Contents

1 Why Measure Software?

- To **determine the quality** of the current product or process and to **make informed comparisons**
- To **predict** qualities of a product/process
- To **improve** quality of a product/process (often continuously *kaizen*)

2 Metrics

2.1 Example metrics

- Error rates (wrong result or illegal action) / Defect rates (failure to do something)
- Code generation rates. Measured by: per **person**, per **team**, per **project**. Will involve **aggregation** see later
- Errors should be categorized by origin, type etc.

2.2 Main objectives of software quality metrics

- *Facilitate* management control, planning and managerial intervention. Based on deviations of actual
  - performance from planned.
  - timetable and budget performance from planned.
- Identify situations for development or maintenance process **improvement** (preventive or corrective actions). Based on:
  - Accumulation of metrics information regarding the performance of teams, units, etc.

2.3 Software quality metrics - Requirements

General requirements:

- Relevant
- Valid
- Reliable
- Comprehensive
- Mutually exclusive

Operative requirements:

- Easy and simple
- Does not require independent data collection
- Immune to biased interventions by interested parties

2.4 Metrics evaluations

Metrics can be evaluated on:

- **Products**: Explicit results of software development activities. Deliverables, documentation, other artifacts produced
- **Process**: Activities related to production of software
- **Resource**: Inputs into the software development activities, such as hardware, knowledge, people.

2.5 Process metrics

Rates of production / productivity, quality of estimates, burn down rate (project velocity), refactoring rate. These can lead to **long term process improvement** by driving reflection on the process and suggestions for improvement

2.6 Product metrics

Error densities, complexity measures, code quality measures, speed and memory measures, code profiles etc, track potential risks, uncover problem areas, Adjust workflow or tasks. They **evaluate teams ability to control quality**

2.6.1 Types of measures

- Direct: Cost, effort, LOC, speed, memory
- Indirect: functionality, quality, efficiency, reliablity, maintainability

2.6.2 Size oriented metrics

- Size of software produced
- LOC - Lines of Code or KLOC - 1000 Lines of code
- SLOC statement lines of code (no whitespace)
- Number of function points (NFP)
- Open used as part of density CONT

2.7 Process metrics categories

- Software oricess quality metrics: Error density, severity metrics
- Software process timetable, error removal effectiveness and productivity metrics.

3 Process metrics

3.1 Definitions

- NCE = total detected by code inspections and testing.
- NDE = total errors detected in the development process.
- WCE = weighted NCE
- WDE = weighted NDE
- MSOT = Milestones completed on time.
- MS = Total number of milestones.
- TCDAM = Total Completion Delays (days, weeks, etc.) for all milestones
- NYF = number software failures detected during a year of maintenance service.
- WYF = weighted number of software failures detected during a year of maintenance service.
- DevH = Total working hours invested in the development of the software system.
- ReKLOC = Number of thousands of reused lines of code.
- ReDoc = Number of reused pages of documentation.
- NDoc = Number of pages of documentation

3.2 Error density metrics

- CED (Code error density) $CED = NCE/KLOC$
- DED (Development error density) $DED = NDE/KLOC$
- WCED (Weighted code error density) $WCDE = WCE/KLOC$
- WDED (Weighted development error density) $WDED = WDE/KLOC$
- WCEF (Weighted code errors per function point) $WCEF = WCE/NFP$
- WDEF (Weighted development errors per function point) $WDEF = WDE/NFP$

3.3 Error severity metrics

- ASCE (Average severity of code errors) $ASCE = WCE/NCE$
- ASDE (Average severity of developmment errors) $ASDE = WDE/NDE$

3.4 Software process timetable metrics

- TTO (Time table observance) $TTO = MSOT/MS$
- ADMC (Average delay of milestone completion) $ADMC = TCDAM/MC$

3.5 Error removal effectiveness metric

- DERE (Development errors removal effectiveness) $DERE = NDE/(NDE + NFY)$
- DWERE (Development weighted erros removal effectiveness) $DWERE = WDE/(WDE + WYF)$

3.6 Software process productivity metrics

- DevP (Development productivity) $DevP = DevH/KLOC$
- FDevP (Function point development productivity) $FDevP = DevH/NFP$
- CRe (Code reuse) $Cre = ReKLOCK/KLOCK$
- DocRe (Documentation reuse) $DocRe = ReDoc/NDoc$

4 LOC based metrics

- Easy to use and compute
- Language & programmer dependent
- Can be counter-productive if used as a basis for reward! (Program can be written in more lines of code and made overly complex)

4.1 Complexity Metrics

- LOC - a function of complexity
- Language and programmer dependent
- Halsteads Software Science (entropy measures)
  - n_1 - number of distinct operators
  - n_2 - number of distinct operands
  - N_1 - total number of operators
  - N_2 - total number of operands

4.2 Halsteads Metrics

- Program length: $N = N_1 + N_2$
- Program vacabulary: $n = n_1 + n_2$
- Estimated length: $N\hat{} = n_1 * log_2 n_1 + n_2 * log_2 n_2$ (Close estimate of length for well structured programs)
- Purity ratio $PR = N\hat{}/N$

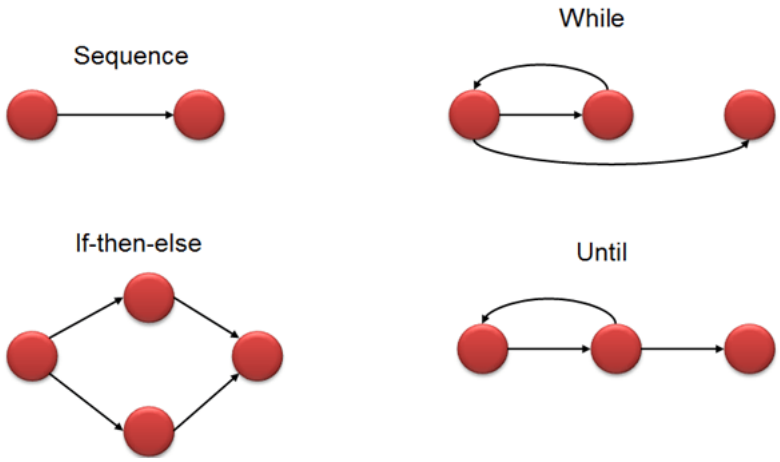
5 Program complexity

- Volume: $V = N \log_2 n$ (Number of bits to provide a unique designator for each of the n items in the program vocabulary)
- Difficulty: $D = (n_1/2) * (N_2/n_2)$
- Program effort $= E = D * V$ (good measure of program understandability)

6 McCabes Complexity Measures

- McCabes 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**

6.1 Flow Graph Notation



6.2 Cyclomatic Complexity

- Set of independent paths through the graph (basis set)
- $V(G) = E - N + 2$
  - E is the number of flow graph edges
  - N is the number of edges
- $V(G) = P + 1$
  - P is the number of predicate nodes (branching)

6.3 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
- This is a quantitative measure of the number of tests needed
- It correlates very well with the rate of errors
- Experimental data shows value of V(G) should be no more then 10 - testing is very difficult above this value

7 McClures Complexity Metric

Complexity = $C + V$

- C is the number of comparisons in a module
- V is the number of control variables referenced in the module

decisional complexity. Similar to McCabe's but with regard to control variables

8 Metrics and Software Quality (FURPS)

- Functionality - features of system
- Usability aesthesis, documentation
- Reliability frequency of failure, security
- Performance speed, throughput
- Supportability maintainability

9 Measures of Software Quality

- Measures of Software QualityCorrectness degree to which a program operates according to specification
- Mintainability degree to which a program is open to change
- Integrity - degree to which a program is resistant to outside attack
- Usability easiness to use

10 Conclusions

- Software metrics can give much useful information about the **quality of code**
- They **allow comparisons** between projects, developers and teams and over time
- They suggest **areas for improvement** and give **early warnings** of problems (as in manufacturing process metrics)
- Management need to know what quality problems exist but
- Developer should know what needs corrected
- Metrics tools can be very useful for code analysis

Reference section

measure
: A quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process. for example: Lines of code (LOC), Number of errors

metric
: quantitative measure of degree to which a system, component or process possesses a given attribute. That is to say a derived measure. May involve guesstimates where human processes are concerned. for example: Error density.

kaizen
: An approach to **continuous** organization improvement with emphasis on continuous measure and understanding of process.

facilitate
: make (an action or process) easy or easier.