

# SOFTWARE QUALITY ASSURANCE

## John Abbott College JPC *Course Overview*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Software Quality Assurance

### Course Outline

- Textbook
- Syllabus
- Evaluation
- Expectations

## Textbook

Kaner, C., J. Falk, & H. Q. Nguyen (1993).  
*Testing Computer Software, Second Edition.*  
International Thomson Computer Press (London).  
ISBN 1-85032-847-1. xv + 480. Index.

## Syllabus -- Lectures

### Day 1

- Software QA Case Studies
- Philosophy and psychology of QA
- Inspections / Walkthroughs / Reviews

### Day 2

- Types of testing
  - Module (Unit) Testing
  - Higher-Order Testing

- Types of errors

### Day 3

- Designing good tests
- Automated testing

## Syllabus -- Readings

<u>Day</u>	<u>Chapters...</u>	
1	1:	An example test series
	2:	The objectives and limits of testing
	14:	Legal consequences of defective software
2	3:	Test types and their place in the software development process
	4:	Software errors
	Appendix: Common software errors	
3	7:	Test case design
	11:	Testing tools
	13:	Tying it together

## Expectations

- Classes begin sharp 08:59:45 & end no later than 13:59 (but expecting 11:45)
  - Do not be late
  - Absence from class requires valid explanation else expulsion from course
- Scan chapters in preparation for next day
- Read chapters at end of day and review using class notes
- Write out answers to review exercises from instructor and submit for credit
- Group submissions not accepted: write out answers in your own words (and not merely blindly copied from textbook).

## Evaluation

- Submit answers to exercises by 09:00 each day + at Cont. Ed. reception desk on the day after Day 3
- 25% off *per hour or part of hour* thereafter for late submissions.
- Teacher wants entire class to get 100%

## Disclaimers

The Instructor has free access to CompuServe because of services as Chief Sysop of the NCSA Forums on CompuServe. The National Computer Security Association receives royalties calculated on total amount of time spent by users logged into the NCSA Forums on CompuServe. Except for these benefits, JINBU Corporation and the National Computer Security Association derive no benefit whatsoever from the purchase or other use of the services and products named in this course, nor do JINBU Corporation or the National Computer Security Association own any part of any companies or organizations selling or marketing these services and products. JINBU Corporation and the NCSA make no recommendations of products or services except on contract for specific clients.

## A Preliminary Test

- Here is a triplet of numbers that follows a secret rule:
  - 2,3,5
- Formulate an hypothesis about what underlying rule generates these numbers
- Write down a set of 3 other triplets that will help you decide whether your rule is correct; ask the instructor if your triplets are OK (accord with the actual underlying rule) or not.
- After the class discussion, write down your comments on what you have learned from this exercise and submit your notes tomorrow morning

## SOFTWARE QUALITY ASSURANCE

John Abbott College

### *Quality Assurance*

### *Failures*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Quality Assurance Failures

- Intuit -- Feb 96
- Chase Manhattan -- Mar 96
- California Demographics -- May 96
- Washington DC traffic -- May 96
- First Natl Bank Chicago -- May 96
- ISP Outages -- June - Dec 96
- Microsoft Files Offensive in Spanish -- July 96
- General Motors Recall -- July 96
- Stock Exchange Problems July - Dec 96
- Ent Federal Credit ATMs -- Oct 96
- WorldNet Outage -- Nov 96
- Amtrak Reservations -- Nov 96
- CIBC Debit Cards -- Nov 96

## Intuit Tax Software

Intuit Inc -- Feb 96

- Calculations wrong in US tax-preparation software
  - TurboTax
  - MacInTax
- Updates on Web site
  - workarounds
  - pay penalties

## Chase Manhattan --

Mar 96 -- New York Times

- Letter intended to go to 89 credit-card customers
  - let them know their accounts default
- Went to 11,000 (of a total of 13,000) customers in error
  - users of secured credit-cards

## California Demographics

May 96 -- RISKS 18.10

- CA legislator constantly receiving mail for single parents
- Algorithms in state demographics programs
  - include assumption that different parental surnames on a birth certificate mean parents were not married to each other
- Credibility of statistics in doubt
- Dec 96: similar furor over Consumer Price Index definitions
  - rooted in agricultural/industrial past
  - not suitable for post-industrial service / information economy

## Washington DC traffic

May 96 -- *Washington Post*

- New version traffic-control software
- Switched to wrong pattern
  - from rush-hour (50 seconds of green)
  - to the weekend cycle (15 seconds of green)
- Resulting chaos doubled many people's commute time
- Estimate \$\$ cost to economy. . . .

## First Natl Bank Chicago

May 96 -- AP

- Single largest accounting error in history
  - 900 customers received erroneous transfer
  - \$900M each
- Total error \$763.9B

## ISP Outages

June 96 -- Reuters; RISKS 18.23; others

- Netcom
  - major Internet Service Provider
  - 13-hour blackout on 18 June
  - 1000s customers unable to receive / send e-mail
  - Share price fell from \$33.24 to \$28.75
- AOL
  - went down 1 hr day after Netcom's problem when new software installed
  - also down 19 hr in Aug 96

## Microsoft Files Offensive in Spanish

July 96 -- AP

- Microsoft provides "localized" versions of thesaurus, dictionary, & speller
- Spanish translations caused uproar in Mexico
  - synonyms for "Indian" offensive; e.g., "savage" and "man-eater"
  - "vicious" and "inverted" for "homosexual"
  - "vicious" and "pervert" for "lesbian"
- Public relations disaster
- Company scrambled to provide substitute files

## General Motors Recall

July 96 -- RISKS 18.25

- Major problem with engine software
  - could result in a fire
- 1996 and 1997 model years
- 292,860 Pontiacs, Oldsmobiles and Buicks

## Stock Exchange Problems July - Dec 96

- July -- Johannesburg exchange down twice
  - new software being installed
- Oct -- Cairo Bourse
  - new software
  - significant drop in share prices
- Dec -- Hong Kong Stock Exchange
  - error in automated calculations of index
  - caused panic selling

## Ent Federal Credit ATMs

- Oct 96 -- RISKS 18.53
- Software
  - failed to register multiple identical withdrawals
  - from same account on same day
  - registered only 1st withdrawal
- Consequences embarrassing
  - had been told repeatedly by customers months before
  - forced to debit 12,000 accounts by \$1.2M in total

## Amtrak Reservations

- Nov 96 -- RISKS 18.64
- On 29 Nov, Amtrak lost access to natl reservation & ticketing software
  - Just before heaviest travel period of year
  - Agents usually had no paper schedules or fares
  - Lack of backup caused major delays in helping customers

## CIBC Debit Cards

- Nov 96 -- RISKS 18.65
- 30 Nov in Ontario
  - Major Canadian bank's debit-card system failed several hours
  - Flaw in software upgrade
  - Half of all transactions across eastern Canada were prevented

## Miscellany & Fun

- 2 Belindas -- Feb 96
  - NZ women w/ identical names and birthdates
- Jewish Publication Society -- May 96
  - Judaica CD-ROM got Christian screen saver
- Baby Bonus from IRS -- June 96
  - Pittsburgh 3-yr- old got \$219,495 income-tax refund
- XXX Jeopardy -- June 96
  - Chicago cable TV showed porn in middle
- Dentist's Patience is Taxed -- Dec 96
  - 16,000 copies of same tax form delivered

## DISCUSSION

- Were you surprised by the QA errors you have studied today?
- What do you think the effects of these QA failures were on
  - the organizations involved?
  - the programming managers?
  - the programmers responsible for the bad code?
- Why do you think these errors became public?
- What lessons do *you* draw from these cases for your own work?

## SOFTWARE QUALITY ASSURANCE

John Abbott College JPC

### *Psychology & Economics*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Psychology & Economics of Program Testing

- Definitions & Orientation
- Economics
- Testing Principles

## Definitions & Orientation

- Trying to *find* errors
- NOT trying to show there are no errors
- *Successful* test finds errors
- Problems of language and psychology

## Economics of Testing

- Costs of errors
  - roughly 10x increase at every level of development
  - analysis, design, coding, implementation
- Costs of finding errors
  - must balance cost of error vs cost of finding error
  - possible test cases usually infinite
  - impossible to locate all errors
  - unnecessary to locate all errors: just significant ones

## Economics of Testing

- Black-Box Testing
  - derive test data from specifications only
  - use exhaustive input testing
  - but include all possible wrong inputs too
  - time and money constraints make it impossible to test everything
- White-Box Testing
  - try to execute all possible execution paths
  - but astronomically high # paths
  - and have to multiply by # of inputs

## Some Principles of Testing

- Define expected values
- Use independent testers
- Pay attention to every result
- Include invalid and unusual inputs
- Look for forbidden results
- Record test cases for re-use
- Errors bespeak more errors
- 80/20 rule (Pareto Principle)

# SOFTWARE QUALITY ASSURANCE

## John Abbott College JPC *Program Inspections, Walkthroughs & Reviews*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Human Testing

- Highly effective
  - apply after analysis/design
  - before coding
  - catch errors early = cheaper & better correction
- Inspections
- Walkthroughs
- Desk Checking
- Peer Rating

## Inspections

- Gerald M. Weinberg (1971). *The Psychology of Computer Programming*. Van Nostrand Reinhold (New York). ISBN 0-442-29264-3. xv + 288. Index.
- Team approach
  - moderator
  - programmer
  - designer
  - QA specialist
- Synergy

## Inspections

- Purpose:
  - find errors
  - find reasons errors were made
  - not to fix the errors right then
- Effective
  - find 30%-70% of all errors found by end of testing process
  - complementary to machine-execution testing
- Especially effective for testing modifications

## Inspections

### Moderator

- Competent programmer
- Not author of program
- Distributes materials for inspection
- Facilitates session
- Records results
- Manages repair later

## Inspections

### Narration

- Programmer explains every line of code
- Focus on branch points and operations
- Other members question logic
- Suggest exceptions
- Identify errors
- Do not allow programmer to correct errors during session
  - Alternative view: bug-fixing leads to further analysis

## Inspections

- Prevent interruptions
- Limit sessions to 90-120 minutes
- Average speed 150 3GL statements/hour
  - 4GL may have fewer statements/hour
- Expect repeated sessions

## Inspections

### Psychological Issues

- Defensiveness is a disaster
  - Adopt ego-less attitude
  - Pride in identifying errors
- Results should be confidential
  - Do not allow management to use #errors as metric of programmer quality
- Other Benefits
  - Improved programming style in group
  - Identify error-prone sections

## Walkthroughs

### The Walkthrough Team

- Moderator
- Secretary
- Expert programmer
- Language expert
- Novice
- Maintenance programmer
- Programmer from another project

## Walkthroughs

- Distribute materials in advance
- “Play Computer”
- Use prepared set of test cases
- Mark state of memory etc. on paper or in spreadsheet(s)
- Test cases are merely framework for questions
- Discussions with programmer most productive

## Desk Checking

- Ineffective for most people
- We see what we expect program to do
- Schema influences perception
- Trading programs marginally better

## Peer Rating

- Not a program testing-method
- Good example of Continuous Improvement or TQM
- Choose examples of best and worse code
- Distribute two anonymous samples at random
- Share analysis and commentary
- Fosters positive attitude towards improvement

## Homework

- Read Chapters 1, 2 and 14 from your textbook
- Answer all the review questions distributed by the instructor
  - Avoid copying the textbook blindly -- you will not remember as much as if you think about the answers yourself
  - Use simple language; usually a few words or sentences will be ample
- Submit your work by 09:00 tomorrow morning.
- Because of the short time available, *do not be late* in submitting your review answers

## SOFTWARE QUALITY ASSURANCE

John Abbott College JPC

**Module (Unit)**

**Testing**

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Module Testing

- Definition
- Benefits of Module Testing
- How to Combine Modules?
- Non-Incremental Testing
- Incremental Testing
- Top-Down vs Bottom-Up
- Performing the Test

## Definition

- Module Testing = Unit Testing
- Large programs cannot practically be tested all at once
- Break down programs into modules
- Test modules individually as first phase

## Benefits of Module Testing

- Manage complexity of testing
- Facilitates debugging
- Encourages parallel testing

## How to Combine Modules

- Non-incremental
  - test all modules independently
  - then combine all modules and test whole program
- Incremental
  - add each module to tested collection
  - stepwise retesting

## Testing Modules Alone

How can we execute a subroutine by itself?

- A *driver* program
  - calls a module and
  - passes parameters to it
- A *stub* program
  - represents an as-yet missing module
  - not simply a place-holder
  - must receive data from calling module
  - must return valid values to calling module

## Incremental Testing

- Detects errors in passing parameters among modules
- Helps locate bugs quickly
- Multiple passes through tested modules can lead to more thorough testing
- General sense is that incremental testing is superior to non-incremental testing

## Top-Down vs Bottom-Up

How to add modules?

- Top-down
  - start with master/main/principal module
  - add subordinate (called) modules one at a time
  - need stubs for lower modules
- Bottom-up
  - start with the modules that call no others
  - add superior (calling) modules one at a time
  - need drivers for upper modules

## Top-Down Tests

Practical Issues

- How to pass more than one value from a stub to the module under test?
  - write several versions of the stub
- Add critical modules as soon as possible
  - get it fixed early to prevent later problems
- Add I/O modules ASAP
  - enable one to enter test values
  - can print or display test results

## Bottom-Up Tests

- Problem: no complete skeleton program until end of testing
- Benefits
  - no limitations on test data (no upper modules)
  - do not need separate stubs for different values of test data

## Performing the Test

- Review test cases before using
  - avoid confusion over source of discrepancies
- Automated test tools (more on Day 3)
- Check for pathological effects
  - examine variables that should be unchanged
- Swap modules to avoid self-tests
- Re-use test cases
- Remember the Prime Directive: seek to *find errors*

## SOFTWARE QUALITY ASSURANCE

John Abbott College JPC

### *Higher-Order Testing*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Higher-Order Testing

- Beyond Module Testing
- Integration Testing
- Function Testing
- System Testing
- Acceptance Testing
- Installation Testing
- Test Planning and Control
- Test Completion Criteria

## Beyond Module Testing

- Practical programs must represent real-world needs
- Programs must do what their users expect and demand
- SDLC: System Development Life Cycle
  - requirements: why program is needed
  - objectives: what and how well
  - external specifications: representation of program to its users
  - how program is constructed

## Beyond Module Testing

- Software errors arise from miscommunication
- JAD (Joint Application Development) and RAD (Rapid Application Development)
  - emphasize constant correction
  - by constant communication with users
- Specific testing phases emphasize corrections to specific phases of SDLC

## Integration Testing

- Establish that interconnections among modules function as required
- Implicit in module testing as previously discussed

## Function Testing

- Find discrepancies between program and external specification
- What the program does as a black box
  - User-eye view

## System Testing

- Definition
- Facility Testing
- Stress Testing
- Volume Testing
- Usability Testing
- Security Testing
- Performance Testing
- Storage Testing
- Configuration Testing
- Compatibility / Conversion Testing
- Installability Testing
- Reliability Testing
- Recovery Testing
- Serviceability Testing
- Documentation Testing
- Procedure Testing

## System Testing

### Definition

- Compare program to original objectives
- Cannot base tests on external specifications
  - are attempting to verify conformity between what the ext specs *represent* and actual behaviour of program
  - therefore work with user documentation + program objectives + program
- Must have written, measurable objectives for program

## System Testing

### Facility Testing

- Refers to the documented features or functions
- Scan objectives sentence by sentence
- Look for failure to comply
- Can usually be done without computer

## System Testing

### Stress Testing

- Show that program cannot handle sudden increase in load, demand, input
- Applies to programs that must have minimum throughput or response time
- Distinct from volume testing
  - volume testing looks at total continuous load to process
  - stress testing looks at effects of sudden imposition of load
- E.g., if specs stipulate ability to handle up to 200 concurrent sessions, try suddenly going from 50 users to 200

## System Testing

### Volume Testing

- Show that system cannot handle maximum required amounts of input
- E.g., if program must be able to handle 200 Gb files, test with 200 Gb files and more
- E.g., if specifications stipulate ability to process 10,000 orders in a batch, test with 10,001 orders

## System Testing

### Usability Testing

- Show that normal users will fail to accomplish their documented goals using program
- Clumsy design
- Unsuitable language
- Meaningless error messages
- Inconsistencies in functions from screen to screen
- Inadequate checks on input
- Useless options
- Difficult data entry

## System Testing

### Security Testing

- Try to violate rules of confidentiality, integrity and availability
- “Tiger Teams” specialize in attacking security
- Be sure to obtain authorization for such attacks!
- Use known attacks as described in security literature
- CERT, CIAC Advisories
- USENET postings
- National Computer Security Association

## System Testing

### Performance Testing

- Show that program is unable to meet specified throughput or response-time requirements
- Example of failure:
  - 10 second Service Level Agreement for all transactions
  - 43 minute response time
  - absence of volume testing allowed poor design to pass
- Often combined with volume and stress testing

## **System Testing**

### **Storage Testing**

- Inability to meet requirements for working correctly with specified storage
  - cannot work with minimum RAM
  - minimum required disk space exceeds capacity
- Incompatibilities with
  - RAM management software
  - virtual memory
  - encryption software
  - compression software

## **System Testing**

### **Configuration Testing**

Look for inability to function with specified

- Hardware--RAM, ROM, CPU, peripherals
- Software--operating system, TSRs, drivers
- Program parameters--directories, max, min users / files / records
- Network operating systems
  - versions
  - parameters

## **System Testing**

### **Compatibility/Conversion Testing**

- Failure of conversions from older systems
- Inability to use existing data files
- Conflicts with other legacy systems
- Incompatibility with older operating systems
- Inability to function on older hardware
- Conflicts with older networks

## **System Testing**

### **Installability Testing**

- Difficulties during installation of new product
- Integration with operating systems
- Special installation software

## **System Testing**

### **Reliability Testing**

- Difficult to demonstrate long MTBF
- Monitor rate of discovery of new errors
- Use mathematical models to estimate reliability

## **System Testing**

### **Recovery Testing**

- Show that system cannot recover according to specifications
- Try deliberate “sabotage”
- Measure MTTR
- Look for permanent data damage

## **System Testing**

### **Other types of tests**

- Serviceability Testing
- Documentation Testing
- Procedure Testing

## **Acceptance Testing**

- Performed by client organization
- Conformity to contract
- Includes installation testing
  - methods of showing that installation failed
  - check files, directories, code libraries, databases

## Test Planning and Control

- Too many organizations act as if there will be no errors found
  - No resources for response
  - Inadequate time allocated for repair
- Need extensive planning
- Regression testing
  - Especially important
  - Tests after every change or set of changes

## Test Completion Criteria

- Bad idea
  - Stop when time runs out
  - Stop when no more errors found
- Better way
  - Complete specific methodology
  - But not possible for all phases
  - Subjective
  - Focusses on test method, not goal
- Best approach
  - Find specific number of errors...
  - ... and try for a few more when you find those

## Test Completion Criteria

How to know when enough errors found?

- Statistical calculations
- Mark-recapture of known errors
- Parallel testing by independent teams

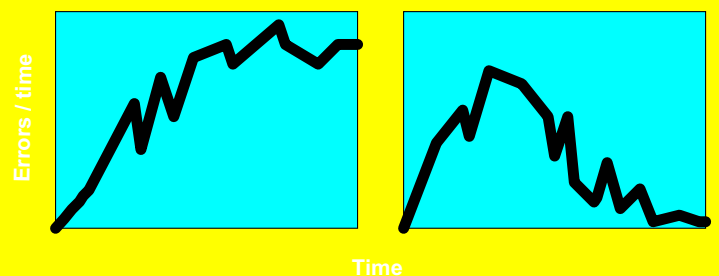
What if there are too few errors in reality?

- Need judgement of test quality
- Independent evaluation

## Test Completion Criteria

Graph errors found per unit time

- Continue searching if success rate high
- Consider stopping when success rate falls



## **Independent Test Agency**

- Hire different organizations for development and testing
- Develop own separate department
- Has worked well in practice
  - high motivation in testing
  - healthy competition
  - development of specialized testing skills

## **SOFTWARE QUALITY ASSURANCE**

**John Abbott College JPC**

### ***Types of Errors***

**M. E. Kabay, PhD, CISSP**  
**Director of Education, NCSA**  
**President, JINBU Corp**

Copyright © 1997 JINBU Corp. All rights reserved

## **Checklists Useful**

Text shows checklist p. 363 ff:

- User I/F
- Error handling
- Boundary-related errors
- Calculation errors
- Initial and later states
- Control-flow errors
- Errors in handling or interpreting data
- Race conditions
- Load conditions
- Hardware
- Source, version and I/D control
- Errors in the testing process

## **User I/F**

- **Functionality**
- **Communication**
- **Command structure and entry**
- **Program rigidity**
- **Performance**
- **Output**

## **Error Handling**

- **Error prevention**
- **Error detection**
- **Error recovery**

## **Boundary-Related Errors**

- **Numeric boundaries**
- **Number and size of parameters**
- **Confines of space**
- **Time-limits**
- **Loops**
- **Memory and storage constraints**
- **Load limitations**
- **Hidden changes in algorithms**

## **Calculation Errors**

- **Logic errors**
  - wrong design or formula
  - wrong breakdown into steps
  - typos
- **Arithmetic errors**
- **Precision problems**

## **Initial and Later States**

- Initialization of variables
- Loop-control variables
- Pointers, flags, registers
- Re-initialization
- Global vs local variables
- Pathological side-effects

## **Control-Flow Errors**

- Program goes haywire
- Program stops
- Program hangs
- Conditions

## **Errors in Handling or Interpreting Data**

- Parameter passing among routines
- Overflows
- Messaging system
- Corruption of data

## **Race Conditions**

- Depending on expected completion sequence
- Checking state before action but without locking
- Deadly embrace

## Load Conditions

- Resource unavailable
- Resources not returned
- Demands contiguous memory
- Input buffer / queue too small
- Fails to clear buffer
- Abbreviation of output
- Defining priorities

## Hardware

- Failing to recognize HW failures
- Destination of output
- Device unavailable
- Returning to wrong pool
- Device forbidden
- Noisy or failed channel
- Time-outs
- Failing to close files
- EOF
- Misunderstood error codes
- Underusing device intelligence
- Initialization

## Source, Version and I/D Control

- Inconsistent version numbers
- Old bugs reappear
- Incomplete update of repeated code
- Missing title at startup
- Missing or wrong version ID
- Missing or bad copyright message
- Source fails to compile exactly to production code
- Distribution media bad, incorrect or virus-infected

## Errors of the Test Process

- Missing bugs
- Finding non-existent “bugs”
- Poor reporting
- Poor follow-up

## Homework

- Read Chapters 3, 4 & Appendix from your textbook
- Answer all the review questions distributed by the instructor
  - Avoid copying the textbook blindly -- you will not remember as much as if you think about the answers yourself
  - Use simple language; usually a few words or sentences will be ample
- Submit your work by 09:00 tomorrow morning.
- Because of the short time available, *do not be late* in submitting your review answers

## SOFTWARE QUALITY ASSURANCE

John Abbott College JPC

### *Test-Case Design*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Test-Case Design

- Design Philosophy
- Equivalence class analysis
- Boundary analysis
- Testing state transitions
- Testing race conditions and other time dependencies
- Function-equivalence testing
- Regression testing
- Error-guessing

## Test-Case Design Philosophy

- Complete testing is impossible
- Therefore define subset of test cases likely to detect most (or at least many) errors
- Intuitive approach is “random-input testing”
  - sit at terminal
  - invent test data at random
  - see what happens
  - worst possible approach

## Equivalence Partitioning

- “A group of tests forms an equivalence class if you believe that:
  - They all test the same thing.
  - If one test catches a bug, the others probably will, too.
  - If one test doesn’t catch a bug, the others probably won’t either.”
    - p. 126
- Subjective process
- Goal is to reduce many redundant tests to a smaller number giving same information
- Focus especially on invalid inputs

## Equivalence Partitioning

Must first identify the equivalence classes

- Range: below, within, above
- Number: fewer, valid, higher
- Set: all members & 1 non-member
- Requirement (set of 1): valid & invalid
- On doubt, split class

## Equivalence Partitioning

Then define specific test cases

- At least one test case for every valid equivalence class
- At least one test case for every invalid equivalence class
- See Figure 7.1, p. 127 in text

## Boundary-Value Analysis

- Cases at boundaries have high value for testing
- Select cases just below, at and just above limits of each equivalency class
- Some testers include mid-range value as well just for additional power of test

## Testing State Transitions

- Every change in output is a state transition
- Test every option in every menu
- If possible, test every pathway to every option in every menu
- Interactions among paths
  - draw menu maps
  - identify multiple ways of reaching every state
  - keep careful records of what you test (can get confusing)

## Testing Race Conditions and Other Time Dependencies

- Check different speeds of input
- Try to disrupt state transitions (e.g, press keys while program switches menus)
- Challenge program just before and just after time-out periods
- Apply heavy load to cause failures (not just poor performance)

## Function-Equivalence Testing

- Use a program that produces known-good output
- Feed same inputs to both the standard program and the program under test
- Compare the outputs
- Automated testing techniques can help
  - for numerical and alphanumeric output
  - for real-time process-control applications

## Regression Testing

- Did the bug get fixed?
  - Some programmers patch symptom
  - Few test effectively
- Check that you can produce bug at will in bad version of code
- Use same tests on revised code
  - Stop if bug reappears
  - Push the testing if bug seems to have been fixed

## Error Guessing

- Need intuitive grasp of what is likely to go wrong in a program
- Look at typically difficult cases (e.g., wrong number of parameters)
- Examine cases that are not explicitly defined in specifications (assumptions by programmer)

## SOFTWARE QUALITY ASSURANCE

John Abbott College

### *Automated Testing*

M. E. Kabay, PhD, CISSP  
Director of Education, NCSA  
President, JINBU Corp

Copyright © 1997 JINBU Corp. All rights reserved

## Why do programs have bugs?

- Involvement makes us blind
- Expectations mask reality
- Interactions are unpredictable
- Testing takes too much time
- Testing is repetitive and tedious

## The Cost of Software Quality

- At least 60% of your development budget
- is used to test only 20-25%
- of your application features

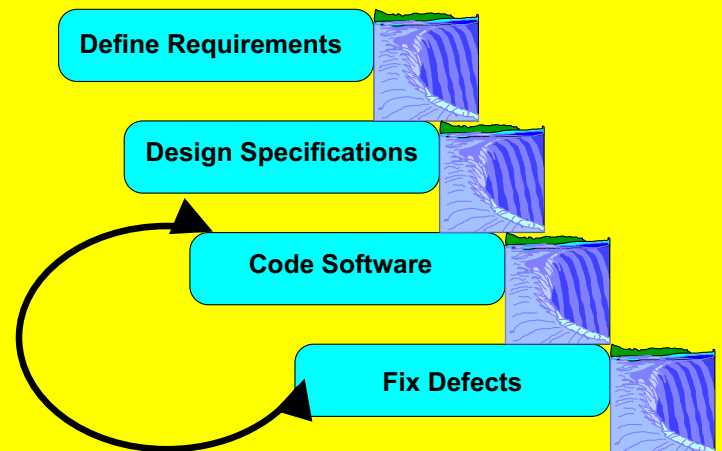
## Quality Assurance Depends on Testing

- Critical examination
- Doing everything feasible to find errors
- Errors are deviations from specification

## To Achieve Software Quality

**We must conduct a critical examination of a system's quality every time we implement a change**

## The System Development Life Cycle (SDLC)



## Why QA?

- Errors in mission-critical systems can be real-world and business disasters
- E.g., software error in chemical-plant in Netherlands
  - allowed operator to enter codes for wrong chemical substances
  - exploded on contact in vat
  - destroyed plant
  - killed two people

## Why automated testing?

- SDLC known as "waterfall"
- Software backlog running 2 years and more
- JAD/RAD more popular than ever
- Therefore user I/F changes constantly

## Current Testing Methods are Inadequate

- Manual input
- Unstructured
- Slow
- Depend on testers' awareness and attention
- Leave no audit trail
- Poor or no statistics
- Manual demonstration of errors

## Consequences of Manual Testing Methods

- Quality is not emphasized during SDLC
- Time pressures always squeeze testing
- Testing never catches all the bugs

## Automated Testing

- Capture/Playback
  - record macros showing mouse movements and alphanumeric input
  - typically no editing language
- Structured Automated Testing
  - tool creates structured, editable script
  - can use databases as source of input
  - intelligent handling of errors

## Limitations of Capture/Playback

- Merely automate manual procedures
- Difficult to maintain as application changes
- Cannot build regression database
- Must wait until application is ready
- No mechanism for detecting errors
- No mechanism for reporting results

## Good Applications Are Easily Maintained and Enhanced

- Structured development
- System documentation
- Metric: ease of reliably changing application
- QA must learn from general programming experience

## Structured Testing

- Modular design
- Documentation
- Segregation of data from procedures
- Re-usability

## Structured Automated Testing

- Define test plan
- Document logic
- Generate test procedures
- Apply test procedures
- Evaluate results

## Benefits of Structured Automated Testing

- Consistent, reproducible testing
- Increased test coverage
- Easier maintenance
- Fully documented testing
- Higher-quality software

## Case Study: COGNOS / Ottawa

- 15 days for testing
- 6 people @ \$300/day
- 3 test phases per product release
- 3000 manual tests per phase
- 12.5% test coverage
- \$81,000 per release @ 12.5%
- \$648,000 per release @ 100%

## Case Study: COGNOS / Ottawa

- "Our goal was to improve the level of testing while at the same time reducing the time and manpower for each release."
  - Doug Clement, COGNOS

## **AutoTester at COGNOS**

- 5 days elapsed time
- 6 people
- 3 test phases
- 24,000 tests/phase
- \$27,000/phase using AutoTester

## **Case Study: HRL CANADA / Ottawa**

- Jacques Joannis, manager
- Built regression database
- 80% of the application features were tested
- Redeployed 60-80% of test group back into development

## **Case Study: New York Life / Toronto**

- Business case showed that up-front cost of implementing automated QA process will be repaid in the first year
- Cost-reduction: lower staffing, shorter cycle
- Cost-avoidance: identifying more errors

## **Sample ROI**

- Take \$1,000,000 SW development budget
- Manual testing costs 60% = \$600,000
- Manual testing: ~20% of application features
- Automated testing: test 90-95% of features
- Automated testing costs 24% = \$260,000
- Real savings: \$300,000 including cost of testing tool

## **Structured Automated Testing**

- Emulates trained human operator
- Single tool for multiple platforms
- Supports corporate standards
- Simplifies training and support

## **PC-based S.A.T. are Platform Independent**

- Works on PCs
- Can test PC programs
- Can test link to hosts using emulators
- Handles online, real-time systems
- Can deal with batch and hard-copy report systems by examining results on screen
- Has been used for ATMs, POS, telephone switches

## **AutoTester Plus Components**

- Script Station
  - Uses screen image and screen definitions
  - Fully generate all scripts for application tests
- Test Station
  - Menu-driven application
  - Uses generated scripts to guide users
  - Users create, manage and execute test cases
  - Apply meaningful and comprehensive tests

## **Solution to the Software Quality Gap**

- Structured development is key to effective maintenance
- Tie test cases to application specs
- Don't write programs to test application
- Test-procedures stored in database

## Independent Analysis

### Gartner Group:

- "AutoTester from AutoTester, Inc. of Dallas reduces testing effort and improves the quality of resulting systems."
- "AutoTester can effectively provide a facility for regression testing most interactive applications at a fraction of the cost and effort required to do so manually."

## Independent Analysis

### Software Quality Engineering:

- "AutoTester is unique because it can test both PC-based applications as well as applications which run on mini or mainframe computers."
- Variable files support in AutoTester allows test cases to be imported from external PC or remote sources, and permits single scripts to exercise unlimited permutations, thus conserving script development and maintenance effort."

## Automated Testing is Suitable for Many Types of Quality Assurance Tests

- Unit tests
- Integration tests
- Regression tests
- Stress tests

## Testing Spans the Organization

- Way of working
- Not gimmick
- Permeates development team
- Management support required
- Involves users
- Plan for pilot project before choosing tool

## Demonstration Disks

- Load the demo disk onto a workstation
- Follow the instructions to proceed through the demonstration.

## Homework

- Readings: Chapters 7, 11, and 13 from your textbook
- Answer all the review questions distributed by the instructor
  - Avoid copying the textbook blindly -- you will not remember as much as if you think about the answers yourself
  - Use simple language; usually a few words or sentences will be ample
- Submit your work by 09:00 tomorrow morning.
- Because of the short time available, *do not be late* in submitting your review answer