Software Metrics Overview
Measurements & Metrics

- Measurements: Raw numbers.

- Metrics: Derived numbers that:
  - Indicate the extent to which some objective is being achieved.
  - Facilitate cross-comparison.
  - Can serve as the basis for actions to improve achievement of the objective.

- Identifying useful metrics is hard work!
  - Many times, we can’t find any for some objectives.
Measurements for Software

- Size: Lines of code, function points.

- Time and effort for different project activities.

- Defects found, classified by phase occurred, phase found, module, type, severity.

- Failures and when they occurred.

- Staffing, requirements changes, customer satisfaction (survey results), etc.
Metrics for Software

- **Product Metrics**
  - Indicate the quality of the product produced.

- **In-Process Metrics**
  - “Barometers” to indicate whether the process appears to be “working normally”.
  - Useful during the development and maintenance process to identify problems and areas for improvement.

- **Project Metrics**
  - Indicate whether project execution (business aspects) are on track.
Software Metrics – Things to Consider

As you see each metric, think about:

- How useful is it? How would this be used?

- How meaningful is it?

- How easy is it to gather? How much extra work is it for developers to generate the numbers?

- Are there ways to “beat / defeat” this metric?  
  - Can you “make it look good” in ways that don’t achieve the objectives?

- What other metrics do you need to get a balanced picture?
Product Metrics Overview

- **Performance**
  - Lots of measurements, lack of good metrics.
    - AFAIK (“As Far As I Know”) disclaimer applies to lots of these.

- **Reliability**
  - Defect density: Defects per KLOC (“1000 lines of code”).
  - Failure intensity: Number of failures per (hour of) operation.

- **Availability**
  - Uptime %

- **Usability**
  - SUMI score: user survey results, relative to “state-of-the-art”.

- **Evolvability, safety, security.**
  - Metrics are more like measurements, value as indicators debatable.

- **Overall**
  - Customer satisfaction: results of customer surveys
  - Customer reported defects: defect reports per customer-month
In-Process Metrics: Quality

- Reliability growth pattern.
  - Failures during system testing plotted vs. time.
  - Expected: spikes during each release, decrease over time.
  - Magnitude of spike related to significance, volume of changes.

- Pattern of defects found (arrivals) during testing.
  - Test defects found plotted vs. time during testing.
  - Should decrease significantly close to release.
  - Can project “latent defects” (defects left at release) from this.

- Defect density (can be tracked during development as well).
  - Defects per KLOC (can be classified by type, module).
    - Highlights “hot spots”.
  - Post-release defect density (product metric).
    - Strong indicator of effectiveness of testing (if product is used!)
In-Process Metrics: Maintenance

- Backlog Management Index
  - Rate of problem arrivals / rate of closure.
    - Should be close to 1, at least for high severity.

- Responsiveness of fixing.
  - Mean closure time, age of open & closed problems, % late fixes
    - Should stay within target values

- Fix quality
  - Number and % of defective fixes (didn’t work or created new bugs).
In-Process Metrics: Management

- Cost of Quality
  - Total effort on quality assurance activities: testing, reviews, procedures.
  - Should be as low as possible – high may indicate “perfectionism”.

- Cost of poor quality
  - Total effort expended on rework.
  - Should be within range (what if it is “too low”, isn’t that great?).

- Phase containment effectiveness / defect removal effectiveness.
  - What % of the errors were detected within that phase?
    - Shows effectiveness of reviews and other quality procedures.
  - Preferably around 0.7 or so.
    - If it is 0.97, is that good? (It is an opportunity.)
Project Metrics

- Cycletime
  - Elapsed time from requirements to delivery.

- Productivity
  - Size of delivered software / total effort.

- Rate of Requirements Change
  - % of requirements that changed plotted vs. time.
  - High requirements change will affect estimation accuracy, cycletime, quality.
Project Metrics - Continued

- **Estimation Accuracy**
  - % difference between estimated and actual.
  - Can be done for cycle time, effort.

- **Staffing Change Pattern**
  - % of turnover (entered, left) plotted vs. time.
  - High staffing change will impact productivity, quality.
Conclusion

- There do exist a number of metrics that can give a meaningful picture of what is going on in a project.

- By designing a metrics program that uses multiple metrics in conjunction with each other, we can get a balanced picture.

- Most of the metrics come from relatively little raw data: size, effort, defects / failures, timeline data.

- There are metrics that can help to identify problems and areas of improvement, as well as metrics that evaluate results.