Vorlesung "Spezielle Themen der Softwaretechnik"

The Cleanroom Method

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

- Principles
- Empirical results
- Typical practices
- Stepwise refinement
  - box structures, verification
- Statistical testing
  - Usage modeling
- Cleanroom and CMMI
Cleanroom classification and goals

- Proposed by Harlan D. Mills, IBM, since 1980
  - 'Cleanroom' stands for defect prevention instead of defect elimination

Goal:
- High, quantified reliability at low cost

Classification:
- Cleanroom is a development approach
- and a management approach

Context:
- Whenever precise specifications can be written early
  - For new development, maintenance, and reengineering
  - Independent of language and technology
- Requires approximately CMM Level 3
Cleanroom principles

Cleanroom development principle:
• Development teams strive to produce products without any defects
  • by careful design and development
  • by verification and review
  • but not by testing

Cleanroom testing principle:
• The purpose of testing is measuring the reliability of the product
  • not improving the reliability

Cleanroom management principle:
• Team-based practices limit the scope of human fallibility and allow for continuous improvement
Empirical results: IBM Cobol SF


- Project developing "Cobol Structuring Facility" COBOL/SF
  - A program analyzer/translator (written in PL/1) for converting Cobol code with GOTOs into structured Cobol code
  - 52 KLOC modified/added to existing 40 KLOC base product

- Overall productivity: +400%
- Overall defect density: 3.4 defects/KLOC
- Field-testing defects: 10 (only 1 of them major)

- The defect reduction is the main reason for the huge improvement in productivity
  - Testing such a system is very laborious
Empirical results: Ellemtel/Ericsson OS32

- L.G. Tann: "OS32 and Cleanroom"
  - 1st Annual European Industrial Symposium on Cleanroom Software Engineering, Copenhagen, Denmark, 1993, pp. 1-40.

- Project developing an operating system for telephone switching systems
  - 73 people staff, 33 months duration
  - 350 KLOC resulting software size

- Development productivity: +70%
- Testing productivity: +100% (tests per hour)
- Testing defect density: 1 defect/KLOC

- These are very big improvements, considering this was a mature development organization already.
Empirical results:
Controlled experiment

  - IEEE Transactions on Software Engineering, 13(9), Sept. 1987
- A controlled experiment:  
  15 teams (10 Cleanroom, 5 conventional) of 3 professional developers each develop the same software
  - duration 6 weeks, 4 milestones,
  - resulting size 800 to 2300 LOC
- Results:
  - The Cleanroom teams developed more functionality
  - All Cleanroom teams kept all milestones, only 2 of the 5 others did
  - The Cleanroom programs were less complex (control flow) and had better annotation
  - The Cleanroom programs had significantly less test failures
  - 86% of the developers missed testing (quality was not affected)
Typical Cleanroom techniques

Small teams
- High motivation, close cooperation, efficiency
  - "Defects are not acceptable!"
- Parallel development
  - Strict modularization has to be done at specification time
- Exact specification
  - All partial specifications are precise and self-contained

Strict separation of development and testing
- Development teams
  - Development teams are strictly forbidden to perform any testing
- Test teams
  - Test teams never modify programs
Typical Cleanroom techniques (2)

Exact specification
- Defect prevention
  - Precise specifications help avoid ambiguity defects
- Verification
  - During development, defects are continually searched for by comparing with specification
- Z, VDM, box method, special grammars

Stepwise refinement
- Specification (black box)
  - Describes WHAT without HOW
- State description (state box)
  - Specification as a state machine (not always useful)
- Process description (clear box)
  - Partial HOW: "Implementation", but may use further black boxes
Typical Cleanroom techniques (3)

Review/verification
- Performed for each refinement
  - State box and clear box
- Grounded in mathematics
  - Convincing argumentation, rarely formal mathematical proof
- Argument is formulated during inspection

Incremental development
- Initially, only basic functionality is developed

Statistical testing
- Usage model ling
  - Test cases are a random sample according to usage model
- Quantitative statement on reliability (certification)
Typical Cleanroom techniques: Note

- None of these techniques is absolutely mandatory

- They can be driven to extremes
  - for instance developers may be prohibited to even compile their code

- They can be relaxed
  - for instance by performing defect testing in addition to statistical testing

- They can be exchanged for others
  - for instance by driving development in some other way than by box refinement
Cleanroom process flow overview

Requirements analysis
Specification
Definition of next increment

Design: next refinement
Verification, correction

Usage modeling
Test case generation

Development team
Test team

Statistical testing
Reliability certification

Lutz Prechelt,prechelt@inf.tu-berlin.de
Problems and Obstacles

Cleanroom is not suited

- if formal specification is difficult
  - which is commonly the case for interactive systems
- if determining the correctness of test outputs is costly
  - but this is a problem for conventional development as well.
  - One could do without reliability certification (no statistical testing)

Necessary preconditions

- Highly trained software engineers
  - Others cannot create reliable verification arguments
- Defined software process (CMM Level 2/3)
  - Immature processes will lack the necessary discipline and control
Specification and design with box structure

- Define black box:
  - define output based on input history

- Define state box:
  - define states for the representation of input history
  - reformulate black box (may introduce several new black boxes)
  - verify reformulation: state box must be equivalent to black box

- Define clear box:
  - define data abstraction for state data
  - reformulate state box (may introduce several new black boxes)
  - verify reformulation: clear box must be equivalent to state box

Continue with black boxes of the refinement
Trivial refinement example

• *black box 1:* triangleType(a, b, c)

**precondition:** a, b, c are positive, real numbers

**postcondition:**
return EQUILATERAL / ISOSCELES / OTHER / NO_TRIANGLE ⇐
the triple (a, b, c) is side lengths of an equilateral / non equilaterial isosceles / non isosceles triangle / cannot be side lengths of a triangle
Refinement example (2)

• *clear box 1:* \texttt{triangleType(a, b, c)}
  IF \texttt{allSidesSatisfyTriangleInequation(a, b, c)}
  THEN return \texttt{trueTriangleType(a, b, c)}
  ELSE return \texttt{NO\_TRIANGLE}

• *black box 2:* \texttt{allSidesSatisfyTriangleInequation(a, b, c)}
  precondition: \(a, b, c\) positive, real numbers
  postcondition: True if each side is shorter than the sum of the other two; else False

• *black box 3:* \texttt{trueTriangleType(a, b, c)}
  precondition: \((a, b, c)\) are the side lengths of a triangle
  postcondition: ...
Refinement example (3)

- **verification clear box 1:**
  (a, b, c) can form triangle ⇔
  the two shorter sides x, y together are longer than the longest, z.
  
  Hence, z > x + y (i.e., "side z satisfies triangle inequation")
  is sufficient for diagnosing a triangle.
  "All sides satisfy triangle inequation" is a stronger condition,
  hence also sufficient.
  
  Hence, clear box 1 is correct.
Refinement example (4)

- **clear box 2:** allSidesSatisfyTriangleInequation(a, b, c)
  return (a < b + c AND b < a + c AND c < a + b)

- **verification clear box 2:**
  3 different cases are tested (→ "triangle"),
  tests are connected by 'AND' (→ "all sides"),
  each test compares one side to the sum of the two others,
  each comparison is by 'less than' (→ correct inequation).
  Hence, the implementation appears to be correct
Refinement example (5)

- **clear box 3: trueTriangleType(a, b, c)**
  
  IF $a = b = c$ THEN return EQUILATERAL
  
  ELSE IF $a = b$ OR $a = c$ OR $b = c$ THEN return ISOSCELES
  
  ELSE return OTHER

- **verification clear box 3:**
  'Equilateral' is a special case of 'isoceles' and must therefore be tested first, this is done here.
  The test for 'equilateral' is correct.
  The test for 'isosceles' must check 3 different pairs (correct), only one needs to be equal (connection with 'OR', correct)
  'Other' is the only remaining case, must be 'ELSE' part. Correct.
  Therefore clear box 3 is correct.
Statistical Testing and Certification

• Most software processes use defect testing
  • Goal: Find as many defects as possible, with as few test cases as possible
  • Testing concentrates on 'difficult' cases.
• Defect testing makes almost no statement about reliability.

• In contrast, Cleanroom uses statistical testing
  • Goal: Quantify reliability, somewhat like acceptance testing
  • Does not specifically aim to find defects
  • Testing reflects typical frequencies of typical cases

• Basis: Usage modelling
  • Based on description of the usage profile (from requirements)
  • Mathematical description with Markov-chains (finite state space, discrete events)
Example: Excerpt from the usage model for a text editor

Probabilistic state machine: States are actions, stochastic sequencing
Testing process

- Any number of test inputs can be generated automatically
  - based on the usage model

- Output correctness predicate?
  - Depends on application
  - Often only plausibility checking is possible

- Measure the intervals between failures
  - Terminate when sufficient reliability can be certified
  - Stop when insufficient reliability has been determined
Reliability certification

• The goal is a statement such as "MTTF(program) ≥ m with confidence K"
  • e.g. "With confidence 95% we can say that this program fails at most once every 2.000.000 steps"

• Computed with statistical methods (binomial distribution)

• Problem:
  When I find and correct a defect, may I still use the data from the previous test runs?
  • Defect models and reliability models may allow this,
  • but then need to rely on assumptions (in particular the non-introduction of new failures).
  • This is beyond the scope of this lecture.
Certification testing: basic idea

Schematic view! Details follow

number of successful test cases

number of failures

refute

continue testing

accept
Details: Binomial distribution

- Given an event (here: failure) with probability $p$ (here: 0.001)
  - i.e. we want to certify 99.9% reliability ($= 1-p$)
- A binomial distribution describes the number $F$ of failures to be expected during $N$ runs (here: $N=3000$)

http://mathworld.wolfram.com/BinomialDistribution.html
Certification testing

- Limit lines for binomial distribution (N trials, p=0.001)

\[ P_{N,p}(F \cdot y) \geq 0.95 \]

\[ P_{N,p}(F \cdot y) \cdot 0.05 \]

continue testing
Cleanroom and CMMI

- CMMI process areas covered by typical Cleanroom practices
- Level 2: Managed
  - Measurement and Analysis MA
    - with respect to reliability only
  - Process and Product Quality Assurance PPQA
    - verification discipline
- Level 3: Defined
  - Technical Solution TS
    - SP 2.3 Design Interfaces Using Criteria: formal specification
  - Verification VER
    - The heart of Cleanroom!
- Level 4: Quantitatively Manag'd
  - Quantitative Project Mgmt QPM
    - Statistical testing
- Level 5: Optimizing
Cleanroom and CMMI (2)

- Cleanroom alone does not get you anywhere with respect to CMMI
  - not even to Level 2
- But the quality culture inspired by Cleanroom was found a useful driver for many improvements up to Level 5:
  - Level 2: PPQA (developers become process-quality-aware);
  - Level 3: VER (reviews become standard practice)
  - Level 4: OPP (defect densities become a natural process benchmark)
  - Level 4: QPM (quantitative quality management is established)
  - Level 5: OID (developers start continuous improvements with respect to defect avoidance, thus opening the organization for process improvement)
Summary: Cleanroom Software Engineering

- Do not accept defects, favor defect prevention over detection
- Exact specification
- Stepwise refinement with box-specification
- Verification during inspection
- Statistical testing based on usage model
- Reliability certification
- Result: very low defect rate, high productivity
Further sources

CleanSoft is a US company specializing in Cleanroom:

- [http://www.cleansoft.com/cleansoft_bibliography.html](http://www.cleansoft.com/cleansoft_bibliography.html)
  - A list of important published works on Cleanroom Software Engineering
- [http://www.cleansoft.com/cleansoft_library.html](http://www.cleansoft.com/cleansoft_library.html)
  - Several nice overview articles by members of CleanSoft
Thank you!