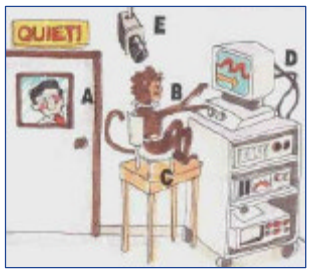
CONCEPT
HEIDELBERG

GMP Compliance for
Computerized Systems Validation
January 16 - 17, 2003 at Istanbul, Turkey

Testing
for Systems Validation



Dr.-Ing. Guenter Generlich
guenter@generlich.de

Testing: Agenda

- Techniques
- Principles
- Levels
- Types
- Data
- Tools
- Mistakes
- Planning
- Costs

Goals of Testing

- Show a system meets its requirements and specifications
- Find and eliminate defects (… not just bugs)
- Find differences between the actual system and its models
- Determine appropriate reliability of the system
- Decide when to release the system in a compliant state
- Use as little resources as possible

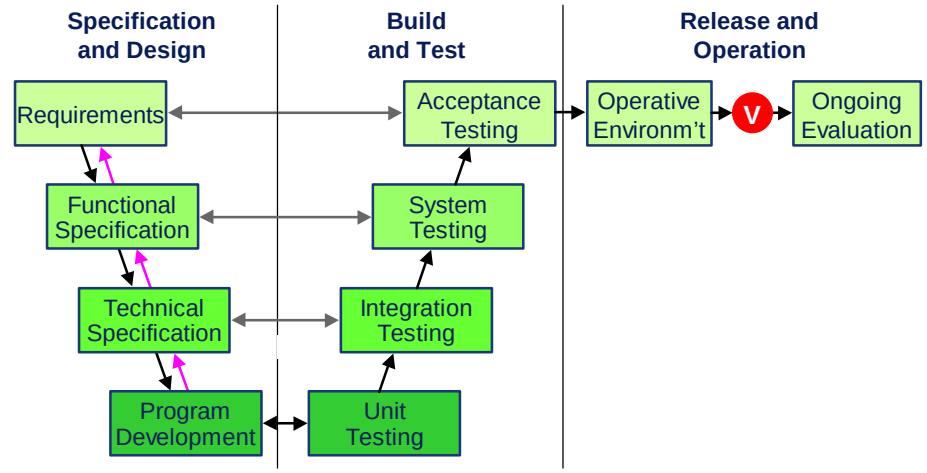
... and for those who like an abstract definition:

- Find cases where the program does **not** do what it is supposed to do
- Find cases where the program does things it is **not** supposed to do

Testing is Not

- Prove that the system is error free
- Prove that the system will work without errors
- Establish that the system performs its functions correctly
- Establish with confidence that the software does its job fully

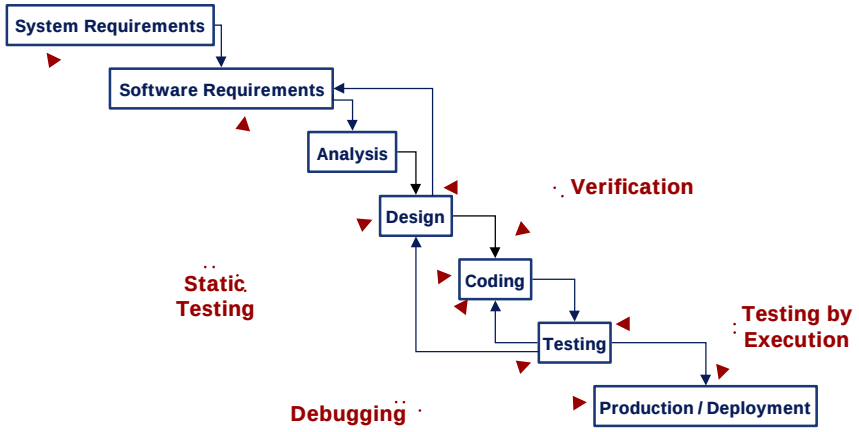
Once Again: The V-Diagram



Computerized Systems Validation
Dr. Guenter Generlich

Testing 6

Testing - Debugging - Verfication



Good Testing Practices (GTP)

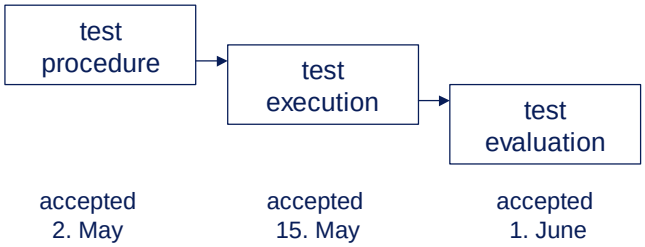
- Test Procedure
- Test Planning
- Test Scheduling
- Test Documentation
- Test Execution
- Test Recording
- Deviations
- Calibrated Tools

Test Procedure GTP1

- Tests are executed according to a pre-defined and pre-approved test procedure (or protocol)
- The test procedure
  - is established on the basis of the requirements and/or appropriate system / equipment specifications
  - refers to the relevant specifications
  - should enable repetition of the test
  - should be based on formal and named documents held under version control according to Good Documentation Practise

Timing of Testing Tasks GTP2

- Testing should not start before the test procedure has been approved
- All test-related dates must be logically consecutive



Test Planning GTP3

- Tests should cover all relevant areas of the relevant equipment or system
- Tests should be planned and executed by persons qualified to the tasks they are executing, e.g.
  - technically skilled
  - knowing the equipment/system
  - trained in the requirements of GTP
- Testers should be as independent as possible
  - exception: acceptance tests
- Do not plan testing assuming that there are no errors

Test Documentation GTP4

The test documentation should

- include name, position and date for authors, reviewers and approvers
- include the test procedure, described in sufficient detail
- show date and signature on each test by the tester and witness or reviewer
- be retained and properly archived

Test Execution GTP5

- There should be pre-determined acceptance criteria or statements of expected results for each test
- During execution test results should be
  - recorded directly onto the test results sheet or
  - refer to printouts or computer generated test execution files (e.g. screen printouts)
- Each test should be concluded with a statement of whether the test met its acceptance criteria
- Test execution should be audited on at least a sample basis by either the user representative or the supplier quality assurance function

Test Recording GTP6

- Manual test recording should use permanent ink
- Shorthand notations such as tick marks should be avoided
- Actual values be recorded where appropriate
- Any corrections should be crossed out with a single line, initialed and dated with a brief explanation
- Correction fluid should not be used

Deviations GTP7

- All deviations should be recorded and be traceable throughout correction and retest into final closure
- Deviation corrections may require regression testing to verify that the corrections did not introduce new problems in other tested areas

Calibrated Tools GTP8

- Any critical instrument inputs and any test equipment should be calibrated with documented evidence of such calibrations, traceable to international standards
- Calibration equipment should be certified, traceable to national standards and referenced

Some Types of System Defects

- Software bugs
- Wrong algorithm
- Wrong functionality
- Missing design or functionality
- Interface specific: internal, external
- Others …
  - weak requirements
  - missing acceptance criteria
  - usability problems
  - ….

Testing Techniques

Black Box Testing

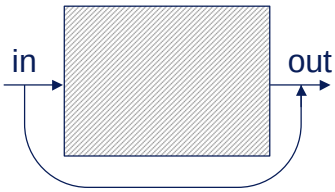
- Assess how well a program meets the requirements
- Assumes the requirements are accepted
- Checks missing or incorrect functionality
- Compares system result with predefined output
- Performance, stress, reliability, security

White Box Testing

- Reveal problems with the internal structure of a program
- Requires detailed knowledge of structure
- Essentially path testing
- Structures can be tested even when structure is vague or incomplete

Testing Techniques

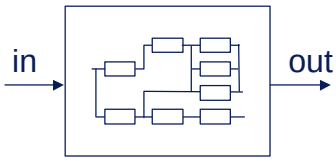
black box - functional



"manual" calculation

acceptance testing
system testing

white box - structural



all little boxes (unit testing) and
linking paths (integration testing)
are individually tested

Testing the Requirements

Testing makes sense and/or becomes possible,
when the requirements are ...

- business related
- verifiable and testable ✍ validatable
- complete and correct
- exact and unambiguous
- actual and/or future directed
- annotated and justified
- internally consistent
- prioritized and timed
- necessary and importand
- below 100%
- uniquely identified
- traceable
- feasible
- manageable
- free of unwarranted design details
- modifiable
- implementation-free
- understood and accepted

Unit Testing

- „Unit“ is an individual component, a function or a small library
- Small enough to test thoroughly
- Exercises one unit in isolation of others
- Easier to locate and remove bugs at this level of testing
- Structural testing in test environment
- Done during code development/programming
- Designed, done and reviewed by programmer
- White Box

Integration Testing

- Units are combined and module is exercised
- Units are tested together to check they work together
- Focus is on the interfaces between units
- Shows feasibility on modules early on
- Tester needs to be unbiased and independent
- Caution: tends to reveal specification errors rather than integration errors
- White box with some black box

System Testing

- The whole system: hardware, software, periphery, documentation, incl. manual parts are tested in detail
- Verify the system correctly implements specified functions
- Testers mimic the end use
- Independent testers and formal approval by (another?) independent function (not developer, tester, or user)
- Ensures ‘softer’ system features are accurately tested (performance, security, reliability, …)
- Black box “alpha testing”

Acceptance Testing

- Completed system tested by end users
- More realistic test usage than ‘system’ phase
- Confirms system meets business/user requirements
- Determine if systems is ready for deployment
- Performed in productive environment
- Black box “beta testing”

Extra Goals for Acceptance Testing

Provide confidence

- of reliability
- of (probable) correctness
- of detection (therefore absence) of particular faults

Testing of Standard Operating Procedures (SOPs)

- Business Change
- Documentation
- Archiving
- Change Management
- Testing
- Error Handling
- Job Description
- Training
- Ongoing / Periodic Review
- Logical & Physical Security
- Start-up & Shutdown
- Backup & Restore
- Maintenance & Repairs

All these can be application independent !

- System Support incl. Help Desk
- Assessments &Auditing (internal and external)
- Service Level Agreements (SLAs)

Testing Rules

- Testing needs to be predefined and repeatable
- Complete testing not possible and not required
- Balance test effort and expert judgement
  - system elements considered
  - degree of details
- A program should not test its own code
- Thoroughly inspect the results of each test
- Special tests for each requirement, e.g.
  - repeated key strokes
  - formal approval
- Base test extend on risks involved
- Highly recommended to work in teams

Regression Testing

"Rerunning test cases which a program has previously executed correctly in order to detect errors spawned by changes or corrections made during software development and maintenance"

FDA Glossary
of Computerized System and
Software Development Terminology

Regression Testing

- Tests modified software
- Verify changes are correct and do not adversely affect other system components
- Selects existing test cases that deemed necessary to validate the modification
- “With bug fixes, four things can happen - three of them are bad”

Types of Systems Testing (1)

Facility	*Does the system provide all the functions required?*
Communications	*Are all communications links working?*
Performance	*What are throughput rates and response times under peak and normal conditions ?*
Volume	*How much data can the system process normally/ continuously?*
Load/Stress	*What happens when we push the system with heavy loads to its limits?*
Configuration	*a - What (extreme/limit) configuration conditions do exist?* *b - Does the system work on all target hardware?*
Reliability	*How reliable is the system over time with a typical workload?*
Serviceability	*How maintainable is the system?*

Types of Systems Testing (2)

Compatibility	*Are there areas where the system has incompatibilities ?*
Recovery	*a - How do we handle hardware and software failures* *b - How well does the system recover from failure?*
Usability	*a - Is the system consistent during use?* *b - Can the users use the system easily?*
Operations	*Are the operators' instructions right and can operators run the system without problems?*
Environment	*How does the new system impact other, existing systems?*
Security	*Can the system withstand attacks or can we find ways to break security provisions?*
Endurance	*Will the system continue to work for long periods?*
Storage	*Are there any unexpected data storage issues?*

Test Data

- Path analysis
- Boundary values
- Common error values
- "Impossible" values
- Random generation
- Actual data

Test-Tools

Computer Aided Testing Tools = CATT

- Test Management
  organization, structure and management of the various testing phases (e.g. planning, preparation, execution, consolidation and evaluation of results, error handling and rerun)
- Function and regression testing
- Load and Performance testing
- New: web testing (function, load, links, …)

… Don't Forget

Also this requires thorough testing:

- User and operations procedures
- Data conversion software and procedures
- Procedures for security, back-up and recovery
- … and all other SOP's

and:

- Involve the users as early as possible
- Train the user before testing
- Monitor the testing process
- Testing can never be complete for non-trivial programs

Metrics

- Host of metrics can be used in testing
- Three themes prevail:
  - Product quality assessment:
    How many defects remain in the system?
  - Risk management:
    What is the risk related to remaining defects?
  - Test process improvement:
    How can we improve our testing results?

Classic Testing Mistakes

- Misunderstanding the role of testing
- Poor planning of the testing effort
- Personnel Issues
- Poor testing methodology
- Too much technology
- (Mis) Use of code coverage

Misunderstanding the Role of Testing

- Test team is responsible for assuring quality
- Purpose of testing is to find bugs
- Usability problems not considered valid bugs
- No focus on an estimate of quality
- Reporting results without context

Poor Planning of the Testing Effort

- Starting testing too late
- Bias towards functional testing
- Incomplete configuration testing
- Delaying load and stress testing until the last minute
- No/incomplete testing procedures
- No/incomplete testing documentation
- Do not rely on beta testing
- Inflexible test plans

Personnel Issues

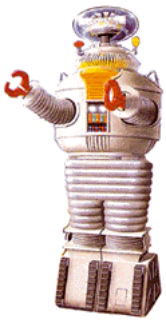
- Testing as a transitional job for new hires
- …or haven for failed programmers
- No domain experts
- Requiring testers who understand
  - Functionality
  - Programming
  - Quality

Poor Testing Methodology

- Paying more attention to execution than to design
- Overly specified test design
- Not exploring “irrelevant” oddities
- Acceptance criteria vague, incomplete, missing
- Not ensuring the product does not do what it is not supposed to do
- Poor bug reporting, poor error handling
- Failing to take notes for the next test effort

Too Much Technology

- Trying to automate all tests
- Expecting regression testing to find a lot of new bugs



(Mis) Use of Code Coverage

- Using coverage as a performance goal
- Selecting test cases based on coverage impact
- Abandoning coverage entirely

Professional Testing

- Defined testing policies
- Test planning process
- Test lifecycle
- Test group
- Test process improvement group
- Test-related metrics
- Tools and equipment
- Controlling and tracking
- Product quality control

Test Documentation

• Test plan	scope, approach, resources, and schedule - updated when system changes
• Test case specification	inputs, drivers/stubs & expected results - updated when system changes
• Test incident report	results of each execution of each test
• Test summary report	lists all failures discovered during tests, evaluation of test results

Test Plan (Example: Acceptance Test)

- Scope and Purpose
- Test Environment
- Assumptions, exclusions, limitations
- Roles and Responsibilities
- Test-Specifications
  - Data
  - Test cases
  - Expected results
  - Acceptance criteria
- Error handling
- Documentation

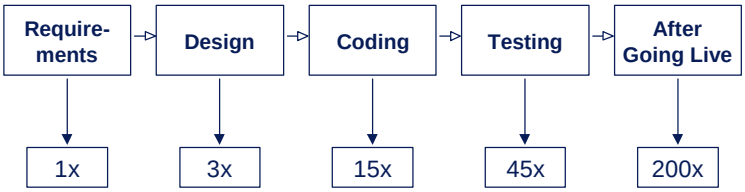
see details at the end of presentation

Test Plan Checklist

- Are the goals of testing for the project defined?
- Is the scope of testing to be performed on the project defined? (e.g. which features to test, which features not to test, localization issues, usability issues)
- Are the testing techniques and strategies to be used on the project outlined?
- Are the artifacts to be produced by testing activities defined?
- Are the risks related to testing and contingency plans for those risks discussed?
- Are any assumptions which may affect the execution of the plan discussed?
- Are the entities or roles & responsibilities for testing described?
- Are information resources upon which testing will depend outlined? (e.g. requirements specifications, design specifications)
- If appropriate, are physical resources related to testing described? (e.g. test lab, software utilities, staffing, schedule)

Costs of a Defect

Cost factors for defect elimination



after Barry Boehm 1981(!!) “Software Engineering Economics”