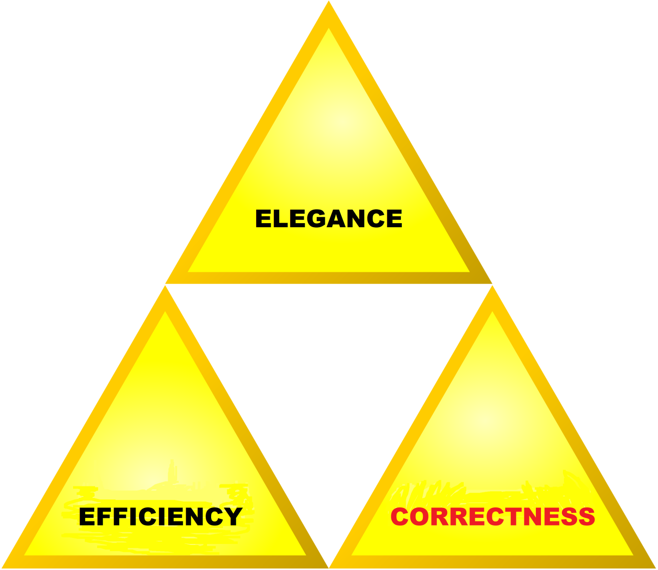
IA169 System Verification and Assurance

Course Intro & Fundaments of Testing

Jiří Barnat

Triforce of Programming



Course Coverage

Topics to be covered ...

- Introduction to Testing
- Symbolic Execution
- Deductive Verification
- Model Checking (4 lectures)
- Bounded Model Checking
- Verification of Real-Time and Hybrid Systems
- Verification of Probabilistic Systems
- CEGAR and Abstract Interpretation
- Assurance, Threat Models, Relevant Security Standards

Prerequisites and Follow-Up

Prerequisites

- Formally none, but we expect ...
- ... capability of basic math reasoning and abstractions.
- ... some experience with coding.
- ... you can handle Unix as a user.

Possible Follow-Up

- IA159 Formal Verification Methods

Course Structure and Marking

Structure

- 2/0/2 + 2 ECTS credits
- Lecture / Consultancy slots / Homework

Evaluation

- Final exam (written test on lectured theory) 50%
- Assignments (six practical tasks) 50%

Grading

- 65% for E or Colloquy or Credit
- 70% for D
- 75% for C
- 80% for B
- 85% for A

Schedule of Lectures and Consultancy Slots

- https://is.muni.cz/auth/el/fi/podzim2020/IA169/um/IA169_semester_schedule_2020.pdf

Fundaments of Testing

http://www.testingeducation.org/BBST/

Testing

Testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.

Empirical Technical

- Conduct experimental measurements.
- Logic and math.
- Modelling.
- Employs SW tools.

Investigation

- Organised and thorough.
- Self-reflecting.
- Challenging.

Testing

Product or Service

- Software.
- Hardware.
- Data.
- Documentation and specification.
- ... other parts that are delivered.

Information

- Not know before.
- Has some value.

Stakeholders

- Who has interest in the success of testing effort.
- Who has interest in the success of the product.

Fundamental Questions of Testing

Mission

- Why do we test? What we want to achieve?

Strategy

- How to proceed to fulfil the mission efficiently?

Oracle

- How to recognise success of the test.

Incompleteness

- Do we realise that testing cannot prove absence of error?

Measure

- How much of of our testing plan has been completed?
- How far we are to complete the mission?

Mission of Testing

Most Typical Mission

- Bug hunting.
- Identification of factors that reduce quality.

Other Missions

- Collect data to support manager decisions, such as: Is the product good enough to be released?
- How much different is the product from product available on market?
- Is the product complete with respect to specification?
- Are individual components logically and ergonomically connected.
- ...

Other Missions – continued

- Support manager decision with empirical results.
- Evaluate the cost of support after release.
- To check compatibility with other products.
- Confirm accordance with the specification.
- Find safe scenarios of product usage.
- To acquire certification.
- Minimise consequences of low quality.
- Evaluate the product for third party.
- ...

Strategy

Strategy

Strategy is a plan, how to fulfil the mission in the given context.

Example: Consider spreadsheet computation in four different contexts.

a) Computer game.
b) Early stage of development of database product.
c) Late stage of development of database product.
d) Driver for medical X-ray scanning device.

Question:

- Will you proceed with the same strategy?

Example – continued

What factors influence strategy selection

- What is the mission?
- How aggressively we need to detect bugs.
- What bugs are less important than others?
- How thoroughly testing will be documented?

Discussion

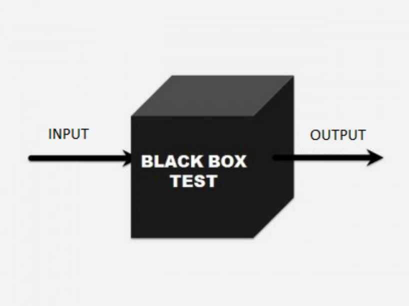
- Assume that a program has an enter field that is expecting numerical values. Is is meaningful to test the product for situation when we enter non-numeric value? (Not mentioned in specification at all.)

Testing Strategies

Black-box Testing

Black-box

- A product under test is viewed as a **black box**.
- It is analysed through the input-output behaviour.
- Inner details (such as source code) are hidden or not taken into account.



White-box, Gray-box Testing

White-box Testing (Glass-box)

- Inner details are taken into account.
- Tests are selected and executed with respect to the inner details of the product, e.g. code coverage.
- Error insertion, modification of the product for the purpose of testing.
- **Basically only extends any Black-box approach.**

Gray-box Testing

- In between of Black-box and White-box.
- Sometimes the same as White-box, inconsistent terminology.

Testing Techniques

Primary Black-box Strategies

- Domain Testing
- Combinatory Testing
- Scenario Testing
- Risk-based Testing
- Functional Testing
- Fuzz Testing (Mutation Testing)

Primary White-box Extensions

- Model-based Testing
- Unit Testing

Support for Developers

- Regression Testing

Section

Oracle

Definition of Oracle

Oracle (in the context of testing) is a detection mechanism or principle to learn that the product passed or failed a test.

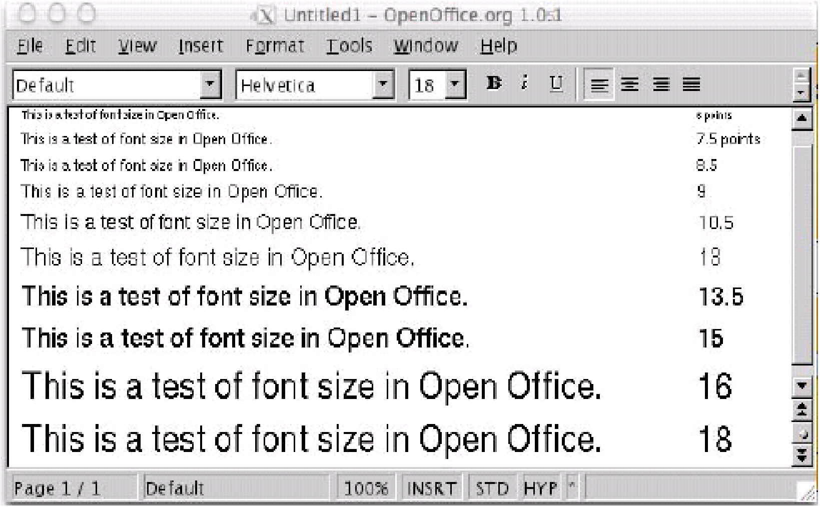
Facts

- If tester claims that the program passed a test, it does not mean the program is correct with respect to the tested property. It depends on the oracle used.
- Basically, any test may fail or pass with a suitable oracle.

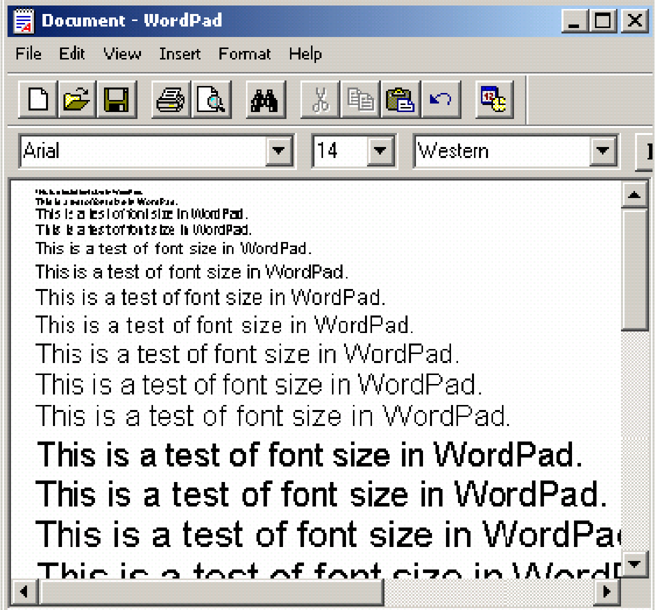
Example

- Does font sizes work properly in OpenOffice, WordPad, and MS Word text editors?

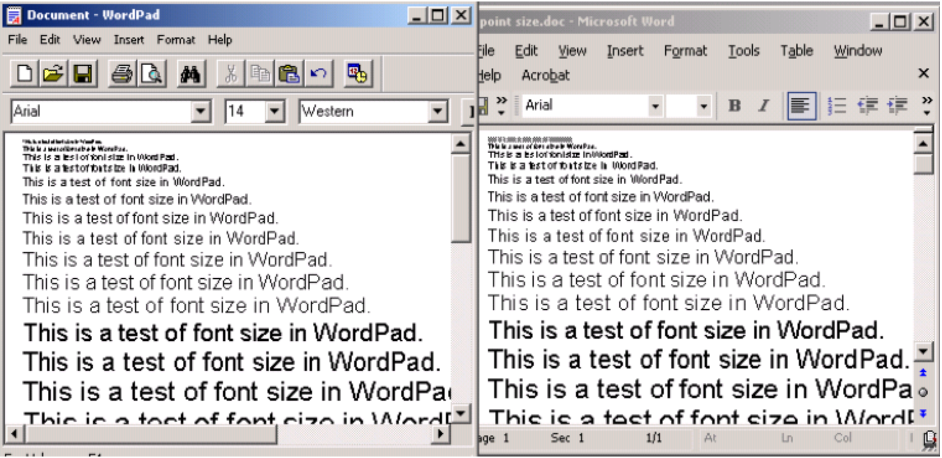
Example – OpenOffice 1.0



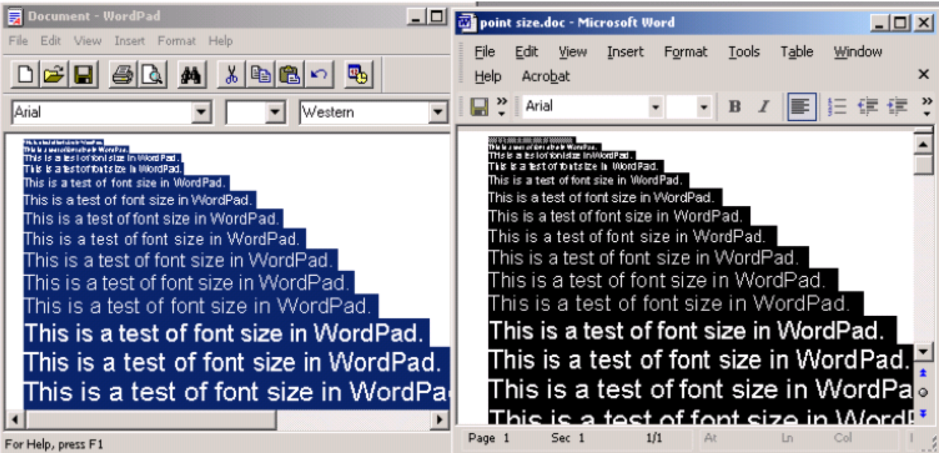
Example – WordPad



Example – WordPad versus MS Word



Example – WordPad versus MS Word (highlighted)



Example – Decisions

Questions

- Is the observed difference in font sizes a bug in WordPad?
- Is the observed difference in font sizes a bug in MS Word?
- Is the observed difference in font sizes a bug at all?

Possible Conclusions

- We do not know if sizes are correct, but we have tendency to believe MS Word rather than to WordPad.
- For WordPad it is not necessary to stick precisely to typographic standards.
- For WordPad it is possibly a bug, but definitely it is not a problem.

Example – Risk-Based Testing

Possible (Pragmatic) Position

- It is/isn't a bug? $\Longrightarrow$ It is/isn't a problem?
- It is necessary to know the context, to guess the metrics that the final consumer will use to judge the issue.
- With some risk we can achieve simplification of the decision.

Simplification in Testing Process

- Avoid tests that obviously does not reveal any problems.
- Avoid tests that obviously reveal only uninteresting problems.

Example – Judge Criteria

How much do we actually know about typography?

- Point definition is unclear. (`http://www.oberonplace.com/dtp/fonts/point.htm`)
- Absolute sizes are difficult to measure. (`http://www.oberonplace.com/dtp/fonts/fontsize.htm`)

From Uncertainty to Heuristics

- How precisely must the sizes agree in order to declare that the sizes are correct?
- Obtaining complete information and evaluation of all the facts is too complicated and costly.
- **Heuristics** are used instead.

Oracle Problem – Heuristics

Decision Heuristics

- Allows for simplification of decision problem.
- Advice, recommendation, or procedure to be used within the given context.
- Should not build on any hidden knowledge.
- Does not guarantee a good decision.
- Various heuristics may lead to contradictory decisions.

Disadvantages

- Heuristics might be subjective.
- If misused, may cause more harm than good.

Consistency as Heuristics

Consistency

- Good heuristics for decision making.

Consistency with ...

- other functions of the product, similar products, history, producer image, specifications, standards, user expectations, the purpose of the product, etc.

Advantages

- Consistency is objective enough.
- Easily described in bug report.

Imperfection in Decision Making

Unintentional Blindness

- Human tester does not consider any test outputs that he/she does not pay attention to.
- Similarly, mechanical tester does not consider test outputs that it is not told to include into decision.

Uncertainty Principle

- The presence of observer may affect what is observed.

Consequence

- It is impossible to observe all possible outputs from a single test.

Oracle and Automation Process

Motivation

- Automation process eliminates human errors.
- Automation leads to repeatable procedures.
- Automation allows faster test evaluation.

Problems of Automation

- It is necessary to automate the decision making (oracle) principle.
- Can we do it? Only partially.

Standard Way of Oracle Automation

- A file of expected outputs, which is required to match precisely with the outputs of a test being executed.
- Example: MS Word could be used to define a the file of expected outputs for testing WordPad.

Problems of Automated Oracle

Amount of Agreement

- Assume MS Word to serve as the file of expected outputs for testing WordPad.
- How exactly is the expected output stored?
- Is 99% agreement still agreement?
- How is the percentage of agreement defined?

False Alarms

- Using outdated expected output.
- Consequence of simplification of decision making.

Undiscovered Errors

- Expected file exhibits the same error as test output.
- Unintentional Blindness.

Measure Methods in Testing

Coverage as a Measure

Coverage

- A set of source code entities that has been checked with at least one test.
- Source-code entities: lines of code, conditions, function calls, branches, etc.
- Used to identify parts that have not been tested yet.

Coverage as a Measure

- Possible test plan is to achieve a given percentage of coverage.
- The percentage than expresses how much of the final product has been tested.
- Numeric expression for managers to see how much of the product remains to be tested.

Coverage as a Measure – Disadvantages

Problems

- Could avoid testing of interesting input data.
- Does not properly test parts of the product that rely on external services.

Using Coverage as a Measure

- The mission is to test all entities of the product, is that OK?
- Complete coverage does not guarantee quality of the product.
- Stimulates to prefer quantity rather than quality.
- Misleading satisfaction (shouldn't feel safe).

Coverage as a Measure – Disadvantages

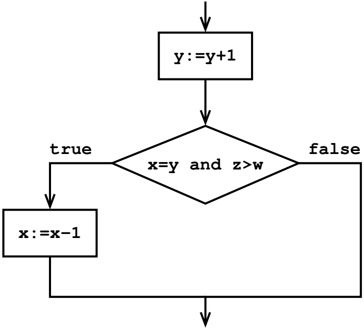
Example

```
Input A        // program accepts any
Input B        // integer into A and B
Print A/B
```

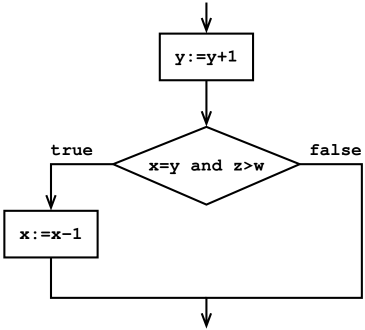
Observation

- Complete coverage is easy achievable.
- For example:
  input: 2,1
  output: 2
- There is of course a hidden bug in the program!

Coverage Criteria for Control-Flow Graphs

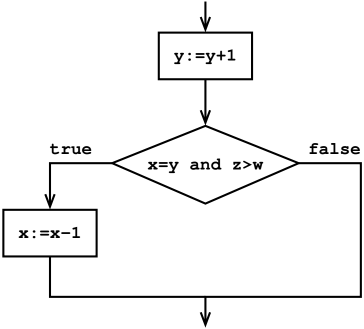


There are various criteria for control-flow graph coverage.



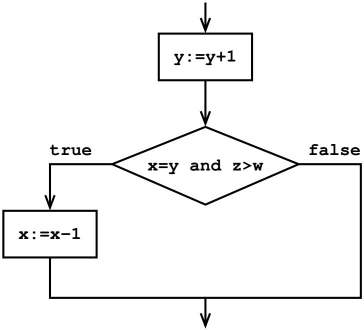
Statement coverage

- Every statement (assignment, input, output, condition) is executed in at least one test.
- Set of tests to achieve full coverage:
  $(x = 2, y = 1, z = 4, w = 3)$



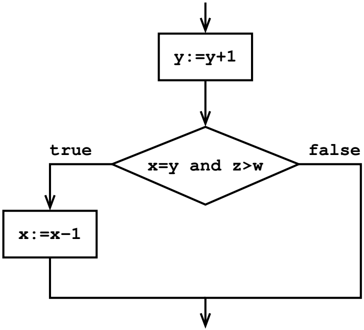
Edge coverage

- Every edge of CFG is executed in at least one test.
- Set of tests to achieve full coverage:
  $(x = 2, y = 1, z = 4, w = 3)$, $(x = 3, y = 3, z = 5, w = 7)$



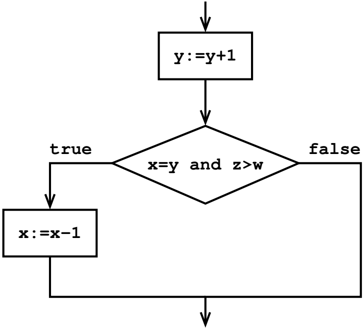
Condition coverage

- Every condition is a Boolean combination of elementary conditions, for example $\mathtt{x} < \mathtt{y}$ or $\mathtt{even}(\mathtt{x})$.
- If it is possible, every elementary condition is evaluated in at least one test to TRUE and in at least one test to FALSE.



Condition coverage

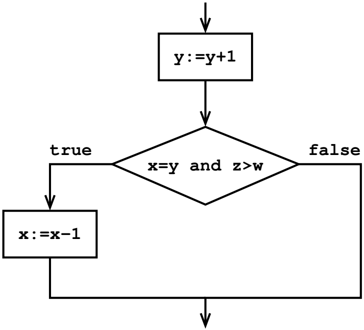
- Set of tests to achieve full coverage:
  $(x = 3, y = 2, z = 5, w = 7)$, $(x = 3, y = 3, z = 7, w = 5)$
- In both cases, only the FALSE branch of IF statement is taken.



Edge/Condition coverage

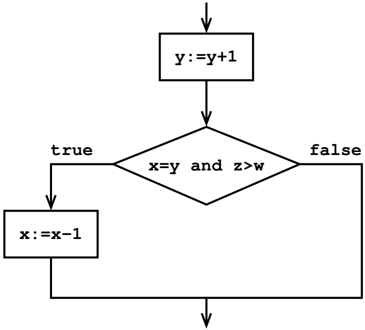
- Edge and Condition coverage at the same time.
- Set of tests to achieve full coverage: $(x = 2, y = 1, z = 4, w = 3)$, $(x = 3, y = 2, z = 5, w = 7)$, $(x = 3, y = 3, z = 7, w = 5)$
- Is the set the smallest possible one?

Coverage Criteria for Control-Flow Graphs



Multiple condition coverage

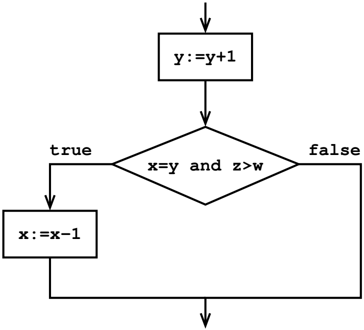
- Every Boolean combination of TRUE/FALSE values that may appear in some decision condition must occur in at least one test.



Multiple condition coverage

- Set of tests to achieve full coverage:
  $(x = 2, y = 1, z = 4, w = 3)$, $(x = 3, y = 2, z = 5, w = 7)$,
  $(x = 3, y = 3, z = 7, w = 5)$, $(x = 3, y = 3, z = 5, w = 6)$
- Exponential grow in the number of tests.

Coverage Criteria for Control-Flow Graphs



Path coverage

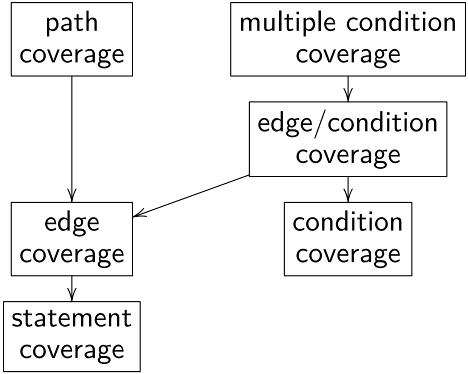
- Every executable path is executed in at least one test.
- The number of paths is big, even infinite in case there is an unbounded cycle in the control-flow graph.

Hierarchy of Coverage Criteria

- Criterion A **includes** criterion B, denoted with A $\rightarrow$ B, if after full coverage of type A we guarantee full coverage of type B.

Hierarchy of Coverage Criteria

- Criterion A **includes** criterion B, denoted with A $\rightarrow$ B, if after full coverage of type A we guarantee full coverage of type B.



Cycle Coverage

Coverage and Number of Cycle Iterations

- All criteria except the path criterion does not reflect the number of iterations over a cycle body.
- In case of nested cycles, systematic testing of all possible executable paths become complicated.

Ad hoc Strategy for Testing Cycles

- Check the case when the cycle is completely skipped.
- Check the case when the cycle is executed exactly once.
- Check the case when the cycle is executed the expected number of times.
- If a boundary n is known for the number of cycle executions, try to design tests where the cycle is executed $n-1$, n, and $n+1$ times.

Coverage for Data-Flow Graphs

Motivation

- Detect usage of undefined variables.
- On some paths, a variable may be set for a specific purpose and later on its value misused for other purpose.
- Control Flow criteria do not guarantee inclusion of tests for above mentioned or likewise situations.

Data Flow Coverage

- Cover paths through control flow graph that go through a location where a variable is used but it is not defined along all incoming paths through control-flow graph.

Support for Code Coverage

C/C++, Linux

- Tools `gcov` and `lcov`.

Example: lcov

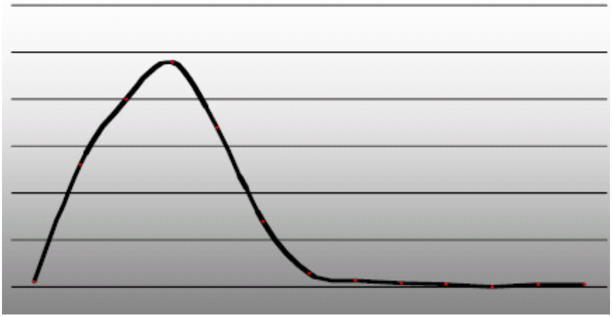
- gcc -fprofile-arcs -ftest-coverage foo.c -o foo
  lcov -d . -z
  lcov -c -i -d . -o base.info
  ./foo
  lcov -c -d . -o collect.info
  lcov -d . -a base.info -a collect.info -o result.info
  genhtml result.info

Statistics on Found/Fixed Bugs per Time Unit

Week statistics

- The number of newly discovered errors.
- The number of fixed errors.
- The ratio of found versus fixed errors.

Visualisation



Weibull Distribution

Observation

- The number of discovered errors per time unit exhibits Weibull Distribution.
- Can be used as a measure for the remaining amount of testing.
- Software Engineering Method to set the release date.

Using Weibull Distribution

- At the moment the curve reaches the peak, the remaining part of the curve may be predicted, hence, a moment in time may be set, when expected number of errors discovered per week drops below a given threshold.
- Parameters of Weibull distribution influence the “width” and “height/slope” of the peak.
- $F(x) = 1 - e^{-ax^{-b}}$ for $x > 0$

Weibull Distribution – Imperfections

Vague Precision

- Testing does not follow the typical usage of the product.
- The probability of error discovery is different for different errors.
- Fix may cause other new errors.
- Bugs are dependent.
- The number of errors in the product changes over time.
- Error insertion exhibits Weibull distribution itself.
- Testing epochs (various testing procedures) are independent.
- ...

Conclusions

- Weibull Distribution is not very reliable.
- Can be used only with large projects for very rough estimation.

Impact of following Weibull Distribution

Assumption

- Software developers are aware of being measured.

Phase one

- Reach the peak as quickly as possible.
- Double reporting of errors.
- Avoid fixing known errors.
- ...

Phase two

- Stick to expected shape of the curve.
- Delay reporting of errors.
- Reporting outside bug-tracking system.
- ...

Incompleteness of Testing

Definition

Observation

- The amount of tests to be run is extremely large.
- Resources for testing are always limited.

What Is Not Complete Testing

- Complete Coverage
  - Every line of code.
  - Every branching point.
  - ...
- When testers do not find more errors.
- Testing plan is finished.

What Is Complete Testing

- There are no hidden or unknown errors in the product.
- If new issue is reported, testing could not be complete.

Reasons for Incompleteness of Testing

The number of tests is too large (infinitely many).

To perform all tests means:

- To test all possible input values of every input variable.
- To test all combinations of input variables.
- To test every possible run of a system.
- To test every combination of HW and SW, including future technology.
- To test every way a user may use the product.

Impossibility to Test All Possible Inputs

Data Bus-Width

- The number of tests grows exponentially with respect to bit used for data representation.
- Domain encoded with n-bits requires 2^n tests.

Other Reasons

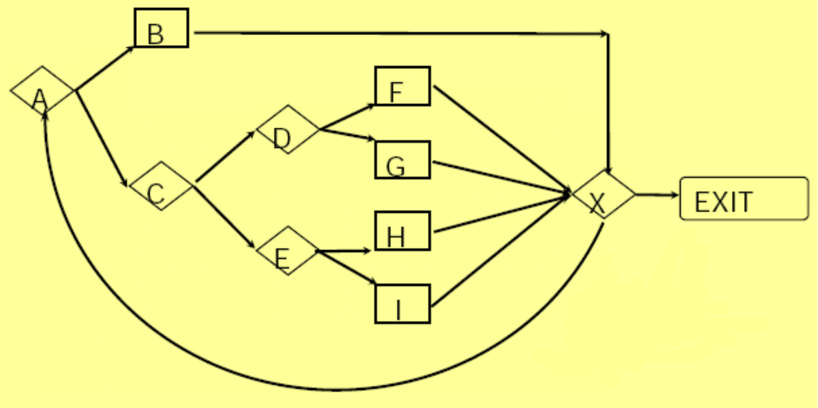
- Timing of actions
- Invalid or unexpected inputs (buffer overflow).
- Edited inputs
- Computer Easter egg

Common Argumentation

- "This is not what the customer would do with our product."

Incapability to Test All Runs

Assume the following system



Questions

- How many different ways it is possible to reach EXIT ?
- How many different ways it is possible to reach EXIT , if $\langle A \rangle$ can be visited at most n-times?

Incapability to Test All Runs

Example

- In [F] is a memory leak, in [B] garbage collector.
- System will reach an invalid state, if [B] is avoided long enough.

Observation

- Simplified testing of paths may not discover the error.
- The error manifests in circumstances that cannot be achieved with a simple test.

Summary for Measure and Incompleteness

Incompleteness

- Testing cannot prove absence of error.
- It is impossible to test all valid inputs.
- Existence of testing plan inhibits testing creativity.

Measure

- There are methods to measure progress in testing phase.
- These are unreliable.
- Focusing strongly on a selected measure may influence the effectiveness of testing.

Self-study

- Reading on MC/DC: `http://shemesh.larc.nasa.gov/fm/papers/Hayhurst-2001-tm210876-MCDC.pdf`
- List, and briefly describe as many black-box testing approaches as you can find or are aware of. `http://www.testingeducation.org/BBST/`
- Optional: Learn about CMAKE and CTEST systems.