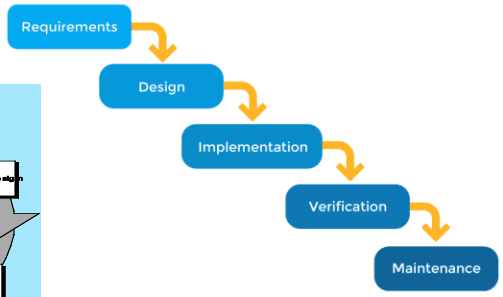
CMPS 310 Software Engineering
Fall 2021

Lecture 16

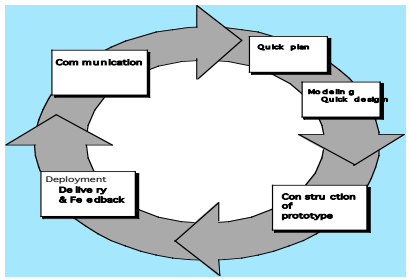
Software Life Cycle Process

Software Life Cycle Process Models

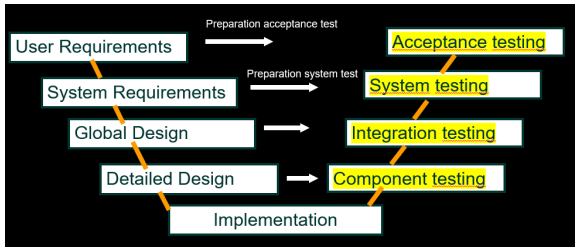
1. Waterfall Model



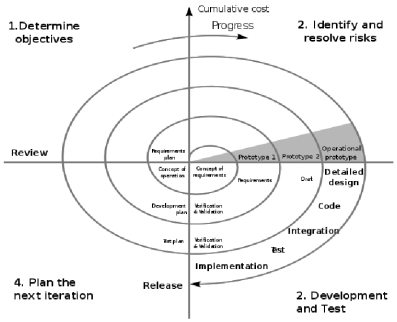
2. Prototyping



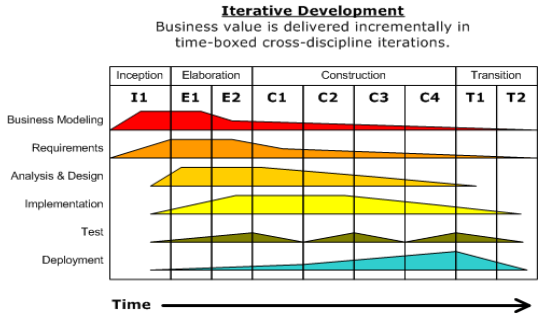
3. V-Model



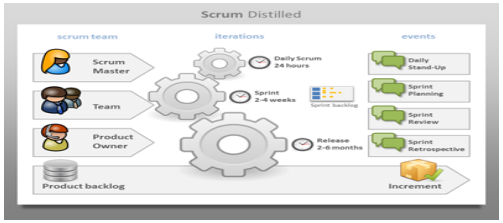
4. The Spiral Model



5. RUP



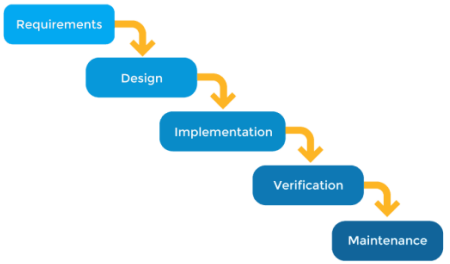
6. Agile process



Software Life Cycle Process and Artifacts

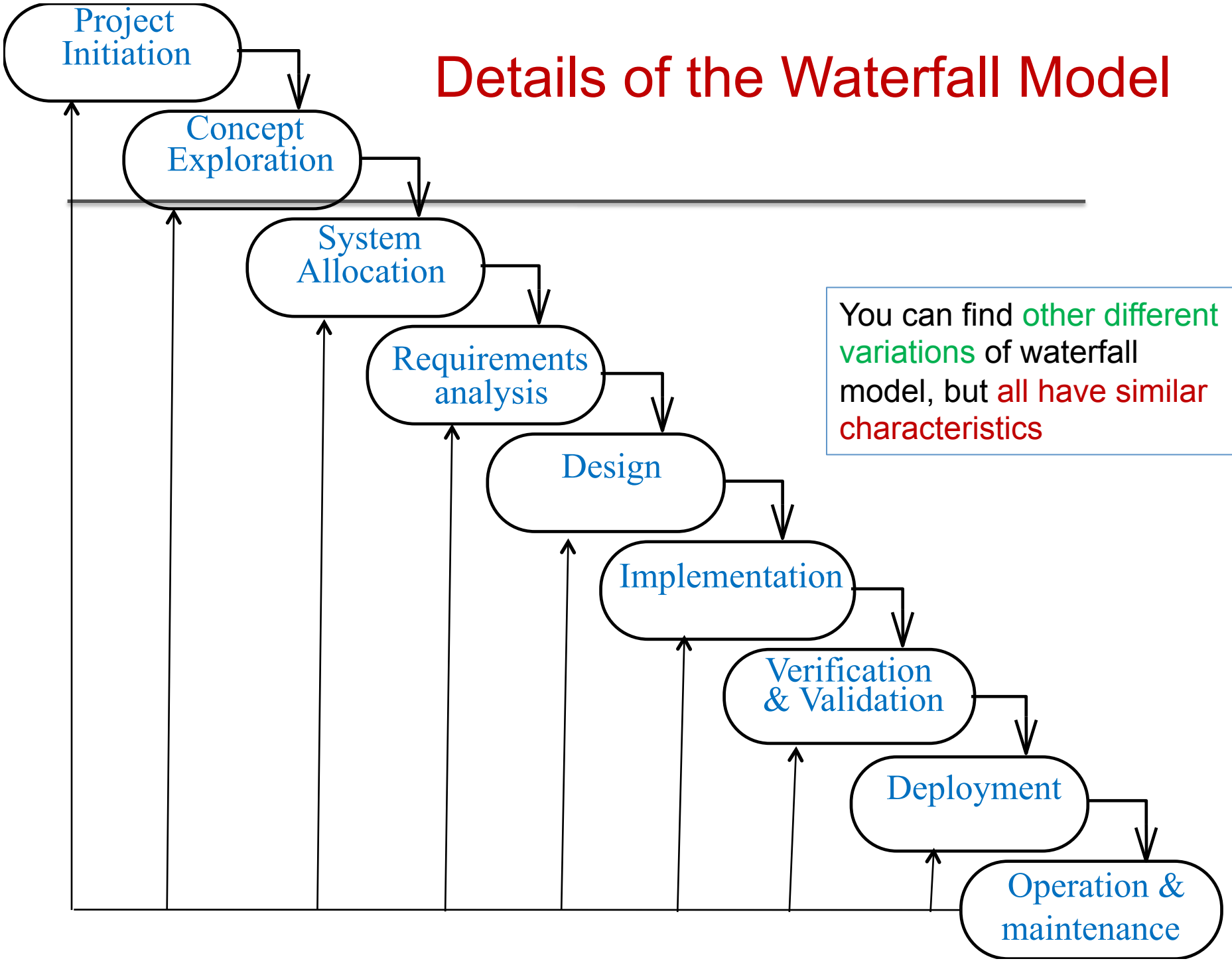
- **Software Process** is the method used to develop software
- It provides a systematic process model
- A process is based on a set of phases or activities
- These phases are ordered according to the process characteristics
- Artifacts are the tangible products of each phase
  - They form the output of phases or activities, and they can also be the input to subsequent phase or activity
  - Examples
    - A model: e.g., use case diagram, or the design model
    - A model element: e.g., a class, a use case, a controller
    - A document: e.g., software architecture
    - Source code
    - Executable code

1. Waterfall model



- Each phase ends with a milestone
- Waterfall model is inherently iterative
- Each milestone has an associated set of artifacts
  - Models or documents up to date with a phase
- Milestones
  - Allow managers to make crucial decisions before moving to the next phase
  - Provide a way to monitor progress
  - Generate data that can be useful for estimating time and staff requirements for other projects
  - are the intermediate or final target products

Details of the Waterfall Model



You can find other different variations of waterfall model, but all have similar characteristics

Properties of Waterfall Model

- One activity (phase) has to be completed before moving to the next activity (phase)
- Waterfall model is primarily document driven
- All requirements must be well understood and fixed from the beginning
- Managers love waterfall models because of
  - Clear milestones
  - Always one activity at a time
  - Possibility to revisit the previous phases
  - Easy to evaluate progress
  - Easy to understand

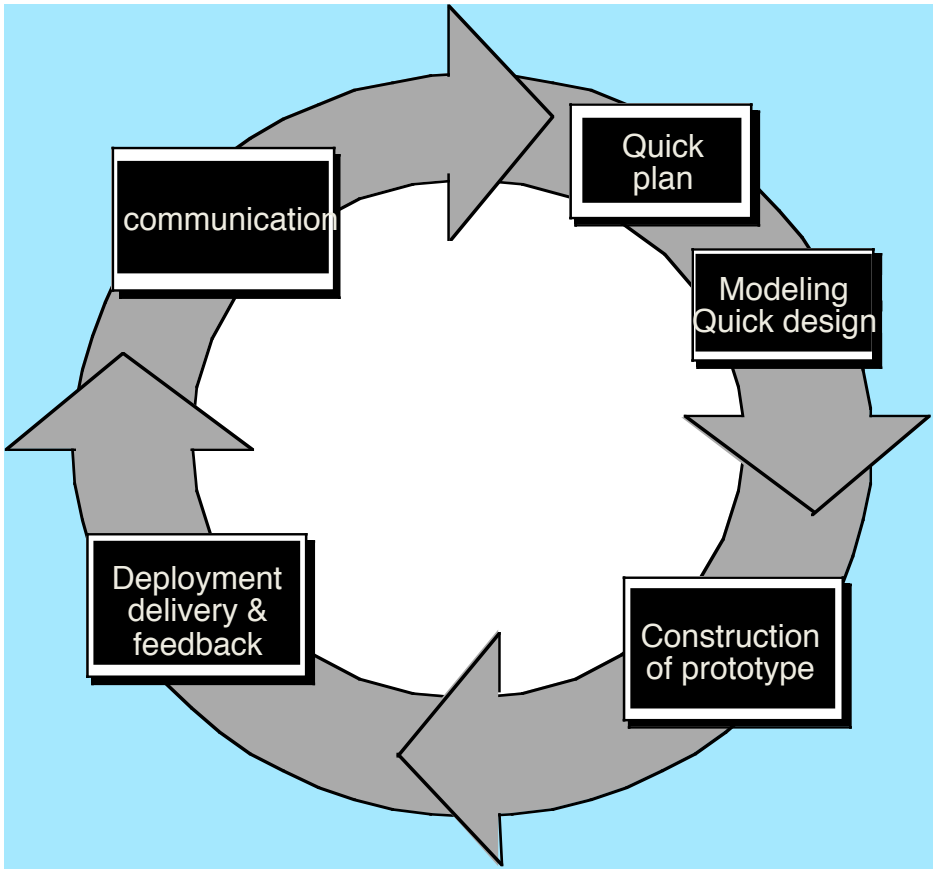
Problems of Waterfall model

- Difficult to accommodate changes
- Difficult to respond to changing customer requirements
- Waterfall model does not explicitly address risks
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements
  - This model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process
  - Few business systems have stable requirements

Advantages

- Easy to understand and implement
- Systematic and disciplined approach
- Reinforces good habits: define-before-design, design-before-code
- Identifies deliverables and milestones
- Document driven: *People leave, documents don't*
- Documentation standards available. e.g., ESA PSS-05 http://www.esa.int/TEC/Software_engineering_and_standardisation/TECBUCUXBQE_0.html
- Works well on large/mature products and weak teams

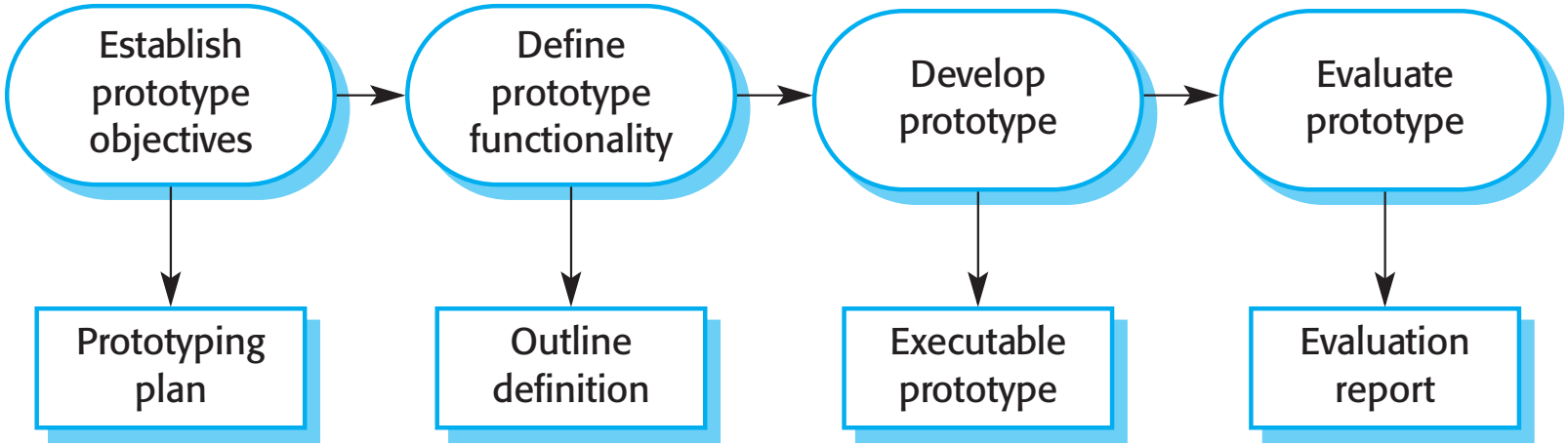
2. Software Prototyping



Software prototyping

- A prototype is an initial version of a system used to demonstrate concepts and try out design options
- A prototype can be used in:
  - requirements engineering process to help with requirements elicitation and validation
  - design processes to explore options and develop a UI design
  - testing process

The process of prototype development

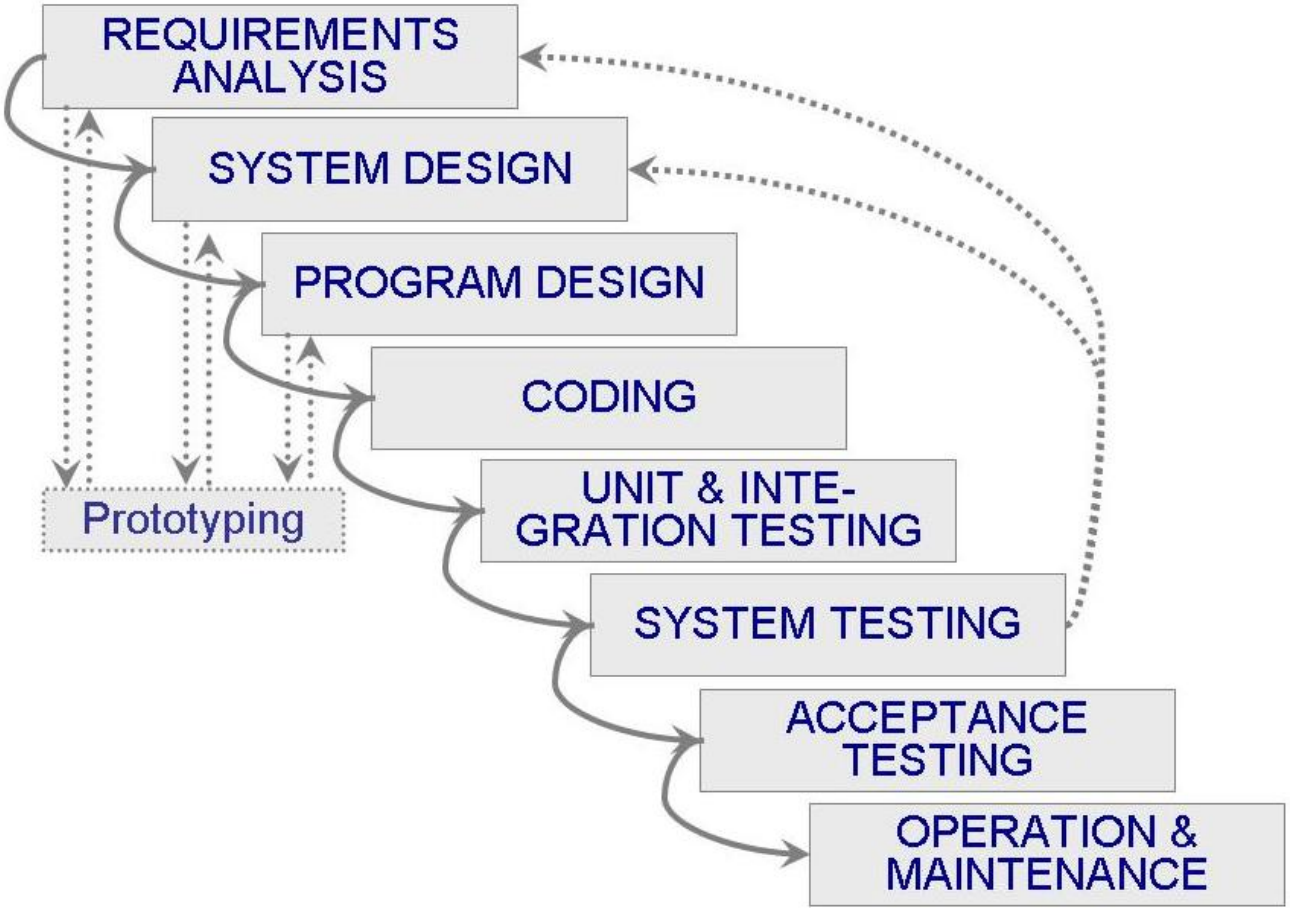


Benefits of prototyping

- Improved system usability
- A closer match to users' real needs
- Improved design quality
- Improved maintainability
- Reduced development effort

Waterfall Model with Prototype

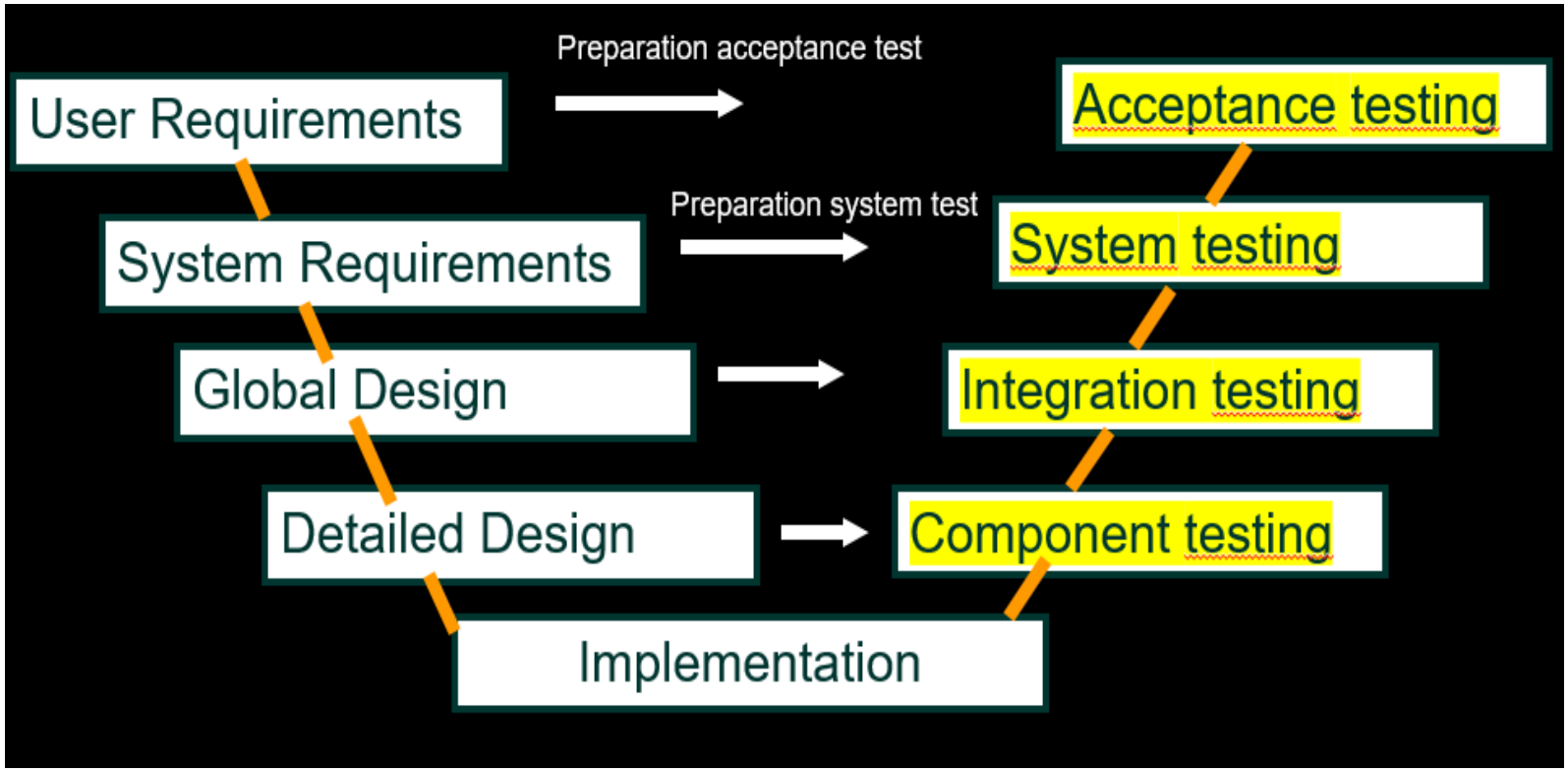
✧ Waterfall model with prototyping



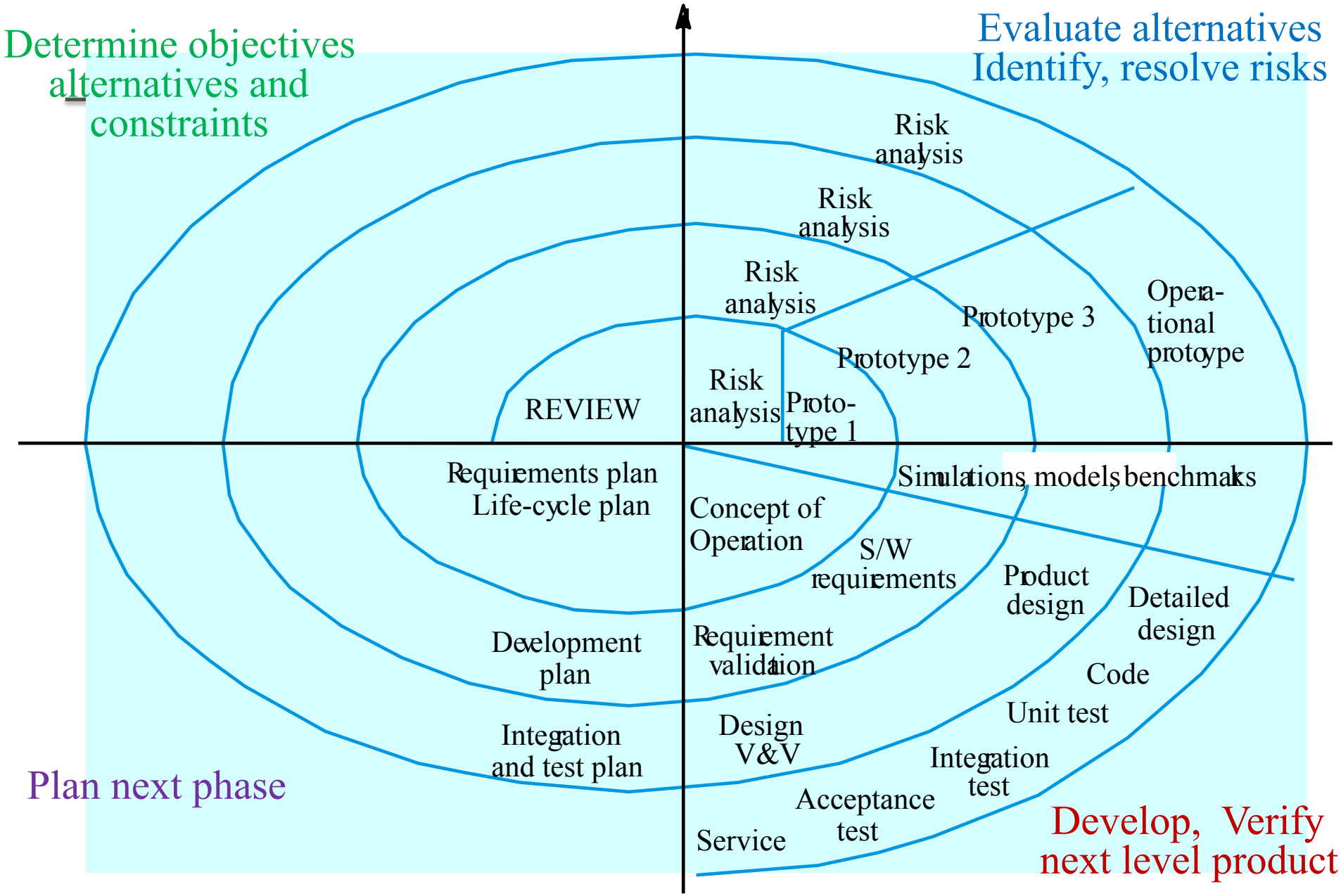
3. V-Model

- V-model is a model that illustrates how testing activities can be integrated into each phase of the life cycle
- Developed to address some of the problems experienced using the waterfall approach
- Defects were being found too late in the life cycle, as testing was not involved until the end of the project
  - Testing needs to begin as early as possible
  - Testing is not only an execution-based activity
  - A variety of activities need to be performed before the end of the coding phase

3. V Model


Preparation acceptance test
User Requirements
Acceptance testing
Preparation system test
System Requirements
System testing
Global Design
Integration testing
Detailed Design
Component testing
Implementation

4. Spiral Model of the Software Life Cycle



4. Boehm's Spiral Model

- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process
- Prototypes are used to explore the system' risky aspects
  - Risk of developing the "wrong" system (what customer doesn't want), a prototype can be a user interface without functionality
  - Other technical risks – e.g. performance, using a new technology, alternative algorithms
- Prototype may be thrown away or evolve into product

Spiral Model Phases

- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process: *requirements analysis, design, coding & testing*
- For each phase, perform the following
  - **Plan**: resource planning, schedule estimation
  - **Evaluate alternatives** applicable for the stage considering the objectives and constraints before making the decision
  - **Analyze risks**: risks are identified and prioritized based on the probability of their occurrence, a mitigation plan is drawn to manage the risks
  - **Engineering**: perform the activities of the stage

Advantages

- Realism: the model accurately reflects the iterative nature of software development on projects with unclear/ complex requirements
- Flexible: incorporates the advantages of the waterfall and evolutionary methods
- Comprehensive model decreases risk
- Good project visibility

Disadvantages

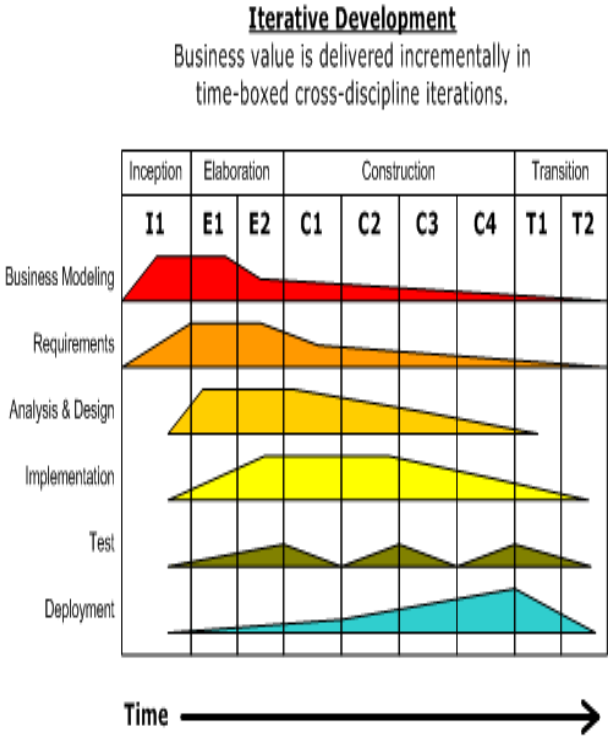
- ✧ Needs technical expertise in risk analysis and risk management to work well
- ✧ Model is poorly understood by nontechnical management, hence not so widely used
- ✧ Complicated model, need competent professional management. High cost and administrative overhead
- ✧ Not suitable for fixed budget projects

5. Rational Unified Process

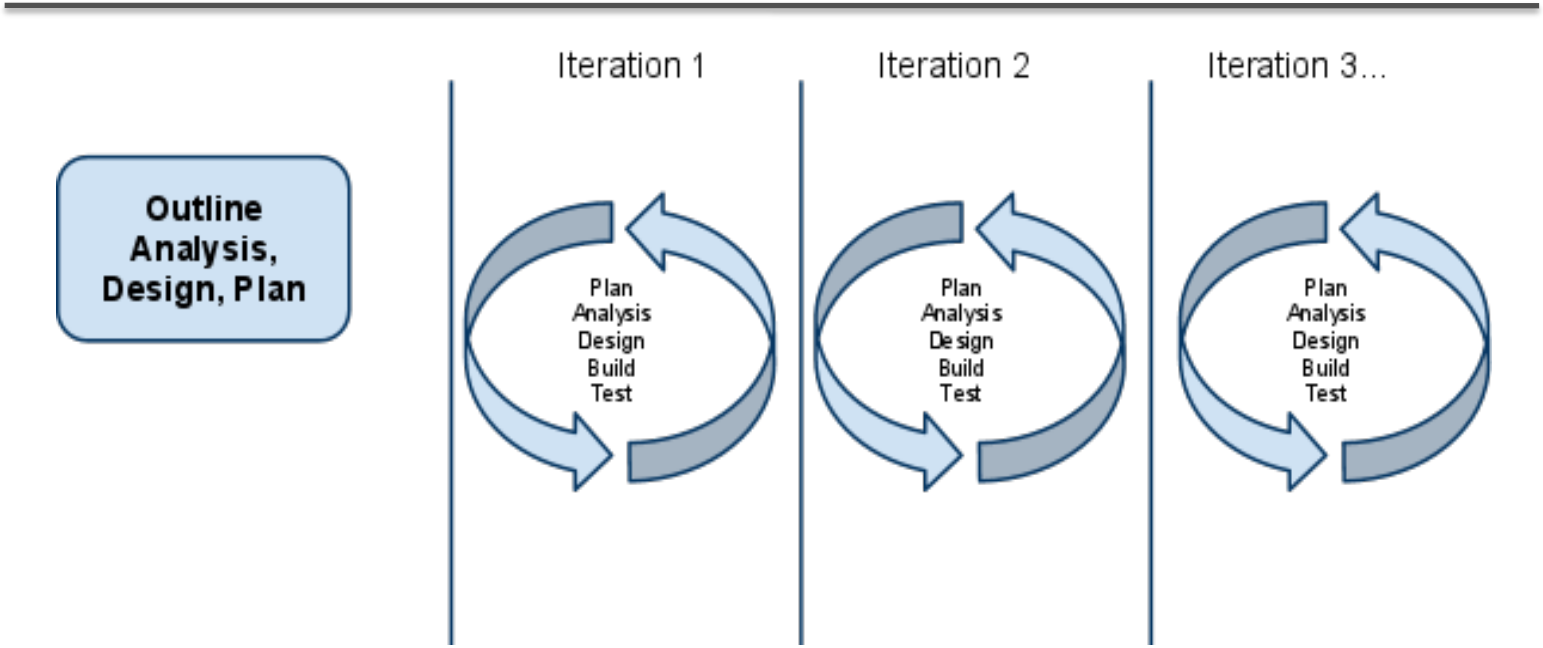
- An iterative software development process framework
- Created by the Rational Software Corporation, a division of IBM since 2003
- RUP is a specific implementation of the unified process
- RUP is included in the IBM Rational Method Composer (RMC) product which allows customization of the process
- In 2006, IBM created a subset of RUP tailored for the delivery of Agile projects - released as an OpenSource method called OpenUP through the Eclipse website

5. Rational Unified Process

- A life-cycle consisting of four phases
  - Inception, Elaboration, Construction, Transition
- Six engineering disciplines
  - Business modelling, requirements, analysis & design, implementation, test, deployment
- IBM Rational Method Composer product a tool for authoring, configuring, viewing, and publishing processes
  - Open source version Eclipse Process Framework (EPF) project



6. Agile Approaches: Scrum



- ✧ Working solution in every iteration
  - Review and refine regularly

Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams

Manifesto for Agile Software Development

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

Individuals and interactions	⇐	Over process and tools
Working software	⇐	Over comprehensive documentation
Customer collaboration	⇐	Over contract negotiation
Responding to change	⇐	Over following a plan

http://agilemanifesto.org/

Waterfall vs. Agile

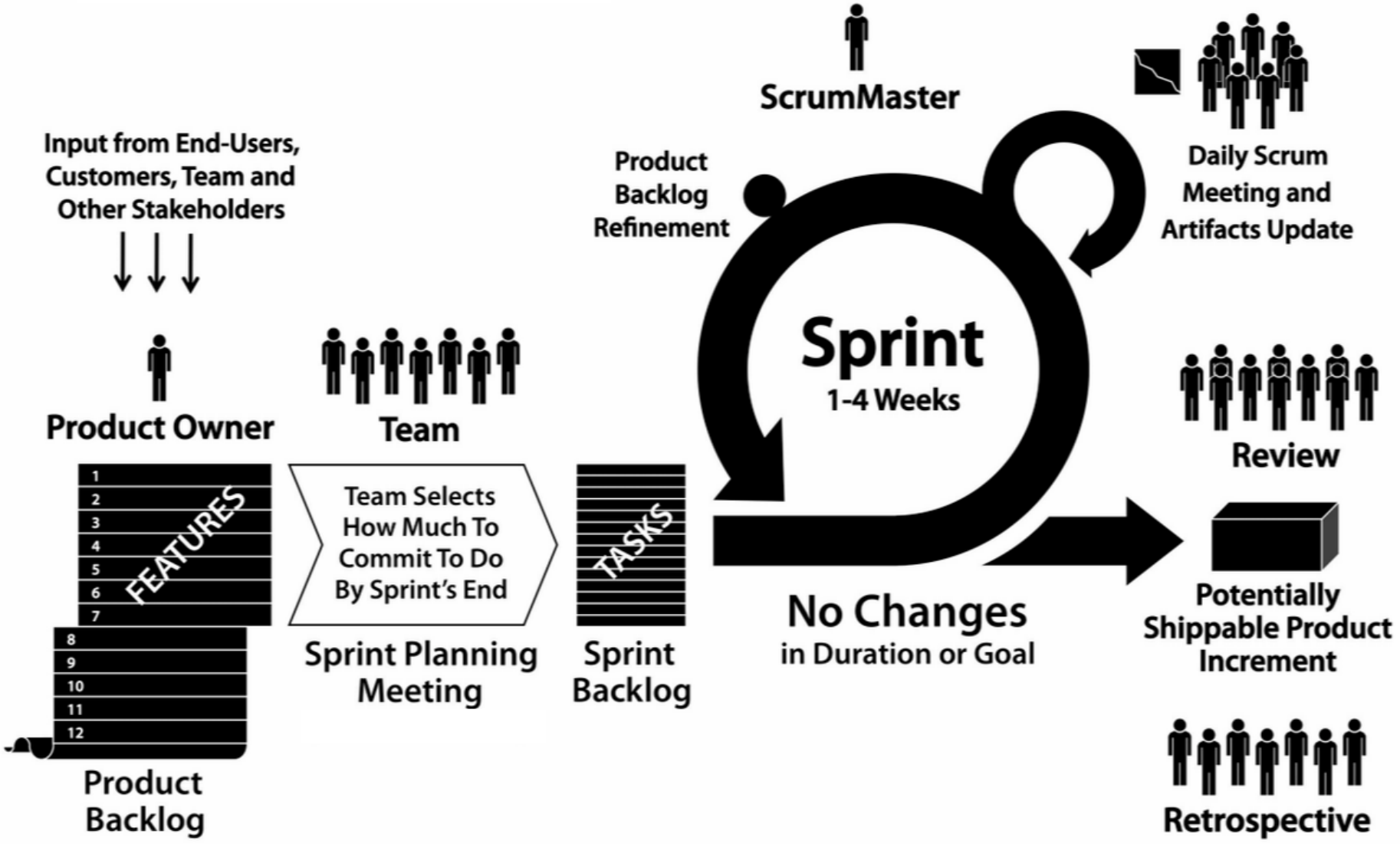
✧ Waterfall: emphasize **Structure**

- If you 100% know exactly what is wanted and everything is predictable then do waterfall !

✧ Agile: emphasize on **Adaptability** (change responsiveness)

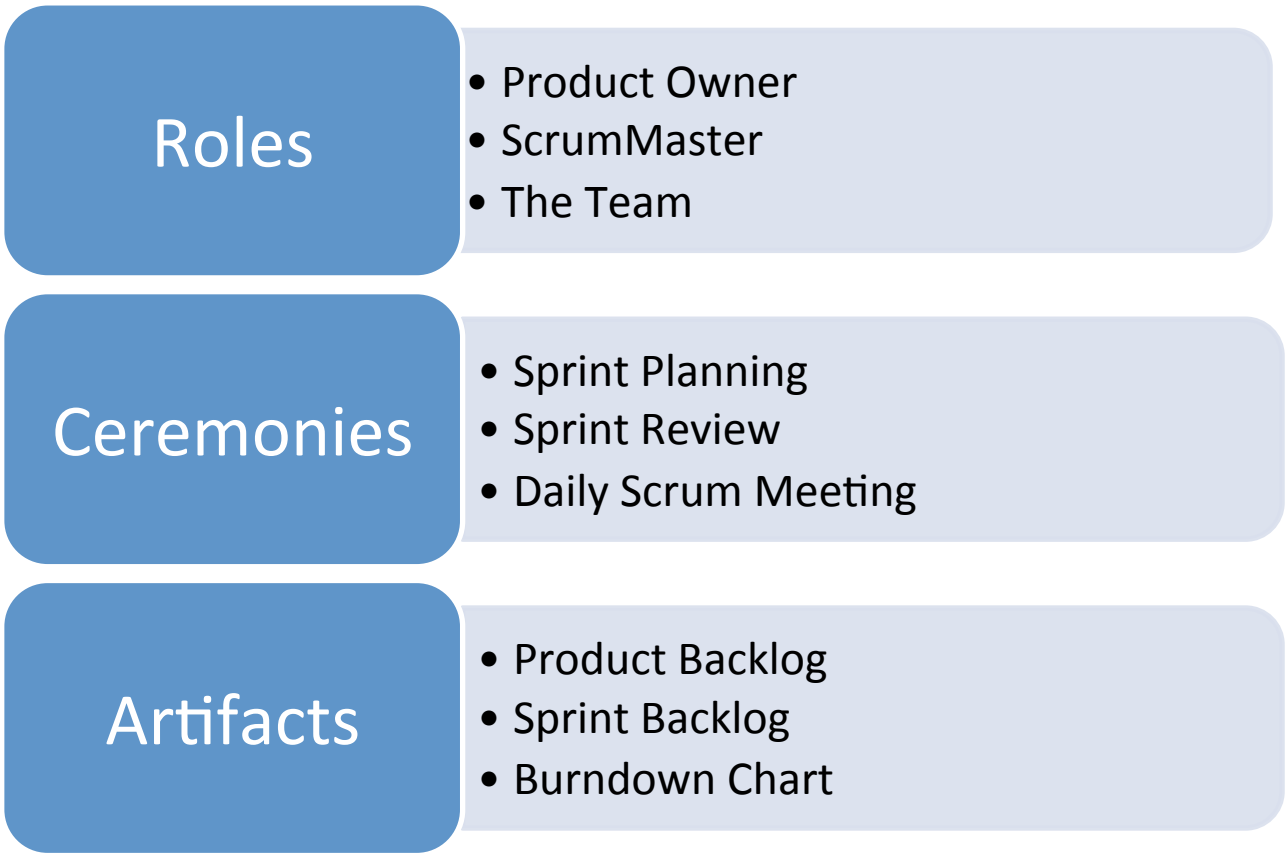
- Requirements are changing frequently
- Agile goal is rapid and incremental software development

SCRUM Process Overview

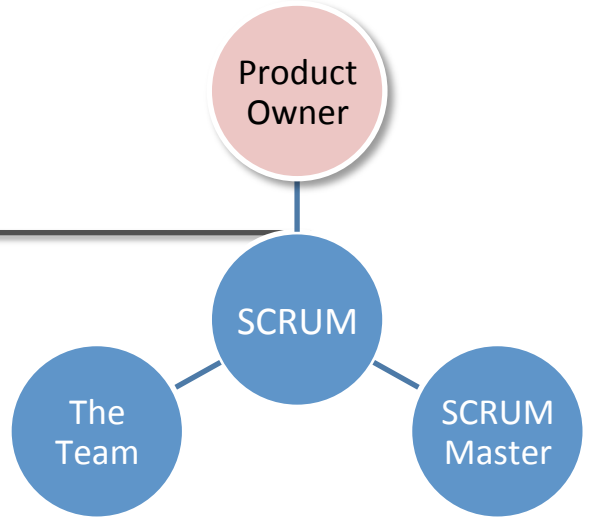


Features of SCRUM

- Scrum is a simple "inspect and adapt" framework that has **three roles**, **three ceremonies**, and **three artifacts** designed to deliver working software in Sprints, usually in iterations of 1 to 4 weeks.



Roles in SCRUM



Product Owner

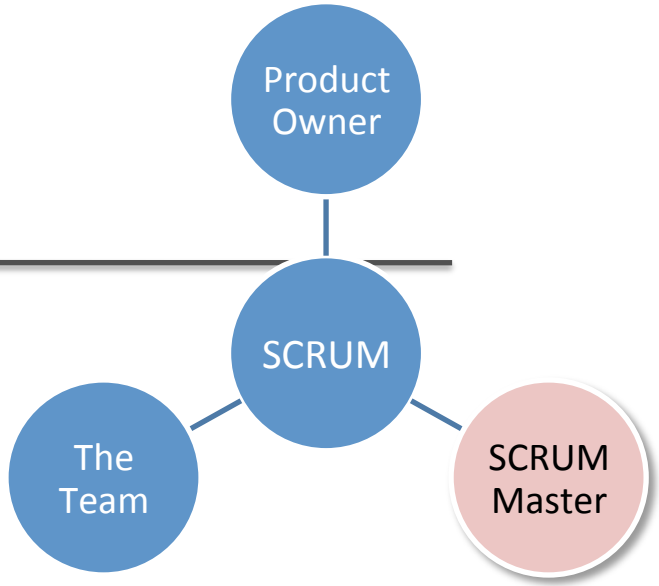
e.g., "As a **registered user** I want to be able to **search the online catalog** so that I can **find items to purchase**."

- Gathers requirements
- Defines the features, writes user stories (similar to use cases)
- Manages and prioritizes the **Product Backlog**

continuously evolving **queue of user stories** created by the Product Owner with input from other stakeholders

- Accepts the software at the end of each iteration
- Manages the Release Plan

Roles in SCRUM

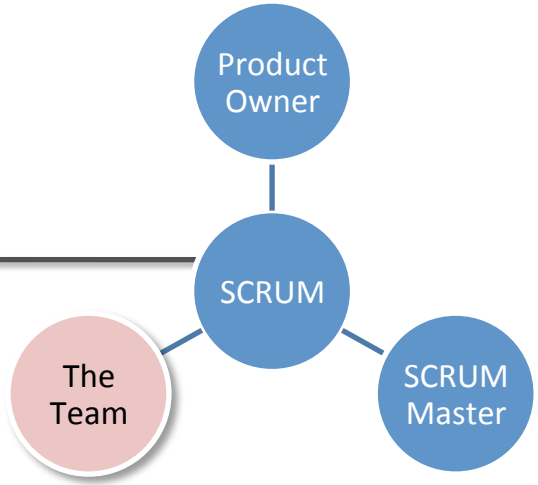


Scrum Master

- Empowers and coaches the team
- Obstacle remover
- Establishes and enforces Scrum rules and responsible

for the success of the process

Roles in SCRUM



The Team

- Self Organizing
- Consists of developers, testers, analysts, architects, writers, designers, quality control
- Optimal team size is 7 people, +/- 2
- Estimates the size of **Sprint Backlog**
- Execute tasks and delivers software incrementally
- Tracks own progress
- Accountable to the Product Owner for delivering as promised

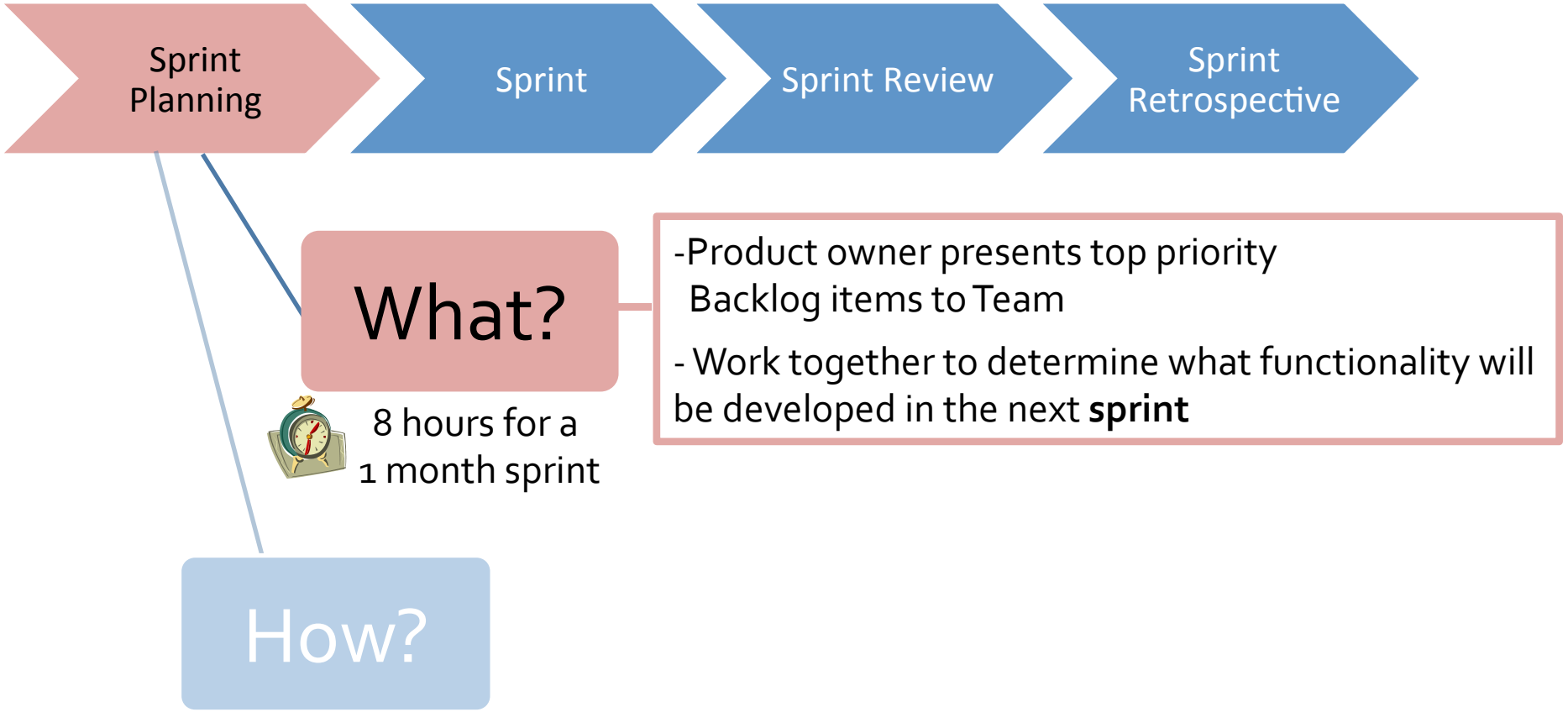
The **list of tasks** required to get the agreed Stories done

What's the process?

- A sprint is considered the "heartbeat" of the Scrum cycle

- Time-Boxing is used to control the duration of each step and must be adhered to

Sprint Planning (1 of 2)



Sprint Planning (2 of 2)

What?

How?

-Team estimates and commits to what it can get "done"

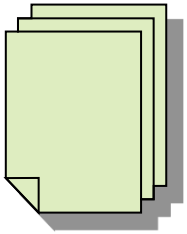
-Clarify / negotiate with Product Owner

-Team decomposes **the sprint backlog** items into tasks

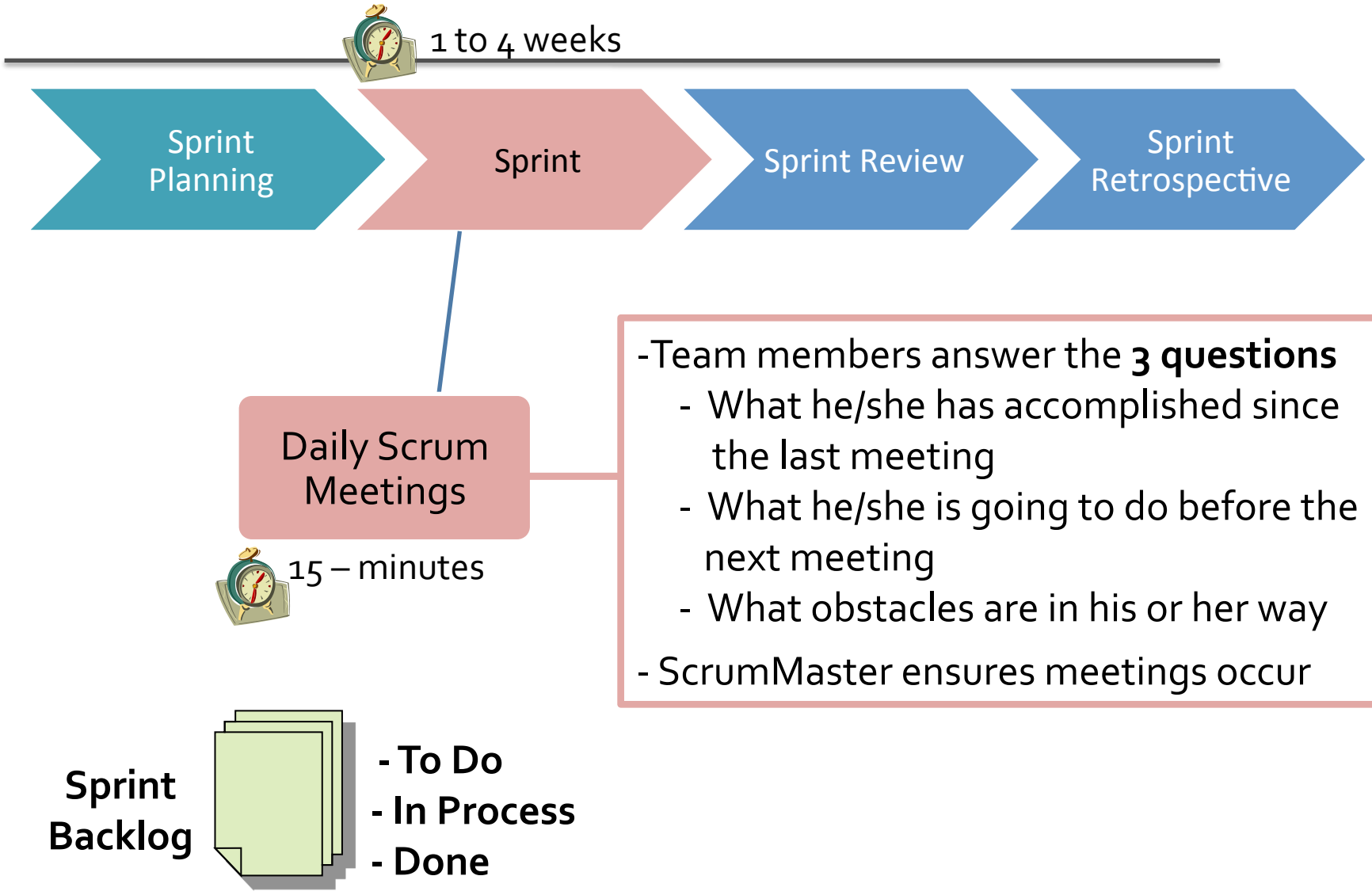
-The result is an agreed **Sprint Backlog**

4 hours for a
1 month sprint

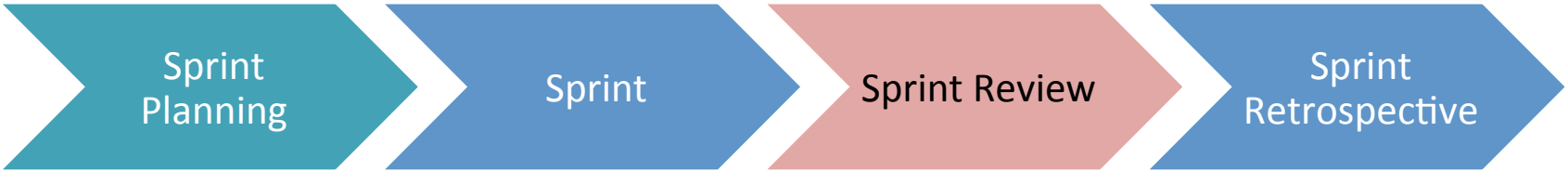
Output: **Sprint Backlog**



The Sprint – Getting It Done



Sprint Review – What Was Completed?



Met Sprint Goal?

- Demo of everything that's been done in the Sprint
-Product Owner signs off Sprint if tests are ok
- Team discusses issues & how to solve them

4 hours for a
1 month sprint

Sprint Retrospective – What Can We Do Better Next Time?

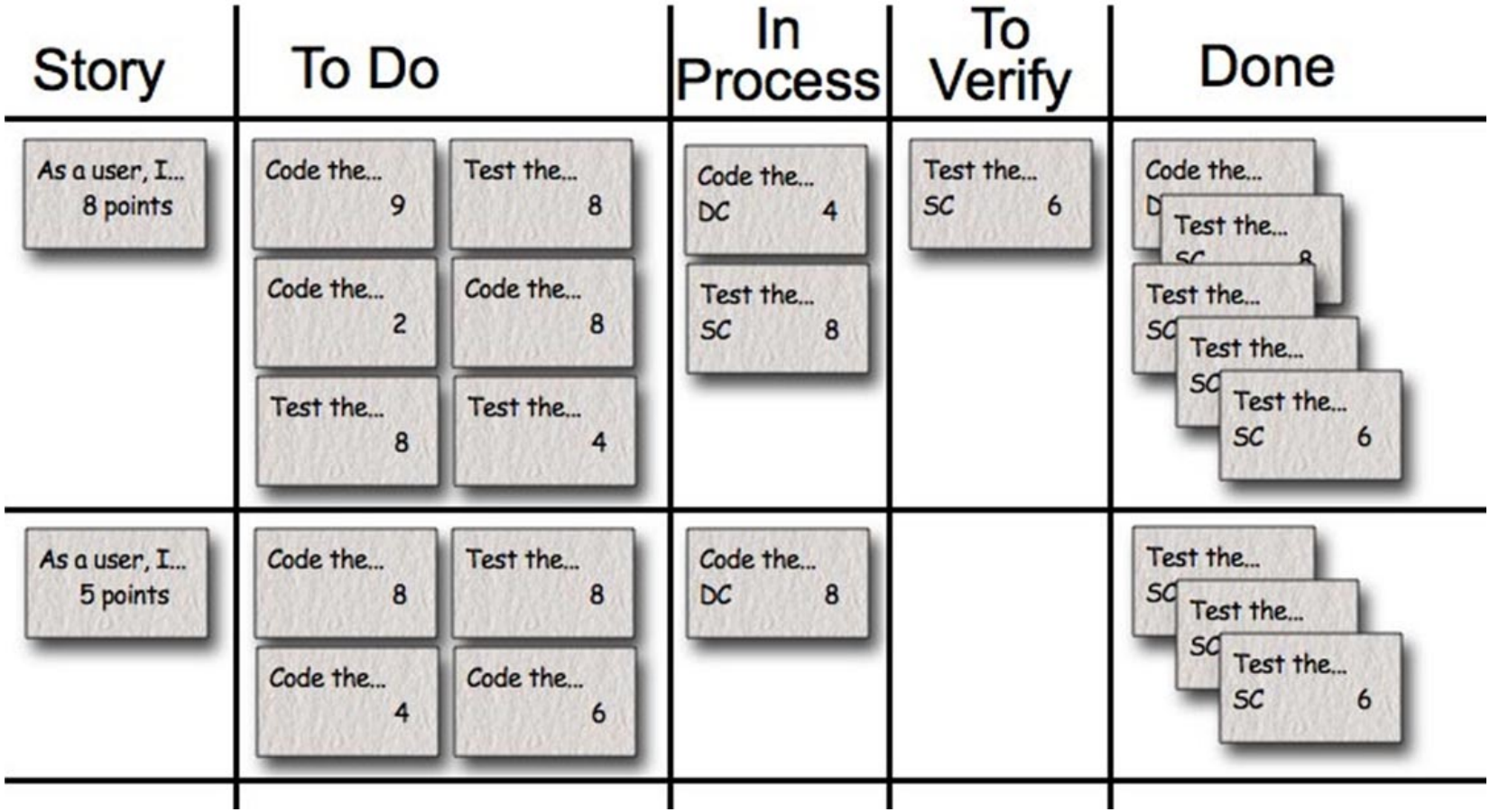
- Review lessons learned and discuss improvement actions to make things smoother for the next sprint
- How did things go with respect to
  - People
  - Relationships
  - Tools
  - Process
- Must be done before starting next sprint planning session

Refine Approach?

3 hours for a 1 month sprint

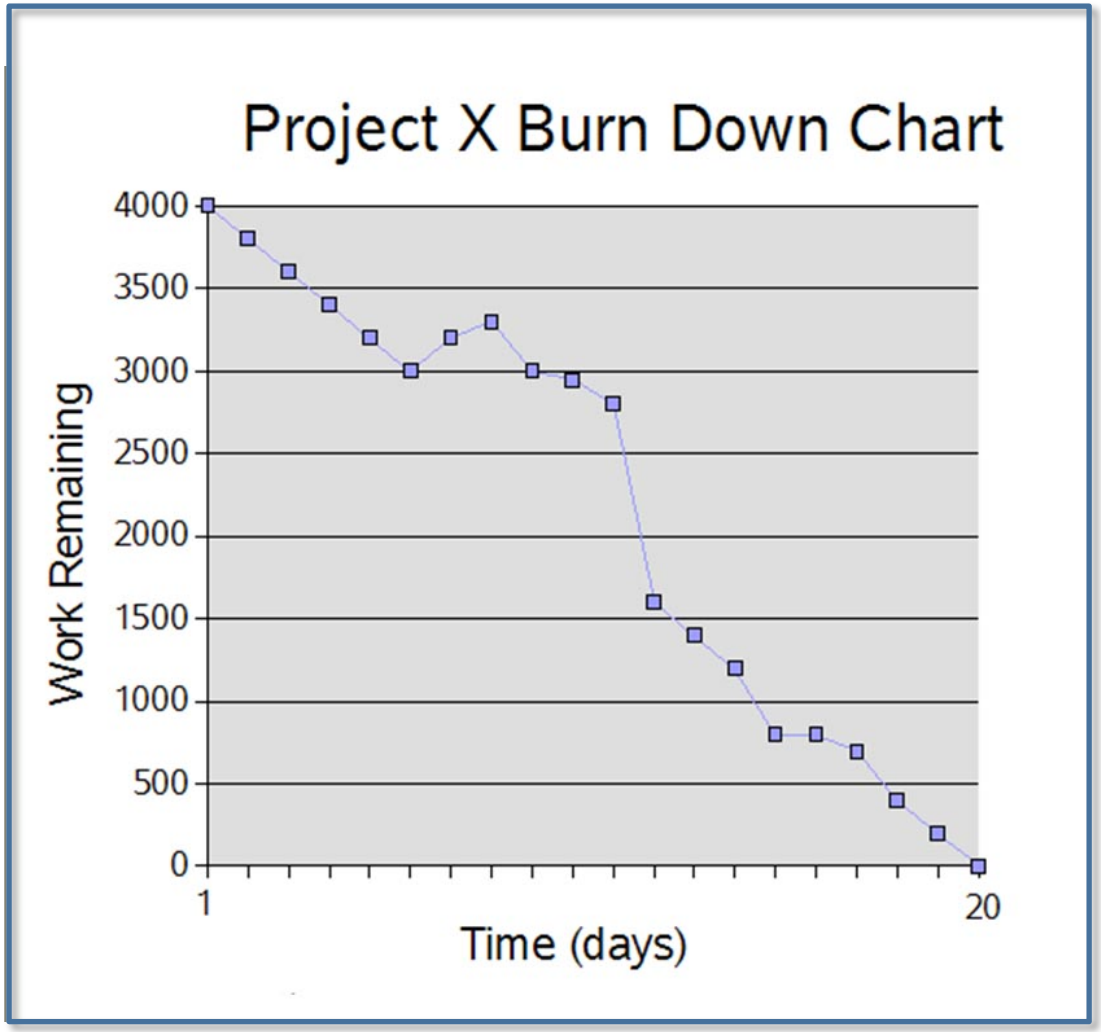
How Are We Doing?

✧ Sprint Backlog Example



How Are We Doing? - Velocity

Shows estimated effort remaining



Benefits of Agile Approach

- ✧ Reduces risk of incorrect user requirements
  - Good where requirements are changing/uncommitted
- ✧ Regular visible progress
- ✧ Catch problems early when you have time to react
- Improved Return on Investment (ROI) through early deployment of software
- Build the right product through incremental improvement

Disadvantages

- Requires extensive customer collaboration
- Costs customers time/money
- Needs committed customers
- May be too customer specific, no broad market
- Difficult to know how long project will last
- Difficult to scale up to large projects where documentation is essential
- May not be suitable for fixed-price project

Scrum vs. Waterfall

	Scrum	Waterfall
Goal / Objective	**Rapid value**	**High predictability**
Customer	• High level of involvement • Continuous Communication and Collaboration • Fully integrated as a team member	• Infrequent team interaction • Organized externally to team as stakeholder
Success Criteria	Working, tested software	Conformation to timeline & budget
Planning	Focus on evolving short-term sprint plan & long-term release plan	Focus on holistic plan defined upfront
Requirements	• Uncertain / unknown • Subject to change • Emergent	• Well known early • Unlikely to change • Defined upfront
Process Controls	Adaptive – responsive to change	Predictive – discourages change
Documentation	Low - emphasis on product	High – emphasis on project docs
Interim Deliverables	Working, tested software	Documentation

Comparison of Life-Cycle Models

Life-Cycle Model	Strengths	Weaknesses
Waterfall model	Disciplined approach – document driven	Product may not meet client's needs
Spiral Model	Risk Driven, prototype development	Developers have to be competent in risk Analysis and risk resolution
Prototyping/ **Iterative and incremental model**	Closely models real-world software production. Underlies the unified process. Shorter delivery, quick to identify inconsistency with requirements, immediate feedback from clients	Lack of complete requirements, entire system scope is not visible
RUP	Comprehensive process, software tool supported	Expensive and time consuming

References

- R. Pressman: Software Engineering- A practitioner's approach
- B. Boehm: Software Engineering Economics
- Barry Boehm, "A Spiral Model of Software Development and Enhancement". In: *ACM SIGSOFT Software Engineering Notes* (ACM) 11(4):14-24, August 1986