Unit 16: Software Metrics

Presented by
Santhan Perampalam

Objectives

- Provide an overview of Software Metrics
  - Measurement
  - Metric Types
  - Scales
- Give examples of where Metrics are used
- Explain some of the issues with Software Metrics
- Look at why Software Metrics is important
- Look at Metrics with regard to Object Oriented programming
- Outlines some of the plans for the future

Software Metrics?

- Control
- *“You cannot control what you cannot measure”*
  – Tom DeMarco
- *“I know no way of judging the future but by the past”*
  – Patrick Henry
- To provide information to support quantitative managerial decision-making during the software lifecycle

Another Definition

- Software Metrics is a collective term used to describe the very wide range of **activities** concerned with **measurement** in software engineering

Measurement

- Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules
- Example:

Entity	Attribute
Coded program	Number of lines of code
Team member	Price
Organisation	Size
Test data	Coverage
Test data	Size

Metric Types

There are 3 metric types:

- Direct Measurement
  - Eg length of source code or duration of testing process. (These measurement activities involve no other attributes or entities)
- Indirect/Derived Measurement
  - Eg Module defect density = Number of defects/ Module Size. (These are combinations of other measurements)
- Predictive Measurement
  - Eg predict effort required to develop software from the measure of its functionality – function point count. (These measurements are systems consisting of mathematical models together with a set of prediction procedures for determining unknown parameters and interpreting results)

Classification of Scales

- Nominal Scale
  - This is the most primitive form of measurement. It is a scale that consists only of different classes that have no ordering
  - Any distinct numbering or symbolic representation of the classes have no magnitude associated with them.
  - Eg IEEE 802.1, 802.2, 802.3,…802.11
- Ordinal Scale
  - The classes are ordered with respect to the attribute. There is no quantitative comparison.
  - Eg programmer skill (low, medium, high)

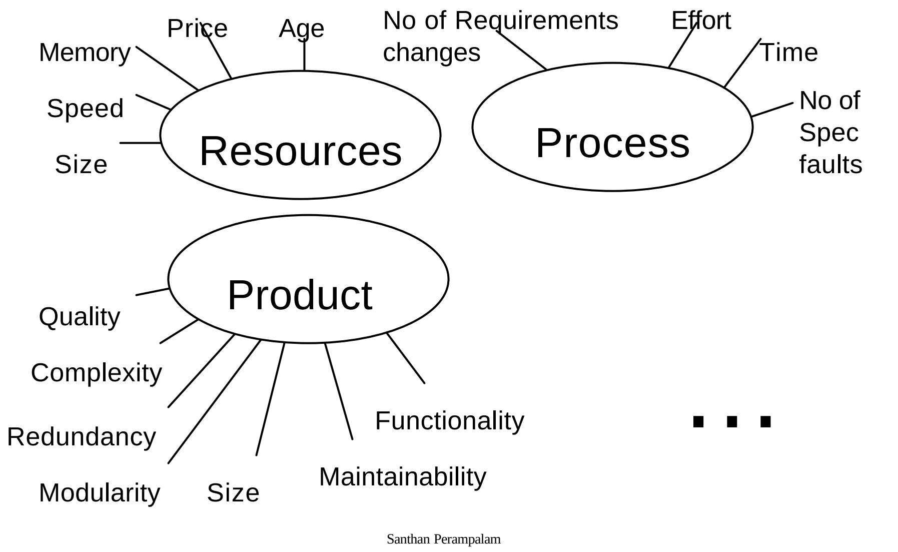
Classification of Scales cont.

- Interval Scale
  - This scale captures information about the size of the intervals that separate classes. Thus we can understand the magnitude of the jump from one class to another.
  - Addition and subtraction operations are permissible
  - Eg programmer capability between: 60^{th} and 80^{th} percentile of population
- Ratio Scale
  - A measurement mapping that the preserves ordering, the size of intervals between entities, and ratios between entities
  - Eg project A took twice as long as project B

Classification of Scales cont.

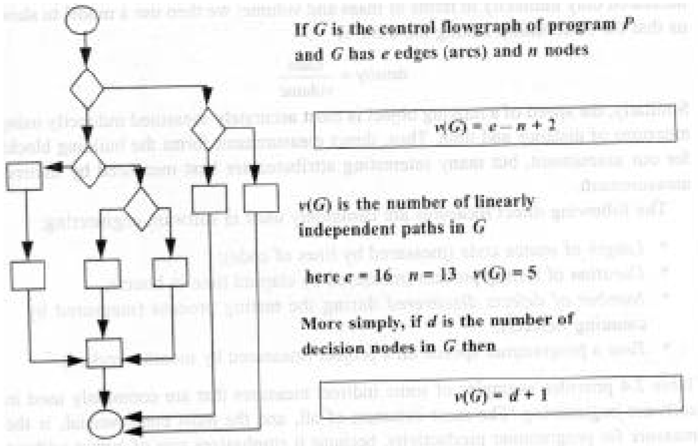
- Absolute Scale
  - States that there is only one way in which the measurement can be made. There is only one possible measurement mapping – the actual count.
  - Eg number of failures observed during integration testing can be measured only by counting the number of failures observed.
  - For better understanding consider:
    - Length of source code (of which LOC is a ratio scalar)
    - Engineer’s age (of which years is a ratio scalar)
    - Number of lines of code (of which LOC is an absolute scalar)

Application Areas



Examples in Metrics (1)

McCabe’s Cyclomatic Number



Examples in Metrics (2)

- Number of Lines of Code
  - *A line of code in any line of program text that is not a comment or a blank line, regardless of the number of statement or fragments of statements on that line. This specifically includes all lines containing program headers, declarations, and executable and non-executable statements*
    - Conte
- Function Point Count
  - A measure of the functionality perceived by the user delivered by the software developer

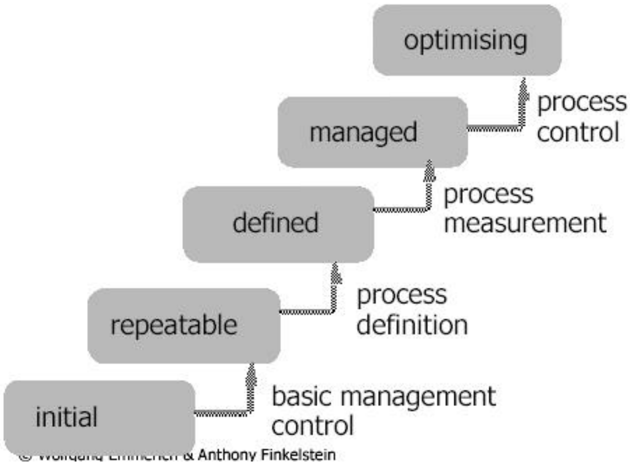
Issues with the Study

- Up until recently there have not been enough studies that directly address the problems of metrics in Object Oriented software (this is due to complexity)
- Determining:
  - what data to collect
  - How to collect it
  - How to process the information once collected
- Time

Why is it Important?

- Other engineering disciplines tell us that measurement is important
- We need to be able to make informed decisions about risk assessment and mitigation
- We need to improve the quality of the products
- We need to be able to predict accurately software economic issues
- We have a proven standard that says it is

CMM revisited



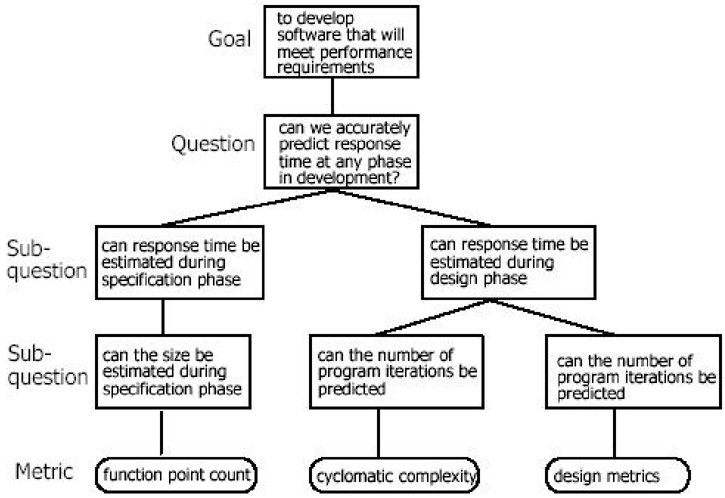
Santhan Perampalam

How do we choose the metric?

- Gilb's Principle of Fuzzy Targets: projects without clear goals will not achieve their goals clearly.
  - Tom Gilb
- We must decide carefully what to measure first

Santhan Perampalam

Goal-Question-Metric



MOOD

- Metrics for Object Oriented Design
- The 6 MOOD metrics:
  - Attribute Hiding Factor (AHF)
  - Method Hiding Factor (MHF)
  - Method Inheritance (MIF)
  - Attribute Inheritance Factor (AIF)
  - Polymorphism Factor (PF)
  - Coupling Factor (CP)
- These metrics have been tested against the 4 complexity issues they were intended measure and have proved successful over a range of empirical data
  - R. Harrison, S J Counsell, R V Nithi

Krakatau

- The best solutions available are CASE tools such Krakatau, which is an off-the-shelf product supporting data collection and analysis for Java (and C/C++)
- Krakatau supports a full range of OO, procedural, language specific, complexity and size metrics
  – Supported metrics include Cyclomatic Complexity, Enhanced Cyclomatic Complexity, Halstead Software Science metrics, LOC metrics and MOOD metrics. In all over 70 metrics are offered.

Plans for the Future

- The future for Software Metric advance as outlined by the paper in ‘The Future of Software Engineering’ lies in the construction of ‘causal’ models
- Followed by the application of Bayesian Bay Nets

Key Points

- To survive a software development organisation must make accurate cost estimates and improve productivity, quality and manage critical risks carefully
- If you do not know where you are now you certainly won't know where you'll be in the future
- To achieve accurate measurements of productivity and quality requires metrics collections and analysis

Santhan Perampalam