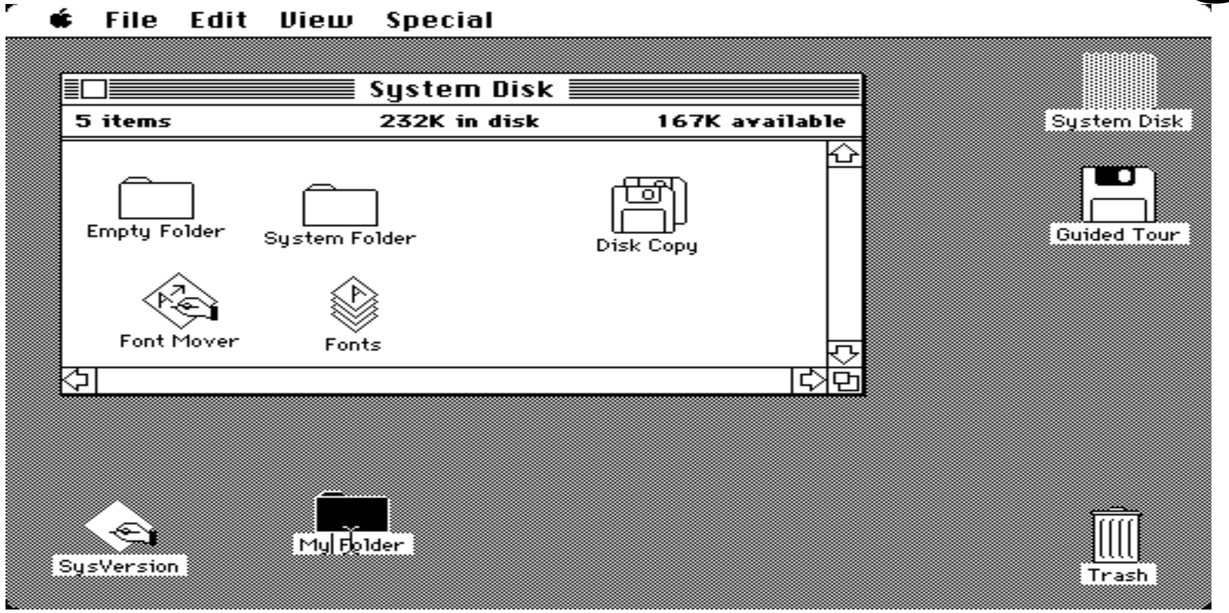
Software Engineering

Hans-Petter Halvorsen

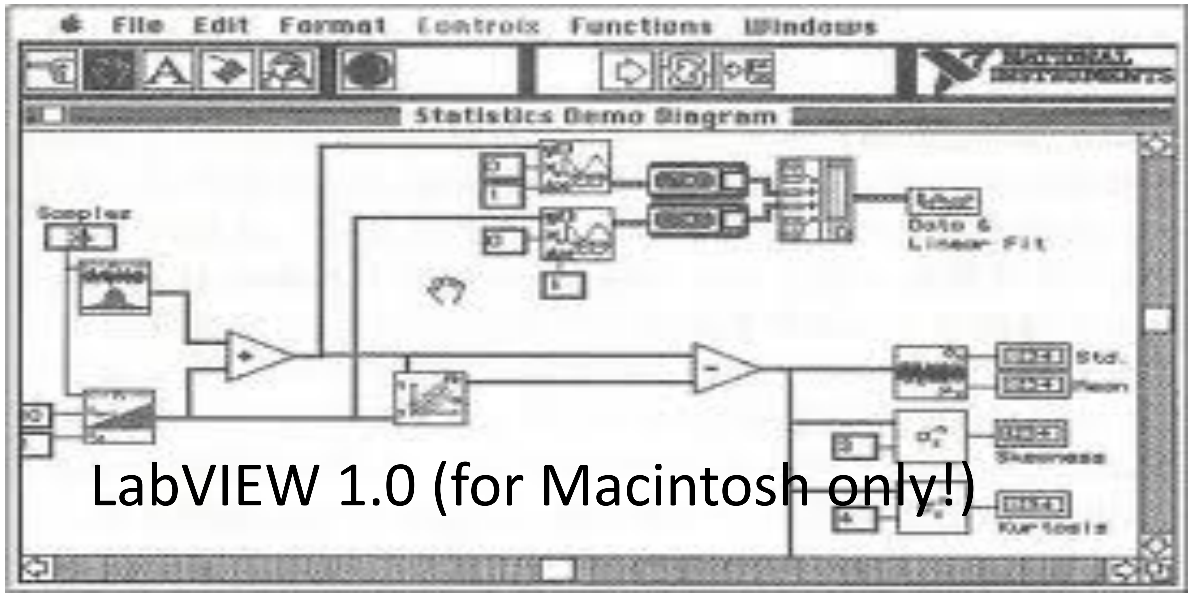
The beginning

Mac OS 1.0

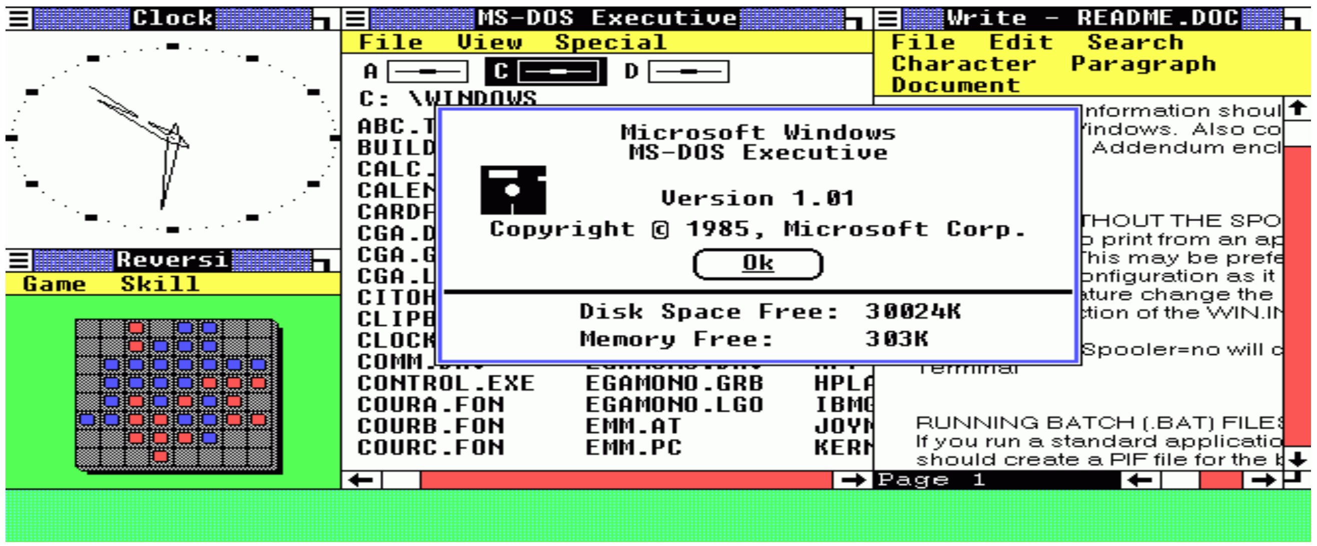


1984: Macintosh

1985: Windows 1.0



LabVIEW 1.0 (for Macintosh only!)



Microsoft Windows
MS-DOS Executive

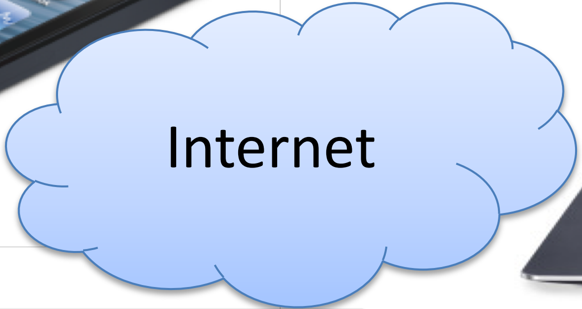
Version 1.01

Copyright © 1985, Microsoft Corp.

Ok

Disk Space Free: 30024K
Memory Free: 303K

Today



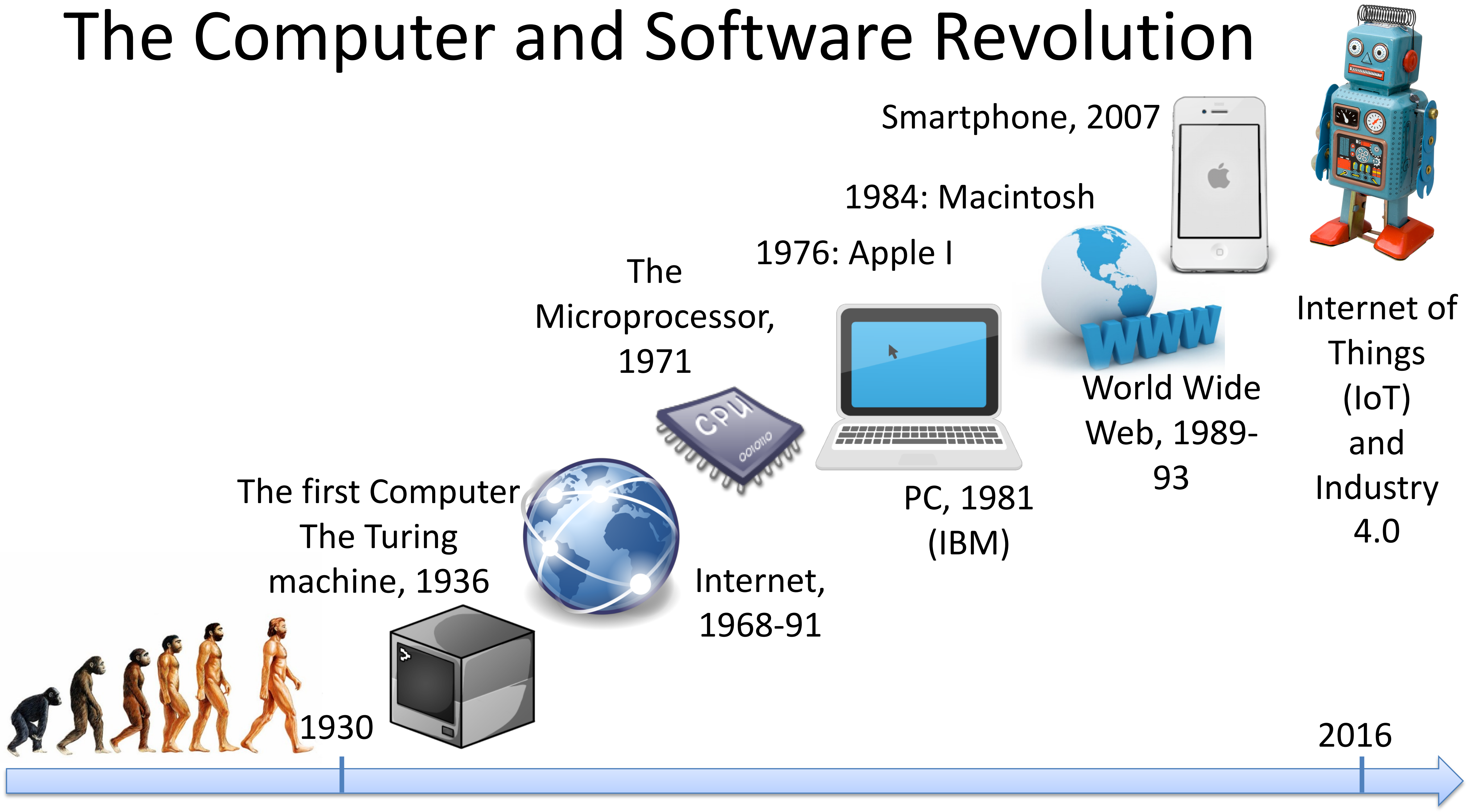
Apps Web

Smartwatches

Smart TVs

Tablets

The Computer and Software Revolution



Why Software Engineering?

- There are many differences between a one-person programming and large software system development.
- The degree of complexities between these two approaches make it necessary to bring more discipline into the development process.
- Modern software engineering is very complex and there are large numbers of failures in many software projects and defects encountered in the software products.
- All infrastructure for human livings rely on Software today
- That's why Software Engineering is needed

Why Software Engineering?

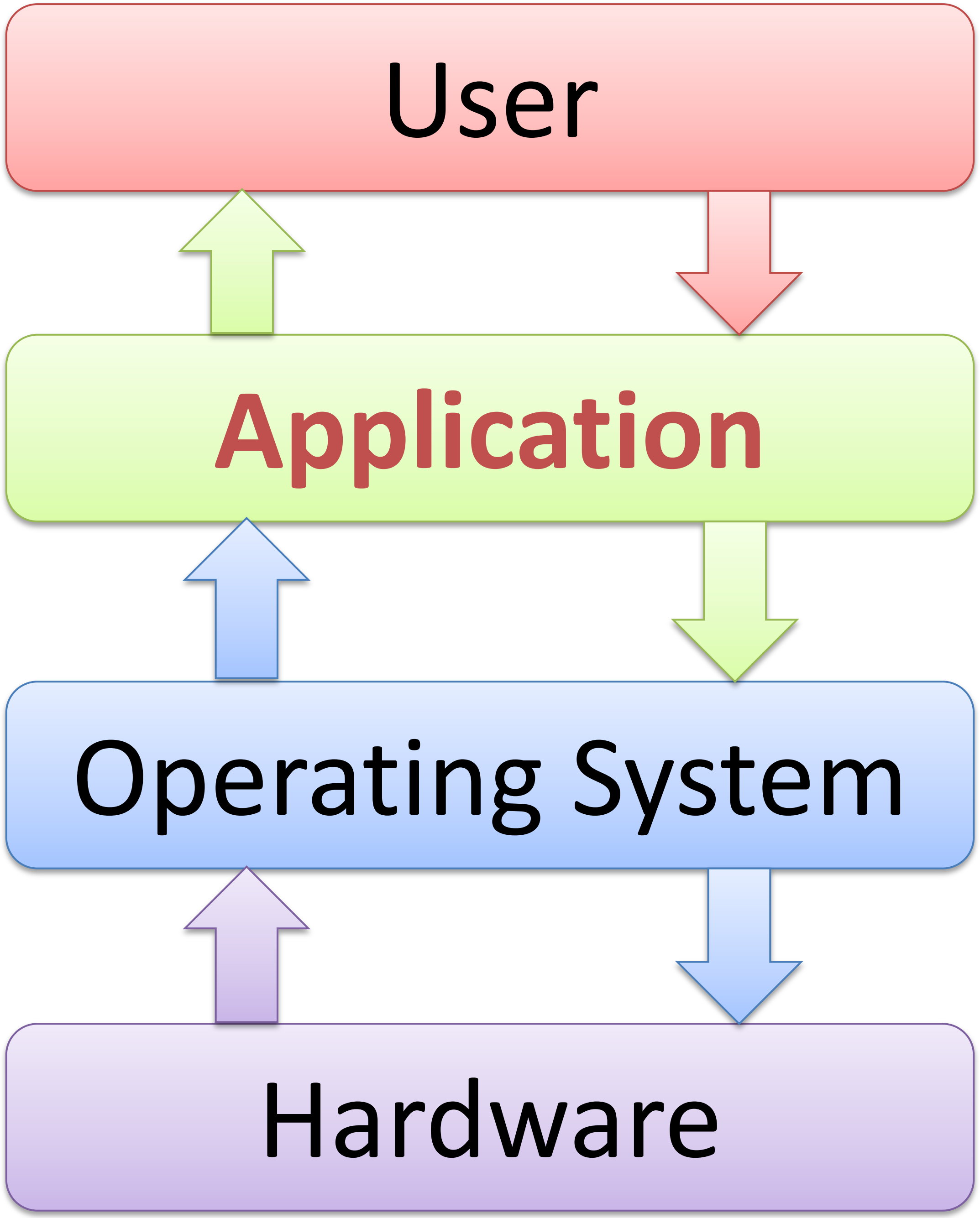
- Understand Customer Requirements
  - What does the customer needs (because they may not know it!)?
  - Transform Customer requirements into working software
- Planning
  - How do we reach our goals?
  - Will we finish within deadline?
  - Resources
  - What can go wrong?
- Implementation
  - What kind of platforms and architecture should be used?
  - Split your work into manageable pieces
- Quality and Performance
  - Make sure the software fulfill the Customers needs

What is Software Engineering?

Software Engineering is the profession of the Development and Management of High Quality Software Systems within given Time and Cost frames

Software Engineering

Planning, Requirements, Design, Implementation, Testing, Deployment, Maintenance, etc.



Who are going to use the software?

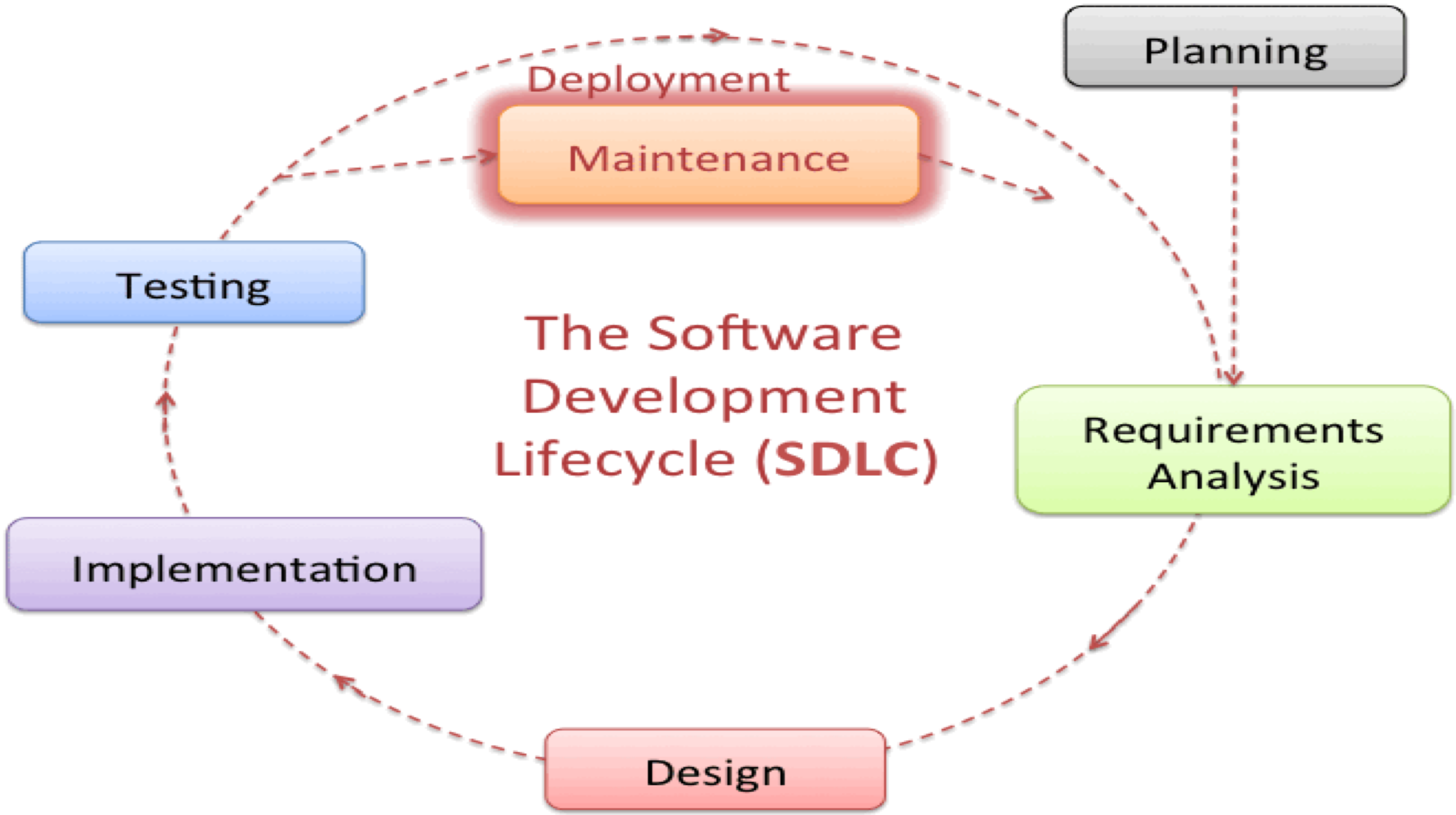
Desktop, Web, Mobile?

Windows, MacOS, Linux, Android, iOS, etc.

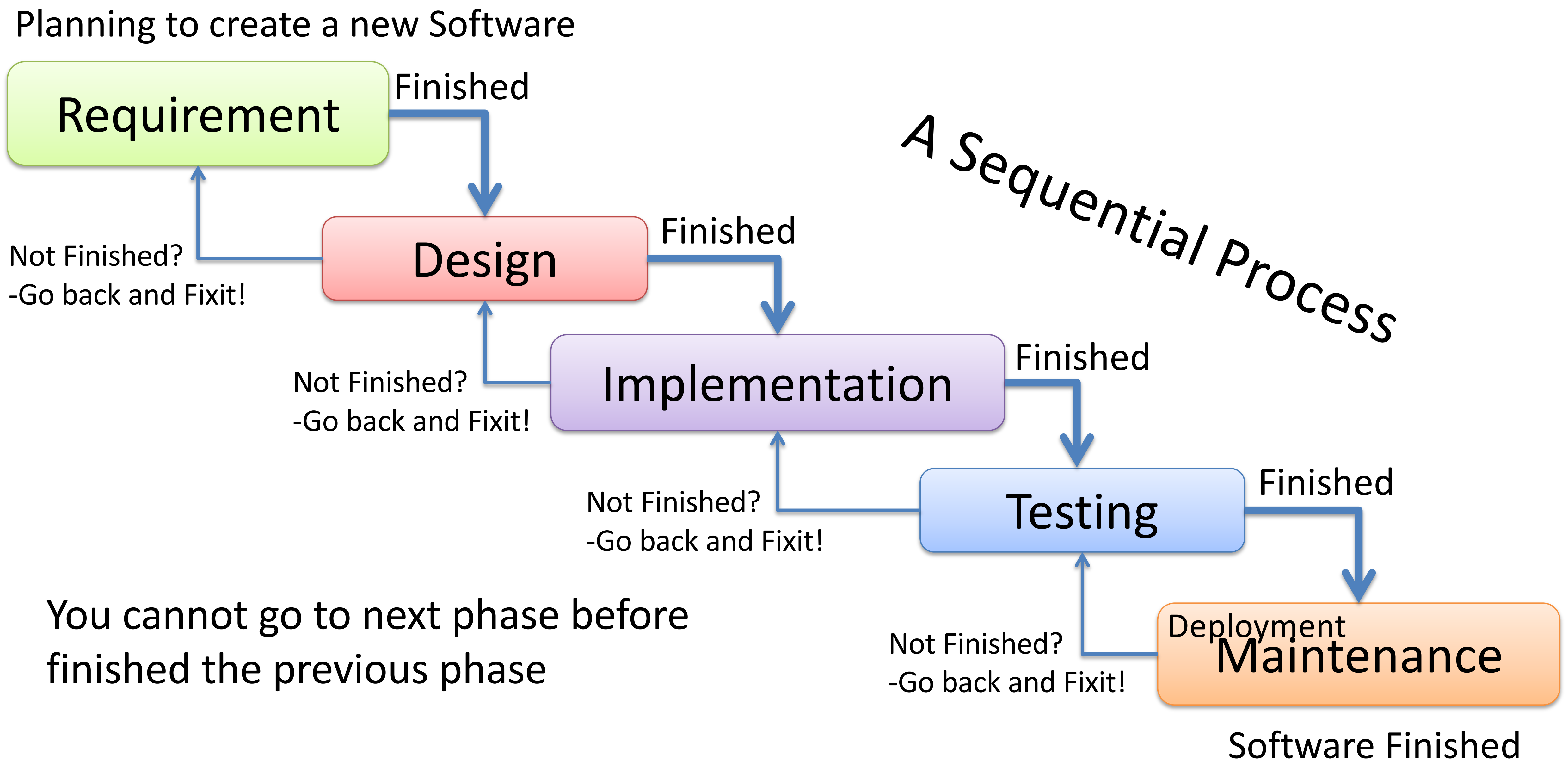
PC, Mac, Smartphone, Tablet, Smart TV, etc.

Software Engineering Disciplines

- Software Planning, Project Management
- Requirements Engineering/Analysis
- Database Modeling
- UML (Unified Modeling Language)
- Software Development Processes (Waterfall, Agile Development, Scrum, ...)
- Software Platforms (Desktop, Mobile, Web, Cloud, ..)
- Software Architecture
- Software Implementation
- Source Code Control and Bug Tracking
- Software Testing
- Software Documentation
- Software Deployment and Maintenance


Planning
Deployment
Maintenance
Testing
The Software Development Lifecycle (SDLC)
Requirements Analysis
Implementation
Design

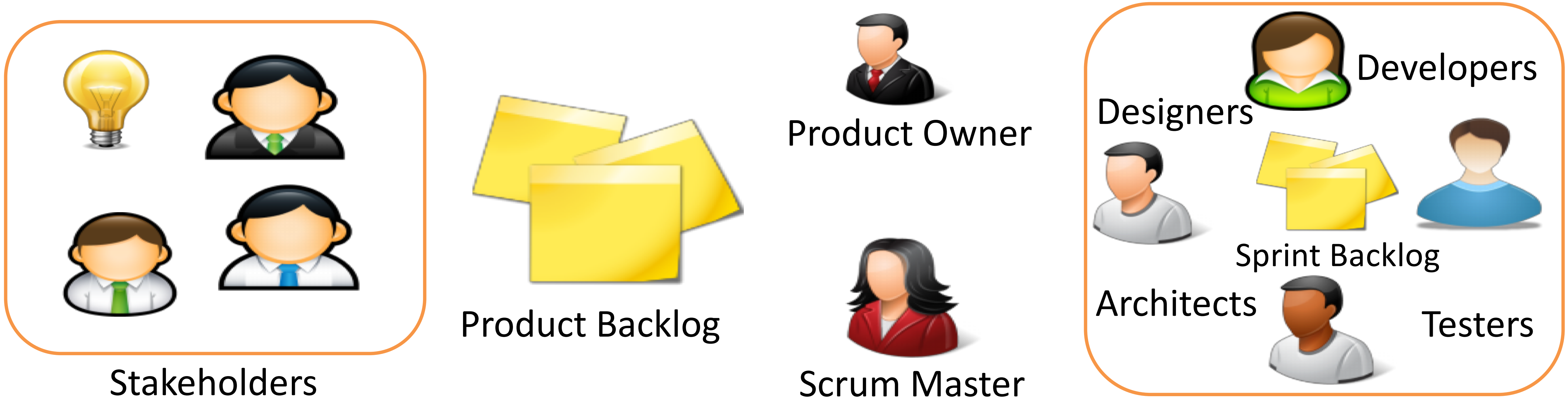
The Waterfall Model



You cannot go to next phase before finished the previous phase

The Scrum Framework

Scrum Members:



Stakeholders

Product Backlog

Product Owner

Scrum Master

Developers

Designers

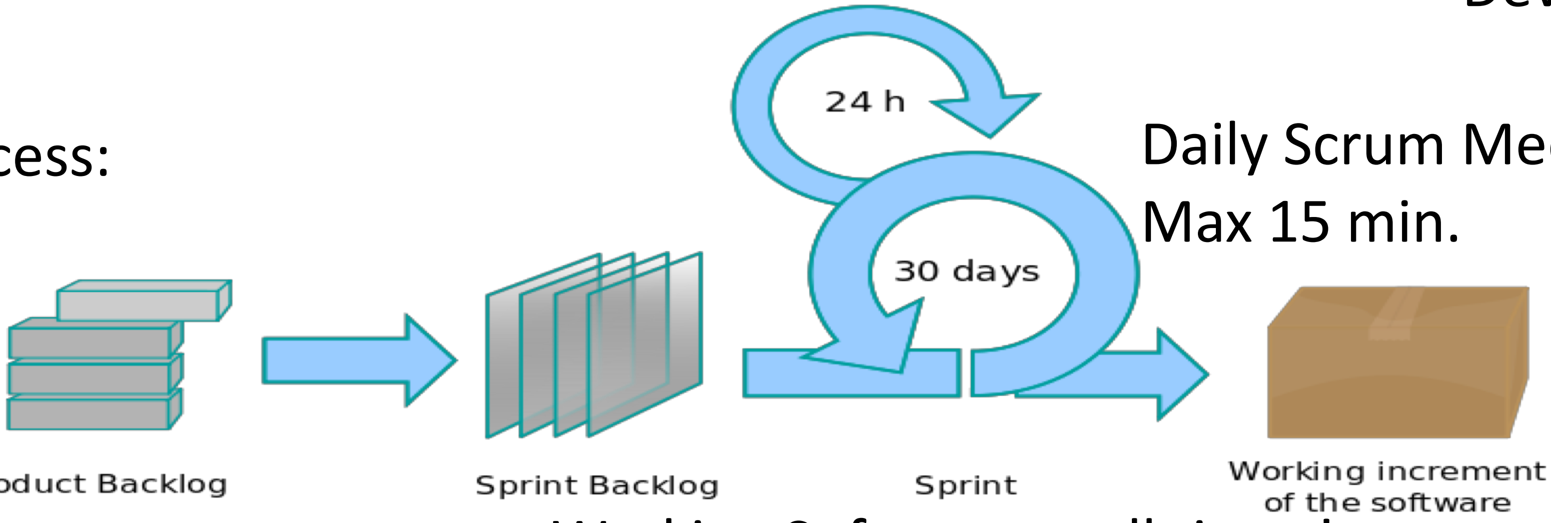
Sprint Backlog

Architects

Testers

Development Team
3-9 persons

Scrum Process:

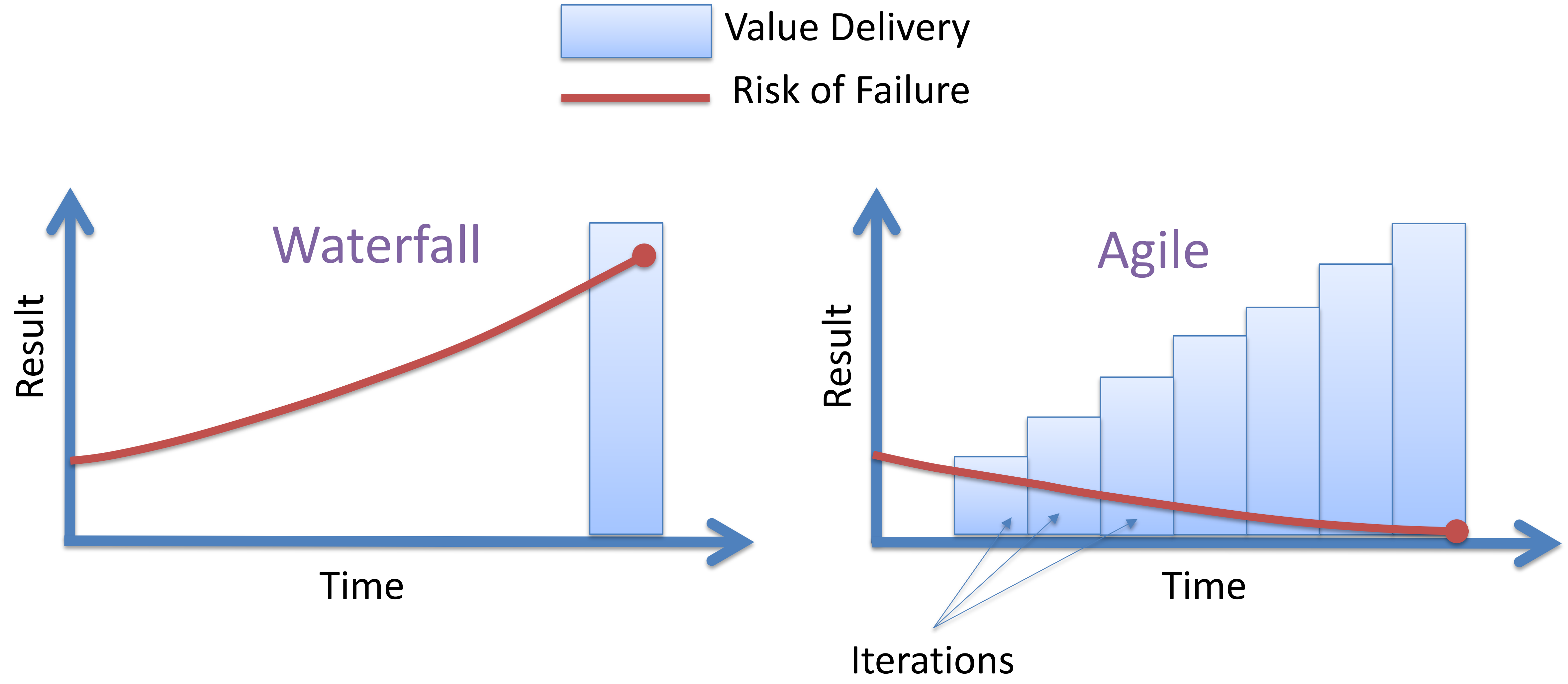


Daily Scrum Meetings
Max 15 min.

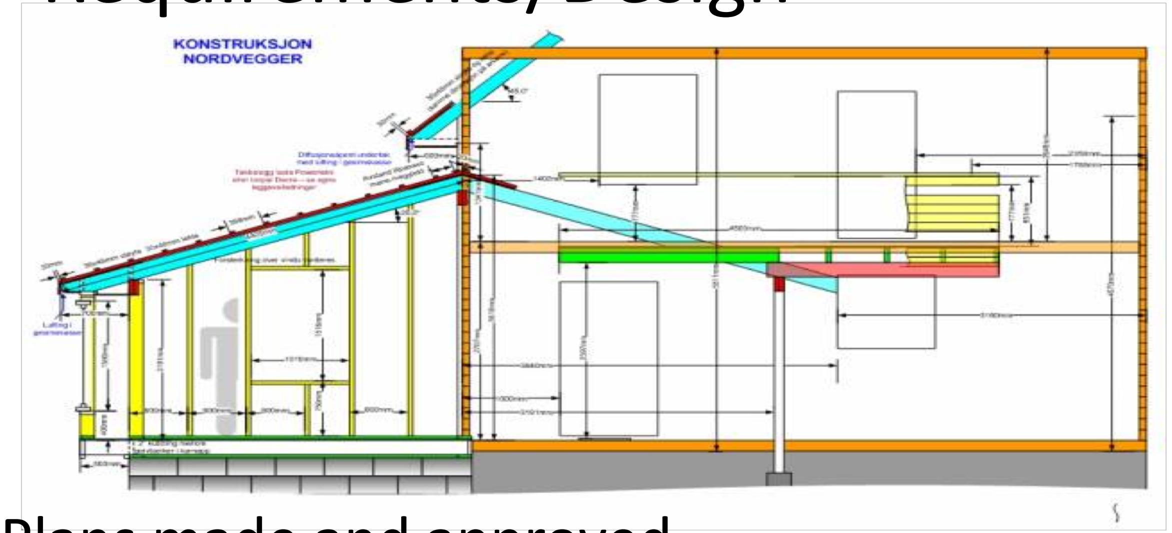
Sprint Review

A Framework for Software Development - Working Software at all times!

Waterfall vs. Agile



Requirements/Design



Plans made and approved

Alpha

Foundation finished, building structure started
A "proof" that you can do it, PoC (Proof of Concept)

Beta

Building structure finished,
Inside work on track

RC

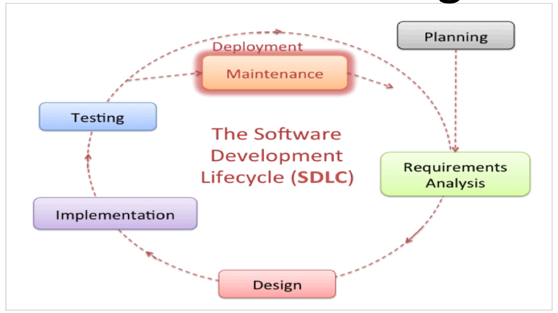
Furniture, Flowers and small
adjustments missing

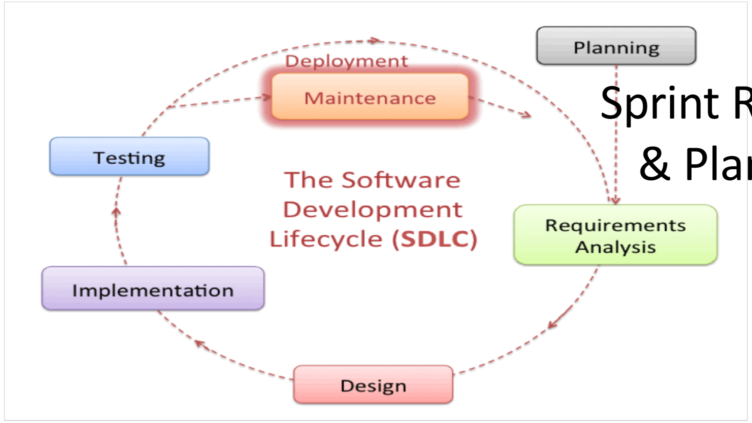
RTM

Ready for Sale or Move in

Software Development

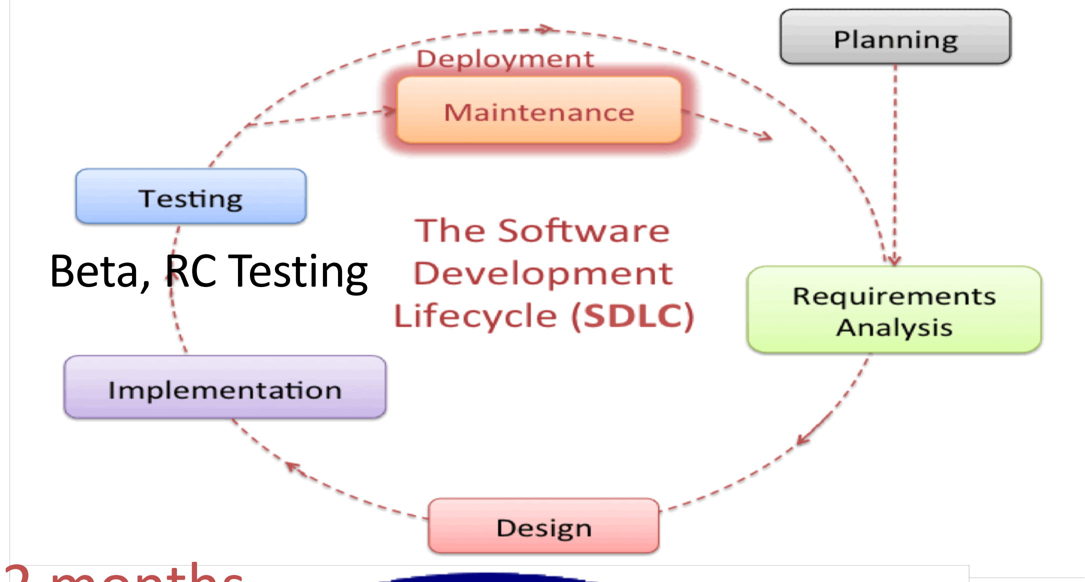
Daily Scrum Meetings





Sprint Reviews & Planning

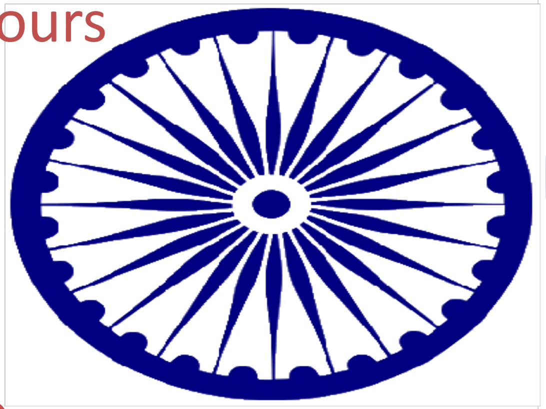
Beta, RC Testing

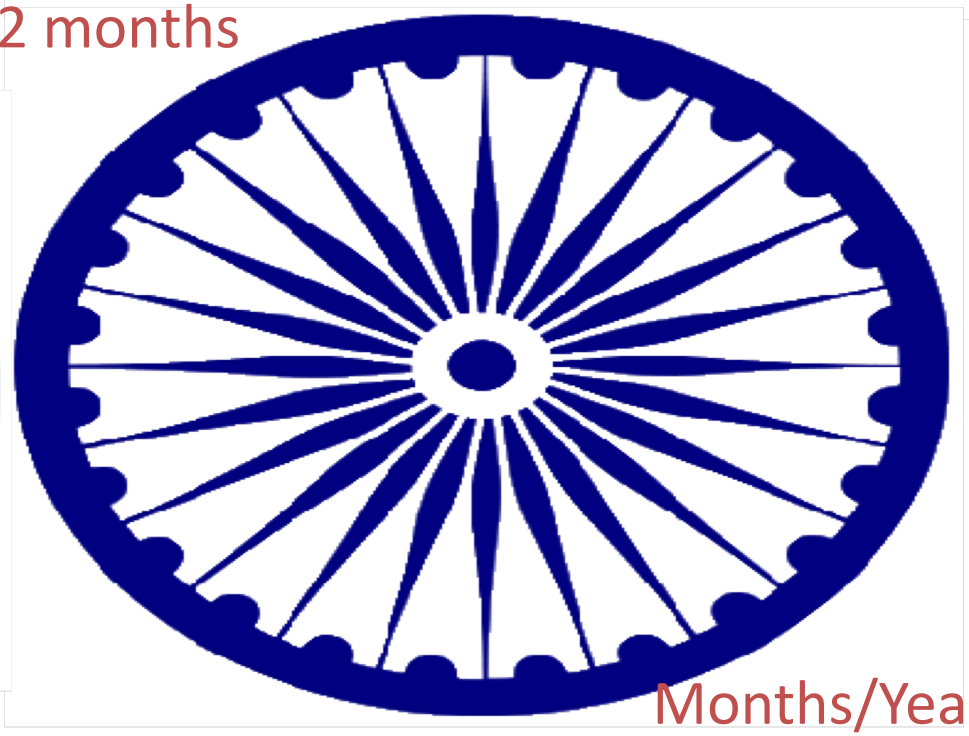


1 -12 months

2-4 weeks

24 hours





Days

Weeks

Months/Years

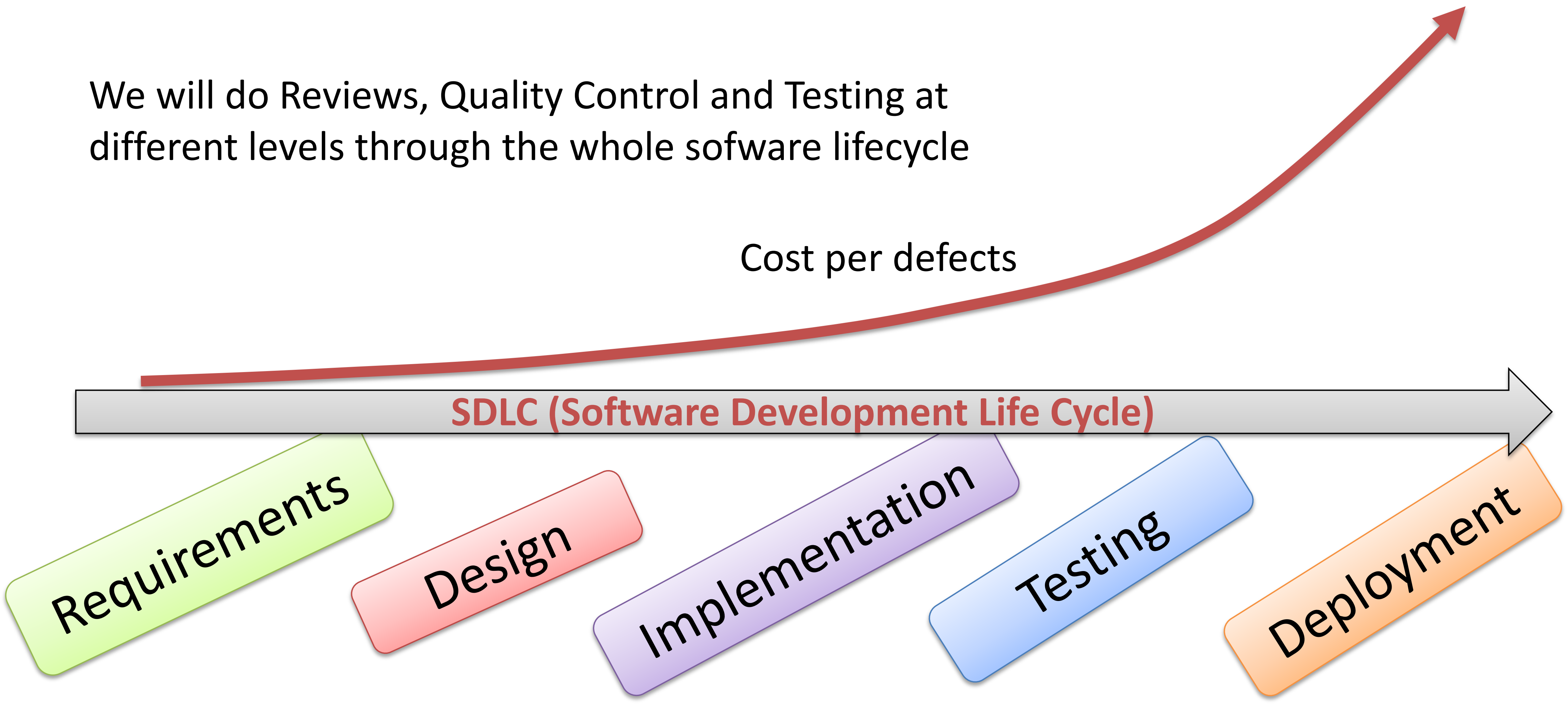
Working Software at all times.
Testing every day

Internal Iterations/Sprints

Public Beta, RC Releases

Why Do Reviews, Quality Control and Testing?

We will do Reviews, Quality Control and Testing at different levels through the whole sofware lifecycle



Software Requirements & Design

Requirements (WHAT):

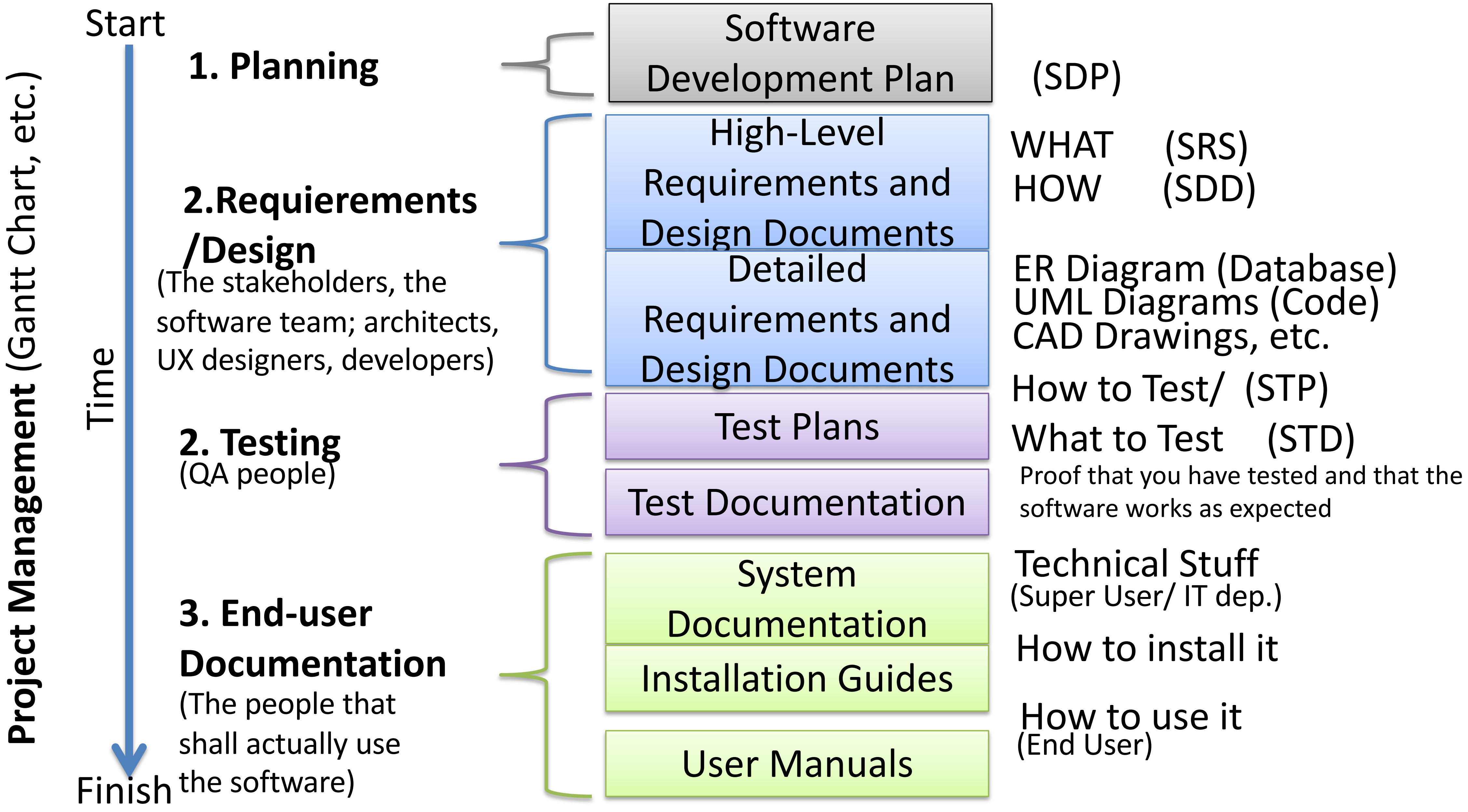
- **WHAT** the system should do
- Describe what the system should do with Words and Figures, etc.
- **SRS** – Software Requirements Specification

Software Design (HOW):

- **HOW** it should do it
- Examples: GUI Design, UML, ER diagram, CAD, etc.
- **SDD** – Software Design Document

Many dont separate SRS and SDD documents, but include everything in a Requirements document. In practice, requirements and design are inseparable.

Typical Software Documentation



Project Management (Gantt Chart, etc.)

Start

Time

Finish

1. Planning

Software Development Plan (SDP)

2.Requierements /Design
(The stakeholders, the software team; architects, UX designers, developers)

High-Level Requirements and Design Documents

WHAT (SRS)
HOW (SDD)

Detailed Requirements and Design Documents

ER Diagram (Database)
UML Diagrams (Code)
CAD Drawings, etc.

2. Testing
(QA people)

Test Plans

How to Test/ (STP)
What to Test (STD)

Test Documentation

Proof that you have tested and that the software works as expected

3. End-user Documentation
(The people that shall actually use the software)

System Documentation

Technical Stuff
(Super User/ IT dep.)

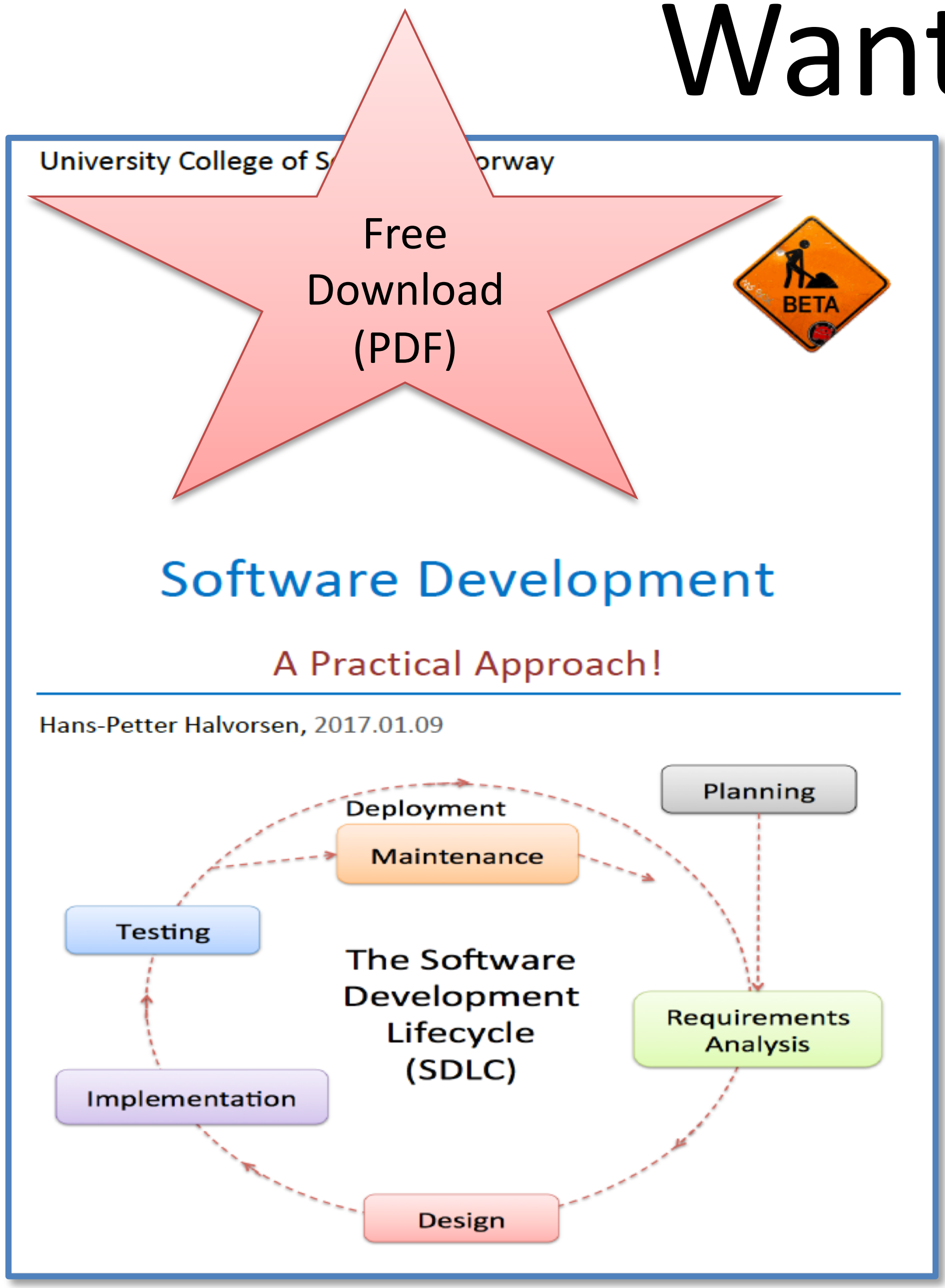
Installation Guides

How to install it

User Manuals

How to use it
(End User)

Want to learn more?



University College of S orway

Free Download (PDF)

Software Development

A Practical Approach!

Hans-Petter Halvorsen, 2017.01.09

https://www.halvorsen.blog

Free Textbook, Videos, and other Resources

https://www.halvorsen.blog/documents/programming/software_engineering

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: http://www.halvorsen.blog