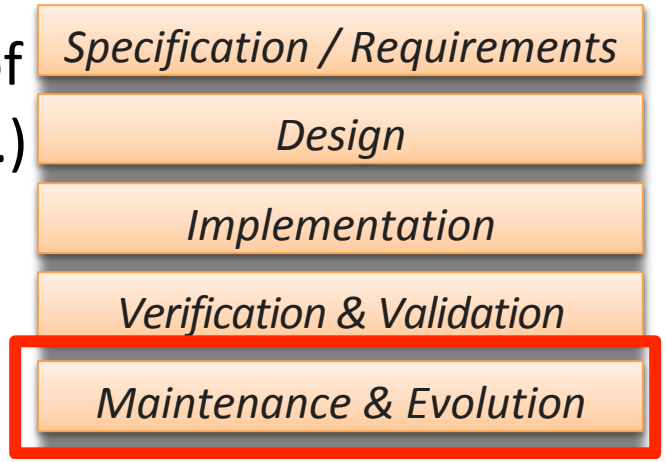
Verification & Validation

Maintenance & Evolution

Maintenance & Evolution

- The modification of a software product after delivery

- Four main types of tasks:
  - Corrective – fixing errors
  - Perfective – implementing new or changed user requirements
  - Adaptive – modifying the system to cope with changes in the environment (e.g., a new OS version)
  - Preventive – increasing software maintainability or reliability (deletion of obsolete capabilities, optimization, etc.)



Maintenance & Evolution

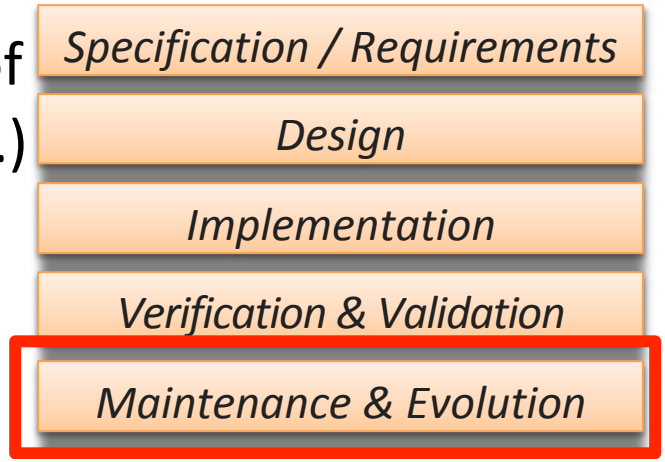
- The modification of a software product after delivery

- Four main types of tasks:

21% – Corrective – fixing errors

75%
- – Perfective – implementing new or changed user requirements
- – Adaptive – modifying the system to cope with changes in the environment (e.g., a new OS version)

– Preventive – increasing software maintainability or reliability (deletion of obsolete capabilities, optimization, etc.)



Maintenance & Evolution

- The modification of a software product after delivery

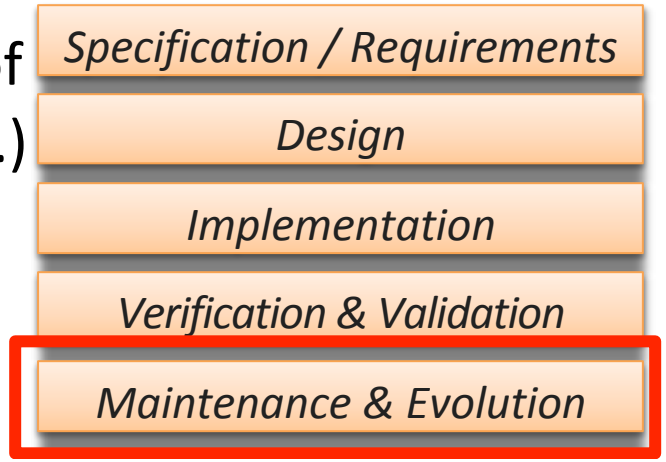
- Four main types of tasks:

21% – Corrective – fixing errors

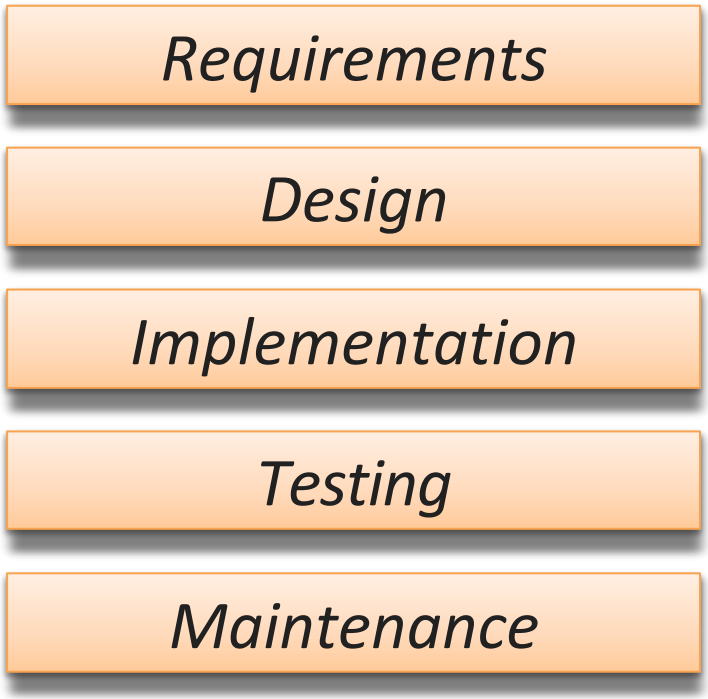
75%
- – Perfective – implementing new or changed user requirements
- – Adaptive – modifying the system to cope with changes in the environment (e.g., a new OS version)

– Preventive – increasing software maintainability or reliability (deletion of obsolete capabilities, optimization, etc.)

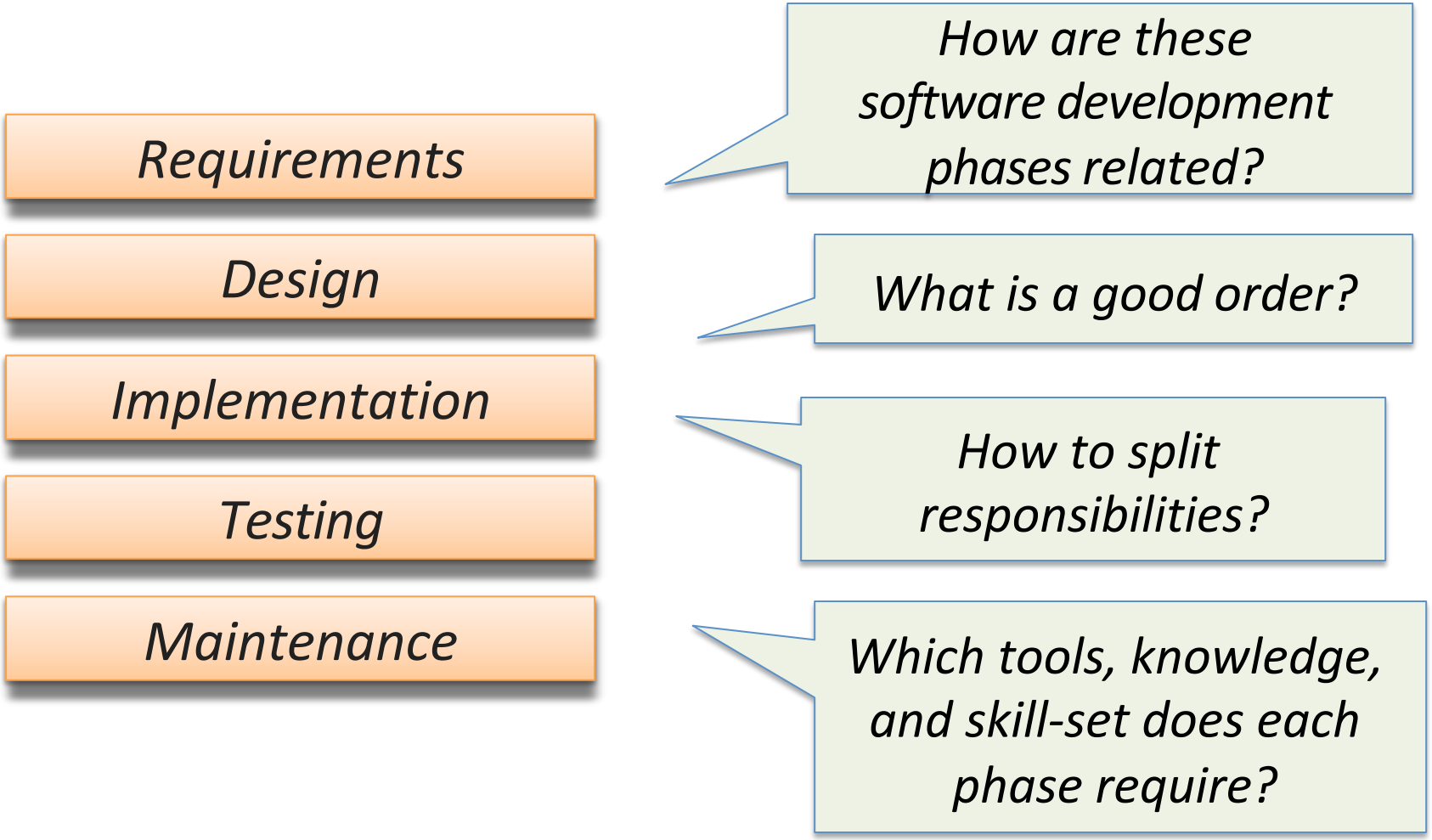
Very few systems are built from scratch!



Essential Tasks in Software Engineering



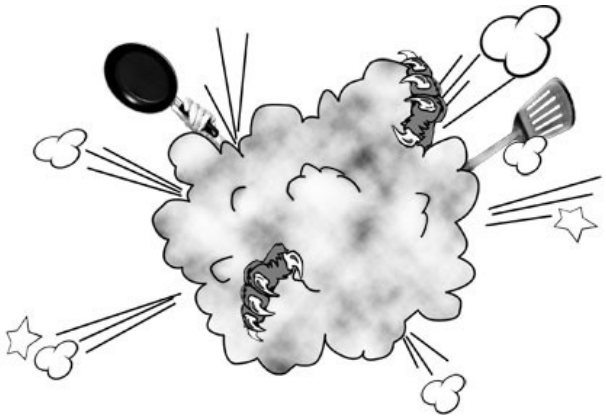
Essential Tasks in Software Engineering



A Software Process

A process defines ***who*** is doing ***what***, ***when***, and ***how*** in the development of a software system

Life Without Software Process



- Advantages:
  - nothing to learn or plan!
  - work on whatever is interesting, ignore the rest

- Disadvantages:
  - may ignore some important tasks (testing, design)
  - not clear when to start or stop doing each task
  - scales poorly to multiple people
  - hard to review or evaluate one's work

Life With Software Process

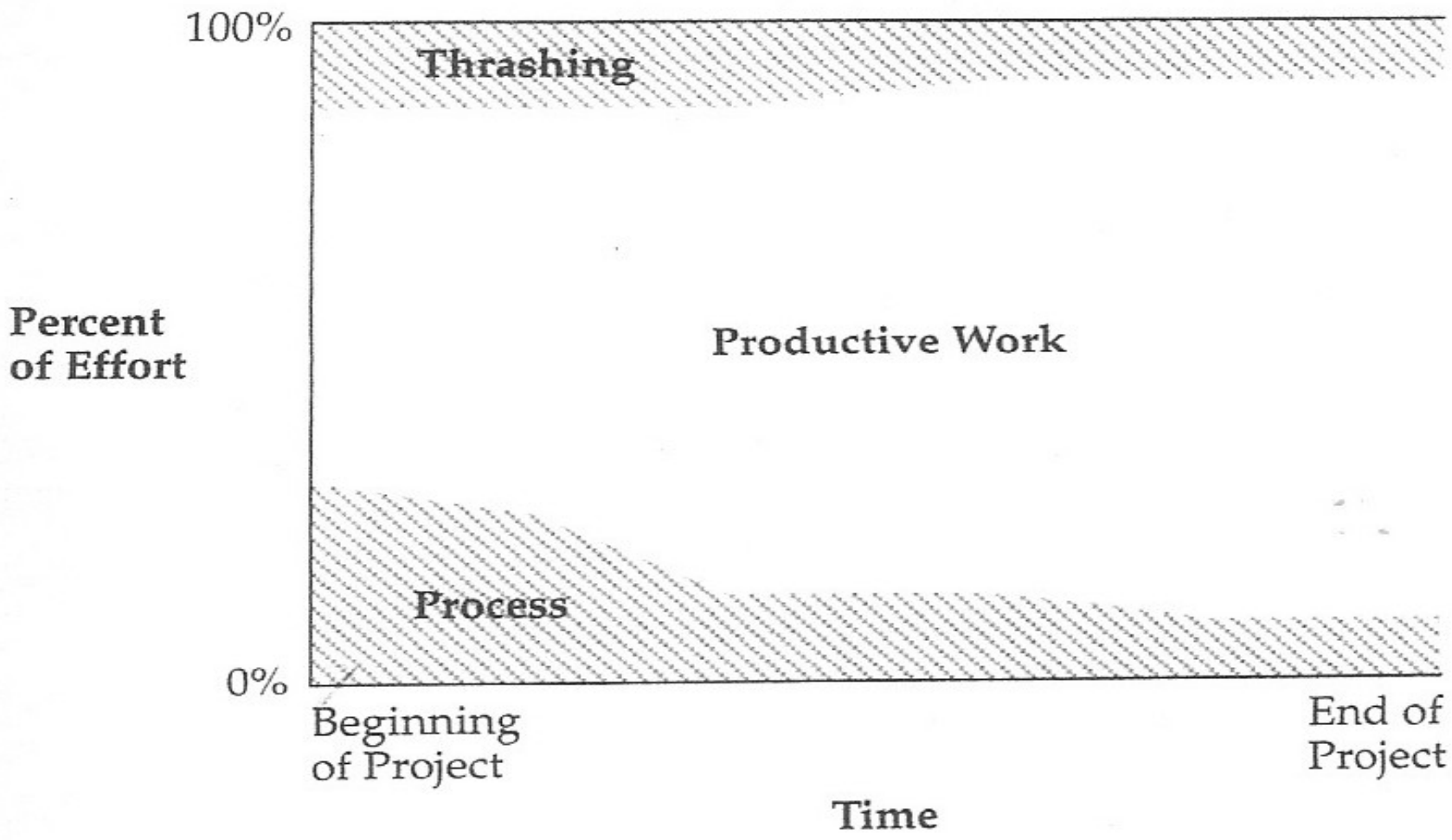
- Advantages:
  - Structures the work.
  - Serves as a management tool.

- Disadvantages:
  - Can lead to compromises and artificial constraints.
  - Risk of over-emphasizing process rather than the product itself!

Main Message

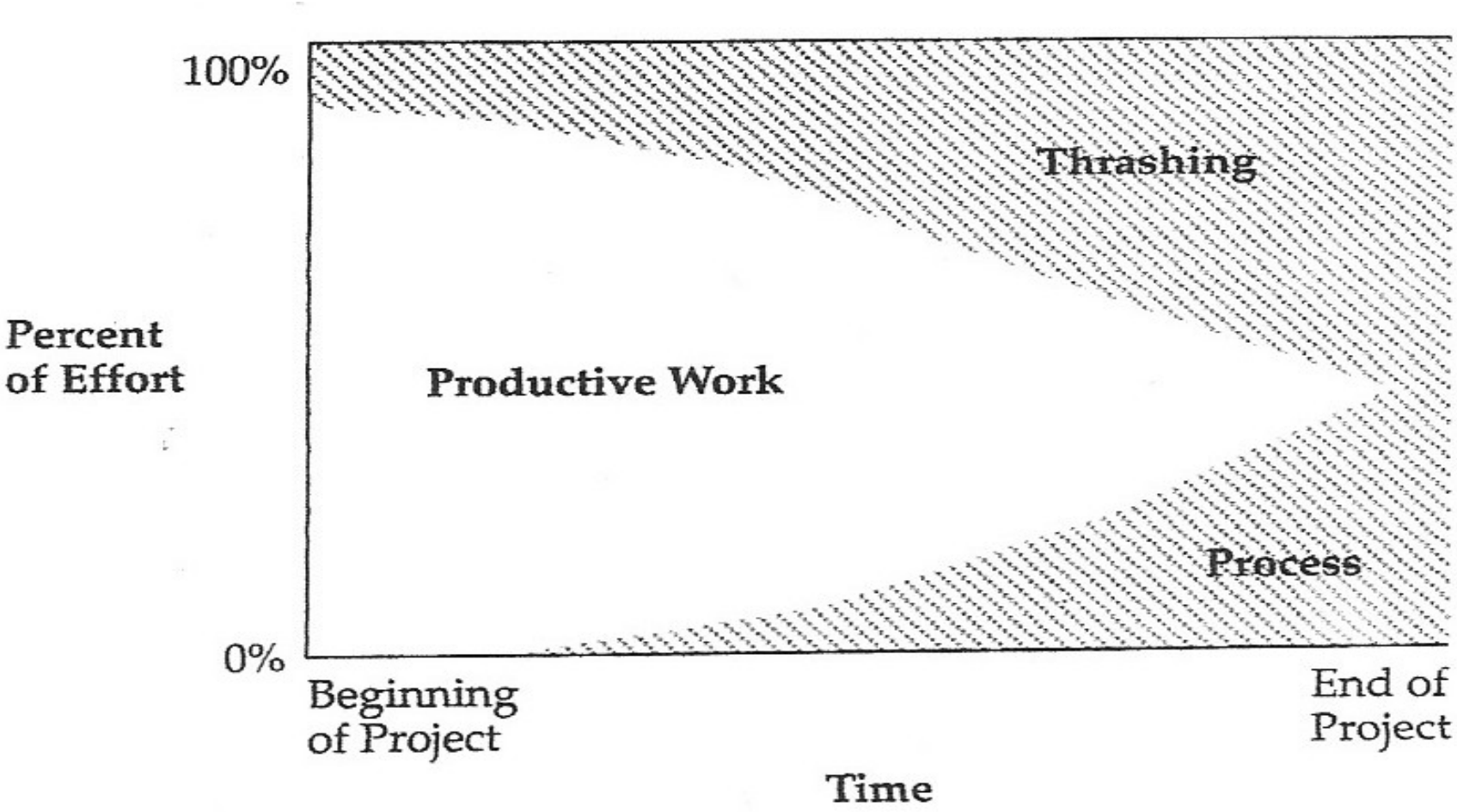
- Follow processes, but do not over-emphasize process over product
- Customize the process for your needs!
- No good or bad processes: it depends on the organizational culture, structure, needs

Adopting a Process Takes Time



Survival Guide, McConnell, p. 25

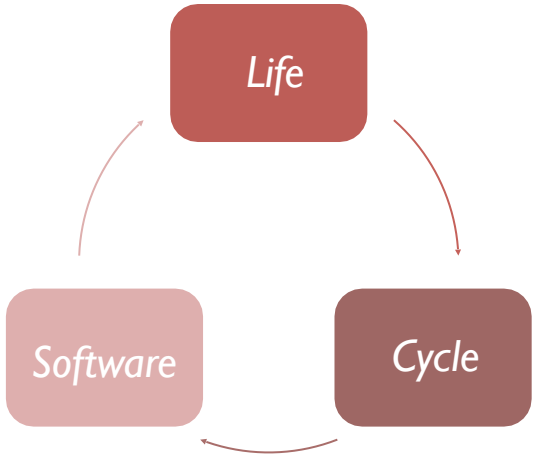
Better done early!



Survival Guide, McConnell, p. 24

Which Software Process Models are You Familiar with?

Popular Software Development Process Models

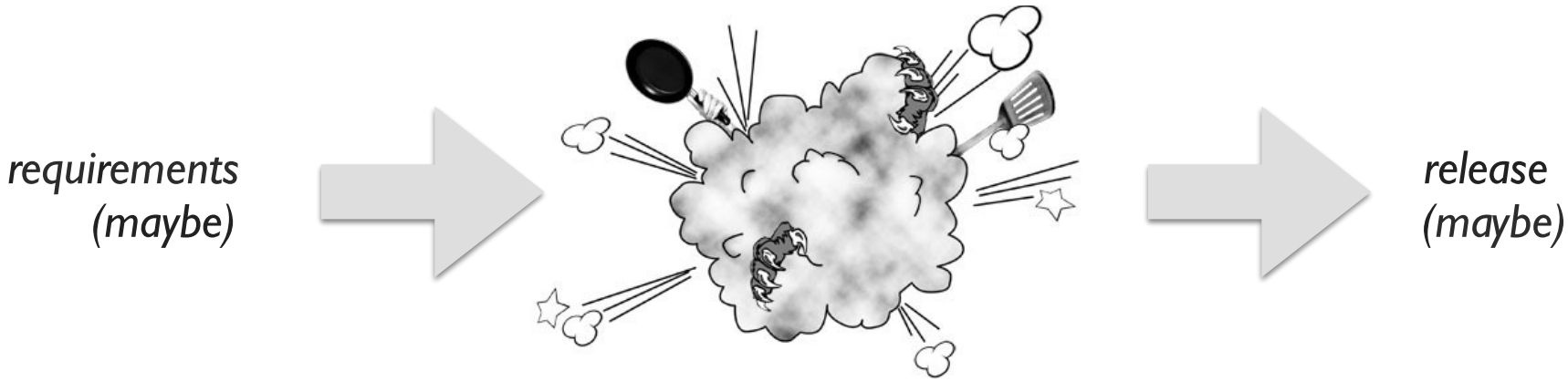


No Plans
- **Code-and-fix**: write code, fix it when it breaks

Plan and Document
- **Waterfall:** perform each phase in order (1970)

Incremental
- **Staged Delivery:** waterfall-like beginnings, then, short release cycle
- **Evolutionary prototyping:** develop a skeleton system and evolve it for delivery
- **Spiral**: triage/figure out riskiest things first (1988)
- **Agile**: *a family of principles* promoting adaptive planning, evolutionary development, early delivery, and continuous improvement (1970-2005+)
  - Most popular: **Scrum** and **Kanban**

Code-and-fix Model

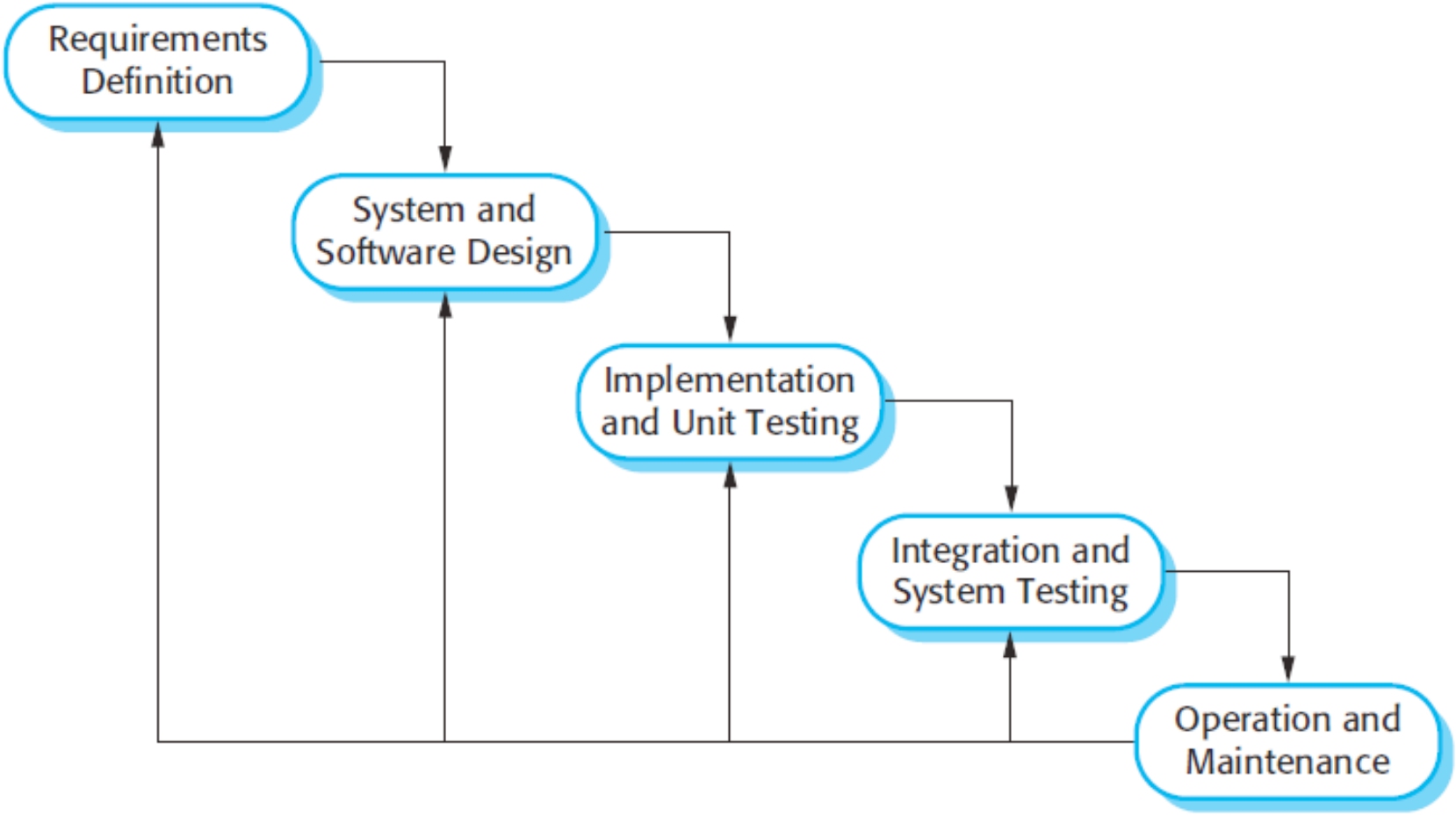


+ Applicable for very small projects and short-lived prototypes

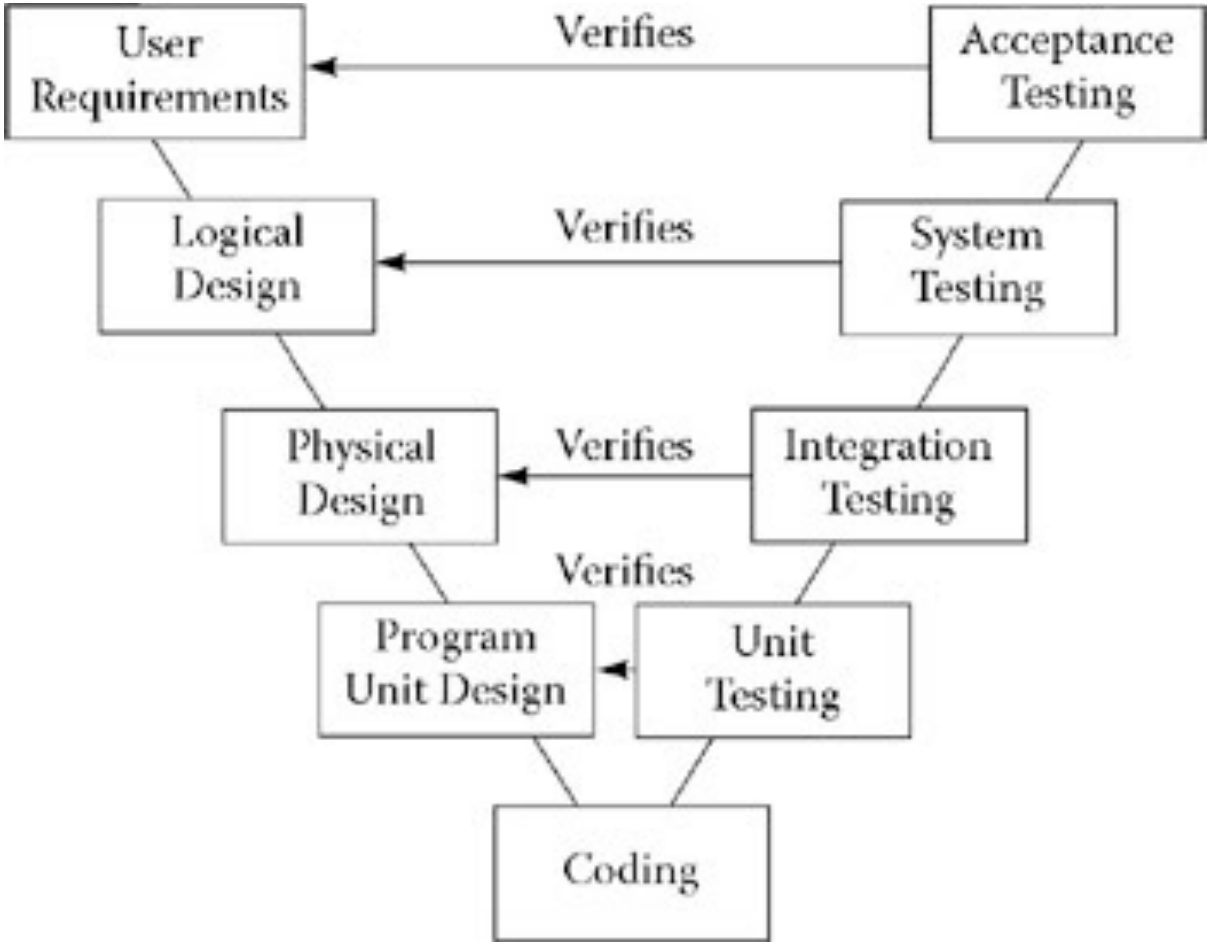
- Unlikely to accommodate changes without a major design overhaul

- No good way to assess and mitigate risks

Waterfall Model



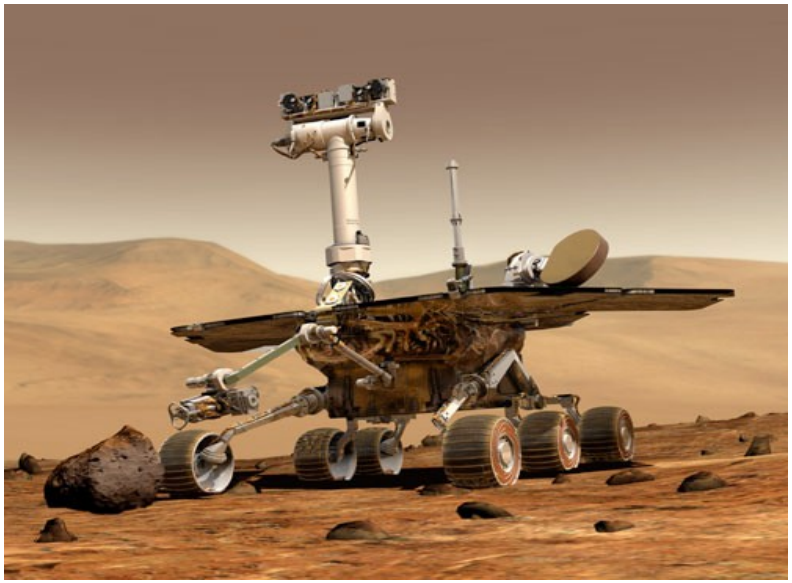
Waterfall V-Model



Is Waterfall Bad?

Waterfall Model Advantages

- Suitable for projects that are very well understood but complex
  - Tackles all planning up-front
  - The idea of no midstream changes equates to an efficient software development process



Waterfall Model Advantages

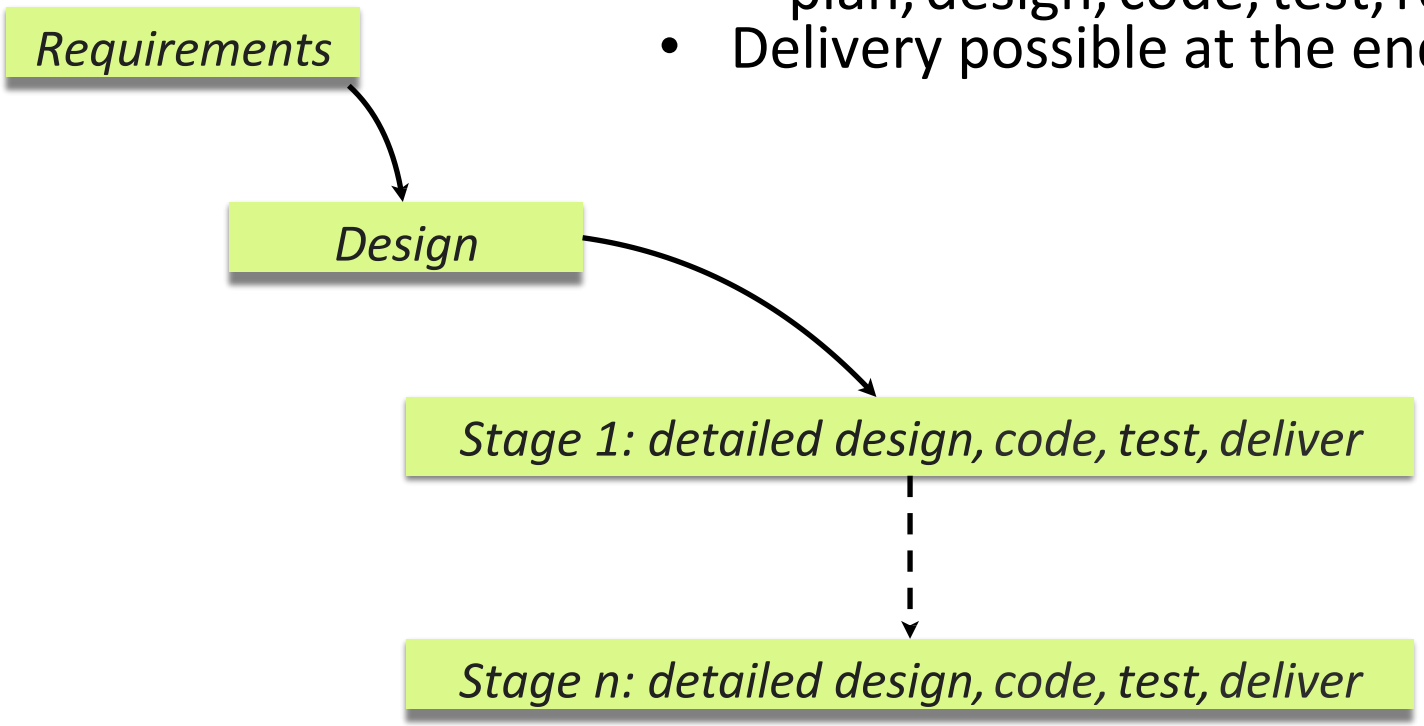
- Suitable for projects that are very well understood but complex
  - Tackles all planning up-front
  - The idea of no midstream changes equates to an efficient software development process
- Supports inexperienced teams
  - Orderly, easy-to-follow sequential model
  - Reviews at each stage determine if the product is ready to advance

Waterfall Model Disadvantages

- Requires a lot of planning upfront (not always easy)
- Integration occurs at the very end
- No sense of progress until the very end
  - no code to show until almost done
- Delivered product may not match customer needs
- Change is costly

Staged Delivery Model

- Waterfall-like beginnings
- Then, short release cycles:
  - plan, design, code, test, release
- Delivery possible at the end of any cycle

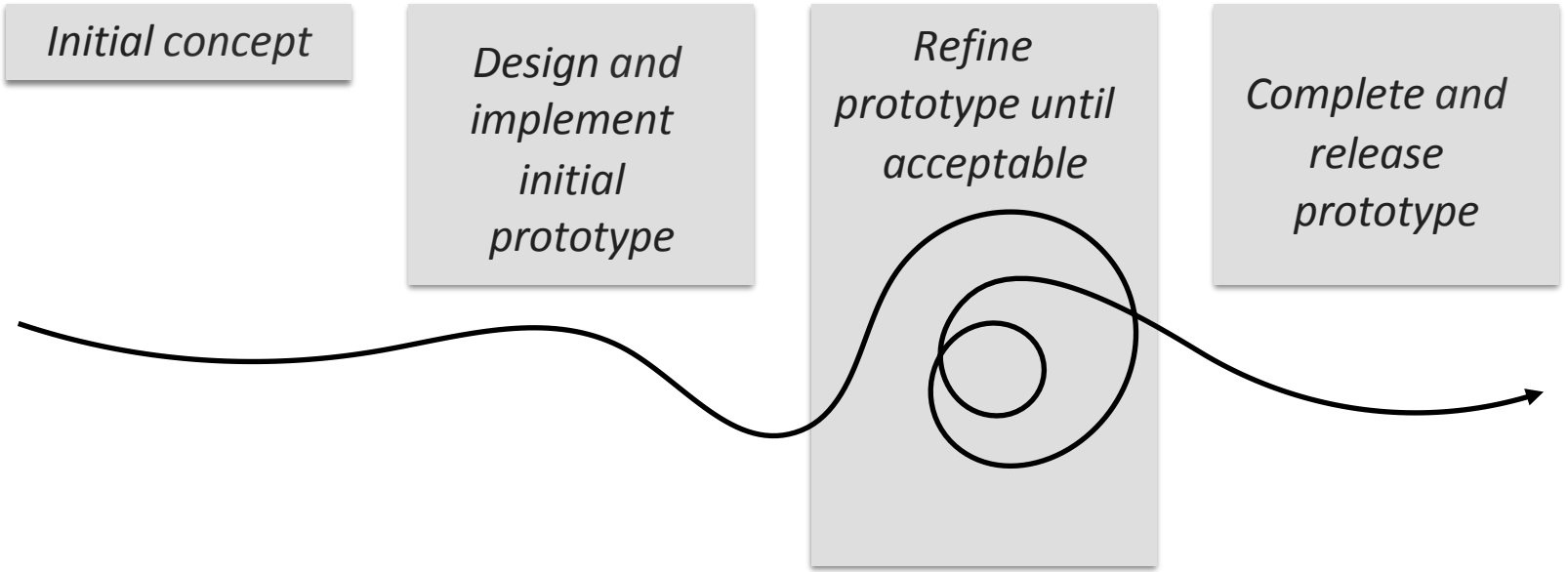


Staged Delivery Model

- Advantages:
  - Intermediate deliveries show progress, satisfy customers, and lead to feedback
  - Can ship at the end of any release cycle
    - Looks like success to customers, even if not original goal
  - Integration problems are visible early

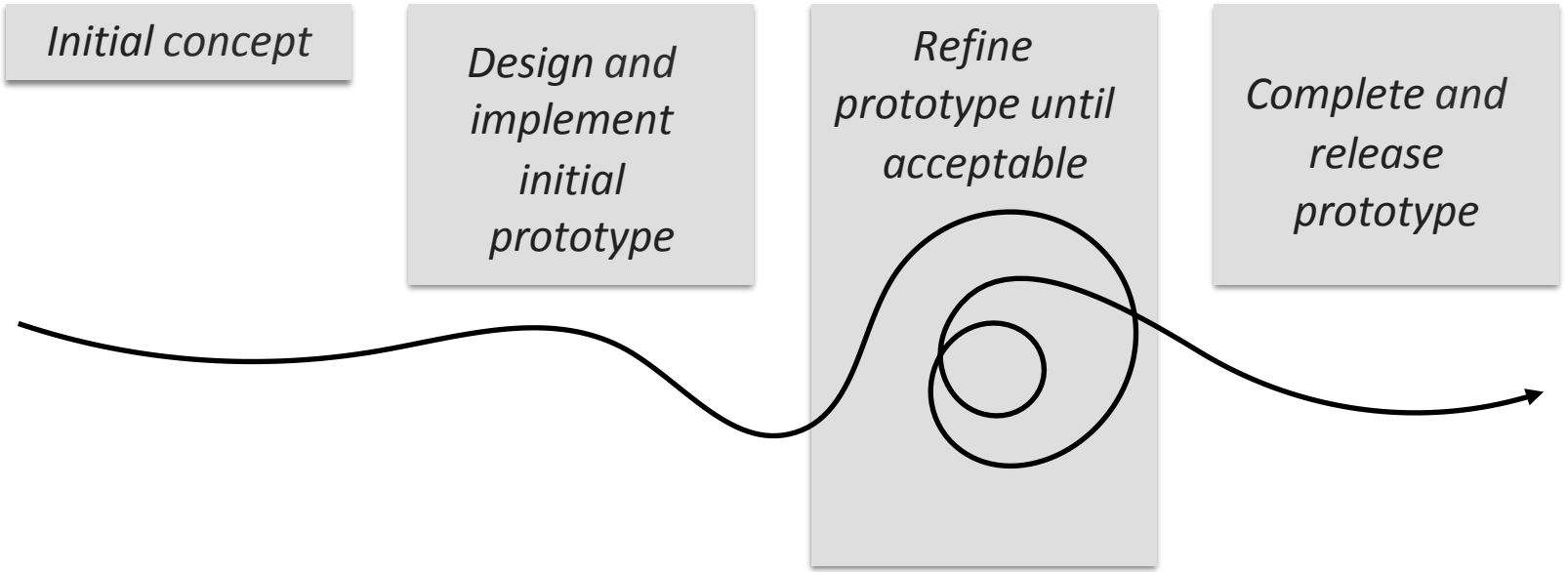
- Disadvantages:
  - Assumes requirements are known up-front

Evolutionary Prototyping



Develop a skeleton system and evolve it for delivery

Evolutionary Prototyping



Develop a skeleton system and evolve it for delivery

Different from staged delivery: requirements are not known ahead of time but rather discovered by feedback.

Evolutionary Prototyping Advantages

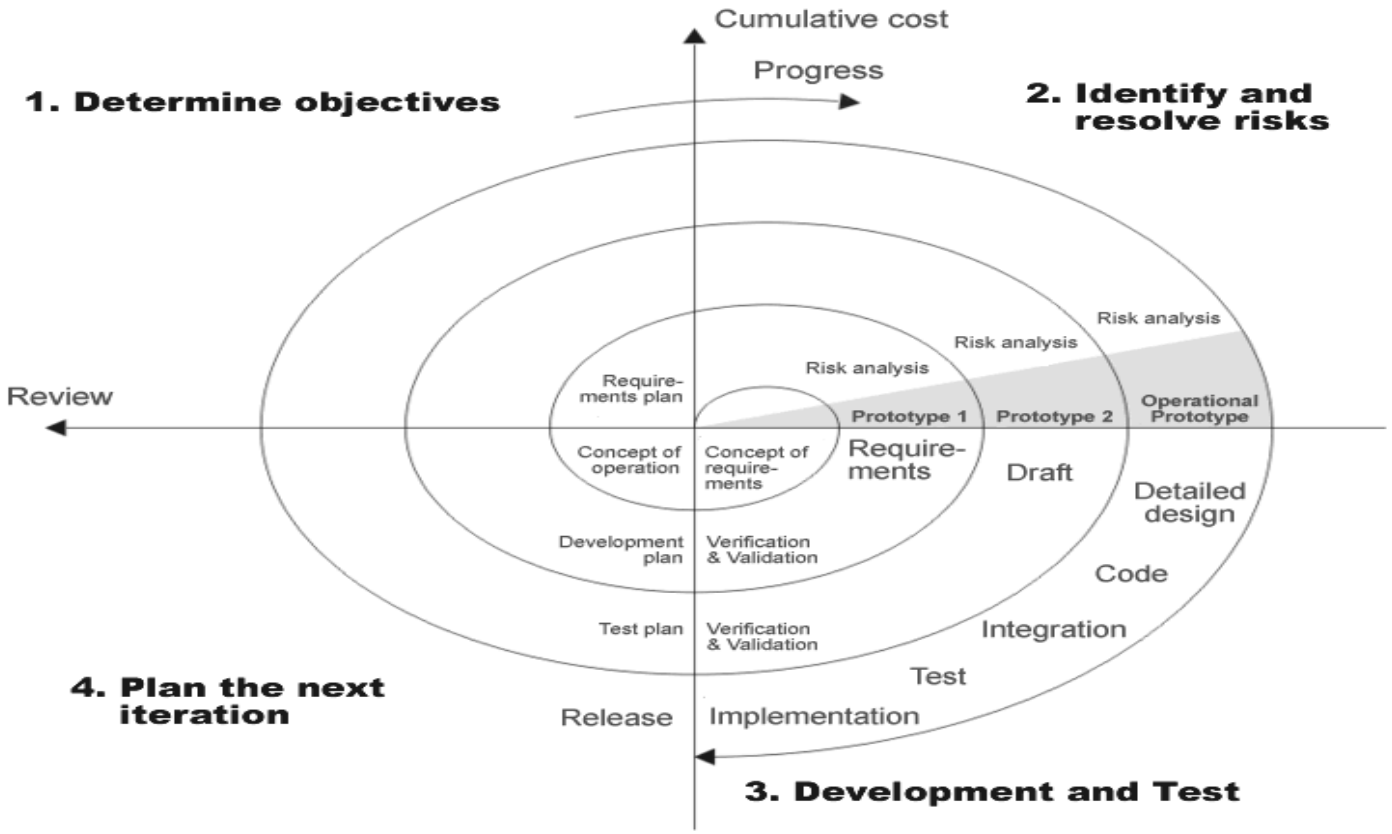
- Advantages:
  - Participatory design / useful feedback loops

Very practical and widely used

- Disadvantages:
  - Assumes the spec is flexible
  - Requires close customer involvement
  - Problems with planning (especially if the developers are inexperienced): hard to estimate completion schedule or feature set

Spiral Model (Risk-Oriented)

- A variation: in each iteration, identify and solve the sub-problems with the highest risk



Spiral Model

- Advantages:
  - Provides early indication of unforeseen problems
  - Decreases risk
  - Especially appropriate at the beginning of the project, when the requirements are still open

- Disadvantages:
  - Developers must be able to assess risk
  - Frequent changes of task

Agile Manifesto (2001)

- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan
- **Working software** over comprehensive documentation
- Value **individuals and interactions** over processes and tools
- …

Agile Manifesto Principles

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Agile Development Practices

- Acceptance test-driven development (ATDD)
- **Backlogs (Product and Sprint)**
- Behavior-driven development (BDD)
- Business analyst designer method (BADM)
- **Continuous integration (CI)**
- Cross-functional team
- Domain-driven design (DDD)
- **Information radiators (scrum board, task board, visual management board, burndown chart)**
- **Iterative and incremental development (IID)**
- Pair programming
- Planning poker
- Refactoring
- **Retrospective**
- **Scrum events (sprint planning, daily scrum, sprint review and retrospective)**
- Story-driven modeling
- Test-driven development (TDD)
- **Timeboxing**
- User story
- User story mapping
- …

Not all are implemented by each agile method!

Not all are implemented by each team!

Agile Then and Now

- *Controversial in 2001*
  - "… yet another attempt to undermine the discipline of software engineering… nothing more than an attempt to legitimize hacker behavior."

  "Manifesto Elicits Cynicism," IEEE Computer, 2001
- *Accepted now*
  - Believed by many to produce better quality software
  - Increasing adoption by industry
  - Today it is "the thing to do"
- *Future?*

SCRUM

Derived from the rugby term “scrum”
(Despite appearances, is a organized test of strength and skill)

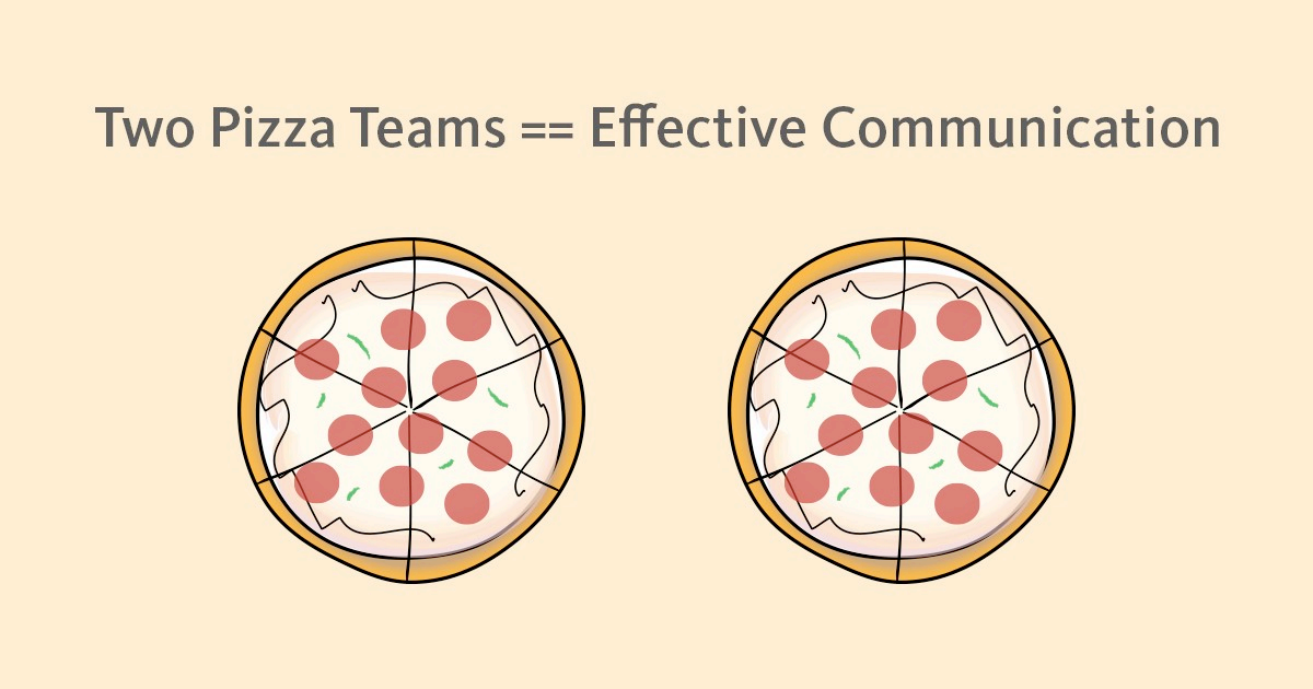
Scrum Roles

- Team members: the implementers of the product
- Product Owner: responsible for initial planning, prioritizing, represents the voice of the customer
- Scrum Master: behaves somewhat like a team lead or project manager, in working to resolve issues blocking team progress
- Users: customers or consumers of the product
- Managers: keep the team's organisation running smoothly

Scrum member rotate through roles (especially Product Owner) each iteration

Scrum Teams

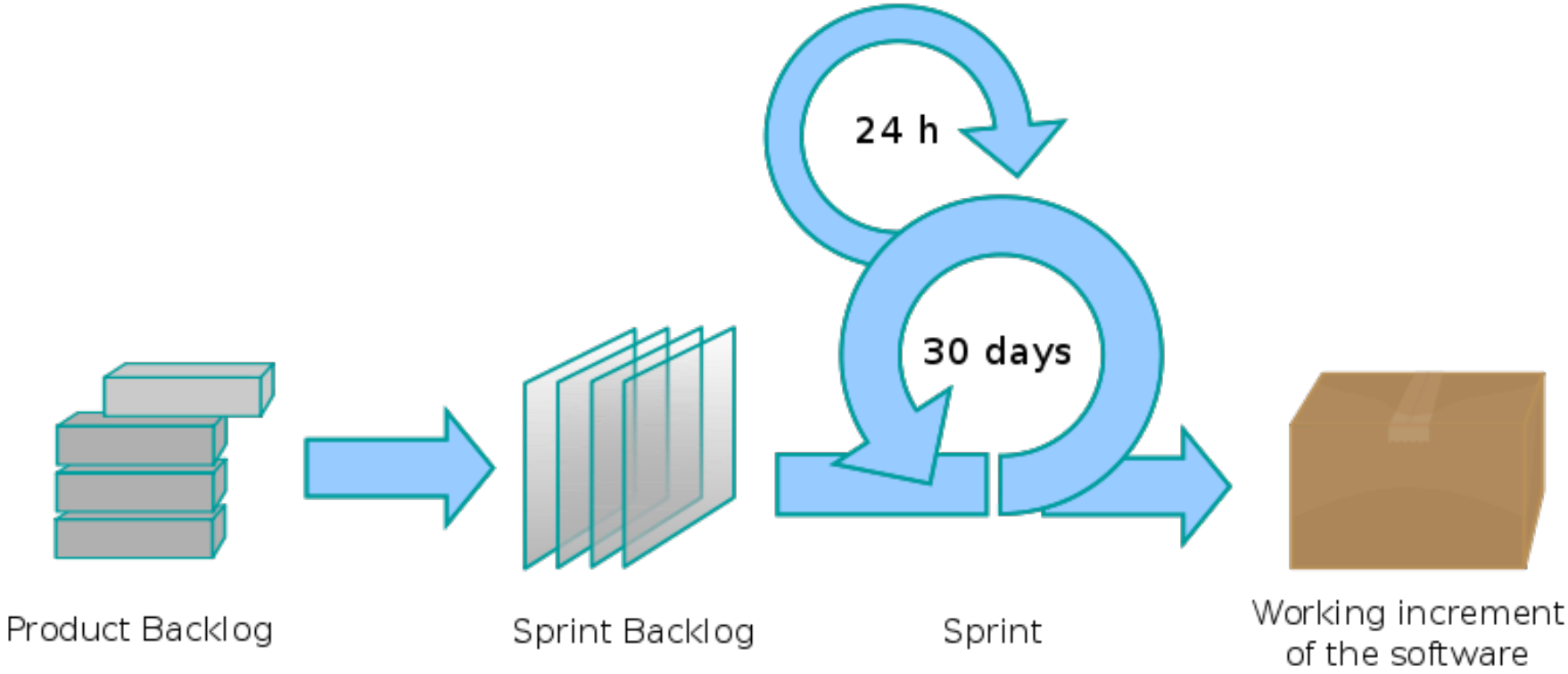
- "2 Pizza" team size (4 to 9 people)
  - "Scrum" inspired by frequent short meetings
  - 15 minutes every day at the same place and time
- Self-organizing



Sprint

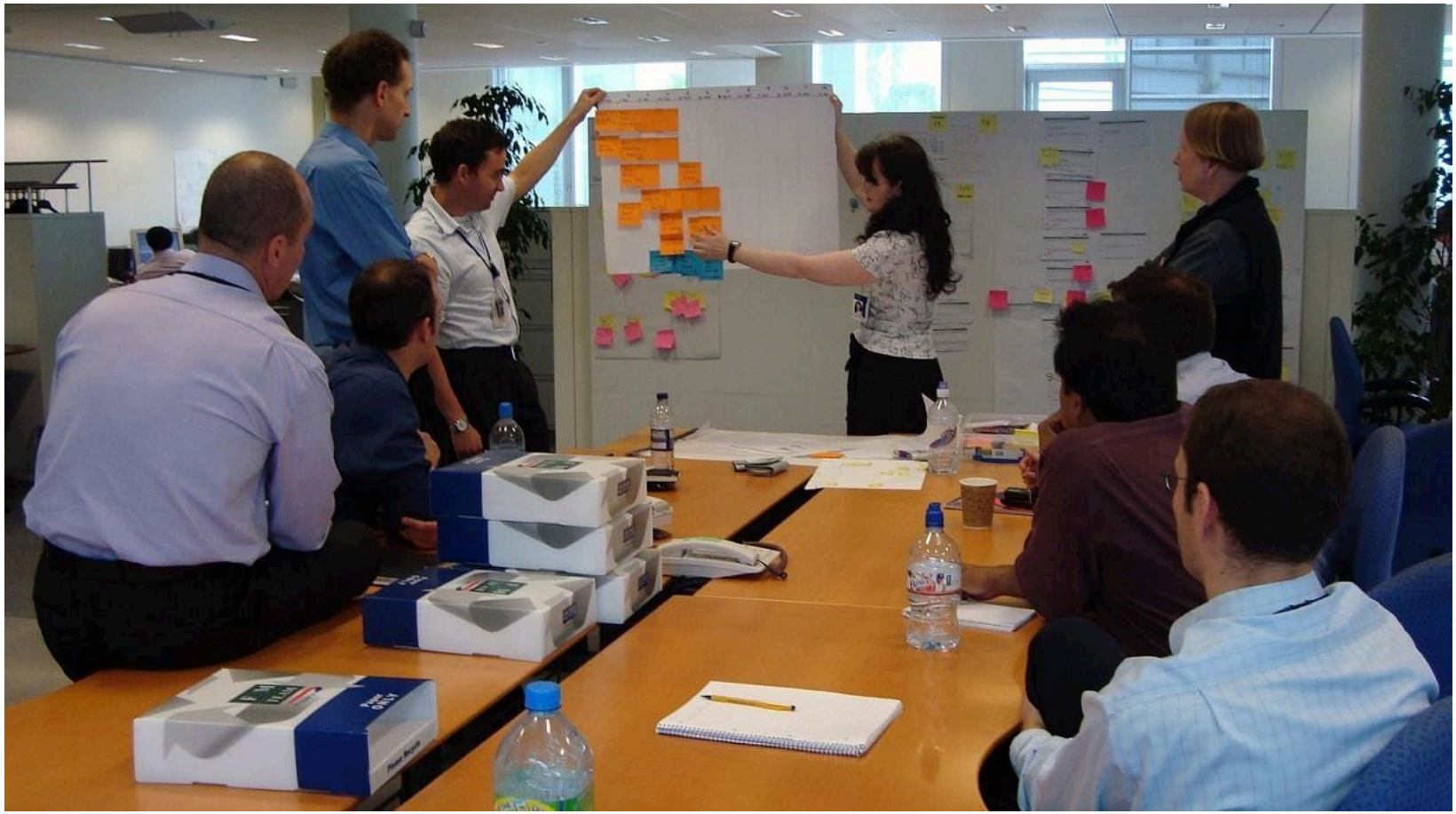
- A sprint (or iteration) is the basic unit of development in Scrum.
- The sprint is a timeboxed effort; that is, it is restricted to a specific duration.
  - normally between one week and one month
  - two weeks is the most common case

Scrum Process



Source: Wikipedia

Sprint Planning



Sprint planning: Communicate the scope of work for the sprint

- Select product backlog items that can be completed in one sprint
- The recommended duration of the meeting is 4 hours for a two-week sprint
  - First 2 hours: the scrum team selects the product backlog items they believe could be completed in that sprint
  - Second 2 hours: the development team identifies the detailed work (tasks) required to complete those product backlog items
- Result: a confirmed sprint backlog

The Daily Scrum Meetings

- Every day
- 15-30 minutes
- Whole world is invited
  - Helps avoid other unnecessary meetings
- Only team members can talk
- Stand-up

The Daily Scrum Meetings

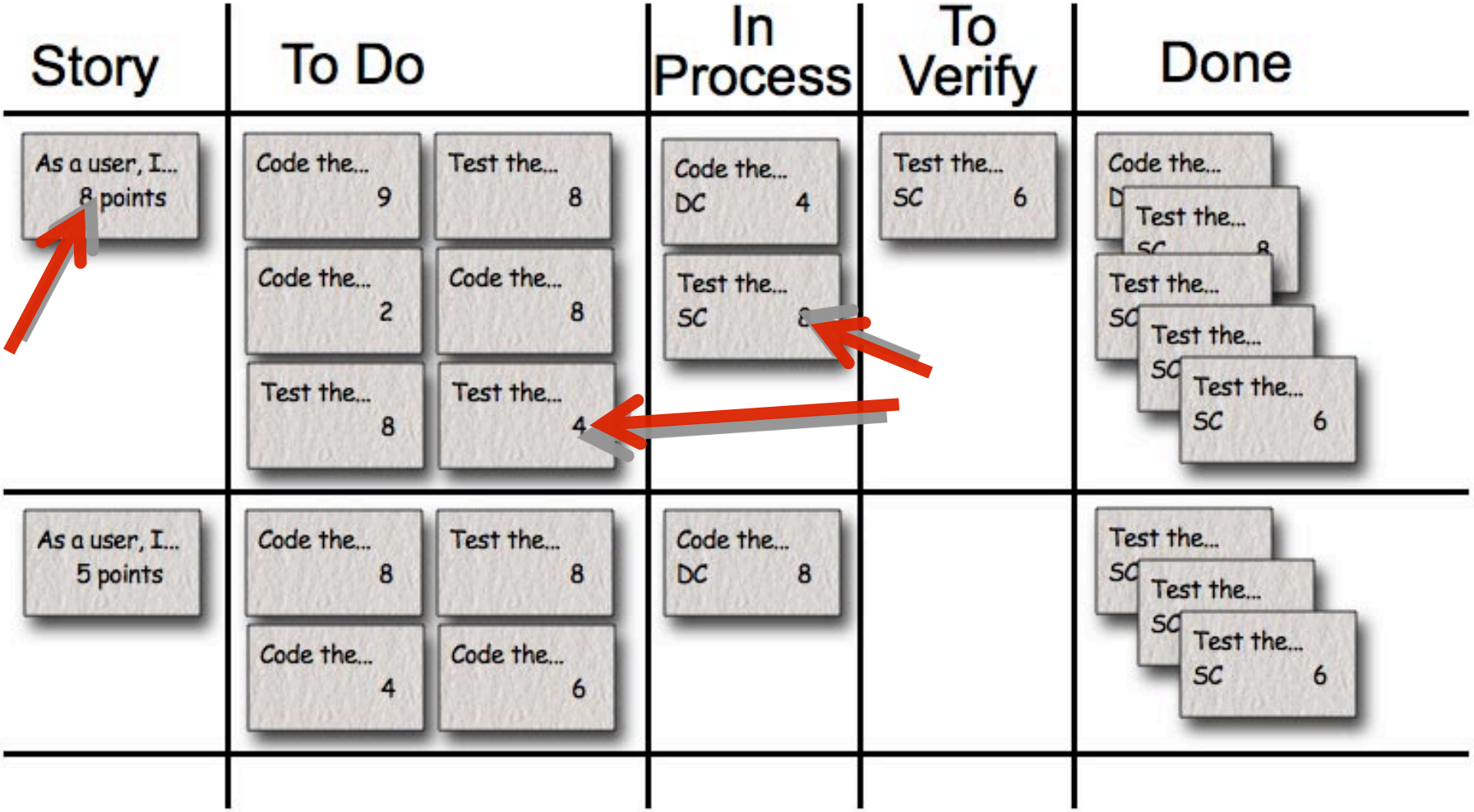
- Everyone answers 3 questions:
  1. What have you completed (relative to the Backlog) since the last Scrum meeting?
  2. What was (or is) in your way to completing this work?
  3. What will you do between now and the next Scrum meeting?

- The goal is not a status report for the ScrumMaster
  - → commitments in front of peers
  - → the Scrum Master attempts to remove barriers
  - → Not for solving technical problems

Keeping Track of Progress

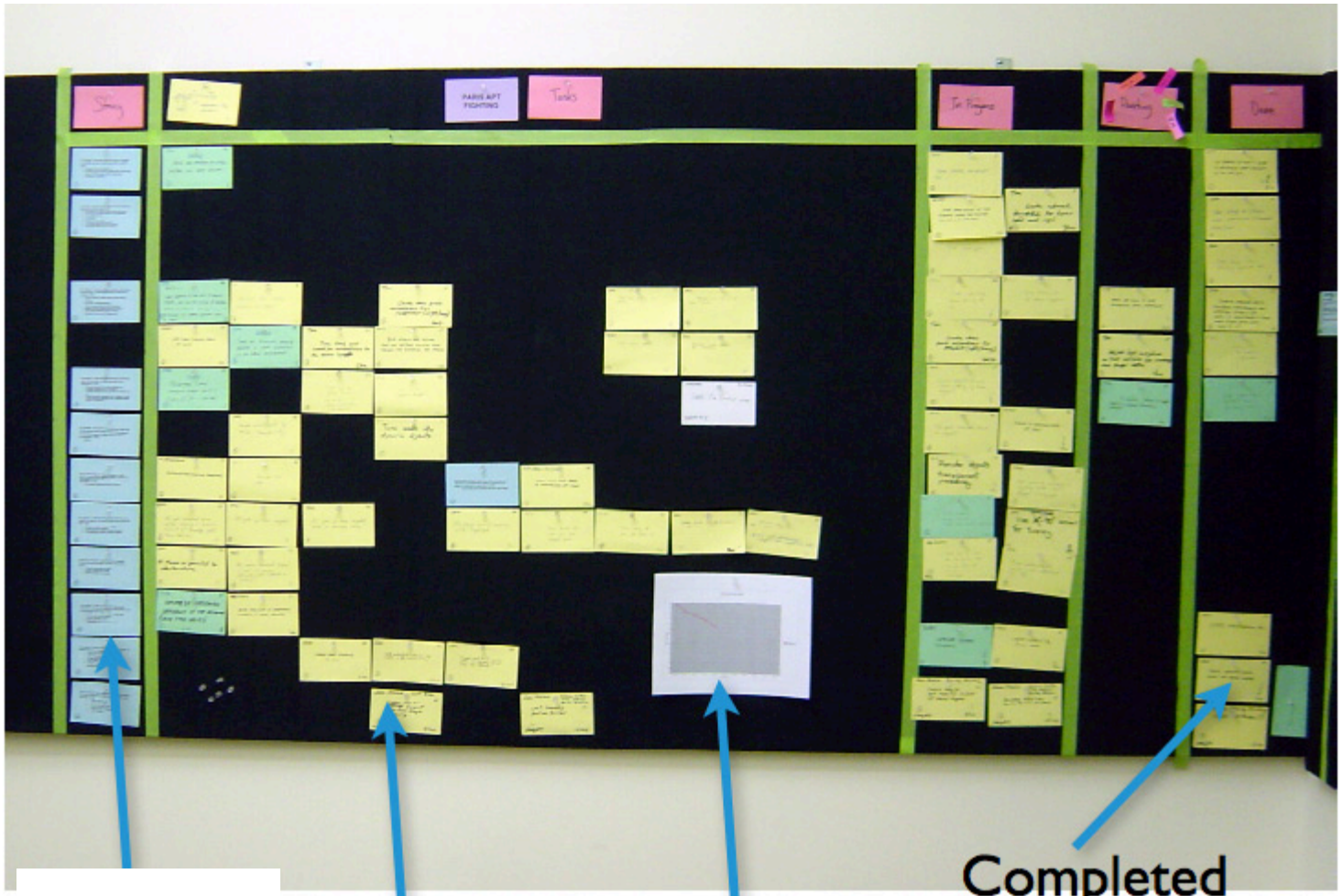
- The Task Board
- The Burndown Chart
- Sprint review and retrospective

The Task Board



Source: www.mountaingoatsoftware.com

The Task Board

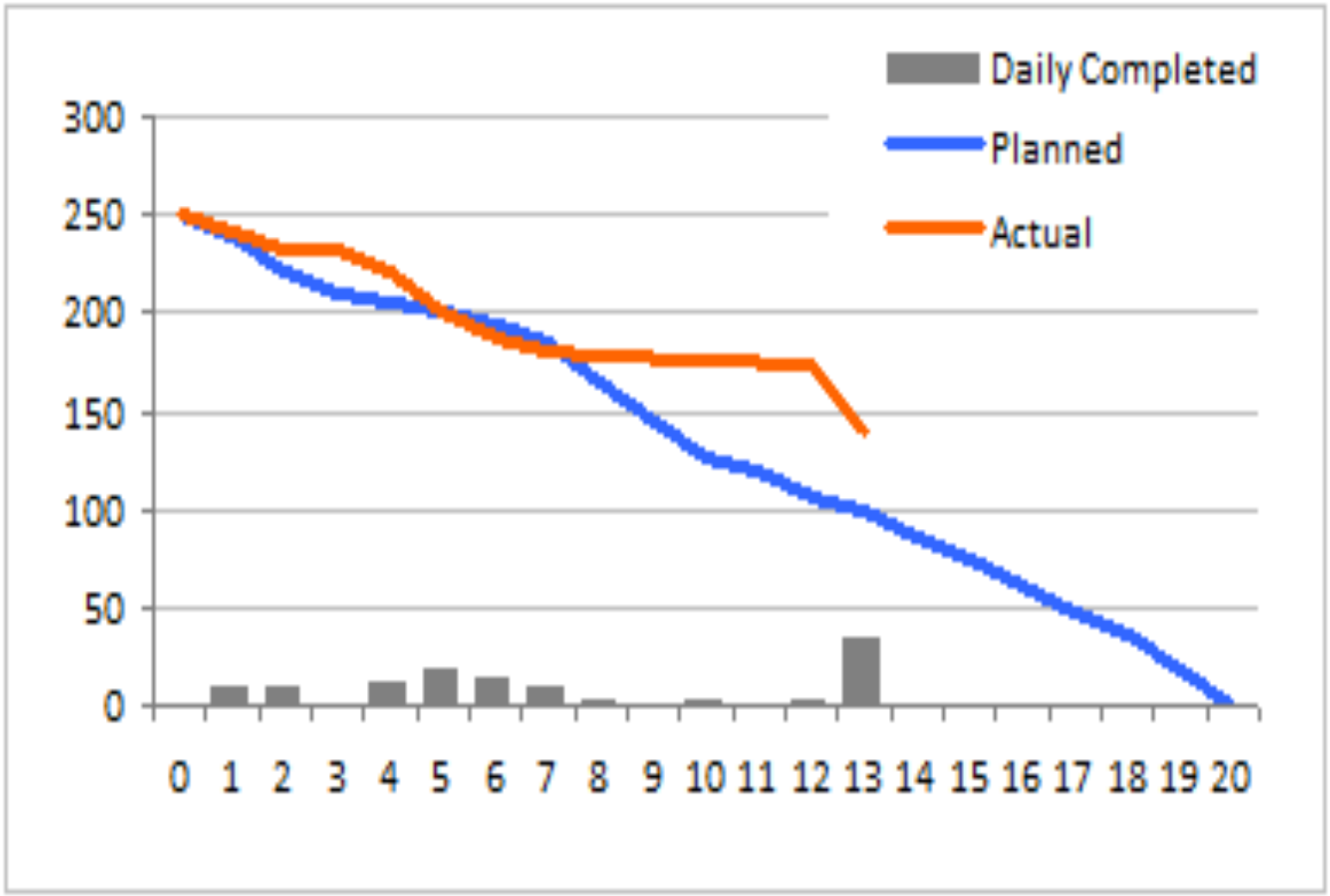


The Burndown Chart

- The X-axis: (real) time
  - hours, days, weeks, iterations, etc.
- The Y-axis: work remaining
  - developer-hours, developer-days, team-days, etc. required to complete the task
  - story points: the amount of effort required to implement a user story

- Are we progressing as planned? Are we ahead of plan? Are we behind?

The Burndown Chart



Sprint Review and Retrospective

At the end of a sprint, the team holds two events:

1. **Sprint review**:
   - Review the work that was completed and the planned work that was not completed
   - Present the completed work to the stakeholders (a.k.a. the demo)
     - Unfinished work cannot be presented
   - The team and the stakeholders collaborate on what to work on next
2. **Sprint retrospective**:
   - What went well during the sprint? What could be improved in the next sprint?
   - The team identifies and agrees on continuous process improvement actions

Scrum: Challenges and Limitations – 1/2

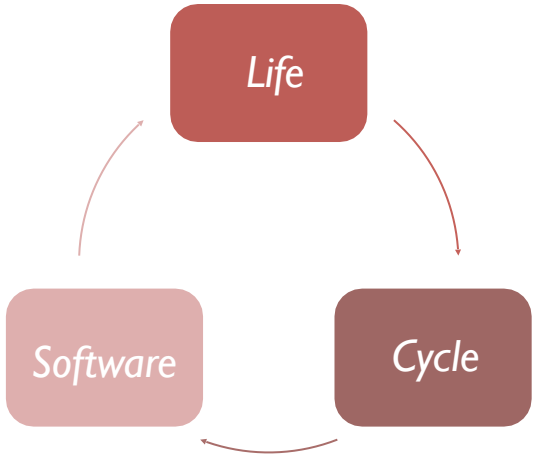
- Teams whose members are geographically dispersed or part-time
  - Recent improvements in technology have reduced the impact of these barriers
  - Yet, the Agile manifesto asserts that the best communication is face-to-face
- Teams whose members have very specialized skills
  - In Scrum, developers should be able to work on any task or pick up work that another developer has started

Scrum: Challenges and Limitations – 2/2

- Products with many external dependencies
  - Deliveries of software from other teams can lead to delays and the failure of individual sprints
  - Teams should allocate time for supporting other teams (office hours)

- Products with regulated quality control (e.g., medical devices or vehicle control)
  - Product increments should be fully developed and tested in a single sprint
  - Less suited for products that need large amounts of safety and compliance testing

Some Software Development Process Models



No Plans

- **Code-and-fix**: write code, fix it when it breaks

Plan and Document

- **Waterfall:** perform each phase in order (1970)

Incremental

- **Staged Delivery:** waterfall-like beginnings, then, short release cycle
- **Evolutionary prototyping:** develop a skeleton system and evolve it for delivery
- **Spiral**: triage/figure out riskiest things first (1988)
- **Agile**: *a family of principles* promoting adaptive planning, evolutionary development, early delivery, and continuous improvement (1970-2005+)
  - Most popular: **Scrum** and **Kanban**

What's the best model?

- Consider
  - The task at hand
  - Customer involvement and feedback
  - Predictability
  - Quality control

Summary

- The choice of a process model depends on the project circumstances, organizational culture, team composition, etc.
- Customize the process for your needs
  - Process models are often combined or tailored to the environment.
- Avoid overemphasizing process over results
  - Don’t become a “slave” of the process