Process & Product Metrics

Software Requirements & Project Management
CITS3220
Issues in Project Management (a reminder)

- Before starting the project:
  - planning, estimation, risk analysis;

- While doing the project:
  - monitoring, evaluation, metrics;

- After the project is done:
  - assessment and improvement
Objectives for Today’s Lecture

After studying this lecture and its core reading material, students will be able,

- To explain the purposes and pitfalls of software measurement
- To evaluate proposed metrics
- To propose suitable metrics for a given situation
Measurement [Fenton & Pfleeger, p5, Hughes]

- **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 rules.

- A **metric** indicates the nature and/or strength of an attribute, usually by means of a number, often in conjunction with a unit.
Q: Why measure?
A: For control

“You cannot control what you cannot measure.” [De Marcos’ rule, 1982]

- What does each software process cost?
- How good is the code being developed?
- Have we found all the faults?
- How productive are the programmers?
Q: Why measure?
A: To improve

- Measurement provides **visibility**
- Process **maturity** = degree of visibility = amount of control and understanding
- Measurement provides baseline measures for making improvements
  - How effective is the code inspection process?
  - Does coding standard XYZ improve SW quality?
  - Do requirement specification standards reduce number of requirements changes later in the process?
Q: Why measure?
A: To understand

- Engineering disciplines use methods that are based on models and theories, and these are underpinned by experiment and measurement.
  - How long will it take to develop this system?
  - How productive are the staff?
  - What techniques should be used to ensure the system is safe?
The Scientific Method

- Models and theories have evolved by using the **scientific method**: state a hypothesis, design and run experiments to test its truth, analyse the results of those experiments.

- **Measurement**: underpins the scientific method
  - measuring the variables to differentiate cases
  - measuring changes in behaviour
  - measuring cause and effect
From theory to practice

Once the scientific method suggests the validity of a model, we continue to use the measurement to apply the theory to practice.

General reasons for measurement are not enough to justify the enterprise.

Software Engineers must have specific, clearly stated objectives for measurement.
Exercise:
Evaluate the following metrics

- See the metrics handout (download from website)
- “Characteristics of Good Metrics” and “Measurement Pitfalls”

- How well does each metric achieve the given measurement goal?
Examining Students’ Knowledge of Software Engineering

- Metric: shoe shiny-ness
- PRO ?
- CON ?
Reducing Road Accidents

- Metric: car speed radar
  - PRO
  - CON
Measuring programmer productivity

- Metric: KLOC per person-month

- PRO

- CON
Software Reliability

- Metric: execution time in seconds between successive crashes of the computer system
- PRO
- CON
Entity-Attribute Framework

- All entities of interest in SW can be classified as either
  - processes,
  - products or
  - resources.
- Anything we may wish to measure is an identifiable attribute of these entities
- See Fenton & Pfleeger Table 3.1, page 76
External attributes …

- can be measured only w.r.t how the product, process or resource relates to its environment
  - e.g. software quality and reliability of products,
  - e.g. stability of processes,
  - e.g. productivity of resources
- Often the most interesting attributes
- but they are hard to measure directly
Internal attributes ...

- can be measured **purely in terms** of the product, process or resource **itself**
  - e.g. size or structure of a product (lines of code, coupling and cohesion),
  - e.g. process duration, effort associated with the process, number of incidents or a specified type arising during the process

- Often used as indirect measures of external attributes
Exercise: suggest metrics for

- planning ability and project slippage
- effort and time
- requirements creep
- defect insertion, propagation, removal
Goal-Question-Metric Paradigm

- List the major goals of the project
- Derive from each goal the questions that must be answered to determine if the goals are being met
- Decide what must be measured in order to be able to answer the questions adequately
- Example see Fenton & Pfleeger Figure 3.2
GQM Worked Example

- **Purpose:** to evaluate the review process in order to improve it
- **Perspective:** Examine the cost of documentation inspections from the viewpoint of the manager
- **Environment:** Development staff resent time spent on documentation inspections. Managers desire a significant increase in software quality

Propose QUESTIONS & METRICS for this goal
Limitations of GQM

- Doesn’t tell you how to combine chosen metrics so that the questions can be answered
- Measurement may be beneficial even when goals are not clearly defined
- Who sets the goals? High level managers or the SW engineers?
- Doesn’t address measurement scale, objectivity, or feasibility of metrics
Some more metrics

- **Time**
  - calendar, total time, individual’s hours

- **Resource utilisation**
  - total person-days of effort, travel costs, computer resources

- **Events**
  - number of occurrences of a particular event such as defects discovered, requirements changes requested, number of lines of code (LOC) modified
Once you have identified useful metrics, assess your organization to see whether it is capable of providing useful information for the measurement.

Process maturity levels: 1. initial, 2. repeatable, 3. defined, 4. managed, 5. optimizing

Metrics for PMLs: 1. baseline, 2. project management, 3. product, 4. process + feedback for control, 5. process + feedback for changing the process