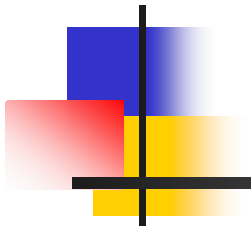
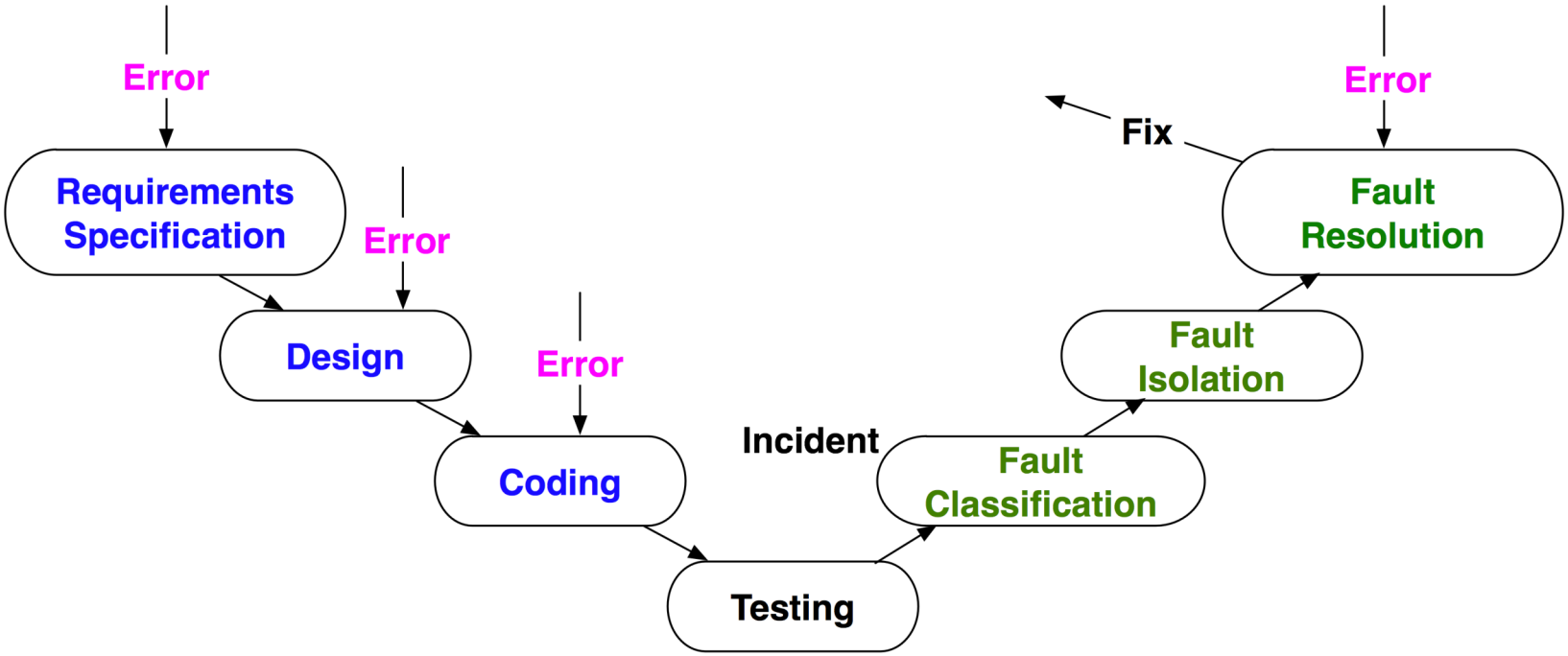
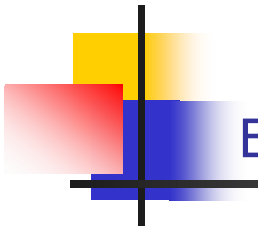


Questions From

Chapter 1

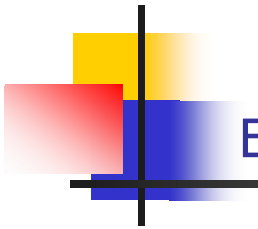
Figure 1.1: Testing life cycle





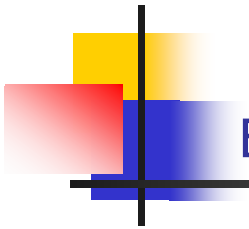
Error vocabulary – 1

- What is an **error**?



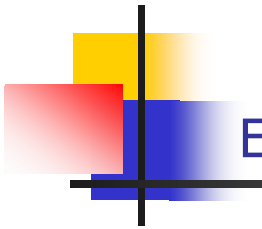
Error vocabulary – 2

- What is an **error**?
  - An **error** (or mistake) is something people make



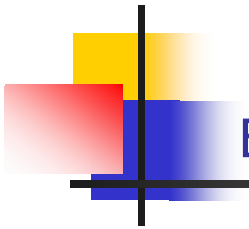
Error vocabulary – 3

- What is an **error**?
  - An **error** (or mistake) is something people make
- What types of error are there?



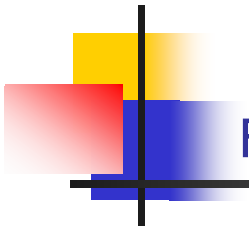
Error vocabulary – 3

- What is an **error**?
  - An **error** (or mistake) is something people make
- What types of error are there?
  - Of **commission**
  - Of **omission**
- Which kind of error is most difficult to detect?



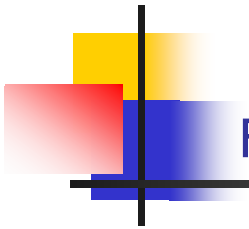
Error vocabulary – 4

- What is an **error**?
  - An **error** (or mistake) is something people make
- What types of error are there?
  - Of **commission**
  - Of **omission**
- Which kind of fault is most difficult to detect?
  - Faults of **omission** are most difficult to detect



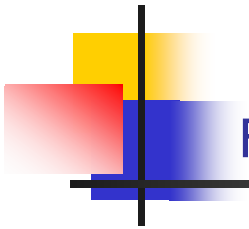
Fault vocabulary – 1

- What is a **fault**?



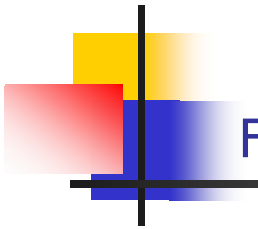
Fault vocabulary – 2

- What is a **fault**?
  - A **fault** is the result of an error: inaccurate requirements text, erroneous design, buggy source code etc.



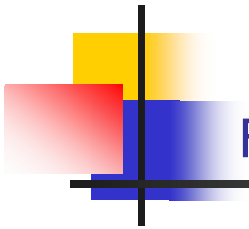
Failure vocabulary – 1

- What is a **failure**?



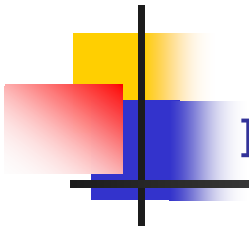
Failure vocabulary – 2

- What is a **failure**?
  - A **failure** is the program's actual incorrect or missing behavior
- When does a failure manifest itself?



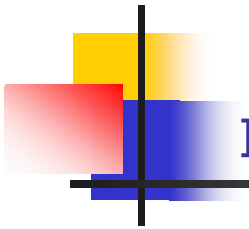
Failure vocabulary – 3

- What is a **failure**?
  - A **failure** is the program's actual incorrect or missing behavior
- When does a failure manifest itself?
  - A failure occurs when a fault executes
  - A fault won't yield a failure without the **conditions** that trigger it.
    - **Example**: if a program yields 2+2=5 on the 10th time you use it, you won't see the failure before or after the 10th use.



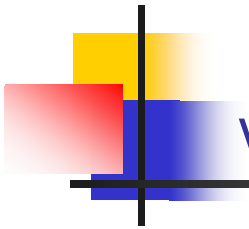
Incident vocabulary – 1

- What is an **incident**?



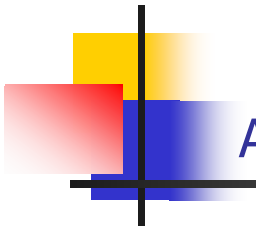
Incident vocabulary – 2

- What is an **incident**?
  - An **incident** is a characteristic of a failure that helps you recognize that the program has failed.



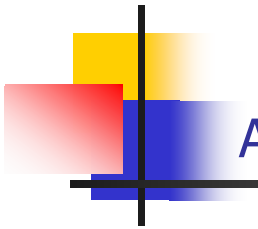
Vocabulary example

- Here's a defective program
  - INPUT A
  - INPUT B
  - PRINT A / B
- What is the error?
- What is the fault?
- What is the critical condition?
- What will we see as the incident of the failure?



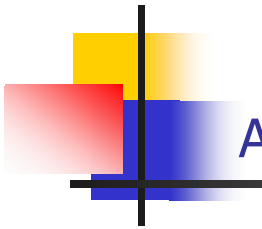
About tests – 1

- What is the purpose of a test?



About tests – 2

- What is the purpose of a test?
  - To verify correct behaviour
  - To find a failure



About test cases

- What is a test case?

- What information do we need to document a test case?

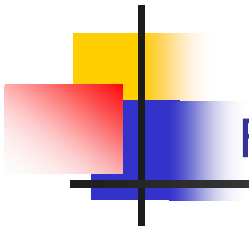


Figure 1.2: Test case information

1 Test case ID

2 Purpose

3 Preconditions

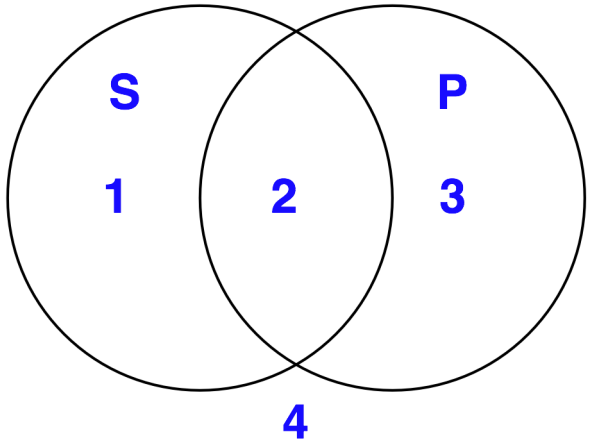
4 Expected outputs

5 Postconditions

6 Execution history

Date Result Version Run by

Figure 1.3: Specified and implemented program behaviours



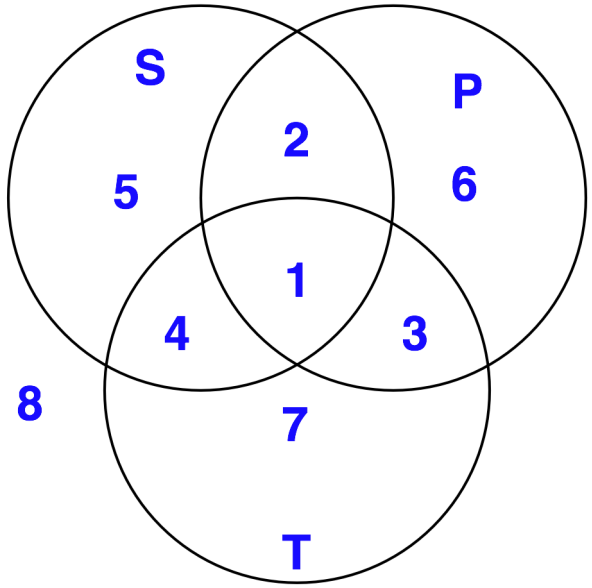
Specification
expected behaviour

Program
observed behaviour

What do the numbered areas represent?

Figure 1.4: Specified, implemented and tested behaviours

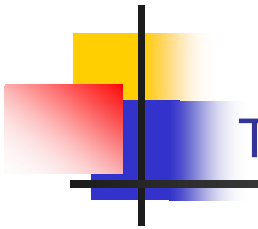
Specification
expected behaviour



Program
observed behaviour

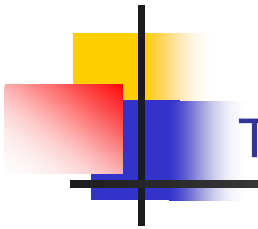
Tested cases
verified behaviour

What do the numbered areas represent?



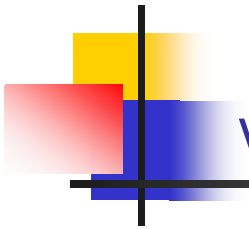
Test case difficulty – 1

- What are the difficulties in making a test case?



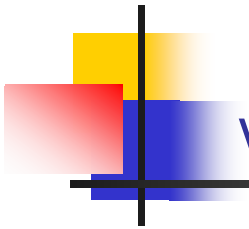
Test case difficulty – 2

- What are the difficulties in making a test case?
  - Setting up preconditions
  - Determining expected output



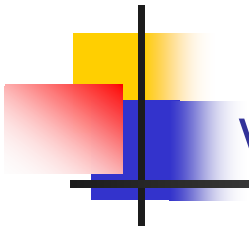
Value of test cases – 1

- Are test cases valuable?



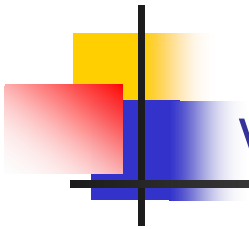
Value of test cases – 2

- Are test cases valuable?
  - Yes
- Why?



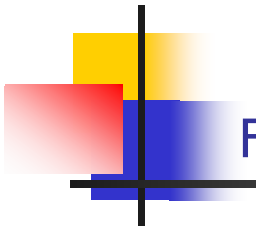
Value of test cases – 3

- Are test cases valuable?
  - Yes
- Why?
  - Difficult to construct
  - Need for verify correctness
  - Need to reuse for regression testing
  - Need to evolve
- What do we do about it?



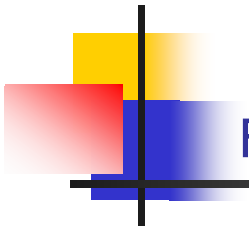
Value of test cases – 4

- Are test cases valuable?
  - Yes
- Why?
  - Difficult to construct
  - Need for verify correctness
  - Need to reuse for regression testing
  - Need to evolve
- What do we do about it?
  - Document
  - Save
  - Use again



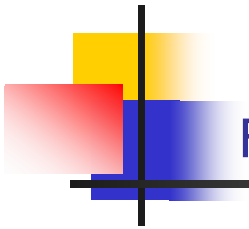
Functional testing – 1

- What are the advantages of functional testing?



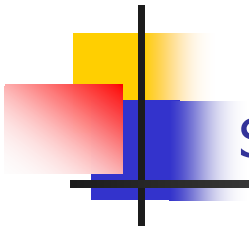
Functional testing – 2

- What are the advantages of functional testing?
  - Independent of implementation
  - Develop in parallel with program text
- What are the disadvantages of functional testing?



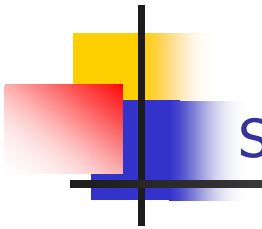
Functional testing – 3

- What are the advantages of functional testing?
  - Independent of implementation
  - Develop in parallel with program text
- What are the disadvantages of functional testing?
  - Redundant tests
  - Gaps in tests
  - Cannot develop test cases for non-specified behaviour



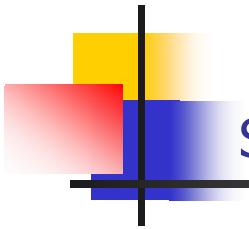
Structural testing – 1

- What are the advantages of structural testing?



Structural testing – 2

- What are the advantages of structural testing?
  - Strong theoretical basis
    - **Nothing is a practical as a good theory!**
  - Leads to good methods for discussing test coverage
  - Can look for unspecified behaviour
- What are the disadvantages of structural testing?

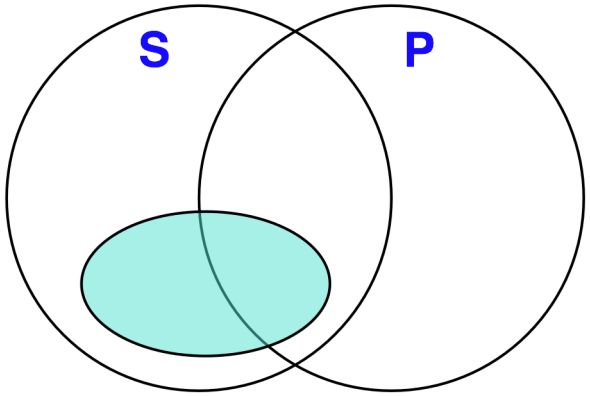


Structural testing – 3

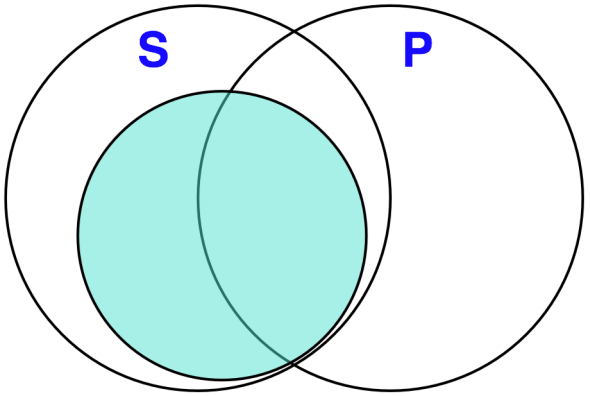
- What are the advantages of structural testing?
  - Strong theoretical basis
    - Nothing is a practical as a good theory!
  - Leads to good methods for discussing test coverage
  - Can look for unspecified behaviour

- What are the disadvantages of structural testing?
  - Cannot find test cases outside the structure of the program

Comparing functional test case identification methods



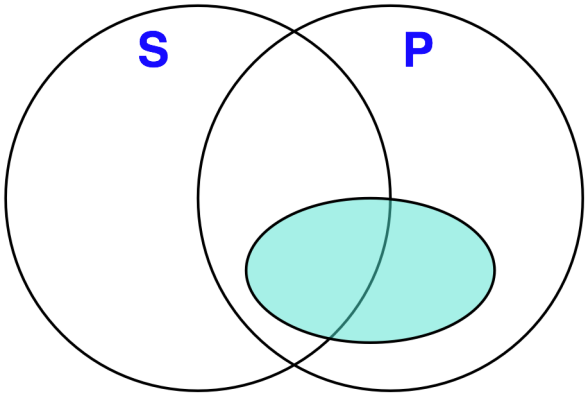
Tested cases
Method A



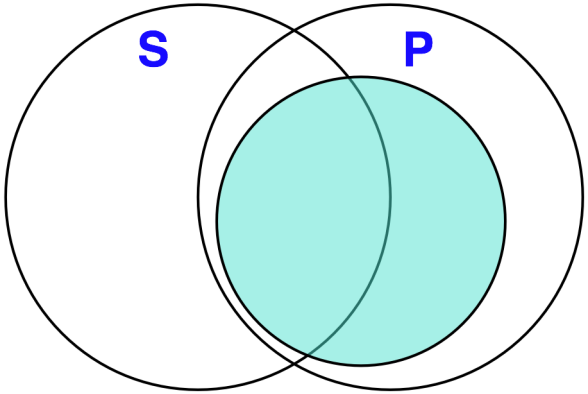
Tested cases
Method B

What do the diagrams represent?

Comparing structural test case identification methods

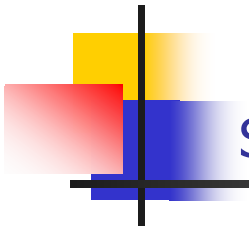


Tested cases
Method A



Tested cases
Method B

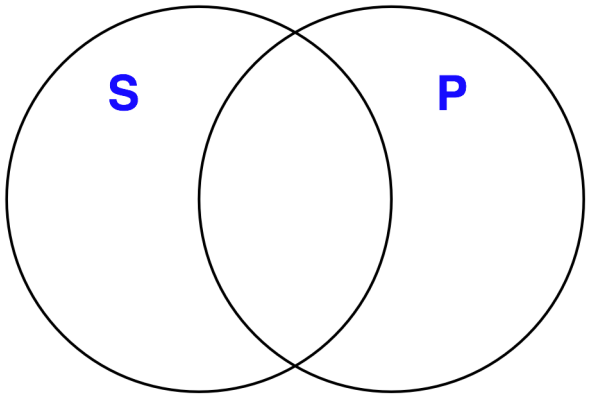
What do the diagrams represent?



Sources of test cases – 1

- Which method functional or structural testing is better?
- Why?

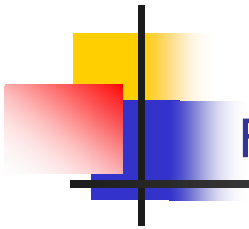
Sources of test cases – 2



Functional
black box
Establishes Confidence

Structural
white box
Seeks Faults

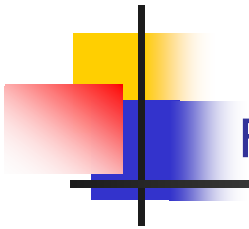
What conclusion can be made?



Faults classified by severity

1	Mild
2	Moderate
3	Annoying
4	Disturbing
5	Serious
6	Very serious
7	Extreme
8	Intolerable
9	Catastrophic
10	Infectious

Of what use is the classification?



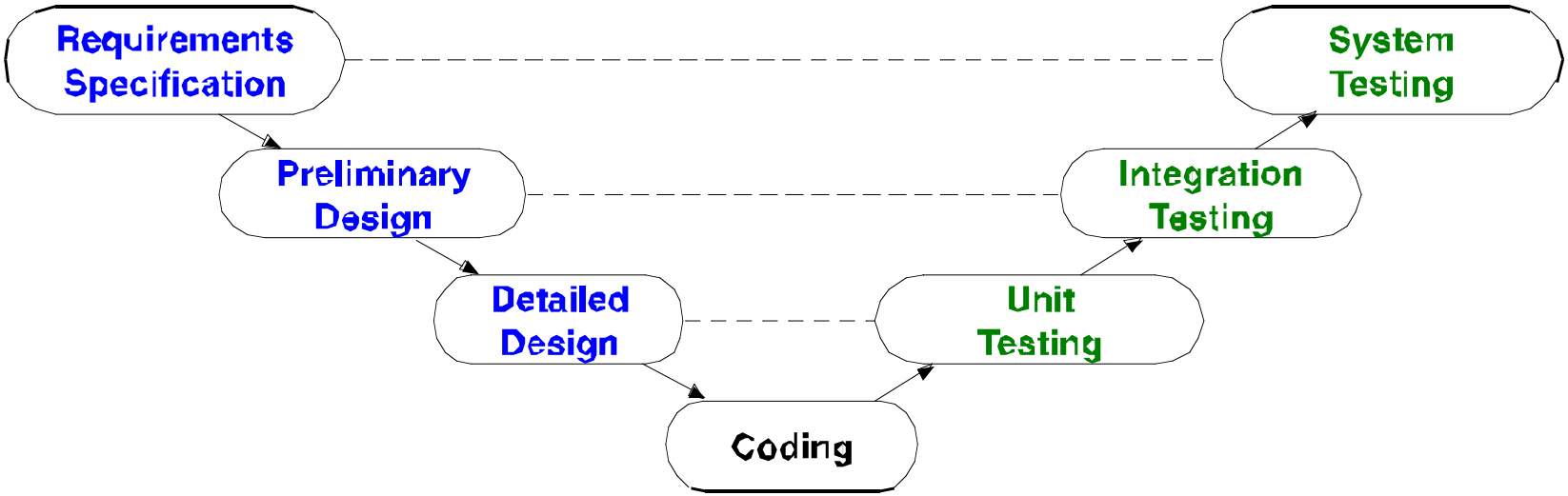
Fault taxonomy

1. Input/output faults
2. Logic faults
3. Computation faults
4. Interface faults
5. Data faults

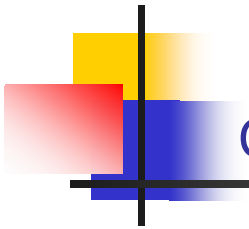
What are typical faults in each type?

Of what use is the taxonomy?

Figure 1.10: Levels of abstraction and testing

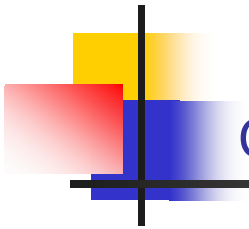


Of what use is this diagram?



Craft of testing – 1

- In conclusion
  What is the craft of testing?



Craft of testing – 2

- In conclusion
  What is the craft of testing?
  - Identify errors we are likely to make
  - Create test cases to find the corresponding faults