A Practical Software Measurement Mechanism

Allen P. Nikora
Autonomy and Control Section
Jet Propulsion Laboratory
California Institute of Technology
Allen.P.Nikora@jpl.nasa.gov

John C. Munson
Computer Science Department
University of Idaho
Moscow, ID 83844-1010
jmunson@cs.uidaho.edu

The work described in this presentation was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The work was sponsored by the National Aeronautics and Space Administration’s IV&V Facility and the JPL Center for Mission Information Systems and Software (CSMISS) Software Engineering Technology Work Area.
Topics

- Measurement Overview
- Fault Measurement and Estimation Mechanism
  - Structural Measurement
  - Fault Identification and Counting
  - Fault Burden Computation
- DARWIN – A Network Appliance for Measurement
- Summary
- References
Measurement Overview

- Measurement is central to any engineering process
- All software design decisions governed by measurable outcomes
- All code development controlled by measurable outcomes
- All software test activity controlled by measures of test activity
- All software reliability decisions quantified
Measurement Overview

Fault content during development can be estimated using measurements of system's structural evolution [Mun98,Niko981].

- Make structural measurements each time a new version of a module is checked into CM repository.
- Analyze raw measurements with respect to a set of measurements from a baseline build
- Compute fault index via principal components analysis [Dillon84]
- Track structural evolution by recording differences between fault indices for subsequent versions of each module comprising the system.
  - Difference is termed fault change
  - Absolute value of difference is termed net fault change
Fault Measurement and Estimation Mechanism

1. **Problem Reports**
   - Extract Repaired Source Files
   - Identify Source Files Repaired

2. **Repaired File IDs**
   - Repaired Source Files
   - Compare Repairs to Faulty Files
   - Fault Regions

3. **Fault Identification and Counting Rules**
   - Extract Faulty Source Files
   - Faulty Source Files

4. **Identify Faults**
   - Fault Identification and Counting Rules

5. **Discovered Faults**
   - Find Initial Fault Occurrence
   - Initial Fault Placement
   - Add fault placement to repository

6. **CM Library**
   - Extract changed source files
   - Most recently changed source files
   - Measure most recently changed source files

7. **Measurement Baseline**
   - Compute fault index
   - Fault indices
   - Place fault indices into repository

8. **Measurements Repository**
   - Add structural measurements to repository
   - module name, revision number, fault count
   - module name, revision number, structural measurements

9. **Compute Fault Burden**
   - Compute Proportional Fault Burden
   - Proportional Fault Burden
   - Develop fault content regression model
   - Compute absolute fault burden
   - Absolute Fault Burden

WOSA 2001
Measurement Mechanism

Attributes

- Two of the three major areas can be automated
- Fault identification and counting is still a manual activity...
  - But is not necessary to get useful information.
  - Proportional fault burdens may be estimated without failure and fault information.
Structural Measurement
What Gets Measured?

Measurement elements for development
- Static source code attributes
- Rate of code change
- Software faults

Measures of design and code quality
- Operational entropy
- Functional entropy

Measurement in software test
- Measures of test efficiency
- Measures of test effectiveness
- Measures of test entropy
Structural Measurement

How Do We Measure?

- Establish a measurement baseline
  - Performed infrequently
  - Must be done at least once at the start of a measurement effort
  - Baseline should also be changed as follows:
    - For a system with multiple releases (e.g., more than 2), change baseline after each release.
    - Re-establish baseline if measurement tools change – experience indicates that no two tools make measurements the same way
  - Establish separate baseline for each programming language
Structural Measurement (cont’d)
How Do We Measure?

- Identify source modules that have changed since the last set of structural measurements was made
  - Measure each time developers check modules into CM library
    - For a large development effort, measurement overhead might become large enough to pose burden on developers
  - At a regular time each day (preferably when few or no developers are on the system), identify all of the modules that have changed since the last time measurements of the system were taken
    - Developers don’t notice measurement overhead
    - Failure of measurement mechanism will not affect developers
Structural Measurement (cont’d)
How Do We Measure?

- Take structural measurements of the identified source modules
  - Measure components that are at the same level of maturity to ensure fault indices have the same meaning
  - CM systems may offer different views of a system
    - components that have completed unit test
    - components that have completed integration test
  - Need to strike balance
    - compare systems of similar maturity, and
    - obtain enough measurements throughout the system's development to constitute a “good” change history
    - Recommend measuring components that have passed unit tests, but have not yet been integrated
Structural Measurement (cont’d)

How Do We Measure?

AR – Anomaly Report
CR – Change Request

AR/CR Tracking Environment

AR/CR Submitted

AR/CR Rejected

AR/CR Waiting Assignment

Development Environment

Dev view

Dev

Dev Complete

Dev Stable view

Subsys Test

Subsys Test Comp

Test view

Integr Build

Integr Test

Test Complete

Release

Measure software and compute fault indices

Development Checkout/checkin

Local Developer Work Areas

CM Extract

Dev Reference Directory

Build

CM Extract

Dev Stable Reference Directory

Build

CM Extract

Integ. Test Reference Directory

Build n.n.n

Build n.n.n Delivery
Structural Measurement (cont’d)

How Do We Measure?

All measurement domains should be represented in measurement

- Prior work indicates measurements fall into following domains [Mun98, Niko981]:
  - Size
  - Structure
  - Style
  - Nesting

- Different languages may have different domains
  - Previous work did not include O-O measurement
Fault Index: the Fault Surrogate

- Varies in direct proportion to software faults
- Used to estimate a module’s fault burden (proportional or absolute)
- Ratio of two module’s fault indices indicates how many more faults one module has than another
- Absolute fault burden can be estimated by using fault index as input to a (regression) model relating the fault index to the number of faults
Fault Standard

A fault standard must be created for all fault recording processes.

A valid fault standard has the properties that:

- All developers will record faults in exactly the same way
- All developers will enumerate faults in exactly the same manner.

Fault standard must be validated by experiment
Fault Identification and Counting

- Identify failures and faults
  - For each failure, identify all source files changed in making repairs
  - Identify individual faults removed from software in response to a failure
    - Compare repaired source files to versions of those source files containing the faults
    - Apply fault identification and counting rules [Niko98, Niko981] to the resulting differences between the two sets of files.
Fault Identification and Counting

- For each fault, identify point at which it was first inserted into the software
  - Search all previous versions of module to identify version in which fault first appeared
  - Measure structural difference between version in which fault initially appeared and immediately preceding version.
  - Develop regression model relating number of faults inserted as a function of structural change.
Fault Identification and Counting

Project management can establish and enforce following policy:

- Do not make repairs and other types of changes to a component at the same time.
- First make the repairs, verify them, and check them into the configuration library.
- Use the repaired version as the basis for enhancing or adding functionality.
Fault Identification and Counting

- Changes made in response to a failure must be separated from requirements changes
  - Failing to do so will lead to unquantifiable noise in the fault measurements
  - Accuracy of regression models will be reduced, and may make their construction impossible.
Fault Identification and Counting

- Some revision control and problem reporting systems have features that will help to implement this policy.
  - The CM system unit of work is a "change package"
    - Change packages created for problem reports, new functionality, or requirements change requests (CR)
    - Work associated with a problem report/CR/new functionality is checked into the change package.
  - Problem reporting system is tied to the CM system such that each new failure report creates a change package in the configuration library
    - Automatically provides a place for developers to submit the repairs
    - Successful enforcement ultimately the responsibility of project management and development teams.
Fault Burden Computation

Proportional Fault Burden

- Compute ratio of its cumulative net fault change to the sum of the cumulative net fault change values for all modules in the system. e.g.
  - the cumulative net fault change for module A is 7,
  - the sum of the cumulative net fault change values for all modules in the system is 140,
  - we would expect module A to have 7/140, or 5%, of the total number of system faults
  - The proportional fault burden of module A would be 5%.
Fault Burden Computation

Absolute Fault Burden
- Use regression model to predict number of faults inserted
  - Inputs: fault change, net fault change
  - Output: number of faults inserted
DARWIN - A Network Appliance for Measurement

DARWIN is an integrated network appliance to capture characteristics of evolving software.

- Requirements
- Fault reports
- Failure reports
- Source code measurements
- Dynamic code measurements
DARWIN as a Database Management Activity
DARWIN as a Management Tool

Management Staff

Multiple Projects

Measurement Database
Summary

Recent work showing that a software system's measured structural is related to its fault content has led to the development of a practical measurement mechanism.

Structural measurement activities can be automated by means of scripts interacting with the revision control system being used for the development effort:
- Scripts run at regularly-scheduled intervals to
  - Identify and measure the modules that have changed since the last time measurements were taken
  - Compute fault indices
- Fault indices alone can be used to estimate the proportional fault burden of a given module at any time.

A network appliance for performing measurement activities has been developed and will be installed at JPL.
- Removes overhead of measurement activities from development platforms; minimizes effect on developers
- Allows multiple projects to use the same measurement capability
  - Consistent measurement allows meaningful comparison of different projects.

WOSA 2001
Summary (cont’d)

- Estimating absolute fault burden requires information obtained by tracing repaired faults back to the version of the module(s) in which they originally appeared
  - Cannot be completely automated
  - Also difficulty of separating changes due to fault repair from changes due to adding or enhancing functionality. Policies can be developed to require that these types of changes be given different types of labels in the configuration library, but software managers and development team leads are ultimately be responsible for enforcing such policies.
References


