Quality Management of Software and Systems (WS19/20)

Problem Set 5

Problem 1: Measuring and Scales

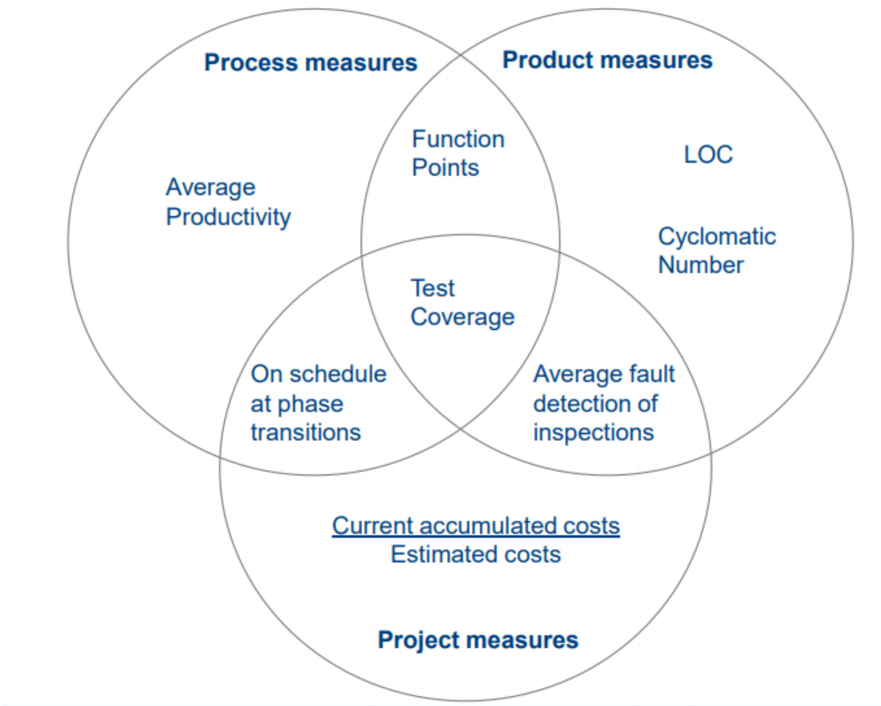
a) What is the objective of measurement? Why is it relevant to software quality?

-

- Substitutes qualitative and usually intuitive statements about software for quantitative and reproducible statements.
- Quality Characteristics could be:
  - Reliability (determined experimentally)
  - Number of faults (determined statistically)
  - Safety (determined in safety cases)
- Brings the quality of software to a defined level (e.g. availability of 99%)
- Helps to have more precise, predictable and repeatable control over the software development process and product -> this way software quality will improve.

b) What can be measured when doing a software development?

-



c) Which measurement scales do you know? Please give a brief explanation and an example of each one.

-

- Nominal scale
  - Free labeling of specific characteristics
  - Inventory numbers of library books (DV 302, PH 002, CH 056, ...)
  - Names of different requirements engineering methods (SA, SADT, OOA, IM, ...)
- Ordinal scale
  - Mapping of an ordered attribute's aspect to an ordered set of measurement values, such that the order is preserved.
  - Mapping of patient arrivals to the waiting list in a medical practice
- Interval scale
  - A scale, which is still valid if transformations like $g(x) = ax + b$, with $a > 0$ are applied
  - Temperature scales in degree Celsius or Fahrenheit. If F is a temperature in the Fahrenheit scale, the temperature in the Celsius scale can determined as follows: $C = 5/9 (F - 32)$. The relations between temperatures are preserved
- Rational scale
  - Scale, where numerical values can be related to each other (percentile statements make sense)
  - Length in meters (It is twice as far from a to b than from c to d)
  - Temperature in Kelvin
- Absolute scale
  - Scale, providing the only possibility to measure circumstances
  - Counting

Summary:

	Count	Order	Build difference	Build ratio
Nominal scale	Yes	No	No	No
Ordinal scale	Yes	Yes	No	No
Interval scale	Yes	Yes	Yes	No
Rational scale	Yes	Yes	Yes	Yes

Problem 2: Halstead Metrics

-

Operators:

- Mathematical operations, function calls, procedure calls, keywords
- Examples: while, for, if, +,-,*,/,=,(….),{…}, int, string, …

Operands:

- Variable, constants, jumps, labels
- Example: int text, i++, i=0, Pi=3.14, …

n_1 : number of different operators

n_2 : number of different operands

N_1 : overall number of used operators

N_2 : overall number of used operands

$n = n_1 + n_2$: size of vocabulary

$N = N_1 + N_2$: length of implementation

Difficulty to understand the program: $D = \frac{n_1 . N_1}{2n_2}$
Volume of the program: $V = N.log_2.n$
Effort for the implementation: $E = D.V$

a) Please do not take into account variable declarations, comments as well as procedure declarations. This means you should start from line "pos = 0;" with the calculation.

- n_1 = 16
- n_2 = 12
- N_1 = 43
- N_2 = 29
- n = 28
- N = 72
- D = 19.33
- V = 346.1
- E = 6690.11

Note: The colon and the parentheses are considered to be a part of switch (…) while for (…) and catch (….)

Loops: for (…), if (…), while (…) etc.

Operators	Occurrences	Operands	Occurrences
=	5	||	4
;	8	++	1
.ToLower	1	pos	4
while	1	0	3
(…)	4	gesamt	2
.Length	1	vokale	2
>	1	text	4
{…}	2	1	3
+=	2	zchn	6
.Substring	1	a	1
,	1	e	1
if	1	i	1
==	5	o	1
"…."	5	u	1

b) Halstead metrics

- n_1 = 13
- n_2 = 7
- N_1 = 20
- N_2 = 12
- n = 20
- N = 32
- D = 11.14
- V = 138.27
- E = 1540.35

Operators	Occurrences	Operands	Occurrences
for	1	,	1
(….)	4	==	1
=	1	Return	2
;	4	pos	5
<=	1	0	1
.Length	1	Schuhmenge	2
++	1	Convert	1
if	1	1	1
ToChar	1	Element	1
.Substring	1	-1	1

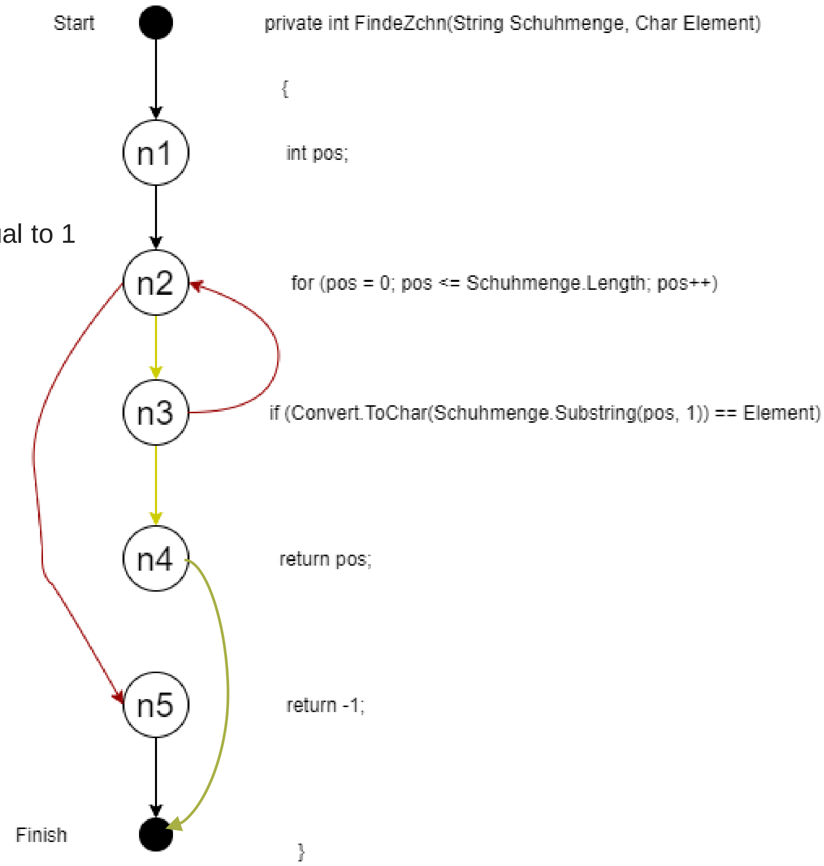
Cyclomatic number:

Equation: m - n + 2P

m: the number of edges of the graph.
n: number of nodes
p: number of nodes that have exit points

Note: For a single program, *P* is always equal to 1

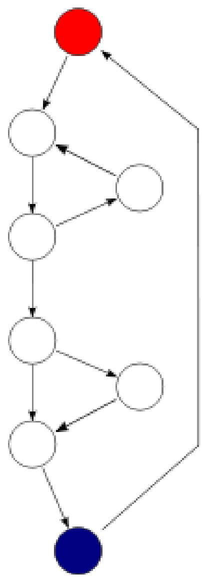
So, 8-7+ (2*1) = 3



Notes:

An alternative formulation is to use a graph in which each ***exit point is connected back to the entry point***. In this case, the graph is said to be strongly connected, and the cyclomatic complexity of the program is equal to the cyclomatic number of its graph (also known as the first Betti number), which is defined as

M = E − N + P

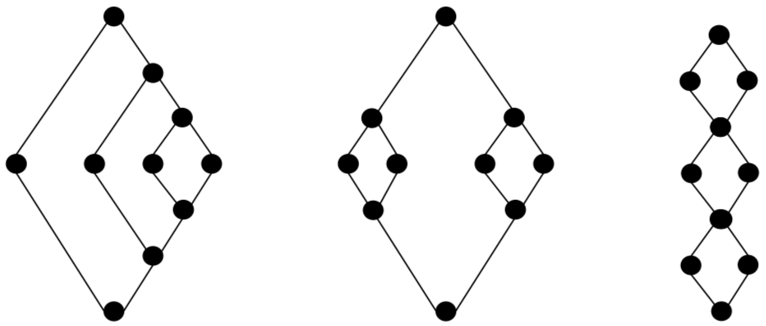


For this graph, E = 10, N = 8 and P = 1, so the cyclomatic complexity of the program is still 3.

Problem 3: Cyclomatic number

Determine the Cyclomatic number for the following flow graphs. **Compare** this with your intuitive impression.

-



G1 G2 G3

Cyclomatic number of all graph is 4 (12-10+2)