We thought that it might be interesting to compare the cost/effectiveness of functional testing and structural testing approaches.

D. Testing Tools

There are many types of testing tools. The basic classification method is to categorize tools into static testing tools, dynamic testing tools and test management tools. Following are the tool types categorized:

* Basic Tool (symbol cross reference, symbolic debugger)
* Code Auditor
* Static Allegation Checker (static analyzer)
* Symbolic Evaluator
* Self-metric Instrumenter
* Assertion Processor
* Test Data Generator
* Test File Generator
* Test Harness (driver/stub generator, test description language, etc.)
* Execution Verifier
* Output Comparator
* Test Coverage Measurement Tool

Note: We have selected only one tool from the basic tool category, the symbolic debugger. The reason is that almost every environment will have some type of symbolic debugger; but probably will not have a symbolic evaluator, test data generator or test description language, tools which are not widely used as yet.

PREDICTING COST-OF-CHANGE

FROM DESIGN STRUCTURE METRICS

Prepared by:
Dennis George
David Gustafson
Sallie Henry
David Hutchens
Dennis Kafura
John Sayler (Editor)

Executive Summary

The fundamental pragmatic objective of metrics for software is to provide means for controlling the costs of software projects. Software systems are always changing: as the requirements or the specifications for the systems change over time, as bugs are fixed, as customers desire enhancements to these systems. These changes incur substantial costs and schedule slips. Software metrics should provide software developers and managers the means to reduce, or at least control, such changes by minimizing and localizing the complexity of these systems.

To date few of the software metrics have been correlated with the costs associated with software projects. Most of the work on software metrics has focused on determining the relationship between a given metric and the complexity of a software system in the hopes that understanding the basis and extent of complexity of a software system would allow control of the cost of producing such systems.

This is a proposal for an experiment to relate software metrics to software project observables, in particular to the cost associated with changing a software system.

Specifically, this experiment will measure the total cost of change of software as a function of various metrics on the design of that software system.

The result of this experiment will be to test the validity of the assumption that cost of change can be directly related to various metrics on the structure of software systems. In addition, those projects participating in this experiment will receive numerous side benefits: an augmented ADA language providing additional structuring tools, the use of a sophisticated configuration management data base which will allow the collection and analysis of statistics regarding the structure of the software, and the use of quantitative tools to aid the design process.

The total cost of this experiment will be $444,000 spread over the two year duration of the experiment. Three separate development

projects will be monitored and evaluated during the course of the experiment.

1.0 Project Definition

1.1 Project Objectives

The objective of this experiment is to relate the costs of software changes during the design, implementation, and testing phases with respect to some metrics on the structure of the software system.

1.2 Cost of Change

The hypothesis of this experiment is that the cost of change is directly related to the structure of the design of a software system as measured by current design metrics. These metrics include the number of modules in the system and their connectivity as well as measures on the code itself, e.g., cyclomatic complexity and software physics computations.

1.3 Importance of Experiment

The cost of producing software systems continue to increase. In order to predict and control these costs, it is necessary to understand the cost of changing software systems since these changes are responsible for the major costs over the life cycle of a software project. It has been well established that small improvement in the design of a system can produce large savings over the life cycle of such systems. In particular, one of the goals of this experiment is to provide a basis for deciding an optimal time to redesign systems which have been heavily modified. As shown by Belady and Lehman [Belady], as systems are continually modified, the cost of maintaining those systems at a given level of performance becomes exponential with time. Thus, a model relating cost of change to system complexity would perhaps extend the useful lifetime of such systems.

1.4 Overview of Experiment

1.4.1 Standard Environment

In order to provide a homogeneous experimental framework, the experiment will be run on three similar real world projects. Further, each of these projects will be implemented in the same high level language, ADA; data will be collected at the same specific check points on each of the projects, each project will use a standard configuration management system and each will standard representation for design information.

1.4.2 Automated Data Collection

Both the data for evaluation of the design metrics and the data for comparison with other metrics, will be automatically collected. In fact, the source for this data will be the ADA source code itself. Analysis programs for these tasks will be provided by the experimental team.

1.4.3 Data Analysis

The analysis team will perform an initial analysis at each of the intermediate check points of the project life cycle in order to ensure the completeness and accuracy of the data collection and provide feedback to the project team. At the completion of the three projects, the analysis team will perform the following analyses:

- Validate or invalidate the hypothesis that cost of change is directly related to design metrics used in this experiment.
- Validate or invalidate the hypothesis that cost of change is or is not directly related to the source code measurements that were gathered.
- Perform a cross correlation of cost of change with all of the measurements that were taken during the experiment across all three projects.

2.0 Context Information

The following constraints will be levied upon each of the three software projects of this experiment:

- Each project will be budgeted at approximately $400,000.
- Each project will be monitored from the initial design through the acceptance testing phase.

- Each project will use a configuration management system to be provided by the analysis team.
- Each project will spend less than one calendar year from initial design through the acceptance testing phase.
- Each project must be in a similar application area.
- Each project will be implemented in the ADA programming language.

While a similar design methodology will be preferable across all projects, it is not a requirement of this experiment.

2.2 Characteristics of the Development Group

The following are (ideal) requirements of the development group working on each of the projects.:

- The development teams should be similar in size.
- Each development team should be knowledgeable in both the problem area and in the use of the ADA programming language.
- Each development team should be willing to collect statistics required by this experiment, and be willing to use and evaluate the configuration management system supplied by the analysis team.

2.3 Characteristics of the Analysis Group

Each member of the analysis team should be knowledgeable about software metrics, the ADA programming language, and software project management techniques and practices.

3.0 Experimental Plan

3.1 Environment

As noted above each of the three projects should involve similar applications preferably under Department of Defense contracts, since the ADA language is being sponsored by the Department of Defense. Each of these projects should be of similar size, use the same language (ADA), the same design syntax, and a standard configuration management system to control the base line products of development and to ensure consistency of the change process across projects. The analysis team will provide two tools for use by the development project. The first is a design pre-processor for ADA which will allow design descriptions, and secondly a configuration management system which will provide the basis for change control and collection of statistics on changes in the system.

3.2 Project Team Responsibilities

The development team, using the CM system will supply, on a regular basis, change reports including:

- Expected size of change (number of lines, modules affected).
- Type of change (enhancement, bug fix).
- Actual size of change (number of lines, modules affected, and actual manpower required).

The development team will also supply to the analysis team base line products at the time of initial design, final design, implementation, end of integration tests, and end of acceptance test. These base line products are the documentation and code of the system, as well as test plans and test results at each of the above times. The development team shall also provide the analysis team with a report on each of the applicable metrics at each of the above times. This report will be generated by a program supplied by the analysis team, which analyzes the CM data base and the ADA source text. The metrics to be reported on are:

- Binding amongst modules.
- Connectivity amongst modules.
- Software Science Metrics (number of operators, number of operands, and derived statistics from them).
- Counts of

- modules
- procedures
- calls

3.3 Post Experiment Evaluation

Upon conclusion of the three development projects, the following analyses will be undertaken by the analysis team:

- Factor analysis to determine the metrics' effects upon the cost of change.
- Multiple regression to judge the predictive capability of each of the metrics.
- Determination of relationships amongst the metrics.

4.0 Discussion of Experiment

4.1 Cost Reduction

A number of decisions were made in the genesis of this proposal in order to reduce the cost of set up, operation, and analysis of this experiment. In particular:

- Provisions for automatic collection and analysis of data.
- Only simple measurements are being taken:
    - information flow (connectivity)
    - number of "packages"
    - number of types
    - data binding (potential, and actual)
- Time span of experiment, from initial design up to product release from Quality Assurance group.

4.2 Experimental Control

In order to establish a well defined experiment, measurements will be taken at only a few specific well defined checkpoints or milestones. In particular, these milestones are the initial design, the final design, the beginning of integration testing, the beginning of acceptance testing, and the completion of acceptance testing.

4.3 Concerns and Risks

This section details some concerns and risks associated with this experiment and our resolution of them.

4.3.1 Design Specification Not Usable

Since the representation of design of a software system is not consistent across the computing community, we have elected to capture essential elements of the design in notations to be added to the ADA programming language. These simple additions will allow us to analyze automatically various characteristics of the design without having to resort to a multitude of design documents.

4.3.2 Change Data Inconsistent/Unusable

It is extremely difficult to gather detailed data concerning software changes. While one would like to track all costs associated with a specific change request to a system, this goal is unrealizable. For example, it is frequently the case that a given change to a software system may introduce new faults in that system which would later on require changes to that system. Therefore, we have elected to gather only the gross cost data which is easily available, i.e., those resources devoted exclusively to enhancements or bug fixes of the system. Our secondary source is the project CM data base, which will contain statistics on the changes to each of the modules of the code.

4.3.3 Current Metrics Do Not Predict Cost of Change

The null hypothesis of this experiment is that cost of change is not related to any of the current metrics that we are using in this experiment. If the null hypothesis is valid, the experiment will still provide:

- Greater understanding of the change process.
- Insight into the interrelationships of the measurements gathered during the experiment.
- Possible suggestions for new metrics.

5.0 Benefits of the Experiment

The proposed experiment provides many benefits both direct and indirect to the funding agency. The major benefits are outlined below.

5.1 Direct Benefits To The Project Team

- The development of Configuration Management tools for the ADA environment.
- The development of a Configuration Management data base for the further analysis of metrics (perhaps retrospectively).
- The development of metric evaluation and analysis tools.
- The development of quantitative design aids.
- The development of extensions to ADA, which will improve its usefulness as a design specification language.
- By obtaining feedback early on in the development process on the cost of change, it might be possible for the development team to choose alternatives which reduce the overall cost for the rest of the life cycle of the project.

5.2 Further Application of the Experimental Plan

This experiment provides a template for the evaluation of relationships between new metrics and the cost of change, i.e., the same experimental framework may be reused to evaluate new metrics.

- This experiment provides a framework for data collection beyond that required for the proposed evaluation.
- This experiment itself is applicable in other environments with other languages (non-DOD, non-ADA); the experiment calls only for a constant environment and a constant language.

5.3 Software Development Environment

The software development environment required for this experiment is highly portable, and provides a number of benefits in and of itself:

- The environment requires a Configuration Management tool which can be used to extract that data required for metric evaluation.
- The software development environment will support fixed base lines of data collection. The scheduling and monitoring of these base lines have proven highly valuable to project management.
- The software development will provide the ability to monitor changes in the software during the development process. This enables project management to track the stability of the evolving system.
- The design representation is precise enough to allow computation of the target design metrics. While simple, these metrics provide insight into the complexity of the evolving system.
- The data collection of change will be integrated into the development environment, again allowing management visibility into the stability of the system.

6.0 Costs

6.1 Costs for Entire Experiment

Configuration Management Tool	
Purchase	$ 25 K
Modify (1/2 man year)	$ 25 K
ADA preprocessor mods (3 man months)	$ 12 K
Inter-module analysis tool (1 man year)	$ 50 K
Intra-module analysis tool (3 man months)	$ 12 K
Analysis Tools (DB analysis) (1 man year)	$ 50 K
TOTAL:	$174 K

6.2 Costs for Individual Projects

10% of 3 projects ($400 K range projects) for cost overhead to development team	$120 K
6.3 Experimenters' Analysis Cost (3 man years)	$150 K
TOTAL:	$444 K

Bibliography

Basili, V. and Turner, A. J., "Iterative Enhancement: A Practical Technique for Software Development", IEEE-TSE (1,4), December, 1975.

Belady, L. A. ,and M. M. Lehman, "A Model of Large Program Development", IBM Systems J., vol. 15, No. 3, 1976, pp. 225-252.

Bonanni, L. E. and Salemi, C. A., Source Code Control System User's Guide, Bell Laboratories, Piscataway, N. J.

Bryan, W., Chadbourne, C., and Siegel S., Tutorial: Software Configuration Management, IEEE Computer Society, IEEE Catalog No. EHO 169-3, 1980.

Halstead, M. H., Elements of Software Science Elsevier, 1977.

Henry, S. M., Information Flow Metrics for the Evaluation of Operating Systems' Structure. Ph. D. Thesis.Iowa State University. August,1979.

Henry, S. M. and Kafura, D. G., "Software Structure Metrics Based on Information Flow", To appear in IEEE Trans. on Soft. Eng., 1981.

Kafura, D. G. and Henry, S., "A Viewpoint on Software Quality Metrics: Criteria,Use and Integration", To appear in Journal of Systems and Software, 1981.

Kafura, D. G. and Henry, S., "The Evaluation of Operating Systems' Structure Using Quantitative Software Metrics, "Technical Report.Iowa State University, 1981.

Kafura, D. G. ,Henry, S. M. ,and Harris, K., "On the Relationships Among Three Software Metrics", Proceedings of the 1980 Sigmetrics Symposium on Software Quality Assurance.

McCabe, T. J., "A Complexity Measure", IEEE Transactions on Software Engineering (2,4), December, 1976.