## Lecture 2

# Software Engineering Processes

Course 2021/2022

# Outline

1. What are Software Processes
2. Generic Process Models
   a. Waterfall
   b. Rapid Application Development
   c. V model
   d. Prototyping
   e. Spiral
3. Agile Methodology

# 1.

# Processes

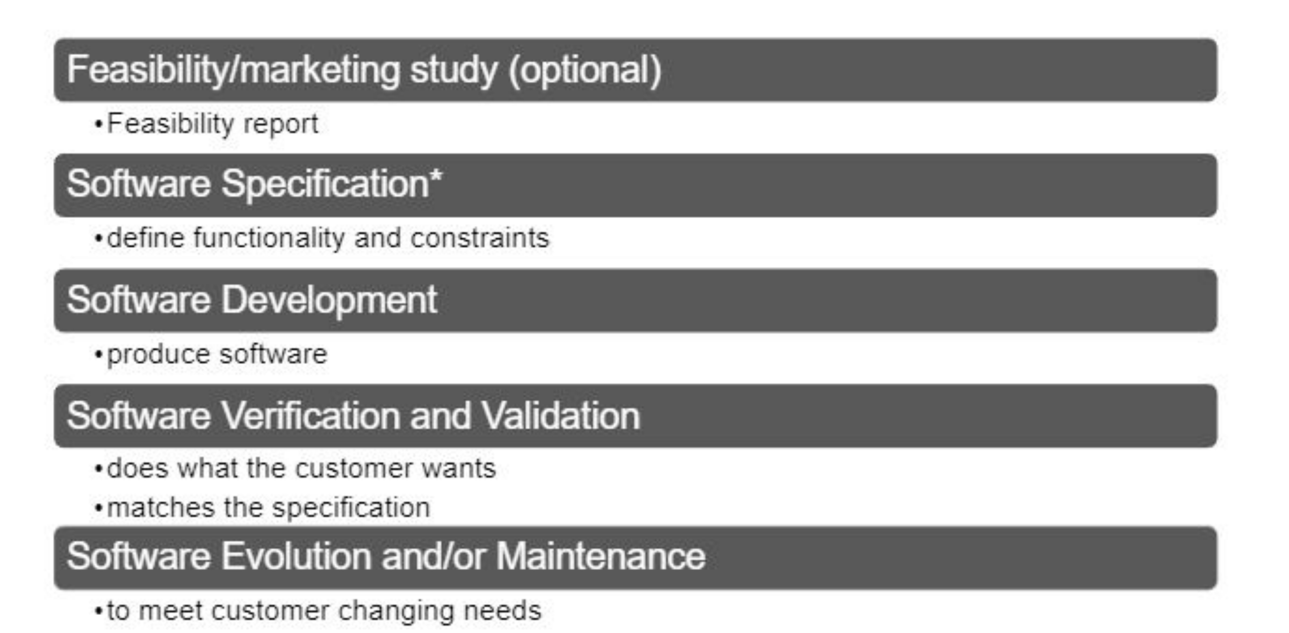
# What is a software process?

The set of activities and associated results which produce a software product or System

(in other words, the sequence of steps required to develop and maintain software)

- sets out the technical and management framework for applying methods, tools and people to the software task
- guides software engineers as they work by identifying their roles and tasks

# What are the common activities



Feasibility/marketing study (optional)

- Feasibility report

Software Specification*

- define functionality and constraints

Software Development

- produce software

Software Verification and Validation

- does what the customer wants
- matches the specification

Software Evolution and/or Maintenance

- to meet customer changing needs

# Software Specification (aka Requirements Engineering)

Understand which services are required (functional and non-functional)

Leads to requirements document:

- get high-level statements from stakeholders (users, administrators, legal entities, etc)
- Get detailed system specification

# Software design and implementation

Develop an executable system for delivery to the customer.

- **Architectural Design:** identify overall structure of the system (components, relations and distribution)
- **Interface Design:** Interfaces between components
- **Component Selection and Design:** look for reusable components or design new ones (can leave details to programmer)
- **Database Design:** Data Structures Design

# Verification and validation

Verification:

Are we building the system right?

(according to the specifications)

Validation:

Are we building the right system?

(meeting the customers expectations)

# Software Evolution

Software is continuously changing over lifetime due to change in requirements and customer needs and other context changes.

# System Delivery

Implementation of the system into the working environment and replacement of the existing system

# System Maintenance

Corrective

Adaptive

Perfective

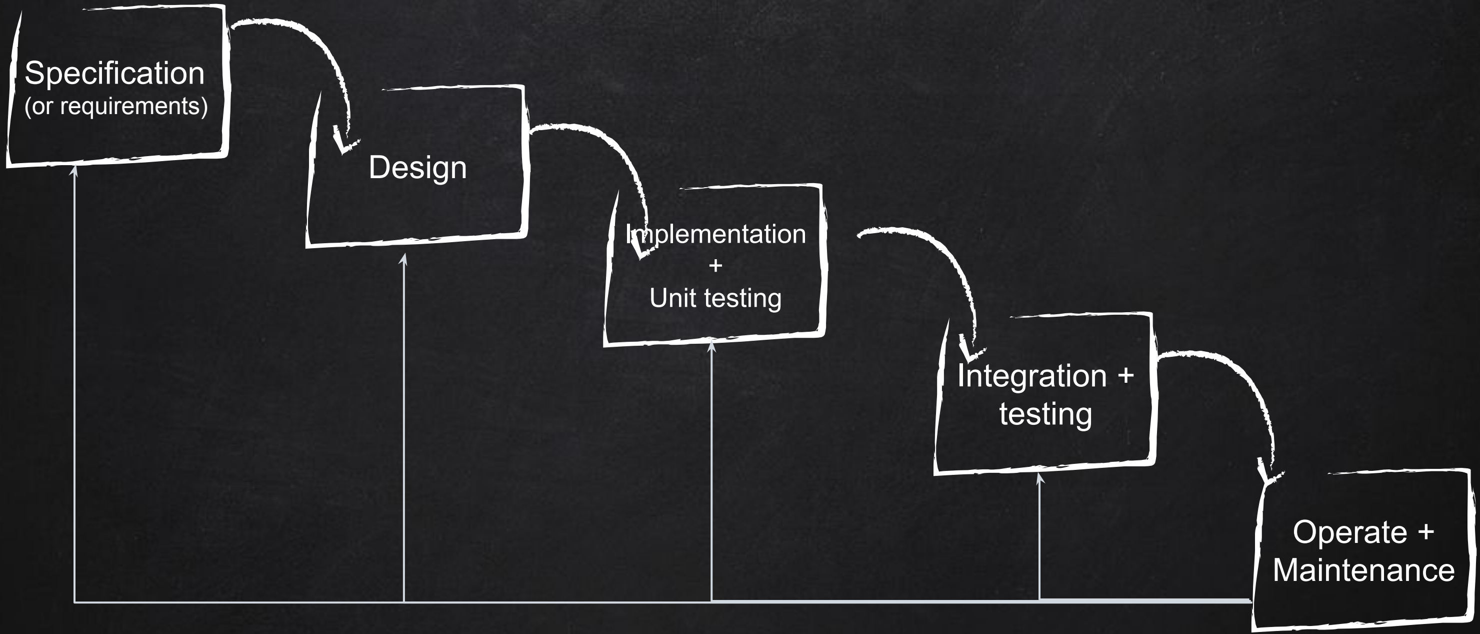
# 2.

# Generic Processes

# There is NO UNIVERSAL PROCESS!

# Waterfall Model

(Royce 1970)



Each phase results in documentation
The following phase should not start until the previous has finished

# Waterfall

## Advantages

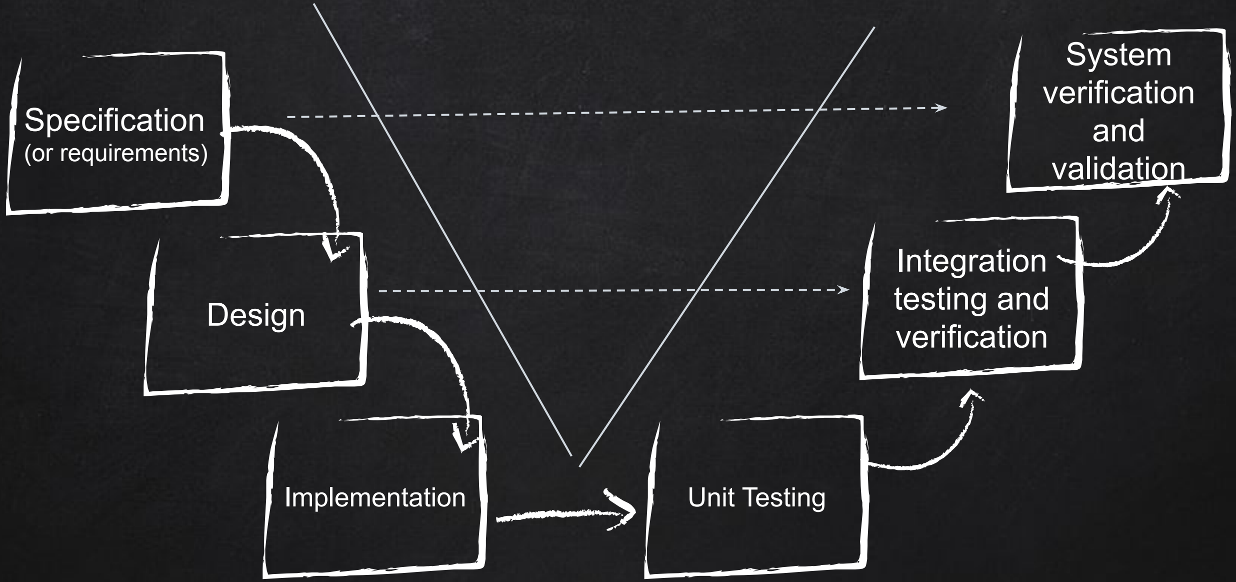
- **Used** in other engineering approaches
- Is **visible** so managers can monitor the process against the plan
- Should only be used when the **requirements are well understood** and the requirements unlikely change during the system development
- More adequate to **large  projects** with large **multidisciplinary teams** or **subsystem integration needs**
- May be suitable for well-understood developments using **familiar technology**

## Disadvantages

- Inflexible partitioning of the project into stages
- **Costs of iteration** lead to costly **documentation** and involve significant rework
- The **customer** only sees the system at the **late phases** where changes are difficult
- **Commitments** must be made at an **early stage** in the process, which makes it difficult to respond to changing

# V MODEL

Adaptation of the waterfall used in Systems Engineering



Specification (or requirements)

Design

Implementation

Unit Testing

Integration testing and verification

System verification and validation

# V-Model

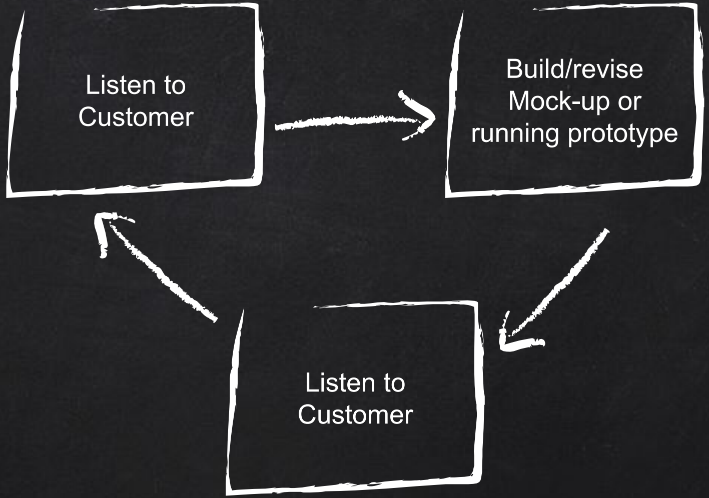
## Advantages

- Minimizes project risks – due to the explicit concerns:
  - Verification (Am I doing things right?)
  - Validation (Am I doing the right thing?)
- Improvement and guarantee of quality
- reduction of total cost over the entire project and systems life cycle

## Disadvantages

- Apart from the mentioned benefits it suffers from the same problems of the waterfall model

# Prototyping



Listen to Customer

Build/revise Mock-up or running prototype

Listen to Customer

# Prototyping

## Advantages

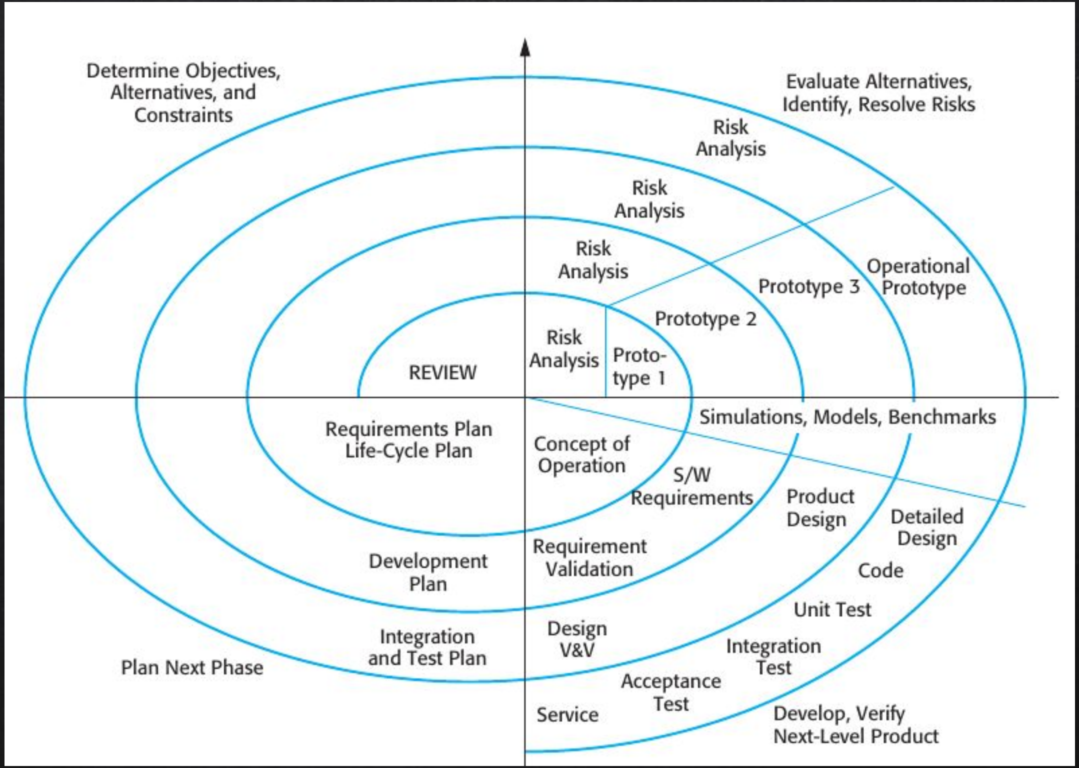
- Ideally mock-up serves as mechanism for identifying requirements and can be integrated in other processes
- Users like the method, get a feeling for the actual system
- Less ideally may be the basis for completed product

## Disadvantages

- prototypes often ignore quality/performance/maintenance issues
- No stable architecture, and poorly structured
- Not cost effective to produce documentation
- may create pressure from users on deliver earlier
- may use a less-than-ideal platform to deliver
- Not adequate for large, complex long-lived systems with different teams developing different parts
-

# Spiral Model

## (Boem 1988)

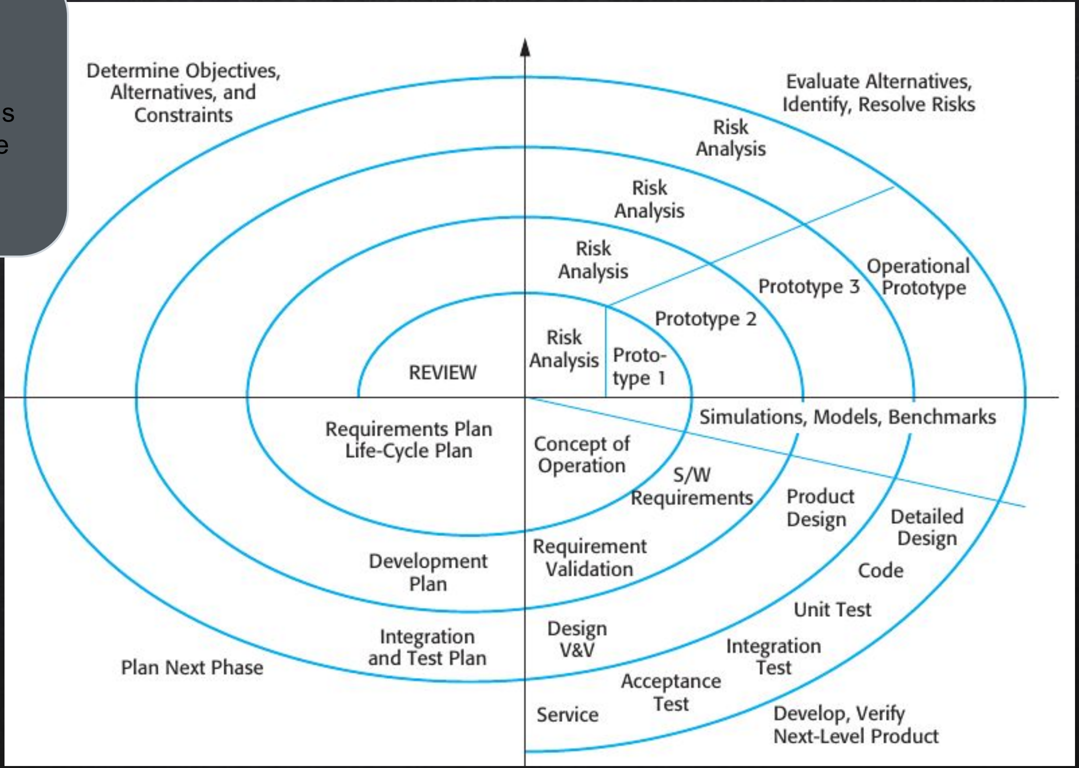


Taken from "Software Engineering", Jan Sommerville 9th edition

# Spiral Model

## (Boem 1988)

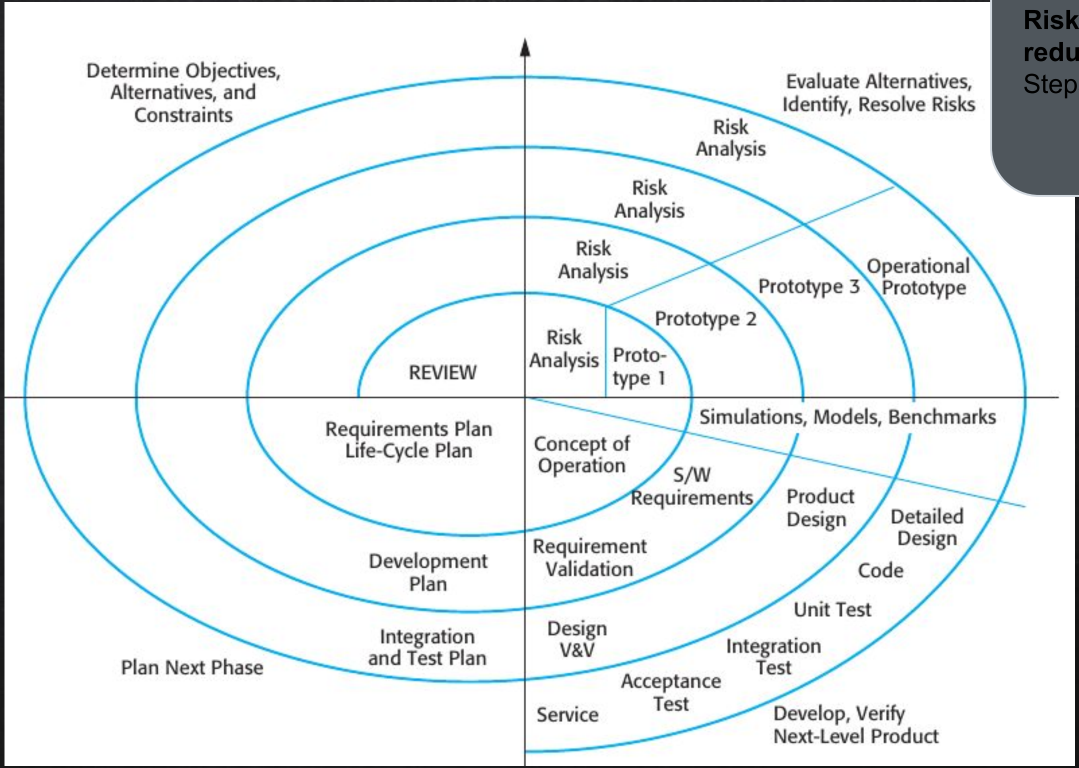
**Objective Setting**
Objectives defined. Also, detailed management plan is defined and project risks are identified.



Taken from "Software Engineering", Jan Sommerville 9th edition

# Spiral Model

## (Boem 1988)

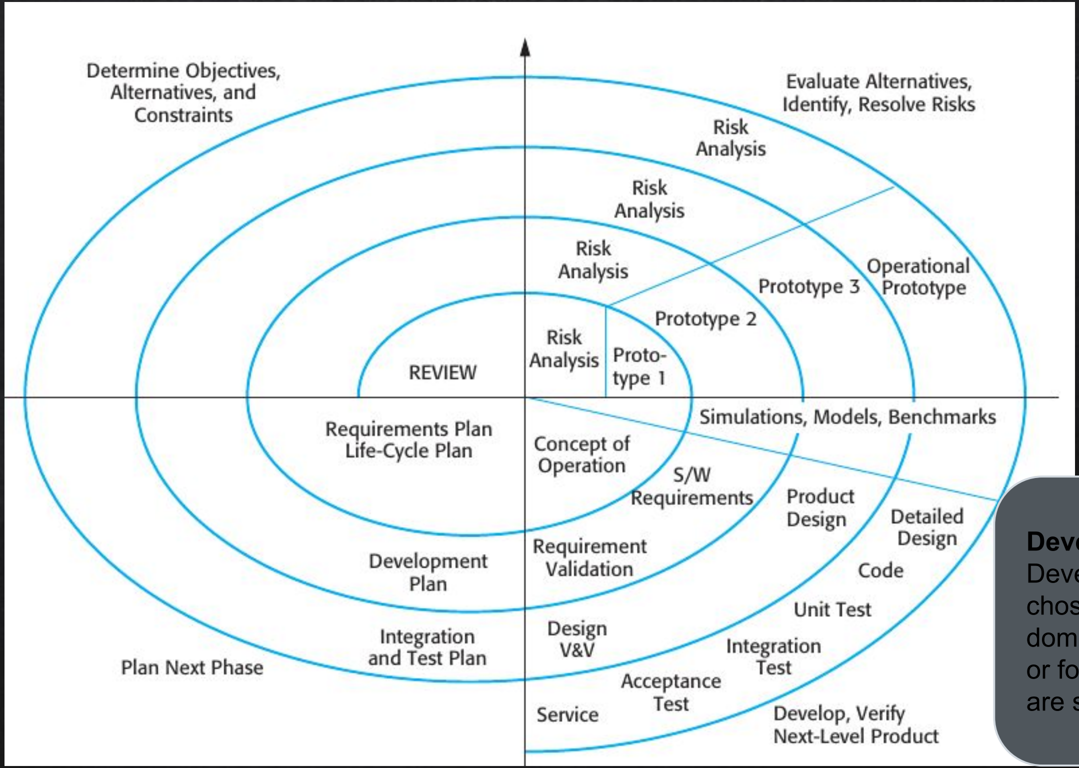


**Risk assessment and reduction**
Steps taken to reduce risk

Taken from "Software Engineering", Jan Sommerville 9th edition

# Spiral Model

## (Boem 1988)

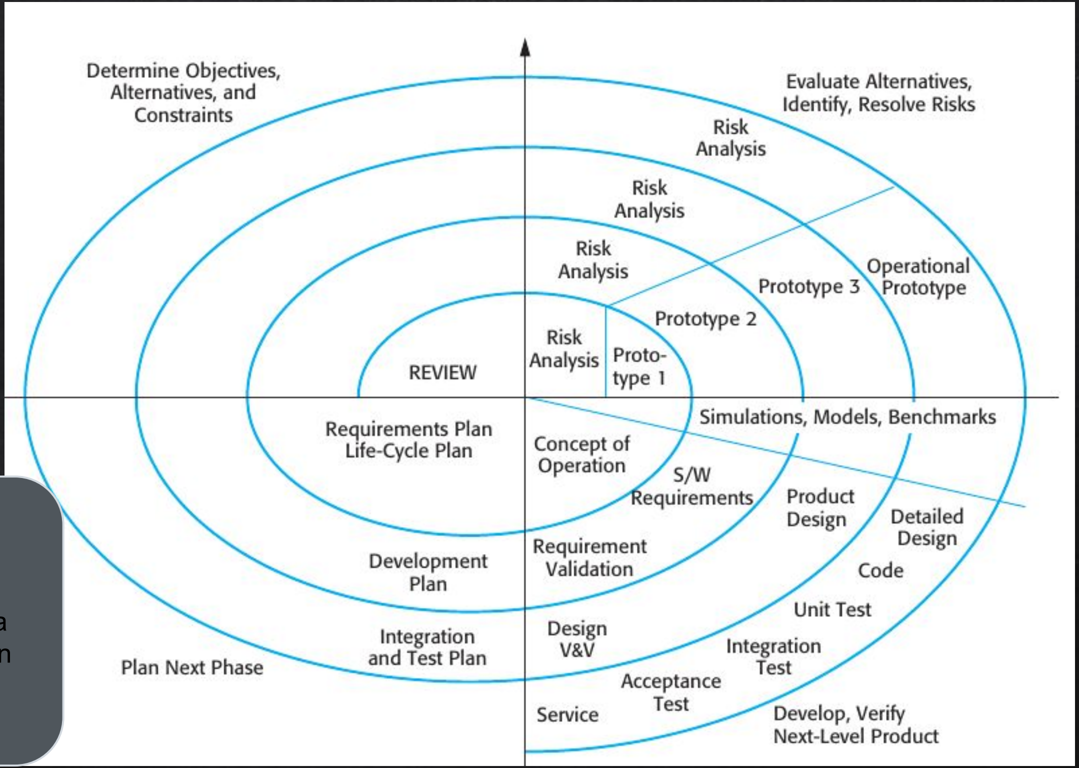


**Development and Validation**
Development model is chosen. (prototyping for e.g dominant user interface risks, or formal waterfall if e.g there are safety risks)

Taken from "Software Engineering", Jan Sommerville 9th edition

# Spiral Model

## (Boem 1988)



**Planning**
Project is reviewed and go-no-go decision made to a further loop. Plans are drawn to the nt phase

Taken from “Software Engineering”, Jan Sommerville 9th edition

# Spiral Model

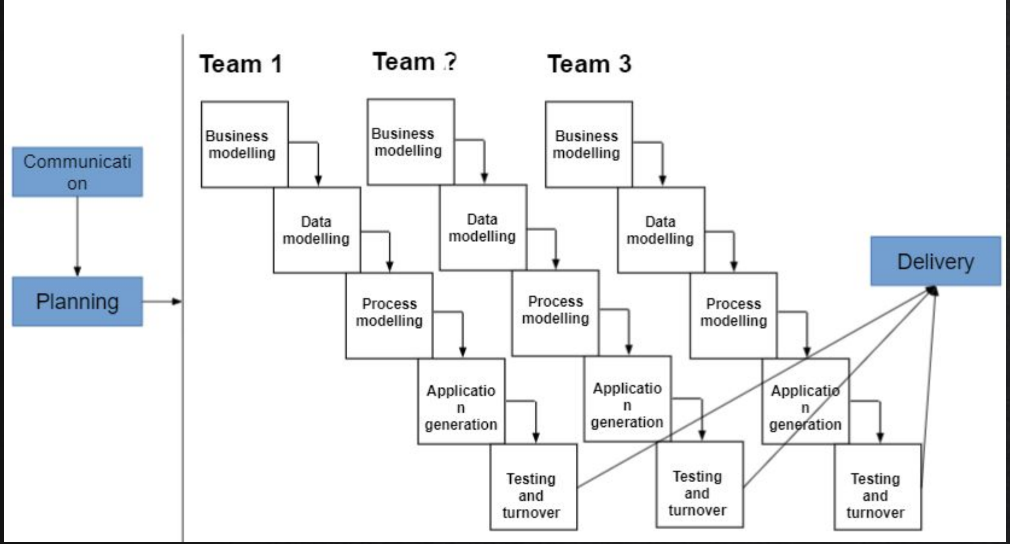
Advantages

- From the traditional is considered one of the best approaches for its incrementality where the stakeholders are soon presented with versions until the system is developed completely
- Adequate for large scale software development
- Very strong in risk handling unanticipated risks
- Follows the complete software life-cycle until it is no longer used

Disadvantages

- Customer interaction/involvement is not at focus
- Requires excellent management and risk assessment

# There are many other processes…
# RAD – Rapid application development



Similar problems to the waterfall

But Faster

Demands for large teams

Demands for careful integration

# There are many other processes…
# Formal Systems Development Methods

- Such as B-method /Schneider 2001; Wordswoth 1996) or VDM (Vienna Development Method) that from formal specifications (mathematical models) there are tools to support model analysis  and then automatic translation to code
- Suited for systems with stringent safety and reliability or security needs
- Mostly embedded systems
- Require very specialized expertise

# Going Agile

# Motivation

Plan driven approaches were developed for software development by large teams, working for different companies, often geographically dispersed working on software for long time.

e.g. software for: Automotive, Avionics

But, too heavy for small and medium business systems

more time spent on organizing and less on developing and testing

# Agile Manifesto values

**The goal is to uncover better ways of developing software.**

- Individuals and Interactions more than processes and tools
  - process and tools help but persons are the key
- Working Software more than comprehensive documentation
  - documents are useful but we should focus “on the real value”
- Customer Collaboration more than contract negotiation
  - instead of distanced client that only shows in beginning and end and only looks for contractual compliance
- Responding to Change more than following a plan
  - being flexible and adaptive dealing with change with small cycles instead of following a rigid pan

That is, while there is value in the items on the right, the manifesto values the items on the left more.
Does not mean we should not do the things on the right!!!!!

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
scottadams@aol.com
www.dilbert.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

Agile approaches focus on software and not on design and documentation (not plan driven).

Ideal for:

- small medium sized products and apps
- custom system development within organization

Specification, design and implementation are intertwined

Adequate when requirements change rapidly, meant to deliver working software quickly.

# Agile Principles

**Customer satisfaction** by early and continuous delivery of valuable software.

**Welcome changing requirements** even in late development.

**Deliver working software frequently** weeks rather than months

**Close, daily cooperation** between customer and developers (provide and help prioritize new requirements)

Projects are built around **motivated individuals** who should be trusted

**Face-to-face conversation** is the best form of communication (co-location)

**Working software is the primary measure of progress**

**Sustainable development**, able to maintain a constant pace

**Continuous attention** to technical excellence and good design

**Simplicity—**the art of maximizing the amount of work not done—is essential

Best architectures, requirements, and designs emerge from **self-organizing teams**

Regularly, the **team reflects on how to become more effective**, and adjusts accordingly

# See you in the next Lecture

# References

Software Engineering, by Ian Sommerville, Pearson, (either 9th or 10th edition)

see chapters 2 and 3

The Agile Manifesto: https://agilemanifesto.org/