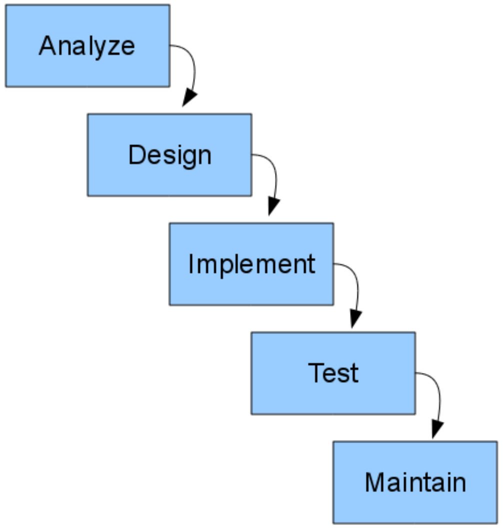
CMPS 411

# Software Process Model



Please read Chapter 2 and 3

**Dr. Abdelkarim Erradi**

Dept. of Computer Science & Engineering

**QU**

# Outline

- Definition of Software Process Model
- Waterfall model
- Spiral Model
- V Model
- Rational Unified Process (RUP)
- Evolutionary Models: Prototyping
- Agile Approaches e.g., Scrum
- Important factors to consider when selecting a SE Process Model

## Software Process Models

- A software process model is a description of the sequence of activities carried out for

- **planning**

- **organizing**, and

- **running** a development project

- In this lecture we:
  - **survey** the important basic models, and
  - consider **how to choose** between them

## Important Process Models

- There are hundreds of different process models to choose from, e.g.:
  - **waterfall,**
  - **spiral**
  - **rapid prototyping**
  - **unified process (UP)**
  - **agile methods such as Scrum …**
- But most are minor variations on a small number of basic models.

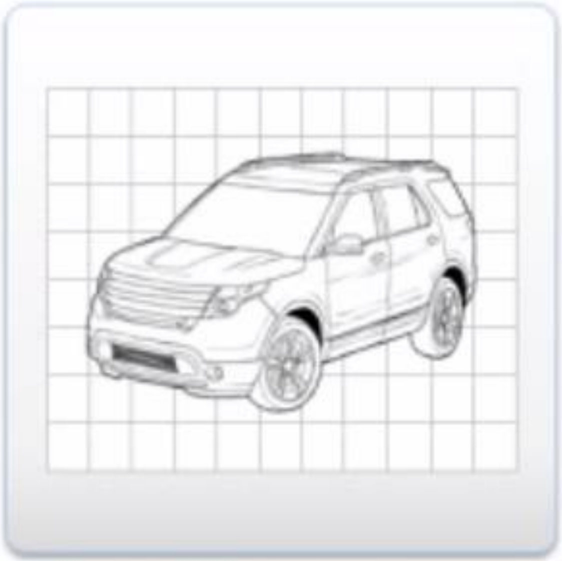
## Waterfall Model

## Waterfall Model

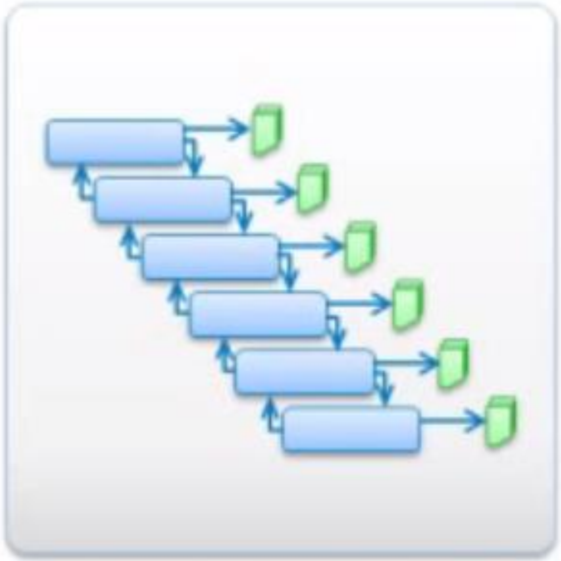
- Each stage signed off before the next one commences
- Need **extensive documentation** as this is the primary communication medium

- Good approach if requirements are fully known and understood, not complex and unlikely to change
- Good for fixed-price contracts

## Scenario for using Waterfall model


Complete Requirements

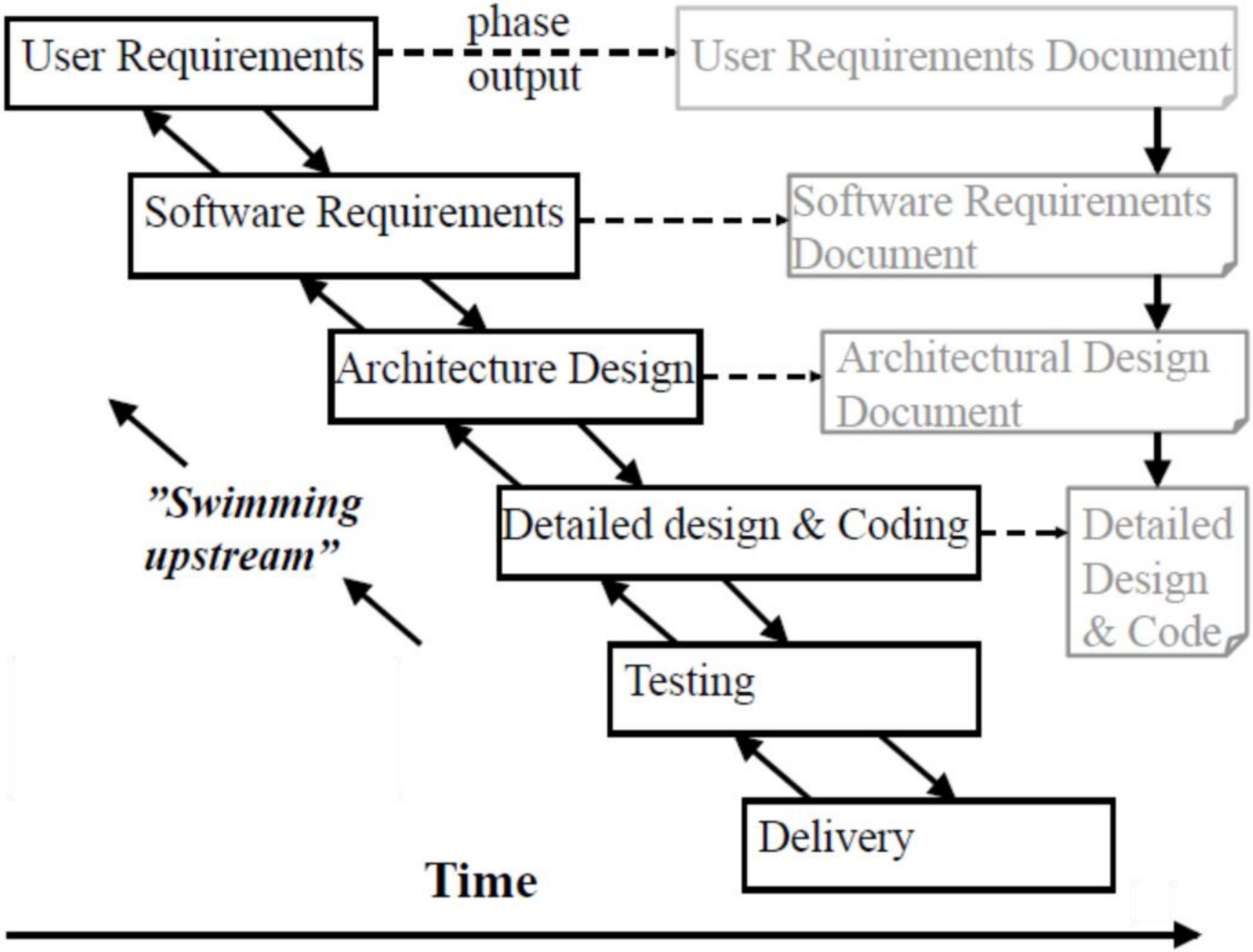
Frozen Requirements


Deliverables at Each Stage

+ Organization have executed similar projects earlier (hence lower risk)

+ Organization desires a fixed-price project contract

# Waterfall Workflow



## 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.

## Disadvantages

- Doesn’t reflect iterative nature of exploratory development.
- Sometimes unrealistic to expect accurate requirements early in a project
- Software is delivered late, often with delays and over-budget due to discovery of serious errors.
- Difficult and expensive to change decisions “swimming upstream”.
- Significant administrative overhead, costly for small teams and projects.

## Spiral Model

## Scenario for using Spiral Model

Many risks expected at each stage of development

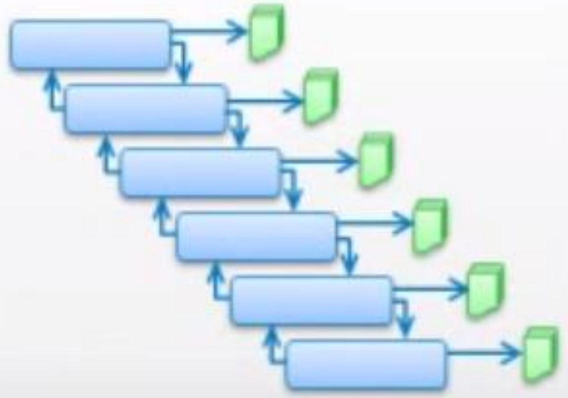
Alternatives available

Customer is in agreement that the project is not fixed budget

# Spiral Model

Prototype Model

Waterfall Model

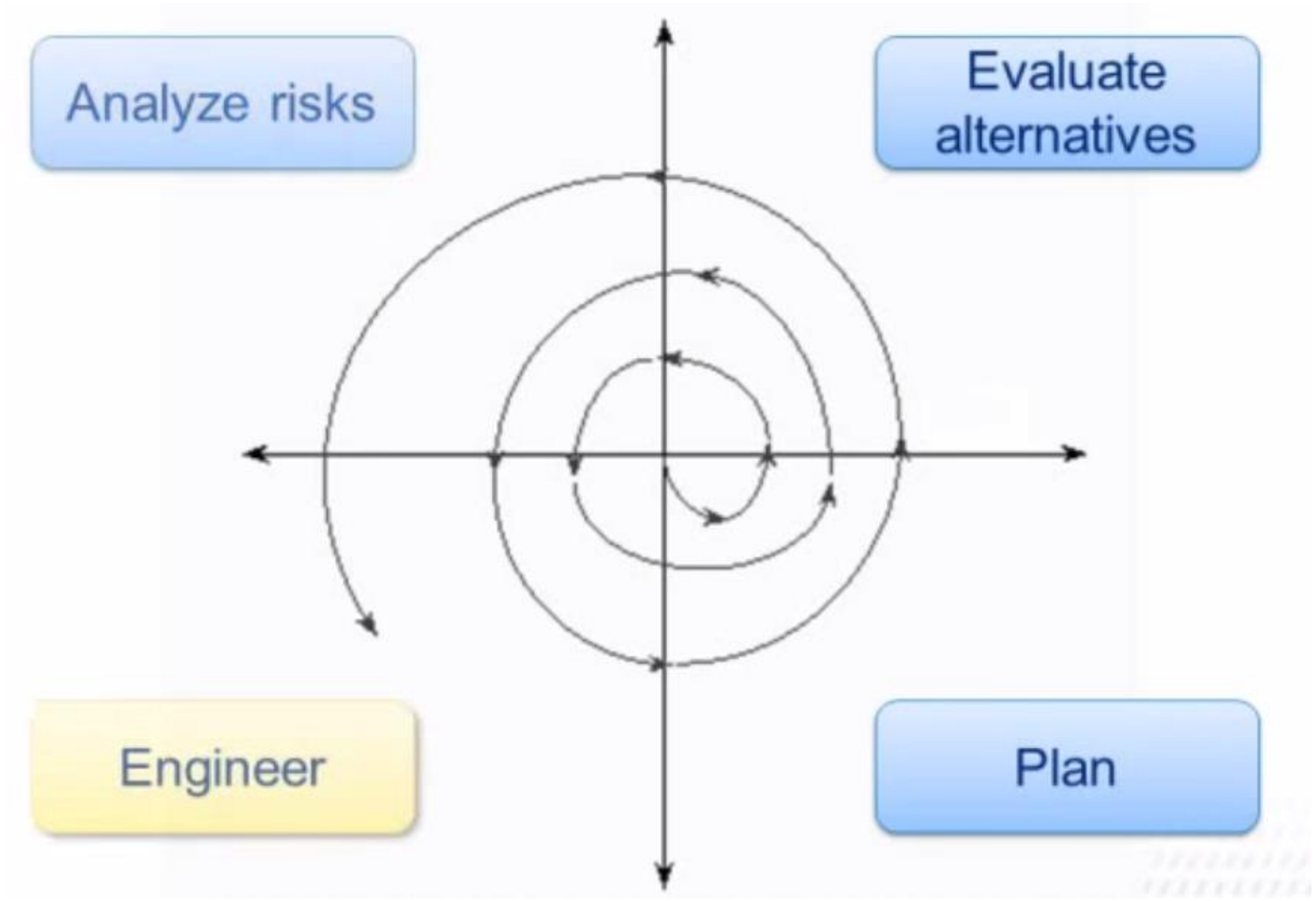


Iterative nature

Controlled and systematic nature

The spiral model couples the iterative nature of the prototyping model and the controlled, systematic nature of the waterfall model

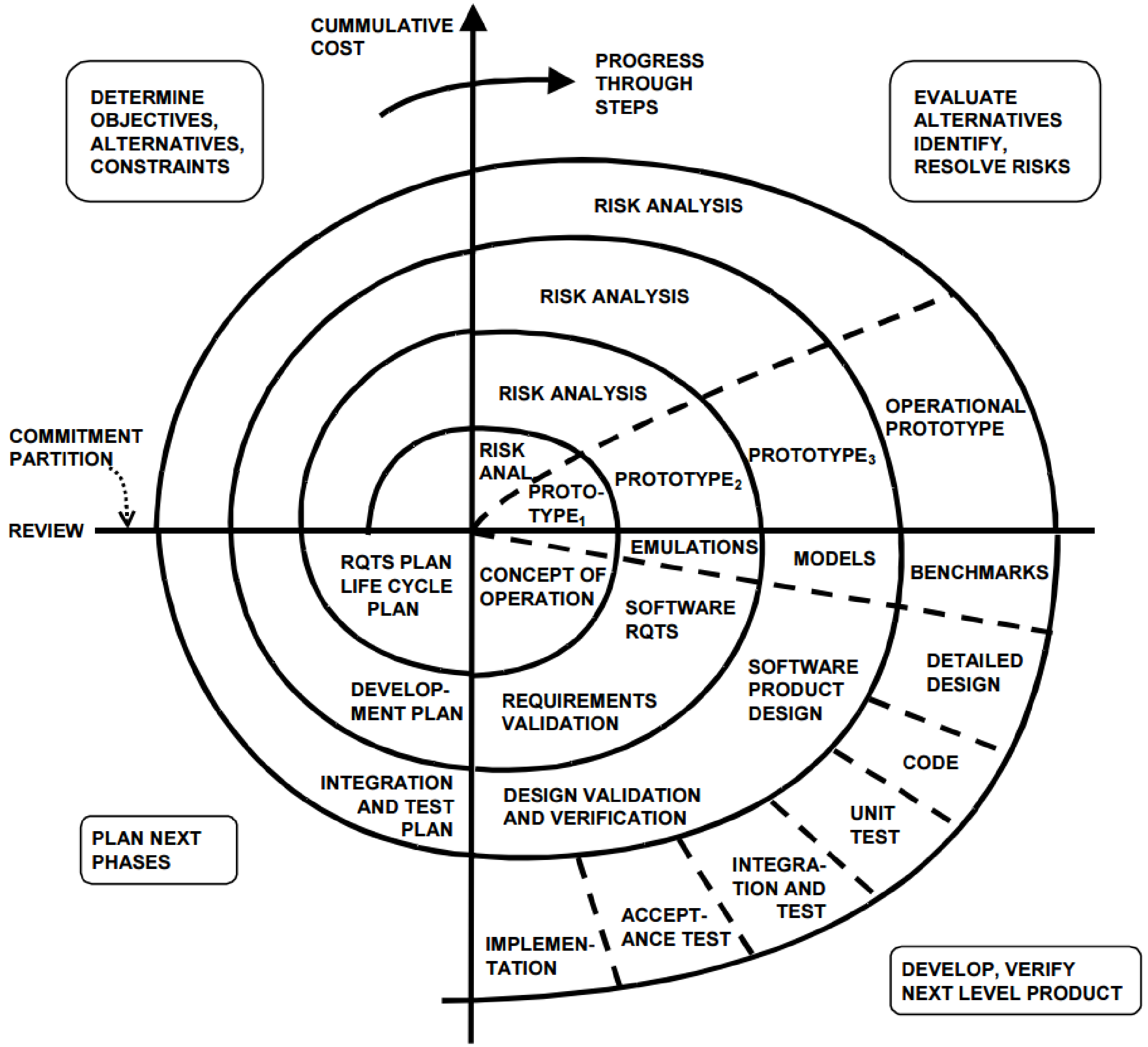
## Spiral Model Phases



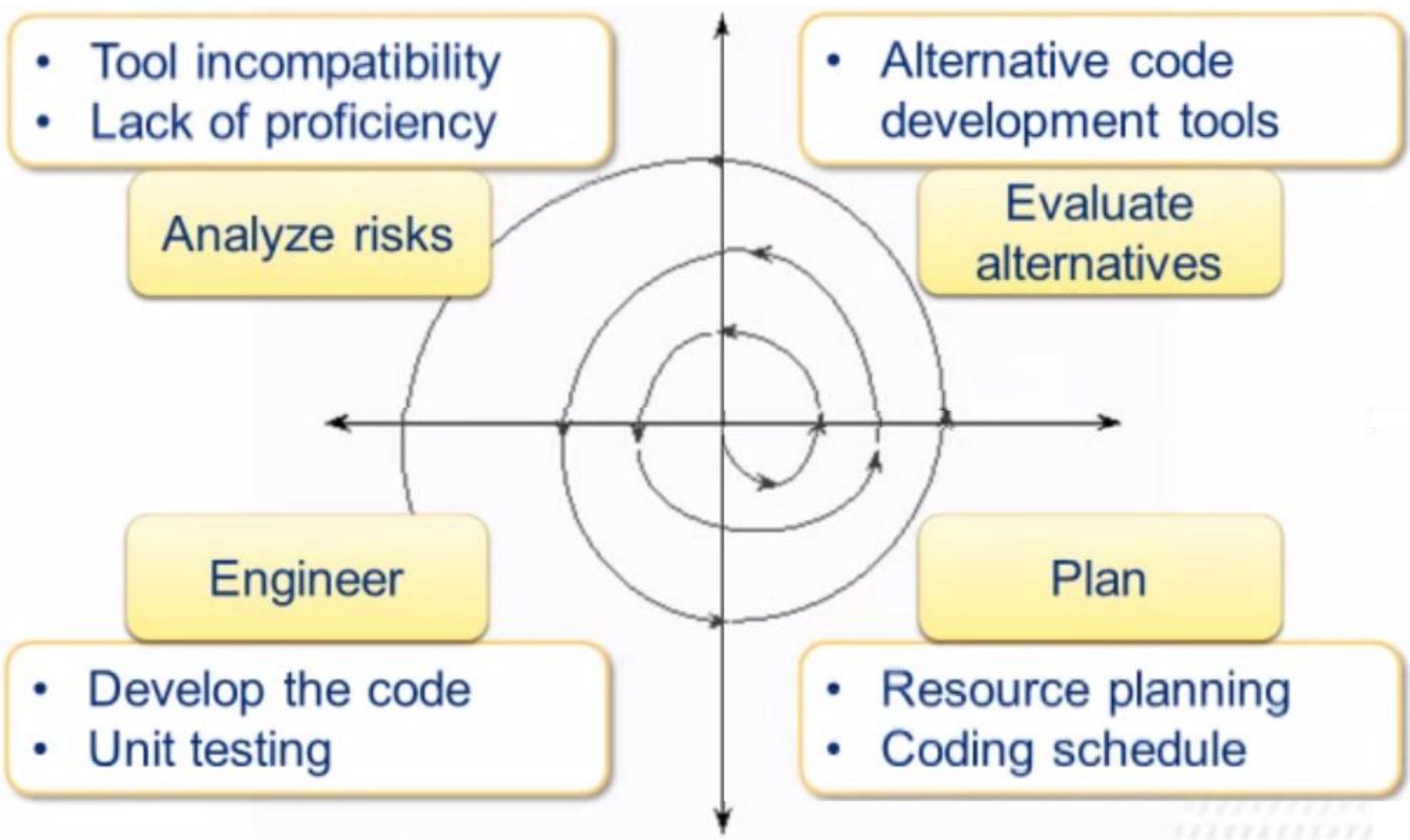
# Spiral Model Phases

- 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

# Illustration @ http://courses.cs.vt.edu/~csonline/SE/Lessons/Spiral/index.html



## Spiral Model applied to Coding Phase



## Advantages

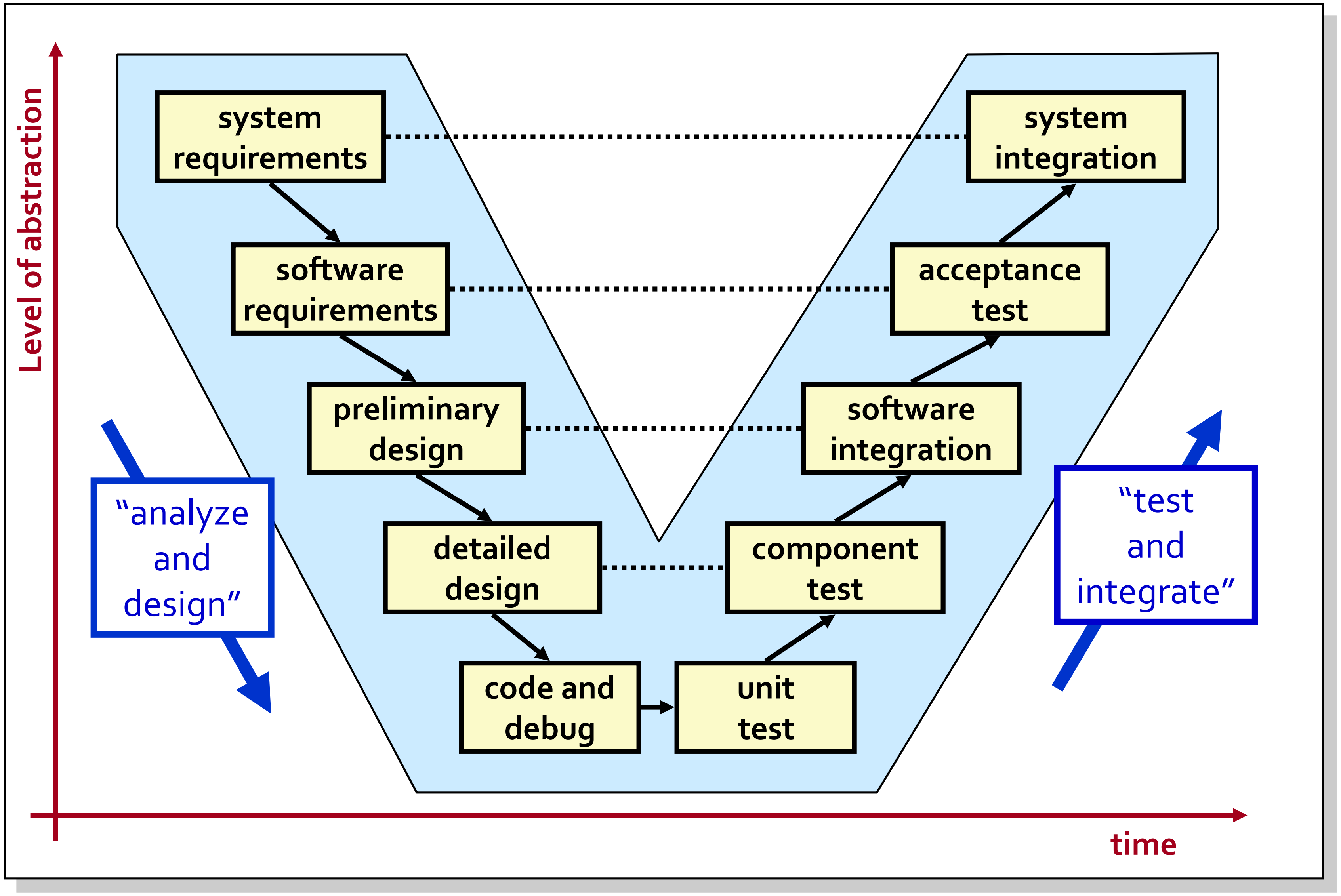
- Realism: the model accurately reflects the iterative nature of software development on projects with unclear/complex requirements
- Flexible: incorporates the advantages of the waterfall and evolutionary methods
- Comprehensive model decreases risk
- Good project visibility

## 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

## V Model

# V Model


Level of abstraction
system requirements
software requirements
preliminary design
detailed design
code and debug
unit test
component test
software integration
acceptance test
system integration
"analyze and design"
"test and integrate"
time

## V-Model

- V- model means Verification and Validation model.
- Just like the water fall model, the V-Shaped life cycle is a sequential path of execution of processes
- Each phase must be completed before the next phase begins
- Testing of the product is planned in parallel with a corresponding phase of development

## Advantages of V-model:

- Simple and easy to use.
- Testing activities like planning test designing happens well before coding
  - This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

## Disadvantages of V Model

- The biggest disadvantage of V-model is that it's very rigid. If any changes happen mid way, not only the requirements documents but also the test documentation needs to be updated.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.

## Rational Unified Process (RUP)

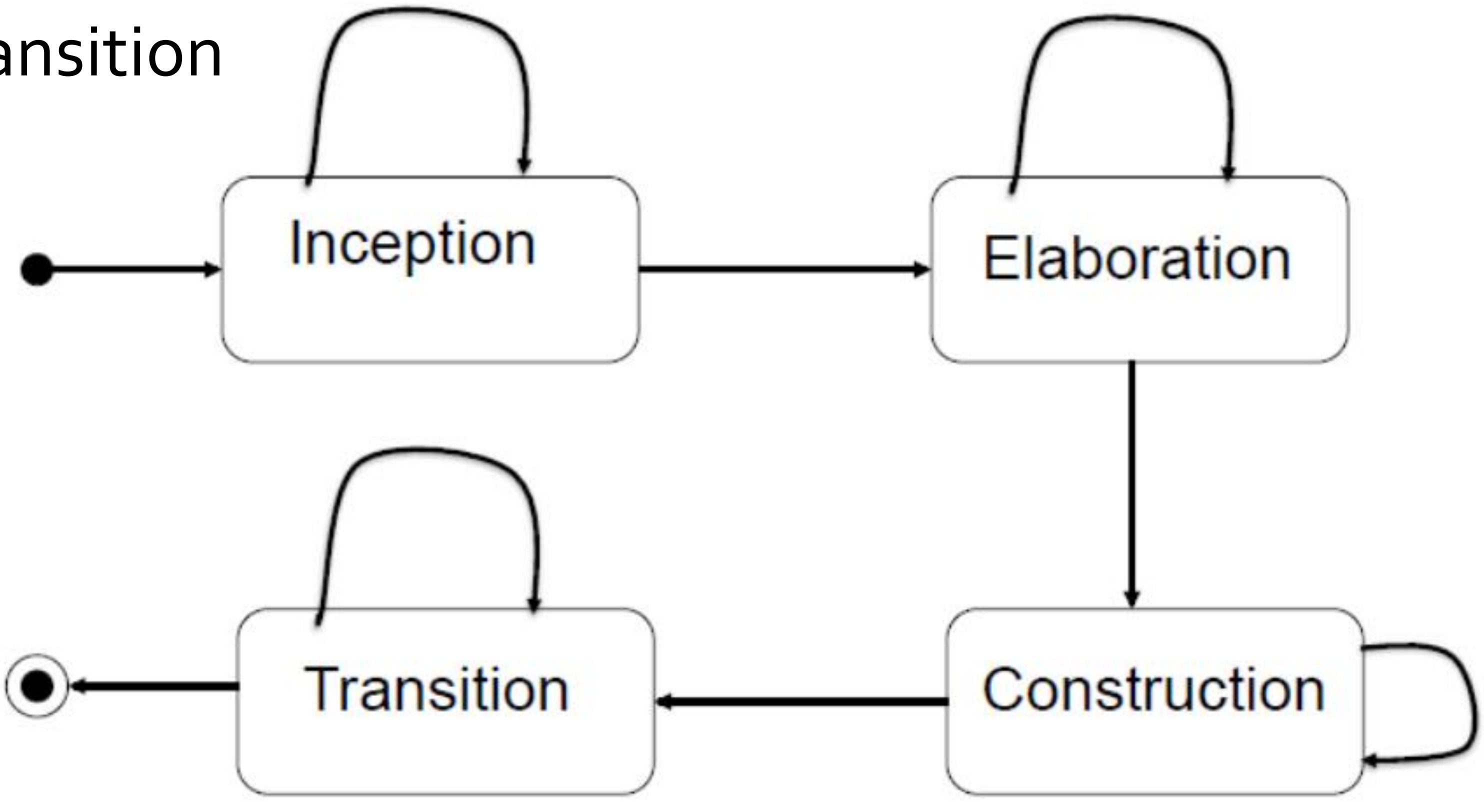
# Rational Unified Process (RUP)

- Iterative and incremental approach for object-oriented systems
  - Big project is split in many mini-projects called **iterations**
  - Each iteration **increments** the overall project result
- Strongly embraces UML for modeling
- **Use-case driven**
- Architecture-oriented

## RUP Phases

All iterations are organized into 4 phases:

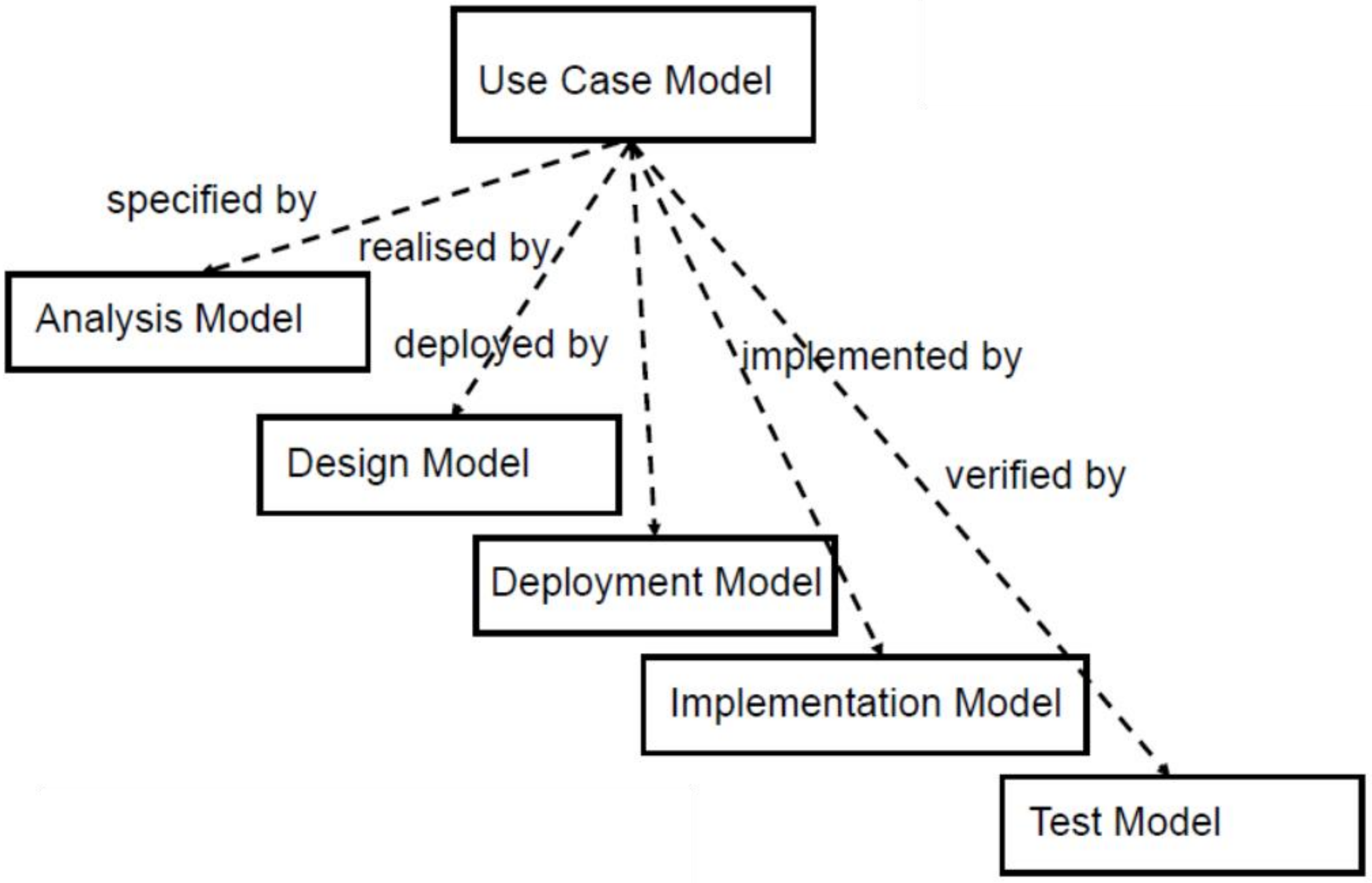
- Inception
- Elaboration
- Construction
- Transition



# RUP Phases

- **Inception** – Establish high-level project requirements.
- **Elaboration** – Detailed requirements. Establish software architecture and consider design tradeoffs. Identify project risks. Estimate and schedule project. Decide on build vs. buy.
- **Construction** – Design, implement and test each component. Integrate components to deliver fully functional software
- **Transition** – release a mature version and deploy in real world

# UML Model-based Development



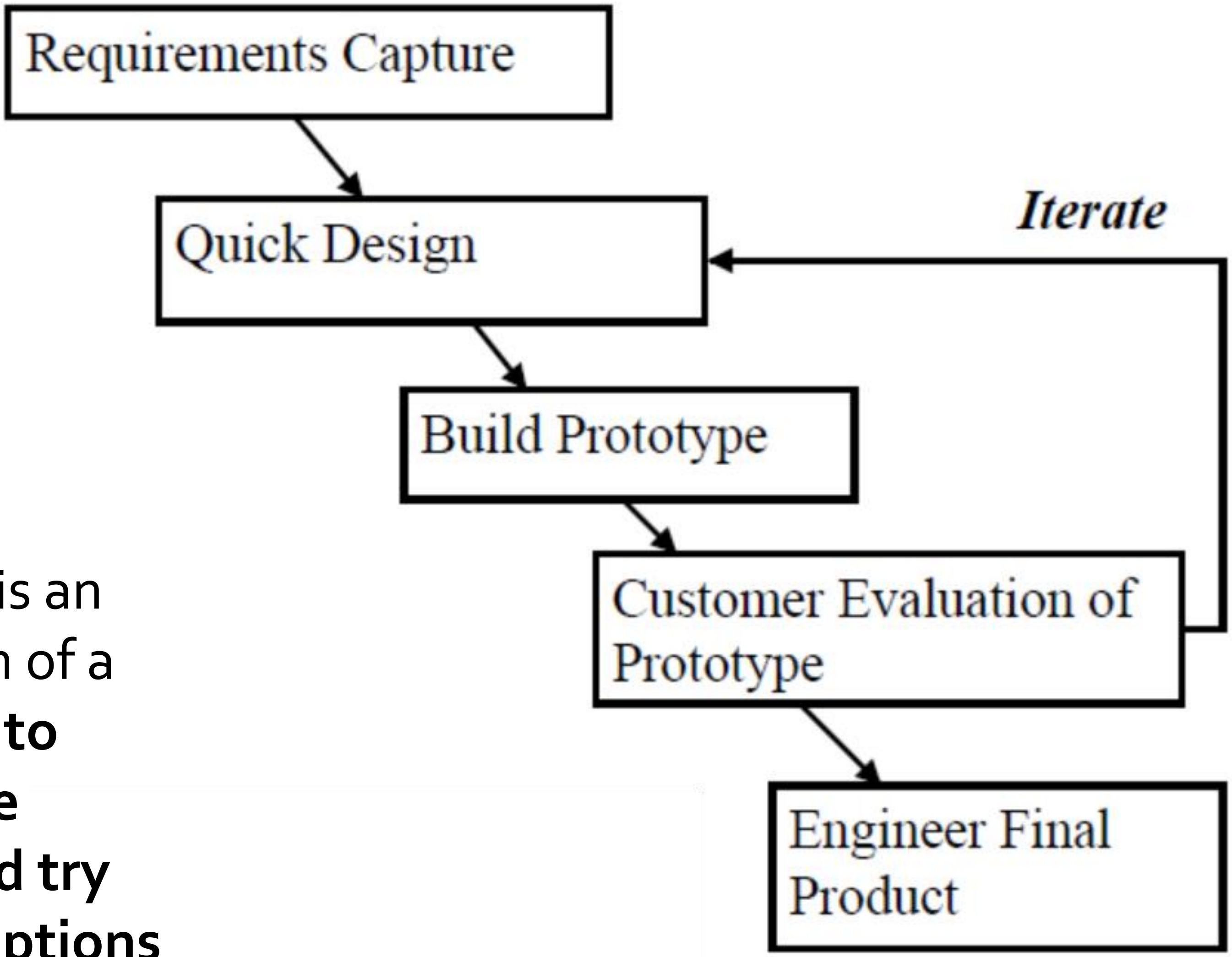
## Advantages of RUP

- Architecture brings us overall understanding of the big and complex system
- Early risk handling
- Easier requirements change management
- High level of reuse
- Project teams learn easier because the project is broken down into small manageable parts
- Better quality assurance
- Extensive UML modeling tools

## Disadvantages of RUP

- Very broad and complex
- You have to customize it to start really using it
- Demands big initial efforts and investments

# Evolutionary Models: Prototyping



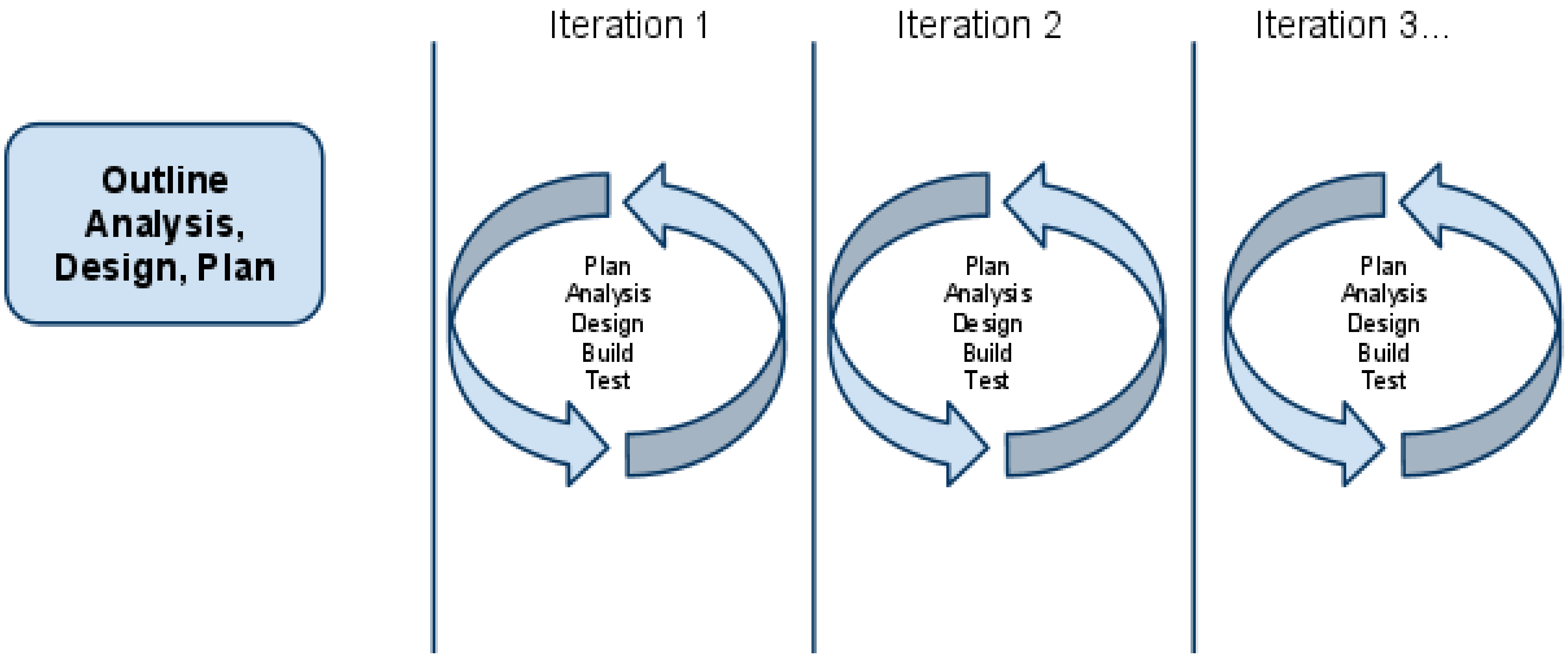
A prototype is an initial version of a system used **to demonstrate concepts and try out design options**

# 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. 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. |

## Agile Approaches
## e.g., Scrum

# Agile Approach


Outline Analysis, Design, Plan
Iteration 1
Plan Analysis Design Build Test
Iteration 2
Plan Analysis Design Build Test
Iteration 3…
Plan Analysis Design Build Test

- Working solution in every iteration
  - Review and refine regularly

## 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/](http://agilemanifesto.org/)

## Waterfall vs. Agile

- Waterfall: emphasize **Structure**

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

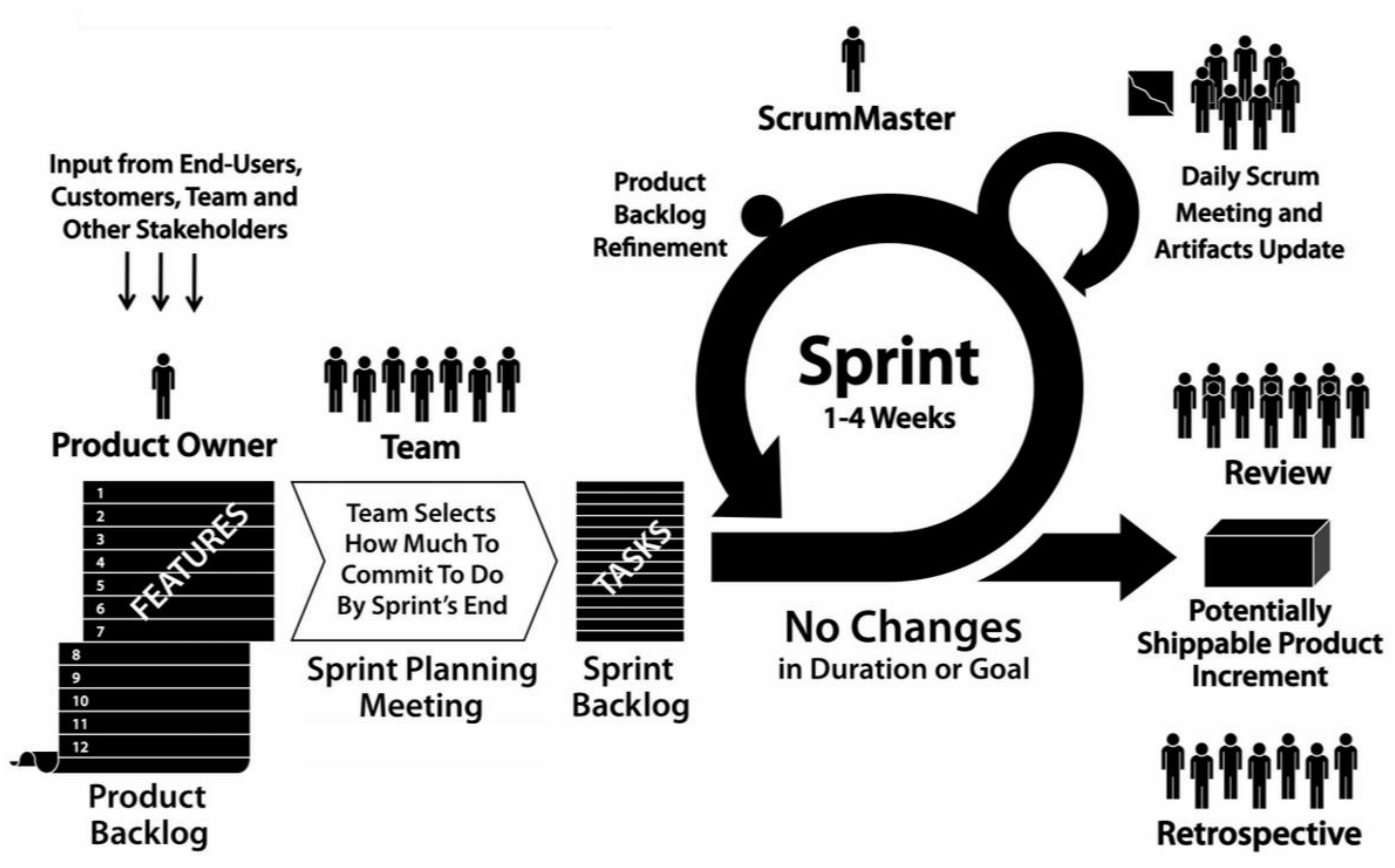
- Agile: emphasize on **Adaptability** (change responsiveness)
  - Requirements are changing frequently
  - Agile goal is rapid and incremental software development

# How SCRUM Can Help?

## Focus on Value Delivery and Adaptability

- Scrum is an **Agile** process
- **Iterative** process
- Rapidly and repeatedly **inspect and adapt**
- Progress measured in the form of **working software**
- See **real progress** every 1-4 weeks
- **Actively** pursue opportunities to **improve**

# SCRUM Process Overview



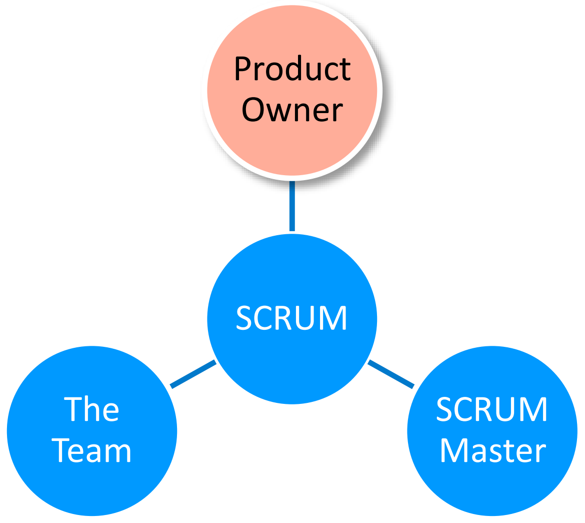
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
1
2
3
4
5
6
7
8
9
10
11
12
FEATURES
Product Backlog
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting
TASKS
Sprint Backlog
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

# 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 | • Product Owner<br>• ScrumMaster<br>• The Team |
| Ceremonies | • Sprint Planning<br>• Sprint Review<br>• Daily Scrum Meeting |
| Artifacts | • Product Backlog<br>• Sprint Backlog<br>• Burndown Chart |

# Roles in SCRUM



## Product Owner

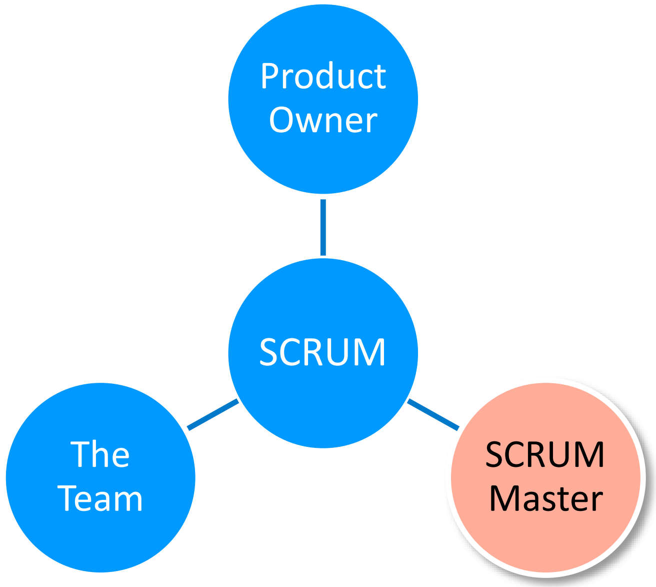
e.g., "As a **registered user** I want to be able to **search the online catalog** so that I can **find items to purchase**."

- 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

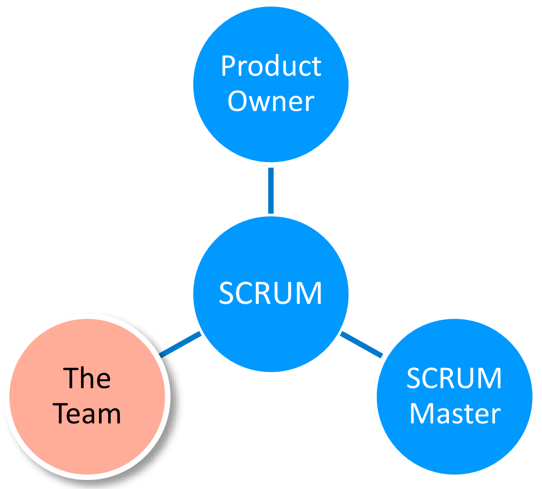


## ScrumMaster

- 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

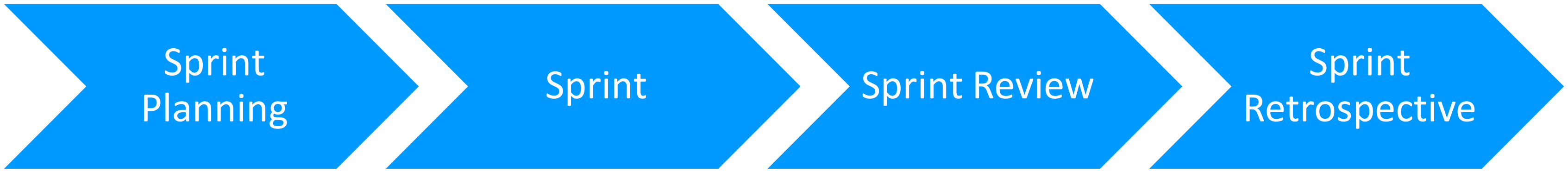
- Self Organizing
- Consists of developers, testers, analysts, architects, writers, designers, quality control, etc.
- Optimal team size is 7 people, +/- 2
- Estimates the size of **Sprint Backlog**

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

- Execute tasks and delivers software incrementally
- Tracks own progress
- Accountable to the Product Owner for delivering as promised

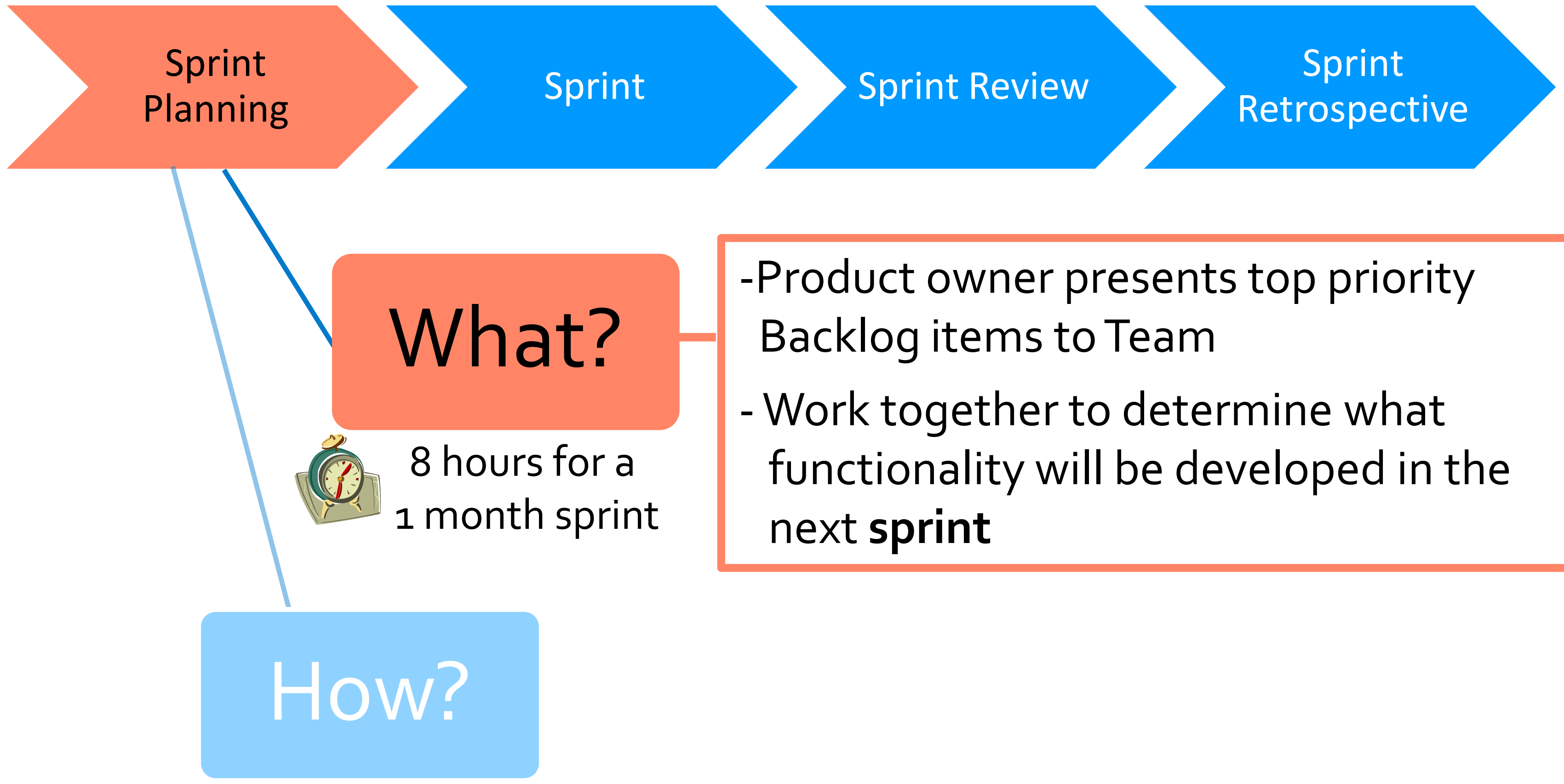
# What's the process?

- 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)

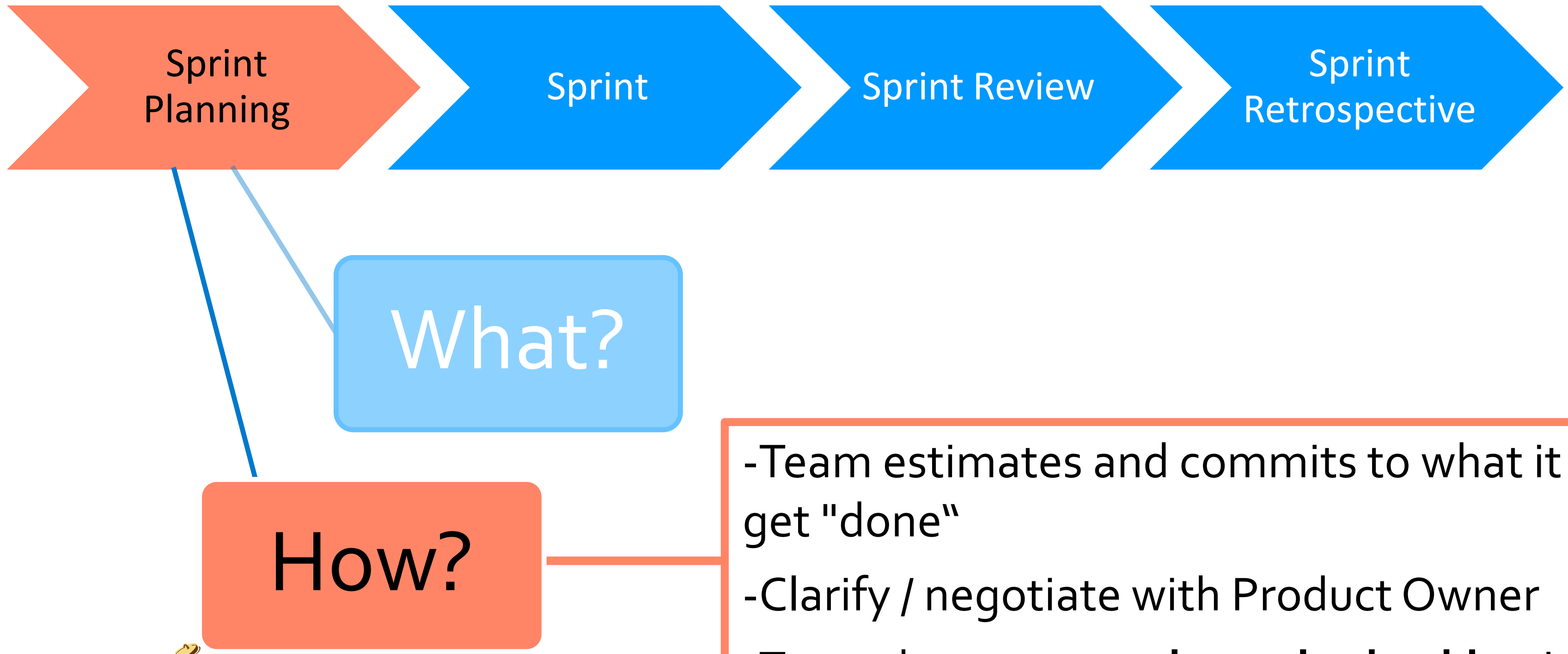


8 hours for a 1 month sprint

-Product owner presents top priority Backlog items to Team

- Work together to determine what functionality will be developed in the next **sprint**

# Sprint Planning (2 of 2)



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

-Clarify / negotiate with Product Owner

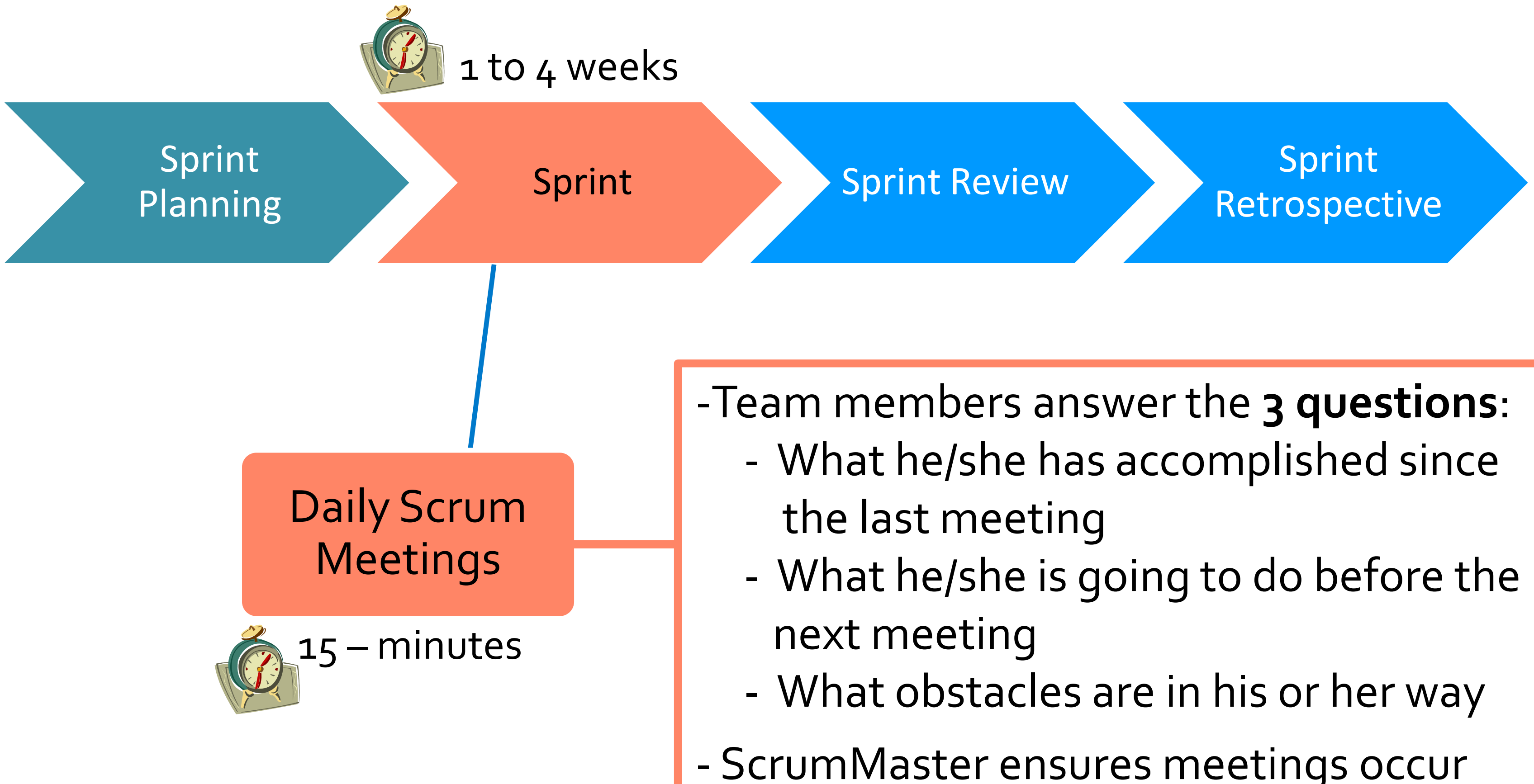
-Team decomposes **the sprint backlog** items into tasks

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

4 hours for a 1 month sprint

Output: **Sprint Backlog**

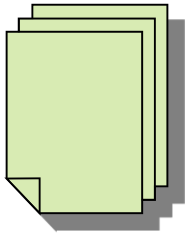
# The Sprint – Getting It Done



-Team members answer the **3 questions**:
- What he/she has accomplished since the last meeting
- What he/she is going to do before the the next meeting
- What obstacles are in his or her way

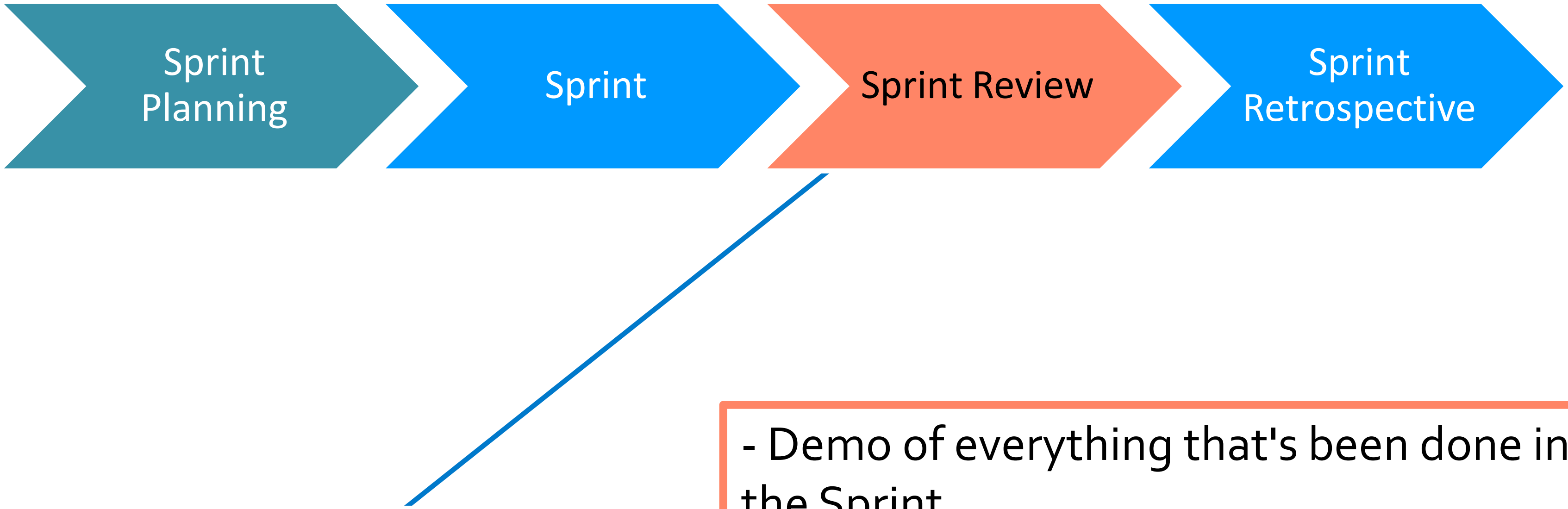
- ScrumMaster ensures meetings occur

**Sprint Backlog**



**- To Do**
**- In Process**
**- Done**

## Sprint Review – What Was Completed?

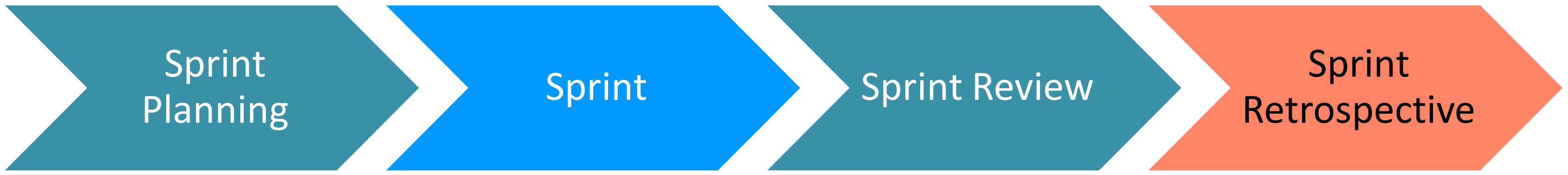


Met Sprint Goal?

4 hours for a 1 month sprint

- 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

# 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

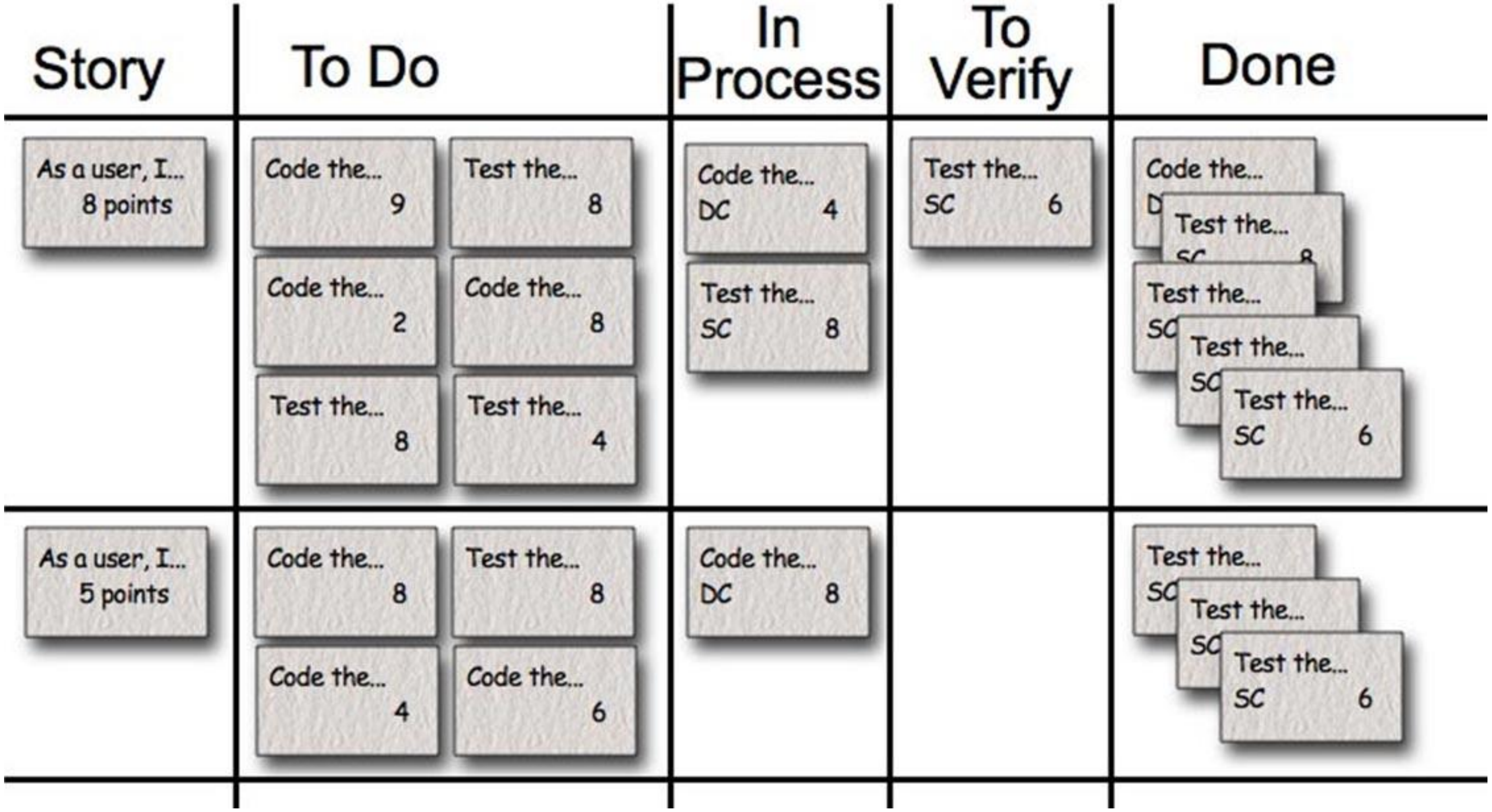
- Must be done before starting next sprint planning session

**Refine Approach?**

3 hours for a 1 month sprint

## How Are We Doing?

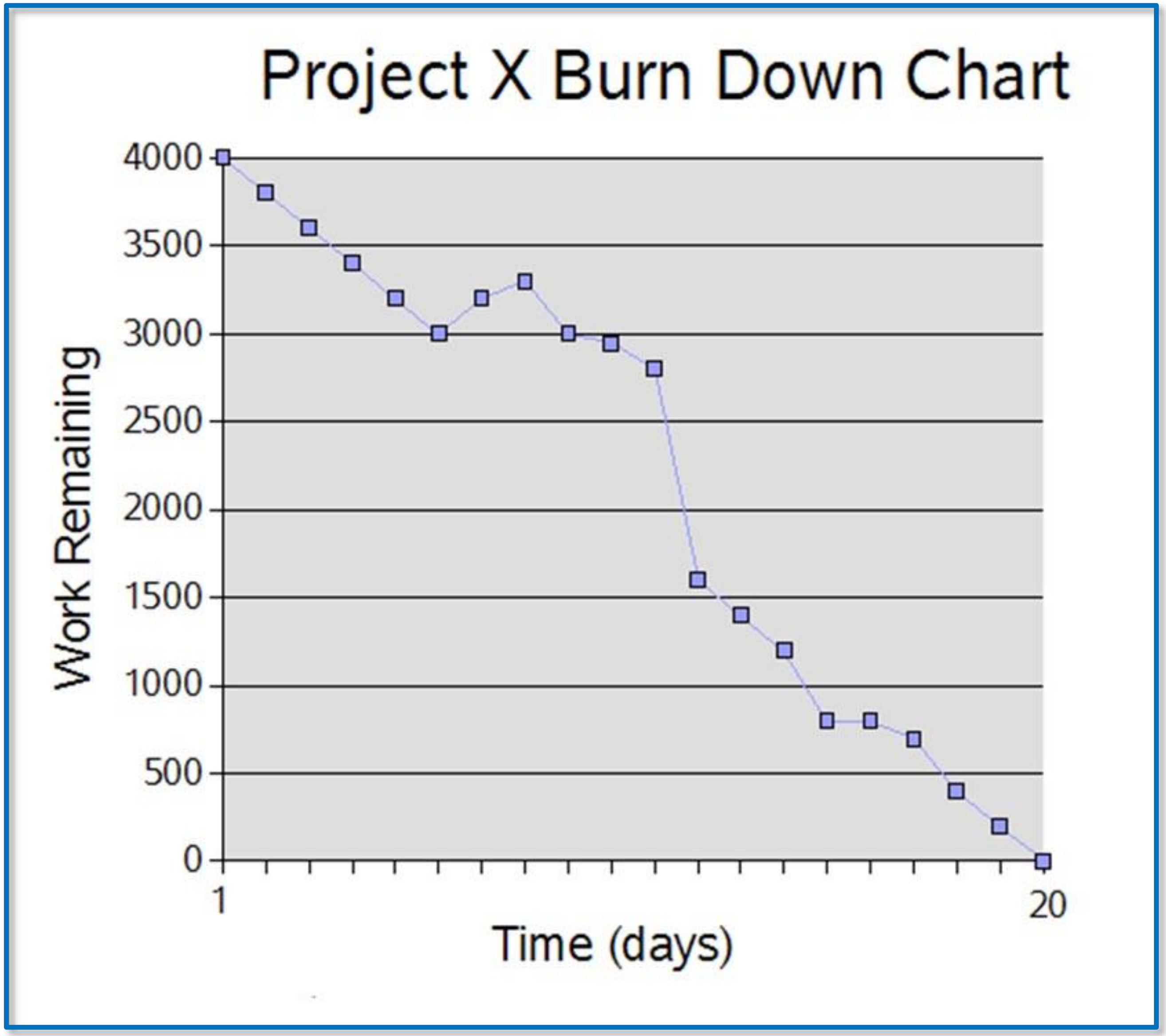
- **Sprint Backlog Example**



| Story | To Do | In Process | To Verify | Done |
|---|---|---|---|---|
| As a user, I... 8 points | Code the... 9<br>Test the... 8<br>Code the... 2<br>Code the... 8<br>Test the... 8<br>Test the... 4 | Code the... DC 4<br>Test the... SC 8 | Test the... SC 6 | Code the... D<br>Test the... SC 8<br>Test the... SC<br>Test the... SC<br>Test the... SC 6 |
| As a user, I... 5 points | Code the... 8<br>Test the... 8<br>Code the... 4<br>Code the... 6 | Code the... DC 8 | | Test the... SC<br>Test the... SC<br>Test the... SC 6 |

# 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<br>• Continuous Communication and Collaboration<br>• Fully integrated as a team member | • Infrequent team interaction<br>• 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<br>• Subject to change<br>• Emergent | • Well known early<br>• Unlikely to change<br>• 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 |

## Important factors to consider when selecting a SE Process Model

## Characteristics of Heavyweight Methodologies

- **Process Oriented -** there is a well defined process supported by tools.
- **Predictive approach** – first plan out a large part of the software process in great detail for a long span of time.

  + lock down requirements early
- **Comprehensive Documentation –** gather all of a customer’s requirements, big design upfront process prior to writing any code.

## Characteristics of Agile Methodologies

- **People Oriented**- consider people as the most important factor of software methodologies.
- **Adaptive** – not afraid of change rather determining how to better handle changes that occur throughout a project
- **Balancing Flexibility and Planning** –detailed plans for the next few weeks, rough plans for the next few months
- **Collaboration –** involve customer feedback on a regular and frequent basis
- **Small Self-organizing teams –** Agile teams discuss and communicate together on all aspects of the project.

## Things to consider (1 of 2)

- **Requirements/Project scope:** is it changing or is it stable?
  => Go agile when requirements/scope are changing
- **Experience**: is this a new technology to your organization? -> if yes go agile

- Do you have experience implementing similar solution before? -> if yes you may consider waterfall

- **Resources/dedication** (fully / non-fully dedicated to the project). Agile needs dedicated team members

# Things to consider (2 of 2)

- **Resources/Physical locations** (bringing team members together / offshore team)

- Can they be co-located => agile is ok

- **Customer involvement**:

- Customer available for continuous feedback => go agile

- **Timelines**: fixed timelines we have to work towards or bit more flexible (flexible => go agile)
- **Documentation**:

- less documentation -> agile
- Compliance/regulatory requirements to meet --> need Waterfall

# Resources

- Scrum in Under 10 Minutes

https://www.youtube.com/watch?v=XUollRltyFM

- Scrum Primer (Excellent 14 pages concise
  scrum)

http://www.scrumprimer.org/scrumprimer20.pdf