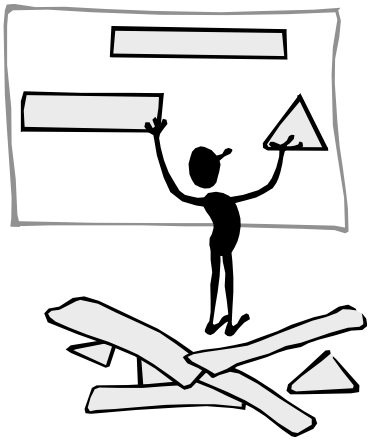
Changing software

Goal:

- Types of maintenance
- Maintenance process
- Experimentation in software engineering
- Design for Maintainability
- Research in Linköping



Definition

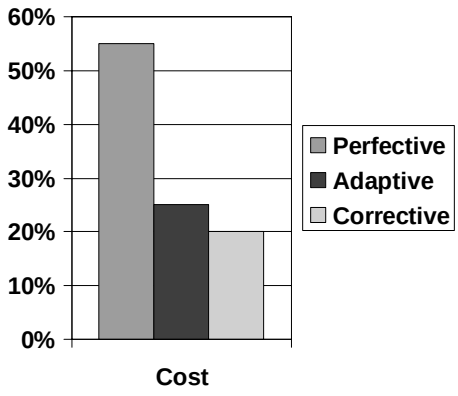
"The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment"

IEEE-STD 1219

- About 70% of all software costs!

Classification

- Corrective maintenance
  - Preventive maintenance
- Adaptive maintenance
- Perfective maintenance



IEEE maintenance model

- Driven by Modification Request (MR)
- Output new baseline (approved product)

- 7 steps:
  - Problem identification
  - Analysis
  - Design
  - Implementation
  - System test
  - Acceptance test
  - Delivery

Problem identification

- Unique identification
- Classification
- Prioritisation
- Decision (accept, reject, further evaluation)
- Scheduling
- Metrics count: number, date of arrival etc.

Analysis

- Feasibility
  - Impact analysis
  - Cost estimation
  - Benefits
- Requirements formulation
- Safety and security impact analysis
- Test strategy
- Plan

Design

- Design
- Verification of requirements
- Test cases
- Update design model

Implementation

- Code
- Unit test
- Update implementation model
- Follow up impact analysis

Testing

- System testing
- Acceptance testing

Maintainability

Boehm et al: Predict maintenance size:

- Size = ASLOC *0.01*
  - Assessment and Assimilation (0-8)
  - Software Understanding (10-50)
  - 0.4 * percentage of changed design
  - 0.3 * percentage of changed code
  - 0.3 * percentage of integrated external code

Effort

- Effort = C1 EAF $(Size)^{P1}$
  - Effort = number of staff months
  - C1 = scaling constant
  - EAF = Effort Adjustment Factor
  - Size = number of delivered, human produced source code instructions (KDSI)
  - P1 = exponent describing the scaling inherent of the process (0.91-1.23)

Metrics

Complexity:

- Cyclomatic number (McCabe(1976)): **V(G) = e – n + 2p**

Modularity:

- avg methods per class / avg lines of code per class

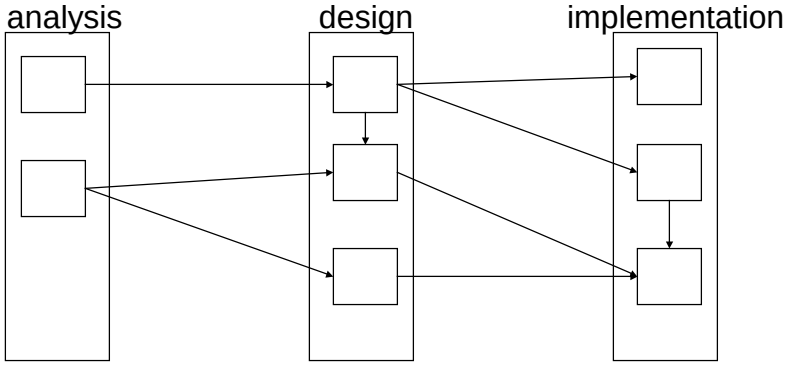
Instrumentation:

- avg number of probe points

Self-descriptiveness

- Readability: 0.295*avg var length+0.499*statement lines+0.13V(G)

Traceability



Design for maintenance

- Configuration management
- Change control
- Identify change-prone properties
  - Factor out parameters
  - Explicitly handle rules and equations
- Low coupling
- High cohesion

Change-prone properties

Instead of:

plot(145,150)

plot(163, 300)

Write:

y:=150

plot(145, y)

plot(163, y*2)

Equation:

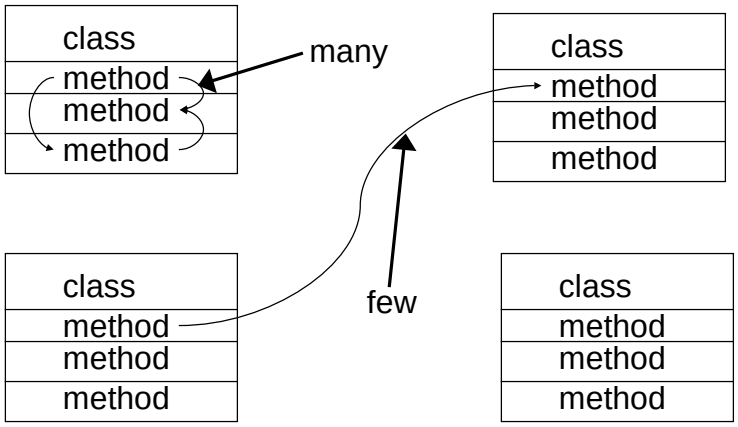
working week = 38.75 h

Rule:

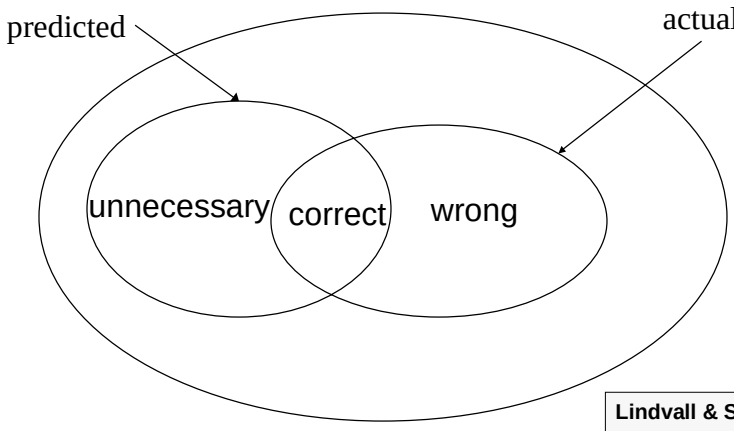
if permanently employed
and more than three
years before
retirement

then offer home-PC

Coupling and cohesion



Research: Impact analysis



Lindvall & Sandahl:

Case-study PMR:

Underprediction factor 1,5 - 6

Research: Tracing system dynamics