Software Quality Engineering:

Testing, Quality Assurance, and Quantifiable Improvement

Jeff Tian, tian@engr.smu.edu
www.engr.smu.edu/~tian/SQEbook

Chapter 7. Testing Activities, Management, and Automation

- Major Testing Activities
- Test Management
- Testing Automation

Test Planning and Preparation

- Major testing activities:
  - ▷ test planning and preparation
  - ▷ execution (testing)
  - ▷ analysis and followup

- Test planning:
  - ▷ goal setting
  - ▷ overall strategy

- Test preparation:
  - ▷ preparing test cases & test suite(s) (systematic: model-based; our focus)
  - ▷ preparing test procedure

Test Planning

- Goal setting and strategic planning.

- Goal setting
  - ▷ quality perspectives of the customer
  - ▷ quality expectations of the customer
  - ▷ mapping to internal goals and concrete (quantified) measurement.
  - ▷ e.g., customer's correctness concerns $\Rightarrow$ specific reliability target

- Overall strategy, including:
  - ▷ specific objects to be tested.
  - ▷ techniques (and related models) to use.
  - ▷ measurement data to be collected.
  - ▷ analysis and followup activities.
  - ▷ key: plan the "whole thing"!

Test Preparation

- Procedure for test preparation
  - ▷ preparing test cases (model-based)
    - – individual test cases
    - – test case allocation
  - ▷ preparing test procedure
    - – basis for test procedure
    - – order, flow, followup
- General concepts
  - ▷ test run: operation instances
  - ▷ input variable: test point
  - ▷ input space: all possible input variable values
  - ▷ test case: static object + input to enable test to start-execute-finish.

Individual Test Case Preparation

- Individual test cases (micro-level) vs. test suite (macro-level)

- From multiple sources:
  - ▷ actual runs (usage-based).
  - ▷ implementation-based (white-box).
  - ▷ specification-based (black-box).
  - ▷ may use similar/earlier products.
  - ▷ (direct) record and replay (less often).
  - ▷ (via) formal models (OP, CFT, BT, etc.)

- Defining input values (model $\Rightarrow$ test cases):
  - ▷ initial/intermediate/interactive input (expected output too?)
  - ▷ exercise path/slice/track/etc
  - ▷ in testing terminology: sensitization

Test Cases Based on Formal Models

- Most organized, systematic test cases are derived from formal testing models:
  - ▷ directly via newly constructed models.
  - ▷ indirectly via exist test cases, etc.

- Model construction steps:
  - ▷ information source identification and data collection
  - ▷ analysis and initial model construction
  - ▷ model validation and improvement

- Model usage:
  - ▷ defining test cases.
    (details with individual models/techniques)
  - ▷ indirectly in analysis/followup (Part IV).

Test Suite Preparation

- Test suite (macro-level)
  - ▷ existing suite: what and where?
    – suitability? selection/screening?
  - ▷ construction/generation of new ones
  - ▷ organization & management:
    often hierarchical, e.g., sc, sn, vn.

- Adding new test cases
  - ▷ estimate # of new test cases
  - ▷ specify new (individual) test cases
  - ▷ integrate to existing test cases

- Allocation to systems/operations
  - ▷ OP-/structure-based allocation
  - ▷ both old and new test cases in suite

Test Procedure Preparation

- Key consideration: sequencing:
  - ▷ general: simple to complex.
  - ▷ dependency among test cases.
  - ▷ defect detection related sequencing.
  - ▷ sequence to avoid accident.
  - ▷ problem diagnosis related sequencing.
  - ▷ natural grouping of test cases.
- Other considerations:
  - ▷ effectiveness/efficiency concerns.
  - ▷ smooth transition between test runs.
  - ▷ management/resource/personnel/etc.

Test Execution

- Major testing activities:
  - ▷ test planning and preparation
  - ▷ execution (testing)
  - ▷ analysis and followup
- Test execution:
  - ▷ execution planning and management
  - ▷ related activities: important part
    - – failure identification and measurement
    - – other measurement

Test Execution

- General steps
  - ▷ allocating test time (& resources)
  - ▷ invoking test
  - ▷ identifying system failures (& gathering info. for followup actions)
- Allocating test time
  - ▷ OP-based: systems/features/operations
    – also coverage concerns for critical parts
  - ▷ coverage-based: func./struc. areas
  - ▷ alternative: bottom-up approach
    – individual test cases $\Rightarrow$ test time
    – sum-up $\Rightarrow$ overall allocation
    – by OP or coverage areas

Test Execution

- Invoking test (OP-based)
  - ▷ OP $\Rightarrow$ input variables (test points)
  - ▷ follow probabilistic distributions (could be dynamically determined)
  - ▷ sequence (what to test first?): COTS, product, supersystem

- Invoking test (coverage-based)
  - ▷ organize sensitized testcases
  - ▷ sequence $\Leftarrow$ coverage hierarchies

- Common part: Retest due to
  - ▷ defect fix $\Rightarrow$ verify fix
  - ▷ code-base or feature change
  - ▷ general regression test

Test Execution

- Identifying system failures (oracle problem):
  - ▷ similar for OP-/coverage-based
  - ▷ analyze test output for deviations
  - ▷ determine: deviation = failure ?
  - ▷ handling normal vs. failed runs
    - – non-blocking failure handling

- Solving oracle problem:
  - ▷ theoretically undecidable.
  - ▷ some cases obvious: crash, hang, etc.
  - ▷ practically based on heuristics:
    - – product domain knowledge
    - – cross-checking with other products
    - – implementation knowledge & internals
    - – limited dynamic consistency checking

Test Execution

- Failure observation and measurement:
  - ▷ Determine: deviation = failure ?
  - ▷ Establish when failure occurred
    – used in reliability and other analysis
  - ▷ Collect failure information (e.g., ODC):
    – what/where/when/severity/etc.

- Defect handling and test measurement:
  - ▷ defect status and change (controlled)
  - ▷ information gathering during testing
  - ▷ Followup activities:
    – fix-verification cycle
    – other possibilities (defer, invalid, etc.)

Test/Failure Measurement

- Example template: (Table 7.1, p.93)
  – information collected at test execution

- *rid* – run identification, consisting of:
  - ▷ *sc* – scenario class,
  - ▷ *sn* – scenario number,
  - ▷ *vn* – variation number with a particular scenario,
  - ▷ *an* – attempt number for the specific scenario variation
- *timing* – start time *t0* and end time *t1*
- *tester* – the tester who attempted the test run
- *trans* – transactions handled by the test run
- *result* – result of the test run (1 indicates success and 0 for failure)

Testing Analysis and Followup

- Major testing activities:
  - ▷ test planning and preparation
  - ▷ execution (testing)
  - ▷ analysis and followup
- Test analysis and followup:
  - ▷ execution/other measurement analyzed
  - ▷ analysis results as basis for followup
  - ▷ feedback and followup:
    - – decision making (exit testing? etc.)
    - – adjustment and improvement.

Testing Analysis and Followup

- Input to analysis
  - ▷ test execution information
  - ▷ particularly failure cases
  - ▷ timing and characteristics data

- Analysis and output
  - ▷ basic individual (failure) case
    - – problem identification/reporting
    - – repeatable problem setup
  - ▷ overall reliability and other analysis? (Chapter 22 and Part IV)

- Followup activities
  - ▷ defect analysis and removal (& re-test).
  - ▷ decision making and management.
  - ▷ test process and quality improvement.

Testing Analysis and Followup

- For individual test runs:
  - ▷ success: continue with normal testing.
  - ▷ failure: see below.

- Analysis and followup for failed runs:
  - ▷ understanding the problem
    by studying the execution record.
  - ▷ recreating the problem (confirmation).
  - ▷ problem diagnosis
    – may involve multiple related runs.
  - ▷ locating the faults.
  - ▷ defect fixing (fault removal)
    – commonly via add/remove/modify code
    – sometimes involve design changes
  - ▷ re-run/re-test to confirm defect fixing.

Testing Analysis and Followup

- Analysis and followup for overall testing:
  - ▷ reliability analysis and followup.
  - ▷ coverage analysis and followup.
  - ▷ defect analysis and followup.
  - ▷ focus of Part IV.

- Analyses: Different focuses:
  - ▷ overall reliability and coverage for usage-based and coverage-based testing.
  - ▷ detailed defect analysis.

- Followup activities: Similar.
  - ▷ decision making and management.
  - ▷ test process and quality improvement.

Test Management

- People's roles/responsibilities in formal and informal testing.

- In informal testing:
  - ▷ "run-and-observe" by testers.
  - ▷ "plug-and-play" by users.
  - ▷ informal testing with ad-hoc knowledge
  - ▷ deceptively "easy", but not all failures or problems easy to recognize.

- In formal testing:
  - ▷ testers, and organized in teams.
  - ▷ management/communication structure.
  - ▷ role of "code owners" (multiple roles?)
  - ▷ 3rd party (IV&V) testing.
  - ▷ career path for testers.

Test Management

- Test team organization:
  - ▷ vertical: project oriented
    - – product domain knowledge,
    - – staffing/resource management hard.
  - ▷ horizontal: task oriented
    - – even distribution of staff/resources
    - – lack of internal knowledge/expertise
  - ▷ Mixed models might work better.

- Users and 3rd party testers:
  - ▷ user involvement in beta-testing and other variations (e.g., ECI in IBM)
  - ▷ IV&V with 3rd party testing/QA
  - ▷ impact of new technologies:
    - – CBSE, COTS impact
    - – security, dependability requirements.

Test Automation

- Basic understanding:
  - ▷ automation needed for large systems.
  - ▷ fully automated: impossible.
  - ▷ focus on specific needs/areas.

- Key issues to consider:
  - ▷ specific needs and potentials.
  - ▷ existing tools available/suitable?
    - – related: cost/training/etc.
  - ▷ constructing specific tools?
  - ▷ additional cost in usage & support.
  - ▷ impact on resource/schedule/etc.

Test Automation

- Automation by test activity areas:
  - ▷ automated test planning&preparation.
  - ▷ automated test execution.
  - ▷ automated test measurement, analysis, and followup.
  - ▷ slightly different grouping due to tightly coupling for measurement & analysis.

- Automation for test execution.
  - ▷ many debuggers: semi-automatic.
  - ▷ task sequencing/scheduling tools.
  - ▷ load/test generator: script $\Rightarrow$ runs
  - ▷ generally easier to obtain test scripts.

Test Automation: JUnit Example

- P. Louridas, "JUnit: Unit Testing and Coding in Tandem" *IEEE Software*, Vol.22, No.4., pp.12-15, July/Aug., 2005.
  (A nice short survey about JUnit.)

- JUnit example (Fig.1 in paper above)
  - ▷ JUnit test setup:
    initialize some complex numbers
  - ▷ JUnit test cases:
    – execution using "assertEquals(x, y)"
    – base test case: x, y numbers
    – general cases: "expected" = op-result?
  - ▷ $\sum$ test cases $\Rightarrow$ test suite

- Still need:
  - ▷ oracle/"expected" above
  - ▷ test cases $\Leftarrow$ techniques (Ch.8$\sim$12)

Test Automation

- Automation for test planning/preparation:
  - ▷ test planning: Human intensive not much can be done ($\approx$ inspection and FV).
  - ▷ test model construction: similar to above.
    – automation possible at a small scale.
  - ▷ test case generation: focus.

- Test case generation:
  - ▷ from test model to test cases.
  - ▷ specific to individual techniques
    – e.g., cover checklist items, paths, etc.
  - ▷ various specific tools.
  - ▷ key: which specific testing technique supported by the specific tool?

Test Automation

- Test measurement, analysis, and followup.
  - ▷ analyses dictate measurements needed.
  - ▷ most common: reliability/coverage.
  - ▷ defect measurement needed in most cases:
    – defect tracking tools.

- Reliability analysis related tools:
  - ▷ analysis/modeling tools.
  - ▷ collecting execution/input/etc. data.
  - ▷ more in Chapter 22.

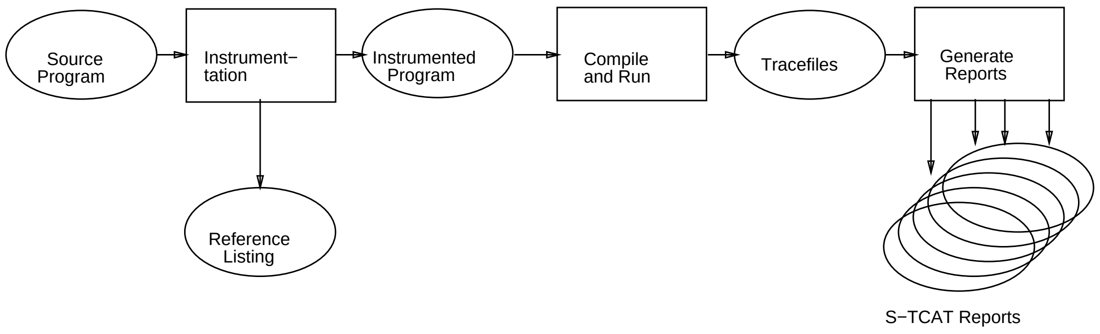
Test Automation

- Coverage-based testing: measuring coverage and compare to pre-set goals.

- Test coverage steps:
  - ▷ preparation: program instrumentation.
  - ▷ measurement step: run and collect data.
  - ▷ analysis step: analysis for coverage.

- Test coverage tools:
  - ▷ different levels/definitions of coverage $\Rightarrow$ different tools.
  - ▷ example tools:
    - – McCabe: execution (control flow) path
    - – S-TCAT: functional coverage
    - – A-TAC: data flow coverage.

Test Automation: Coverage Example



- Test coverage analysis with S-TCAT (Fig 7.1, p.100).
  - ▷ S-TCAT: functional coverage
  - ▷ results: 2 reports:
    1. list of covered functions
    2. function-#times-used

Summary

- Test activities:
  - ▷ planning&preparation: focus of Part II.
  - ▷ execution&measurement: common.
  - ▷ analysis&followup: focus of Part IV.
- Test management:
  - ▷ different roles and responsibilities.
  - ▷ good management required.
- Test automation:
  - ▷ set realistic expectations.
  - ▷ specific areas for automation, esp. in execution, measurement, and analysis.