A Perspective on Testing

Introduction to Software Testing

Motivation

- Judging (V&V)
  - Software Quality (not crash too often 不要當機)
    - doing things right (verification 事情做**好**)
    - meet specification
    - internal process for development
  - Acceptability (meet requirement 正確處理)
    - doing right things (validation 做**對**的事)
    - meed user’s needs
    - external process for the user
- Discovering
  - Triggering Problems in the Software
- Debugging Support
  - Localize Problems

Definitions (http://istqb.org/downloads/glossary.html)

- Error (bug)
  - mistake
  - coding error is called bug
  - during development process
- Fault (defect)
- Failure
- Incident
- Test
- Test case

Definitions

- Error (bug)
- Fault (defect)
  - the result of an error
  - the representation of an error
  - defect
  - faults types
    - fault of commission (enter something incorrect)
    - fault of omission (fail to enter correct info)
      - hard to detect
      - code review

- Failure
- Incident
- Test
- Test case

Definitions

- Error
- Fault

- Failure
  - code corresponding to a fault executes
  - failures only to faults of commission
  - how about faults of omission
    - code reivew
- Incident
- Test
- Test case

Definitions

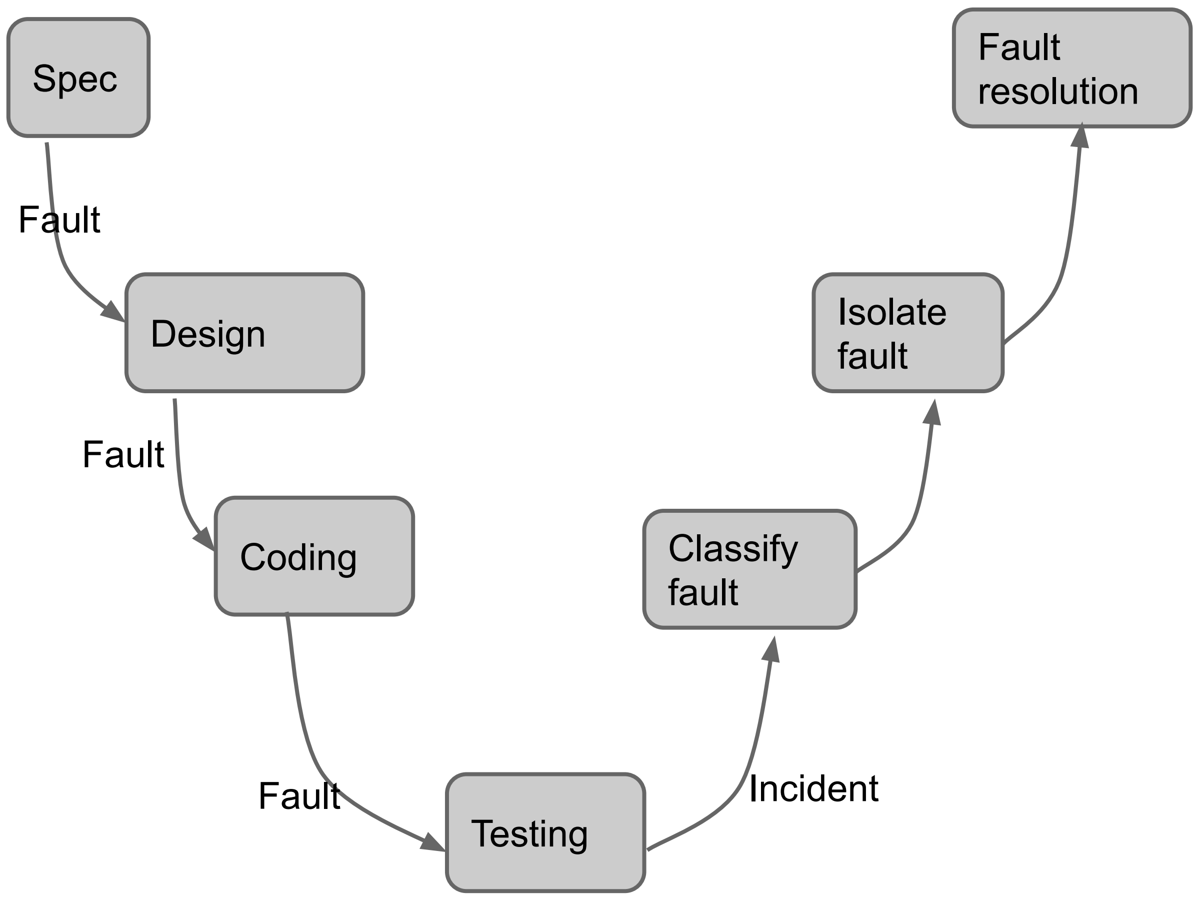
- Error
- Fault
- Failure
- Incident
  - symptom associated with a failure alerting the user to the occurrence of a failure
- Test
  - exercise software with test cases
  - to find failures or to demonstrate correct execution
- Test case

Definitions

- Error
- Fault
- Failure
- Incident

- Test
  - exercise software with test cases
  - to find failures or to demonstrate correct execution
- Test case
  - with an identity associated with a program behavior
  - with a set of inputs and expected outputs

A testing life cycle



Testing Process

- Test Planning
- Test Case Development
- Running Test Cases
- Evaluating Test Results

Test Cases

- Test case identifier
- brief statement of purpose (business rule)
- description of precondition
- actual test case inputs
- expected outputs
- description of postconditions
- execution history (for test management)
    - date when the test was run
    - person who run it
    - version on which it was run
    - pass/fail result

About the “Output” portion

- Judgement if outputs of an executed set of test case inputs are acceptable
  - Optimal solution, or
  - reference testing
    - tested in the presence of expert users

reference implementation

Test case execution

重點：in > out > compare < ensure

- establish necessary precondition
- provide test input
- observe the output
- compare these with the expected output
- ensure the expected postcondition to see if the test passed

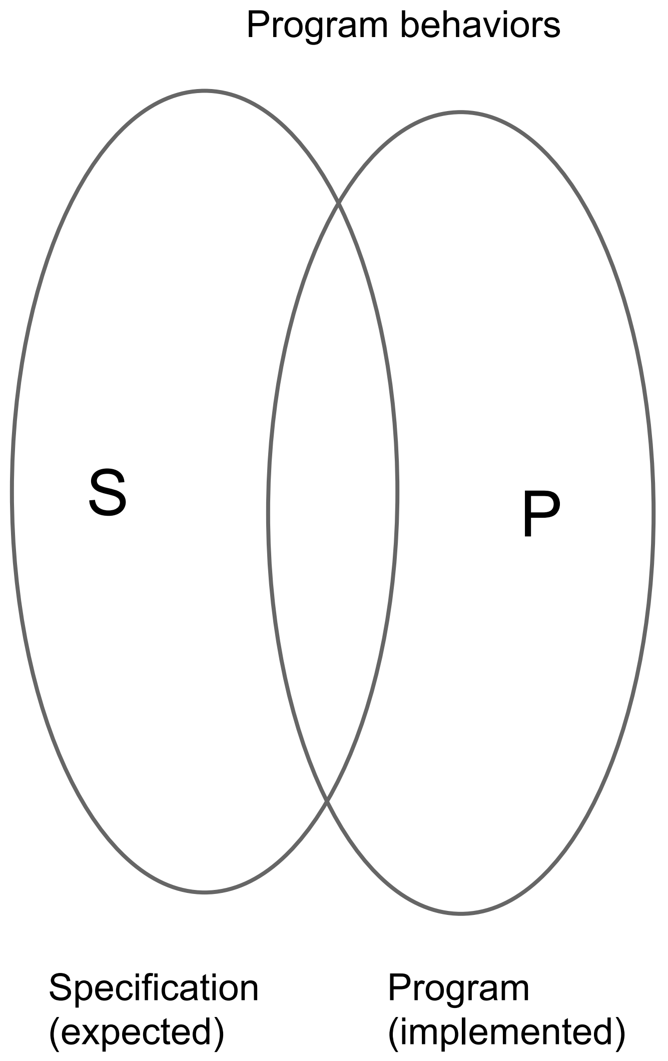
Test Case are Valuable (as source code)

- Test cases need to be
  - developed
  - reviewed
  - used
  - managed
  - saved
- Moreover
  - influence “vague design” : TDD (test driven development)
  - influence “detailed design” : TFD (test first development)

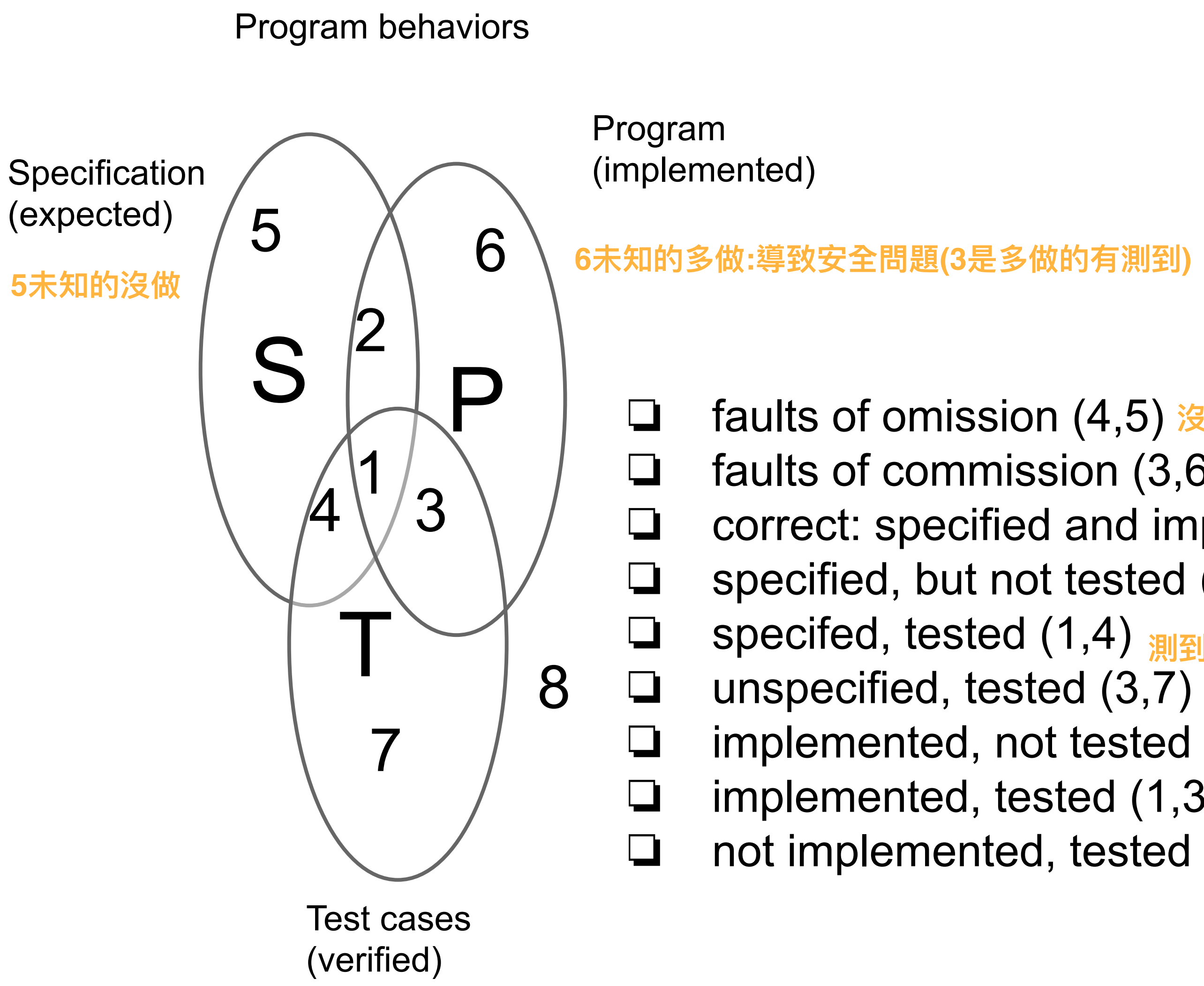
TDD and TFD

- TDD
  - vague design
  - no ideas on how to design and write test first without specific APIs to “ask questions”
  - help form the ideas on what we want to do
  - more on “design” and “development”
- TFD
  - detailed design
  - with designed APIs, write test first with the specific APIs before coding to “answer problems”
  - help relealize the ideas on how we can do
  - more on “implementation”

Specified and implemented program behaviors



Specified, implemented, and tested behaviors



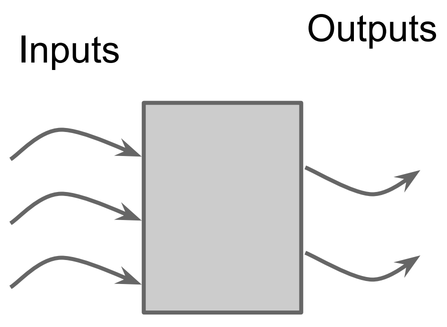
- ❑ faults of omission (4,5) 沒做
- ❑ faults of commission (3,6)
- ❑ correct: specified and implemented (1,2)
- ❑ specified, but not tested (2,5) 測不到
- ❑ specifed, tested (1,4) 測到但沒做
- ❑ unspecified, tested (3,7) 不在規格但測到
- ❑ implemented, not tested (2,6) 多做測不到
- ❑ implemented, tested (1,3) 做了測到
- ❑ not implemented, tested (4,7) 沒做但測到

Identifying Test Cases

- Functional Testing
  - Spcification-based
- Structural Testing
  - Code-based

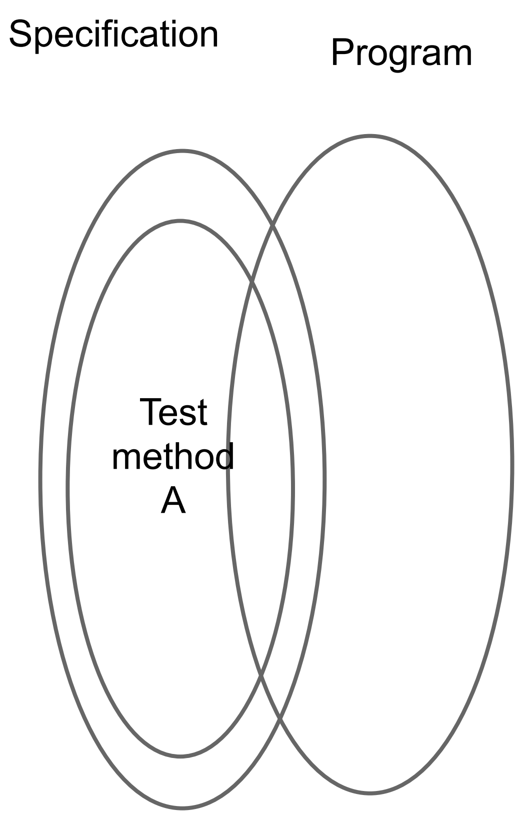
Specification-Based Testing

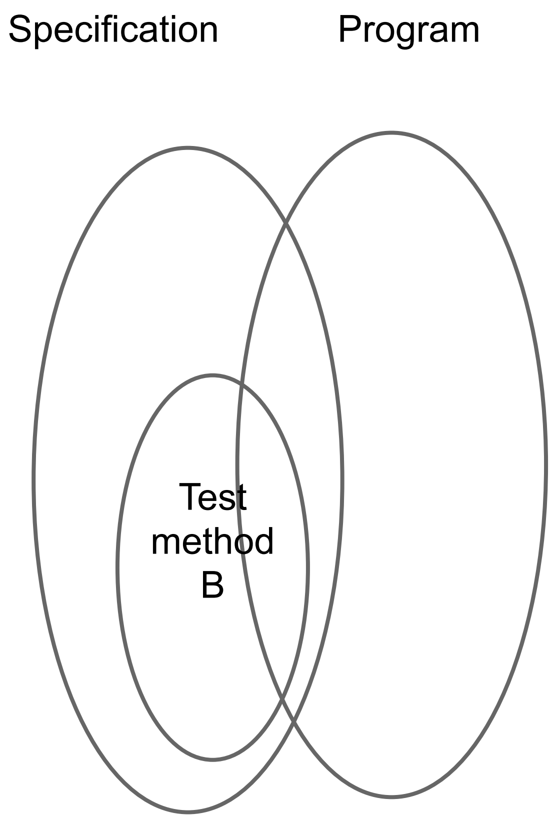
- functional testing
- black box testing
- advantages
  - implementation independent
  - test case development occurs in parallel with implementation
- disadvantages
  - significant redundancies exist among test cases
  - compounded by the possibility of gaps of untested software
  - hard to find unspecified behaviors



Engineer's black box

Comparing specification-based test





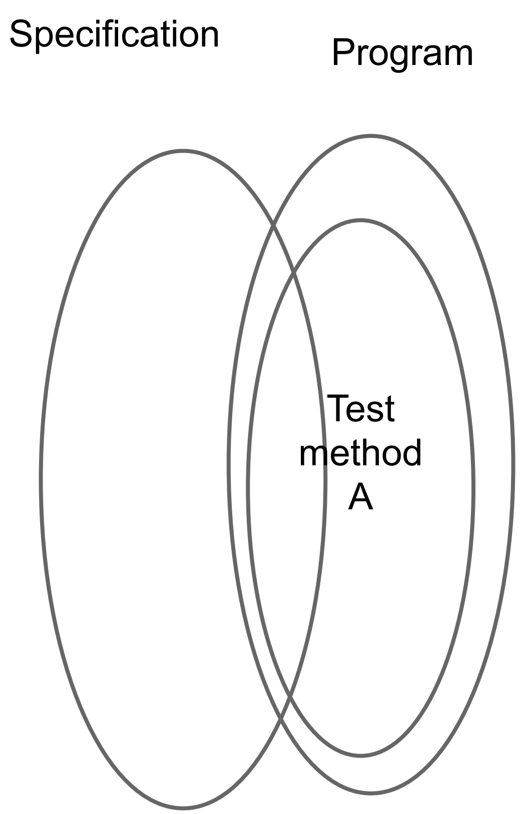
Specification-based testing

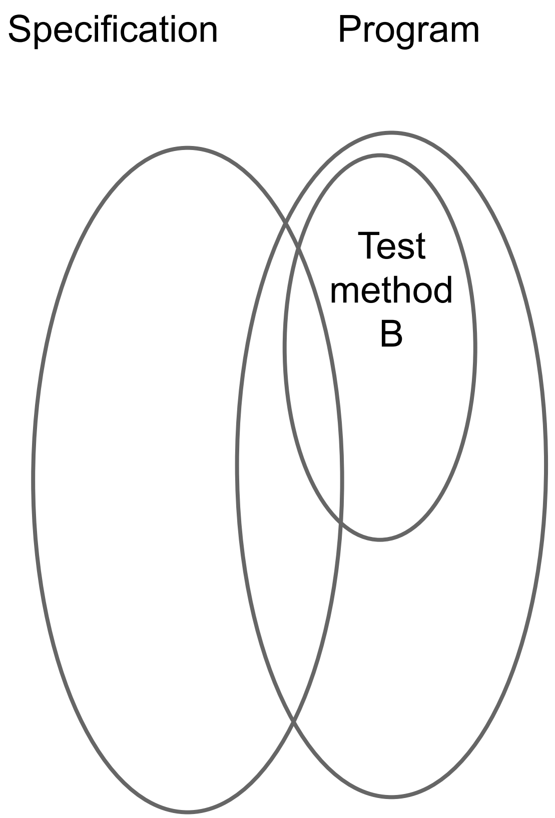
- Chap 3: mathematical background (Discrete math for testers)
- Chap 5: Boundary Value Testing
  - boundary value analysis
  - robustness testing
  - worst-case analysis
  - special value testing
- Chap 6: Equivalence Class Testing
  - input (domain) equivalence classes
  - output (range) equivalence classes
- Chap 7: Decision Table-Based Testing
  - Conditions as input
  - actions as outputs
  - rules interpreted as test cases

Code-Based Testing

- White box (clear box) tetsing
- Implementation of the black box is known and used to identify test cases
- Use of test coverage metrics
  - explicitly state the extent to which a software item has been tested
- Disadvantages
  - specified, but not implemented

Comparing code-based test



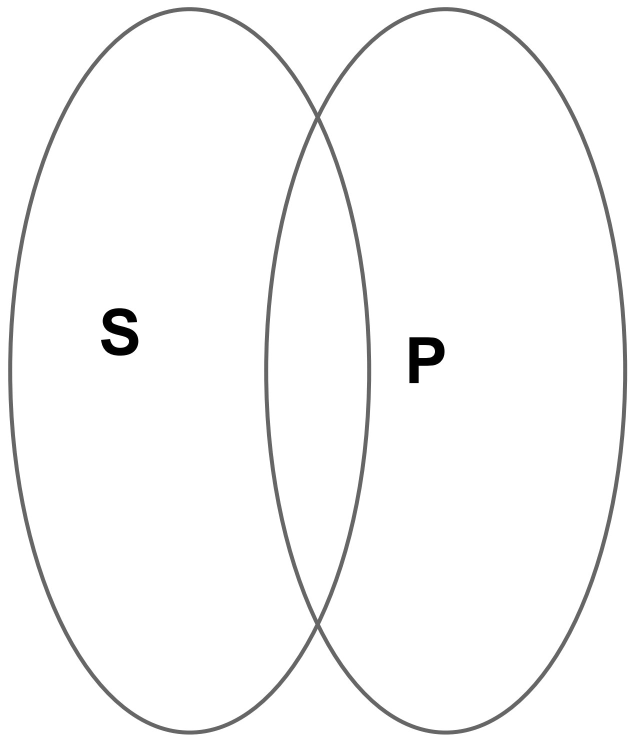


- ❏ Method A identifies larger set of test case than method B
- ❏ Both are in the set of programed behavior

Code-based Testing

- Chap 4: Graph Theories for Tester
- Chap 8: Path Testing
- Chap 9: Data Flow Testing
- Chap 10: Slice Testing

Sources of test Cases



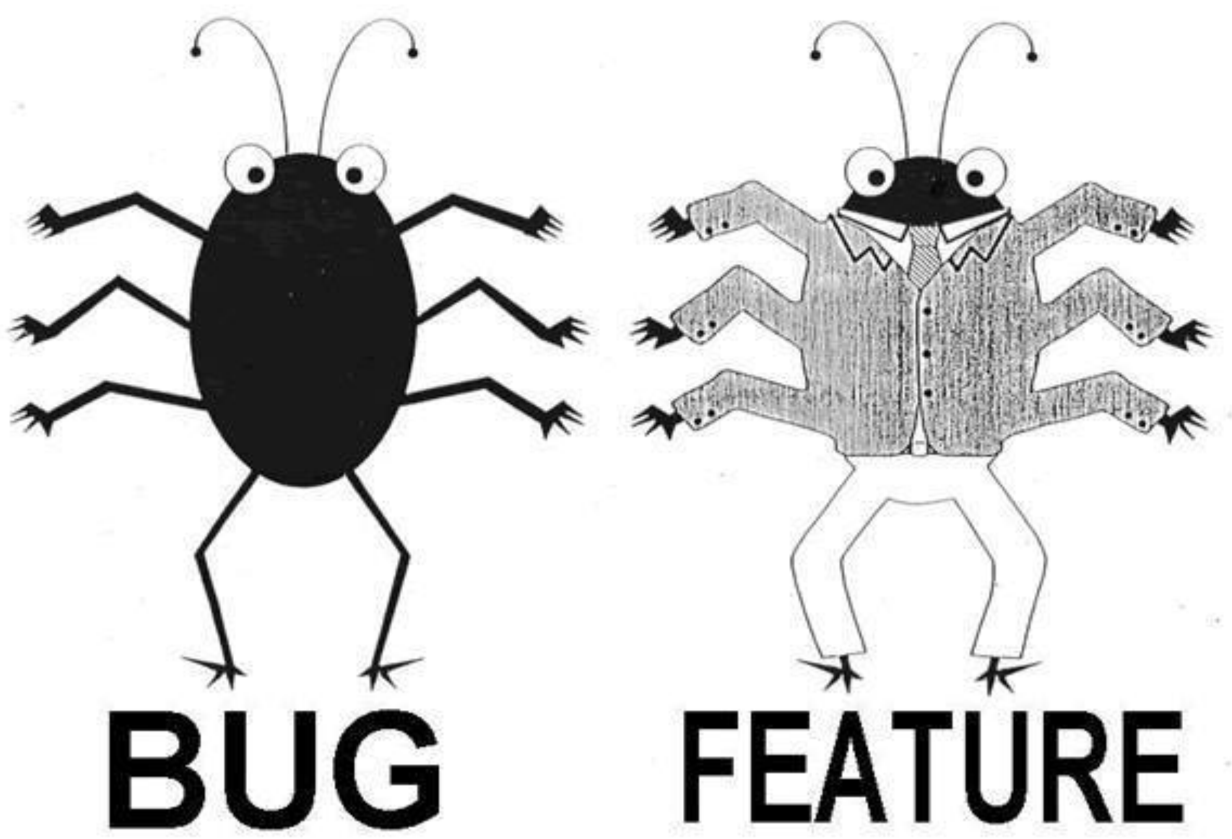
Spec-based
functional
black box
(establishes confidence)

Code-based
structural
white/clear box
(seeks faults)

Specification-Based versus Code-Based

- Program Behaviors
  - specified, but not implemented
    - code-based test cannot recognize
  - not specified, but implemented
    - trojan horse is an unspecified behavior
    - bugs
- Combination
  - confidence of specification-based testing
    - redundant and gap
  - measurement of code-based testing (coverage metrics)
    - mitigations of the above problems

Bugs vs. Features



Bugs are Features in disguise.

Relationship between T,S, and P

- Test cases in T: determined by the test case identification method
  - know the kind of errors
  - know the kind of faults
  - apply suitable test case identification methods

Fault Taxonomies

- Distinctions
  - **error** during process
    - process: how we do something
    - SQA (software quality assurance): improve the process
      - improve the process
  - **fault** of the product
    - product: the end result of a process
    - Testing: product oriented
      - discover the faults in a product

Faults classifications

- development phase (corresponding error)
- consequences of corresponding failures
- difficulty to resolve
- risk of no resolution
- anomaly occurrence
  - one time only
  - intermittent
  - recurring
  - repeatable

Types of Faults

- Software anomaly
  - departure from the expected
  - review checklists for fault classifications
    - http://www.processimpact.com/pr_goodies.shtml
- Input/Output Faults
- Logic Faults
- Computation Faults
- Interface Faults
- Data Faults

Inut/Output Faults

Type	Instances
Input	Correct input not accepted
	Incorrect input accepted
	Description wrong or missing
	Parameters wrong or missing
Output	Wrong format
	Wrong result
	Correct result at wrong time(too early, too late)
	Incomplete or missing result
	Spurious result
	Spelling/grammar
	Cosmetic

Cosmetic Faults

- Affect system in the least in terms of functionality
- Examples
  - Spelling Mistakes
  - UI Anamolies (font related variance, 12->14, sound difference)
  - Background color of specific fields

Logic and Computation Faults

Logic Faults
Missing case(s)
Duplicate case(s)
Extreme condition neglected
Misinterpretation
Missing condition
Extraneous condition(s)
Test of wrong variable
Incorrect loop iteration
Wrong operator (e.g., < instead of ≤)

Computation Faults
Incorrect algorithm
Missing computation
Incorrect operand
Incorrect operation
Parenthesis error
Insufficient precision(round-off, truncation)
Wrong built-in function

Interface and Data Faults

Interface faults

Incorrect interrupt handling
I/O timing
Call to wrong procedure
Call to nonexistent procedure
Parameter mismatch(type, number)
Incompatible types
Superfluous inclusion

Data Faults

Incorrect initialization
Incorrect strorage/access
Wrong flag/index value
Incorrect packing/unpacking
Wrong variable used
Wrong data reference
Scaling or units error

Incorrect data dimension
Incorrect subscript
Incorrect type
Incorrect data scope
Sensor data out of limits
Off by one
Inconsistent data

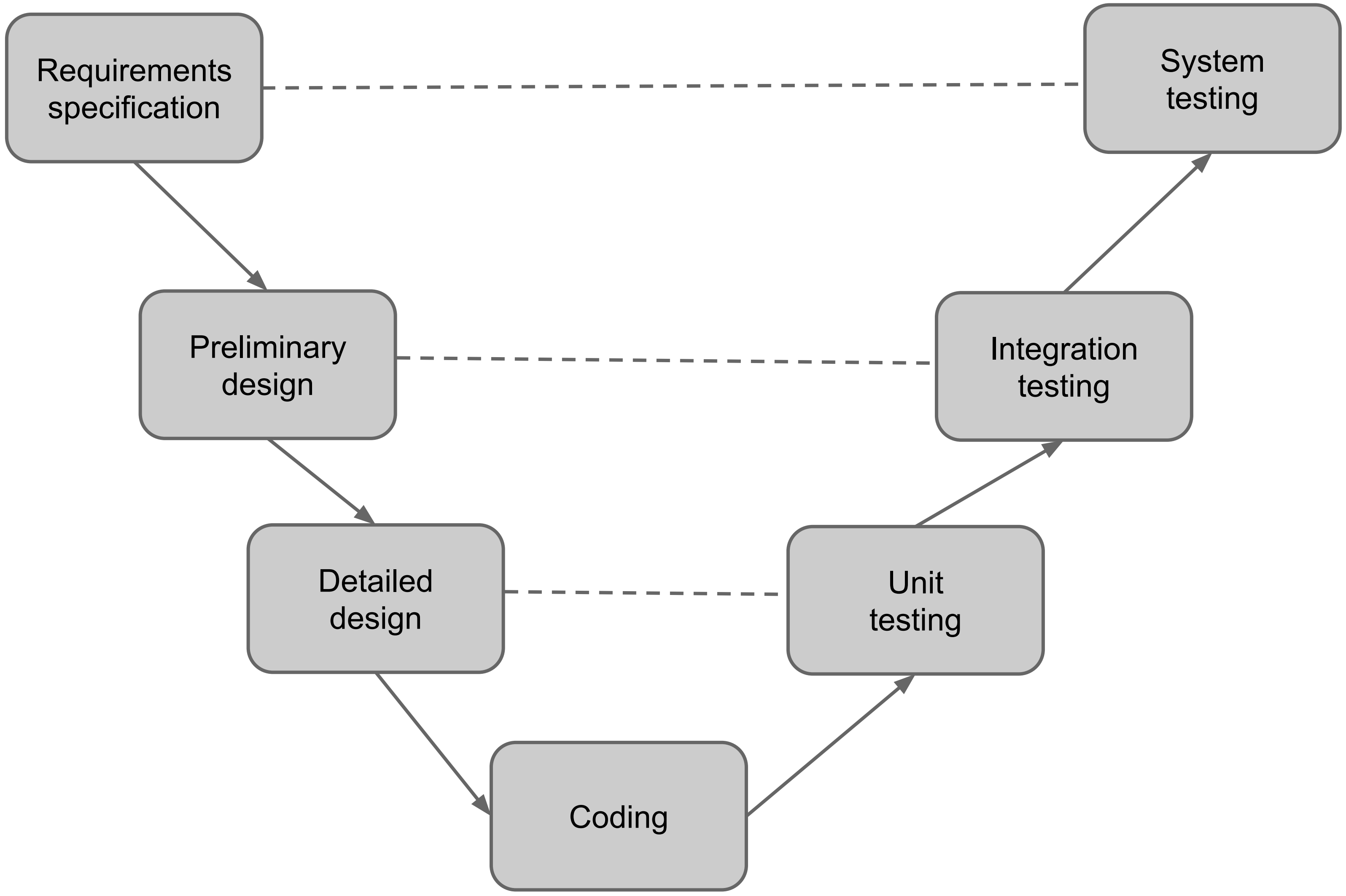
Levels of Testing

- Code-based testing
  - unit-level
  - integration and system level
- Spec-based testing
  - system-level

Relationship between Abstraction levels

- Specification => System Testing
    - specification-based testing
- Preliminary design => Integration Testing
- Detailed design => Unit Testing
    - code-based testing

Levels of abstraction and testing in waterfall model (V-Model)



Requirements specification
System testing
Preliminary design
Integration testing
Detailed design
Unit testing
Coding