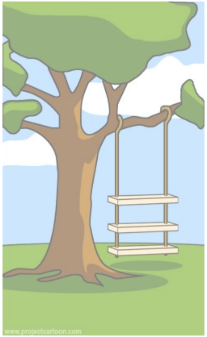
Software Engineering

10. Project Management | Thomas Thüm | February 9, 2022

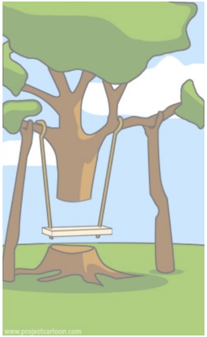
Software Engineering
Programming Languages

ulm university universität ulm

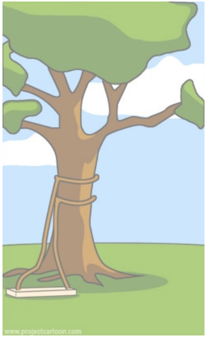
Project Management



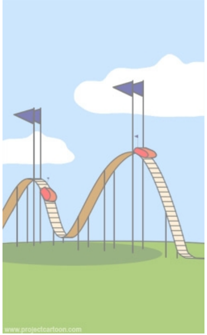
how the customer explained it



how the analyst designed it



how the programmer implemented it



how the customer was billed

Lecture Overview

1. Introduction to Project Management
2. Project Planning and Scheduling
3. Summary on Software Engineering I

Lecture Contents

1. Introduction to Project Management
   - Software Development Project
   - Project Management
   - Activities in Project Management
   - Risk Management
   - People Management
   - Lessons Learned
2. Project Planning and Scheduling
3. Summary on Software Engineering I

Software Development Project [Ludewig and Lichter]

Software Development Project

- aka. software engineering project
- temporary activity with start and end date
- has goals
  - creation / modification of a software product
  - creation / modification of components for future projects
  - gain experience / knowledge
  - capacity utilization (Mitarbeiterauslastung)
  - . . .
- is successful if goals are largely fulfilled

Project Management [Sommerville]

Motivation

“Good management cannot guarantee project success. However, bad management usually results in project failure: The software may be delivered late, cost more than originally estimated, or fail to meet the expectations of customers.”

Goals of Project Management

- “deliver the software to the customer at the agreed **time**
- keep overall **costs** within budget
- deliver software that meets the customer’s **expectations**
- maintain a coherent and well-functioning development **team**”

Project Management Depends on . . .

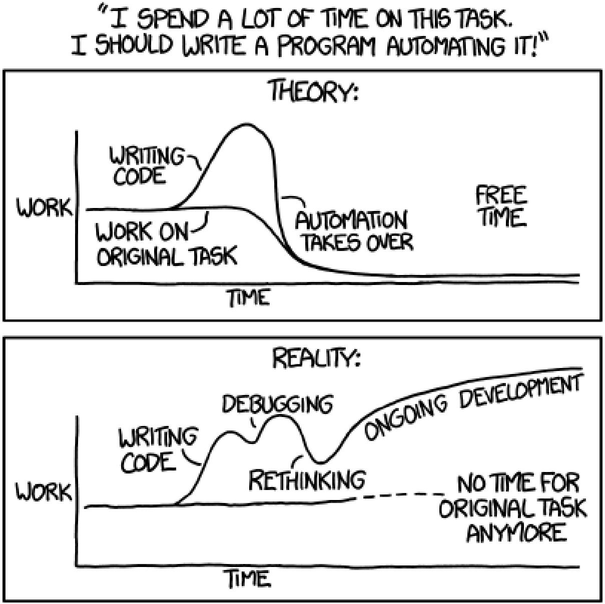
company size large companies have management hierarchies and reporting / budgeting / approval processes

customers external customers (i.e., government agencies) usually have policies

software size large systems require multiple development teams in different companies / locations

software type safety-critical systems require all design decisions to be documented

dev. process project management heavily depends on process model


"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"
THEORY:
WORK
TIME
WRITING CODE
WORK ON ORIGINAL TASK
AUTOMATION TAKES OVER
FREE TIME
REALITY:
WORK
TIME
WRITING CODE
DEBUGGING
RETHINKING
ONGOING DEVELOPMENT
NO TIME FOR ORIGINAL TASK ANYMORE

Activities in Project Management [Sommerville]

Project Planning

“Project managers are responsible for **planning, estimating, and scheduling** project development and assigning people to tasks. They supervise the work to ensure that it is carried out to the required standards, and they **monitor progress** to check that the development is on time and within budget.”

Risk Management

“Project managers have to **assess the risks** that may affect a project, monitor these risks, and take action when problems arise.”

People Management

“Project managers are responsible for **managing a team** of people. They have to choose people for their team and establish ways of working that lead to effective team performance.”

Reporting

“Project managers are usually responsible for **reporting on the progress** of a project to customers and to the managers of the company developing the software. They have to be able to communicate at a range of levels, from detailed technical information to management summaries.”

Proposal Writing

“The first stage in a software project may involve writing a proposal to **win a contract** to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out. It usually includes **cost and schedule estimates** and justifies why the project contract should be awarded to a particular organization or team. Proposal writing is a critical task as the survival of many software companies depends on having enough proposals accepted and contracts awarded.”

Risk Management [Sommerville]

Risk

Probability insignificant, low, moderate, high, very high

Severity insignificant, tolerable, serious, catastrophic

Classification of Risks

Project Risks affect project schedule or resources: loss of an experienced system architect may result in longer development time

Product Risks affect software quality: purchased component may not scale

Business Risks affect organization / company: product of a competitor may reduce number of sales

Stages in Risks Management

1. **Risk Identification** identify possible project, product, and business risks
2. **Risk Analysis** assess likelihood and consequences
3. **Risk Planning** plan how to address risks: avoidance or minimization of effects
4. **Risk Monitoring** regularly assess risks and revise plans if needed

Risks in Agile Development

reduced risks for requirements changes, increased risks for loss of stuff due to fewer documentation

People Management [Sommerville]

Motivation

"The people working in a software organization are its **greatest assets**. It is expensive to recruit and retain good people, and it is up to software managers to ensure that the engineers working on a project are as **productive** as possible. In successful companies and economies, this productivity is achieved when people are respected by the organization and are assigned responsibilities that reflect their skills and experience."

In Practice

"Software engineers often have strong **technical skills** but may lack the softer skills that enable them to **motivate and lead a project development team**."

Critical Factors

1. **Consistency** treat people comparably with similar rewards
2. **Respect** let all people contribute and respect their differences in skills
3. **Inclusion** consider views of least experienced peoples
4. **Honesty** manager is honest about own skills and team performance

Teamwork

"Most professional software is developed by project teams that range in size from two to several hundred people. However, as it is impossible for everyone in a large group to work together on a single problem, **large teams are usually split** into a number of smaller groups. Each group is responsible for developing part of the overall system."

Introduction to Project Management

Lessons Learned

- Software development projects
- Project management: goals, influences, activities
- Risk and people management
- Further Reading: Sommerville, Chapter 22 Project Management and Ludewig and Lichter, Chapter 7.2 (Software-Projekte)

Practice

- See Moodle
- Risk identification and analysis: Give an example for a risk of a messenger app in Moodle (2–3 sentences) and specify probability, severity, and classification
- Risk planning and monitoring: How to address the risk mentioned by one of your colleagues? What could change during the project?

Lecture Contents

1. Introduction to Project Management
2. Project Planning and Scheduling
   - Project Planning
   - Gantt Chart
   - Network Diagram
   - Gantt Charts vs Network Diagrams
   - Lessons Learned
3. Summary on Software Engineering I

Project Planning

Anonymous [wikiquote.org]

“A goal without a plan is just a wish.”

Project Planning [Sommerville]

At the Proposal Stage

- when bidding for a contract
- enough resources?
- price for the bidding?
- not all requirements known (i.e., system requirements) ⇒ inevitable speculative

Software Pricing

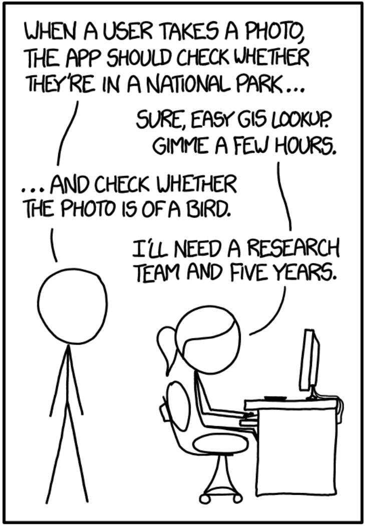
- effort costs (software engineers / managers)
- hardware and software costs (incl. hardware maintenance and software support)
- travel and training costs
- price = estimated costs + profit + contingency (extra effort, 30–50%)

On Project Startup

- who will work on the project?
- how to split into increments?
- refine initial estimates

Throughout the Project

- update plan based on new insights
- learn about the software and team capabilities
- estimates get more accurate

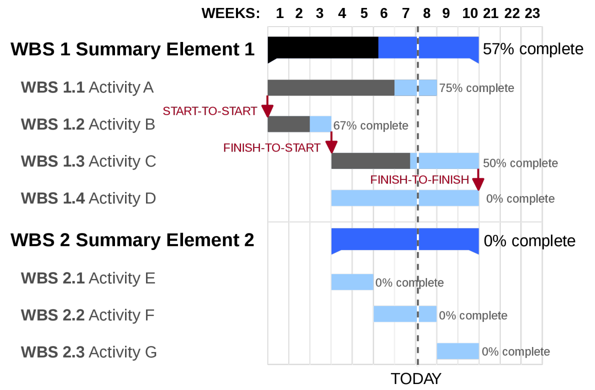

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Gantt Chart [Ludewig and Lichter, Sommerville]

Gantt Chart

- named after Henry L. Gantt (1861–1919)
- bar chart with timeline on x axis and activities on the y axis
- optional: progress bars and marker for observation date
- optional: dependencies between tasks
- optional, not shown: highlight dependencies on the critical path
- **critical path**: tasks whose delay also delays the project



Network Diagram [Ludewig and Lichter, Sommerville]

Network Diagram (Netzplan)

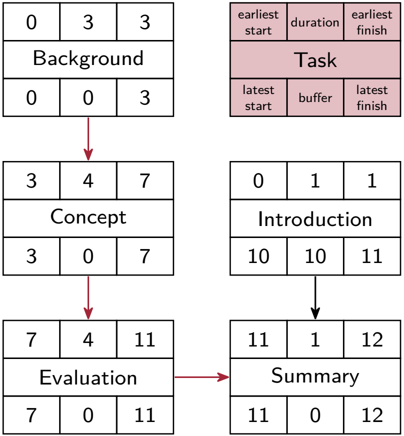
- aka. PERT charts
- directed, acyclic graph
- nodes represent tasks
- edges represent dependencies

Metra Potential Method

Given project start date and **duration** of each activity we can compute:

- **earliest start** and **earliest finish** time with **forward pass**
- **latest start** and **latest finish** time with **backwards pass**
- **buffer** (time span between earliest and latest start/finish)

Example Network for a Bachelor's Thesis



earliest start | duration | earliest finish
Task
latest start | buffer | latest finish

0 | 3 | 3
Background
0 | 0 | 3

3 | 4 | 7
Concept
3 | 0 | 7

0 | 1 | 1
Introduction
10 | 10 | 11

7 | 4 | 11
Evaluation
7 | 0 | 11

11 | 1 | 12
Summary
11 | 0 | 12

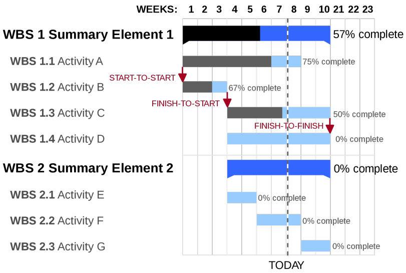
Gantt Charts vs Network Diagrams [Sommerville]

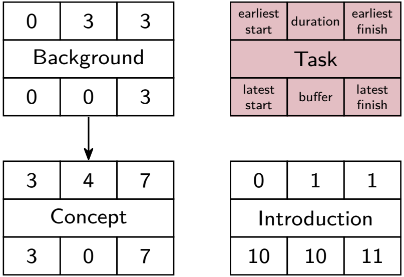
Gantt Chart

- very common technique
- many tools available
- great visualization of timing and progress

Network Diagram (Netzplan)

- clear visualization of dependencies
- explicitly includes buffer times (cf. metra potential method)





0	3	3
Background		
0	0	3

3	4	7
Concept		
3	0	7

earliest start	duration	earliest finish
Task		
latest start	buffer	latest finish

0	1	1
Introduction		
10	10	11

Project Planning and Scheduling

Lessons Learned

- Project planning (incl. software pricing)
- Project scheduling with Gantt charts and network diagrams
- Further Reading: Sommerville, Chapter 23 Project Planning and Ludewig and Lichter, Chapter 8.3.2 (Projektphasen)

Practice

- See Moodle
- Search for a tool to create Gantt charts and use it to schedule the writing of a term paper or bachelor thesis
- Upload your schedule to Moodle and report about your experiences with the tool

Lecture Contents

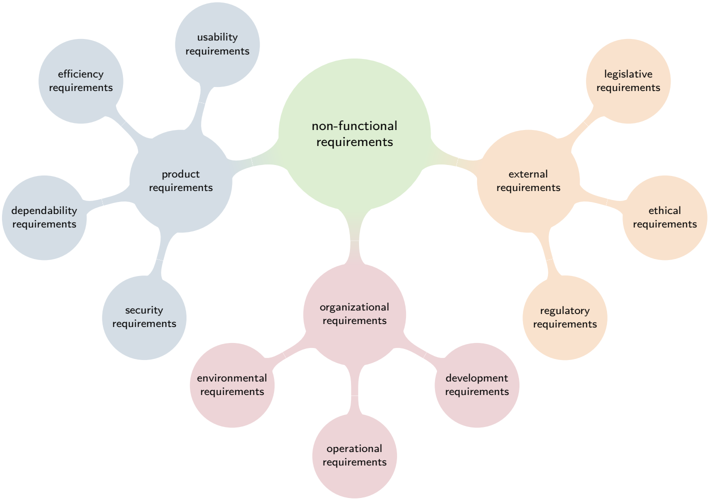
1. Introduction to Project Management

2. Project Planning and Scheduling

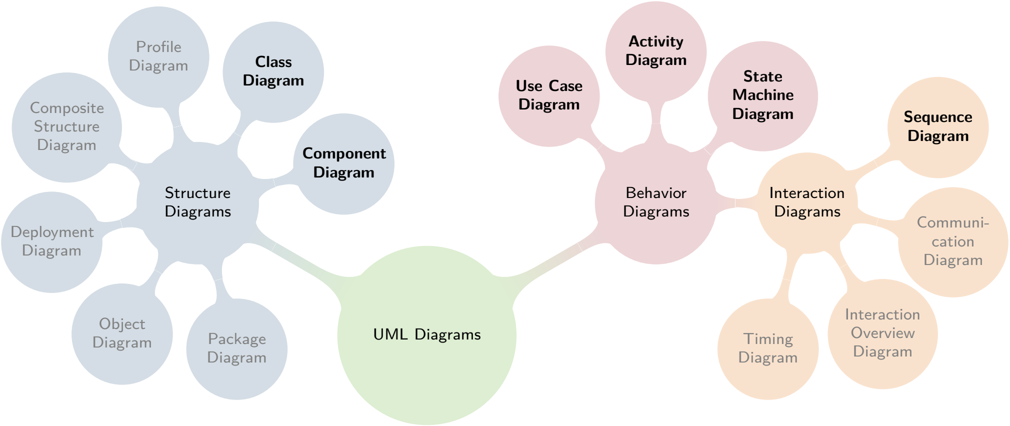
3. Summary on Software Engineering I
   - Recap: Software Engineering vs Programming
   - Recap: Requirements
   - Recap: Modeling with UML Diagrams
   - Recap: Architecture
   - Recap: Implementation
   - Recap: Design Patterns
   - Recap: Quality Assurance
   - Recap: Process Models
   - Recap: Project Management
   - Lessons Learned

Recap: Software Engineering vs Programming

Recap: Requirements



Recap: Modeling with UML Diagrams [UML 2.5.1]



Recap: Architecture

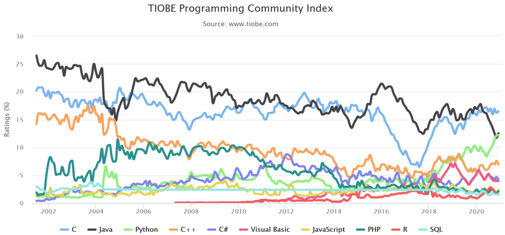
Architectural Pattern (Architekturmuster)

"Architectural patterns capture the essence of an architecture that has been used in different software systems. [...] Architectural patterns are a means of reusing knowledge about generic system architectures." [Sommerville]

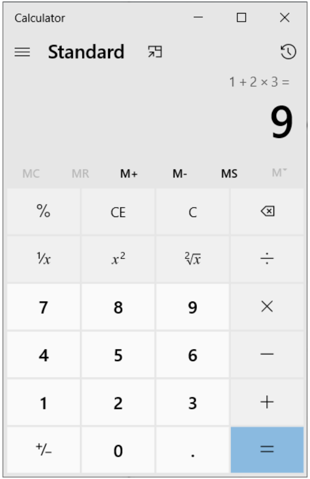
Goals

- preserve knowledge of software architects
- reuse of established architectures
- enable efficient communication

Recap: Implementation

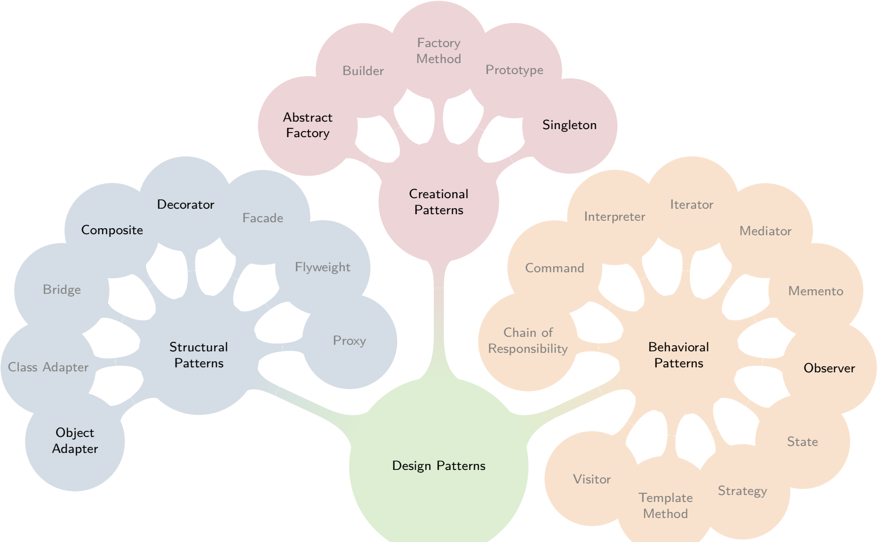

TIOBE Programming Community Index

Source: www.tiobe.com

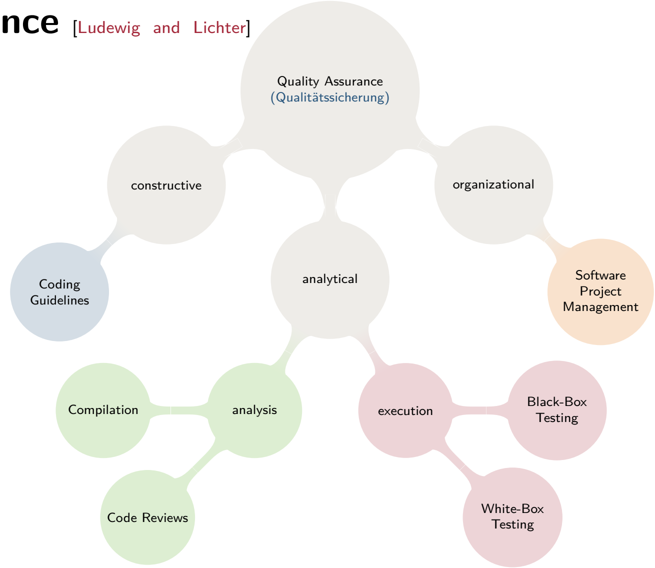

1 + 2 × 3 =

9

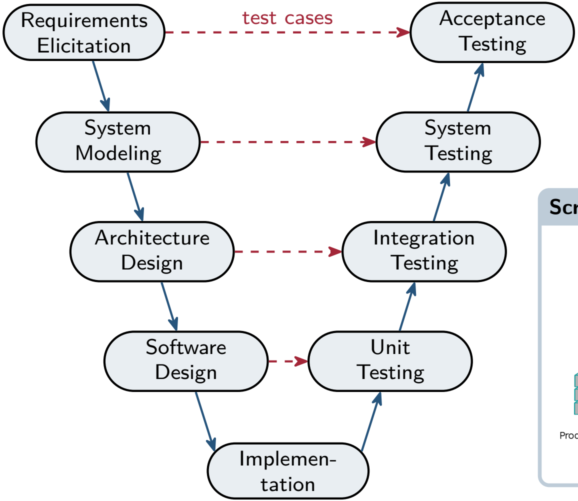
Recap: Design Patterns [Gang of Four (GoF)]



Recap: Quality Assurance [Ludewig and Lichter]



Recap: Process Models



Requirements Elicitation

test cases

Acceptance Testing

System Modeling

System Testing

Architecture Design

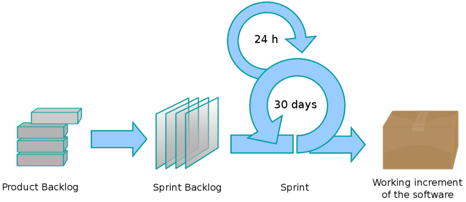
Integration Testing

Software Design

Unit Testing

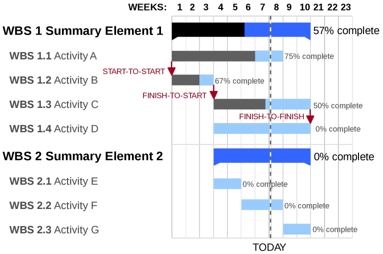
Implemen-tation

Scrum



Recap: Project Management

Gantt Charts

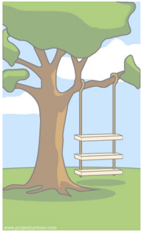


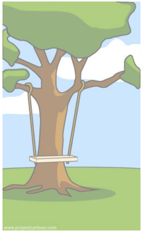
Network Diagrams

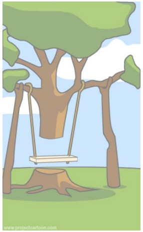
0	3	3
Background		
0	0	3

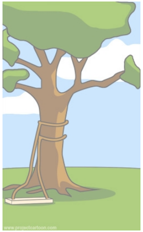
earliest start	duration	earliest finish
Task		
latest start	buffer	latest finish

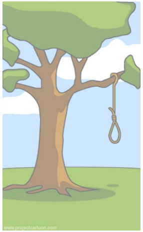
Software Engineering I


requirements

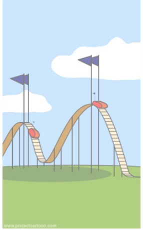

modeling


architecture and design


implementation


testing

process


management and pricing

Software Engineering II

Summary on Software Engineering I

Lessons Learned

1. Software, its impacts, its engineering
2. Requirements, different kinds, its engineering, use case diagrams
3. System modeling, UML, activity/state machine diagrams
4. Software architecture, component diagrams, architectural patterns
5. Software design, class/sequence diagrams
6. Implementation, programming languages, coding conventions, tooling
7. Design patterns, structural (object adapter, composite, decorator), creational (singleton, abstract factory), behavioral (observer)
8. Quality assurance, compilation, code reviews, white-box/black-box testing
9. Process models, Waterfall, V-Model, Scrum
10. Project management, planning, scheduling

Practice

- See Moodle
- Answer the quiz in Moodle to track your learning progress