Sixth Annual Software Engineering Workshop

Goddard Space Flight Center

December 2, 1981

Cleanroom Software Development

M. Dyer and H. D. Mills

The 'cleanroom' software development process is a new IBM technical and organizational approach to developing software with certifiable reliability. Key ideas behind the process are well structured software specifications, randomized testing methods and the introduction of statistical controls; but the main point is to deny entry for defects during the development of software. This latter point suggests the use of the term 'cleanroom' in analogy to the defect prevention controls used in the manufacture of high technology hardware.

The present state of the art in software development is to conceive and design a system in response to perceived requirements, then test the system with cases perceived to be typical to those requirements. The result is frequently a system which works well against inputs similar to those tested for, but one which is unreliable in unexpected circumstances. In fact, the evidence obtained by such testing is entirely anecdotal rather than statistical.

In the 'cleanroom', we embed the entire software development process within a formal statistical design, in contrast to executing selected tests and appealing to the randomness of operational settings for drawing statistical inferences. Instead, we introduce random testing as a part of the statistical design itself so that when development and testing is completed, statistical inferences can be made about the future operation of the system.

We believe there are several major benefits to such a procedure. One benefit is derived from standard statistical procedures in which a formal statistical design permits objective statements about properties of the system. But it is believed that an even more important benefit will arise from effects on the developers through the discipline of the statistical design on their activities. In fact, we believe that developing systems under stringent statistical controls will induce significant behaviour modifications on software developers.

Presently, when developers conceive early tests to check the correct operation of a system, they are able to identify just those parts of the system that will have to function correctly to pass those tests. Therefore, they can develop systems in phases, and control the testing such that the system under development is protected from unwanted testing. As a consequence, system parts may be omitted or done perfunctorily since the choice of tests is under the control of the developers.

We have in mind a different circumstance in testing under statistical control, namely, that from the outset tests are selected at random out of an expanding (top down) hierarchy of operational test cases. Therefore, the system designer must be prepared to deal with a growing, but always coherent, set of eventualities. It is believed that this circumstance, which may seem unfair or impossible at first glance, will dramatically change the way software development is done, by forcing a system approach top down rather than permitting bottom up pieces to be conceived and built under the protection of developer-selected testing.

THE VIEWGRAPH MATERIALS
for the
H. MILLS/M. DYER PRESENTATION FOLLOW

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

DEFINITION

- TECHNICAL AND ORGANIZATIONAL APPROACH TO DEVELOPING SOFTWARE PRODUCTS WITH CERTIFIABLE RELIABILITY

LOGICAL EXTENSION OF

- SOFTWARE RELIABILITY THEORY
- MODERN SOFTWARE ENGINEERING PRACTICES
- FUNCTIONAL ORGANIZATIONAL STRUCTURE

GOALS

- PRODUCT RELIABILITY
  - INITIALLY ADDRESS PRODUCTS IN THE RANGE OF 10-25K SLOCS
  - RELIABILITY TARGETS OF MTBF'S MEASURED IN MONTHS AND YEARS
- STATISTICAL DESIGN
  - EXPECTATION OF CORRECT SOFTWARE DESIGNS
  - "BLACKBOX" TESTING OF SOFTWARE
  - TESTING FOR THE OPERATIONAL ENVIRONMENT
- PROCESS CONTROLS
  - SOFTWARE PRODUCT ENGINEERING FUNCTION
  - MANAGEMENT TO RELIABILITY COMMITMENTS

M. Dyer
IBM

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

RELIABILITY MODEL

- BASED ON SOFTWARE OPERATING FAILURES, NOT ERRORS IN THE CODE
- DIFFERS FROM HARDWARE MODELS, LOGICAL NOT PHYSICAL FAILURES
- REASONABLENESS DEMONSTRATED USING PUBLISHED SOFTWARE FAILURE DATA

STATISTICAL APPROACH

- INPUT/OUTPUT SPECIFICATIONS
- INPUT PROBABILITY DISTRIBUTIONS
- STOCHASTIC PROCESS INTRODUCED THROUGH RANDOMLY SELECTED RUNS
- MTBF STATISTICS DEVELOPED FROM CYCLE/FAILURE RATIO
- CERTIFICATION BASED ON FAILURE FREE EXECUTION INTERVALS, NOT ERROR FREE CODE

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

CLEANROOM DEVELOPMENT METHOD

- STARTS WITH STRUCTURED SPECIFICATION
  - STATE MACHINE MODEL
- SOFTWARE DESIGN ENGINEERING PROCESS
  - MODERN DESIGN METHODS
  - FIRST TIME CORRECT PROGRAMS
- SOFTWARE PRODUCT ENGINEERING PROCESS
  - IDENTIFICATION OF PRODUCT INPUTS AND PROBABILITY DISTRIBUTIONS
  - SOFTWARE INTEGRATION INTO PRODUCT FORM
  - COLLECTION/CORRELATION OF FAILURE STATISTICS (MTBF)
  - CERTIFICATION TO CUSTOMER
- SOFTWARE MANAGEMENT
  - RELIABILITY COMMITMENTS
  - PRODUCT VISIBILITY THROUGH MTBF MEASUREMENTS

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

DESIGN FUNDAMENTALS

- MODERN DESIGN METHODS
  - STATE MACHINES AND FUNCTIONS
  - STEPWISE REFINEMENT AND CORRECTNESS PROOFS
  - DATA TYPING AND ABSTRACTION
  - PROCESS DESIGN LANGUAGE (PDL) DOCUMENTATION
- MODERN IMPLEMENTATION METHODS
  - PROGRAM SUPPORT LIBRARIES
  - HIGH-ORDER PROGRAMMING LANGUAGES
  - STRUCTURED PROGRAMMING
  - REVIEWS AND INSPECTIONS

DESIGN INNOVATIONS

- STATISTICAL DESIGN APPROACH
  - DESIGN ALWAYS EXPOSED TO RANDOMIZED OPERATING INPUTS
  - EMPHASIS ON TOP-DOWN IMPLEMENTATION STRATEGY
- ELIMINATION OF SOFTWARE DEBUGGING
  - FOCUS TESTING ON OPERATING ENVIRONMENT
  - FOCUS DESIGN ON CORRECTNESS

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

PRODUCT ENGINEERING STRATEGY

- o CERTIFICATION BY INDEPENDENT GROUP
  - TESTING FROM SOFTWARE SPECIFICATION WITH DESIGN DETAILS HIDDEN
  - SEPARATION OF RESPONSIBILITIES AND INTERACTIONS
- o TEST DEVELOPMENT
  - ANALYSIS OF INPUT PROBABILITY DISTRIBUTIONS
  - STATISTICAL/DISCRETE INPUT VALUES
  - INITIALIZATION AND OUTPUT VALUES
  - CONCURRENCY CONSIDERATIONS FOR PERFORMANCE TESTS
- o TEST EXECUTION
  - SELECTION OF RANDOM INPUT SAMPLES
  - RECORDING OF FAILURE FREE EXECUTION MATERIALS
  - GENERATION OF MTBF STATISTICS
- o FAILURE DIAGNOSTIC SUPPORT
  - FAULT LOCALIZATION
  - REGRESSION TESTING

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

SOFTWARE DESIGN ENGINEER

- CREATES THE PRODUCT
- RESPONSIBILITY
  - IMPLEMENTATION OF AN APPROVED SPECIFICATION
  - DELIVERY OF CORRECT SOFTWARE TO THE PRODUCT ENGINEER
- OUTPUTS
  - SOFTWARE PRODUCT DESIGN
  - SOFTWARE PRODUCT CODE
  - SOFTWARE PRODUCT DOCUMENTATION

SOFTWARE PRODUCT ENGINEER

- CERTIFIES THE PRODUCT
- RESPONSIBILITY
  - VALIDATION OF THE PRODUCT AGAINST THE SPECIFICATION
  - DELIVERY OF A CERTIFIED SOFTWARE TO THE CUSTOMER
- OUTPUTS
  - SOFTWARE PRODUCT TEST PLANS/PROCEDURES
  - SOFTWARE PRODUCT INTEGRATION PLANS/PROCEDURES
  - SOFTWARE PRODUCT LIBRARIES
  - SOFTWARE PRODUCT TEST REPORTS

CLEANROOM SOFTWARE DEVELOPMENT PROCESS

TOOL REQUIREMENTS

- LIBRARY SYSTEM
  - DESIGN DOCUMENTATION
  - PRODUCT CODE
  - CERTIFICATION TEST SAMPLES
- STATISTICAL MODEL
  - MTBF CALCULATIONS
  - TREND ANALYSES
- SOFTWARE UTILITIES
  - TEST SAMPLE BUILD
  - TEST EXECUTION CONTROL
  - DATA COLLECTION/REDUCTION