Chair of Mobile Business & Multilateral Security

Lecture 08

Business Informatics 2 (PWIN) WS 2019/20

ICS Development I

Software Engineering

Prof. Dr. Kai Rannenberg

Chair of Mobile Business & Multilateral Security
Johann Wolfgang Goethe University Frankfurt a. M.

Agenda

- Introduction to Software Engineering
- Software Engineering Process Overview
- Software Development Process Models

What is Software?

- What is software?
  - Computer programs and associated documentation.
  - Software is developed for a particular customer (individual software) or for a general market (standard software).
- What are the attributes of good software?
  - Good software is supposed to deliver the required functionality and performance to the user and has to be maintainable, reliable and usable.

Who Needs Software?

- Most software used in organisations is built for people with specific needs.
  - A *stakeholder* is anyone who has an interest (or a stake) in the software.
  - A *user* is someone who uses the software in order to perform tasks.
  - Sometimes stakeholders are users; but most of the time stakeholders do not use software.
    - For example, a senior manager (e.g. CEO or CTO in a company) usually has a stake in the software to be built, even if they are never going to use it.

Source: Stellmann, Greene (2006)

Who Builds Software?

- Software is typically built by a team of software engineers, which include:
  - **Business analysts** or **requirements analysts**, who gather requirements for a software by interviewing users and stakeholders
  - **Designers and architects**, who plan, design, and model the technical architecture and system of the software
  - **Programmers**, who write the code for the software
  - **Testers**, who verify that the software meets its requirements and behaves as expected

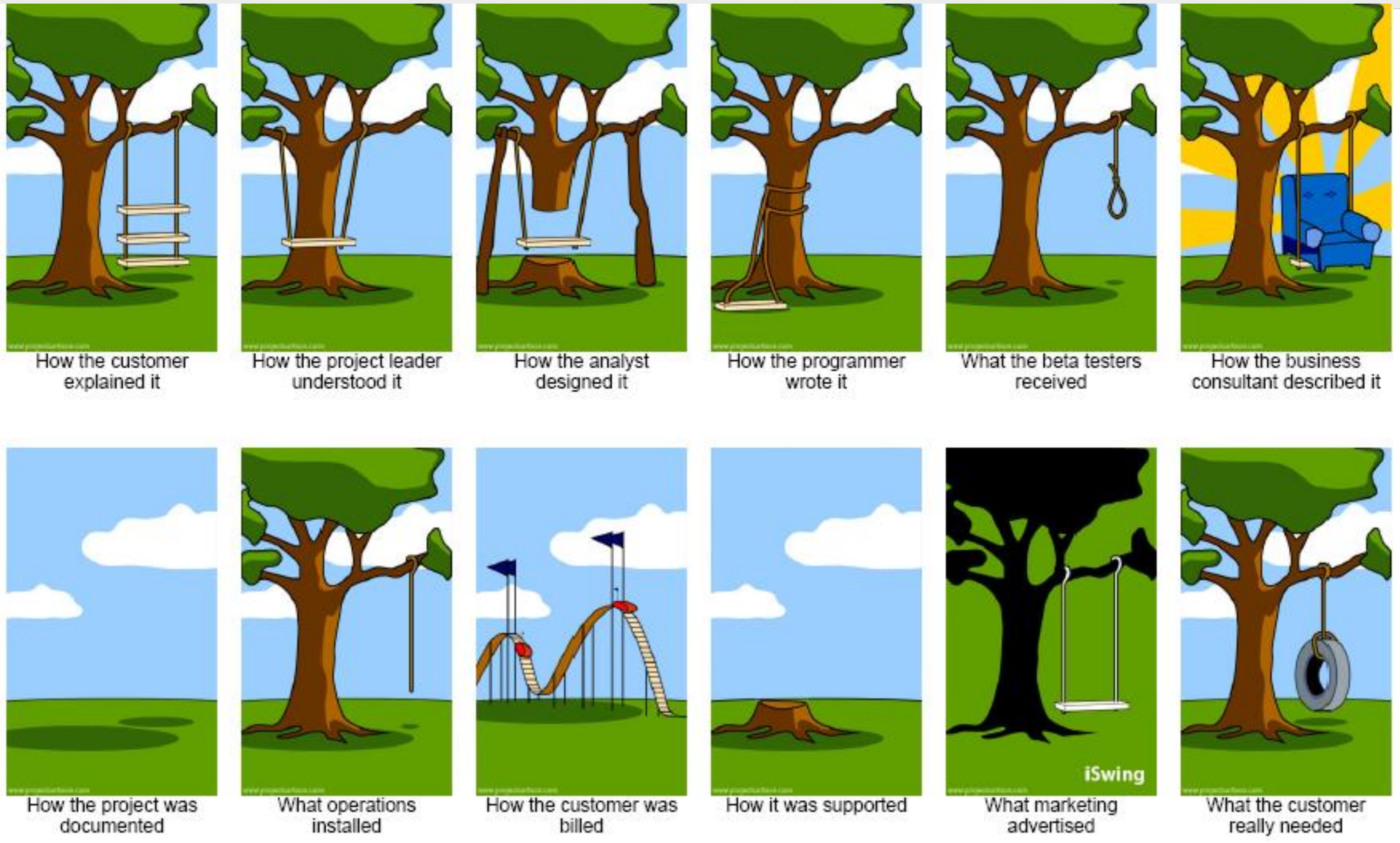
Source: Stellmann, Greene (2006)

Why Do Software Development Projects Fail?

- People begin programming before they understand the problem.
- The team has an unrealistic idea about how much work is involved.
- Mistakes are injected early but discovered late.
- Managers try to “test” quality into software.

Source: Stellmann, Greene (2006)

Why Do Software Development Projects Fail?


How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

How to Ensure That Software Projects Succeed?

- **Application of "Good Engineering Practices"**
  - Managers and teams often want to skip important engineering practices – especially effort estimation, continuous reviews, requirement acquisition and testing.
  - If it would be faster to build the software without these practices, they would never be used.
  - The reason for applying these practices is to save time and increase software quality by accurate planning and revealing mistakes early.
  - Not applying theses practices increasing development time while reducing the software quality.

Source: Stellmann, Greene (2006)

Software Engineering (SE)

- **Software Engineering (SE)** is a discipline that is concerned with all aspects of software production from the early stages of *system specification and system design* down to *rollout and system maintenance*.

- **Engineering** as discipline means applying appropriate theories and methods to solve problems while considering organisational and financial constraints.

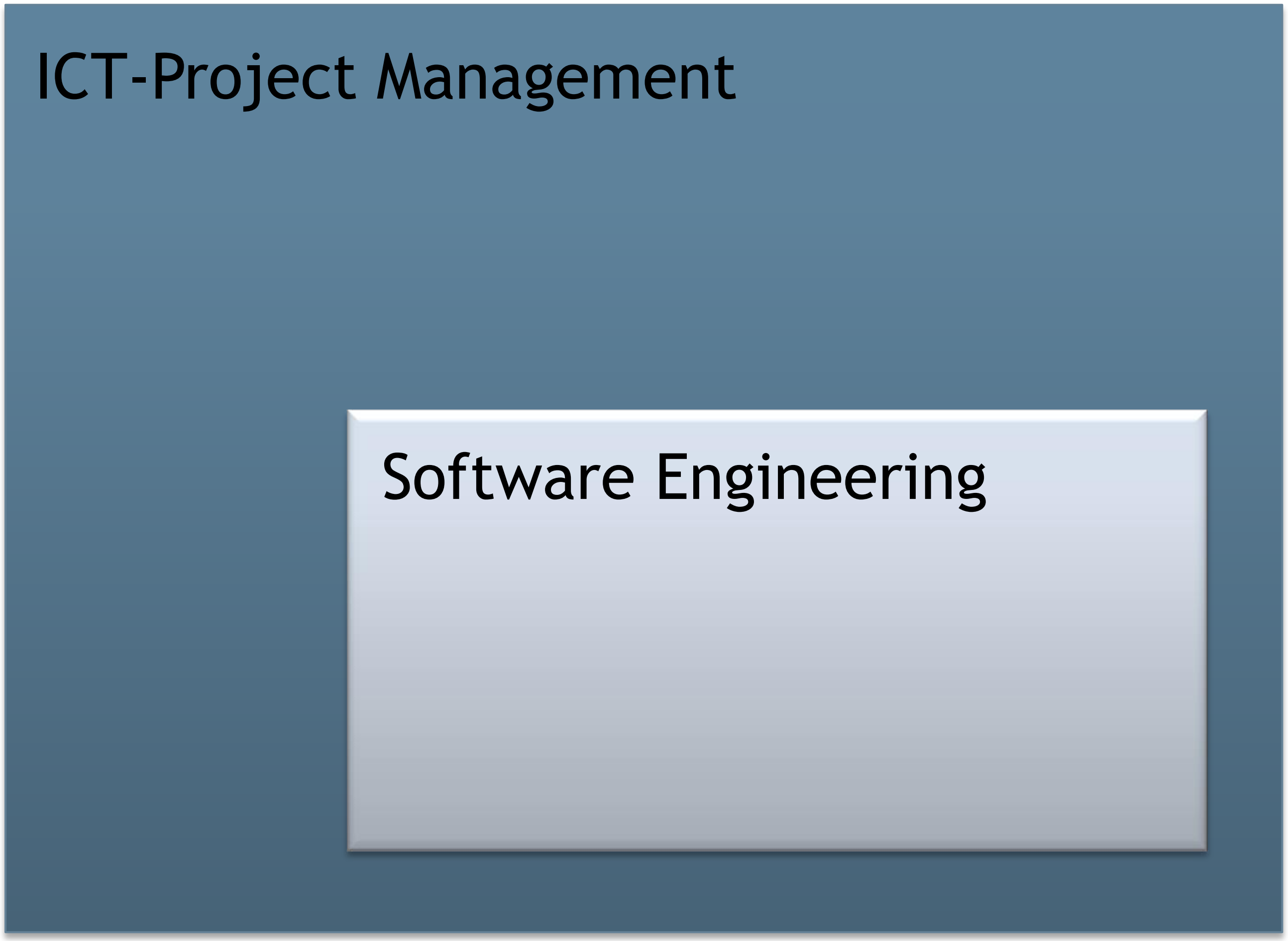
- **Software Engineering** covers all aspects of software production
  - Technical development process (main task)
  - Project management, development of tools, methods etc. in order to support software production (supporting tasks)

Source: Sommerville (2007)

Important
Software Engineering Objectives

- Development of software according to specified quality standards
- Avoidance of disastrous time delays and exceed of budget
- Addressing of changing requirements while staying on budget and deadlines

ICT-Project Management vs. Software Engineering



Software Project Planning

Vision and Scope Document

Software Project Plan

Project Schedule

Risk Plan

Project Management

Vision and Scope Document

- One of the most important tools of a project manager
- Enables that stakeholders and developers share a common understanding of the needs – and the needs addressed by the software

- Typical document outline

  1. Problem Statement
     a) Project background
     b) Stakeholders
     c) Users
     d) Risks
     e) Assumptions

  2. Vision of the Solution
     a) Vision statement
     b) List of features
     c) Scope of phased release *(optional)*
     d) Features that will not be developed

DILBERT by Scott Adams
YOUR USER REQUIRE-MENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.

Source: Stellmann, Greene (2006)

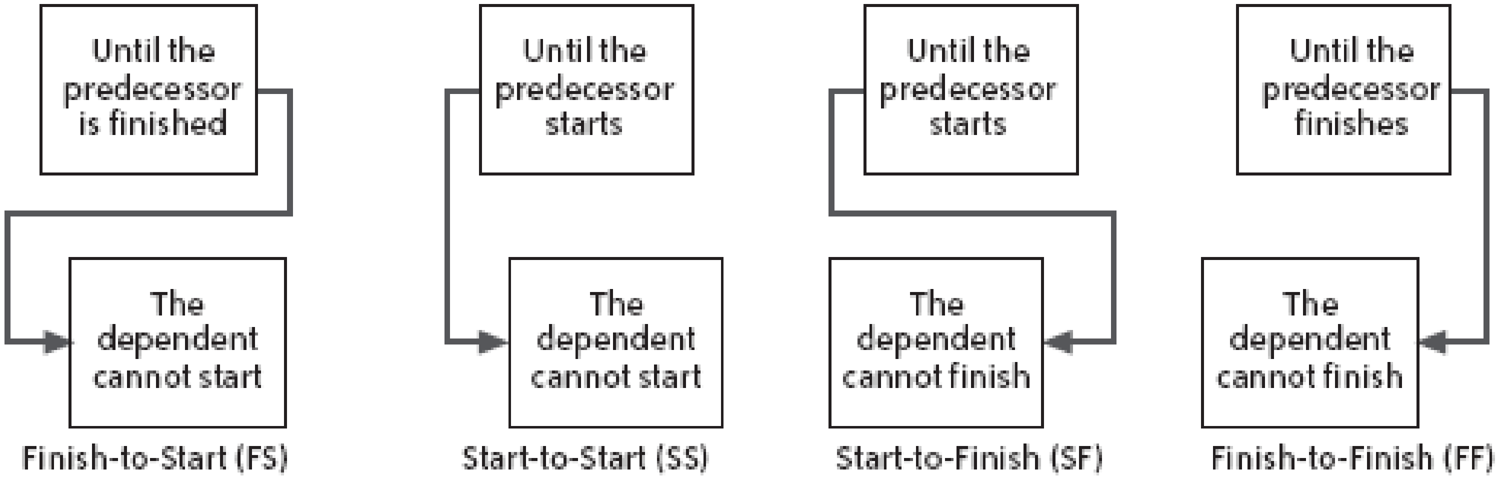
Project Plan

- Used by many people in an organisation
  - **Project manager:** Communication of project status to stakeholders, planning of team activities
  - **Team members:** Understanding the context of their work
  - **Senior manager:** Verifying that costs and schedule are under control
  - **Stakeholders:** Ensuring the project is on track
- *Project plan* consists of:
  - **Statement of work (SOW):** Describes list of features to be developed and their required estimated effort
  - **Resource list:** List of all resources required for the project
  - **Work breakdown structure:** List of required tasks to develop the software
  - **Project schedule:** Assignment of resources and calendar time to a required task
  - **Risk plan:** Risks that could threaten the project and potential means to mitigate these risks

Source: Stellmann, Greene (2006)

Developing a Project Schedule (1)

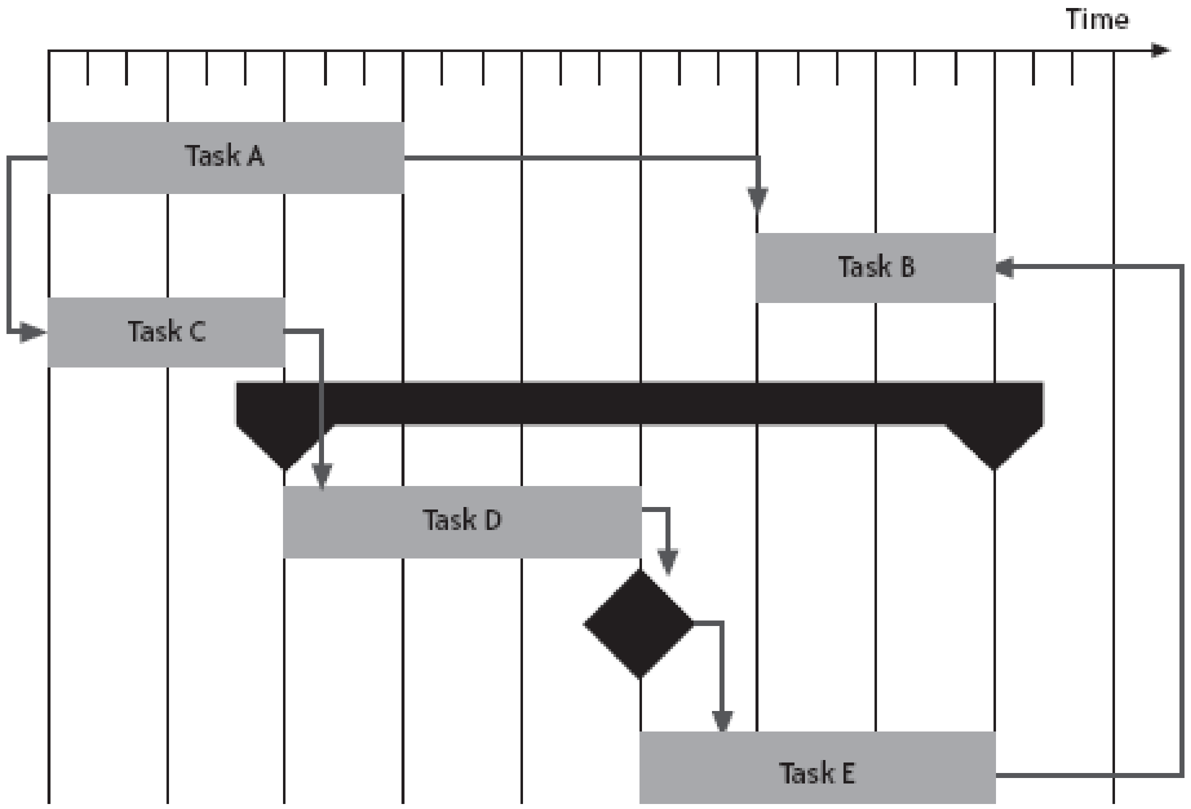
1. Allocate resources to the task
2. Identify dependencies between tasks


Until the predecessor is finished
The dependent cannot start
Finish-to-Start (FS)
Until the predecessor starts
The dependent cannot start
Start-to-Start (SS)
Until the predecessor starts
The dependent cannot finish
Start-to-Finish (SF)
Until the predecessor finishes
The dependent cannot finish
Finish-to-Finish (FF)

Source: Stellmann, Greene (2006)

Developing a Project Schedule (2)

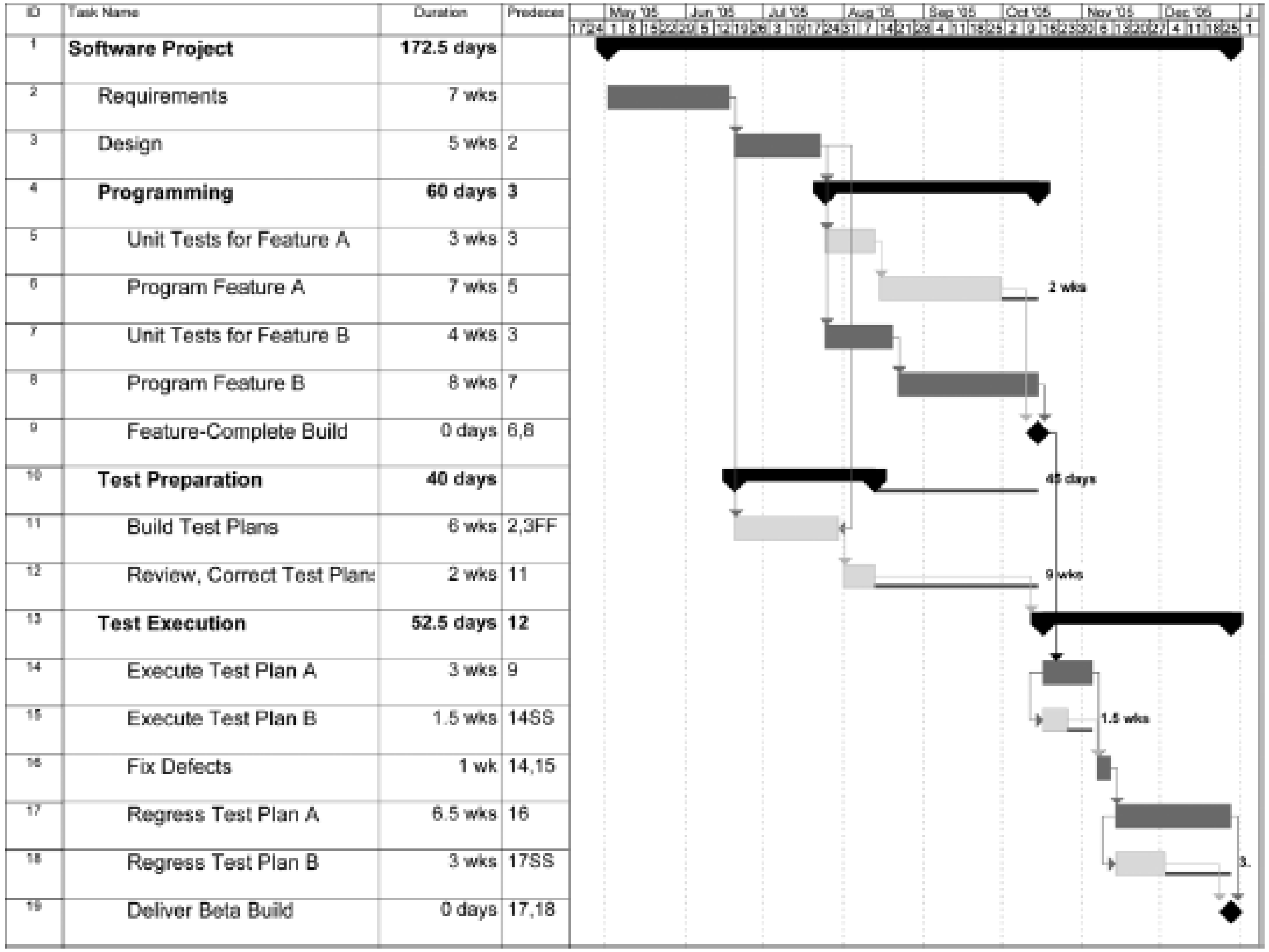
3. Create a schedule



Source: Stellmann, Greene (2006)

Developing a Project Schedule (3)

Project Schedule



ID	Task Name	Duration	Predeces
1	**Software Project**	**172.5 days**	
2	Requirements	7 wks	
3	Design	5 wks	2
4	**Programming**	**60 days**	**3**
5	Unit Tests for Feature A	3 wks	3
6	Program Feature A	7 wks	5
7	Unit Tests for Feature B	4 wks	3
8	Program Feature B	8 wks	7
9	Feature-Complete Build	0 days	6,8
10	**Test Preparation**	**40 days**	
11	Build Test Plans	6 wks	2,3FF
12	Review, Correct Test Plans	2 wks	11
13	**Test Execution**	**52.5 days**	**12**
14	Execute Test Plan A	3 wks	9
15	Execute Test Plan B	1.5 wks	14SS
16	Fix Defects	1 wk	14,15
17	Regress Test Plan A	6.5 wks	16
18	Regress Test Plan B	3 wks	17SS
19	Deliver Beta Build	0 days	17,18

Source: Stellmann, Greene (2006)

Risk Plan

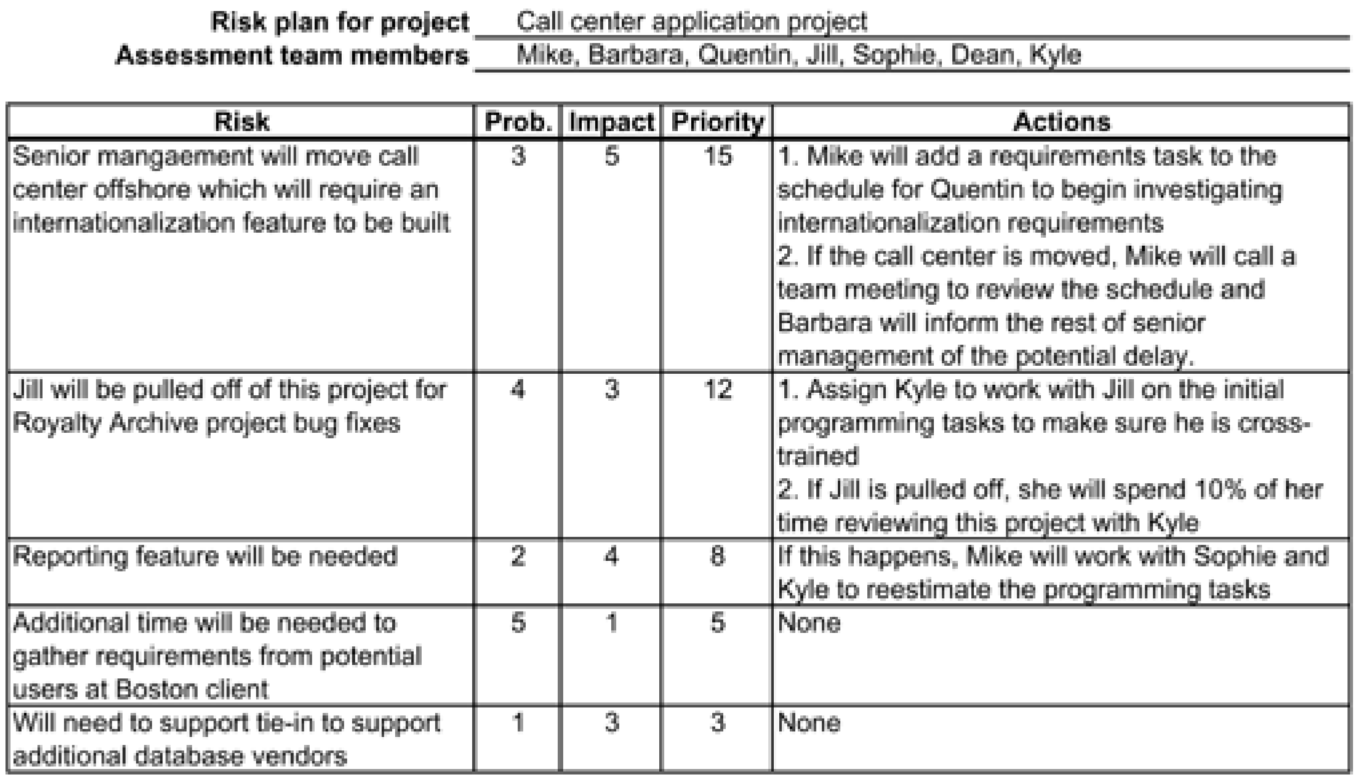
- A risk plan is a list of all risks that threaten the project, along with a plan to mitigate some or all of those risks.

- Building a risk plan
  1. Brainstorming of potential risks
  2. Estimate the impact of each risk
  3. Make a mitigation plan

Source: Stellmann, Greene (2006)

mobile business

Example of a Risk Plan



Risk plan for project Call center application project

Assessment team members Mike, Barbara, Quentin, Jill, Sophie, Dean, Kyle

Risk	Prob.	Impact	Priority	Actions
Senior mangaement will move call center offshore which will require an internationalization feature to be built	3	5	15	1. Mike will add a requirements task to the schedule for Quentin to begin investigating internationalization requirements 2. If the call center is moved, Mike will call a team meeting to review the schedule and Barbara will inform the rest of senior management of the potential delay.
Jill will be pulled off of this project for Royalty Archive project bug fixes	4	3	12	1. Assign Kyle to work with Jill on the initial programming tasks to make sure he is cross-trained 2. If Jill is pulled off, she will spend 10% of her time reviewing this project with Kyle
Reporting feature will be needed	2	4	8	If this happens, Mike will work with Sophie and Kyle to reestimate the programming tasks
Additional time will be needed to gather requirements from potential users at Boston client	5	1	5	None
Will need to support tie-in to support additional database vendors	1	3	3	None

Source: Stellmann, Greene (2006)

mobile business

Agenda

- Introduction to Software Engineering
- Software Engineering Process Overview
- Software Development Process Models

Diversity of Software Engineering Approaches

- There are many different types of software and there is no universal set of SE methods which is applicable to all of these.
- The types of Software Engineering methods and tools to be applied depend on
  - the type of application to be developed,
  - the requirements of the customer and
  - the background of the development team.
- Examples for different software projects:
  - Adding new functions to ERP production system
  - Development of a proprietary standard software (e.g. Office suite)
  - Building of a website

Software Engineering Process

- The Software Engineering Process is a structured set of activities to develop a software.

- Many different Software Engineering processes exist, but all of them share the following aspects
  - **Requirements Specification:** Definition of the behaviour of a software
  - **Design and Implementation:** Designing (e.g. modelling) and implementing (e.g. programming/coding) the software
  - **Validation**: Evaluating the features of the software against the specified requirements
  - **Evolution**: Modifying the software in response to changed customer needs.

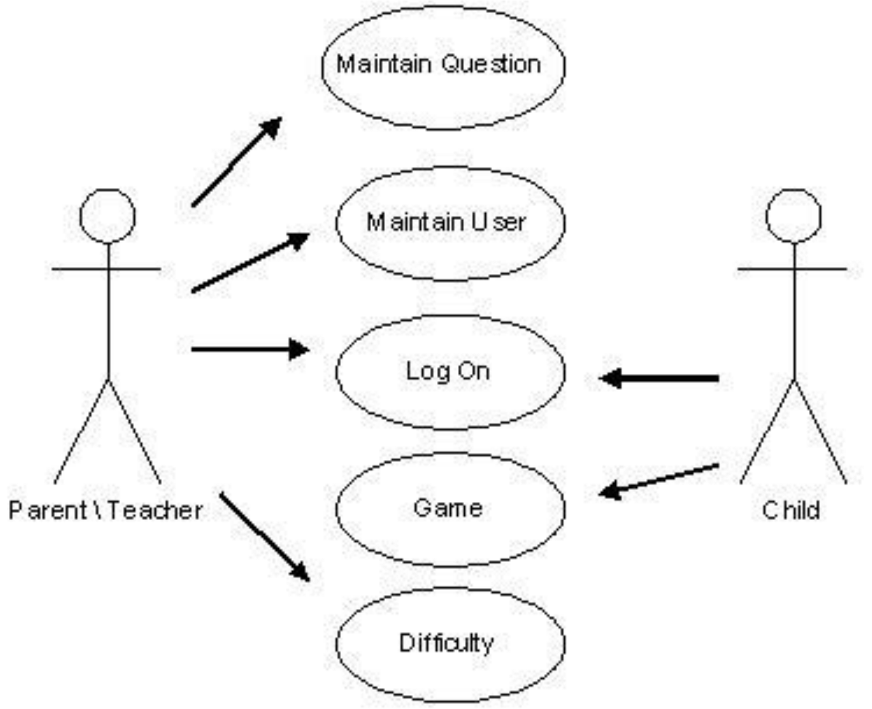
Requirements Specification (1): Software Requirements

- **Software requirements** specify the desired behaviour of a software.
- Requirements analysts (or business analysts) generate software requirements specifications through *requirements elicitation*.
  - Interviews with the users, stakeholders and anyone else whose perspective needs to be taken into account
  - Observation of the users at work
  - Prototyping of software
  - ...

  The gathered insights are summarised and send back to the users / stakeholders in order to make sure everybody shares a common understanding about them.
- Software requirements should be documented in a **Software Requirements Specification**, which complies with the corresponding IEEE Standard.

Requirements Specification (2): Use Cases

- A use case is a description of a specific interaction that a user may have with a software.
- Use cases are simple means for describing the functionality of a software.
- Use cases do not describe any internal workings of the software, nor do they explain how the software is going to be implemented.



Source: WikiCommons (2011)

mobile business

Requirements Specification (3): Functional vs. Non-Functional Requirements

- **Functional requirements** define the explicitly perceptible behaviour of a software.
  - Login,
  - Calculations,
  - Configuration Options,
  - Features (e.g. display of customer information)
  - ...
- **Non-functional requirements** define characteristics of a software, which do not affect its behaviour (software quality attributes).
  - Usability
  - Performance
  - Error handling
  - ...

Design & Implementation

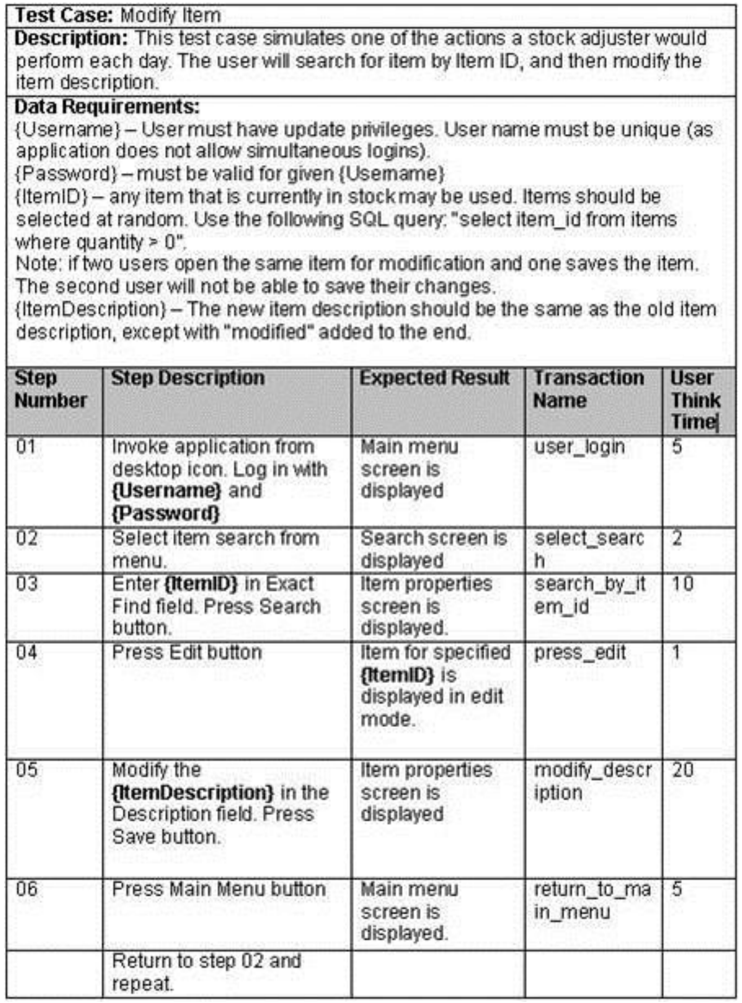
- **Vision and Scope** documents the needs of an organisation
- **Requirements** specify the required behaviour of software in order to satisfy those needs
- **Design** specifies how software requirements are to be technically **implemented**

Validation (1)

- A **test case** specifies a user test in order to evaluate a specific software behaviour.
- Test cases are very similar to use cases as they provide step-by-step instructions for the interaction between the user and the software.
- A **test plan** is an organised list of all required test cases to run through in order to evaluation the functionality of a software against its specified requirements.

Validation (2)

- A typical test case is outlined in a table, and includes:
  - A unique *name* and *number*
  - A short *description of the test case*
  - *Preconditions* which describe the state of the software before the test case
  - *Steps* that which make up the interaction during the test
  - *Expected Results*, which describe the expected state of the software after the test case was run through



Test Case: Modify Item

Description: This test case simulates one of the actions a stock adjuster would perform each day. The user will search for item by Item ID, and then modify the item description.

Data Requirements:
{Username} – User must have update privileges. User name must be unique (as application does not allow simultaneous logins).
{Password} – must be valid for given {Username}
{ItemID} – any item that is currently in stock may be used. Items should be selected at random. Use the following SQL query: "select item_id from items where quantity > 0".
Note: if two users open the same item for modification and one saves the item. The second user will not be able to save their changes.
{ItemDescription} – The new item description should be the same as the old item description, except with "modified" added to the end.

Step Number	Step Description	Expected Result	Transaction Name	User Think Time
01	Invoke application from desktop icon. Log in with **{Username}** and **{Password}**	Main menu screen is displayed	user_login	5
02	Select item search from menu.	Search screen is displayed	select_search	2
03	Enter **{ItemID}** in Exact Find field. Press Search button.	Item properties screen is displayed.	search_by_item_id	10
04	Press Edit button	Item for specified **{ItemID}** is displayed in edit mode.	press_edit	1
05	Modify the **{ItemDescription}** in the Description field. Press Save button.	Item properties screen is displayed	modify_description	20
06	Press Main Menu button	Main menu screen is displayed.	return_to_main_menu	5
	Return to step 02 and repeat.			

Evolution

- Change control is a method for implementing only those changes that are worth pursuing while preventing unnecessary or overly costly changes from derailing the project.

- Establishing a Change Control Board
  - Project manager
  - Important stakeholders
  - Designers, programmers, testers
  - …

- Change Control Board decides which of the requested changes are actually going to be implemented.

Example: Software Development Process for the NATO Interoperable Submarine Broadcast System (NISBS)

- Requirements Specification:

“The primary mission of the NISBS is to provide the U.S. with a NATO-interoperable message preparation, management, format and transmit capability. The NISBS shall be capable of […]”

- Design and Implementation:

„Prepare Source Code Record (SCR) and Executable Object Code Record (EOCR) in accordance with 4.2 […]“

- Validation:

“Evaluate test plans and tests against criteria: traceability to requirements, […] appropriateness of test standards and methods used […]“

- Evolution:

“[…] Assist with retirement or replacement of the system as needed.”

NOTE: In a software development plan each step is described in extensive detail!

Source: Space and Naval Warfare Systems Center (ed.) (1999)

Agenda

- Introduction to Software Engineering
- Software Engineering Process Overview
- Software Development Process Models

Plan-driven vs. Agile Software Development

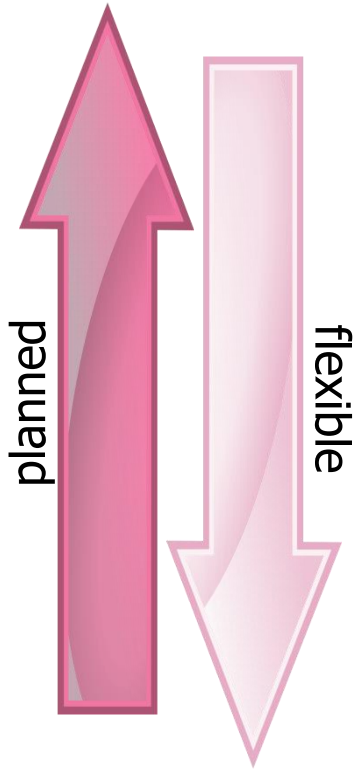
- Plan-driven SD consists of processes in which all activities have been planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.

Software Development Process Models

- Describe the development process by defining the process steps and results
- Define principles, methods and tools for the development process
- Determine chronological sequence for planning, development and implementation of projects
- Are available in a wide variety of approaches

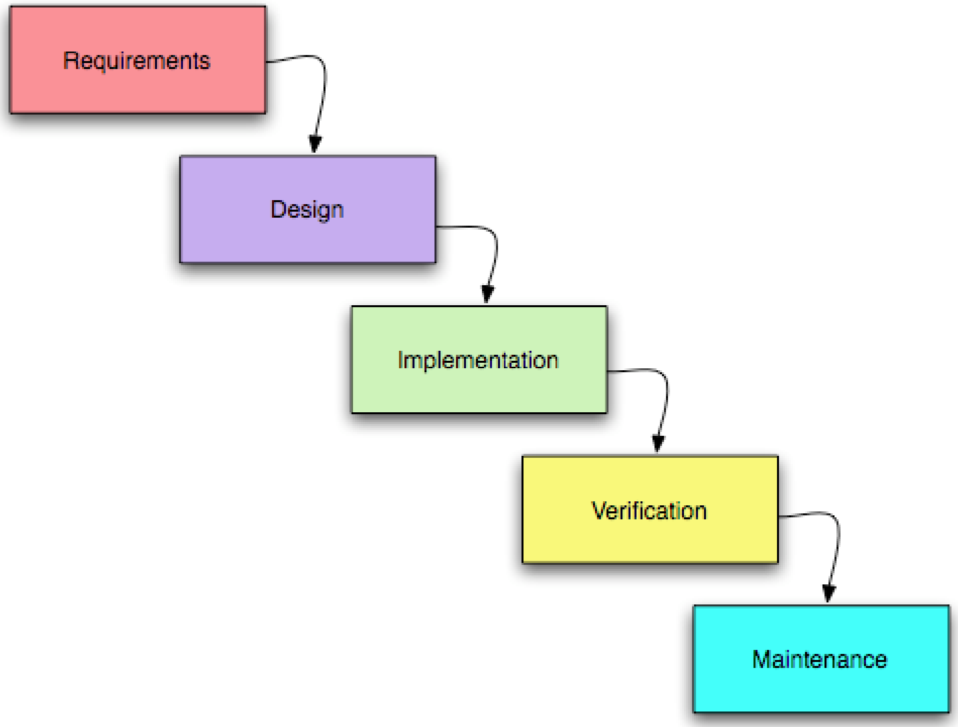
Classification of Process Models

- Sequential model
  - Consecutive phases with an increasing granularity and milestones as results of phases
- Modified sequential models
  - Phases are interleaved with an increasing granularity and milestones as results of phases
- Evolutionary models
  - No phases with defined results. Instead iterative cycles of “design, implementation and validation”
- Agile models
  - Only a general framework for an approach, few rules, very flexible, dynamic phases

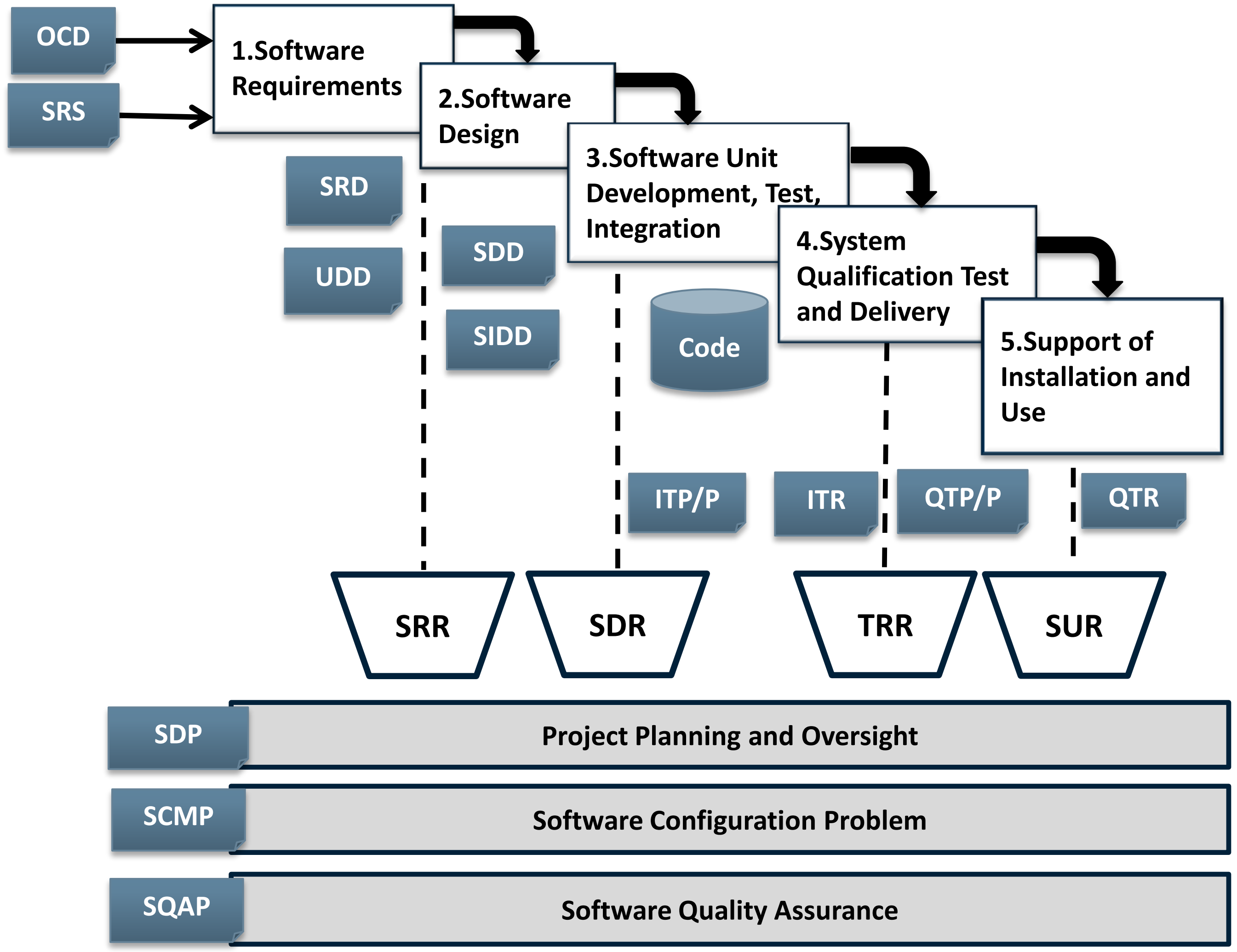


Sequential Models

- The waterfall model (first described by Royce in 1970)
- There seem to be at least as many versions as there are authors - perhaps more



Example of a Sequential Model: NATO Interoperable Submarine Broadcast System (NISBS)


OCD
SRS
1.Software Requirements
2.Software Design
3.Software Unit Development, Test, Integration
4.System Qualification Test and Delivery
5.Support of Installation and Use
SRD
UDD
SDD
SIDD
Code
ITP/P
ITR
QTP/P
QTR
SRR
SDR
TRR
SUR
SDP
Project Planning and Oversight
SCMP
Software Configuration Problem
SQAP
Software Quality Assurance

Source: Space and Naval Warfare Systems Center (ed.) (1999)

OCD: Operational Concept Document
SRS: System Requirements Specification
SRD: Software Requirements Description
UDD: User Documentation Description
SRR: System Requirements Review
SDD: Software Design Description
SIDD: Software Interface Design Description
SDR: Software Design Review
ITP/P: Integration Test Plan/Procedures
ITR: Integration Test Report
TRR: Test Readiness Review
QTP/P: Qualification Test Plan/Procedures
QTR: Qualification Test Report
SUR: Software Usability Review
SDP: Software Development Plan
SCMP: Software Configuration Management Plan
SQAP: Software Quality Assurance Plan

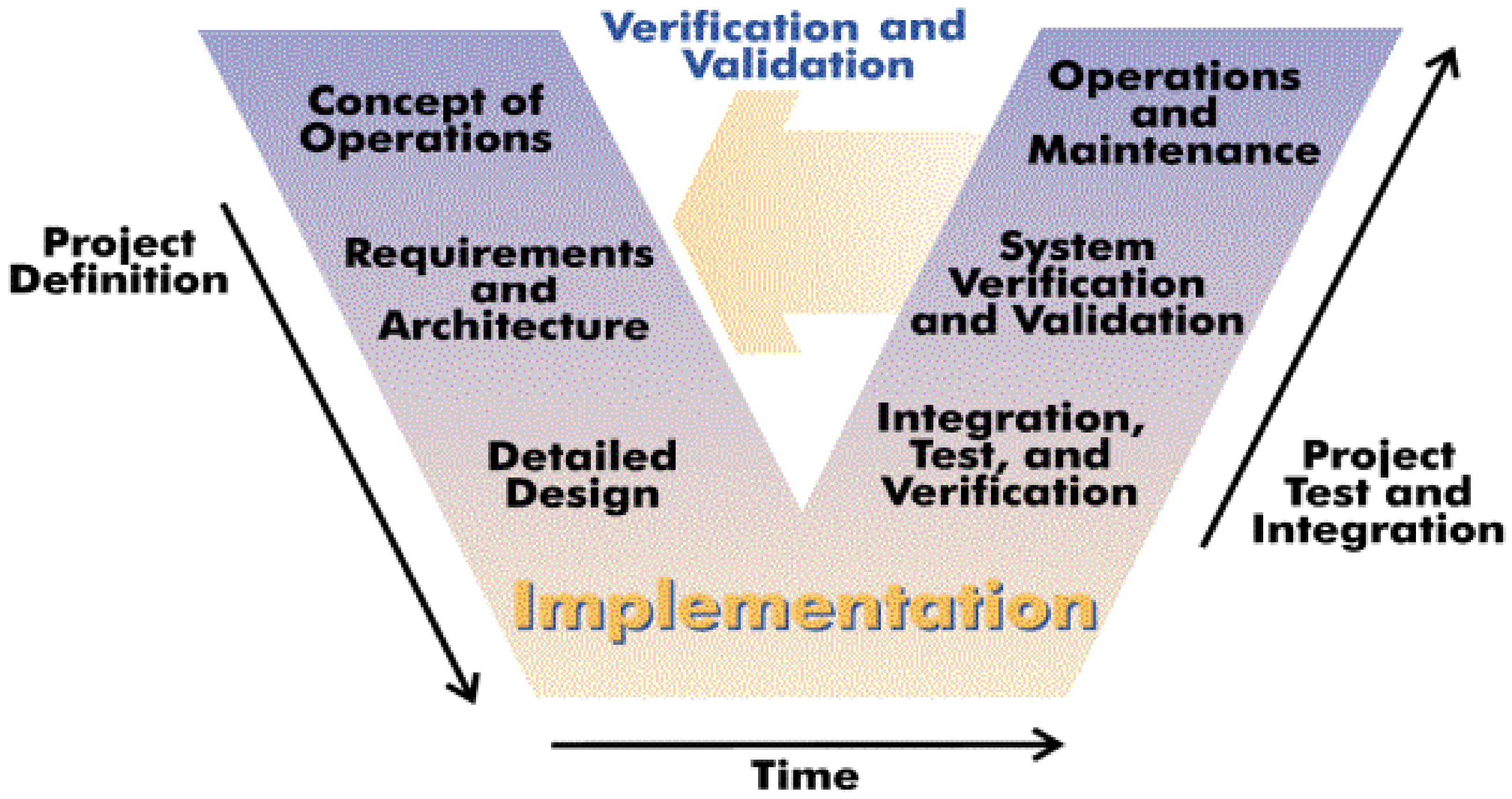
Sequential Models

- One or more documents are produced after each phase and on which one has to “sign off”.

- Aspects worth mentioning:
  - “Water does not flow up” → It is difficult to change an artifact produced in the previous phase.
  - Approach should only be used if requirements are clear and well understood.
  - Reflects traditional engineering practice
  - Simple management approach

mobile business

V-Model


Verification and Validation
Concept of Operations
Requirements and Architecture
Detailed Design
Implementation
Integration, Test, and Verification
System Verification and Validation
Operations and Maintenance
Project Definition
Project Test and Integration
Time

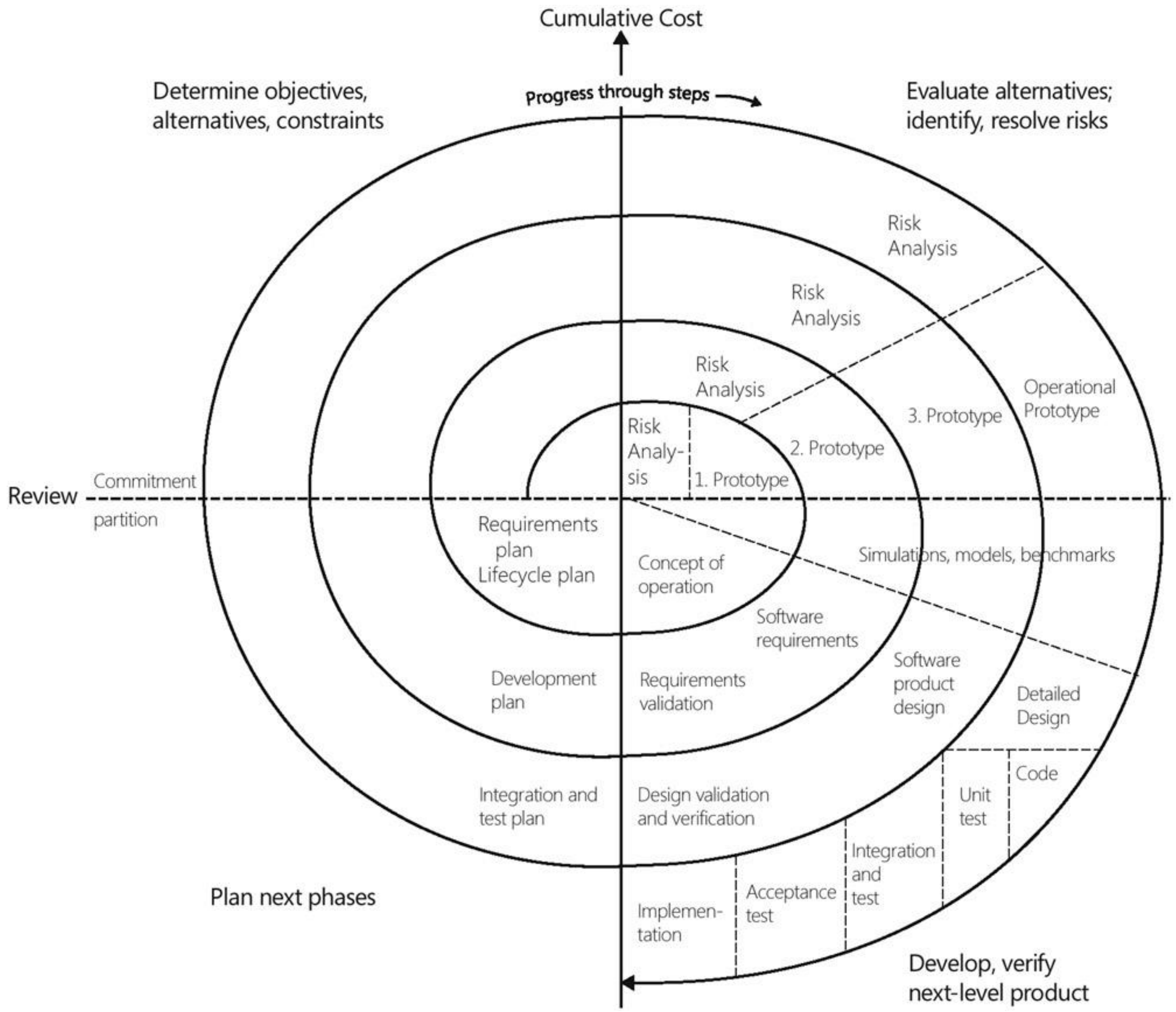
- Horizontal lines denote the information flow between activities at the same abstraction level.
- First proposal in 1979

Source: Clarus (2005)

V-Model

- Similar to pure waterfall model, but makes the dependencies between development and verification activities explicit.
- The left half of the “V” represents *development* and the right half system *validation*.
- Note the requirements specification includes requirements elicitation and analysis.

Evolutionary Models Example: Spiral Model



Source: Marciniak (2002)

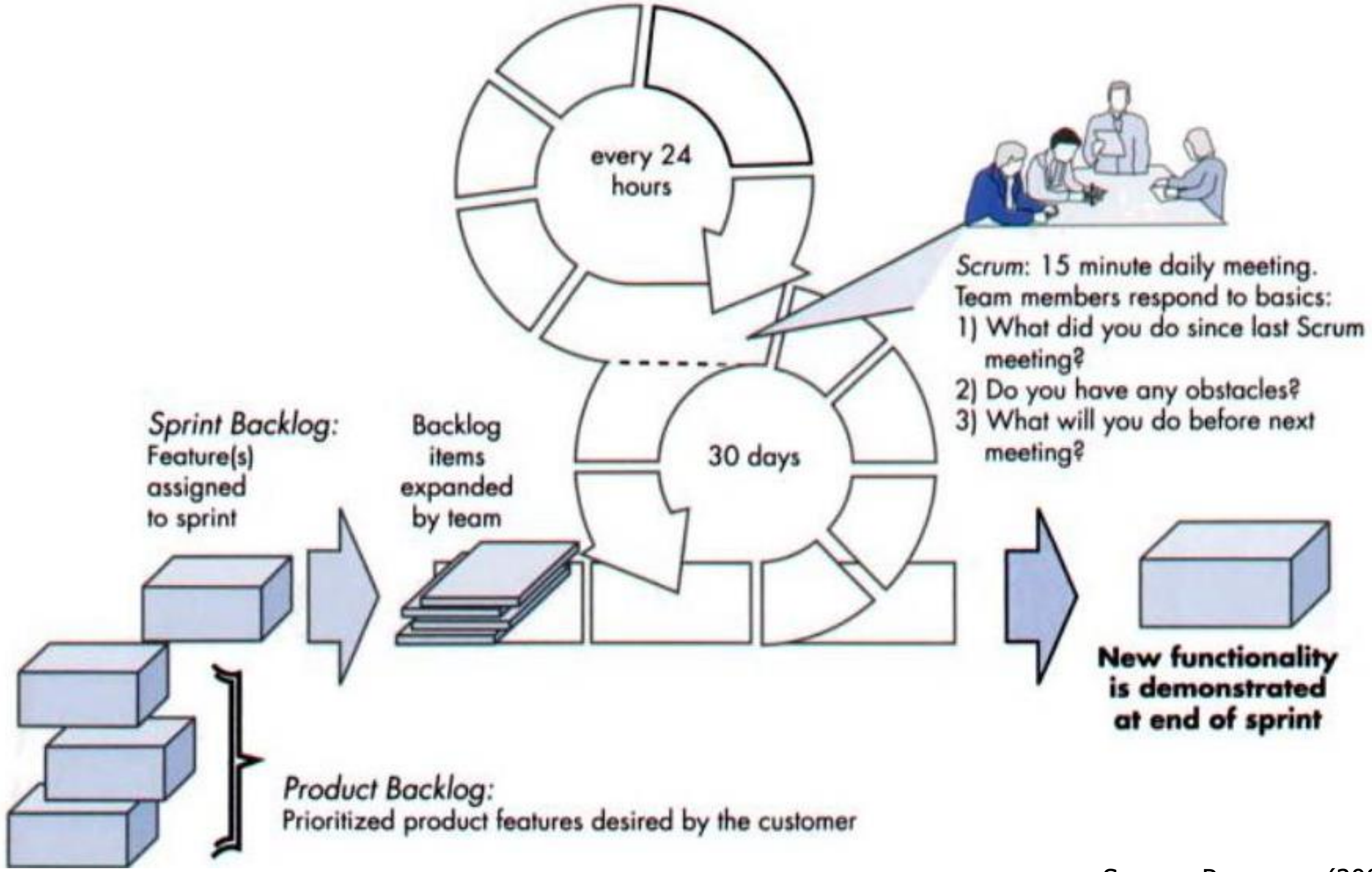
Evolutionary Model Example: Spiral Model

- Basic Concept
  - Develop an initial implementation, demonstrate it to user, get feedback and refine it until an adequate system has been produced.
- Two types of evolution models:
  - *Exploratory*
  - *Throw-away prototyping*
- Advantages
  - Estimates for budget, schedule, etc. become more realistic as work progresses
- Disadvantages
  - Requires expertise in risk evaluation and mitigation
  - Appropriate only for large systems

Agile Models

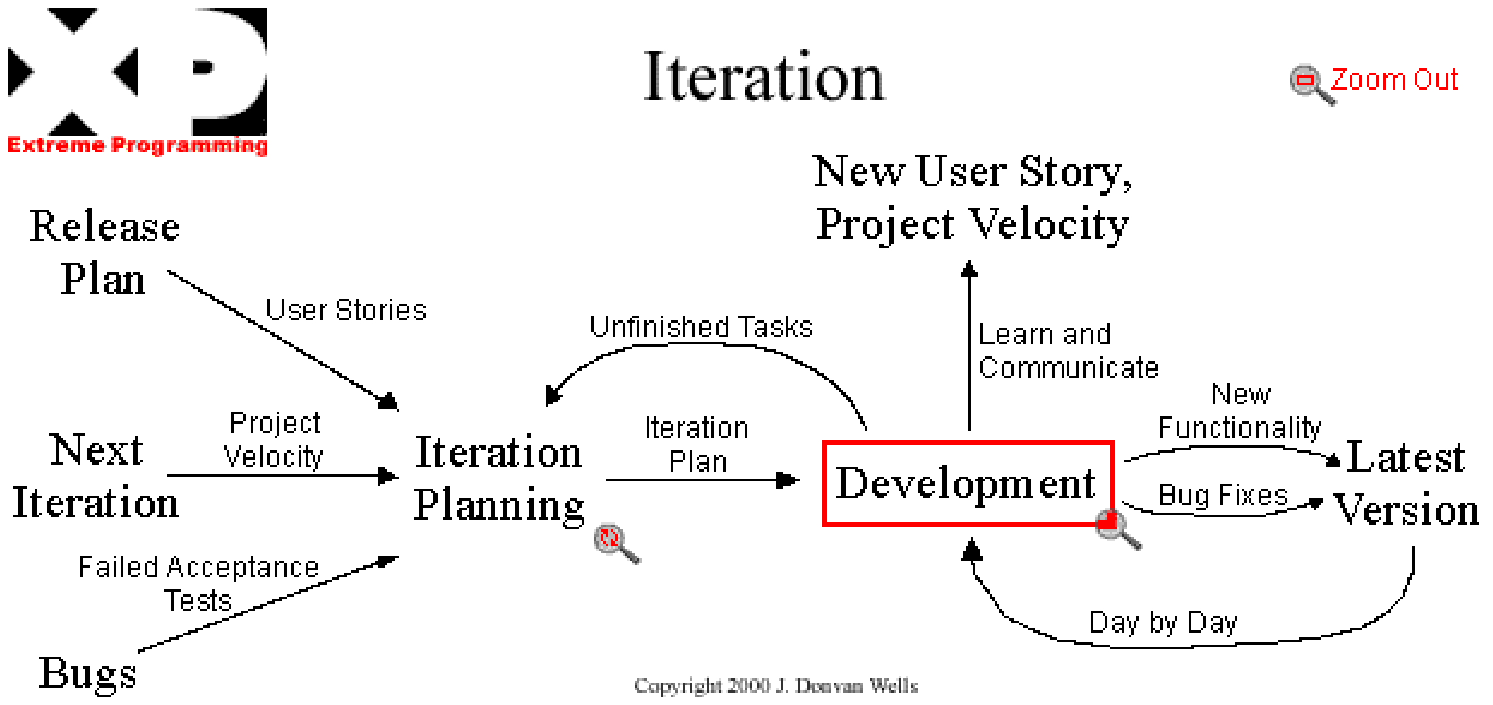
- Characteristics
  - Only a general development framework
  - Strong integration and interaction with the customer during the development process
  - Short development cycles (e.g. 6-8 weeks)
  - Continuous change of project specifications / requirements
  - Direct and informal communication between the project participants
  - Little documentation
  - Requires a lot of discipline of all participants
  - Examples: eXtreme Programming, SCRUM, Feature-Driven Development
- To be applied under the following circumstances:
  - Specifications are uncertain and subject to continuous change
  - Innovative projects

Agile Models Example: Scrum


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

Source: Pressman (2005)

Agile Models Example: Extreme Programming



Source: www.extremeprogramming.org, 2011

Literature

- Clarus (2005) “Concept of Operations”, Federal Highway Administration (FHWA), Publication No. FHWA-JPO-05-072, 2005.
- Extremprogramming (2011) http://www.extremeprogramming.org
- Marciniak J. (ed.) (2002) ”Encyclopedia of Software Engineering”, 2nd. Edition, 993-1005, Wiley, 2002.
- Pressman R, (2005) “Software Engineering: A Practitioner's Approach”, Mcgraw Hill Book, 6th edition, 2005
- Project Cartoon (2011) http://www.projectcartoon.com
- Royce, W. (1970) "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26 (August): 1–9.
- Sommerville I. (2007) „Software Engineering“, Pearson Studium, 8th edition, 2007.
- Stellmann, A.; Greene, J. (2011) “Applied Software Project Management“, O‘Reilly Media Inc 2006.
- Space and Naval Warfare Systems Center (ed.) (1999) „Software Development Plan (SDP) for the NATO Interoperable Submarine Broadcast System (NISBS)“, San Diego, 1999.