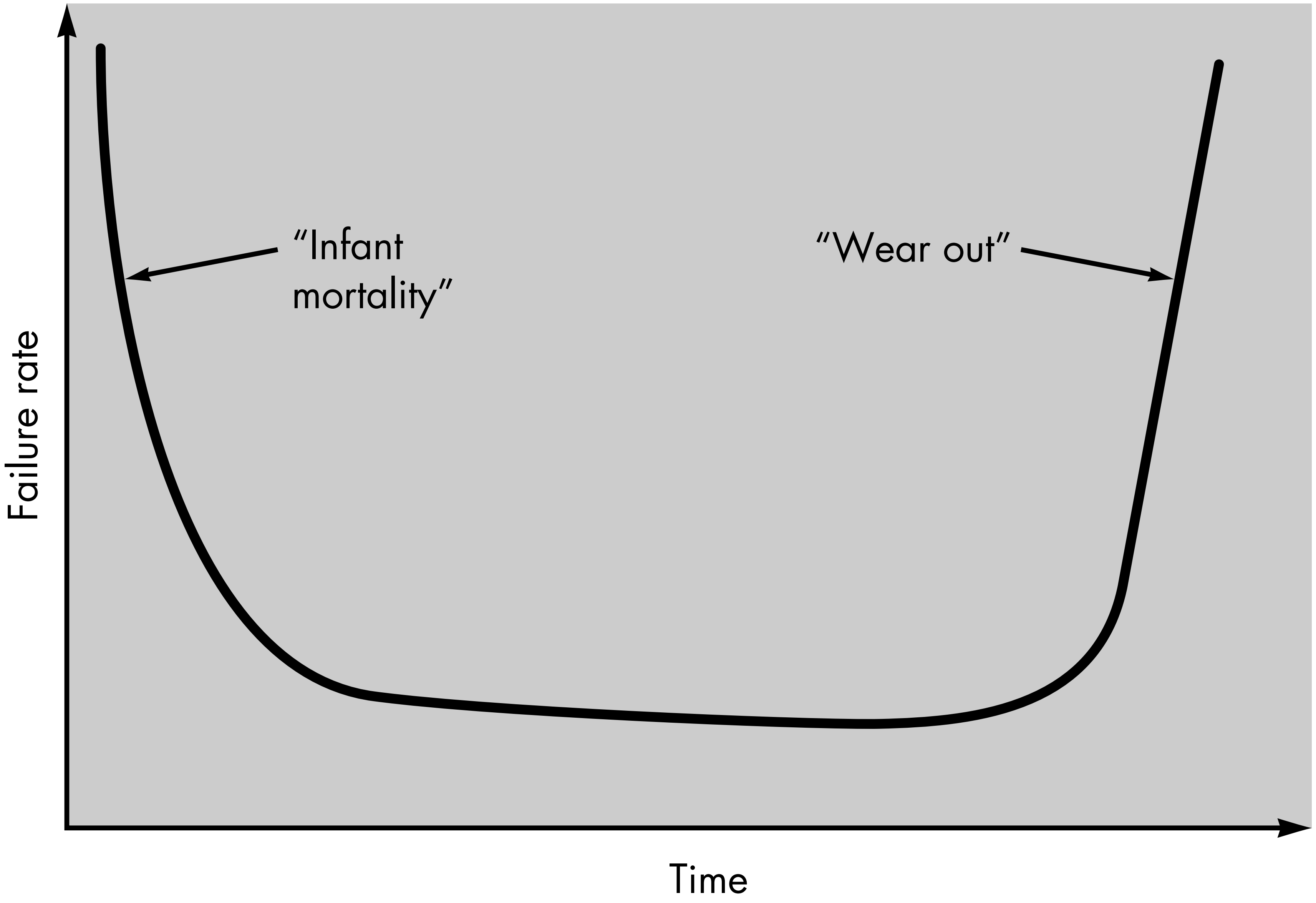
Software Development Lifecycle

Tuesday, November 19th

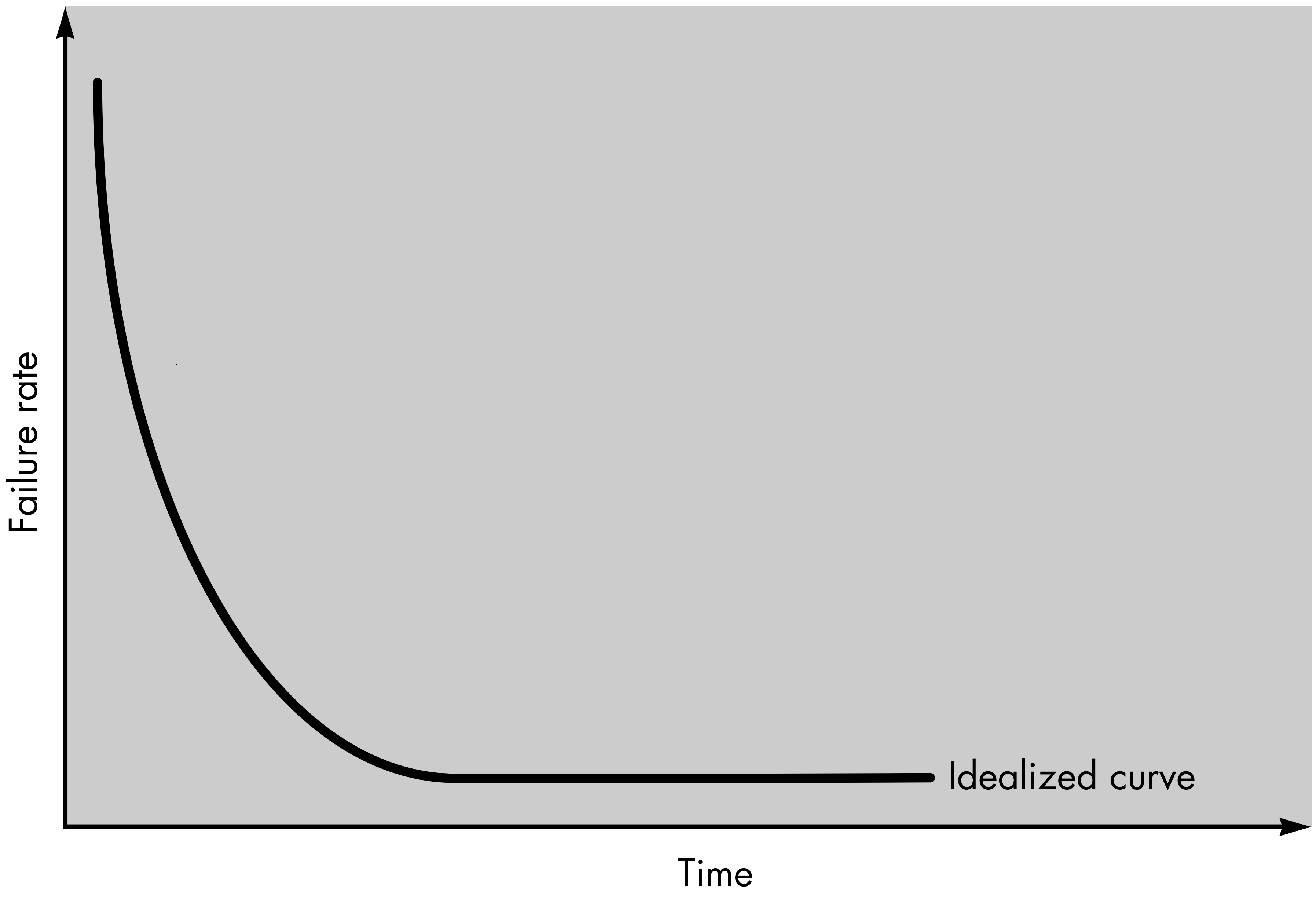
Oregon State University

Hardware wears out

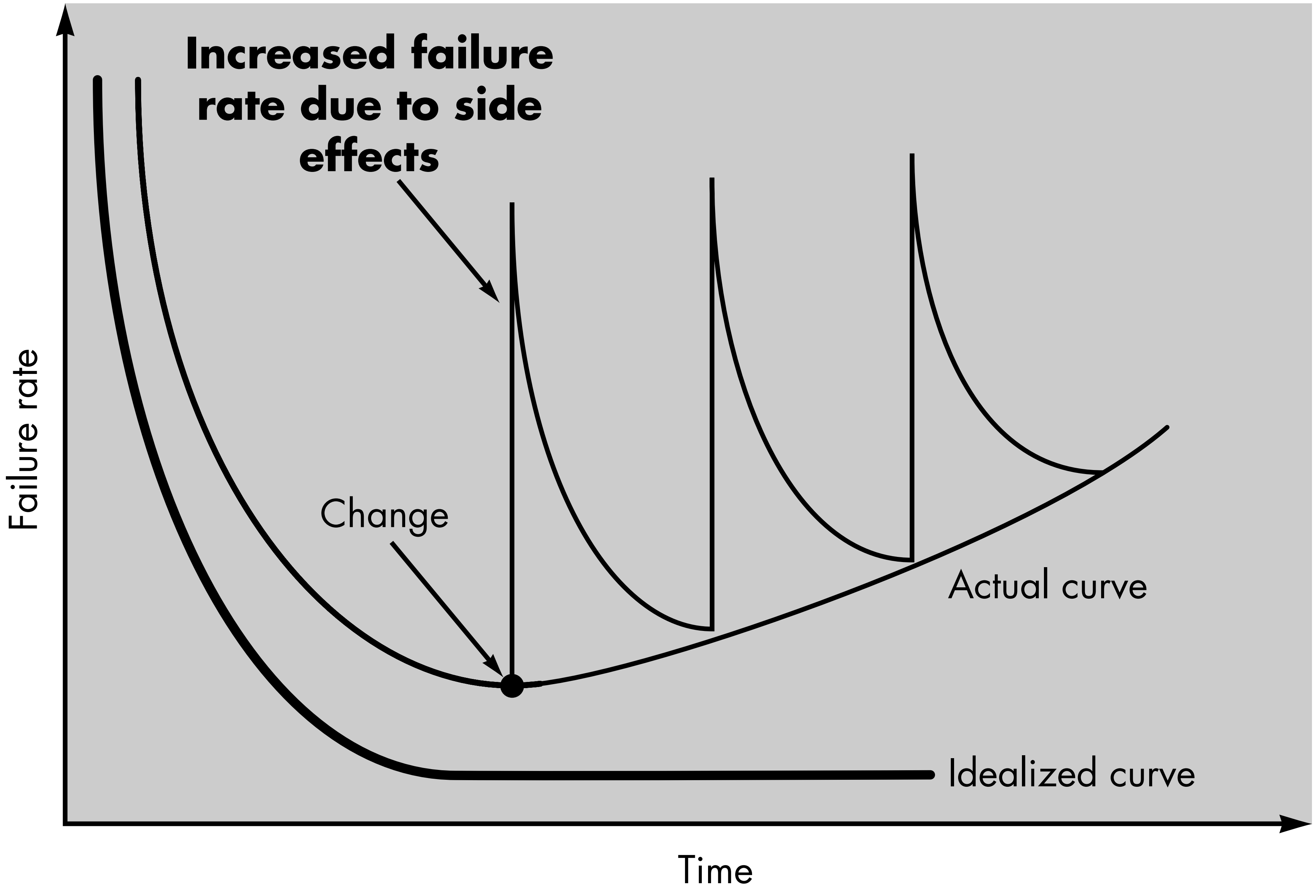


Failure curve for hardware

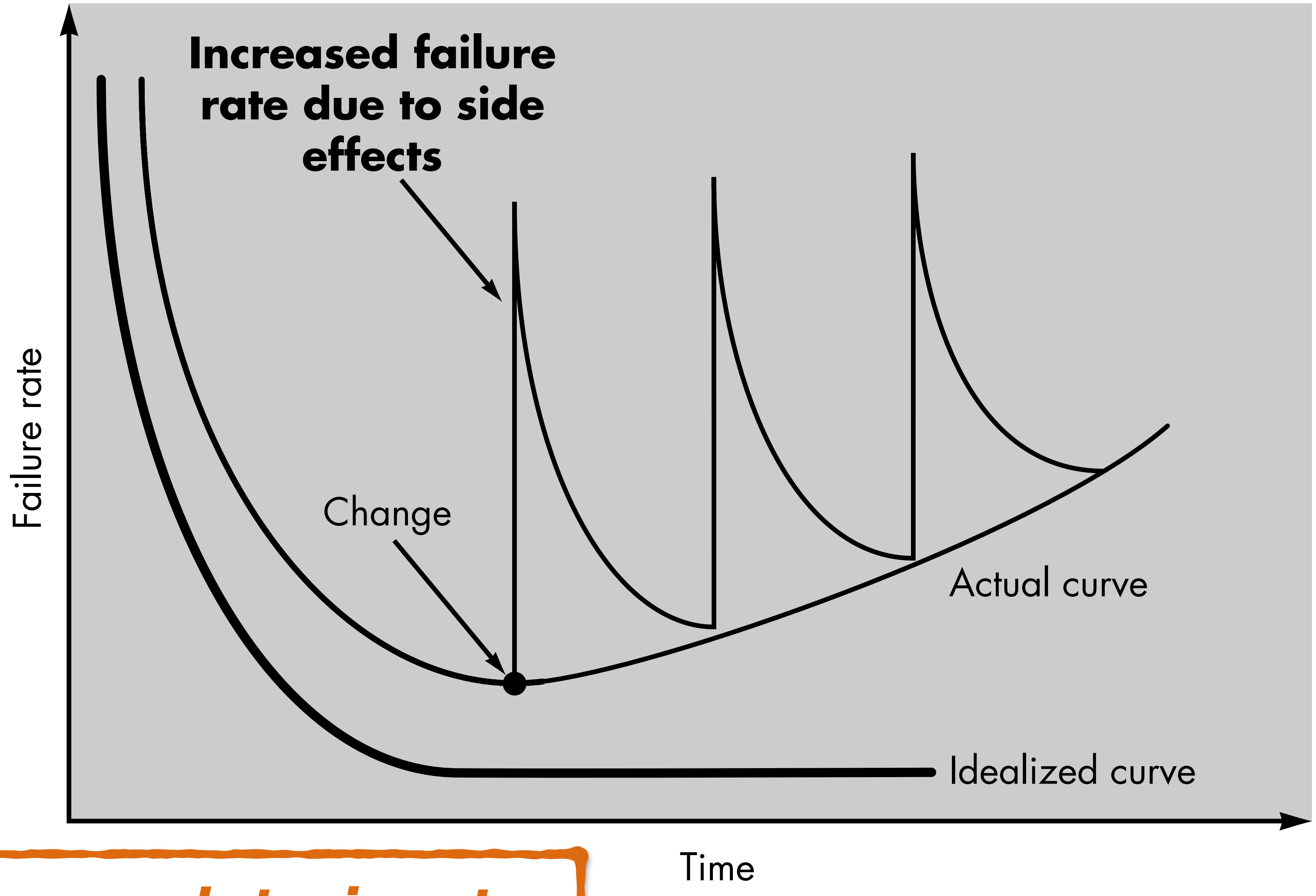
Software doesn't wear out



Software doesn't wear out

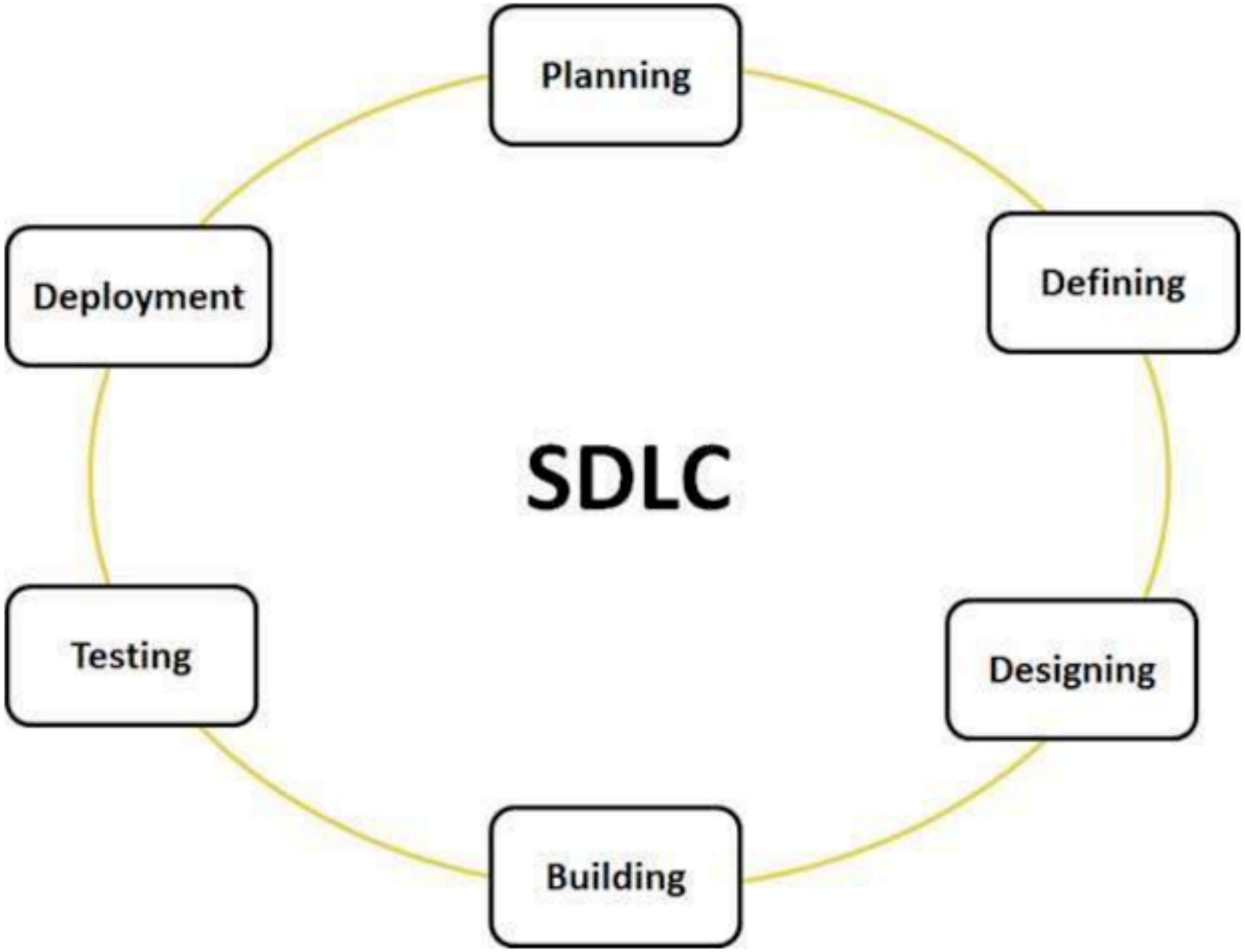


Software doesn't wear out



Software deteriorates

SDLC – Software Development Lifecycle



https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

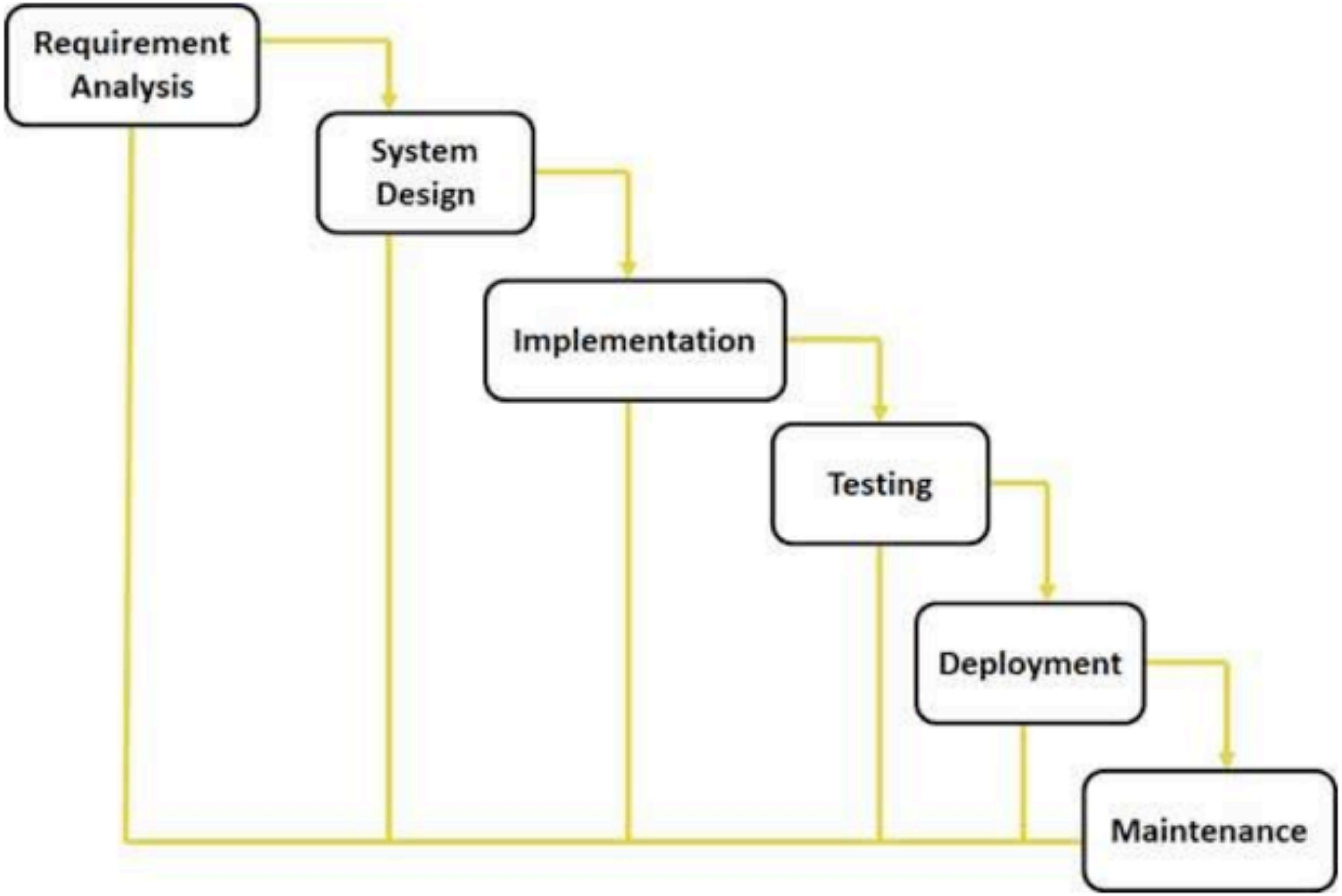
Big Bang Model

Develop code

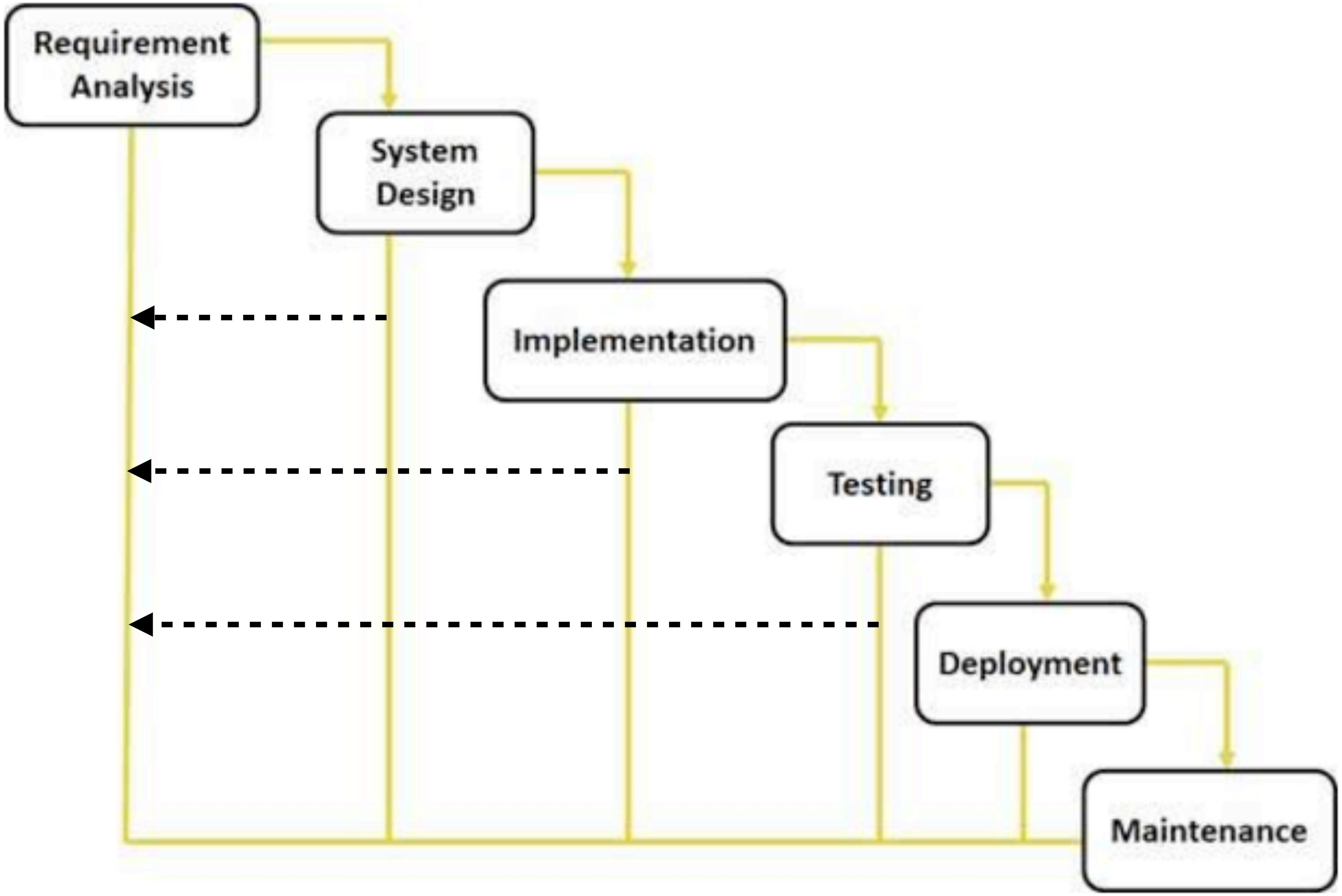
Understand requirements as you go ahead

Basically, no planning, defining, or designing

Waterfall


Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Waterfall


Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Pros

Pros

Well documented requirements & documentation

Easy to manage phases across teams

Cons

Cons

Rigid phases

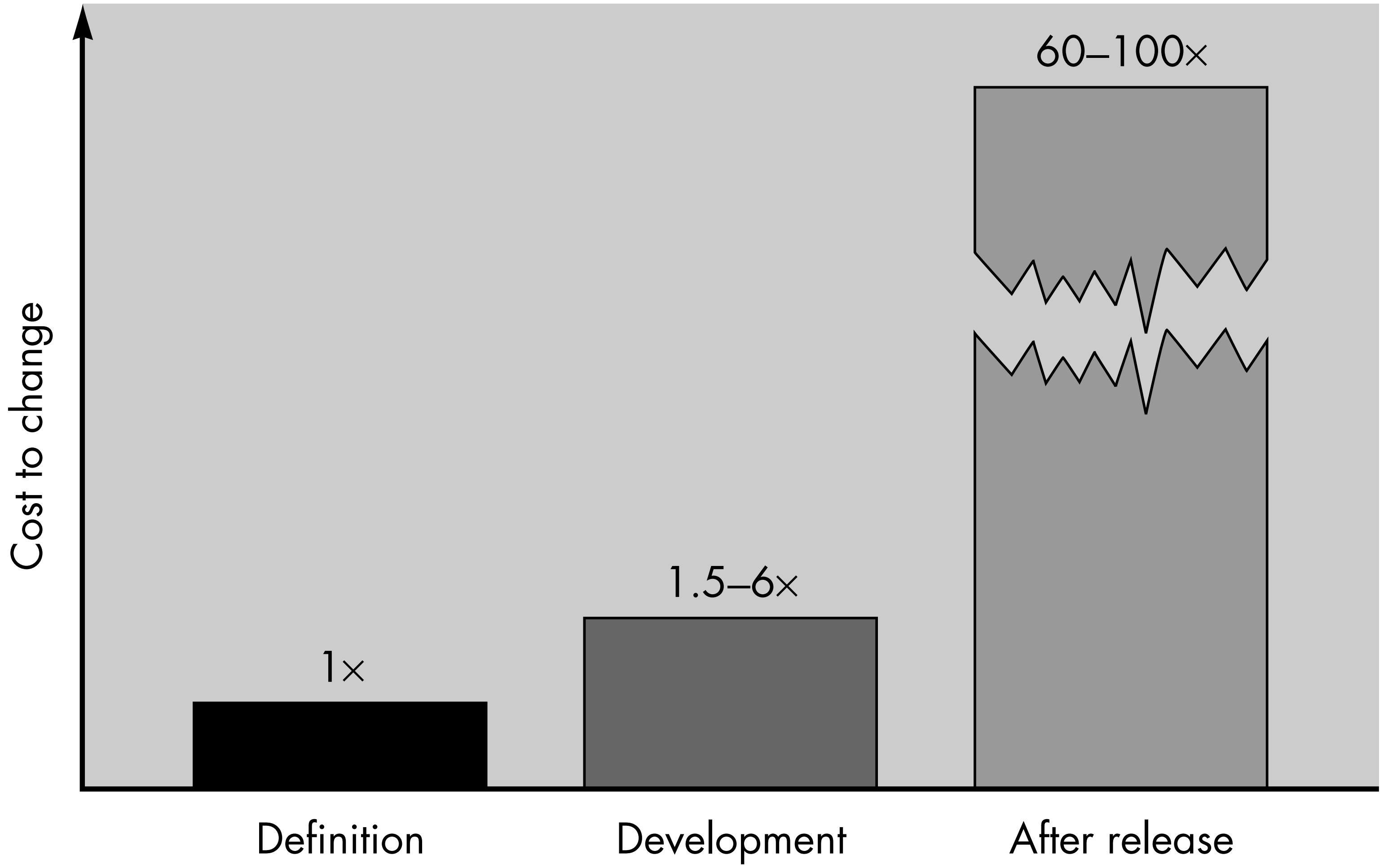
No working software until late stage

Not much reflection or revision

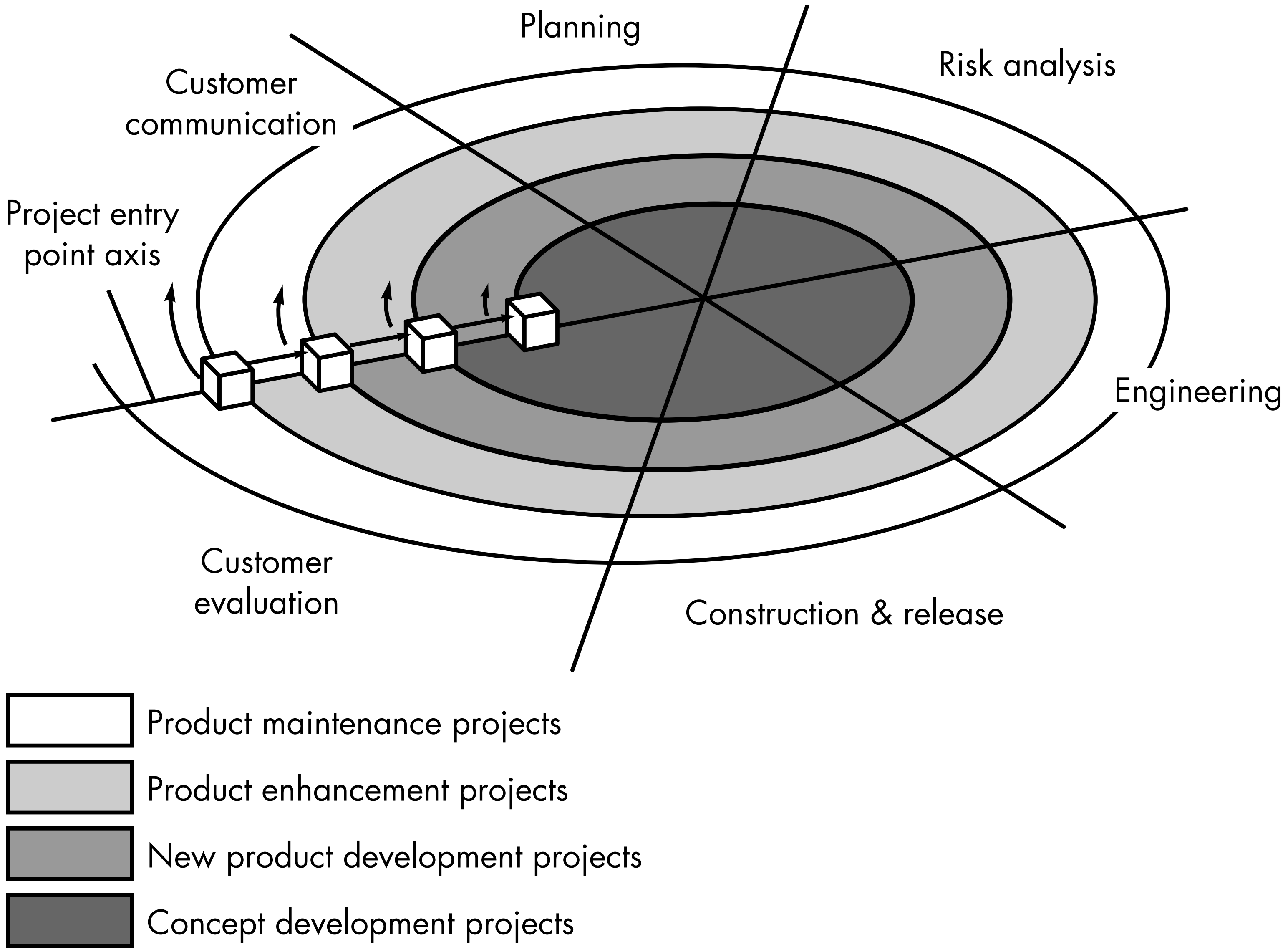
Big Bang Integration at the end

The impact of change

(for waterfall)



Spiral model



Pros

Pros

Used for medium – high risk projects

Complex and unclear requirements that need evaluation

Early involvement with system development & users

Cons

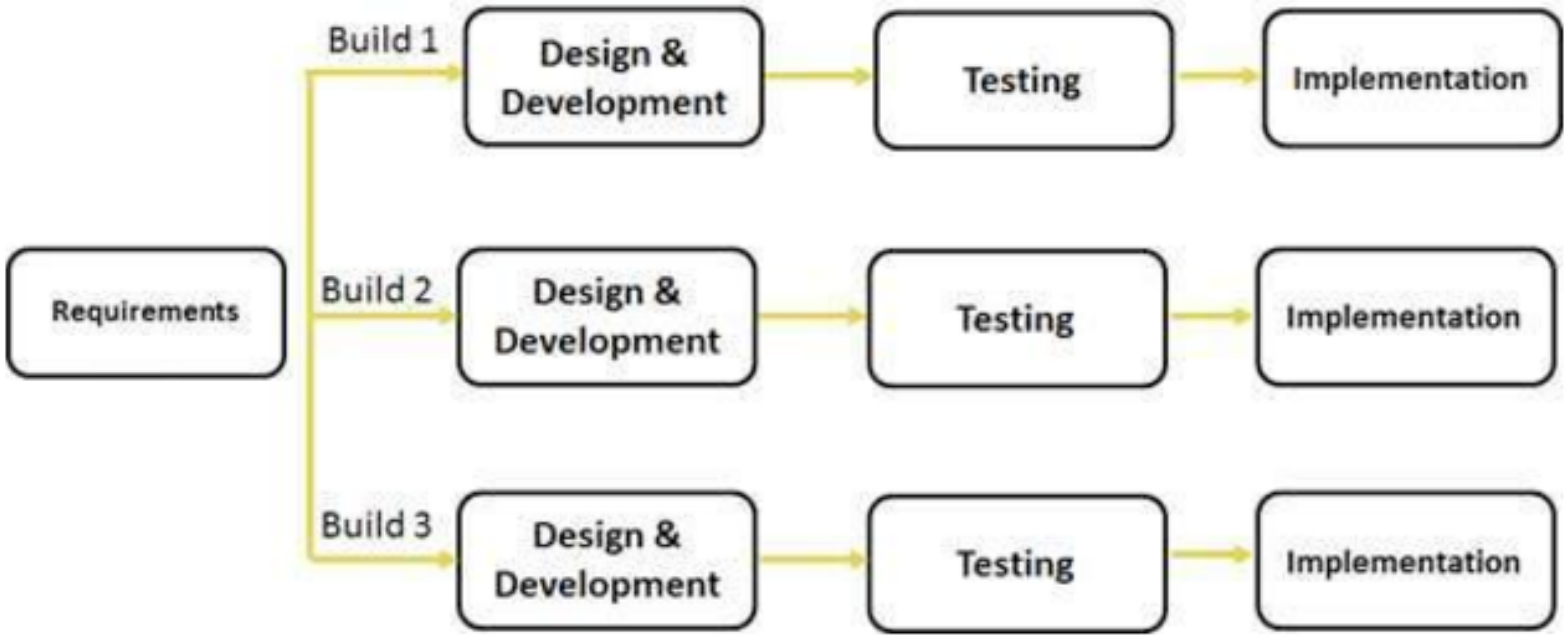
Cons

Management & process is complex

Large number of cycles require lots of documentation

When is end of cycle not always clear

Iterative model



Pros

Pros

Major requirements (and risks) are identified upfront

Working model at early stage

Parallel development can be planned

Suited for large, mission critical systems

Cons

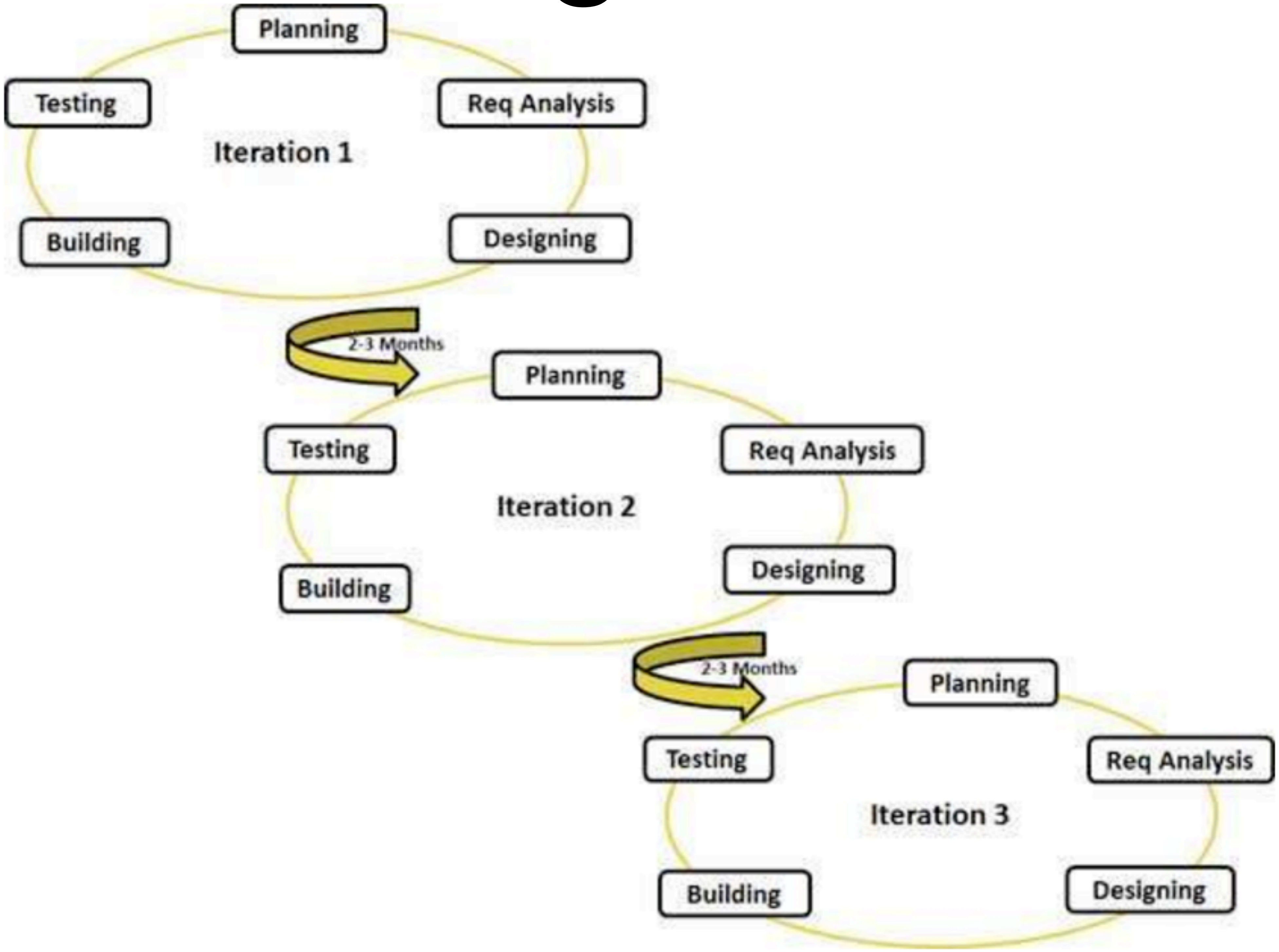
Cons

Defining iterations may require definition of complete system

Not all requirements is gathered upfront; changing requirements still expensive

Increased pressure on user engagement

Agile


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Pros

Pros

Manage changing requirements

Minimal planning or documentation

Promotes team work & collaboration

Quickly change directions

Cons

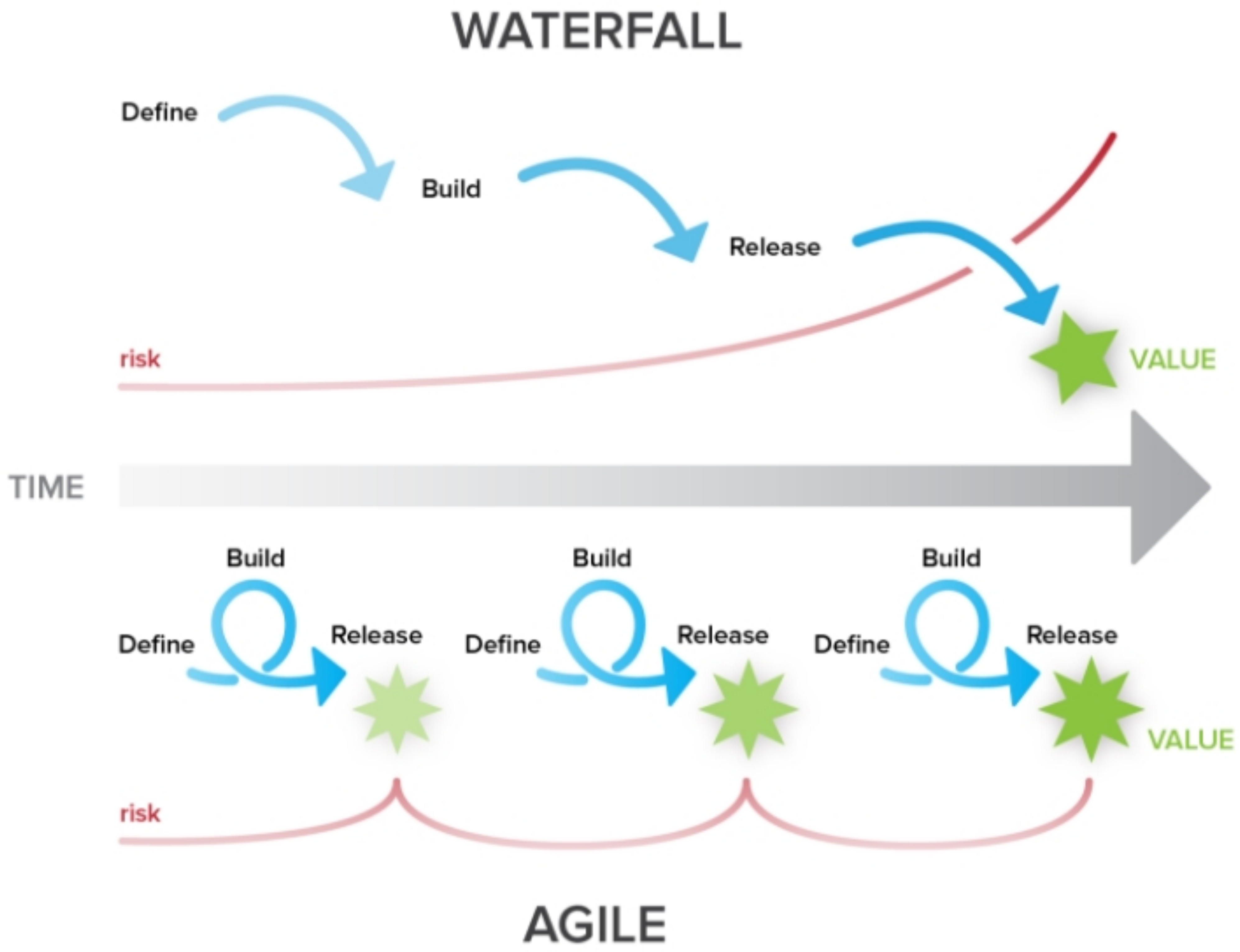
Cons

Overall plan/agile manager

Can't handle complex dependencies

Iterations determine scope of project

Heavy reliance on personnel (minimal documentation, newcomer onboarding, customer interaction)


WATERFALL
Define
Build
Release
risk
VALUE
TIME
Build
Build
Build
Define
Release
Define
Release
Define
Release
VALUE
risk
AGILE

Agile methods

Scrum

Kanban

Extreme Programming

DSDM (Dynamic Software Development Method)

Feature Driven Development (FDD)

Behavior Driven Development (BDD)

http://www.guru99.com/agile-scrum-extreme-testing.html

Extreme Programming

One the first agile methods

TDD, continuous integration, refactoring were originally introduced by XP.

XP Practices

Pair Programming

TDD

Continuous Integration

Refactoring

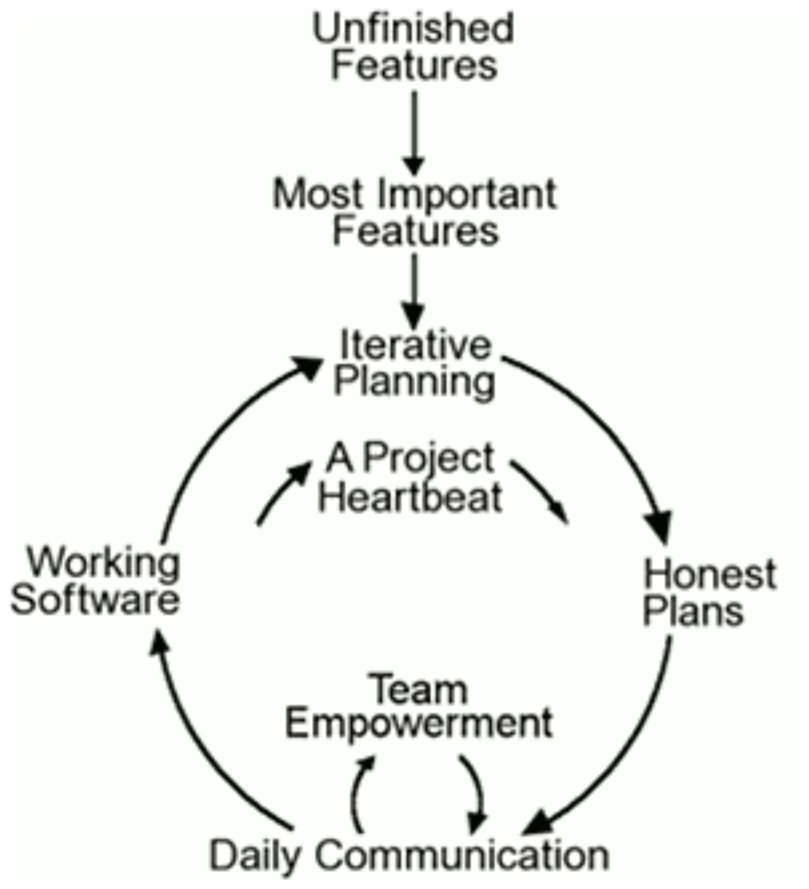
Small Releases

Coding Standards

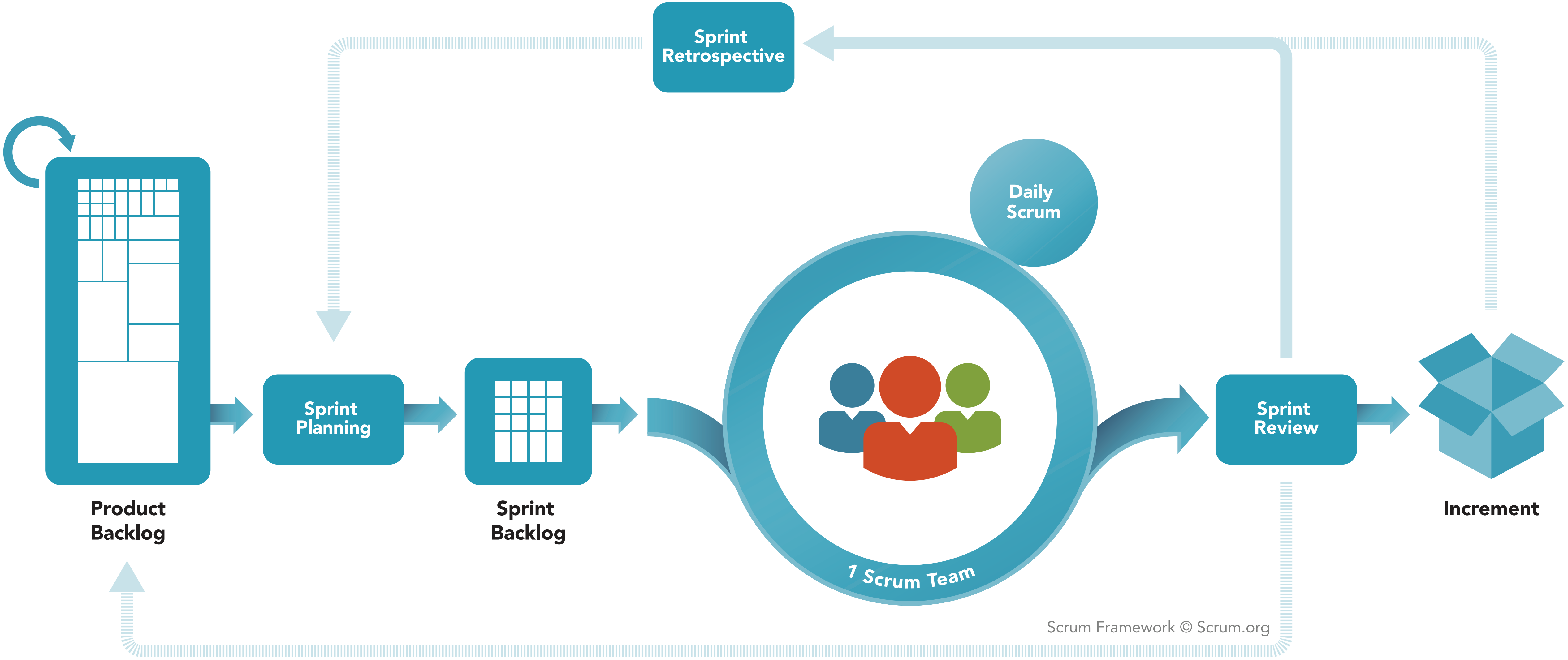
Collective Code Ownership

Simple Design

Sustainable Pace



Scrum



Scrum Framework © Scrum.org

Scrum terminology

Product Backlog: An ordered list of everything that is known to be needed in the product. A Product Backlog is never complete.

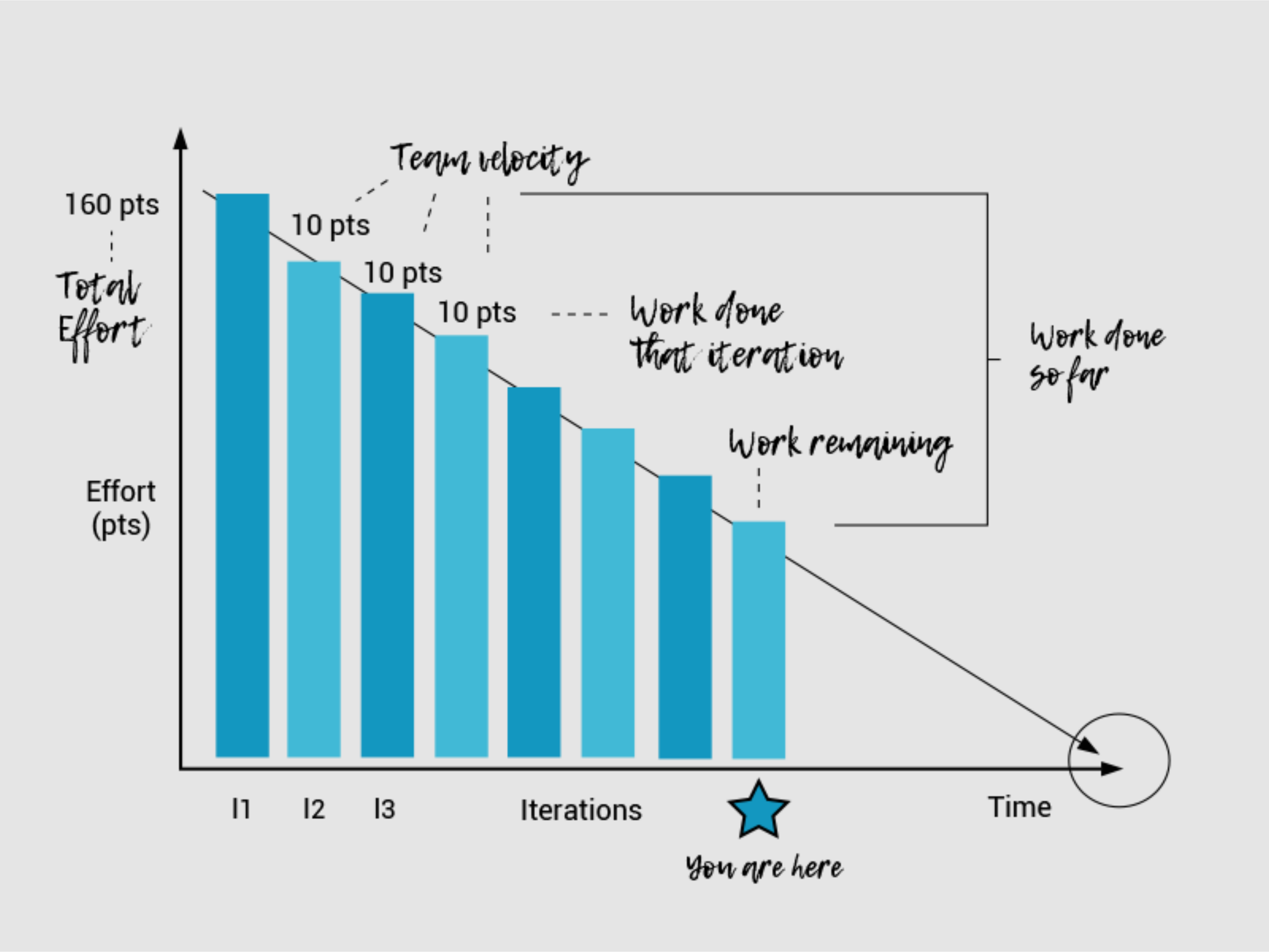
Increment: The sum of all the Product Backlog items completed during a Sprint plus the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be “Done.”

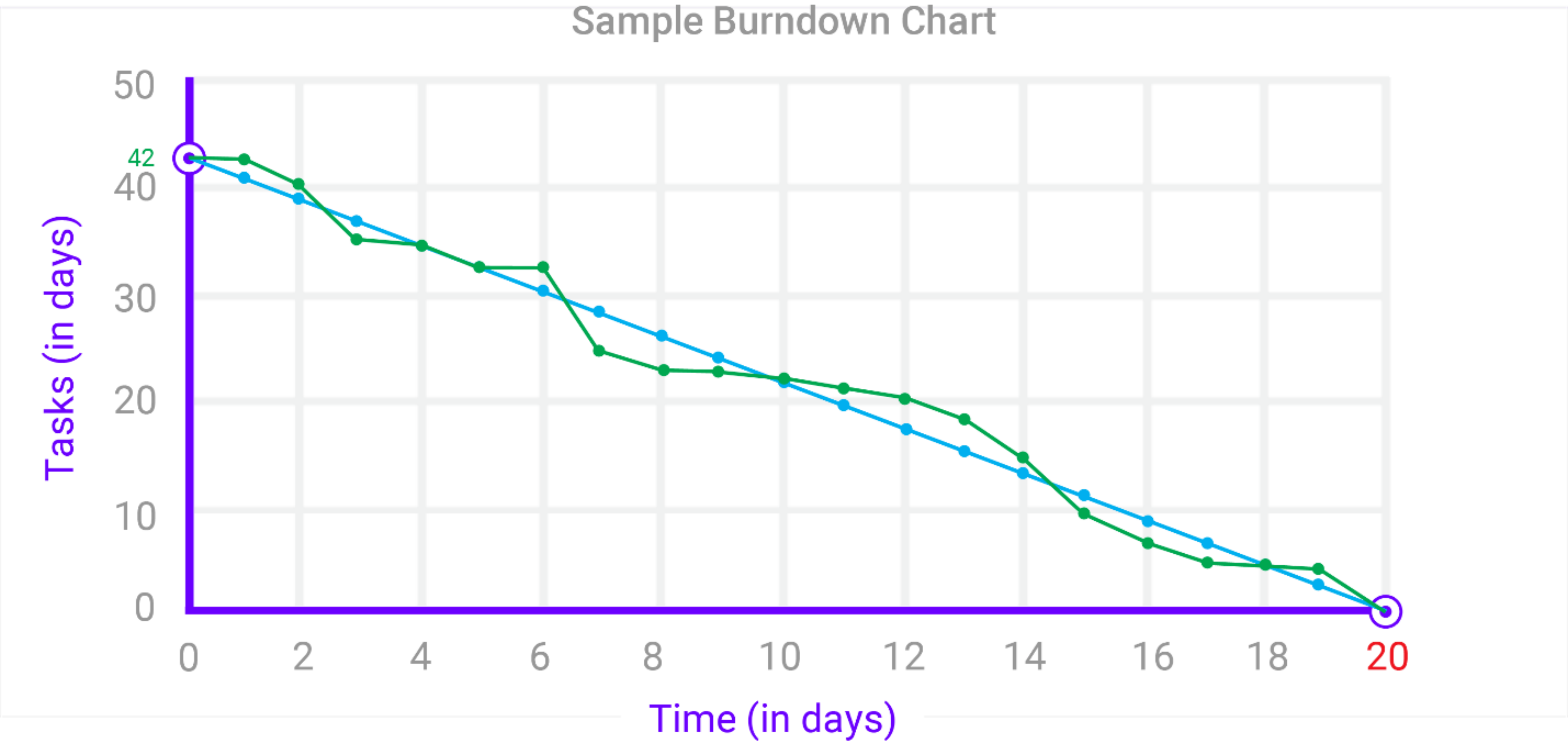
Sprint Backlog: the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal

Scrum terminology

Story Points: A unit of measure for expressing an estimate of the overall effort that will be required to fully implement a product backlog item or any other piece of work.

Velocity: The sum of all the story points that are "Done" at the end of the Sprint.


Team velocity
160 pts
Total Effort
10 pts
10 pts
10 pts
Work done that iteration
Work done so far
Work remaining
Effort (pts)
I1
I2
I3
Iterations
Time
You are here


Sample Burndown Chart
50
42
40
30
20
10
0
Tasks (in days)
0
2
4
6
8
10
12
14
16
18
20
Time (in days)

Scrum roles

Product Owner: is responsible for maximizing the value of the product resulting from the work of the Development Team.

The sole person responsible for managing the Product Backlog

Clearly expressing Product Backlog items.

Ordering the items in the Product Backlog

Scrum Roles

Development Team: consists of professionals who do the work of delivering a potentially releasable Increment of “Done” product at the end of each Sprint.

They are self-organizing. No one tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality.

Scrum Roles

Scrum Master: responsible for promoting and supporting Scrum.

Helping the team to reach consensus for what can be achieved during a specific period of time.

Removing obstacles that are impeding the team's progress.

Protecting the team from outside distractions.

Scrum Activities

Sprint planning:

What can be delivered in the Increment resulting from the upcoming Sprint?

How will the work needed to deliver the Increment be achieved?

Time-boxed to a maximum of eight hours for a one-month Sprint.

Scrum Activities

Daily Scrum: a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours.

Sprint Review: held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.

Scrum Activities

Sprint Retrospective: an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

The team discusses:

What went well in the Sprint

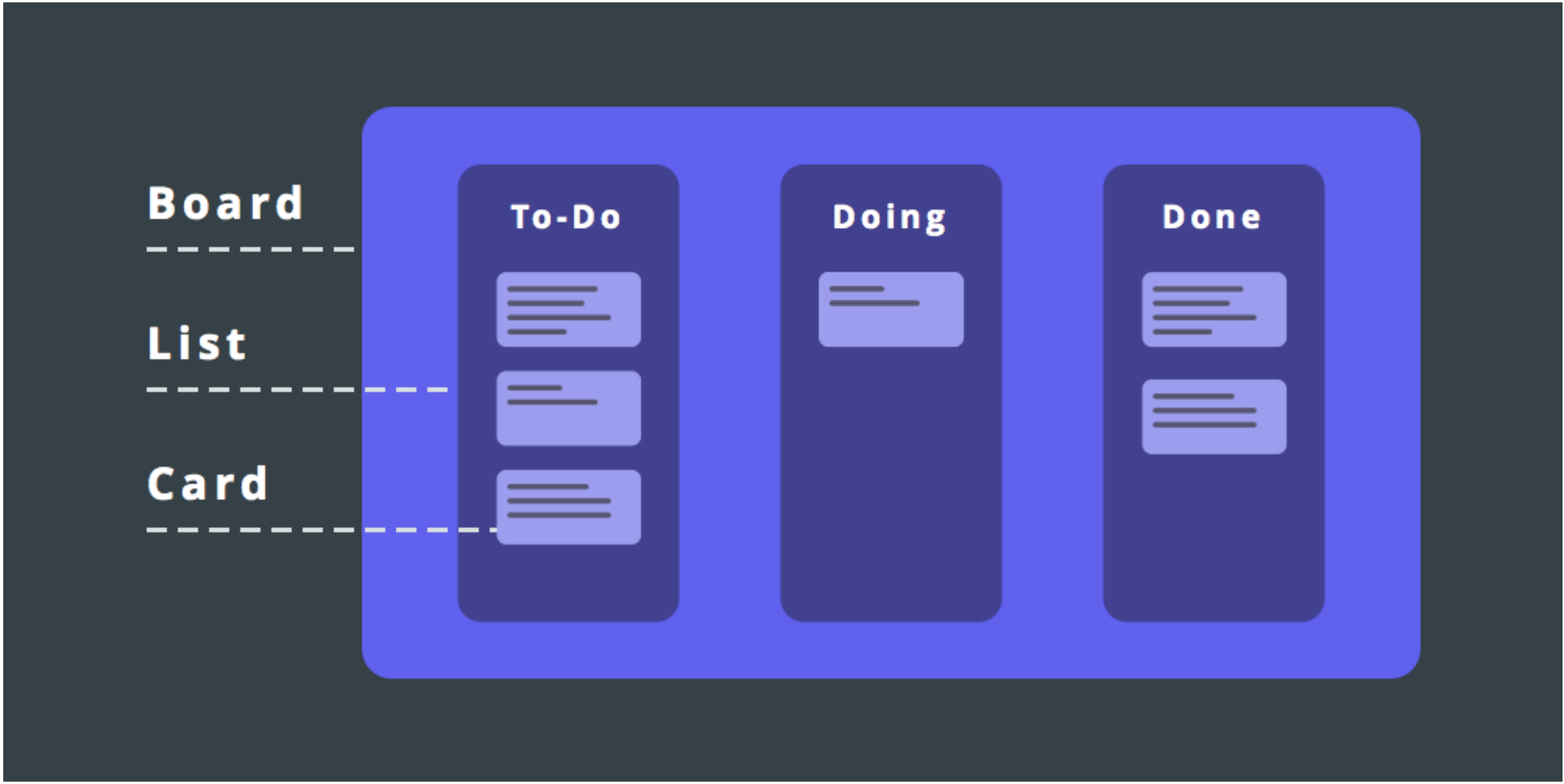
What could be improved

What will we commit to improve in the next Sprint

Kanban

Work flows **continuously** through the system, instead of being organized into distinct timeboxes.

Work items are represented visually on a **kanban board,** allowing team members to see the state of every piece of work at any time.


Board
List
Card
To-Do
Doing
Done

Work in Progress

In Kanban, Work in Progress is **limited.**

This allows the team to develop a **flow,** without loosing time switching between different tasks

The board allows the team to identify **blockers**, and clear them out quickly.

When to choose a particular kind of process

When to choose a particular kind of process

Waterfall is often a good choice for small systems whose requirements can be fully understood before any design or coding.

When to choose a particular kind of process

Waterfall is often a good choice for small systems whose requirements can be fully understood before any design or coding.

Spiral is often a good choice for larger systems with vague requirements and many alternatives for designing and coding.

When to choose a particular kind of process

Waterfall is often a good choice for small systems whose requirements can be fully understood before any design or coding.

Spiral is often a good choice for larger systems with vague requirements and many alternatives for designing and coding.

Agile is often a good choice for systems where you can rapidly create something very small but useful, and then expand from there.

Oregon State University