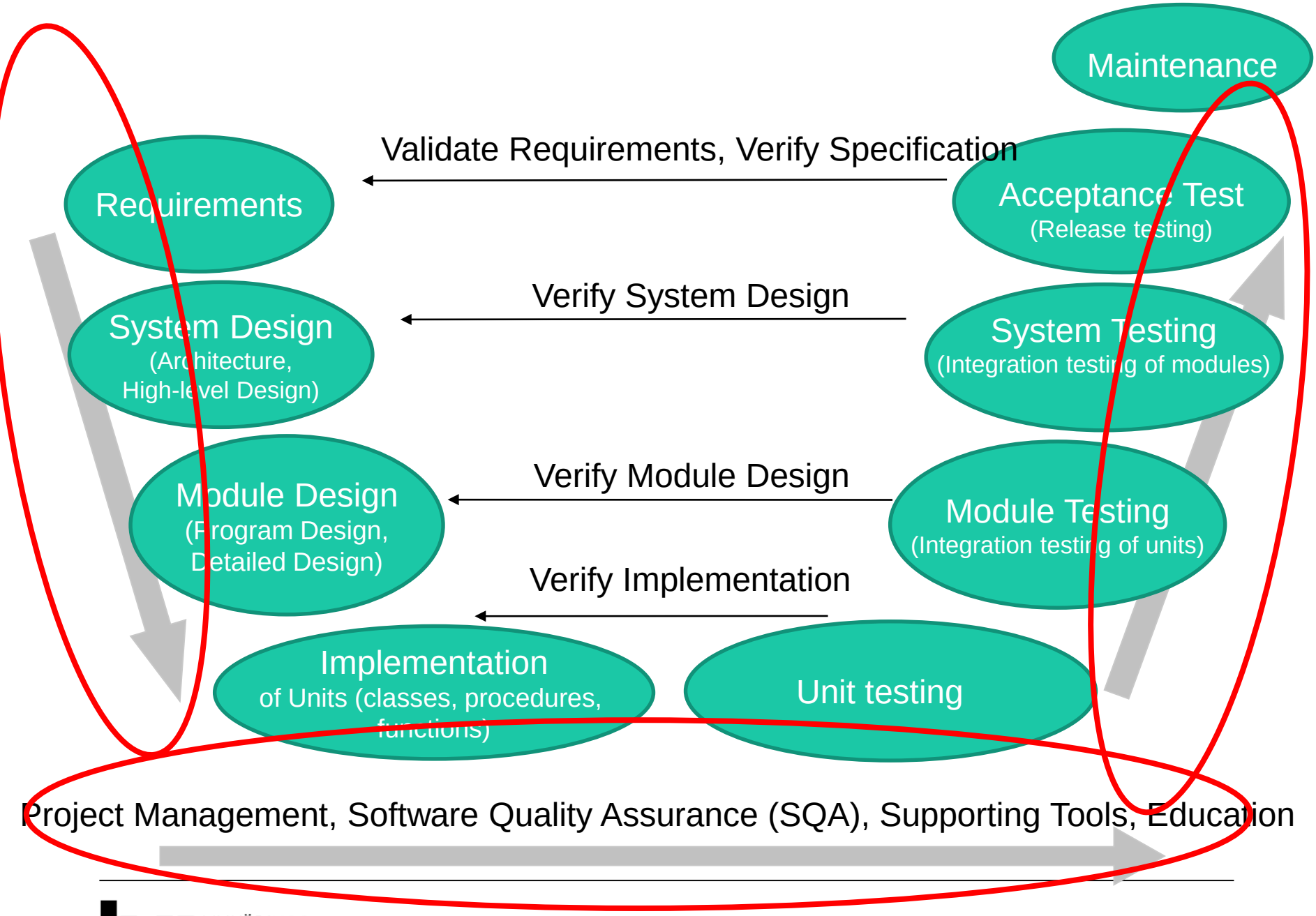
Software Metrics

Kristian Sandahl

LINKÖPING UNIVERSITY

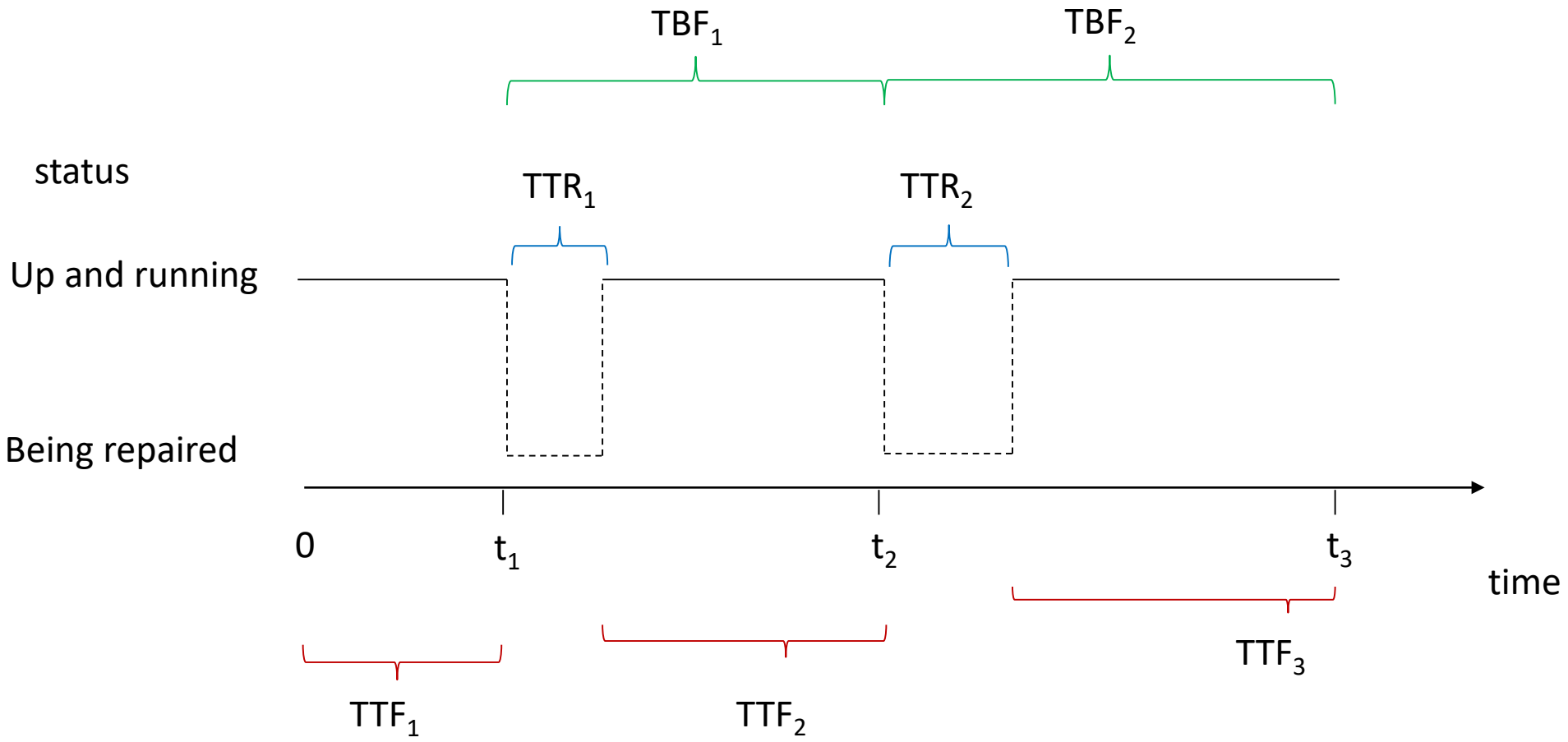

Maintenance
Validate Requirements, Verify Specification
Requirements
Acceptance Test
(Release testing)
Verify System Design
System Design
(Architecture,
High-level Design)
System Testing
(Integration testing of modules)
Verify Module Design
Module Design
(Program Design,
Detailed Design)
Module Testing
(Integration testing of units)
Verify Implementation
Implementation
of Units (classes, procedures,
functions)
Unit testing
Project Management, Software Quality Assurance (SQA), Supporting Tools, Education

Quality factors

- Correctness
- Reliability
- Efficiency
- Usability
- Integrity
- Maintainability
- Flexibility
- Testability
- Security
- Portability
- Reusability
- Interoperability
- Survivability
- Safety
- Manageability
- Supportability
- Replaceability
- Functionality

Measuring these requires both research, experience and imagination.

Simplified model with repair time



Reliability growth model

- The probability that the software executes with no failures during a specified time interval
- MTTF = Mean Time To Failure
- Approximation: MTTF/(1+MTTF)
- Example
- Easier to manage: Failure intensity, [failures / hours of execution time]
- Another approximation: $\lambda = (1-R)/t$
- Example

Similar pattern: Availability and Maintainability

- Measure Mean Time To Repair (MTTR) and Mean Time To Failure (MTTF)
- Availability, A:
- A = MTTF/(MTTF+MTTR)

- Measure Mean Time To Repair (MTTR)
- Maintainability, M:
- M = 1/(1 + MTTR)

Measure usability?

Relevance	Efficiency	Learnability	Attitude
• number of good and bad features recalled by users	• time to complete a task	• ratio of successes to failures (over time)	• percent of favorable/unfavorable user comments
• number of available commands not invoked by users	• percent of task completed	• time spent in errors	• number of good and bad features recalled by users
• number of available commands invoked by users	• percent of task completed per unit time (speed metric)	• percent or number of errors	• number of users preferring the system
• number of times user needs to work around a problem	• time spent in errors	• number of commands used	• number of times user loses control of the system
• percent of task completed	• number of commands used	• frequency of help and documentation use	• number of times the user is disrupted from a work task
	• frequency of help and documentation use	• time spent using help or documentation	
	• time spent using help or documentation	• number of repetitions of failed commands	

Measurement - metrics

Most common use:

- Measurement – directly measured on:
  - Document, no of pages
  - Design, no of model elements
  - Code, no of lines
  - Process, iteration length
  - Quality, avg no of hours to learn a system
- Metrics – is a combination of measurements, eg. number of faults found in test/hours of testing

Computation of cyclomatic complexity

Cyclomatic complexity has a foundation in graph theory and is computed in the following ways:

1. Cyclomatic complexity V(G), for a flow graph, G, is defined as:

$$V(G) = E - N + 2P$$

E: number of edges

N: number of nodes

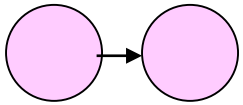
P: number of disconnected parts of the graph

2. Cyclomatic complexity V(G), for a flow graph, G, with only binary decisions, is defined as:

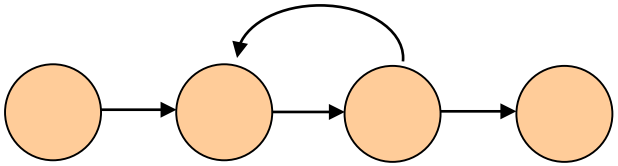
$$V(G) = b + 1$$

b: number of binary decisions

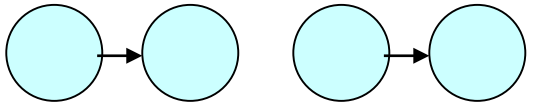
Examples of Graphs and calculation of McCabe's Complexity Metric



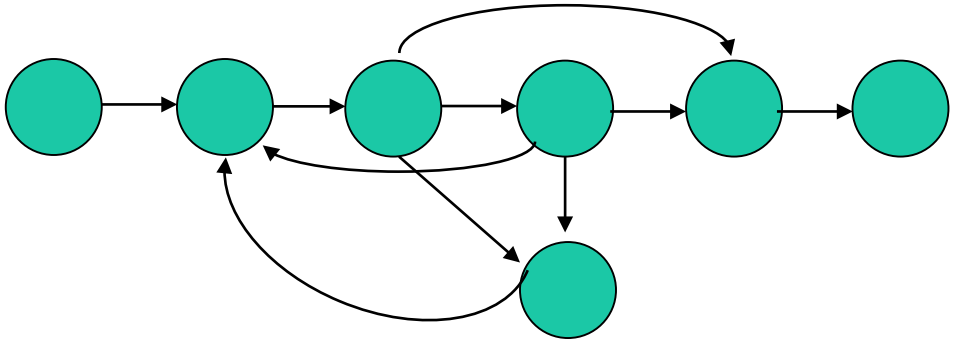
E= 1, N=2 , P=1
V= 1-2+2 = 1



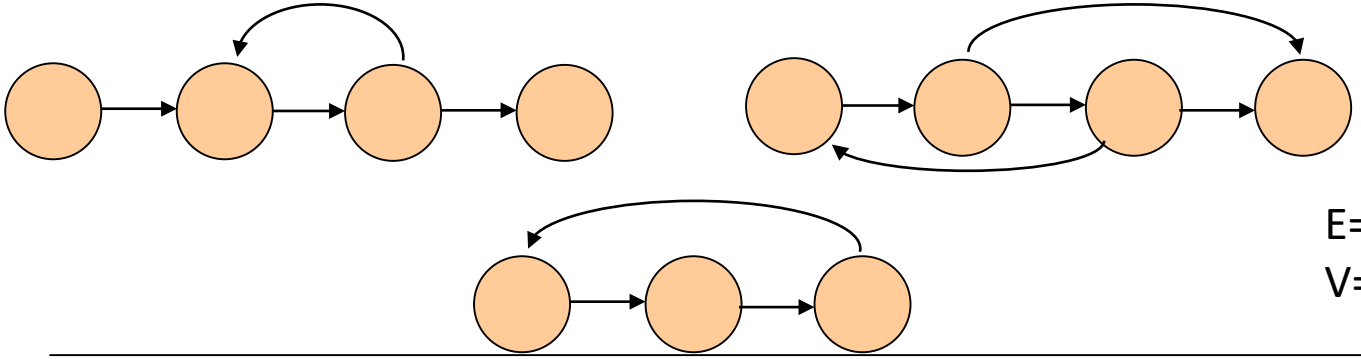
E= 4 , N=4 , P=1
V= 4-4+2 = 2



E= 2 , N= 4 , P=2
V= 2-4+4 = 2

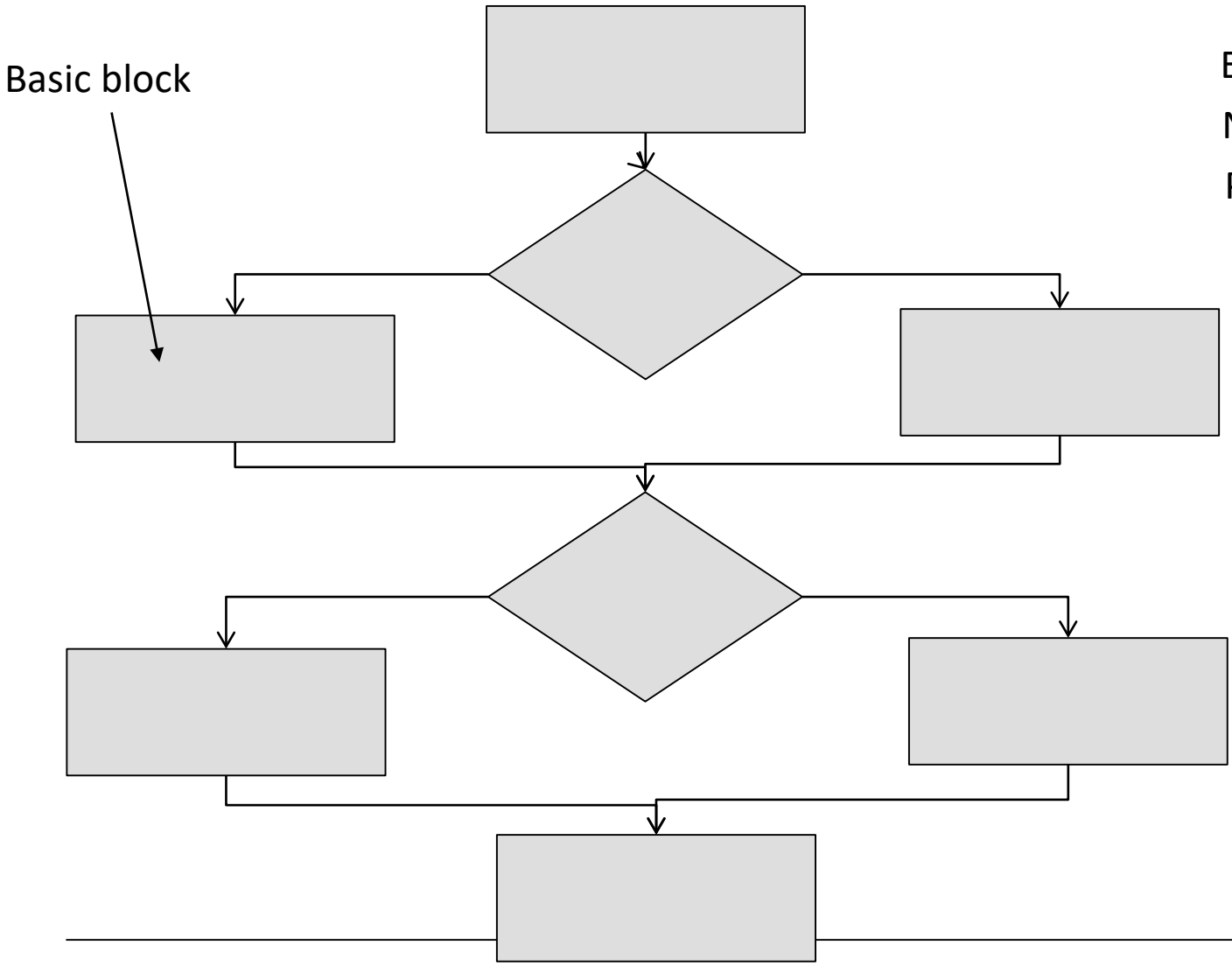


E= 10, N= 7 , P=1
V= 10-7+2 = 5



E= 12, N= 11 , P= 3
V= 12-11+6 = 7

Control-flow



E = 9

N= 8

P= 1

B = 2

V = 3

Software metrics

- Usage-based metrics
- Verification & Validation metrics
- Volume metrics
- Structural metrics
- Effort metrics
- Direct measurement
- Indirect measurement

Note: Pedagogical model only!

Usage based metrics - example

- Description: Number of good and bad features recalled by users.
- How to obtain data: Set up a test scenario. Let test users run the scenario. Collect number of good and bad features in a questionnaire afterwards.
- How to calculate the metric: Take the average of number of good and no. bad features. Two values.
- Relevant quality factor: Relevance – many good and few bad features indicates a good match with the users' mind-set.

Verification and validation metrics - example

- Description: Rate of severe defects found in inspection of design description.
- How to obtain data: Perform an inspection according to your process. Make sure that severity is in the classification scheme.
- How to calculate the metric: Divide the number of defects classified with highest severity with total number of defects in the Inspection record.
- Relevant quality factor: Safety – a high proportion of severe defects in design indicates fundamental problems with the solution and/or competence.

Volume metrics - example

- Description: Number on non-commented lines of code.
- How to obtain data: Count non-commented lines of the code with a tool.
- How to calculate the metric: See above.
- Relevant quality factor: Reliability – it is often hard to understand a large portion of code, the fault density is often higher for large modules.

Structural metrics - example

- Description: Maximum depth of inheritance tree.
- How to obtain data: Count the depth of the inheritance tree for all classes with a tool.
- How to calculate the metric: Take the maximum value of the classes.
- Relevant quality factor: Understandability – It is hard to determine how a change in a higher class will affect inherited/overridden methods.

Effort metrics - example

- Description: Time spent in testing.
- How to obtain data: Make sure that testing activities are distinguished in time reporting forms. Make sure that all project activities are reported.
- How to calculate the metric: Sum the number of hours for all activities in testing for all people involved.
- Relevant quality factor: Testability – a comparably long testing time indicates low testability.

The Goal Question Metric approach

- Outside the written exam we can use a top-down approac: Goal-Question-Metric (GQM)

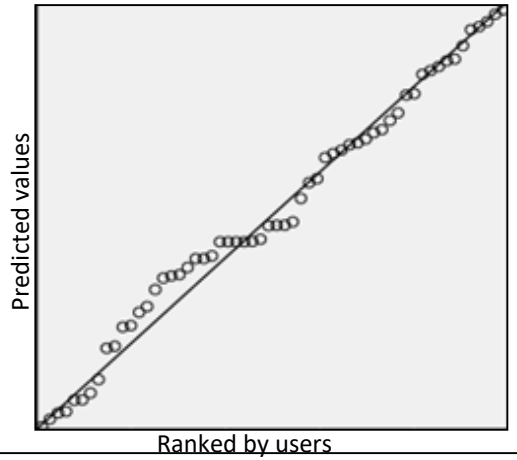
Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question		What is the current change request processing speed?
Metrics		Average cycle time Standard deviation % cases outside of the upper limit
Question		Is the performance of the process improving?
Metrics		$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction

Basili, Caldiera, Rombach (1994)

Research

Metric	Threshold Value	
Non-Self Transitions	-	
Transitions/State	Middle level state	Top Level State
	4 -5	3-4
State Depth	3	

Rank = 1.2 + 0.007NonSelfTransitions + 0.17Transitions/state + 0.25StateDepth



Rezaei, Ebersjö, Sandahl, Staron
Identifying and managing complex modules in executable software design models
IWSM Mensura 2014 conference

Software Metrics/Kristian Sandahl

www.liu.se

LINKÖPING UNIVERSITY