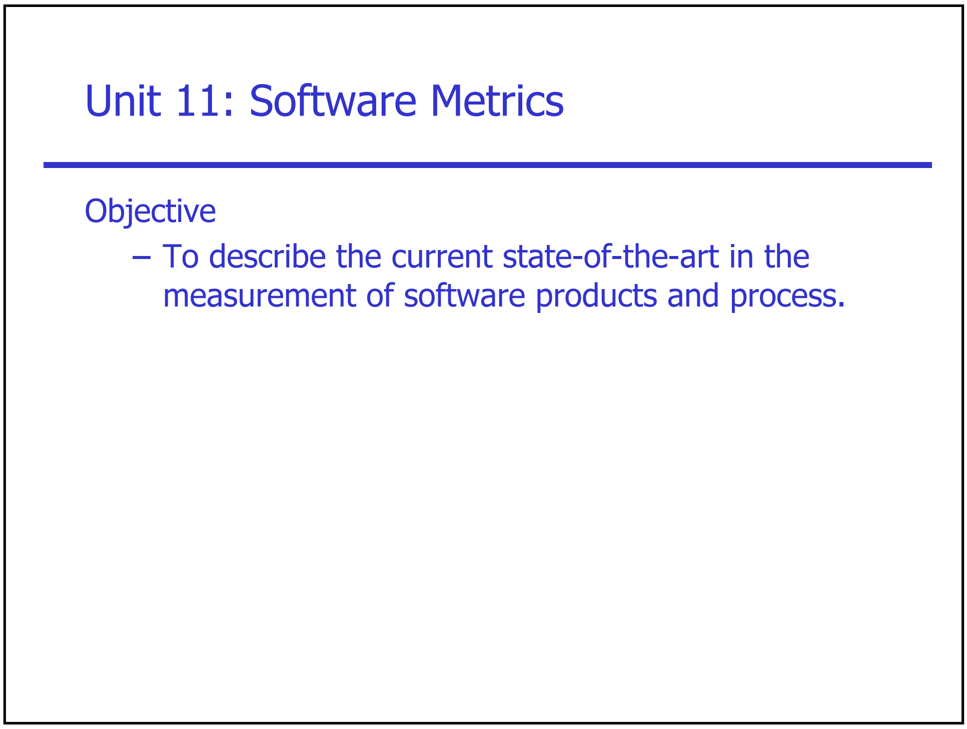
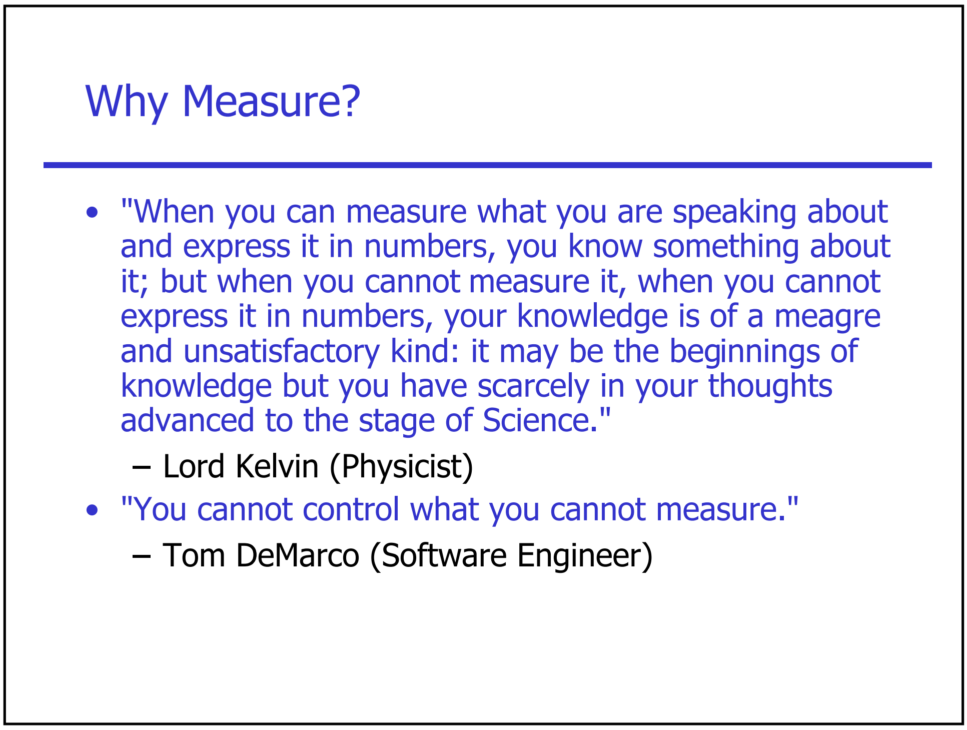


Unit 11: Software Metrics

Objective

- To describe the current state-of-the-art in the measurement of software products and process.



Why Measure?

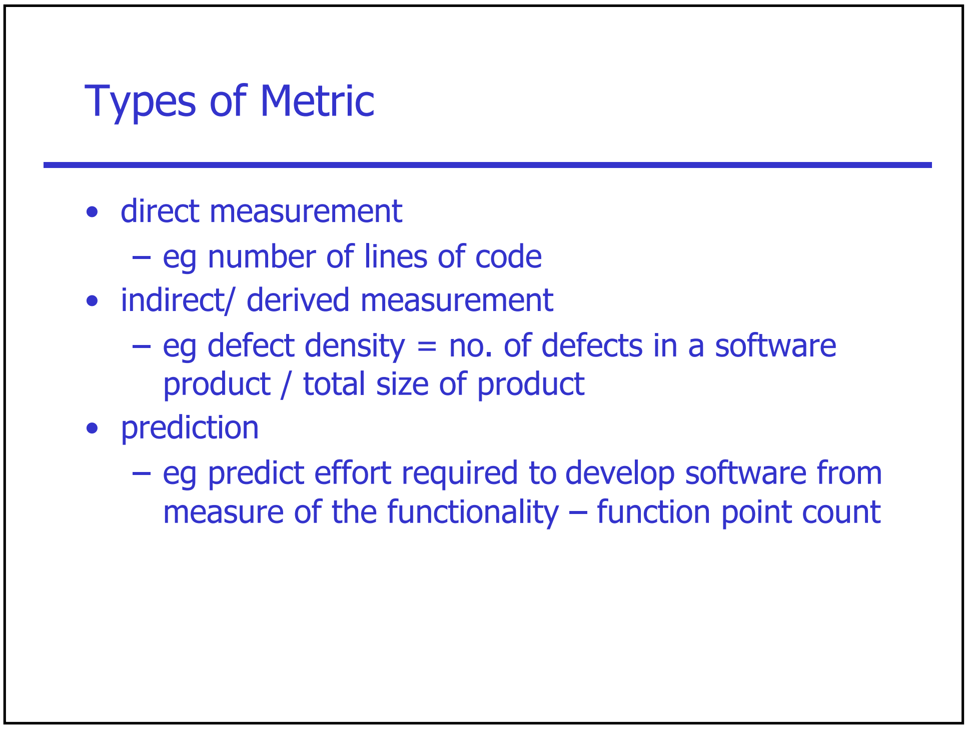
- "When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind: it may be the beginnings of knowledge but you have scarcely in your thoughts advanced to the stage of Science."
  - Lord Kelvin (Physicist)
- "You cannot control what you cannot measure."
  - Tom DeMarco (Software Engineer)

What is 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

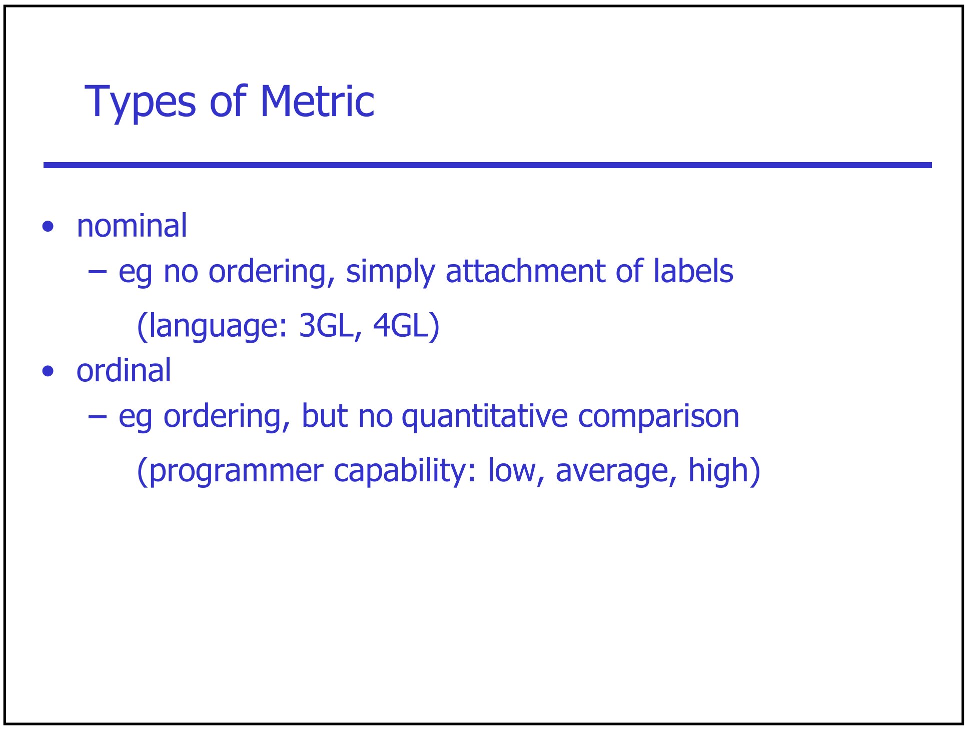
Examples of Entities and Attributes

Entity	*Attribute*
Software Design	Defects discovered in design reviews
Software Design Specification	Number of pages
Software Code	Number of lines of code, number of operations
Software Development Team	Team size, average team experience



Types of Metric

- direct measurement
  - eg number of lines of code
- indirect/ derived measurement
  - eg defect density = no. of defects in a software product / total size of product
- prediction
  - eg predict effort required to develop software from measure of the functionality – function point count



Types of Metric

- nominal
  - eg no ordering, simply attachment of labels

    (language: 3GL, 4GL)
- ordinal
  - eg ordering, but no quantitative comparison

    (programmer capability: low, average, high)

Types of Metric

- interval
  - eg between certain values

    (programmer capability: between 55th and 75th percentile of the population ability)
- ratio
  - eg (the proposed software is twice as big as the software that has just been completed)
- absolute
  - eg the software is 350,000 lines of code long

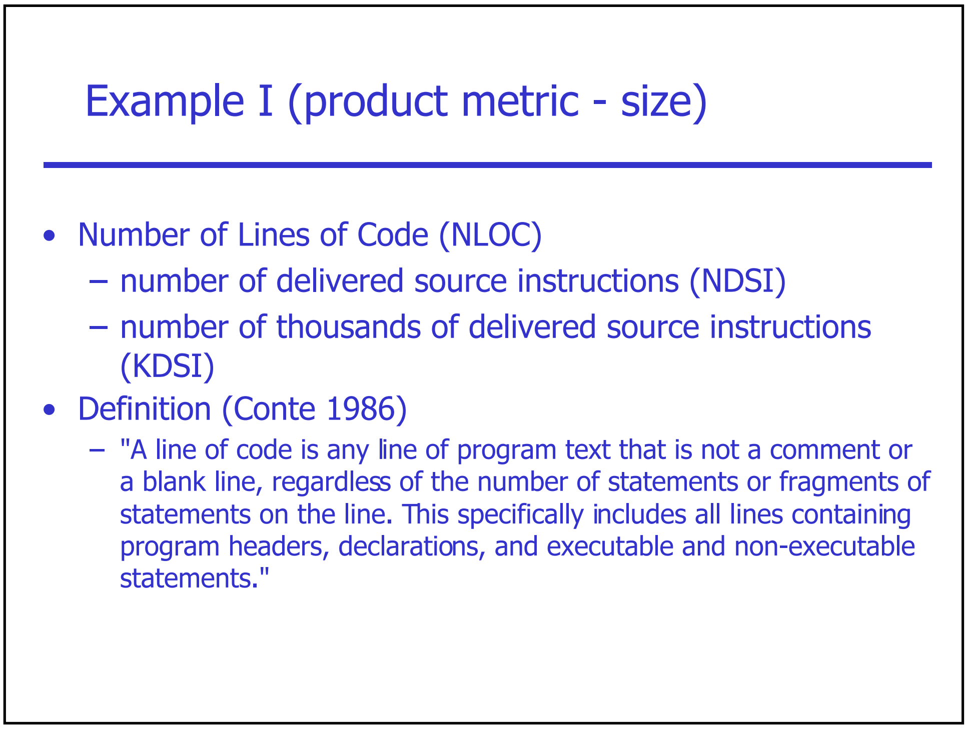
Types of Metric

- product metrics

  size metrics

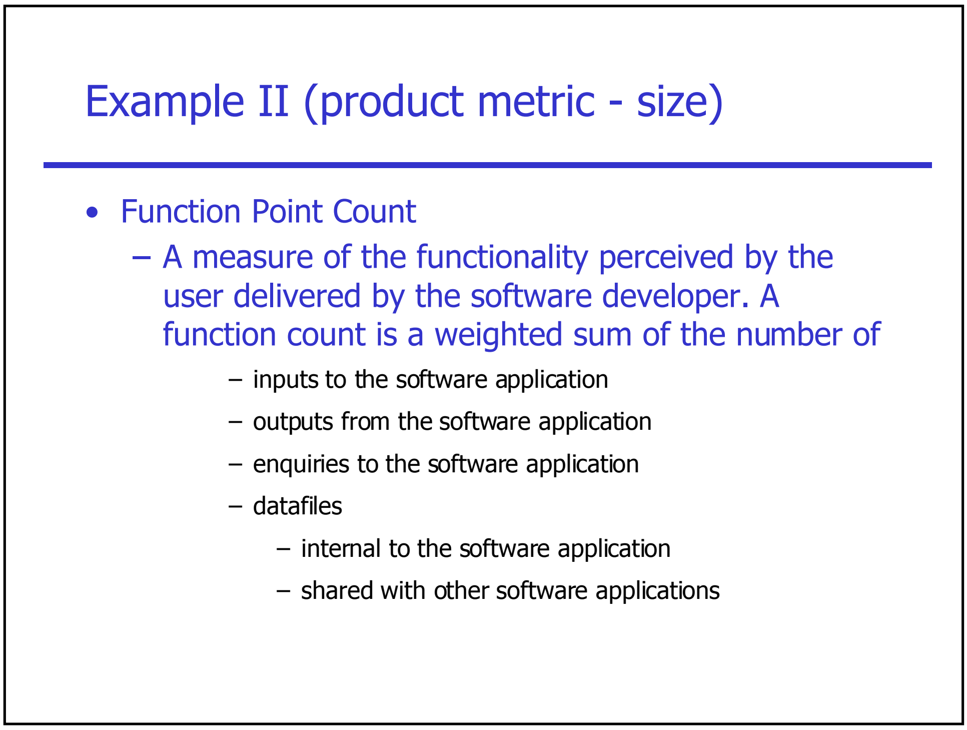
  complexity metrics

  quality metrics
- process metrics
- resource metrics
- project metrics



Example I (product metric - size)

- Number of Lines of Code (NLOC)
  - number of delivered source instructions (NDSI)
  - number of thousands of delivered source instructions (KDSI)
- Definition (Conte 1986)
  - "A line of code is any line of program text that is not a comment or a blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all lines containing program headers, declarations, and executable and non-executable statements."



Example II (product metric - size)

- Function Point Count
  - A measure of the functionality perceived by the user delivered by the software developer. A function count is a weighted sum of the number of
    - inputs to the software application
    - outputs from the software application
    - enquiries to the software application
    - datafiles
      - internal to the software application
      - shared with other software applications

Example (product metric - complexity)

- Graph Theoretic Metric
  - The McCabe Complexity Metric

    a software module can be described by a control flow graph where

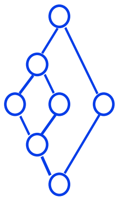
    - each node correspond to a block of sequential code
    - each edge corresponds to a path created by a decision

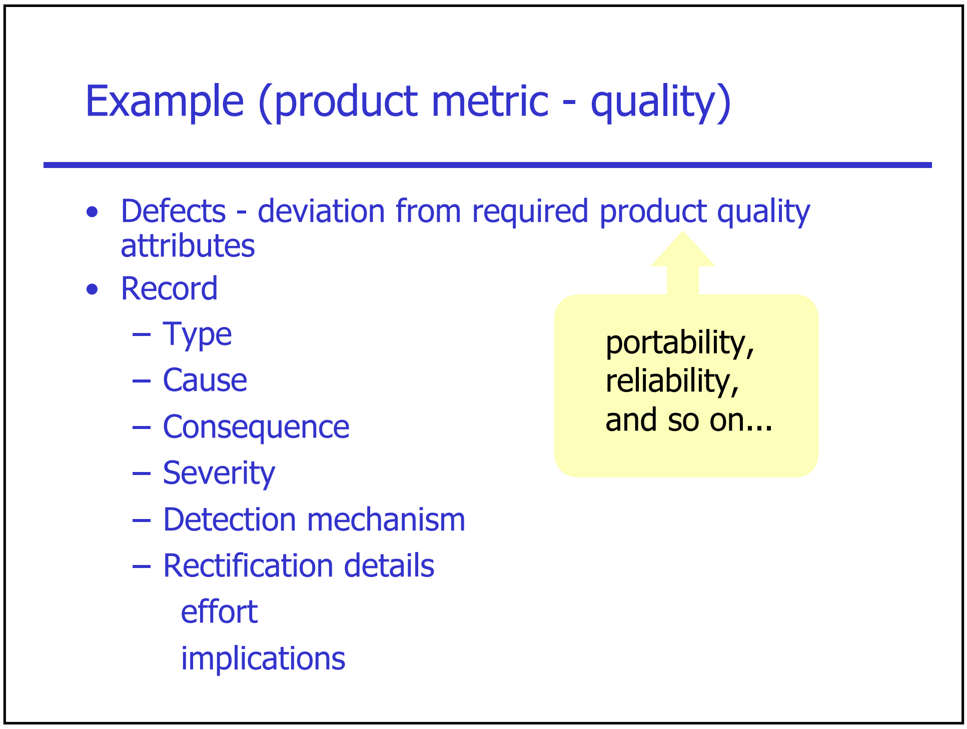
Example (product metric - complexity)

- V(G) = e – n + 2p
  - e = number of edges in the graph
  - n = number of nodes in the graph
  - p = number of connected module components in the graph

e=8
n=7
p=2

V(G)=5





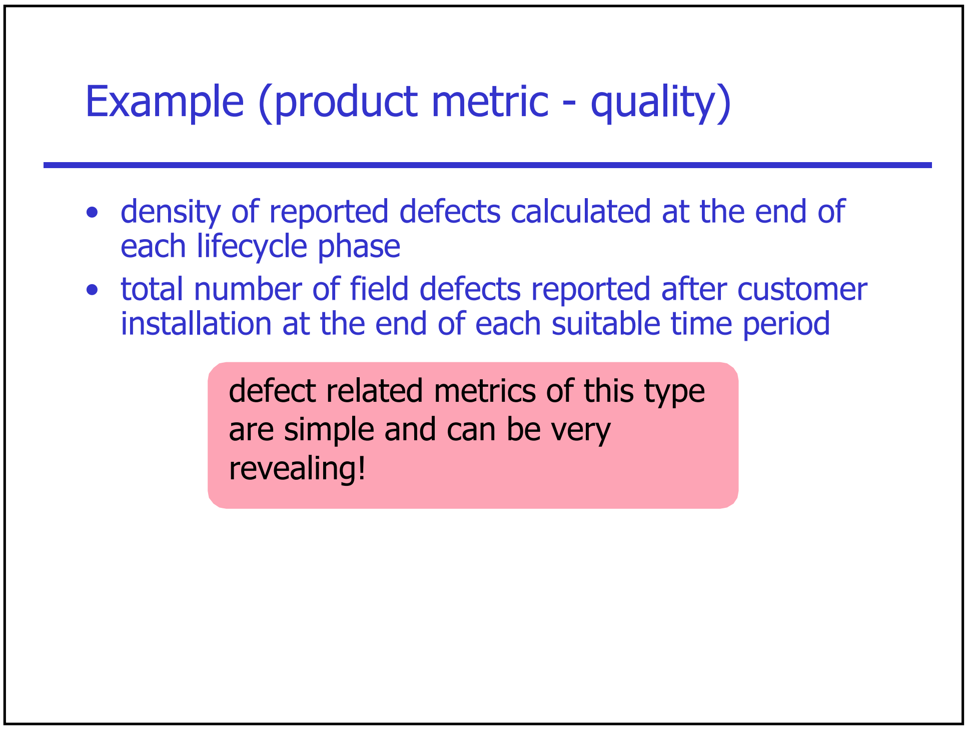
Example (product metric - quality)

- Defects - deviation from required product quality attributes
- Record
  - Type
  - Cause
  - Consequence
  - Severity
  - Detection mechanism
  - Rectification details

    effort

    implications

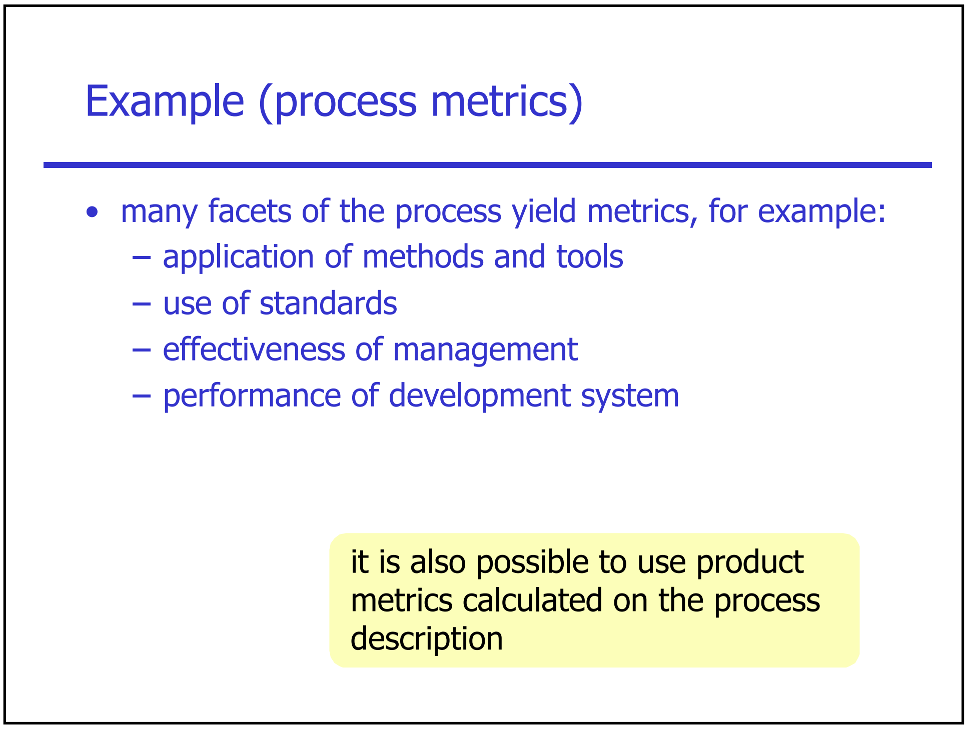
portability,
reliability,
and so on...



Example (product metric - quality)

- density of reported defects calculated at the end of each lifecycle phase
- total number of field defects reported after customer installation at the end of each suitable time period

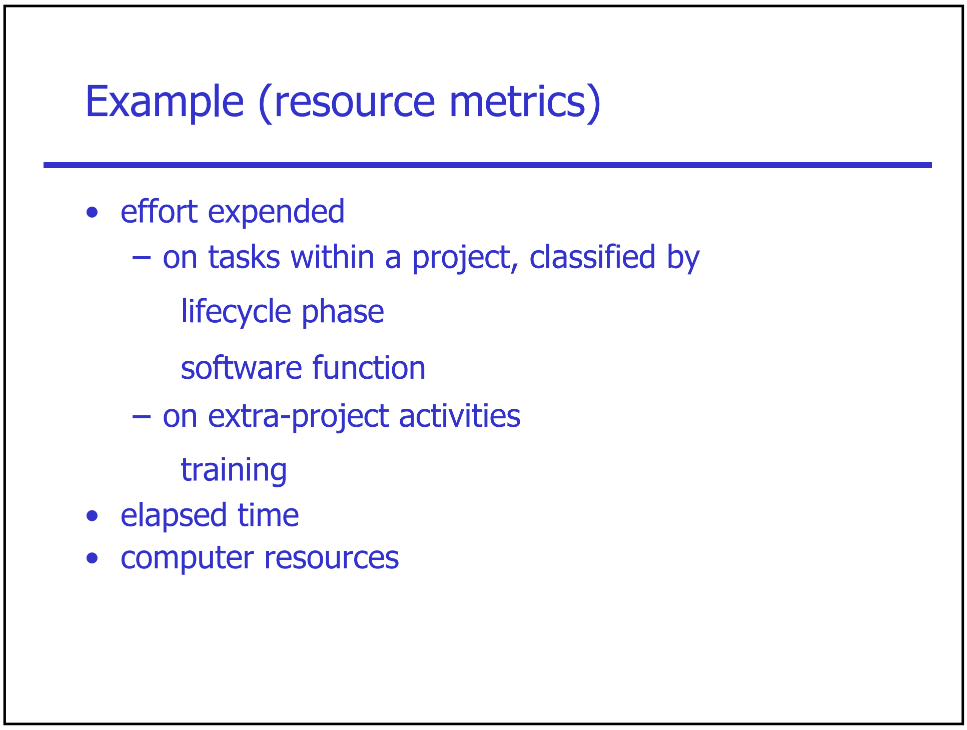
defect related metrics of this type are simple and can be very revealing!



Example (process metrics)

- many facets of the process yield metrics, for example:
  - application of methods and tools
  - use of standards
  - effectiveness of management
  - performance of development system

it is also possible to use product metrics calculated on the process description



Example (resource metrics)

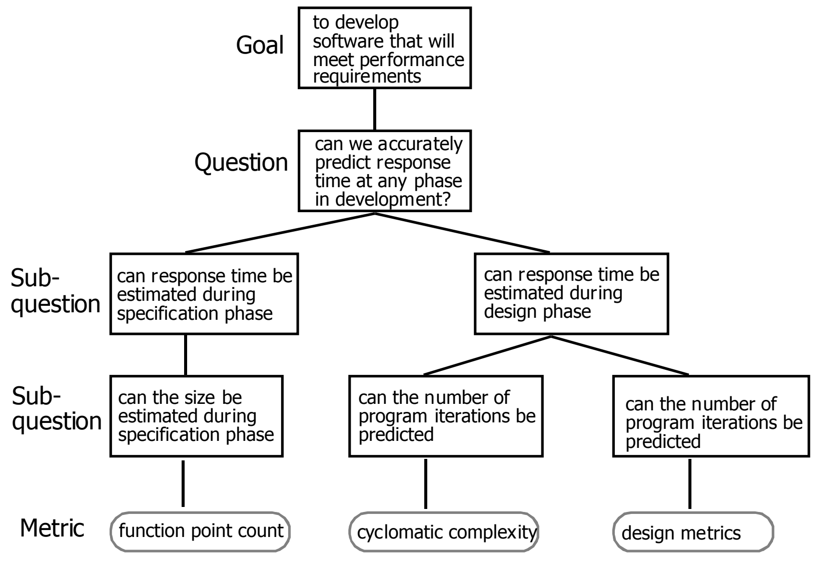
- effort expended
  - on tasks within a project, classified by
    - lifecycle phase
    - software function
  - on extra-project activities
    - training
- elapsed time
- computer resources

A Metrics Programme

- software development organisations should have a metrics programme in order to:
  - calibrate models that can be used to forecast project/product behaviour
  - give measures that can be used to control the software development process

involves metrics AND data collection

GQM (goal-question-metric) Approach



Goal

to develop software that will meet performance requirements

Question

can we accurately predict response time at any phase in development?

Sub-question

can response time be estimated during specification phase

can response time be estimated during design phase

Sub-question

can the size be estimated during specification phase

can the number of program iterations be predicted

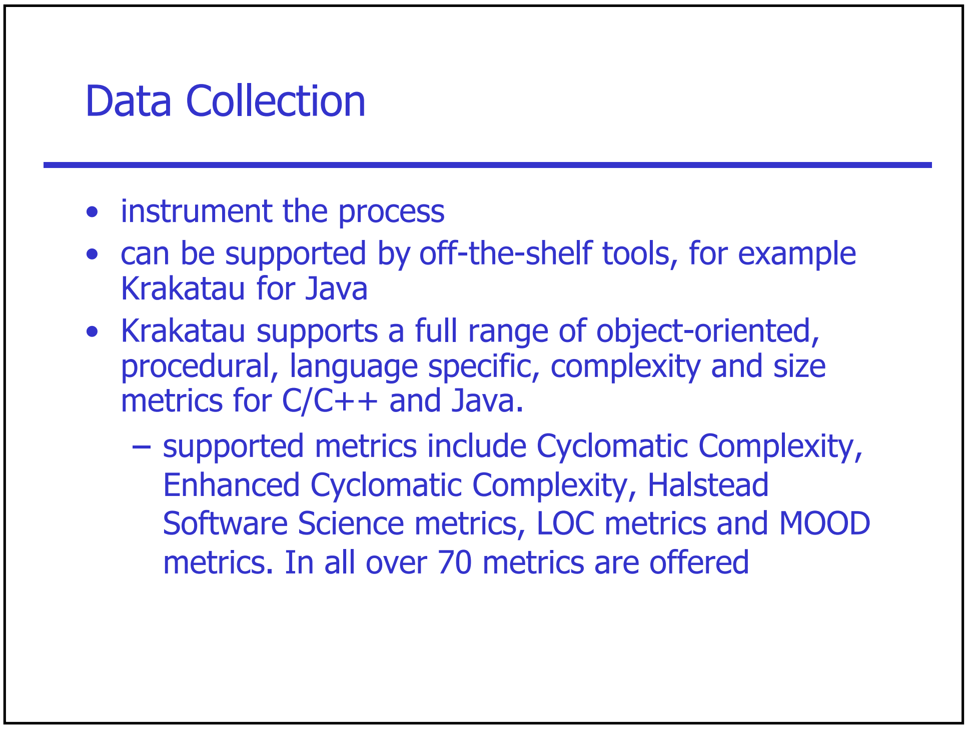
can the number of program iterations be predicted

Metric

function point count

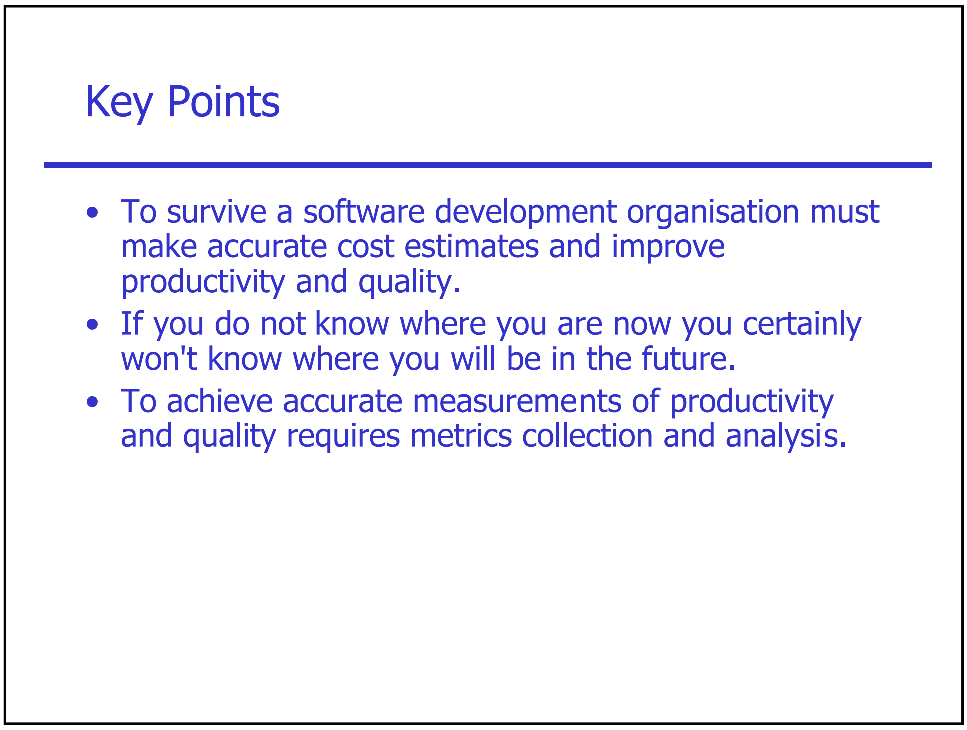
cyclomatic complexity

design metrics



Data Collection

- instrument the process
- can be supported by off-the-shelf tools, for example Krakatau for Java
- Krakatau supports a full range of object-oriented, procedural, language specific, complexity and size metrics for C/C++ and Java.
  - supported metrics include Cyclomatic Complexity, Enhanced Cyclomatic Complexity, Halstead Software Science metrics, LOC metrics and MOOD metrics. In all over 70 metrics are offered



Key Points

- To survive a software development organisation must make accurate cost estimates and improve productivity and quality.
- If you do not know where you are now you certainly won't know where you will be in the future.
- To achieve accurate measurements of productivity and quality requires metrics collection and analysis.