N92-19429

EARLY EXPERIENCES BUILDING A SOFTWARE QUALITY PREDICTION MODEL

W. W. Agresti, W. M. Evanco, M. C. Smith

The MITRE Corporation

Abstract

Early experiences building a software quality prediction model are discussed. The overall research objective is to establish a capability to project a software system's quality from an analysis of its design. The technical approach is to build multivariate models for estimating reliability and maintainability. Data from twenty-one Ada subsystems have been analyzed thus far to test hypotheses about various design structures leading to failure-prone or unmaintainable systems. Current design variables highlight the interconnectivity and visibility of compilation units. Other model variables provide for the effects of reusability and software changes. Reported results are preliminary because additional project data is being obtained and new hypotheses are being developed and tested. Current multivariate regression models are encouraging, explaining 60-80% of the variation in error density of the subsystems.

Introduction

A typical shortcoming of large-scale software development is the uncertainty concerning the consequences of design decisions until much later in the development process. Greater capability is needed during the design activity to assess the design itself for indications that, when implemented, the resulting system will have particular quality characteristics. This paper discusses the early experiences in a research project to evaluate the quality of Ada designs.

The research objective is to test the hypothesis that Ada software quality factors can be predicted during design. The technical approach is build

- This research was sponsored by The MITRE Corporation under the Mission Oriented Investigation and Experimentation (MOIE) program

- Authors' Address: The MITRE Corporation, 7525 Colshire Drive, McLean, Virginia 22102

Proceedings of the Fifteenth Annual Software Engineering Workshop, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, November 1990

multivariate models to estimate reliability and maintainability using characteristics of the design. The orientation to Ada is due to its prevalence in mission-critical systems under development and its ability to serve as a notation for software design. This role for Ada as a design language has been recognized as American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE) Standard 990-1987.

Previous research has established relationships between design or code characteristics and quality factors [1, 2]. A recent system-level design measure, incorporating both control flow and data flow in FORTRAN systems, shows a strong correlation with reliability [3]. The COnstructive QUAlity MOdel (COQUAMO) is being developed to estimate software quality, basing its estimate on the observed quality of previous projects [4].

Quality Estimation Models

Building the estimation models depends on having access to three classes of project data:

- Design, expressed in Ada, from which design characteristics can be extracted
- Environmental factors that influence the quality of the software but cannot be deduced from the design artifact itself – for example, level of reuse or volatility of changes to the software
- Data characterizing what resulted when the design was implemented, tested, and fielded – for example, reported errors and effort to maintain the software

The basic form of the estimation models is shown in Figure 3. Independent, explanatory variables in the models represent architectural design characteristics. Additional explanatory variables account for the effects of the organization and its development process. By the error term in the model, we will learn if we have been successful in explaining the variation in quality factors by using design characteristics and environmental factors.

Ada Design Representation

One of the first issues we faced was developing an effective representation from which we could extract design characteristics. Our interest was in the static architecture of units in subsystems, not in the arrangement of statements within a unit. We viewed the subsystem as being composed of design units and relations as illustrated in Figure 4. Our analysis of Ada identified several candidates to serve as design units in our structure: program units, compilation units, and library units. All three units have participated in our model building, but compilation units have been particularly useful as a structural unit because they also serve as the unit of observation for reporting errors and changes.

Our Ada analysis identified fifteen kinds of Ada compilation units: generic package specification, generic package body, and so on as shown in Figure 5. The compilation units are further divided into library units and secondary units (see Figure 5) and serve as the design unit nodes in the graphical Ada design structures in Figure 4. The nodes are related to one another by the design relations of context coupling, specification/body, parent/subunit, and generic template/instantiation. These design units and design relations comprise our representation of static Ada architecture. This Ada design representation is discussed further in [5].

Software Project Data

Project data used in the analysis is summarized in Figure 6. The twenty-one subsystems included 2,143 compilation units. Declarations are listed in Figure 6 because they play a key role in the hypotheses we are examining. One of our underlying themes is that a developer does not declare objects, types, subprograms, etc. unless they are needed. Thus, the number and distribution of these declarations is of interest to us in characterizing designs.

Our models attempt to explain variation in quality, and Figure 6 shows our project data exhibits significant variation. The data was obtained from the National Aeronautics and Space Administration (NASA)/Goddard Space Flight Center (GSFC) Software Engineering Laboratory (SEL). Reliability is measured as error density and varies in the range 1.4 - 17.0 errors per thousand source lines of code for the twenty-one subsystems. Maintainability varied across the subsystems as 26 - 89% error corrections requiring less than or equal to one hour to complete.

Hypotheses About Design Structure

We are pursuing simple hypotheses about design decision making, the resulting design artifact, and the influences of design on reliability and maintainability . Figure 7 outlines an example of a general hypothesis that excessive context coupling contributes to errors. The rationale is that greater arc density in the directed graph in Figure 7 increases the likelihood of introducing an error, because a greater number of relationships must be understood.

Figure 8 expands on library unit B of Figure 7. We have found that a library unit aggregation – a library unit and its declarative scope – to be a very effective unit of granularity for our analysis of Ada designs. Figure 8 shows a second level of design decision making that occurs inside a library unit aggregation. We are interested in whether the designer has made any effort to manage the visibility of the 103 declarations that have been imported into

unit B. By having 100 of the declarations brought in (via a "with" clause) to the specification, they are visible throughout the other units in the library unit aggregation, cascading through the structure. We don't know which of these declarations are used by each unit, but we want to record their visibility to the other units in the library unit aggregation. The measure of cascaded imports in Figure 8 takes visibility into account: 100 of the imports are visible to five units (=> 500 cascaded imports) and three of the imports are visible to two units (=> 6 cascaded imports), for a total of 506 cascaded imports.

Preliminary Results of Statistical Analyses

Figure 9 summarizes the variables that have been introduced into our models thus far. Context coupling and visibility follow the example in Figure 8, while import origin records the fraction of declarations imported from within the subsystem. Two environmental factors have been analyzed to date: volatility captures the relative number of changes that have been made in the subsystem, and custom code is the percentage of new and extensively modified code used in the subsystem. Custom code is essentially the opposite of reuse.

The preliminary model explaining variation in total error density (Figure 10) includes the explanatory variables of context coupling, visibility, and volatility. In this model and other similar regression analyses we have conducted, the coefficients have the expected signs: error densities increase as context coupling, visibility, and volatility increase.

Because of our interest in architectural design decisions, we conducted additional regression analyses which concentrated on errors occurring during system and acceptance testing. Our rationale was that, by eliminating errors reported during unit testing (and, therefore, more likely to be errors in implementing a single unit), we were reflecting more strongly the architectural (inter-unit) relationships. Figure 11 summarizes a model to estimate errors reported during system and acceptance testing. Again, context coupling and visibility are included as explanatory variables. Now, however, *custom code is a significant factor in explaining the variation in error density.* The explanatory power (as indicated by the coefficient of determination) is stronger for the model in Figure 11.

Summary

Early results in building estimation models for reliability and maintainability are encouraging. We have developed representations for the static structure of Ada systems using compilation units and library unit aggregations, allowing us to test hypotheses about the effects of different structures on reliability and maintainability . Context coupling measures consistently figure strongly in the multivariate regression analyses we have

conducted. Visibility and import origin measures provide further refinement. The models show strong effects of volatility and custom code on reliability .

We stress the preliminary nature of the quantitative results, based as they are on twenty-one Ada subsystems. We look forward to continuing to explore hypotheses with additional data, leading to the development of more robust models that can be subjected to validation.

Acknowledgement

We acknowledge the cooperation of Mr. Frank McGarry and Mr. Jon Valett of the NASA/GSFC SEL in allowing us to use SEL data for this research.

References

1. T. J. McCabe and C. W. Butler, "Design Complexity Measurement and Testing," Communications of the ACM, December 1989, pp. 1415-1425.

2. S. M. Henry and C. A Selig, "Predicting Source-Code Complexity at the Design Stage," IEEE Software, March 1990, pp. 36-44.

3. D. N. Card and W. W. Agresti, "Measuring Software Design Complexity," Journal of Systems and Software, March 1988, pp. 185-197.

4. B. Kitchenham, "Measuring Software Quality," Proceedings of the First Annual Software Quality Workshop, Rochester, New York, 1989.

5. W. W. Agresti, W. M. Evanco, and M. C. Smith, "An Approach to Software Quality Prediction from Ada Designs," Proceedings of the Second Annual Software Quality Workshop, Rochester, New York, 1990.

Early Experiences Building a Software Quality Prediction Model

W. W. Agresti, W. M. Evanco, M. C. Smith
MITRE Washington Software Engineering Center
28 November 1990

MITRE

Research Project Overview

- Objective:
  - Test the hypothesis that Ada software quality factors can be predicted during design
- Technical Approach:
  - Build multivariate models to estimate reliability and maintainability
  - Use characteristics of the software design captured in Ada design language

Basic Form of the Estimation Models

Reliability $= f_1(DC_1, DC_2, \ldots, EF_1, EF_2, \ldots \mid a_1, a_2, \ldots) + e_1$

Maintainability $= f_2(DC_1, DC_2, \ldots, EF_1, EF_2, \ldots \mid b_1, b_2, \ldots) + e_2$

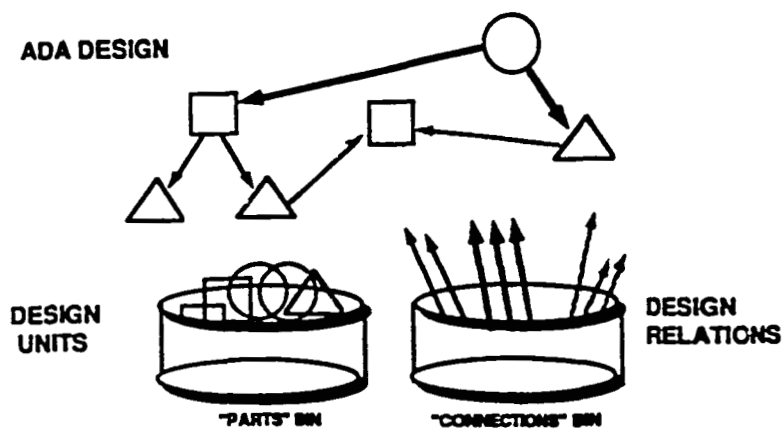
where -

- DC_i : design characteristic variable
- EF_i : environmental factor variable
- a_i, b_i : model parameters
- e_i : error term (unexplained variation)

MITRE

Figure 3

Representing Ada Design Structure



MITRE

Figure 4

"Parts" in Ada Static Structure

15 Compilation Units in Ada as Library Units (L) or Secondary Units (S)

	Specification	*Body*	*Subunit*	*Instantiation*
Generic Package	L	S	S	L
Package	L	S	S	N/A
Generic Subprogram	L	S	S	L
Subprogram	L	L/S	S	N/A
Task	N/A	N/A	S	N/A

MITRE

Figure 5

Profile of Current Project Data

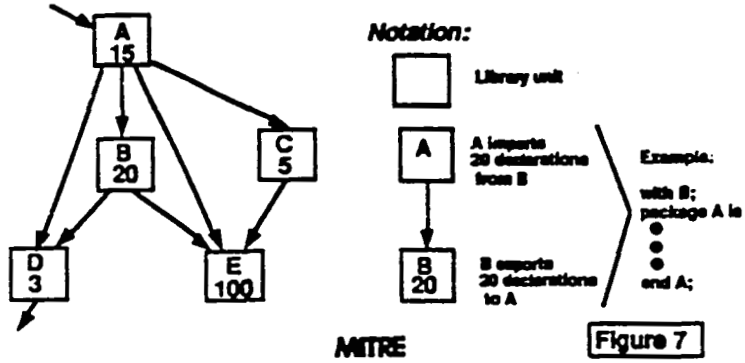
- Twenty-one subsystems from NASA/GSFC SEL:
  - Interactive, ground support software for flight dynamics and telemetry processing applications
  - 183 K non-comment, non-blank source lines of Ada (KSLOC)
  - 601 Library Units
  - 2,143 Compilation Units
  - 29,849 Declarations
- Variation in dependent variables:
  - Reliability range: 1.4 - 17.0 errors/KSLOC
  - Maintainability range: 26 - 89 % "easy" fixes (requiring ≤ 1 hour)

MITRE

Figure 6

Exploring Simple Hypotheses About Design Structure

- Example of a general hypothesis: Excessive context coupling contributes to complexity which, in turn, contributes to errors
- Example of context coupling to access the resources of library units:

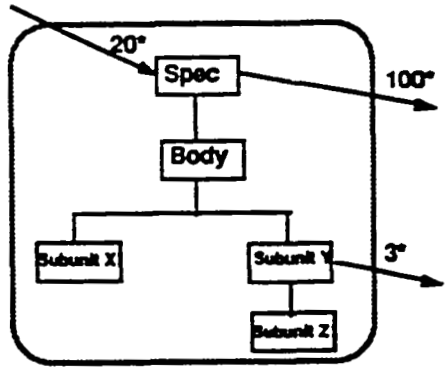


MITRE

Figure 7

Inside a Library Unit Aggregation to Show Imported and Exported Declarations

A Library Unit Aggregation



* number of declarations

Static Measures:

imports = 103
exports = 20
cascaded imports = 506

MITRE

Figure 8

Model Variables

- Design Characteristics:
  - Context Coupling: # imports / # exports
  - Visibility: # cascaded imports / # imports
  - Import Origin: # internal imports / # imports
- Environmental Factors:
  - Volatility: # changes / # library units
  - Custom Code: % new and extensively modified code

MITRE Figure 9

Preliminary Model for Reliability: Total Errors (errtot)

- Dependent variable: TOTERRSL = errtot / KSLOC

$$\ln(\text{TOTERRSL}) = \underset{(.36)^*}{.65} + \underset{(.11)}{.27}\ln(X_1) + \underset{(.16)}{.05}\ln(X_2) + \underset{(.08)}{.27}\ln(X_3)$$

X_1 = context coupling
X_2 = visibility
X_3 = volatility

adjusted $R^2 = .72$

* Standard deviation of the parameter estimate

MITRE Figure 10

Preliminary Model for Reliability: System and Acceptance Testing Errors (errsa)

- Dependent variable: SYACERRSL = errsa / KSLOC

$$\ln(\text{SYACERRSL}) = .77 + .19 \ln(X_1) + .07 \ln(X_2) + .97 \ln(X_3)$$
$$(.65)^* \quad (.18) \qquad (.21) \qquad (.24)$$

X_1 = context coupling
X_2 = visibility
X_3 = custom code

adjusted $R^2 = .78$

* Standard deviation of the parameter estimate

MITRE

Figure 11

Current Research Activity

- Continue to develop process models and hypotheses about design decision-making and design structures -- and their relationships to reliability and maintainability
- Explore classification trees and other alternative analytical methods
- "Call for Ada Project Data" -- to test hypotheses and calibrate multivariate models

MITRE

Figure 12

VIEWGRAPH MATERIALS

FOR THE

W. AGRESTI PRESENTATION

6269-0

Early Experiences Building a Software Quality Prediction Model

W. W. Agresti, W. M. Evanco, M. C. Smith
MITRE Washington Software Engineering Center
28 November 1990

MITRE

Research Project Overview

- **Objective:**
  - **Test the hypothesis that Ada software quality factors can be predicted during design**
- **Technical Approach:**
  - **Build multivariate models to estimate reliability and maintainability**
  - **Use characteristics of the software design captured in Ada design language**

W. Agresti
MITRE

MITRE

Figure 2

Basic Form of the Estimation Models

$$\text{Reliability} = f_1(DC_1, DC_2, \ldots, EF_1, EF_2, \ldots \mid a_1, a_2, \ldots) + e_1$$

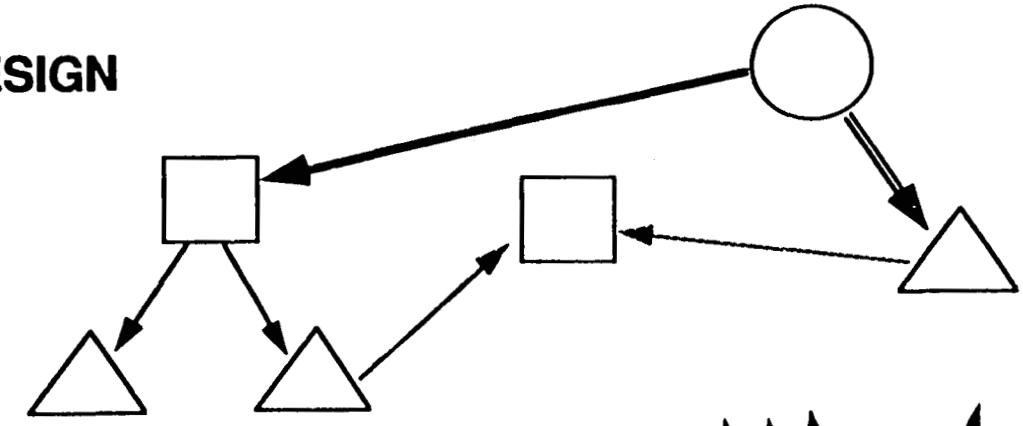
$$\text{Maintainability} = f_2(DC_1, DC_2, \ldots, EF_1, EF_2, \ldots \mid b_1, b_2, \ldots) + e_2$$

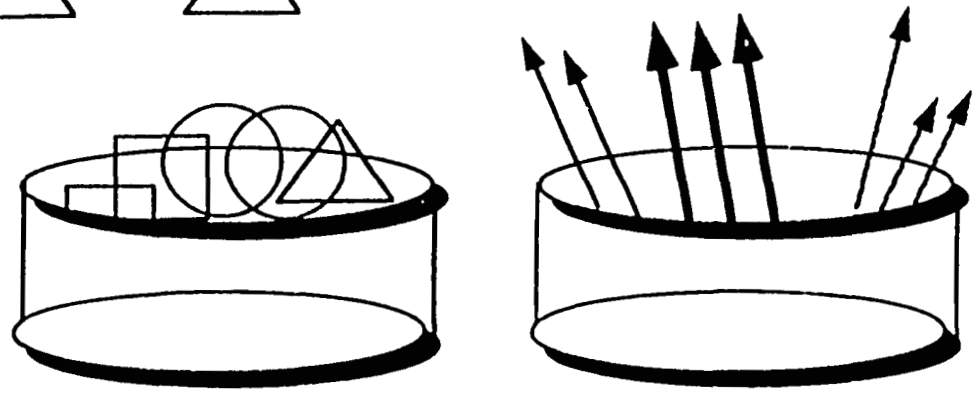
where -

- DC_i : design characteristic variable
- EF_i : environmental factor variable
- a_i, b_i : model parameters
- e_i : error term (unexplained variation)

W. Agresti
MITRE

Representing Ada Design Structure





W. Agresti
MITRE

MITRE

Figure 4

"Parts" in Ada Static Structure

15 Compilation Units in Ada as Library Units (L) or Secondary Units (S)

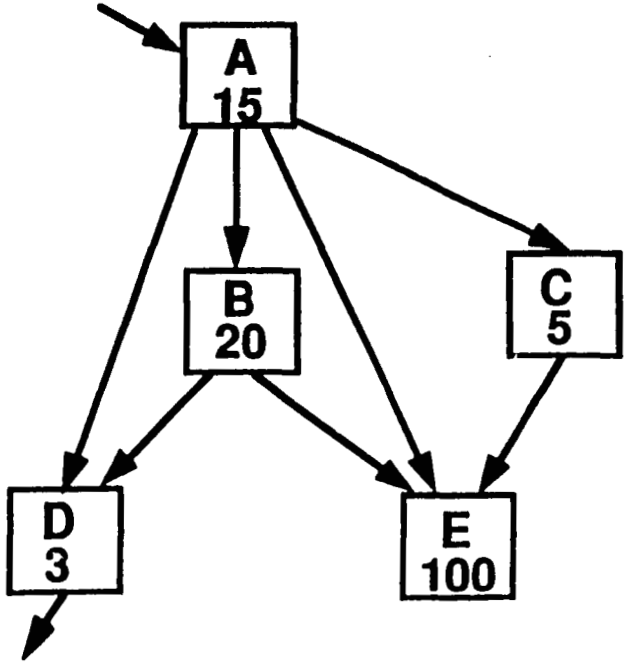
	Specification	*Body*	*Subunit*	*Instantiation*
Generic Package	L	S	S	L
Package	L	S	S	N/A
Generic Subprogram	L	S	S	L
Subprogram	L	L/S	S	N/A
Task	N/A	N/A	S	N/A

Profile of Current Project Data

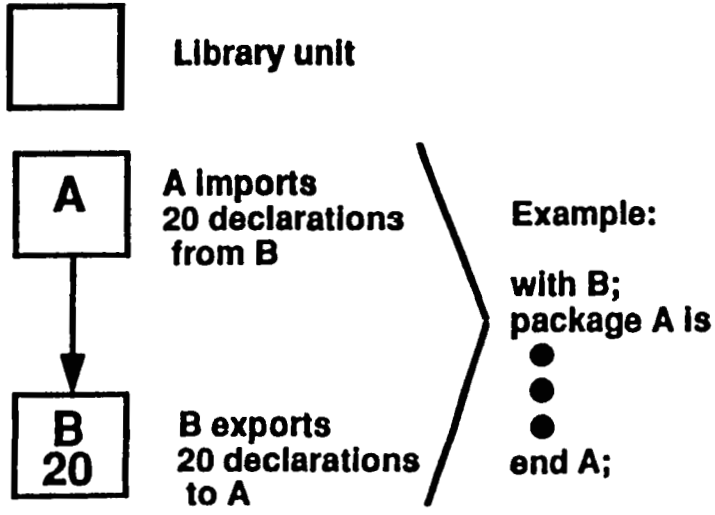
- Twenty-one subsystems from NASA/GSFC SEL:
  - Interactive, ground support software for flight dynamics and telemetry processing applications
  - 183 K non-comment, non-blank source lines of Ada (KSLOC)
  - 601 Library Units
  - 2,143 Compilation Units
  - 29,849 Declarations
- Variation in dependent variables:
  - Reliability range: 1.4 - 17.0 errors/KSLOC
  - Maintainability range: 26 - 89 % "easy" fixes (requiring ≤ 1 hour)

Exploring Simple Hypotheses About Design Structure

- Example of a general hypothesis: Excessive context coupling contributes to complexity which, in turn, contributes to errors
- Example of context coupling to access the resources of library units:



Notation:

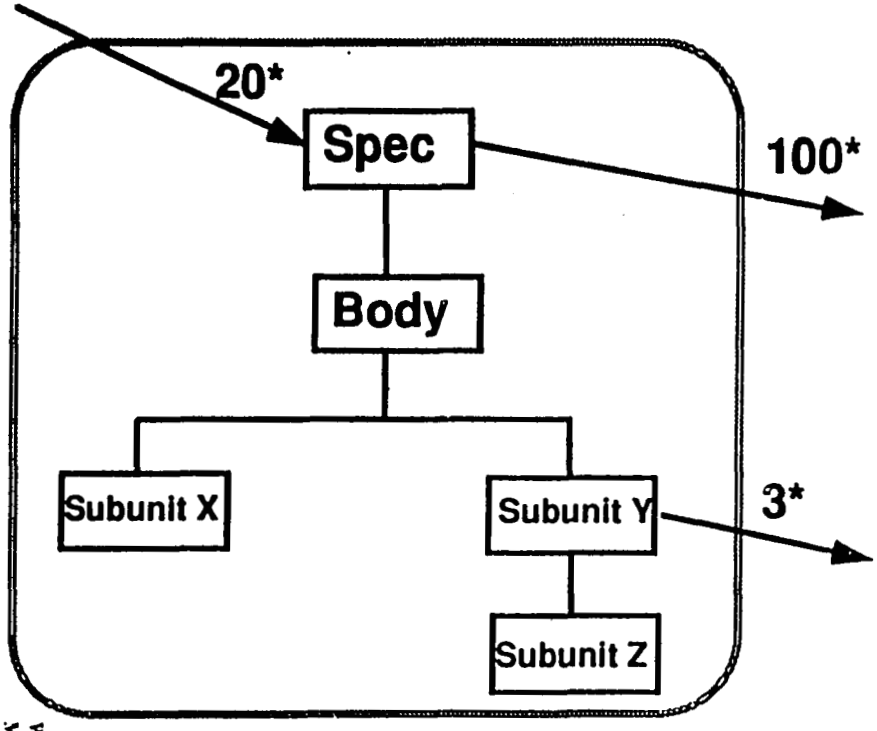


MITRE

Figure 7

Inside a Library Unit Aggregation to Show Imported and Exported Declarations

A Library Unit Aggregation



Static Measures:

 # imports = 103
 # exports = 20
 # cascaded imports = 506

*** number of declarations**

MITRE

Figure 8

Model Variables

- **Design Characteristics:**
  - **Context Coupling:** # imports / # exports
  - **Visibility:** # cascaded imports / # imports
  - **Import Origin:** # internal imports / # imports
- **Environmental Factors:**
  - **Volatility:** # changes / # library units
  - **Custom Code:** % new and extensively modified code

W. Agresti
MITRE

MITRE

Figure 9

Preliminary Model for Reliability: Total Errors (errtot)

- **Dependent variable: TOTERRSL = errtot / KSLOC**

$$\ln(\text{TOTERRSL}) = .65 + .27 \ln(X_1) + .05 \ln(X_2) + .27 \ln(X_3)$$
$$(.36)^* \quad (.11) \qquad (.16) \qquad (.08)$$

X_1 = context coupling

X_2 = visibility

X_3 = volatility

adjusted $R^2 = .72$

* Standard deviation of the parameter estimate

MITRE

Figure 10

Preliminary Model for Reliability: System and Acceptance Testing Errors (errsa)

- **Dependent variable: SYACERRSL = errsa / KSLOC**

$$\ln(\text{SYACERRSL}) = \underset{(.65)^*}{.77} + \underset{(.18)}{.19}\ln(X_1) + \underset{(.21)}{.07}\ln(X_2) + \underset{(.24)}{.97}\ln(X_3)$$

X_1 = context coupling

X_2 = visibility

X_3 = custom code

adjusted $R^2 = .78$

W. Agresti
MITRE

* Standard deviation of the parameter estimate

MITRE

Figure 11

Current Research Activity

- Continue to develop process models and hypotheses about design decision-making and design structures -- and their relationships to reliability and maintainability
- Explore classification trees and other alternative analytical methods
- "Call for Ada Project Data" -- to test hypotheses and calibrate multivariate models