SOFTWARE ENGINEERING

CO3001

CHAPTER 9 – SOFTWARE QUALITY & QUALITY ASSURANCE

Anh Nguyen-Duc
Quan Thanh Tho

WEEK 9

Adapted from https://iansommerville.com/software-engineering-book/slides/

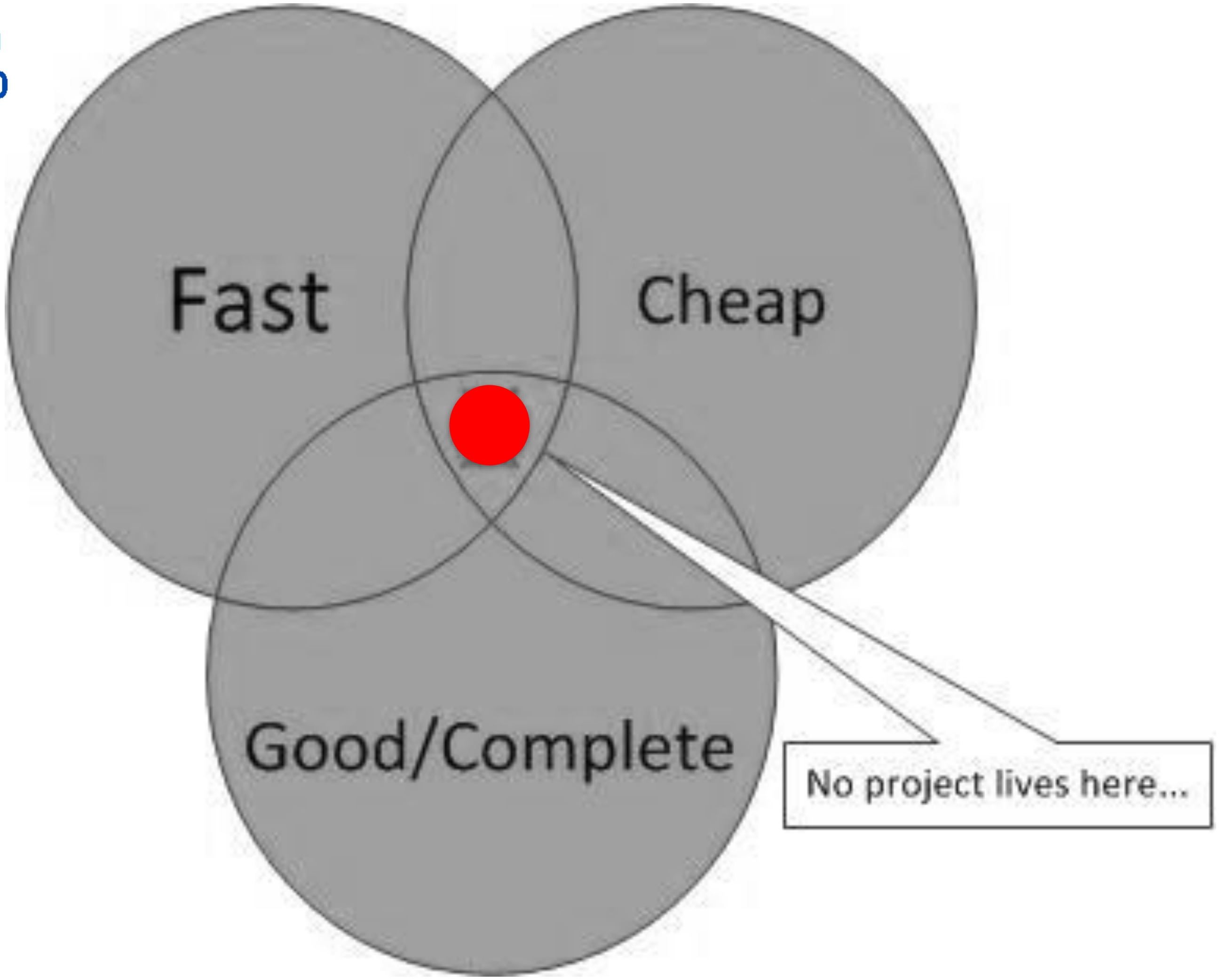
TOPICS COVERED

- ✓ Software Quality & its importance
- ✓ Development testing
- ✓ Test-driven development
- ✓ Release testing
- ✓ User testing

THE IMPORTANCE OF SOFTWARE QUALITY

WHY IS SOFTWARE QUALITY IMPORTANT?

DIMECC Program
NEED FOR SPEED



DEFINITION OF QUALITY

- ✓ (ISO) defines **quality** as the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs (ISO8042:1994) or the degree to which a set of inherent characteristics fulfils requirements. (ISO9000:2000).
- ✓ **Conformance to requirements** means the project s processes and products meet written specifications.
- ✓ **Fitness for use** means a product can be used as it was intended.
- ✓ ***Quality aspects:***
  - *product*: delivered to the customer
  - *process:* produces the software product
  - *resources:* (both the product and the process require

PROCESS QUALITY VS. PRODUCT QUALITY

- ✓ Quality can mean the difference between excellence and disaster
  - Airbus A400M Atlas crash in 2015, 4 killed

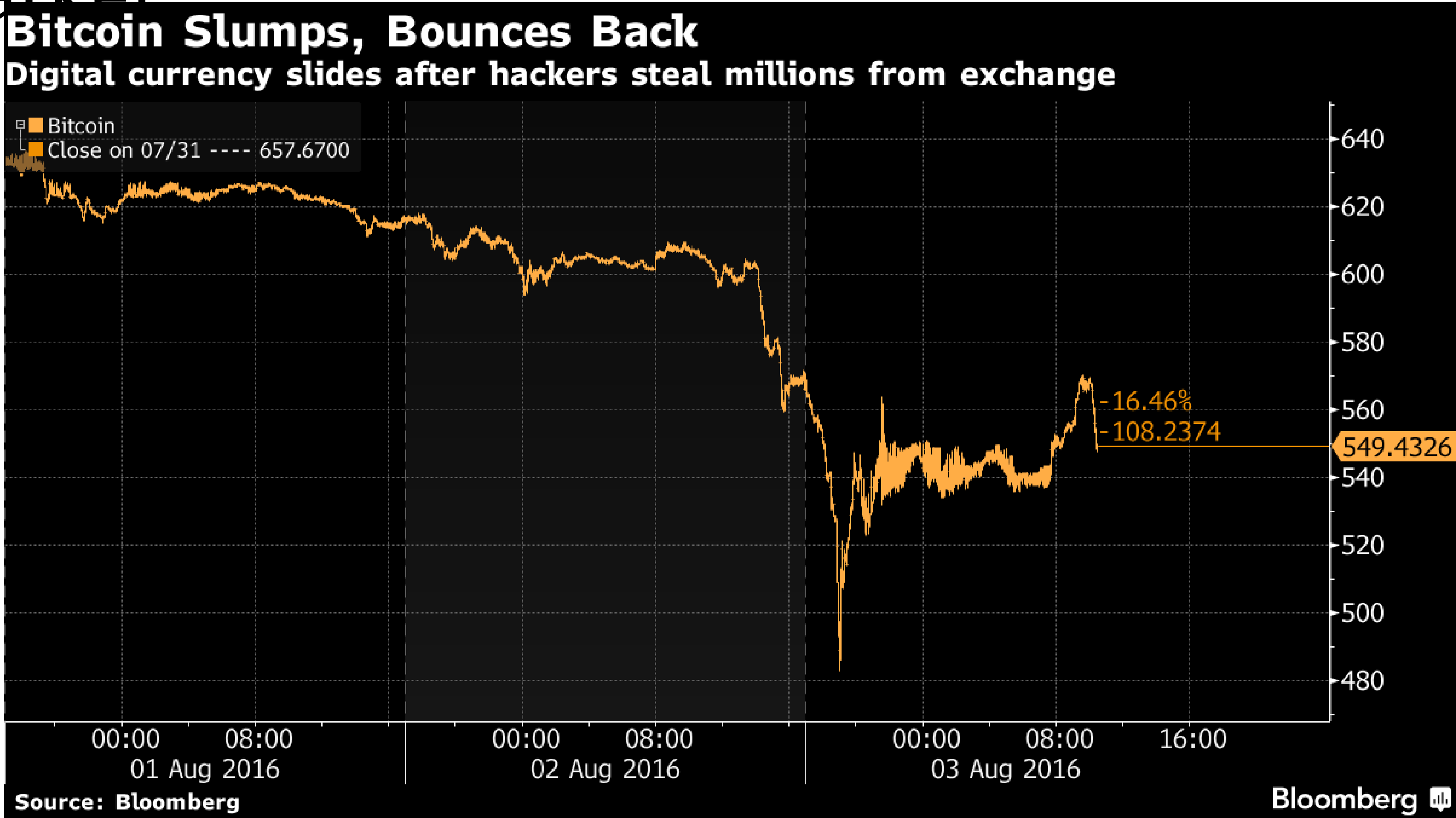
PROCESS QUALITY VS. PRODUCT QUALITY

*“The black boxes attest to that there are no structural defects [with the aircraft], but we have a serious **quality** problem in the final assembly.”*

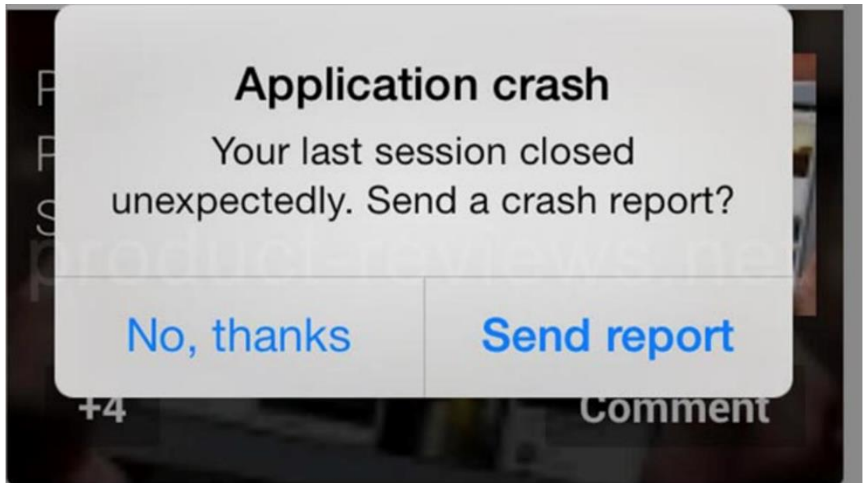
*“…either a weakness in the **test procedure** of planes before they fly, or a problem that results from the implementation of these procedures.”*

PRODUCT OR PROCESS ISSUE?

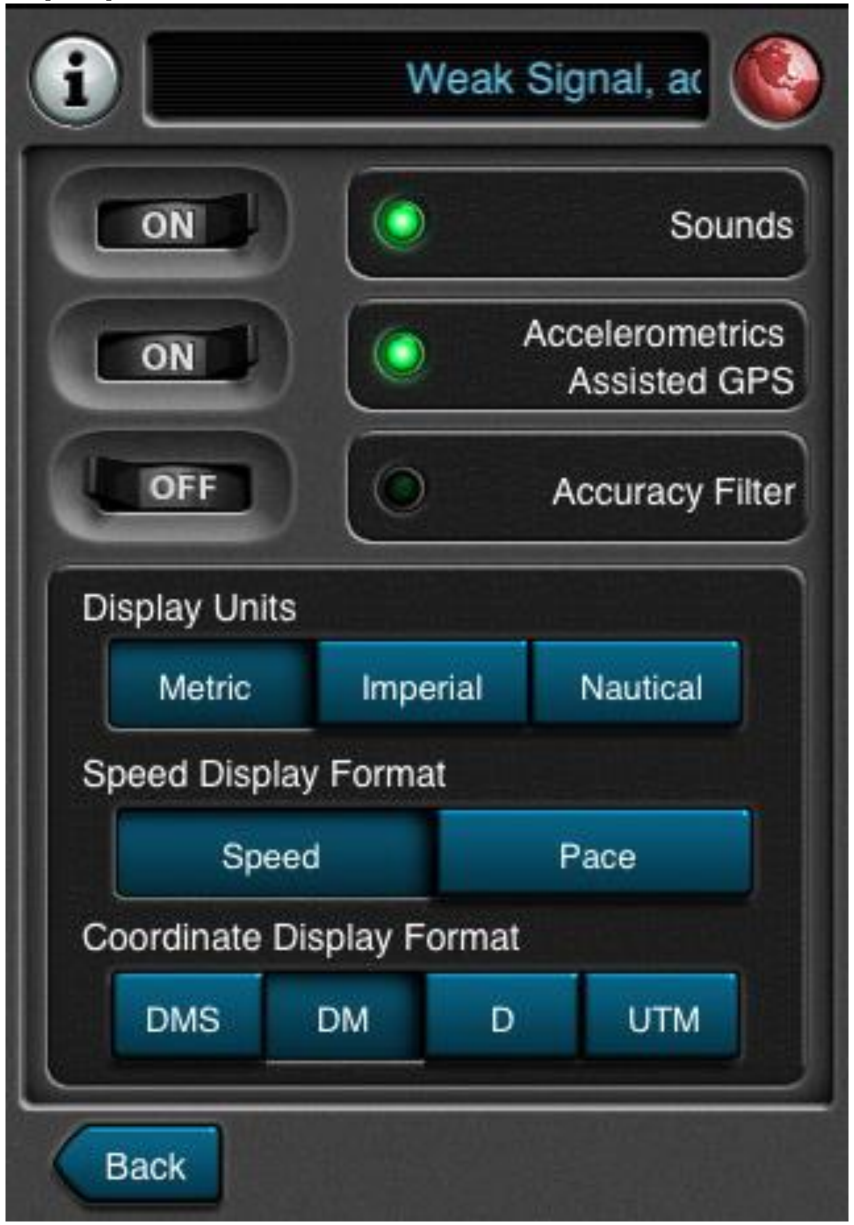
- ✓ 8/2016: Security breach with Bitcoin cost 72 mil. Usd lost in market



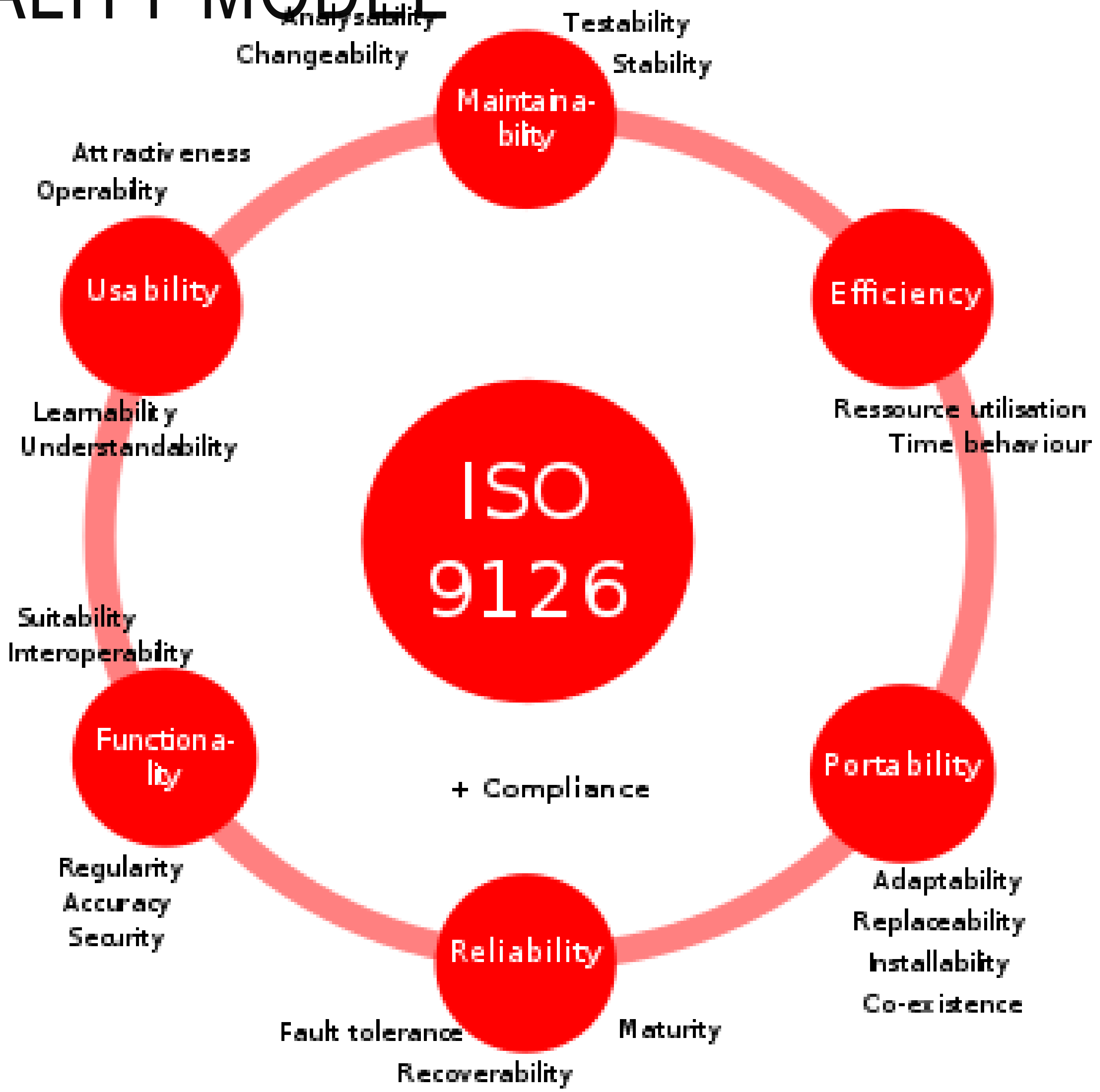
SOFTWARE QUALITY ATTRIBUTES (1)


Application crash

Your last session closed unexpectedly. Send a crash report?



SOFTWARE QUALITY MODEL



PROGRAM TESTING

- ✓ Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
- ✓ **Can reveal the presence of errors NOT their absence.**
- ✓ Testing is part of a more general verification and validation process, which also includes static validation techniques.

PROGRAM TESTING GOALS

- ✓ To demonstrate to the developer and the customer that the software meets its requirements.
  - validation testing
- ✓ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.
  - defect testing

QUALITY ASSURANCE

The Product of Testing is CONFIDENCE

PSYCHOLOGY OF TESTING (1)

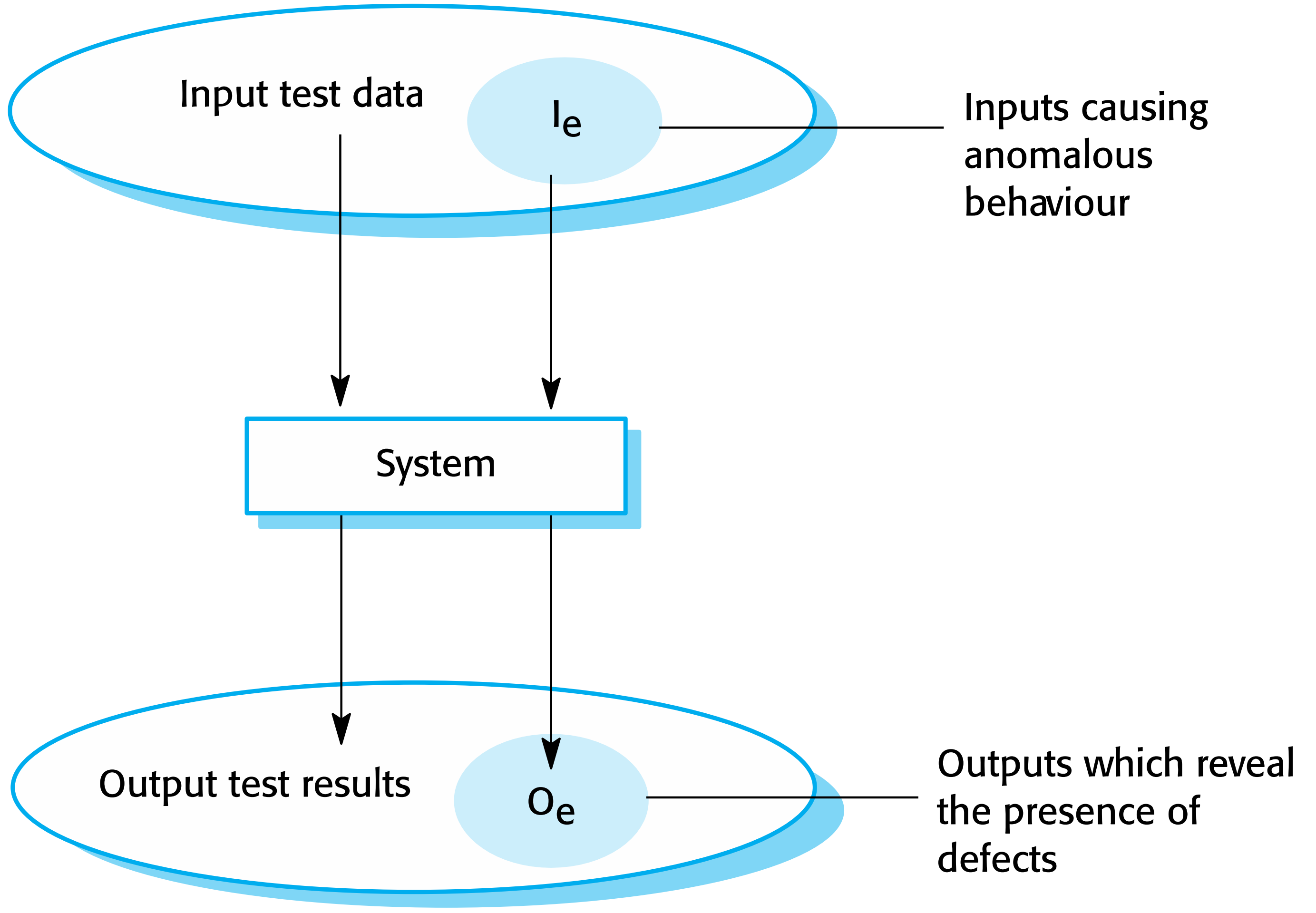
- ✓ A program is its programmer's baby!
  - Trying to find errors in one's own program is like trying to find defects in one's own baby.
  - It is best to have someone other than the programmer doing the testing.
- ✓ Tester must be highly skilled, experienced professional.
- ✓ It helps if he or she possesses a diabolical mind.

PSYCHOLOGY OF TESTING (2)

- ✓ Testing achievements depend a lot on what are the goals.
- ✓ Myers says (79):
  - If your goal is to **show absence of errors**, you will not discover many.
  - If you are trying to **show the program correct**, your subconscious will manufacture safe test cases.
  - If your goal is to **show presence of errors**, you will discover large percentage of them.

Testing is the process of executing a program with the intention of finding errors (G. Myers)

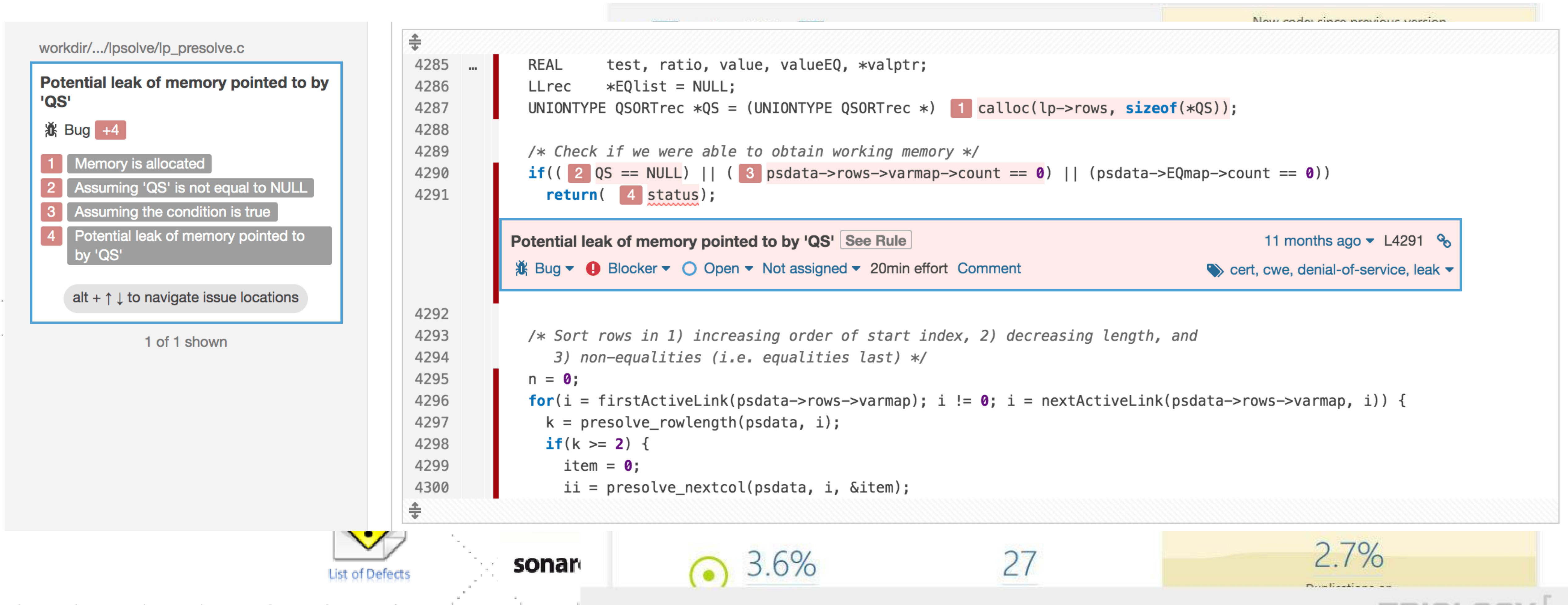
AN INPUT-OUTPUT MODEL OF PROGRAM TESTING


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

INSPECTIONS VS. TESTING

- ✓ Software inspections
  - Concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplement 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.

STATIC TESTING VS. DYNAMIC TESTING



workdir/.../lpsolve/lp_presolve.c

Potential leak of memory pointed to by 'QS'

Bug +4

1 Memory is allocated
2 Assuming 'QS' is not equal to NULL
3 Assuming the condition is true
4 Potential leak of memory pointed to by 'QS'

alt + ↑ ↓ to navigate issue locations

1 of 1 shown

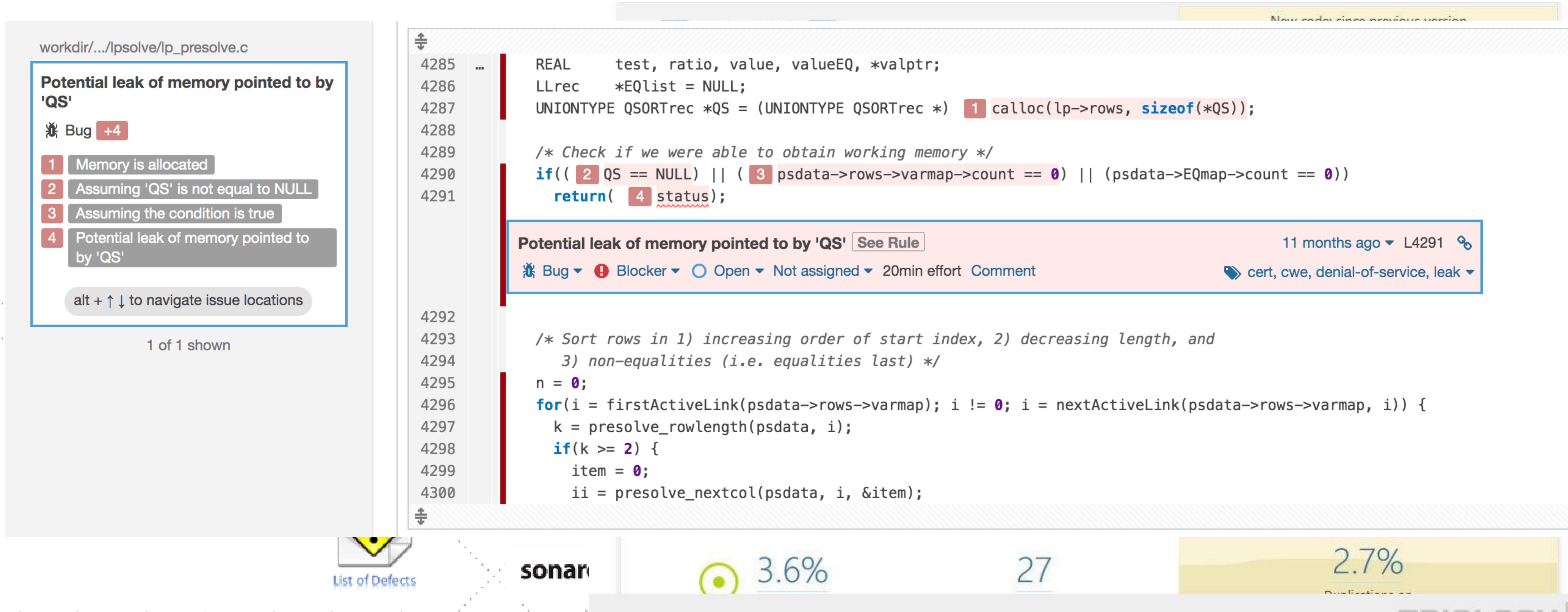
```
4285    REAL      test, ratio, value, valueEQ, *valptr;
4286    LLrec     *EQlist = NULL;
4287    UNIONTYPE QSORTrec *QS = (UNIONTYPE QSORTrec *) 1 calloc(lp->rows, sizeof(*QS));
4288
4289    /* Check if we were able to obtain working memory */
4290    if(( 2 QS == NULL) || ( 3 psdata->rows->varmap->count == 0) || (psdata->EQmap->count == 0))
4291      return( 4 status);
```

Potential leak of memory pointed to by 'QS' See Rule — 11 months ago L4291

Bug · Blocker · Open · Not assigned · 20min effort · Comment — cert, cwe, denial-of-service, leak

```
4292
4293    /* Sort rows in 1) increasing order of start index, 2) decreasing length, and
4294       3) non-equalities (i.e. equalities last) */
4295    n = 0;
4296    for(i = firstActiveLink(psdata->rows->varmap); i != 0; i = nextActiveLink(psdata->rows->varmap, i)) {
4297      k = presolve_rowlength(psdata, i);
4298      if(k >= 2) {
4299        item = 0;
4300        ii = presolve_nextcol(psdata, i, &item);
```

STATIC TESTING VS. DYNAMIC TESTING



workdir/.../lpsolve/lp_presolve.c

Potential leak of memory pointed to by 'QS'

Bug +4

1 Memory is allocated
2 Assuming 'QS' is not equal to NULL
3 Assuming the condition is true
4 Potential leak of memory pointed to by 'QS'

alt + ↑ ↓ to navigate issue locations

1 of 1 shown

```
  REAL      test, ratio, value, valueEQ, *valptr;
  LLrec     *EQlist = NULL;
  UNIONTYPE QSORTrec *QS = (UNIONTYPE QSORTrec *) 1 calloc(lp->rows, sizeof(*QS));

  /* Check if we were able to obtain working memory */
  if(( 2 QS == NULL) || ( 3 psdata->rows->varmap->count == 0) || (psdata->EQmap->count == 0))
    return( 4 status);
```

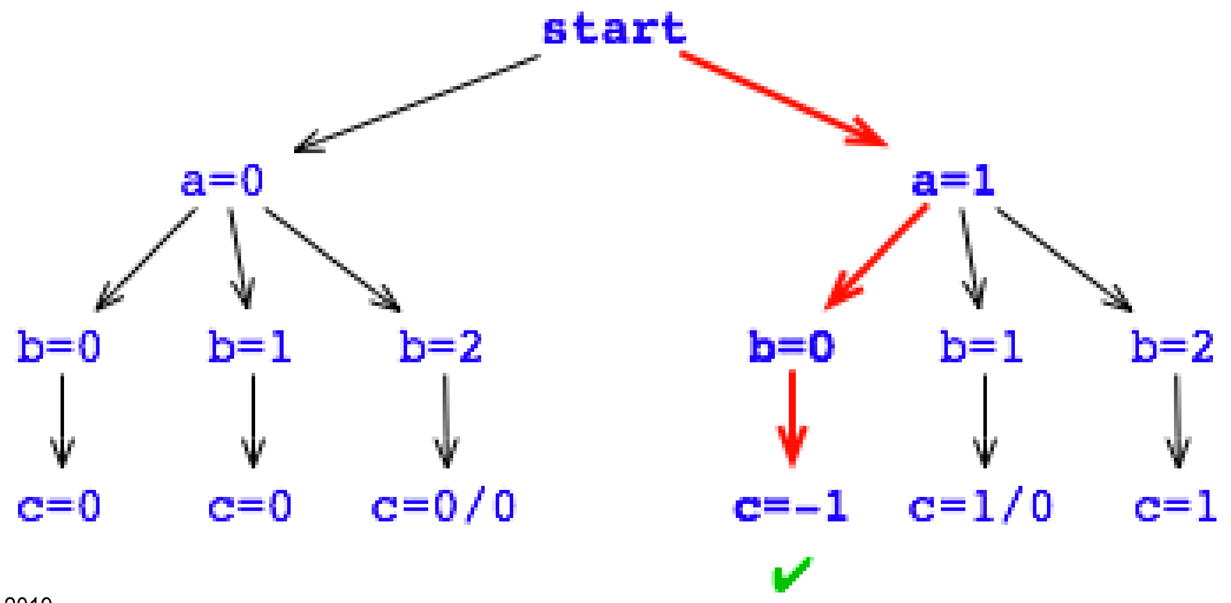
Potential leak of memory pointed to by 'QS' See Rule — 11 months ago L4291

Bug · Blocker · Open · Not assigned · 20min effort · Comment — cert, cwe, denial-of-service, leak

```

  /* Sort rows in 1) increasing order of start index, 2) decreasing length, and
     3) non-equalities (i.e. equalities last) */
  n = 0;
  for(i = firstActiveLink(psdata->rows->varmap); i != 0; i = nextActiveLink(psdata->rows->varmap, i)) {
    k = presolve_rowlength(psdata, i);
    if(k >= 2) {
      item = 0;
      ii = presolve_nextcol(psdata, i, &item);
```

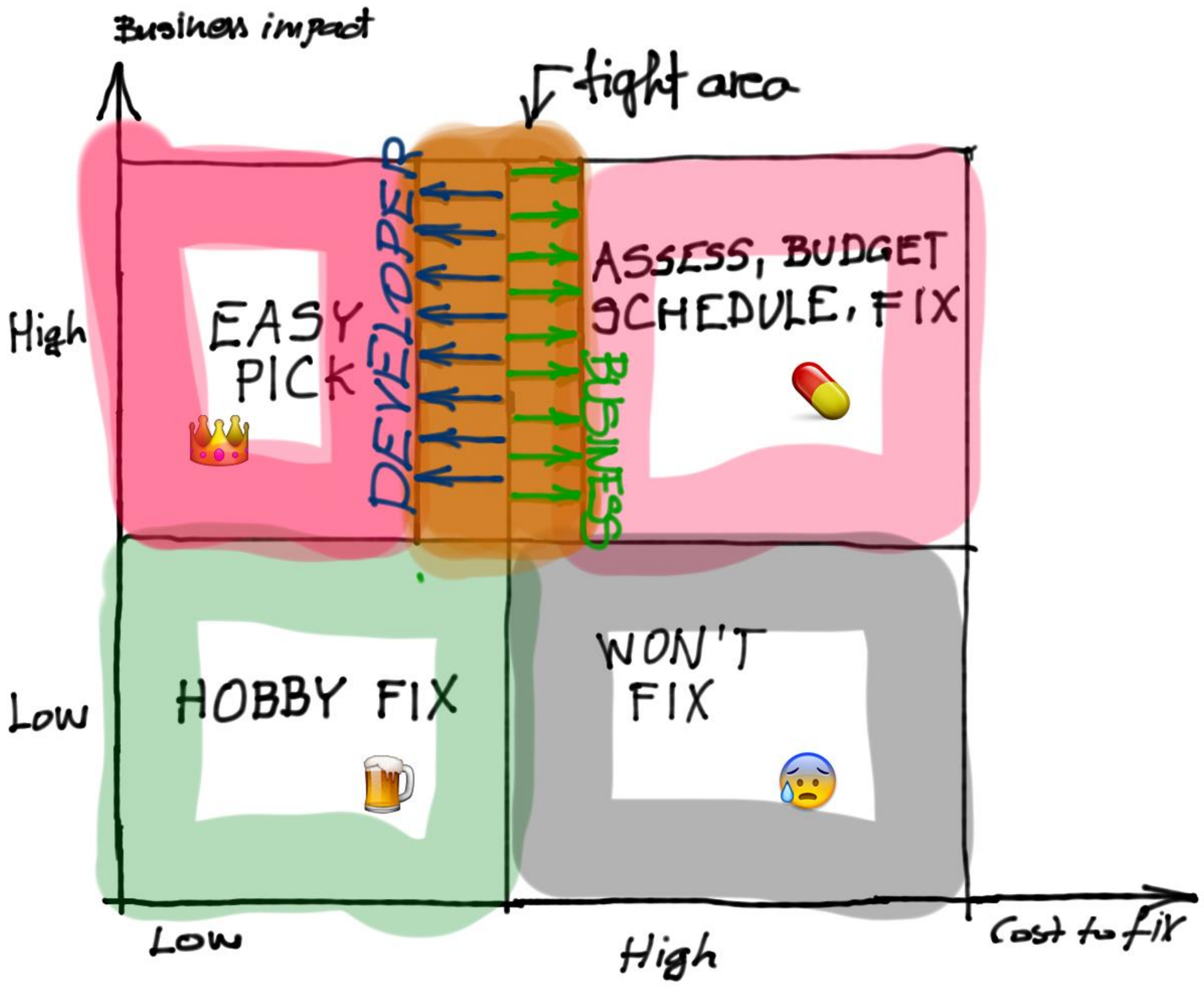
INSPECTIONS VS. TESTING

NASA's Java PathFinder (JPF for short) is an instrumented JVM that provides a state model checker. It runs your program by trying all potential execution paths through it, checking properties as it goes to detect problems like deadlocks or unhandled exceptions.

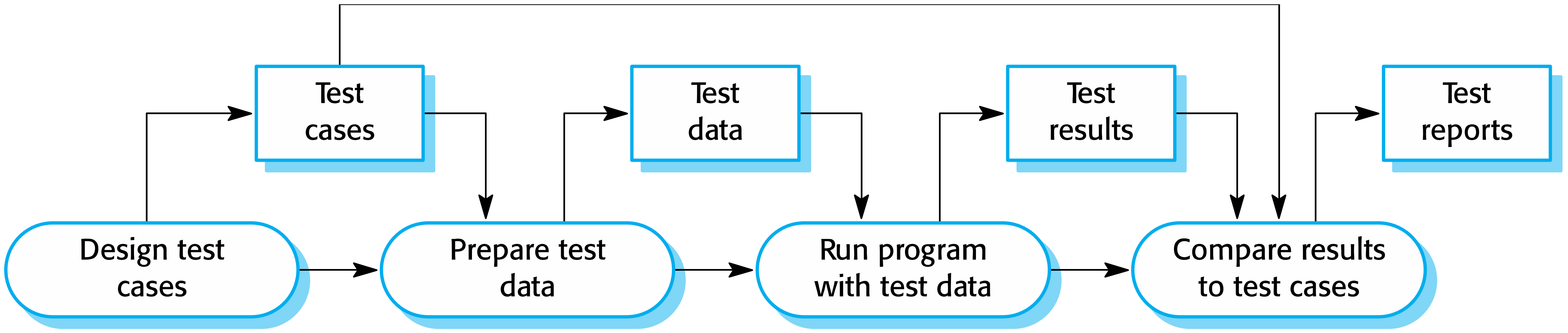


```
① Random random = new Random()

② int a = random.nextInt(2)

③ int b = random.nextInt(3)

④ int c = a/(b+a -2)
```

BUG TRIAGING



A MODEL OF THE SOFTWARE TESTING PROCESS



STAGES OF TESTING

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

DEVELOPMENT TESTING

DEVELOPMENT TESTING

carried out by the team developing the system.

- ✓ Unit testing:
  - for individual program units or object classes
  - focus on testing the functionality of objects or methods.
- ✓ Component testing:
  - several individual units are integrated to create composite components
  - focus on testing component interfaces.
- ✓ System testing:
  - some or all of the components in a system are integrated and the system is tested as a whole
  - focus on testing component interactions.

UNIT TESTING

- ✓ Unit testing is the process of testing individual components in isolation.
- ✓ It is a defect testing process.
- ✓ Units may be:
  - Individual functions or methods within an object
  - Object classes with several attributes and methods
  - Composite components with defined interfaces used to access their functionality.

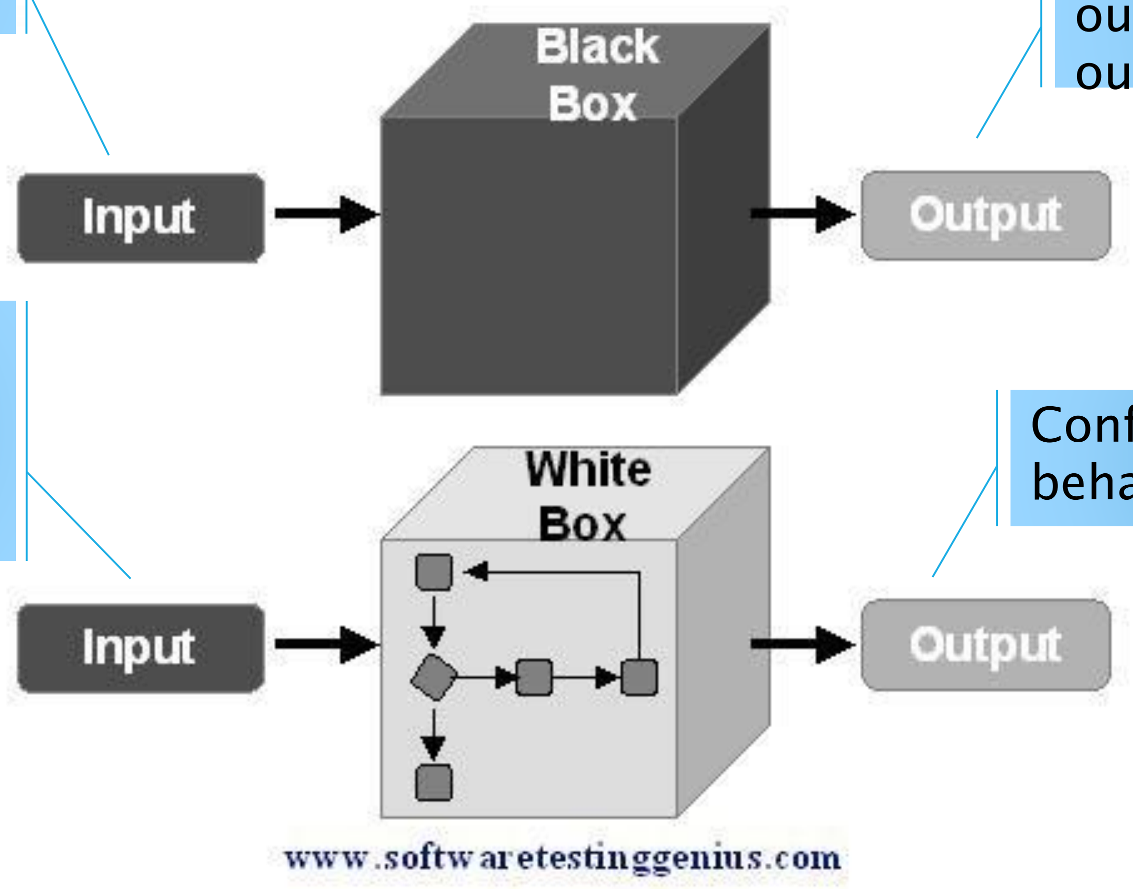
UNIT TESTING: BLACK-/WHITE-BOX TEST

from requirements

Compare actual output with require output

from requirements & key design elements

Confirm expected behaviour



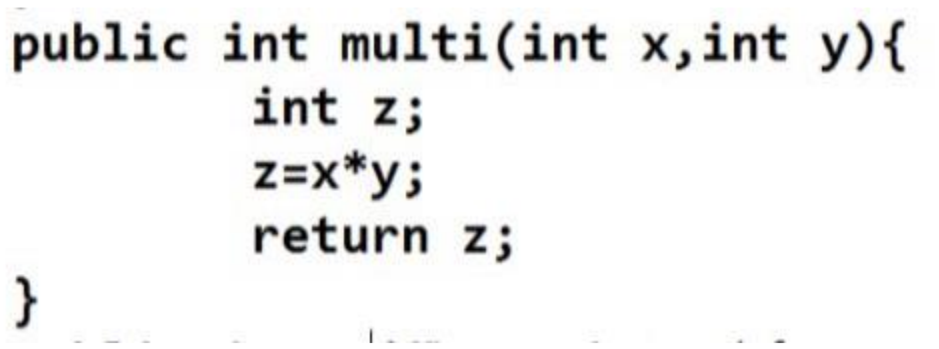
Gray-box: mix of black- and white-box testing

```
for (i=0; i<numrows; i++)
  for (j=0; j<numcols; j++);
    pixels++;
```

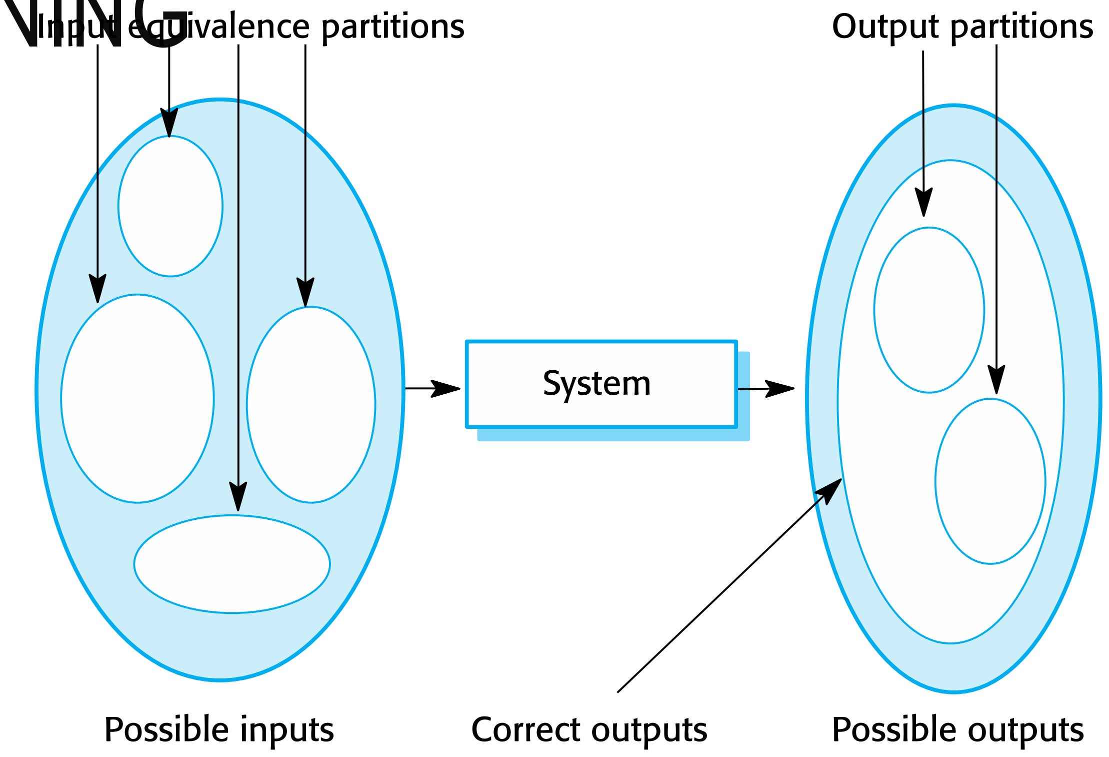
```
int minval(int *A, int n) {
  int currmin;

  for (int i=0; i<n; i++)
    if (A[i] < currmin)
      currmin = A[i];
  return currmin;
}
```

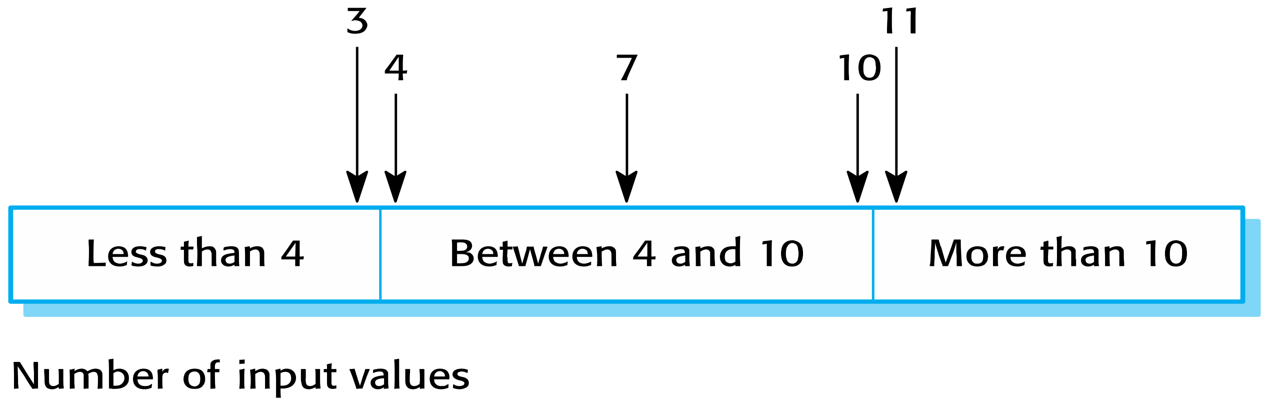
```
switch (i) {
  case 1:
    do_something(1); break;
  case 2:
    do_something(2); break;
  case 3:
    do_something(1); break;
  case 4:
    do_something(4); break;
  default:
    break;
}
```

EQUIVALENCE PARTITIONING

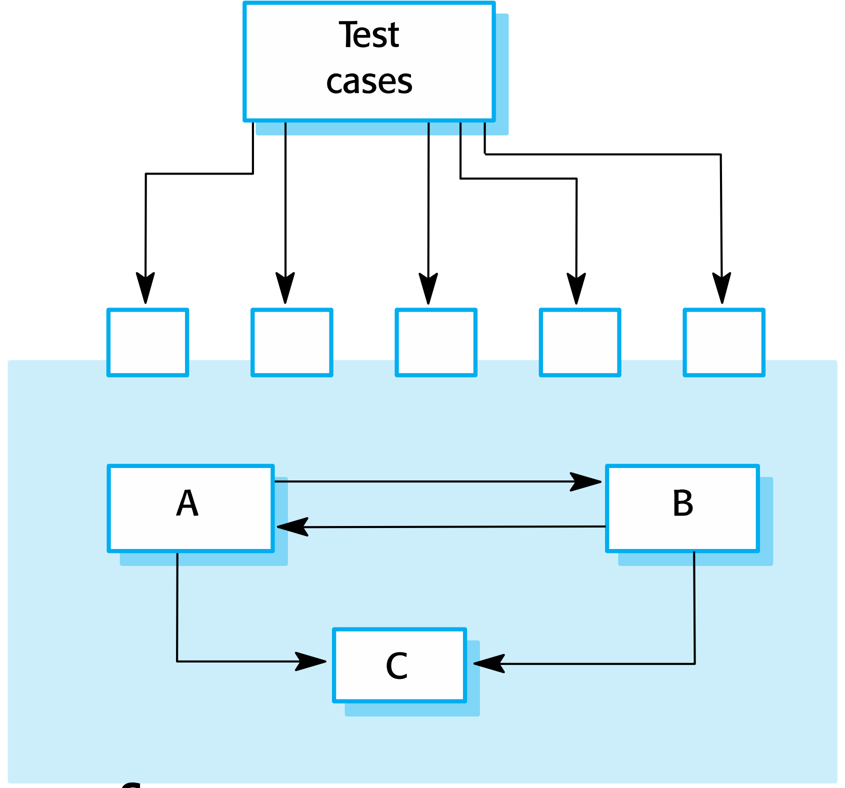


```
public int multi(int x,int y){
        int z;
        z=x*y;
        return z;
}
```





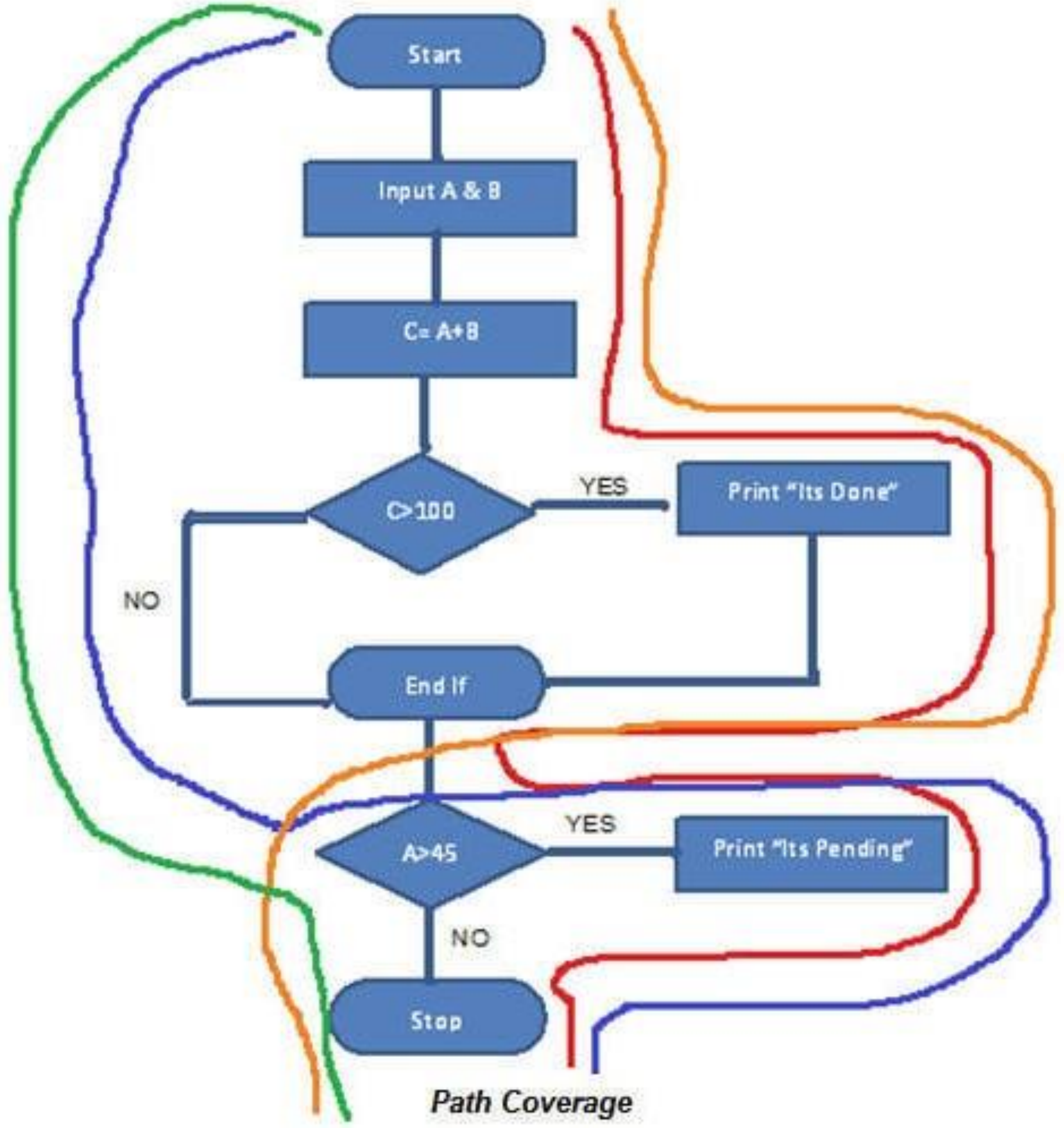
INTERFACE TESTING



- ✓ Detect faults due to
  - interface errors
  - or invalid assumptions about interfaces.
- ✓ Interface types
  - Parameter interfaces
  - Shared memory interfaces
  - Procedural interfaces
  - Message passing interfaces

WHITE-BOX TESTING

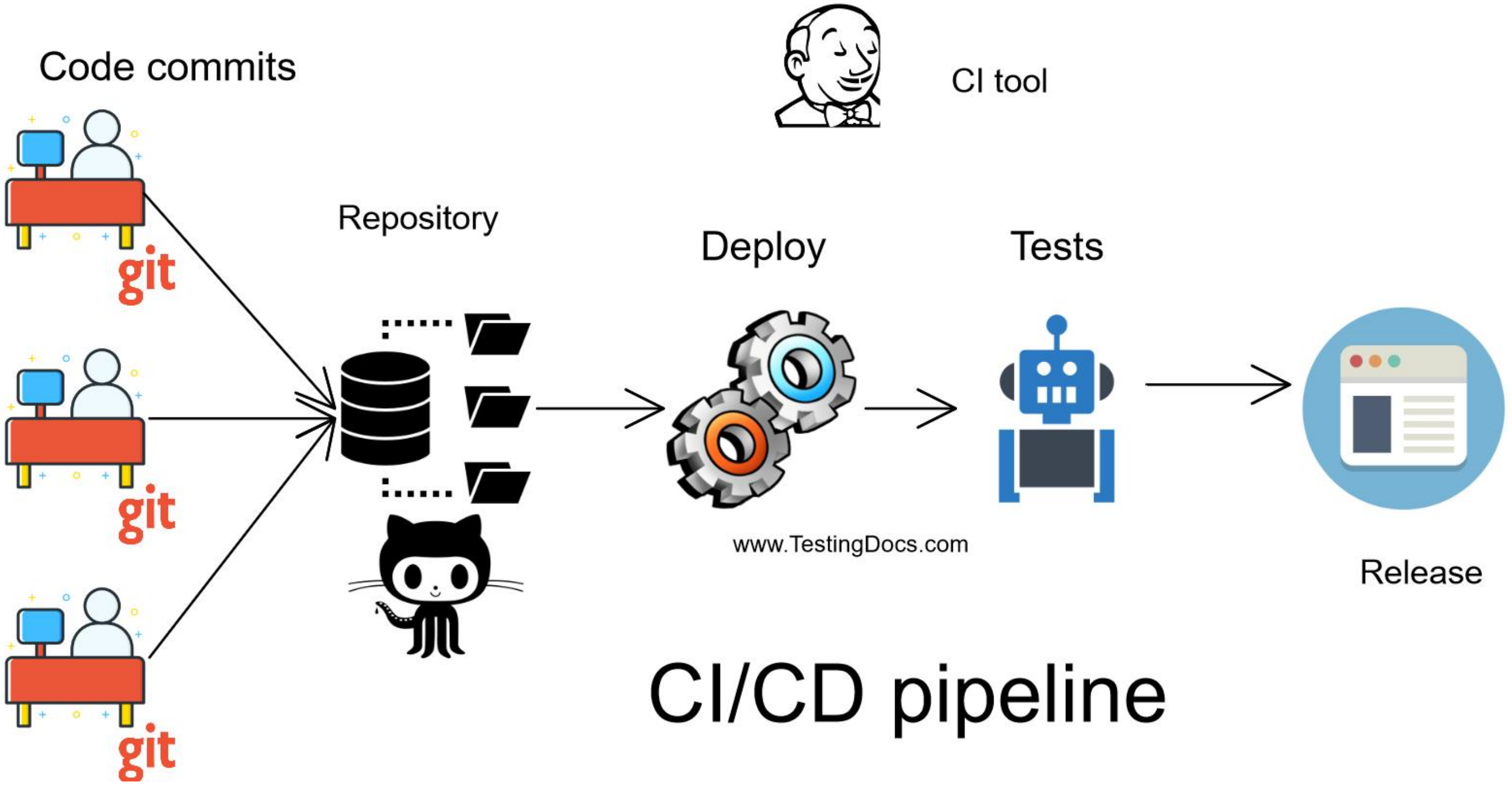
- ✓ Statement coverage
- ✓ Branch coverage
- ✓ Path coverage



Path Coverage

AUTOMATED TESTING

- ✓ Whenever possible, unit testing should be automated
- ✓ Us t)



SYSTEM TESTING

System testing during development = to create a version of the system and then testing the integrated system.

- ✓ Focus on testing the interactions between components.
  - System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces.
- ✓ And tests the emergent behaviour of a system.

TYPES OF SYSTEM TESTS

- ✓ Volume
  - Subject product to large amounts of input.
- ✓ Usability
  - Measure user reaction (e.g., score 1-10).
- ✓ Performance
  - Measure speed under various circumstances.
- ✓ Configuration
  - Configure to various hardware / software
- ✓ Compatibility
  - with other designated applications
- ✓ Reliability / Availability
  - Measure up-time over extended period.
- ✓ Security
  - Subject to compromise attempts.
- ✓ Resource usage
  - Measure usage of RAM and disk space etc.
- ✓ Install-ability
  - Install under various circumstances.
- ✓ Recoverability
  - Force activities that take the application down.
- ✓ Serviceability
  - Service application under various situations.
- ✓ Load / Stress
  - Subject to extreme data & event traffic

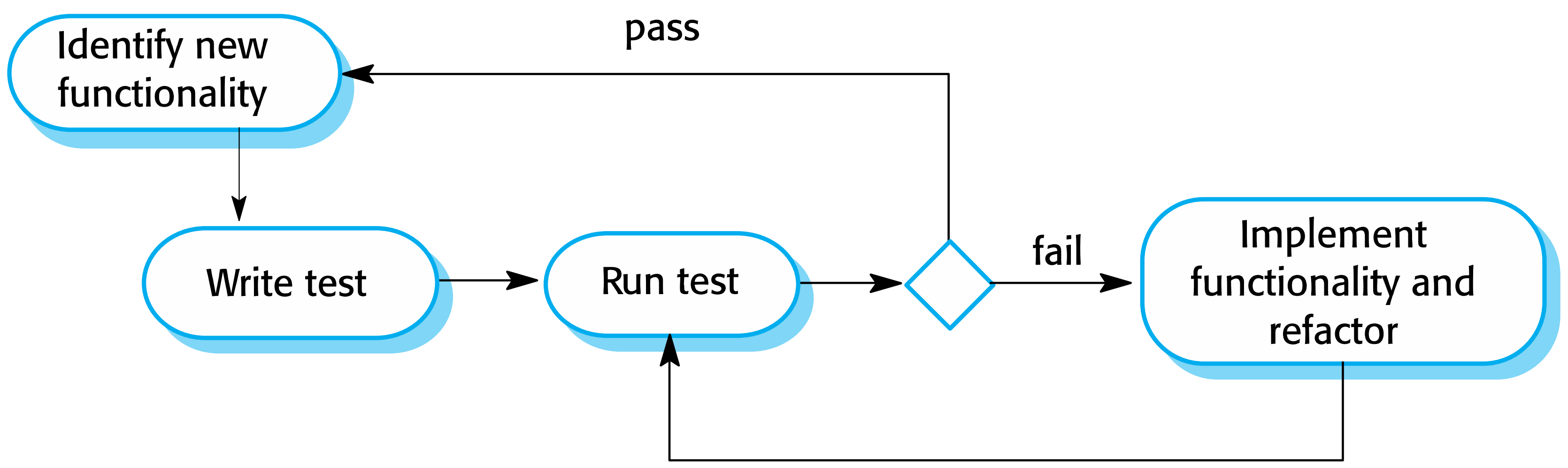
USE-CASE TESTING

The use-cases developed to identify system interactions can be used as a basis for system testing.

- ✓ Each use case usually involves several system components so testing the use case forces these interactions to occur.
  - The sequence diagrams associated with the use case documents the components and interactions that are being tested.

TEST-DRIVEN DEVELOPMENT

inter-leave testing and code development


Identify new functionality
pass
Write test
Run test
fail
Implement functionality and refactor

Benefits of test-driven development

- Code coverage
- Regression testing
- Simplified debugging
- System documentation

REGRESSION TESTING

Test the system to check that changes have not 'broken' previously working code.

- ✓ Better with automated testing
- ✓ All tests are re-run every time a change is made to the program.
- ✓ Tests must run 'successfully' before the change is committed.

TOPICS COVERED

- ✓ Software Quality & its importance
- ✓ Development testing
- ✓ Release testing
- ✓ User testing

RELEASE TESTING

RELEASE TESTING

Test a particular release of a system that is intended for use outside of the development team.

- ✓ Primary goal: to convince that it is good enough for use.
  - Show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.
- ✓ Is usually a black-box testing
  - tests are only derived from the system specification.
- ✓ Is a form of system testing.

REQUIREMENTS BASED TESTING

Involves examining each requirement and developing a test or tests for it.

- ✓ Example: Mentcare system requirements:
  - If a patient is known to be allergic to any particular medication, then prescription of that medication shall result in a warning message being issued to the system user.
  - Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system.
  - Set up a patient record with a known allergy. Prescribe the medication to that the patient is allergic to, and check that the warning is issued by the system.
  - Set up a patient record in which allergies to two or more drugs are recorded. Prescribe both of these drugs separately and check that the correct warning for each drug is issued.
  - Prescribe two drugs that the patient is allergic to. Check that two warnings are correctly issued.

PERFORMANCE TESTING

Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.

- ✓ Tests should reflect the profile of use of the system.
- ✓ Is usually a series of tests
  - the load is steadily increased until the system performance becomes unacceptable.
- ✓ Stress testing
  - is a form of performance testing where the system is deliberately overloaded to test its failure behaviour.

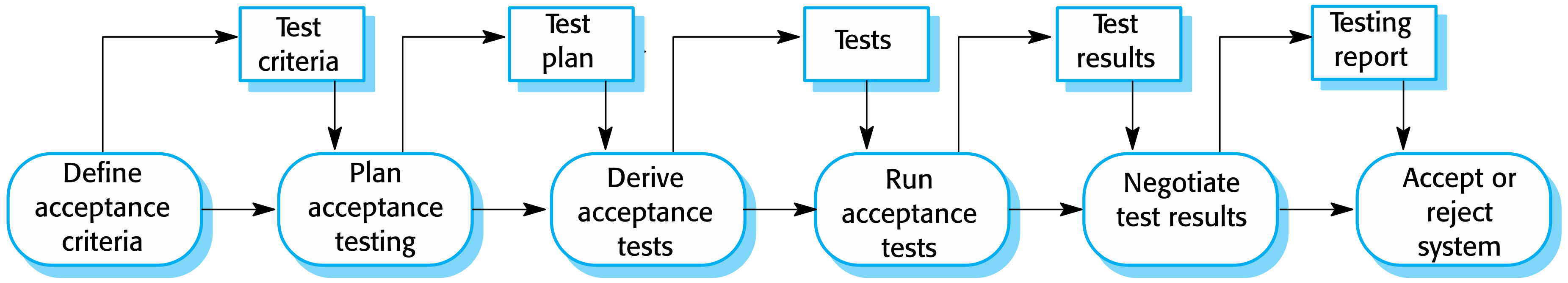
USER TESTING

USER TESTING

A stage in which users or customers provide input and advice on system testing.

- ✓ User testing is essential, even when comprehensive system and release testing have been carried out.
- ✓ Types of user-testing
  - Alpha testing
  - Beta testing
  - Acceptance testing

STAGES IN THE ACCEPTANCE TESTING PROCESS



- ✓ Define acceptance criteria
- ✓ Plan acceptance testing
- ✓ Derive acceptance tests
- ✓ Run acceptance tests
- ✓ Negotiate test results
- ✓ Reject/accept system

STOPPING CRITERIA

- ✓ Completing a particular test methodology
- ✓ Estimated percent coverage for each category
- ✓ Error detection rate
- ✓ Total number of errors found
- ✓ ?

SUMMARY

- ✓ Testing can only show the presence of errors in a program. It cannot demonstrate that there are no remaining faults.
- ✓ Development testing: development team
- ✓ Development testing includes unit testing, component testing, and system testing
- ✓ When testing software: try to ‘break’ the software by using experience and guidelines
- ✓ Wherever possible, you should write automated tests
- ✓ Test-first development: tests are written before the code
- ✓ Scenario testing involves inventing a typical usage scenario and using this to derive test cases.
- ✓ Acceptance testing: user testing process => if the software is good enough to be deployed and used in its operational environment.