SW Measurement

PA017 SW Engineering II $\rightarrow$ Aspects of SW Development Management

Jaroslav Ráček **Josef Spurný**

Faculty of Informatics, Masaryk University

November 1, 2022

Definitions

Measure

- Quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process
- A fundamental or unit-specific term
- Example: Number of errors

Metric

- Quantitative measure of degree to which a system, component or process possesses a given attribute
- Metric is context-specific, i.e., it has goal / intent to measure [something]
- Metric is derived from one or more measures
- *A handle or guess about a given attribute*
- Example: Number of errors found per person hours expended

Why Measure Software?

- Determine and manage quality of the current product or process
  - Removing unnecessary complexity / size will have impact on SW quality
- Predict qualities of a product/process
- Improve quality of a product/process

Example Metrics

Frequently used metrics related to SW quality and testing:

- Defects rates
- Errors rates
- As measured per:
  - whole software
  - functional module
  - development phase
  - time unit
- Errors should be categorized by:
  - origin (analysis, design, coding...)
  - type (trivial, severe, critical...)
  - cost (usually as a consequence of previous categories) - as impacting customer's business or as resources required to remove error/defect/bug...

Metrics Classification

Products

- Explicit results of software development activities
- Deliverables, documentation, by products
- Example: LOC, FP, errors per LOC

Processes

- Activities related to production of software
- Example: FP unused in final product, Man-Days, milestones, effort

Resources

- Inputs into the software development activity
- Hardware, knowledge, people
- Example: Server count, certifications, staff seniority, available technologies, experience with given technology

Process vs. Product

Process Metrics

- Insights of process paradigm, software engineering tasks, work product, or milestones
- Lead to long term process improvement

Product Metrics

- Assesses the state of the project
- Track potential risks
- Uncover problem areas
- Adjust workflow or tasks
- Evaluate teams’ ability to control quality

Types of Measures

Direct Measures ("*hard*", internal attributes)

- Objective, relatively easy to measure and quantify by units
- Cost, Effort, LOC, Speed, Memory

Indirect Measures ("*soft*", external attributes)

- May be less objective, have to be inferred, assessed or judged
- Functionality, Complexity, Quality, Efficiency, Reliability, Maintainability

Size Oriented Metrics

- Size of the software produced
- Used in COCOMO
- Examples:
  - Lines Of Code (LOC)
  - 1000 Lines Of Code (KLOC)
  - Effort measured in person months
  - Errors/KLOC
  - Defects/KLOC
  - Cost/LOC
  - Documentation Pages/KLOC

LOC Metrics

- Advantage: easy and fast to calculate
- Disadvantage: dependency on programmer (seniority) & language (high vs low level)
  - Illustration: compare beginner vs. senior code for certain function
  - Usually meaningful only when team / project (technology) is unchanged

Function Oriented Metrics

- Function Point Analysis [Albrecht ’79, ’83]
- International Function Point Users Group (IFPUG)
- Size-based metric, complexity is used only partially (GSP - general system characteristics)
- Indirect measure
- Derived using empirical relationships based on countable (direct) measures of the software system (domain and requirements)

Revisited Function Points Computation

Function points count

= $(0.65 + \frac{GSP}{100}) * UFP$, where

GSP = sum of general system characteristics evaluation (14 adjustment values) determined for each organization via empirical data
UFP = unmodified function points

Using FP

- As oposed to LOC, FP is programmer & language independent
- Focused on delivery of functionality / added value to customer
- Yet usage is similar to LOC:
  - Errors per FP
  - Defects per FP
  - Cost per FP
  - Pages of documentation per FP
  - FP per person month

FP & Languages

As already mentioned, we can derive LOC from FP

Language	Average	Median	Low	High
Assembler	209	203	91	320
C	148	107	22	704
C++	59	53	20	178
HTML	43	42	35	53
J2EE	57	50	50	67
Java	55	53	9	214
JavaScript	54	55	45	63
.NET	60	60	60	60
Visual	50	52	14	276

FP Controversy

Similarly to LOC metrics, FP has proponents and opponents:

- Proponents claim that:
  - FP is programming language independent
  - FP is based on data that are more likely to be known in the early stages of a project, making it more attractive as an estimation approach
- Opponents claim that:
  - FP requires some skill and "*mathemagic*" because the computation is based on subjective data
  - Counts of the information domain can be difficult to collect
  - FP has no direct physical meaning …it’s just a number

Complexity Metrics

- Language and programmer dependent
- Halstead's Complexity
  - Is dependent on actual implementation of the code
  - Perceives code as a sentence and investigates its "*richness*"
  - The following measures are used:
    - n_1 - number of unique operators
    - n_2 - number of unique operands
    - N_1 - total number of operators
    - N_2 - total number of operands

Example

```
if (k < 2)
{
if (k > 3)
x = x*k;
}
```

- Unique operators: if ( ) { } > < = * ;
- Unique operands: k 2 3 x
- n_1 - 10
- n_2 - 4
- N_1 - 13
- N_2 - 7

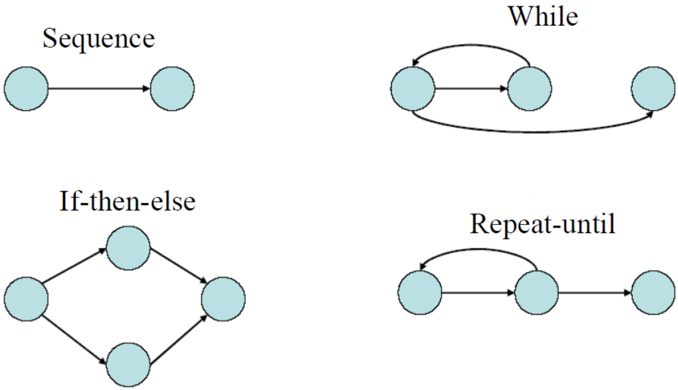
Halstead's Metrics

- Length = $N = N_1 + N_2$
- Vocabulary = $n = n_1 + n_2$
- Estimated "*optimal*" (well-structured) program length
  - $N_e = n_1 log_2 n_1 + n_2 log_2 n_2$
- Purity ratio $PR = \frac{N_e}{N}$
  - Ideal value is close to 1
  - Can show how complex / "long" / "short" code a programmer writes
  - Values far from 1 can cause problems with readability of code for others – impact on future service of SW
  - Average "lifespan" of programmer in one company is 5 years

McCabe's Complexity Measures

- McCabe's metrics are based on a control flow representation of the program
- A program graph is used to depict control flow
- Nodes represent processing tasks (one or more code statements)
- Edges represent control flow between nodes

Flow Graph Notation



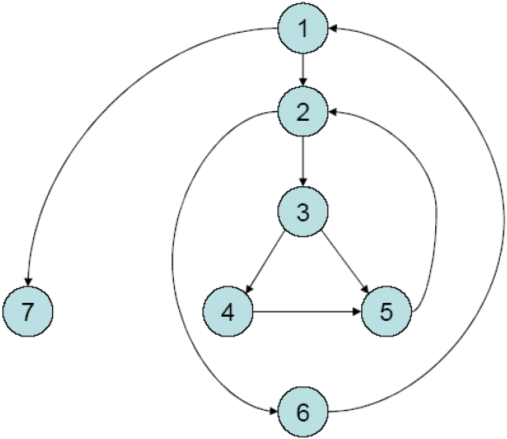
Cyclomatic Complexity

- Set of independent paths through the graph
- V(G) = $E - N + 2$, where
  - E is number of edges
  - N is number of nodes

Example

```
i = 0;
while (i < n-1) do
    j = i + 1;
    while (j < n) do
        if A[i] < A [j] then
            swap (A [i], A [j]);
        end do;
        i= i+1;
end do;
```

Example - Flow Graph



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

$V(G) = 9 - 7 + 2 = 4$

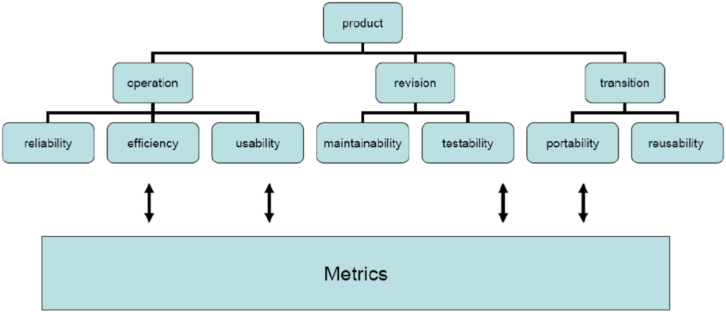
Cyclomatic Complexity - Meaning

- V(G) is the number of (enclosed) regions/areas of the planar graph
- Number of regions increases with the number of decision paths and loops
- Experimental data shows value of V(G) should be no more then 10 (for given level of decomposition)
- A quantitative measure of testing difficulty and an indication of ultimate reliability
- Already tested units are considered as one node, despite they might have their own inner complexity

Cyclomatic Complexity & Testing

- Higher cyclomatic complexity usually means lower productivity and higher potential for bugs
- Application in whitebox testing
- Testing for V(G) > 10 is very difficult
- Cyclomatic complexity has impact on provision of support & service of SW

Quality Model & Metrics



MUNI
FACULTY
OF INFORMATICS