110041

N95- 16456

A Study of Software Standards Used in the Avionics Industry

Kelly J. Hayhurst
Assessment Technology Branch
Research Technology Group

Within the past decade, software has become an increasingly common element in computing systems. In particular, the role of software used in the aerospace industry, especially in life- or safety-critical applications, is rapidly expanding. This intensifies the need to use effective techniques for achieving and verifying the reliability of avionics software. Although certain software development processes and techniques are mandated by government regulating agencies such as the Federal Aviation Administration (FAA) and the Department of Defense, no one methodology has been shown to consistently produce reliable software. The knowledge base for designing reliable software simply has not reached the maturity of its hardware counterpart.

To date, existing software development methods and standards have been accepted largely based on intuitive arguments or anecdotal evidence. The data typically collected from a software development process include a description and some classification of faults identified during the prescribed development and verification activities and the final software product. From a statistical perspective, this represents a single replicate of development information. From this single replicate, some insight could be gained into the feasibility and impact of the software development method on that particular implementation of software. However, the single replicate does not provide enough information to make statistical inferences with confidence about the effectiveness of the development method in general and it provides little information about the operational behavior of the software. To provide the empirical data necessary to scientifically evaluate and improve software processes and product reliability, controlled experimentation that accounts for the performance of software during operation is needed.

In an effort to increase our understanding of software, Langley Research Center has conducted a series of experiments over the past 15 years with the goal of understanding why and how software fails. With an increased understanding of the failure behavior of software, improved methods for producing reliable software and assessing reliability can be developed. As part of this program, the effectiveness of current industry standards for the development of avionics software is being investigated. This study involves the generation of a controlled environment to conduct scientific experiments on software processes.

The Guidance and Control Software (GCS) project involves the establishment of an experimentation test-bed to monitor and study the application of software development methods and collect data that can be used to make statistical inferences about the effectiveness of those methods. This test-bed allows the development and simulated operational testing of multiple implementations of a guidance and control application that was adapted from the terminal descent phase of the Viking lander. The test-bed is comprised of software requirements for the guidance and control application, a configuration management and data collection system, and a software simulator to run the control software in a simulated operational environment. The simulator is designed to allow one or more implementations of the GCS to run in a multitasking environment and to collect data on the comparison of the results from multiple implementations.

This test-bed provides a capability for empirically investigating the effectiveness of software development methods along with investigating the reliability of the resultant software. Currently, the GCS test-bed is being used to investigate development and verification techniques that comply with the Requirements and Technical Concepts for Aviation RTCA/DO-178B guidelines, "Software Considerations in Airborne Systems and Equipment Certification." The DO-178B guidelines are used by every commercial civil transport airframer and equipment vendor since compliance with

C-2

these guidelines is required by the FAA for developing software to be used in systems or equipment certified for use in commercial aircraft.

The purpose of the DO-178B document is to provide guidelines for the production of software for airborne systems that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. It is hoped that following the guidelines in DO-178B will ensure the production of reliable software that is documented, traceable, testable, and maintainable. The guidelines, however, do not stipulate specific reliability requirements for the software product since currently available reliability estimation techniques do not provide results in which confidence can be placed to the level required for certification purposes.

The DO-178B guidelines decompose the software life cycle into three major processes: a software planning process, software development processes, and integral processes. The software planning process defines and coordinates all of the project activities. The software development processes are those processes that actually produce the software product. These include the requirements, design, code, and integration processes. And finally, the integral processes ensure the correctness, control, and confidence of the software life cycle processes and their outputs. The integral processes consist of the verification, configuration management, quality assurance, and certification liaison processes.

To study the effectiveness of the DO-178B guidelines on the quality of the software, a simple case study in which two GCS implementations are being developed is being conducted. Two teams consisting of a programmer and a verification analyst have each been tasked to develop an implementation of the GCS following the DO-178B guidelines within the GCS test-bed. An extensive problem reporting system captures relevant software error information throughout the DO-178B development process. This data includes: a description of the software errors found; the activity when the error was detected, such as design review, unit testing, or integration testing; and, action taken with respect to the error. This data will allow us to not only look at the number of faults detected but, more importantly, the class of faults found at different development stages and the relationship among the classes of faults found by the different verification techniques. This information coupled with the effort data for all development and verification activities could provide some insight into the effectiveness of the various development and verification methods.

After the two implementations have completed the DO-178B development process, the final software products will undergo testing in the simulated operational environment to help identify any remaining faults. These results could provide further insight into the effectiveness of the development methods and the reliability of the final software products.

Due to the extent of the data collection and configuration management procedures used in the test-bed, any phase in the life cycle of the GCS implementations can be reproduced. This gives a researcher the capability to go back to any one of the stages of the development process, apply a different development or verification technique to the software, and compare the resulting software to any previously developed implementation. Hence, the GCS development and verification environment can serve as a test-bed for the analysis of various software development and verification processes.

Many lessons have been learned about conducting software experiments during the course of this study of the DO-178B guidelines. A primary lesson is that a simple case study is not an adequate experiment design to evaluate an entire software development process. Conducting a more statistically rigorous software experiment, however, would require significant resources in terms of time and man-power. Development of the GCS test-bed, though, is a step toward conducting the experimentation necessary to provide the empirical data we need to scientifically evaluate and improve software processes and product quality.

The presentation provides further detail about the study of the DO-178B guidelines and the effort to conduct valid software experiments.

A Study of Software Standards Used in the Avionics Industry

Kelly J. Hayhurst
Assessment Technology Branch
Research and Technology Group

The Role of Computers in LaRC R&D Workshop
June 15, 1994

Outline

- Background
- Software Standards
- Guidance and Control Software Project
- Summary

Background

- Software is used in a wide variety of applications:
  - video games, answering machines, anti-lock brakes on cars, automatic teller machines, ..
- Software has many benefits compared to its hardware counterpart:
  - allows for more complex logic
  - provides increased flexibility
  - easier to modify
- Use of software is increasing in life- and safety-critical applications
  - avionics, Airbus 320
  - control of nuclear power plants

Software Engineering

- Software is a logical rather than a physical system element
  - Software is developed or "engineered" -- not manufactured

The establishment and use of sound engineering principles to economically obtain *reliable* software that works efficiently on real machines

- Engineering: the application of a systematic approach based on *science and mathematics*, toward the production of a product, process, or system

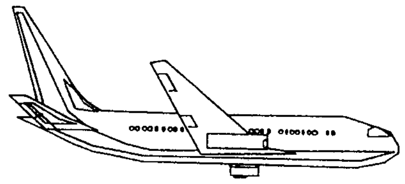
Reliable Software

- Achieving reliable software is a global problem
  - no one knows how to generate perfect software
- Many proposed software reliability models (since '64)
  - Inadequate for estimation about life-critical software
    - most consider reliability growth based on faults found in development, as opposed to *operational reliability*
  - Often based on simplistic (unverified) assumptions
    - constant failure rates
    - stochastic independence
- Little existing data available to validate models

Software Dilemma

➡ Software can significantly expand system capability

➡ Since we don't know how to build perfect software -- Risk



How do we deal with these risks?

Software Standards

- There are a number of software guidelines/standards used in industry
  - DO-178B, used by the Federal Aviation Administration (FAA)
  - DoD-2167A, used by the Department of Defense
  - ISO 9000
- Provide the guidelines for the production of software that
  - performs its intended function
    - with some level of confidence that complies with the given requirements

Software Standards

- Many software development techniques, models and standards exist and are in use
  - most have been accepted largely based on logical arguments or anecdotal evidence

"...we need to codify standard practices for software engineering -- just as soon as we discover what they should be. Regulations *uninformed by evidence*, however, can make matters worse."

-- from Digital Woes (Why We Should not Depend on Software), by Lauren Ruth Wiener

Focus

We need to become "...informed by evidence"

- Conduct scientific experiments to understand:
  - software failure
    - need to examine operational behavior of software
  - the effect of different software development techniques
    - relate that understanding to process models and standards

Conduct Experiments!!
Collect Empirical Evidence!!

Software Experiments in ATB

GOAL

Establish a controlled environment to conduct scientific experiments to address:
☆ the reliability of software and
☆ the effectiveness of software development methods

- Guidance and Control Software (GCS) Project
  - study of the RTCA/DO-178B guidelines (Software Considerations in Airborne Systems and Equipment Certification)
  - "sponsored" by the FAA

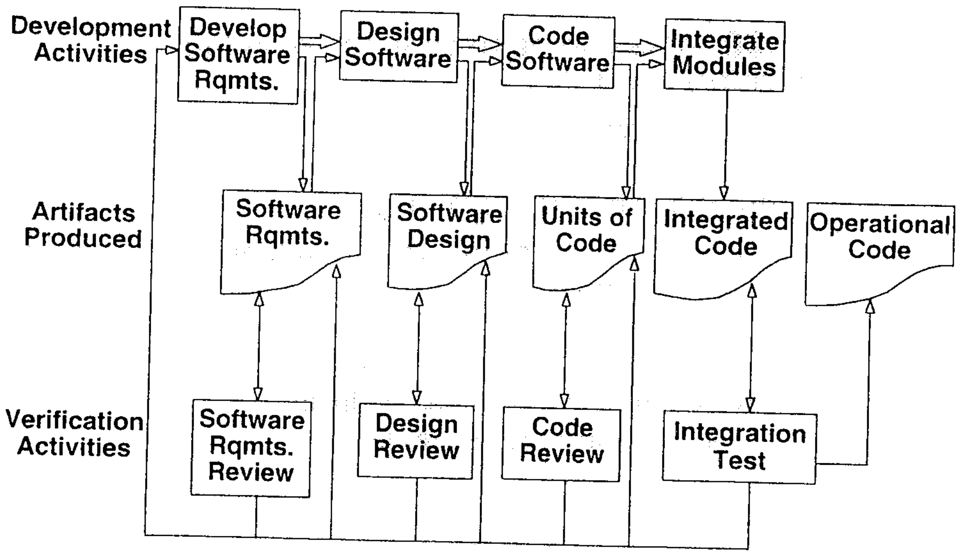
RTCA/DO-178B Guidelines

- FAA requires compliance with DO-178B for software developed for embedded commercial aircraft equipment
  - software designers must take a disciplined approach to software development
- Gives general guidelines for software development and verification according to “software levels” -- A - E
  - A: anomalous behavior causes catastrophic failure condition
  - E: anomalous behavior has no effect on operational capacity

Software Life Cycle Processes

- Planning Process: defines and coordinates the software development activities
- Development Processes:
  - Software Requirements Process
  - Software Design Process
  - Software Coding Process
  - Integration Process
- Integral Processes: ensure correctness, control and confidence
  - Software Verification Process
  - Software Configuration Management Process
  - Software Quality Assurance Process
  - Certification Liaison Process

Development & Verification Flow



DO-178B Life Cycle Data

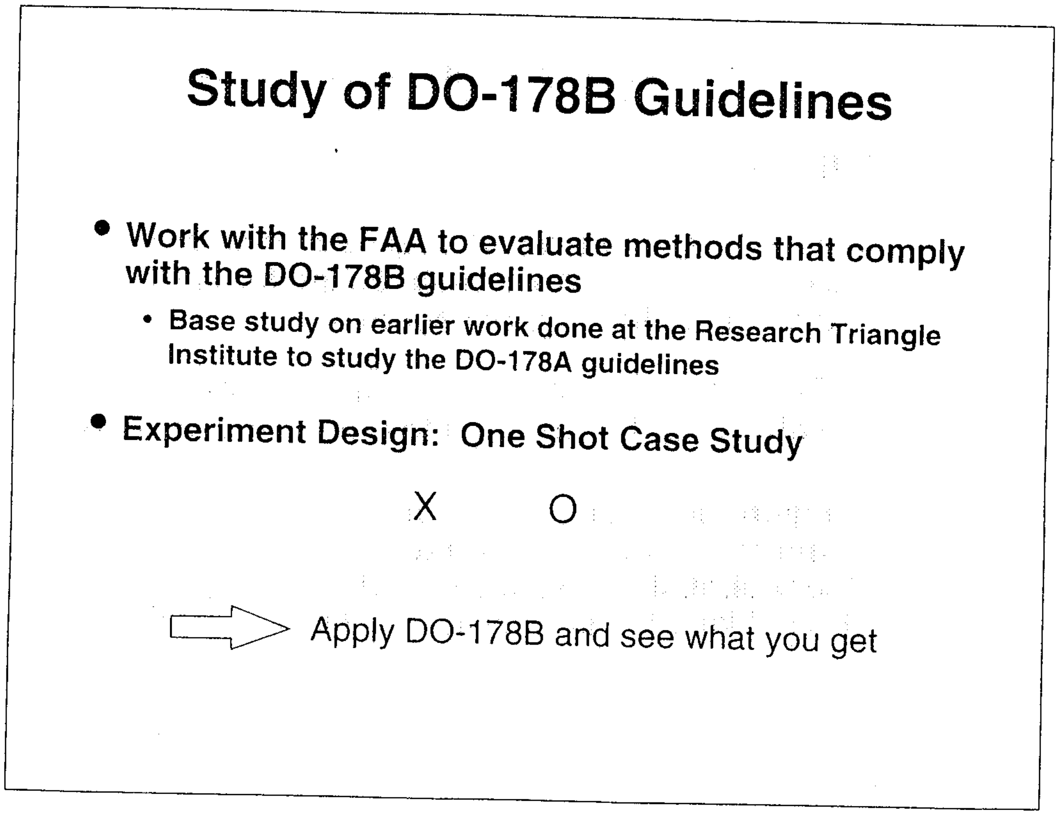
Life Cycle Process	Life Cycle Data
Planning	Plan for Aspects of Certification Development Standards Accomplishment Summary
Development	Requirements Data Design Description Source Code Executable Object Code
Integral	Verification Plan Verification Procedures & Cases Verification Results Configuration Management Plan Configuration Management Records Development Environment Configuration Index Configuration Index Quality Assurance Plan Quality Assurance Records Problem Reports

CASE Tools

- CASE tools can be used in the development of airborne software
  - Any tool used must be qualified
- Qualification is done by type:
  - Software Development Tools: whose output is part of the airborne software
    - ex. source code generator
  - Software Verification Tools: tools that cannot introduce errors -- but may fail to detect them
    - ex. analysis of complexity tool

CASE Tools Qualification

- For Software Development Tools:
  - show that the development process used for the tool is equivalent to that used for the airborne software
- For Software Verification Tools:
  - show that the tool complies with its operational requirements under normal operating conditions

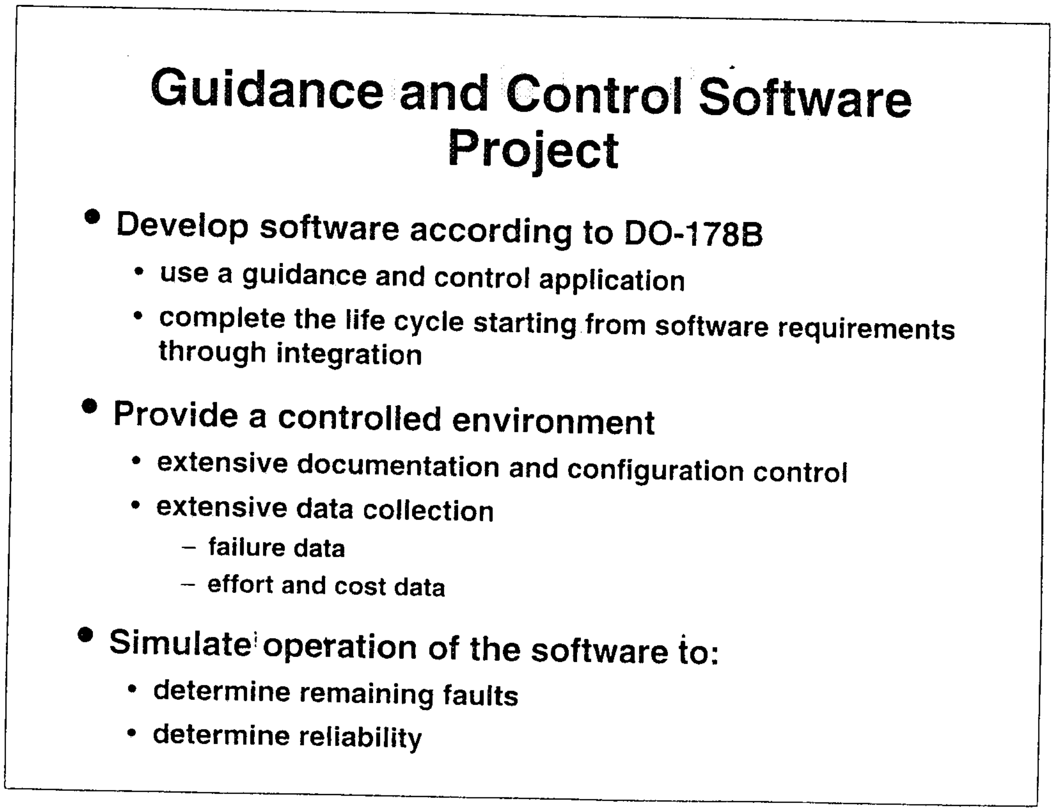


Study of DO-178B Guidelines

- Work with the FAA to evaluate methods that comply with the DO-178B guidelines
  - Base study on earlier work done at the Research Triangle Institute to study the DO-178A guidelines
- Experiment Design: One Shot Case Study

X O

⇨ Apply DO-178B and see what you get



Guidance and Control Software Project

- Develop software according to DO-178B
  - use a guidance and control application
  - complete the life cycle starting from software requirements through integration
- Provide a controlled environment
  - extensive documentation and configuration control
  - extensive data collection
    - failure data
    - effort and cost data
- Simulate operation of the software to:
  - determine remaining faults
  - determine reliability

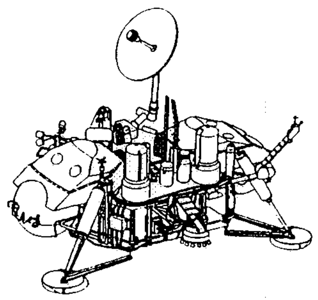
The GCS Application

Purpose:

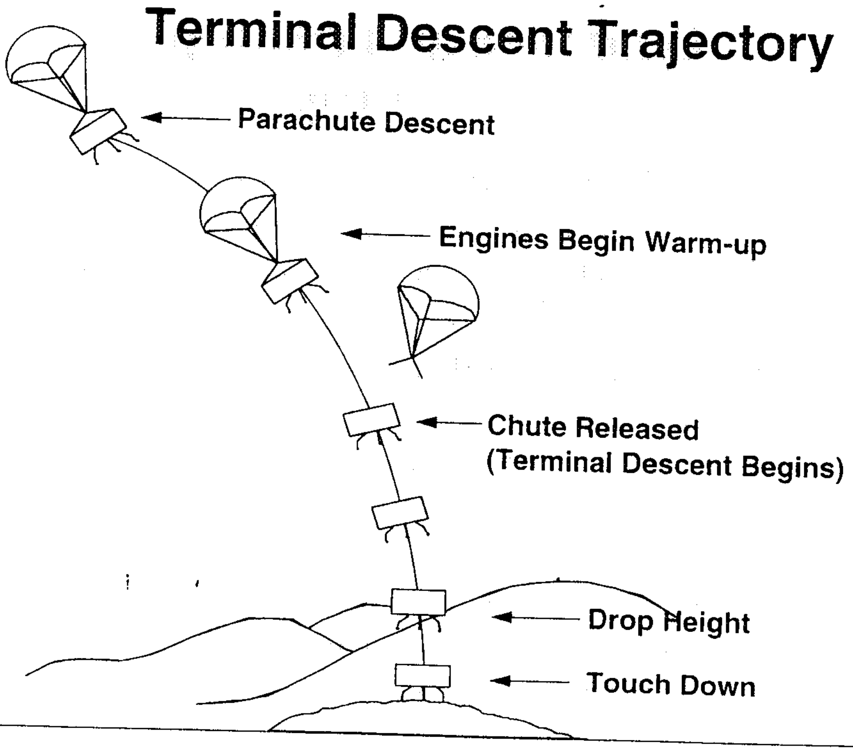
(1) Provide guidance and engine control of a planetary landing vehicle during terminal descent to the planet's surface

(2) Communicate sensory information about the vehicle and its descent to a receiving device

- Requirements are based on a simulation program used to study the probability of success of the 1976 Viking Lander mission to Mars



Terminal Descent Trajectory



Software Composition

The guidance and control software is composed of:

11 Functional Units which are divided into:

3 Subframes:

Sensor Processing
Guidance Processing
Control Law Processing

1 Frame = 1 iteration of the 3 subframes

1 Trajectory = ~ 2000 frames

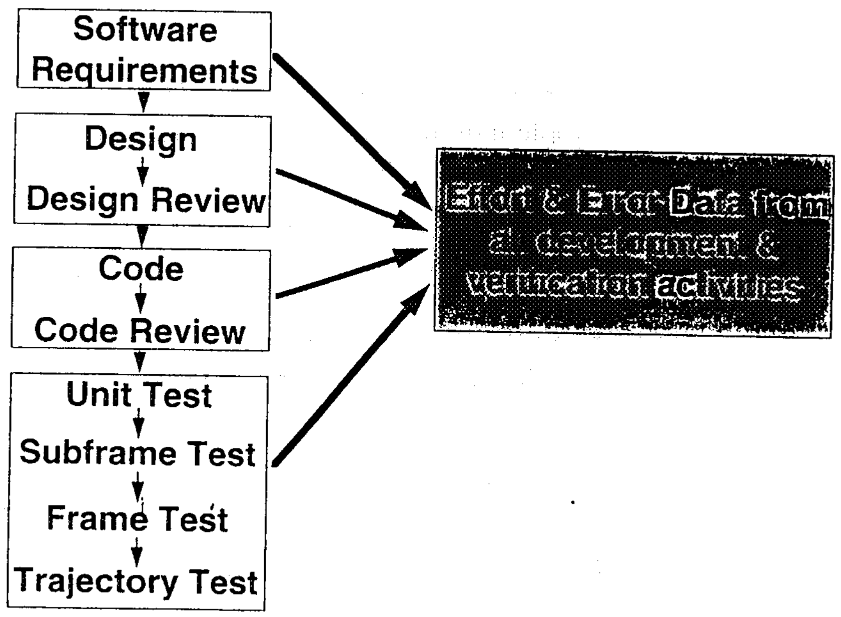
GCS Development Processes

- Producing 2 GCS implementations
  - each implementation has a designated programmer & verification analyst
- Each development team uses the same software high-level requirements document
- Designs generated using team*work*
  - conduct design review using formal inspection procedures
- Implementations coded in FORTRAN
  - projected size: 1500 - 2000 lines of code
  - conduct code review using formal inspection procedures

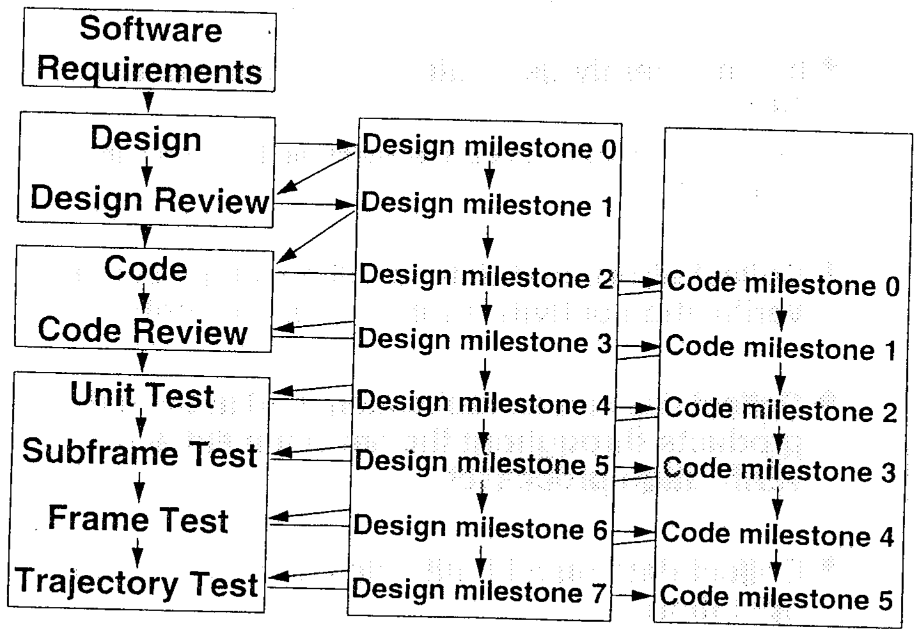
Integration Process

- Code is integrated at 4 levels: functional units
  subframes
  frames
  trajectory
- Testing conducted at all 4 levels to:
  - demonstrate that the software satisfies its requirements
  - demonstrate (with high confidence) that errors which could lead to unacceptable failure conditions have been removed
- 100% coverage for requirements-based tests
- 100% modified condition/decision coverage

Development Products



More Products



Software Products

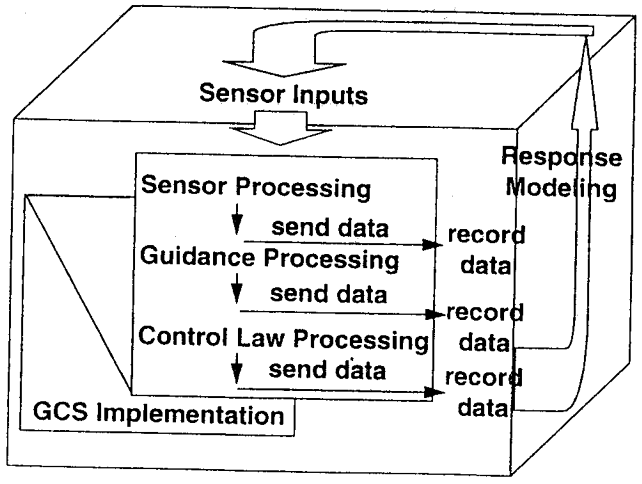
- Each software product (requirements, design, code, test cases, documentation) is placed under configuration control starting with the initial version
  - the Code Management System (CMS) by Digital Equipment Corp. is being used
- Each subsequent change to a software product is controlled and captured by the configuration management system
- All versions of any software product are preserved and can be reproduced

Experiment Basics

- Independently generate "n" implementations of the GCS
  - each following the development methodology defined in DO-178B
- Collect effort/cost data for all development and verification activities for each implementation
- Collect data on all faults identified in the software products throughout the development and verification processes
- Collect data on all faults identified in simulated operation

GCS Simulator

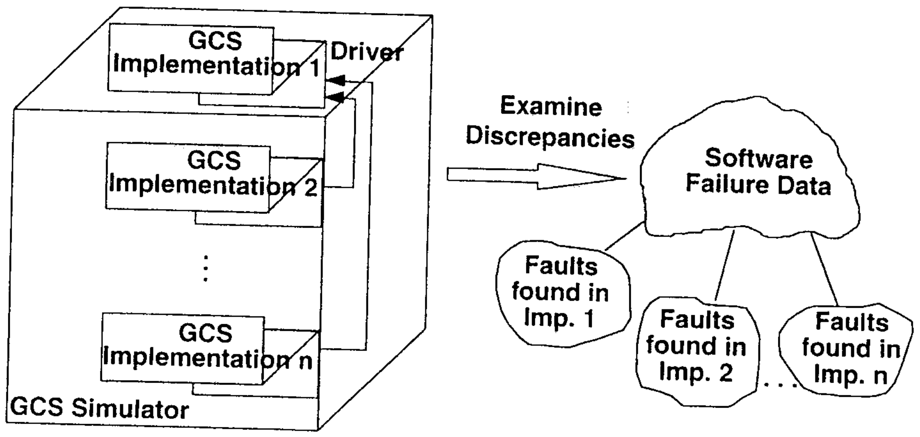
- Provides inputs (about environment & lander) for sensor processing
- Performs response modeling for the guidance and control
- Receives data



GCS Simulator

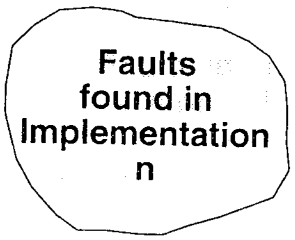
- Serves as a testbed for back-to-back testing of multiple GCS implementations (up to 28)
- For back-to-back testing, one implementation is designated as the "driver" implementation
- The results of all implementations are checked at the end of each subframe
  - for limit errors, comparing each variable against its predetermined valid range
  - for accuracy errors, comparing results of each implementation with results of the driver implementation
- All miscomparisons are recorded and investigated to determine the source of the problem

Operational Failure Characterization



- Use the software failure data to
  - estimate reliability of final version of each implementation
  - determine effectiveness of the development methodology

Understanding the Failure Data



Questions of Interest

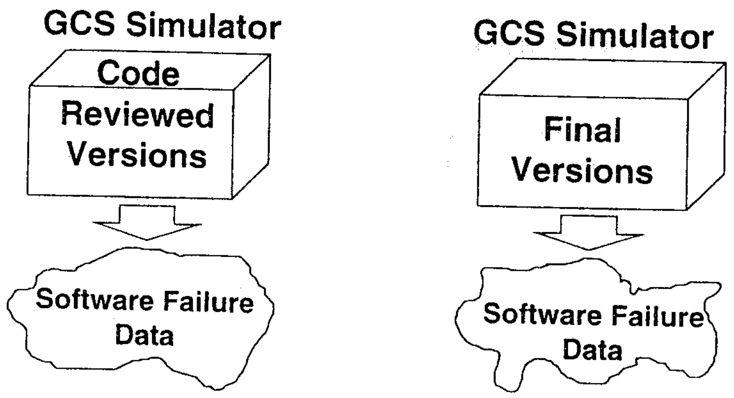
-- How many faults in the set?

-- What types of faults?

-- Are there any critical faults?

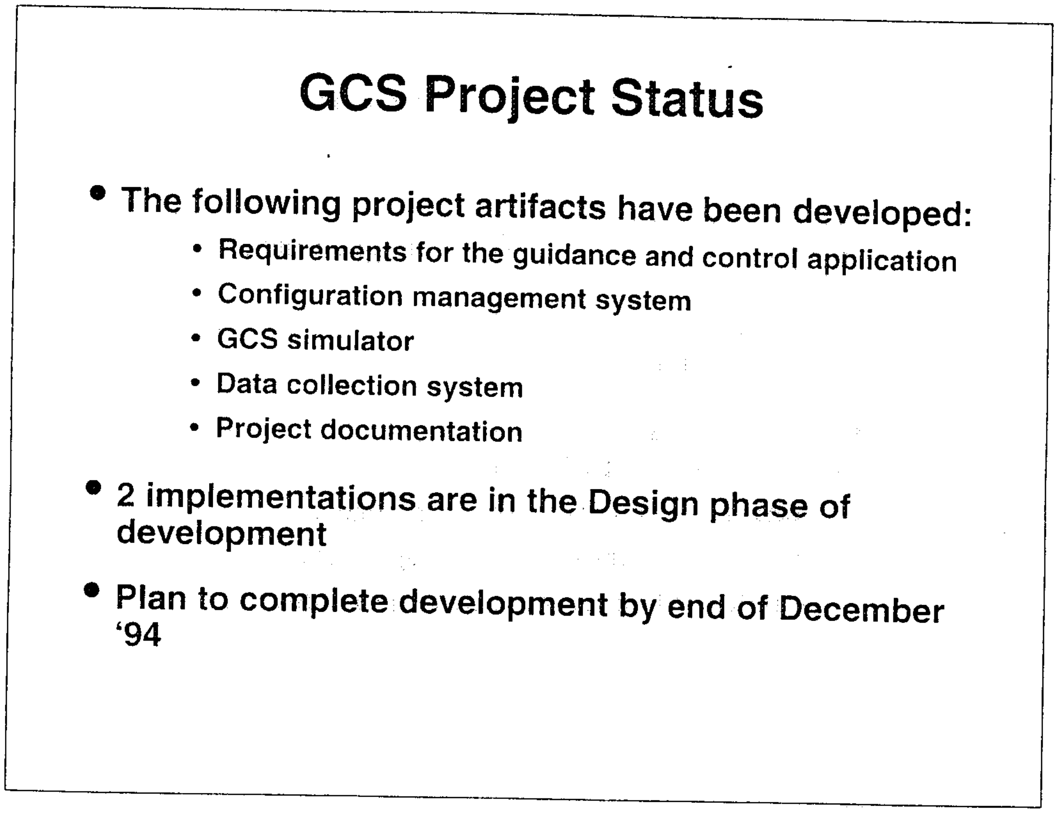
-- Are there classes of faults found during random testing that are different than those found during DO-178B development cycle?

Studying Effectiveness



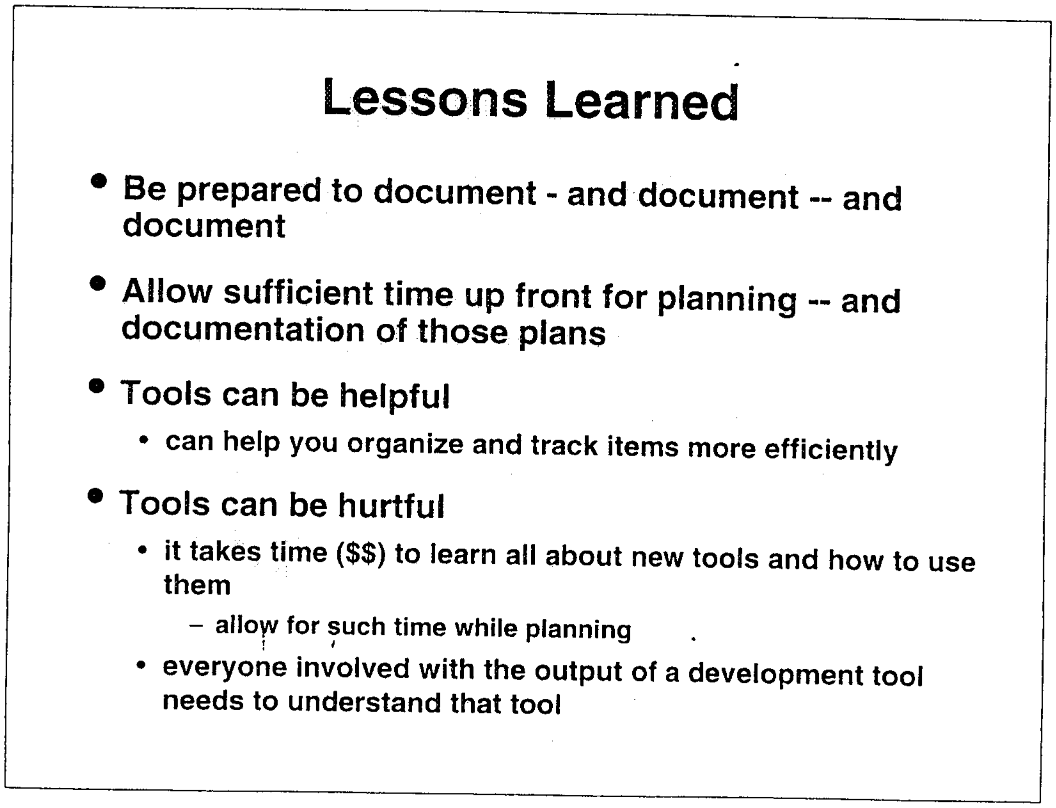
Are these fault sets equivalent?

-- Is the integration process more effective (or efficient) compared to other fault detection methods?



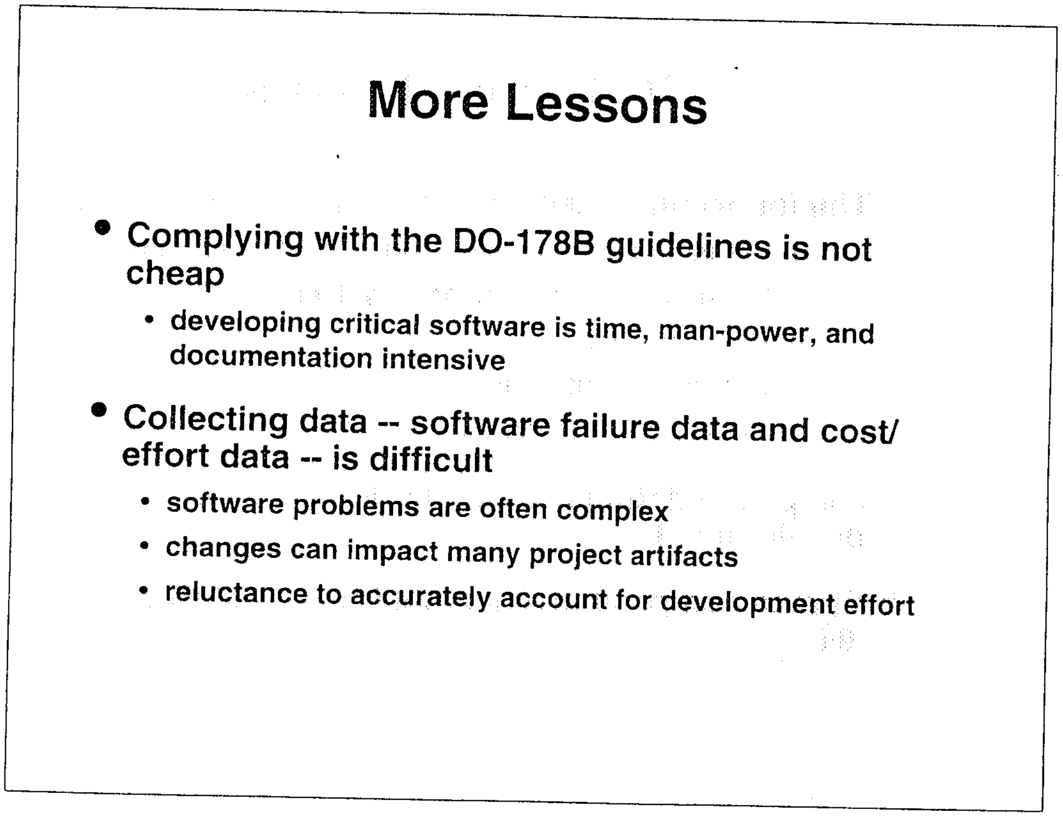
GCS Project Status

- The following project artifacts have been developed:
  - Requirements for the guidance and control application
  - Configuration management system
  - GCS simulator
  - Data collection system
  - Project documentation
- 2 implementations are in the Design phase of development
- Plan to complete development by end of December '94



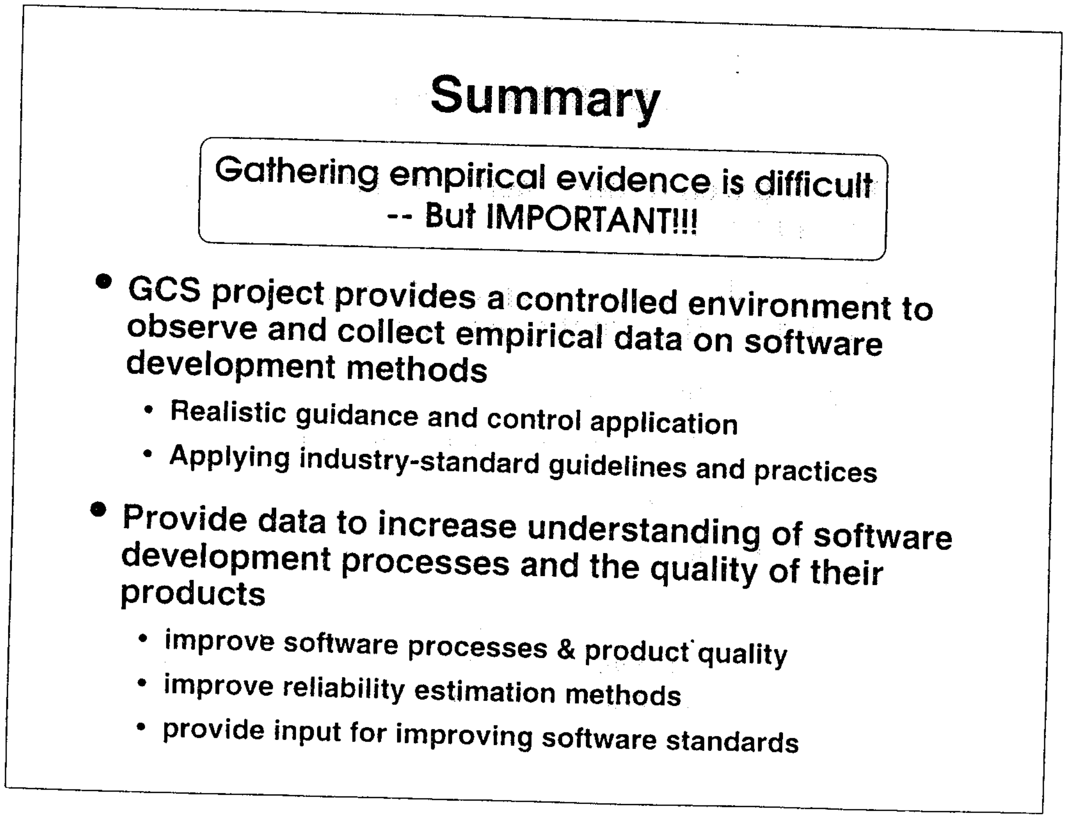
Lessons Learned

- Be prepared to document - and document -- and document
- Allow sufficient time up front for planning -- and documentation of those plans
- Tools can be helpful
  - can help you organize and track items more efficiently
- Tools can be hurtful
  - it takes time ($$) to learn all about new tools and how to use them
    - allow for such time while planning
  - everyone involved with the output of a development tool needs to understand that tool



More Lessons

- Complying with the DO-178B guidelines is not cheap
  - developing critical software is time, man-power, and documentation intensive
- Collecting data -- software failure data and cost/effort data -- is difficult
  - software problems are often complex
  - changes can impact many project artifacts
  - reluctance to accurately account for development effort



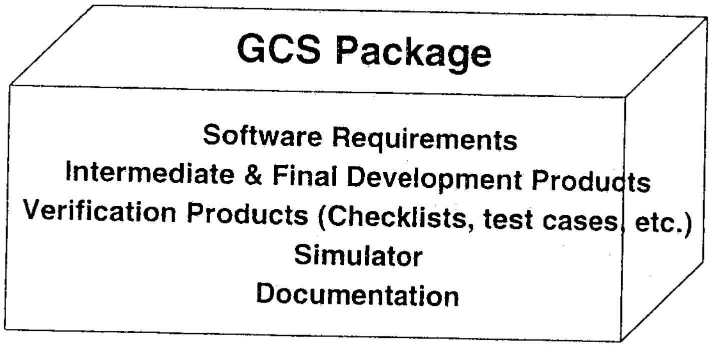
Summary

Gathering empirical evidence is difficult
-- But IMPORTANT!!!

- GCS project provides a controlled environment to observe and collect empirical data on software development methods
  - Realistic guidance and control application
  - Applying industry-standard guidelines and practices
- Provide data to increase understanding of software development processes and the quality of their products
  - improve software processes & product quality
  - improve reliability estimation methods
  - provide input for improving software standards

Project Plans

- Make the GCS testbed available to other researchers
- Improve the experiment design to allow more statistical analysis



GCS Package

Software Requirements
Intermediate & Final Development Products
Verification Products (Checklists, test cases, etc.)
Simulator
Documentation