CISC327 - Software Quality Assurance

Lecture 25

Software Metrics

The Rest of the Course

- Software Metrics
- The full “Piece of Crap” lecture, if there’s time

Introduction to Software Metrics

- Today we begin looking at measurement of software quality using software metrics
  - What are software quality metrics?
  - Some basic measurement theory
  - Sample reliability metrics

Software Quality Metrics

- Applying Measurement to Software
  - Software metrics are measurable properties of software systems, their development and use
  - Wide range of different measures:
    - properties of the software product itself
    - the process of producing and maintaining it
    - its source code, design, tests, etc.
  - Examples:

Number of failures	Number of lines of code per programmer per month
Number of lines of code	Number of failures per 1,000 lines of code
Time to build	Number of decisions per 1,000 lines of code

What are Metrics Good For?

- Reliability and Quality Control
  - Metrics help us to predict and control the quality of our software
  - **Example**: By measuring relative effectiveness of defect detection and removal of various testing or inspection methods, we can choose the best one for our software products

What are Metrics Good For?

- Cost Estimation and Productivity Improvement
  - Metrics help us predict effort to produce or maintain our software, and to improve our scheduling and productivity
  - **Example**: By measuring code production using different languages or tools, we can choose those that give the best results

What are Metrics Good For?

- Quality Improvement
  - Metrics help us to improve code quality and maintainability
  - **Example**: By measuring complexity of our program code, we can identify sections of code most likely to fail or difficult to maintain

Kinds of Metrics

- Three Basic Kinds
  - There are three kinds of software quality metrics: product metrics, process metrics and project metrics
- Product Metrics
  - Product metrics are those that describe the internal and external characteristics of the product itself
  - **Examples**: size, complexity, features, performance, reliability, quality level
  - Most common software metrics are of this kind
  - Product metrics apply regardless of software process
    - but the measurements may be influenced by process!

Kinds of Metrics

- Process Metrics
  - Process metrics measure the process of software development and maintenance, to improve it
  - **Examples**: effectiveness of defect removal during development, pattern of defect arrival during testing, response time for fix
- Project Metrics
  - Project metrics are those that describe the project characteristics
  - **Examples**: number of developers, development cost, schedule, productivity

Measurement Basics

- If You Want to Know, 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 beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of *science*"

Lord Kelvin

Measurement Basics

- ... But Make Sure You Know What You Are Measuring
  - "In truth, a good case could be made that if your knowledge is meager and unsatisfactory, the last thing in the world you should do is make measurements. The chance is negligible that you will measure the right things accidentally."

George Miller

Measurement Basics

- Definition of Measurement
  - Measurement is the process of empirical objective assignment of numbers to entities, to characterize an attribute
- Huh?
  - Entity = an object or event, such as a source program
  - Attribute = a feature or property of an entity, such as the size of the program
  - Objective = based on a well-defined rule whose results are repeatable, such as counting the number of source lines in the program

Measurement Basics

- Definition of Measurement
  - Measurement is the process of empirical objective assignment of numbers to entities, to characterize an attribute
- In Other Words...
  - Each entity is given a number, which tells you about its attribute
  - **Example**: Each source program has a source line count, which tells you about its size

Measurement Basics

- Example Measurements

Entity	Attribute	Measure
Person	Age	Years since birth
Person	Age	Months since last birthday
Source code	Length	# Lines of Code (LOC)
Source code	Length	# Executable Statements
Testing process	Duration	Time in hours from start to finish
Tester	Efficiency	Number of faults found per KLOC
Testing process	Fault frequency	Number of faults found per KLOC
Source code	Quality	Number of faults found per KLOC
Operating system	Reliability	Mean time to failure / rate of failure occurrence

Measurement Basics

- Common Mistakes in Software Measurement
  - It's easy to make mistakes in choosing what or how to measure software characteristics
  - To avoid mistakes, stick to the definition of measurement

1. You must specify both an entity and an attribute, not just one or the other
   - **Example**: Don't just say you are measuring a program, say what property of the program you are measuring
   - **Example**: Don't just say you are measuring the size of the software, say what artifact of the software you are measuring the size of (e.g., source code)
   - Common bad habit outside software engineering

Measurement Basics

- Common Mistakes in Software Measurement

2. You must define the entity precisely
   - **Example**: Don't just say program,
     say program source code
3. You must have a good intuitive understanding of the attribute before you propose a measure for it
   - **Example**: We have good evidence that size is related to number of source lines

Measurement Basics

- Common Mistakes in Software Measurement
  - It is a mistake to propose a measure if there is no clear consensus on what attribute it is characterizing
    - **Example**: Number of defects per KLOC (1000 lines of code) - characterizes quality of code, or quality of testing?
  - It is a mistake to redefine an attribute to fit an existing measure
    - **Example**: If we've measured # defects found this month, don't mistake that as an indicator of code quality

Kinds and Uses of Software Measurement

- Kinds of Measurement
  - Two distinct kinds of measurement
    - Direct and indirect measurement
- Uses of Measurement
  - Two distinct uses for measurement
    - Assessment (the way things are now)
    - Prediction (the way things are likely to be in the future)
  - Measurement for prediction requires a prediction system

Direct Measurement

- Some Direct Software Measures
  - Direct measures are numbers that can be derived directly from the entity without other information
  - Examples:

Length of source code	Measured by number of lines
Duration of testing process	Measured by elapsed hours
Number of defects discovered during the testing process	Measured by counting defects
Effort of a programmer on a project	Measured by person-months worked

Indirect Measurement

- Some Indirect Software Measures
  - Indirect measures are numbers that are derived by combining two or more direct measures to characterize an attribute
  - Examples: (x divided by y)

Programmer productivity	$\frac{\text{Lines of code produced}}{\text{Person-months of effort}}$
Program defect density	$\frac{\text{Number of defects}}{\text{Length of source code}}$
Requirements stability	$\frac{\text{Original number of requirements}}{\text{Total number of requirements}}$
Test effectiveness ratio	$\frac{\text{Number of items covered}}{\text{Total number of items}}$

Predictive Measurement

- Prediction Systems
  - Measurement for prediction requires a prediction system, consisting of:
  1. A mathematical model

     Example: $E = a\ S^b$, where E is the effort to be predicted, S is the estimated size in lines of code, and a and b are constants; if $b = 2$, the model says effort is quadratic in LOC
  2. A procedure for determining the model parameters

     Example: Analyze past project data to determine a and b
  3. A procedure for interpreting the results

     Example: Use Bayesian probability analysis to determine the likelihood that our prediction is accurate within 10%

Reliability Metrics

- Probability of failure on demand (POFOD)
  - The probability that a demand for service from a system will result in a system failure
  - POFOD = 0.001 means that there is a 1/1,000 chance that a failure will occur when a demand is made
- Rate of occurrence of failures (ROCOF)
  - The probable number of failures likely to be observed in a certain time period (e.g., one hour)
  - Reciprocal of ROCOF is the mean time between failures (MTBF)
    - If ROCOF is two failures/hour, MTBF = 30 min.

Reliability Metrics

- Availability
  - The ability of a system to deliver services when requested, or the probability that a system will be operational when a demand is made for service
  - Availability of 0.9999 means that, on average, the system will be available for 99.99% of operating time

Availability	Explanation
0.9	The system is available for 90% of the time. In a 24 hour period, the system will be unavailable for 144 minutes
0.99	In a 24-hour period, the system is unavailable for 14.4 minutes
0.999	The system is unavailable for 84 seconds in a 24-hour period
0.9999	The system is unavailable for 8.4 seconds in a 24-hour period, or roughly one minute per week

Software Measurement

- Software metrics help us understand the technical process that is used to develop a software product
  - The process is measured to be improved
  - The product is measured to increase its quality
- But..
  - Measuring software projects is controversial
  - It is not yet clear which are the appropriate metrics for a software project, or whether people, processes, or products can be compared using metrics

Summary

- Metrics and Measurement
  - Measurement is about characterizing the attributes of entities
    - Can be direct or indirect
    - Can be for either assessment or prediction
- References
  - Sommerville Ch. 23
- Assignment #4
  - due Wednesday