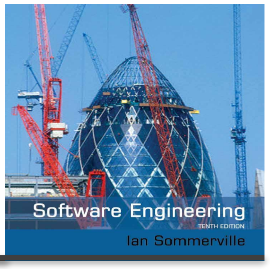
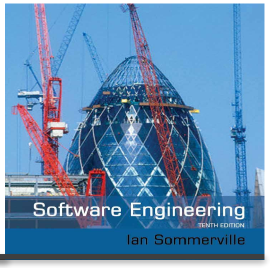


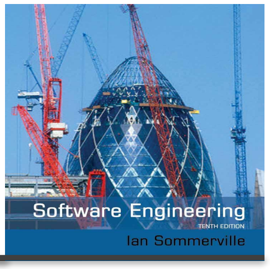
Software Testing

Program testing



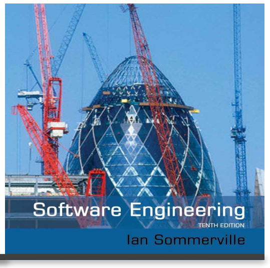
- Testing:
  - Showing what a program is doing.
  - Detecting anomalies in the system.
- Generally, software testing uses sample data
- **Custom software:** at least 1 test for every requirement.
- **Generic Software:** tests for every feature present in product release
- **Testing** is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.

Program testing



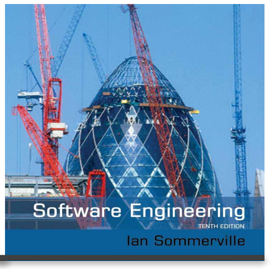
- Testing can be done manually or automatically (via automated test cases, e.g., Junit in Java).
- Testing is a broader part of Software verification & validation
  - Are we building the right product? (validation)
  - Are we building the product right? (verification) ~ checking for functional and non-functional testing
- Broadly, testing can be classified as black-box testing / white box testing.
  - Black box testing: testing system without checking inner details
  - Whitebox testing: testing each component separately.

Program testing



- Testing could be done to check the functional or nonfunctional requirements both named functional testing & non-functional testing, respectively.
  - Functional testing: checking for each functionality in the code
  - Non-functional testing: checking for performance, scalability, system recovery, etc.

Validation and defect testing

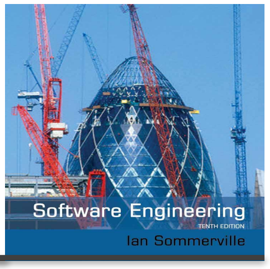


- Testing goals:
  - Confirms to developer and customers that system is up to user requirements
  - Find bugs or anomalies in the system.

- **The first goal leads to validation testing**

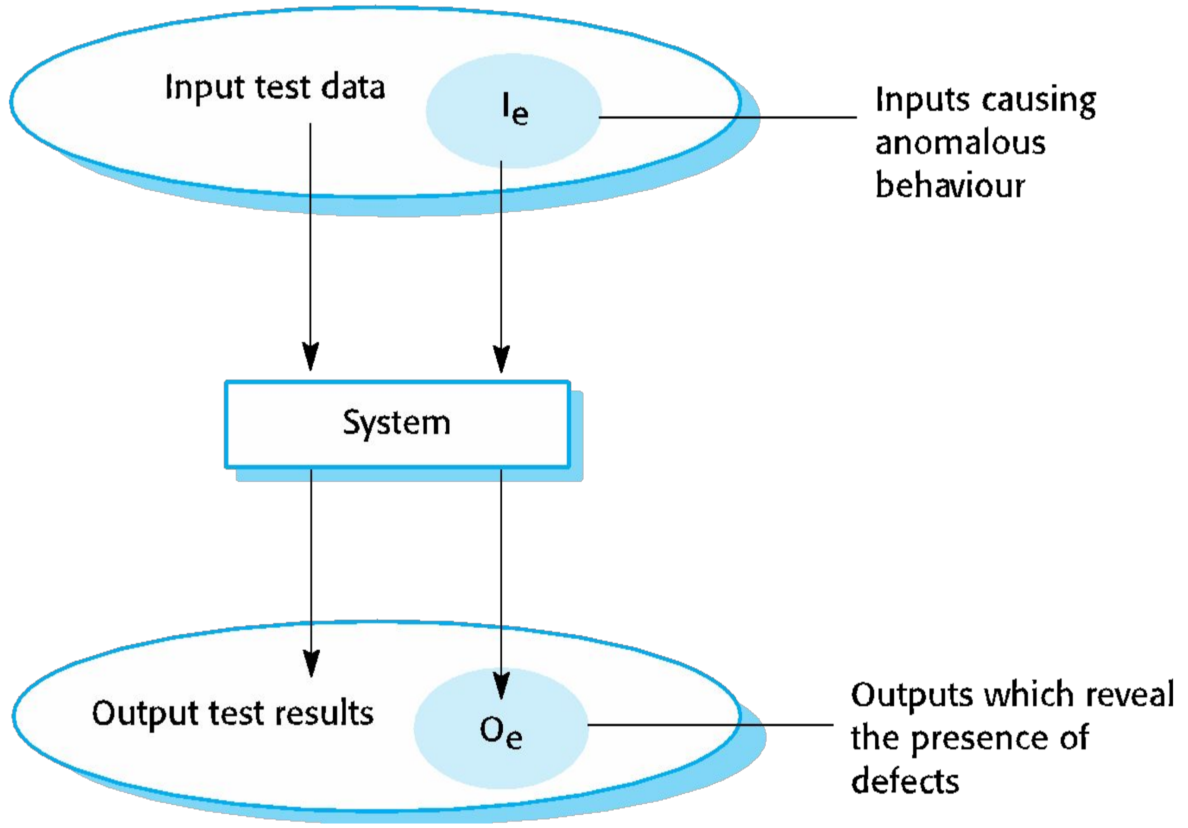
- **The second goal leads to defect testing**

Testing process goals

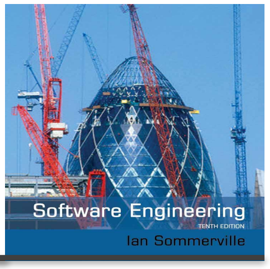


- **Validation testing**
  - To demonstrate to the developer and the system customer that the software meets its requirements
  - A successful test shows that the system operates as intended.
- **Defect testing**
  - To discover faults or defects in the software where its behavior is incorrect or not in conformance with its specification
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.
- No hard boundary is defined between these two testing techniques.

An input-output model of program testing validation & defect both

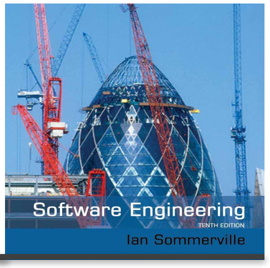

Input test data
I_e
Inputs causing anomalous behaviour
System
Output test results
O_e
Outputs which reveal the presence of defects

V & V confidence



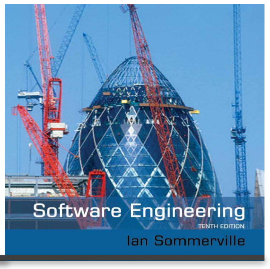
- Aim of V & V is to claim that the system is ‘fit for purpose.
- Depends on:
  - **Software purpose**
    - The criticality of the software to the stakeholders.
    - e.g., file alarm system vs an LMS
  - **User expectations**
    - Users may have low expectations for newly installed systems but they expect them to be reliable depending upon the software deployment age.
  - **Marketing environment**
    - Getting a product to market early may be more important than finding defects in the program.

Inspections and testing



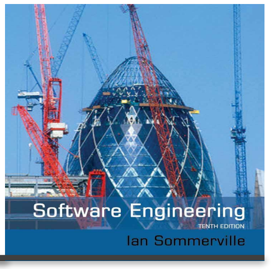
- Software inspections
  - Concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplemented by tool-based document and code analysis.
- Software testing
  - Concerned with exercising and observing product behaviour (dynamic verification)
  - The system is executed with test data and its operational behaviour is observed.

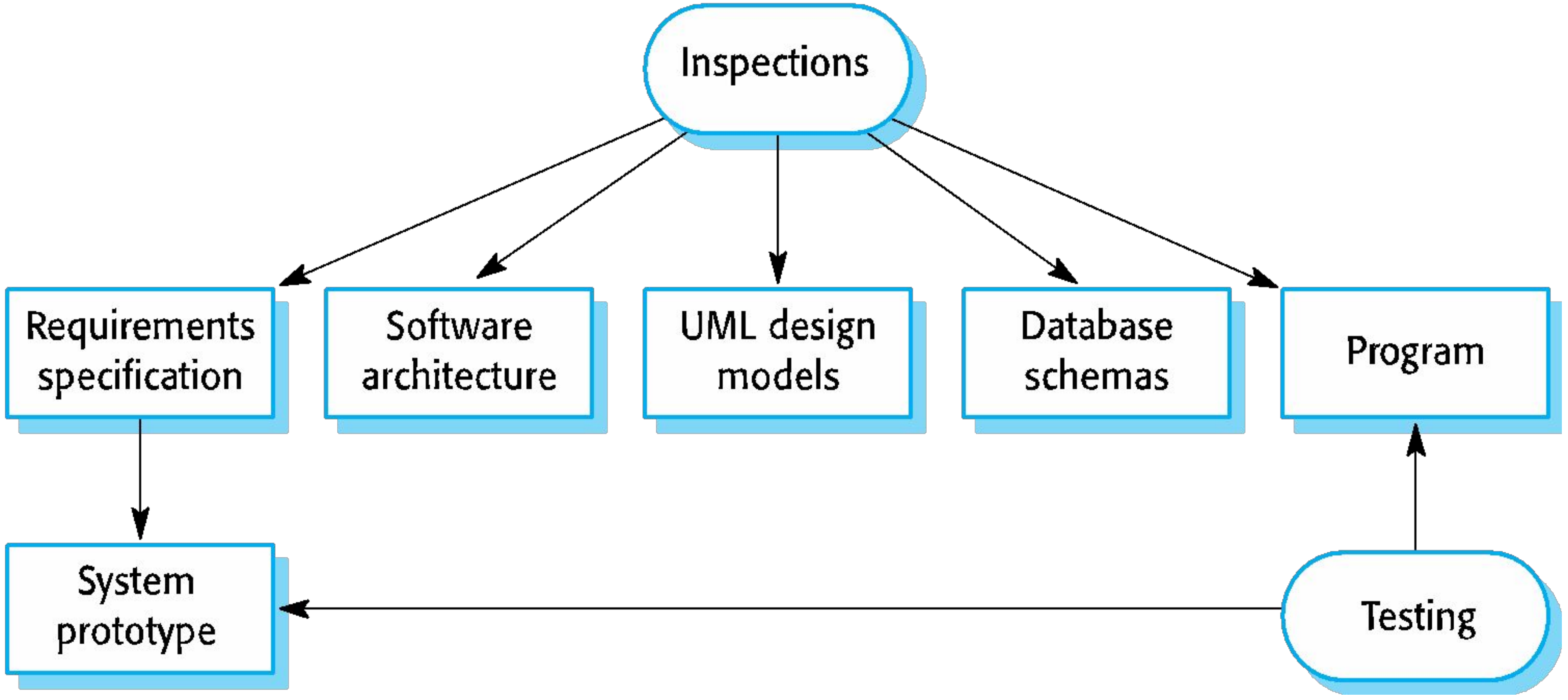
Software inspections



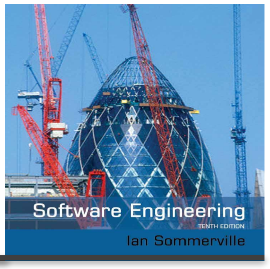
- Inspections:
  - people examining the representations with the aim of discovering anomalies and defects.
- Inspections do not require execution of a system so may be used before implementation. (static V& V technique)
- Can be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Inspections and testing



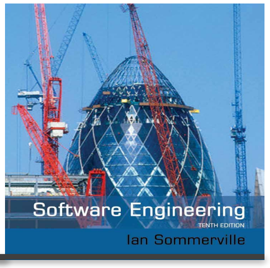


Advantages of inspections



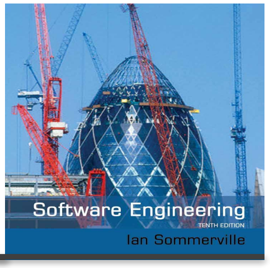
- During testing, errors can mask (hide) other errors. In case of any error occurrence, we cannot predict that the anomalies occurring after first error is because of previous one or the new errors. Because inspection is a static process, you don't have to be concerned with interactions between errors.
- Incomplete versions of a system can be inspected without additional costs. **If a program is incomplete**, **then you need to develop specialized test harnesses to test the parts that are available.**

Advantages of inspections



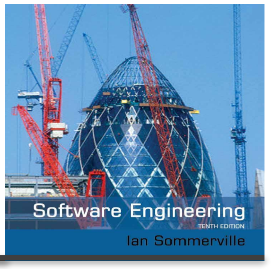
- An inspection can also **ensure the compliance with standards, portability and maintainability.** You can **check for inefficient codes, poor programming styles that could make system maintenance difficult.**
- **Inspections are more effective for defect discovery than testing.**
- **Problems in inspection:**
  - **Can't check defects that arise because of unexpected interaction between systems.**

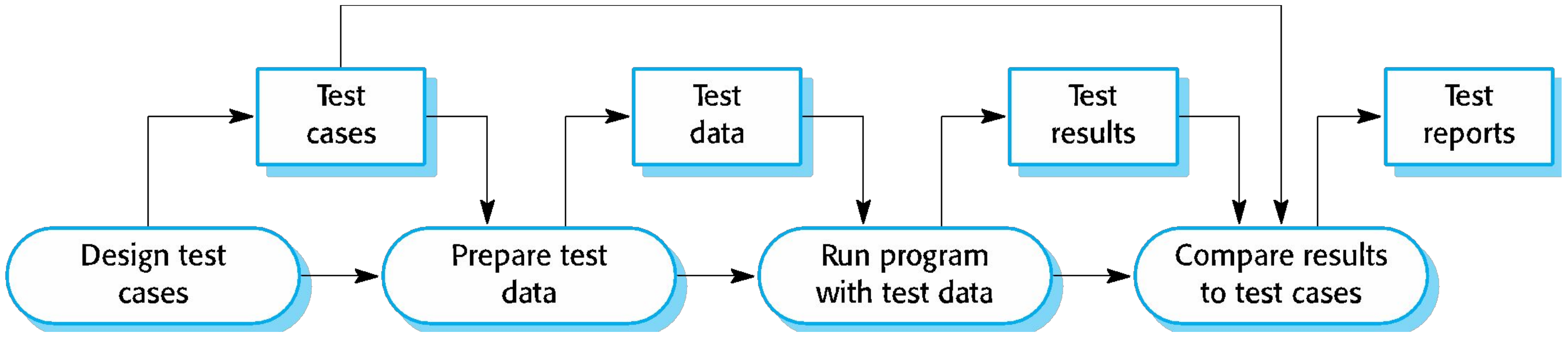
Inspections and testing

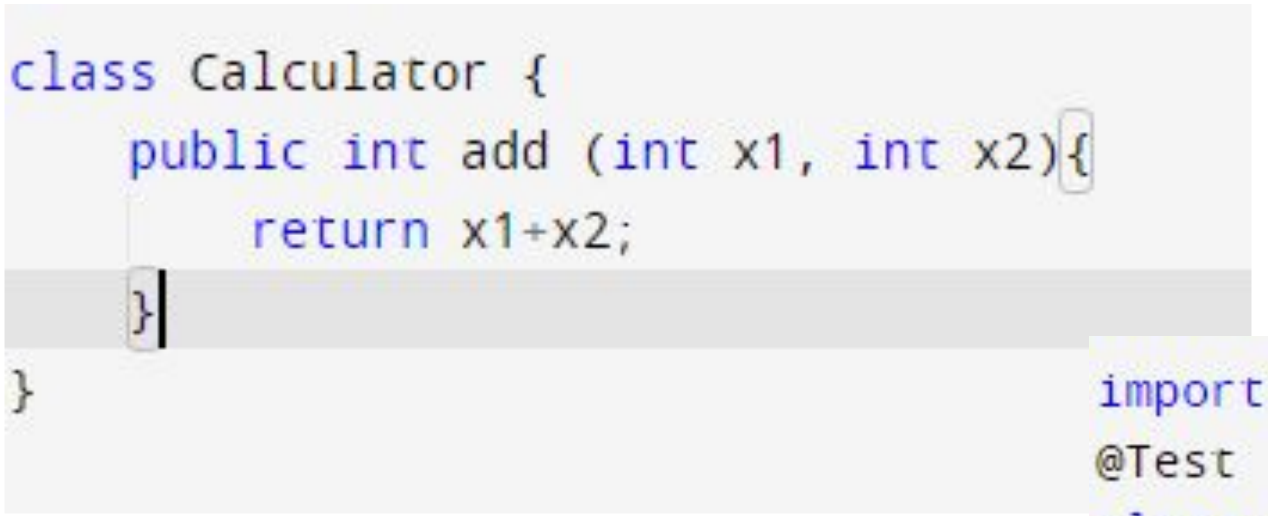


- Inspections and testing are complementary and not opposing verification techniques.
- Both should be used during the V & V process.
- **Inspections cannot check non-functional characteristics such as performance, usability, etc. as its static.**

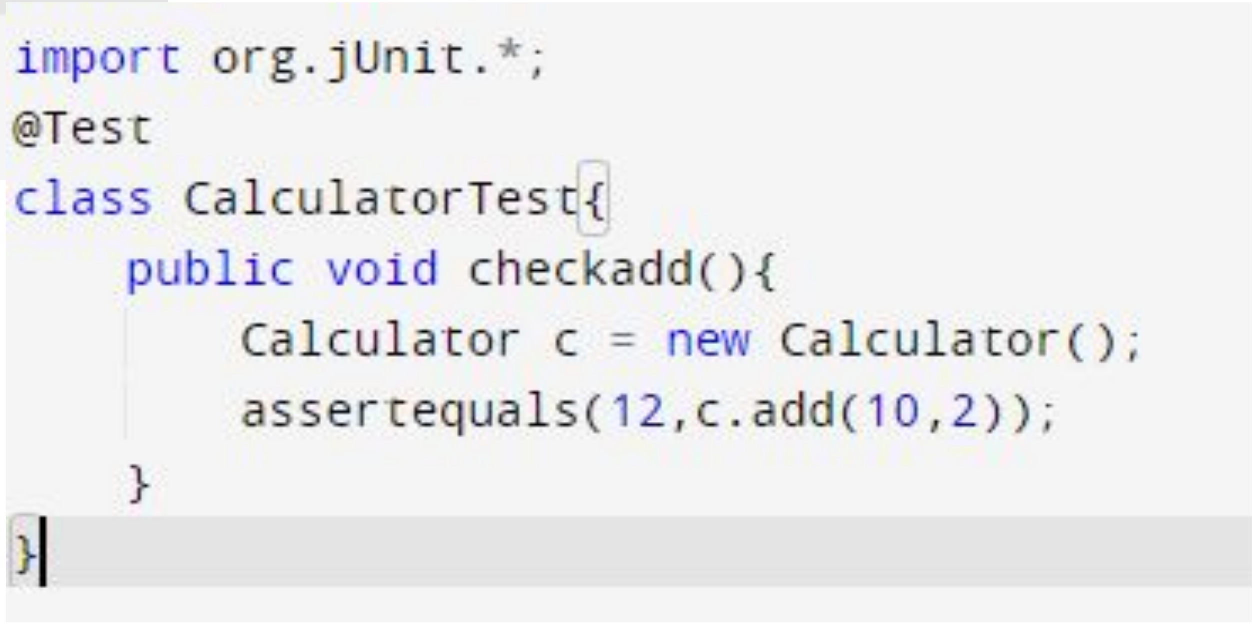
A model of the software testing process



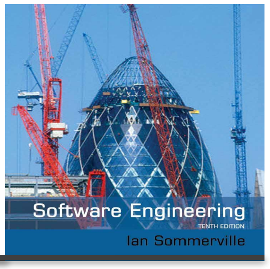




```
class Calculator {
    public int add (int x1, int x2){
        return x1+x2;
    }
}
```



```
import org.jUnit.*;
@Test
class CalculatorTest{
    public void checkadd(){
        Calculator c = new Calculator();
        assertequals(12,c.add(10,2));
    }
}
```



Stages of testing

- Development testing:
  - where the system is tested during development to discover bugs and defects.
  - Done by developers
- Release testing:
  - where a separate testing team tests a complete version of the system before it is released to users.
  - Done by QAs
- User testing:
  - where users or potential users of a system test the system in their own environment.