Cleanroom Software Engineering

CSC 532 - Software Engineering
Professor: Dr. Box
Student: Jan Durand
Date: 10/20/2011
What is it?

- "Theory-based, team-oriented process for development and certification of high-reliability software systems under statistical quality control"

- Developed by Harlan Mills and other colleagues at IBM Federal Systems Division in the early 1980s

- Emphasizes defect prevention over defect removal

- Provides objective statistically-based fitness measure
Overview

Figure 1. Cleanroom Process Flow

Main Components

- Incremental development process model
- Formal methods for specification and design
- Correctness verification of developed code
- Statistically-based, independent testing
Incremental Process Model

● Each increment is a functional subset of final product

● Tested in environment similar to that of final product

● Increment feedback can improve subsequent increments

● Technique for managing risk
Specification/Design Method

● Box Structure Method used to "bridge gap between system-level and design-level thinking"

● Software program viewed as a functional mapping; transforms input (domain) into output (range)

● Three levels of abstraction:
  Black box - behavioral view
    ○ State box - finite state machine view
    ○ Clear box - procedural view
Correctness Verification

- Team uses mathematical verification techniques to verify the correctness of software
- Replaces unit testing of traditional development methods
- More thorough and cost-effective than unit testing
Statistically-based, Independent Testing

- Software tested based on statistical principles
- Probability-weighted usage model generated from all possible software uses
- Test cases randomly generated based on usage model
- Statistical analysis performed on test results to estimate software quality:

\[
MTTF = MR^c
\]
Advantages

● Quality:
  ○ 1994-97, 2.3 defects/KLOC vs trad. 10 defects/KLOC on first execution

● Productivity:
  ○ Less time required to debug and rework software
  ○ 800 NCSS/Man-Month vs Industry 120 NCSS/Man-Month

● Life cycle costs:
  ○ Decrease in testing, error correction, and maintenance

● Return on investment:
  ○ Cleanroom adoption costs can be recovered on first project
Real-World Applications

- **COBOL Structuring Facility product (40 KLOC)**
  - 1500-3000 errors eliminated in team reviews
  - 740 LOC/Man-Month vs Industry 150 LOC/Man-Month

- **NASA Space-Transportation Planning System (45 KLOC)**
  - + productivity 69%, - error rate 45%, - resource usage 60-80%

- **US DoD Picatinny Arsenal project (90 KLOC)**
  - + productivity 320%, 20:1 return-on-investment

- **Ericsson Telecom OS32 operating system (350 KLOC)**
  - + testing productivity 114%, + design productivity 70%
  - + quality 157%, - failures per KLOC 50%
  - - resource usage 4.4%
Disadvantages

- Too theoretical and mathematical
- Requires intensive training
- Denying programmer access to compiler is unrealistic
References


