

## **Change and Defect Models and Metrics**

### **Change and Defect Models and Metrics**

These models and metrics capture information about modifications to the software system documents

They can be categorized in a variety of ways, usually on a nominal scale

Models are typically presented as distributions of the various classes of changes

Change models and metrics can be used to

- characterize the project
- provide insight into the selection of the appropriate processes
- provide insight into the product

## Change and Defect Models and Metrics

---

### Change Data Classes

Changes can be categorized by

**purpose** e.g., enhancement, adaptive, corrective, preventive

**type** e.g., requirements, specification, design, architecture, planned enhancements, insert/delete debug code, improve clarity, optimize: space or time, feature, enhancement, bug

**cause** e.g., market/external and internal needs

**size** e.g., number of lines of code or number of components affected,

**deposition** e.g., rejected as a change, not relevant, under consideration, being worked on, completed, saved for next enhancement

**level of document changed** e.g., changes back to requirements document

**number of customers affected** e.g., effects certain customer classes

## Change and Defect Models and Metrics

---

### Issues associated with nominal scale data

Changes are usually classified on a nominal scale, based upon a classification scheme

It should be easy to classify an object into its nominal category

The categories need to be well defined

The categories need to be orthogonal

The categories need to have intuitive meaning for the classifier

## **Change and Defect Models and Metrics**

---

### **Example Definitions**

Enhancement change: an addition to the original requirements observable to the user, e.g., adding a new functional capability

Adaptive change: modification based upon the environment in which the system lives but does not add new capability observable to the user, e.g., modifying the system to interact with a change to the hardware or operating system

Corrective change: a modification made to correct a problem with the existing software solution, e.g., fixing a bug

Preventive change: a modification that will make the system easier to modify in the future, e.g., restructuring the modules

## **Change and Defect Models and Metrics**

---

### **Sample Change Metrics**

number of enhancements per month

number of changes per line of code

number of changes during requirements

number of changes generated by the user vs. internal request

number of changes rejected/ total number of changes

Change Report history profile

## Change and Defect Models and Metrics

### Defect Definitions

#### Errors

Defects in the human thought process made while trying to understand given information, to solve problems, or to use methods and tools

#### Faults

Concrete manifestations of errors within the software

- One error may cause several faults
- Various errors may cause identical faults

#### Failures

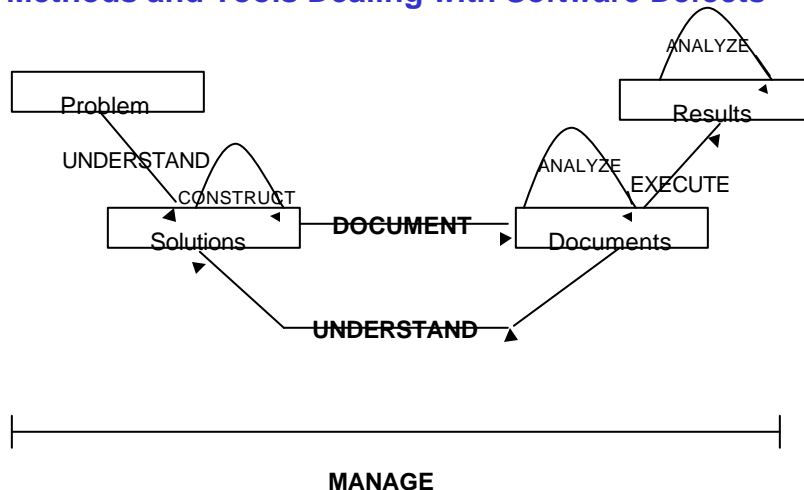
Departures of the operational software system behavior from user expected requirements

- A particular failure may be caused by several faults
- Some faults may never cause a failure  
(difference between reliability and correctness)

IEEE/Standard

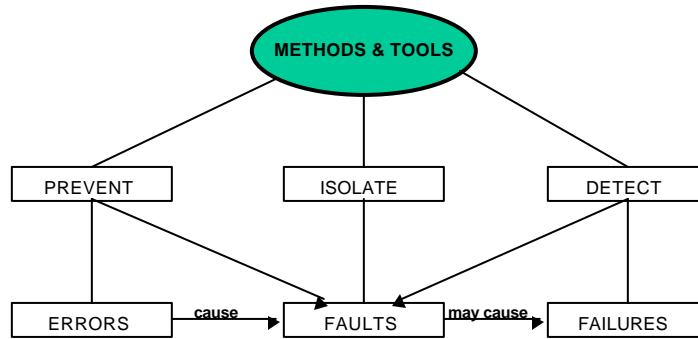
## Building Models

### Methods and Tools Dealing with Software Defects

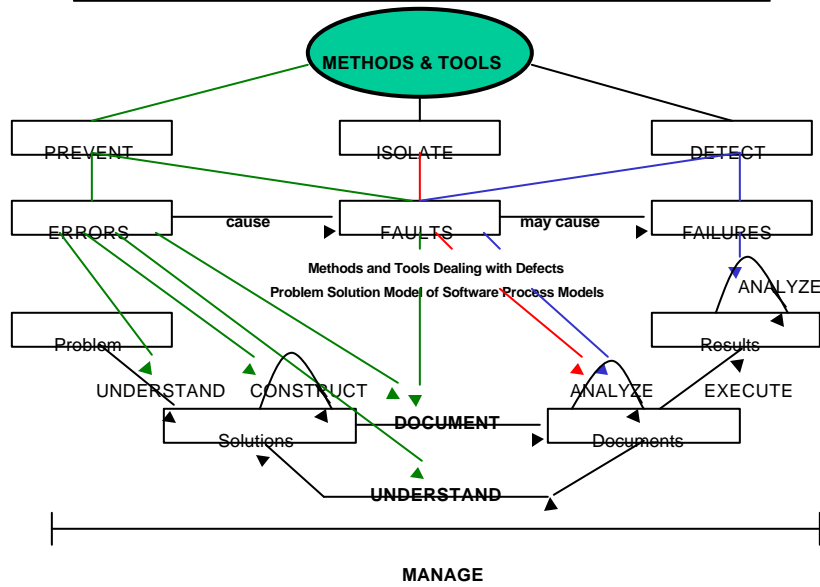


## Building Models

### Methods and Tools Dealing with Defects



## Combining Models



## Change and Defect Models and Metrics

---

### Error Data Classes

**Error origin** - the phase or activity in which the misunderstanding took place.

Example subclasses: requirements, specification, design, code, unit test, system test, acceptance test, maintenance

**Error domain** - the object or domain of the misunderstanding.

Example subclasses: application, problem-solution, notation <semantics, syntax>, environment, information management, clerical

**Document error type** - the form of misunderstanding contained in a document.

Example subclasses: ambiguity, omission, inconsistency, incorrect fact, wrong section

**Change effect** - the number of modules changed to fix an error

## Change and Defect Models and Metrics

---

### Fault Data Classes

**Fault detection time** - the phase or activity in which the fault was detected.

Example subclasses: requirements, specification, design, code, unit test, system test, acceptance test, maintenance

**Fault Density** - number of faults per KLOC

**Effort to Isolate/Fix** - time taken to isolate or fix a fault usually in time intervals

Example subclasses: 1 hour or less, 1 hour to 1 day, 1 day to 3 days, more than 3 days

**Omission/commission** - where omission is neglecting to include some entity and commission is the inclusion of some incorrect executable statement or fact

**Algorithmic fault** - the problem with the algorithm

Example subclasses: control flow, interface, data <definition, initialization, use>.

## **Change and Defect Models and Metrics**

---

### **Failure Data Classes**

**Failure detection time** - the phase or activity in which the failure was detected

Example subclasses: unit test, system test, acceptance test, operation

**System Severity** - the level of effect the failure has on the system

Example subclasses: operation stops completely, operation is significantly impacted, prevents full use of features but can be compensated, minor or cosmetic

**Customer Impact** - the level of effect the failure has on the customer

Example subclasses: usually similar to the subclasses for system severity but filled out from the customer perspective so the same failures may be categorized differently because of subjective implications and customer satisfaction issues

## **Change and Defect Models and Metrics**

---

### **Sample Defect Metrics**

Number of faults per line of code

Number of faults discovered during system test, acceptance test and one month, six months, one year after system release

Ratio of faults in system test on this project to faults found after system test

Number of severity 1 failures that are caused by faults of omission

Percent of failures found during system test

Percent of interface faults found by code reading

## **Building Defect Baselines**

---

### **Error Origin Classification**

**What are the major problem areas in our projects?**

**Is there a pattern? I.e., do similar types of problems arise again and again?**

**Can we build baselines that characterize our errors, faults or failures?**

**Can we use these baselines to evaluate the effectiveness of a new technology or other form of change in our environment?**

**Is there a pattern associated with the defects that will support prediction?**

## **Building Defect Baselines**

---

### **Error Origin Classification**

Environment:

NASA/GSFC, SEL

Ground support software for unmanned spacecraft control

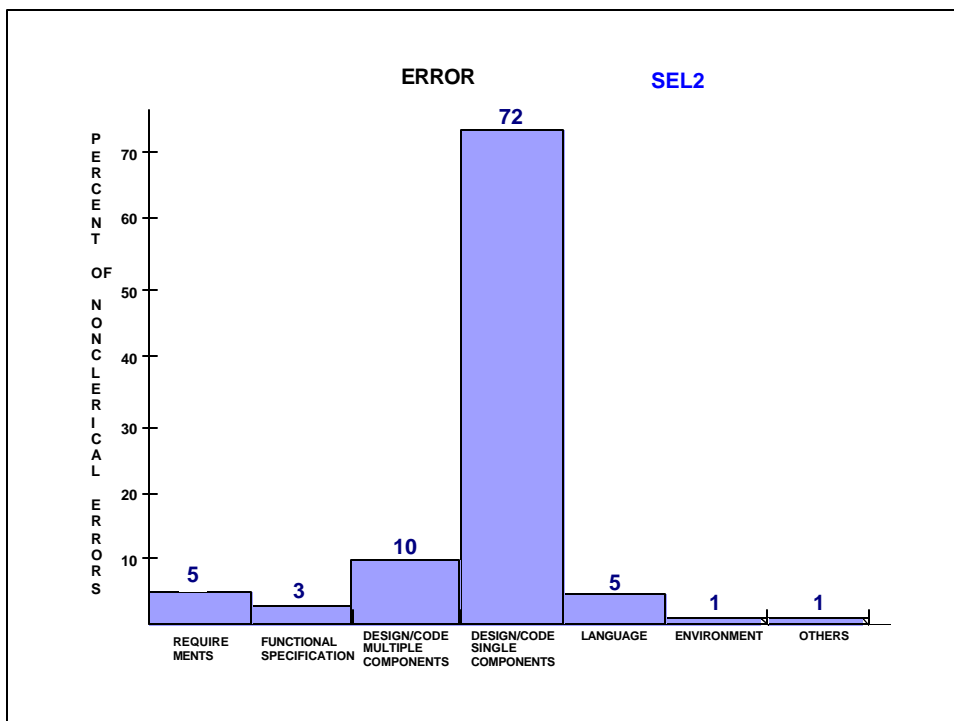
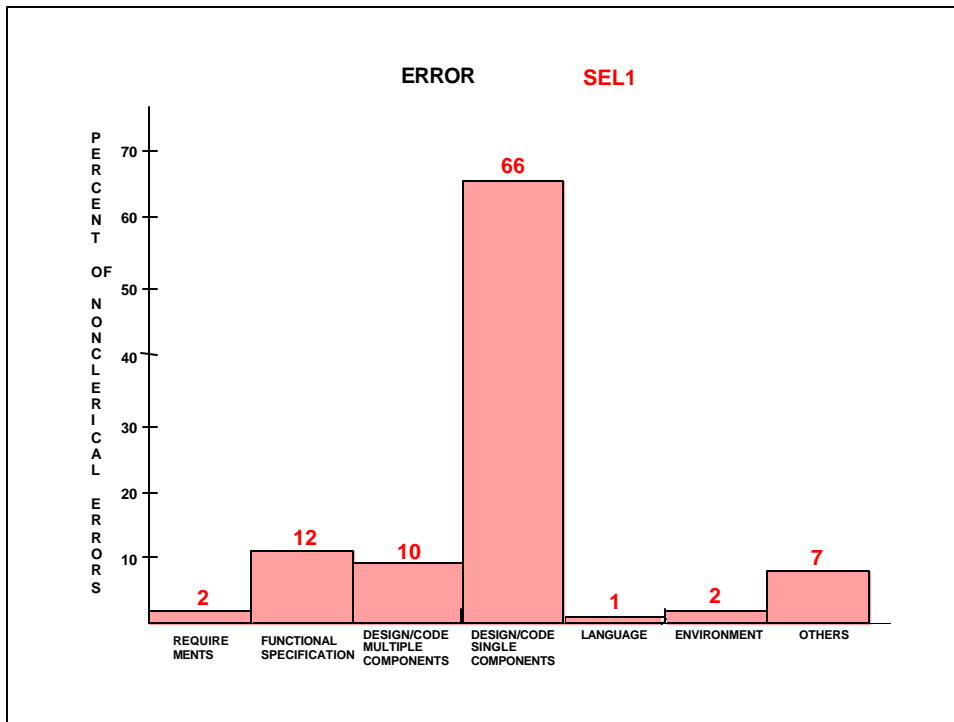
50K to 120K source lines of Fortran code

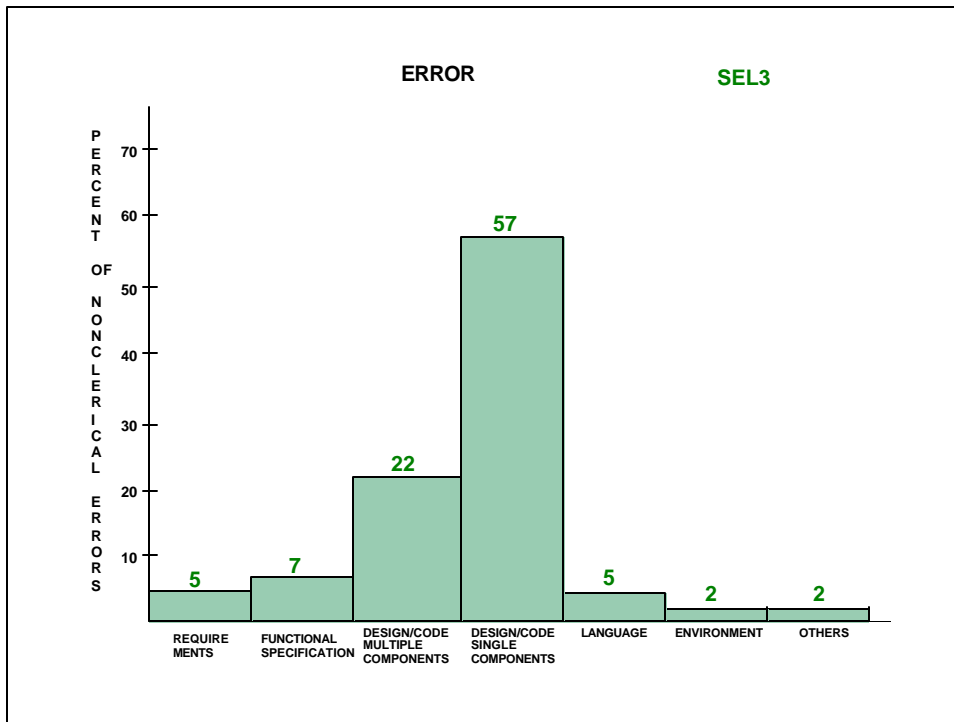
Design through acceptance test

Example Classification

What is the distribution of errors by error origin (i.e., according to the phase in which the misunderstanding took place)

Phases: Requirements, Functional Specification, Design or Coding of Multiple Components, Design or coding of a single component, Language, Environment, Other.





## Building Defect Baselines

---

### Error Origin Classification

There is a pattern

Not what might have been expected

There were many changes to the requirements and function specification which were classified as changes rather than defects

Several defects were identified and changed before entering the test phase

## **Change and Defect Models and Metrics**

---

### **Issues**

What is a change?

What is a defect (error, fault failure)?

What do you count? When do you start counting?

How do you identify them? How do you compare them from different populations? What are false positives?

What is a change vs. a defect?

How do you classify them? What is the environment? What kinds of meta data do you need or consider? What is the politics involved in the collection process, feedback process?

How do we analyze? How evaluate? How do we build predictive models?

## **Models and Metrics**

---

### **Issues**

Sound definition of terms

Insight into the object, its use or environment

The model should generate associated measurement

The measurement scale and value set need to be well defined

Concern about the cost and accuracy of the data collected