Test Architecture for Software Testing

대전대학교 이 지 현

2014/05

Outline

1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example test architecture
6. Summary

Business Success Requires Software Prowess

- Software pervades every sector.
- Software has become the bottom line for many organizations, even those who never envisioned themselves in the software business.

Costly Software Failures

Google

Toyota in Prius global recall after braking software fault

Car maker will re

By Martyn Williams

Toyota plans to re
controls the antil

ntilock braking software in 400,000 cars worldwide

are Err

craft

Ariane 5:
exception-handling
d self
maiden
t to 16-bit

Software fault delays Melbourne flights

AAP DECEMBER 16, 2013 6:15PM

Recommend Share

A SOFTWARE fault ha
flight delays at Melbo

Planes arriving at Melbo
Monday morning were "
flights departing from ot
delayed by about five m
Airservices Australia sa

The maximum delay for
than 30 minutes, it said in a statement.

The software fault was detected in its digital tower technology at its Melbourne air traffic

Software Error Blamed for $308,000 Stock Loss

Tweet g+1 ShareThis

ASHEVILLE, N.C. (CN) - A Millennium Trust software error cost a financial adviser $308,000 by confusing the number of dollars the adviser want to buy for its customers with the number of shares, the adviser claims in court.

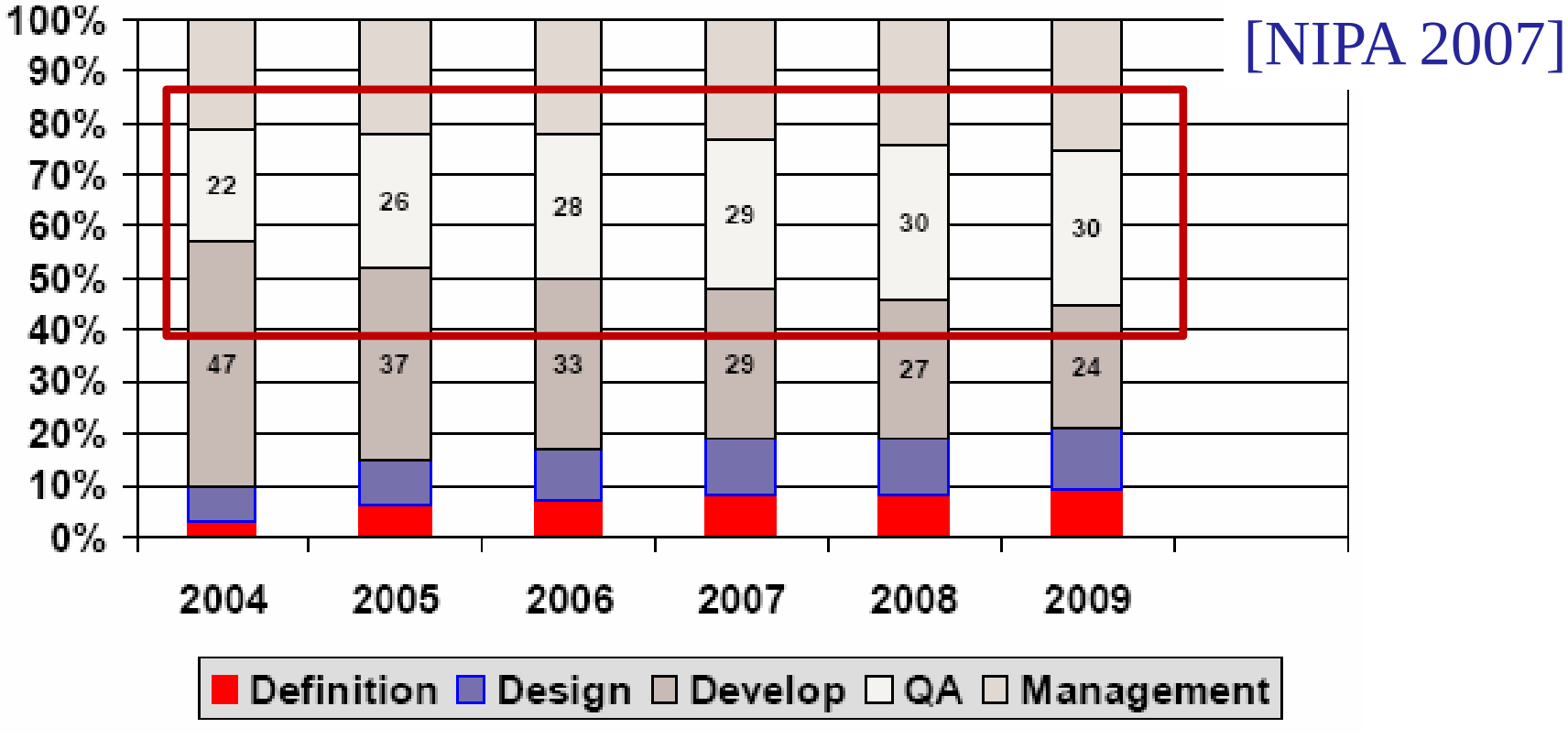
W. Wall and Company a certified financial adviser in Asheville, sued Millennium Trust Company in Buncombe County Court.

Technical issues at Melbourne Airport's control tower have caused flight delays in Melbourne

Costly Software Failures

Software bugs or errors cost the U.S. economy an estimated $59.5 billion annually

► More than $22.2 billion could be eliminated by an improved testing infrastructure. [NIST, 2006]



[NIPA 2007]

What Does This Mean?

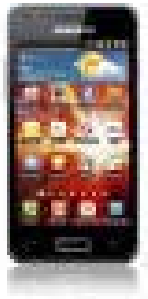
Software testing is becoming more important !!

Outline

1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example
6. Summary

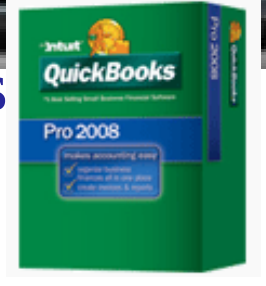
Testing in the Today's SW

- Embedded SW is now ubiquitous
- Real-time software, more safety critical
- Today's SW is much bigger and more complex
  - ✓ Enterprise applications
  - ✓ Embedded control applications
- Security has now become essential
  - ✓ Secure software is reliable software
- Emerging new technologies
  - ✓ Software product line, cloud computing, mobile Apps



5 million LOCs

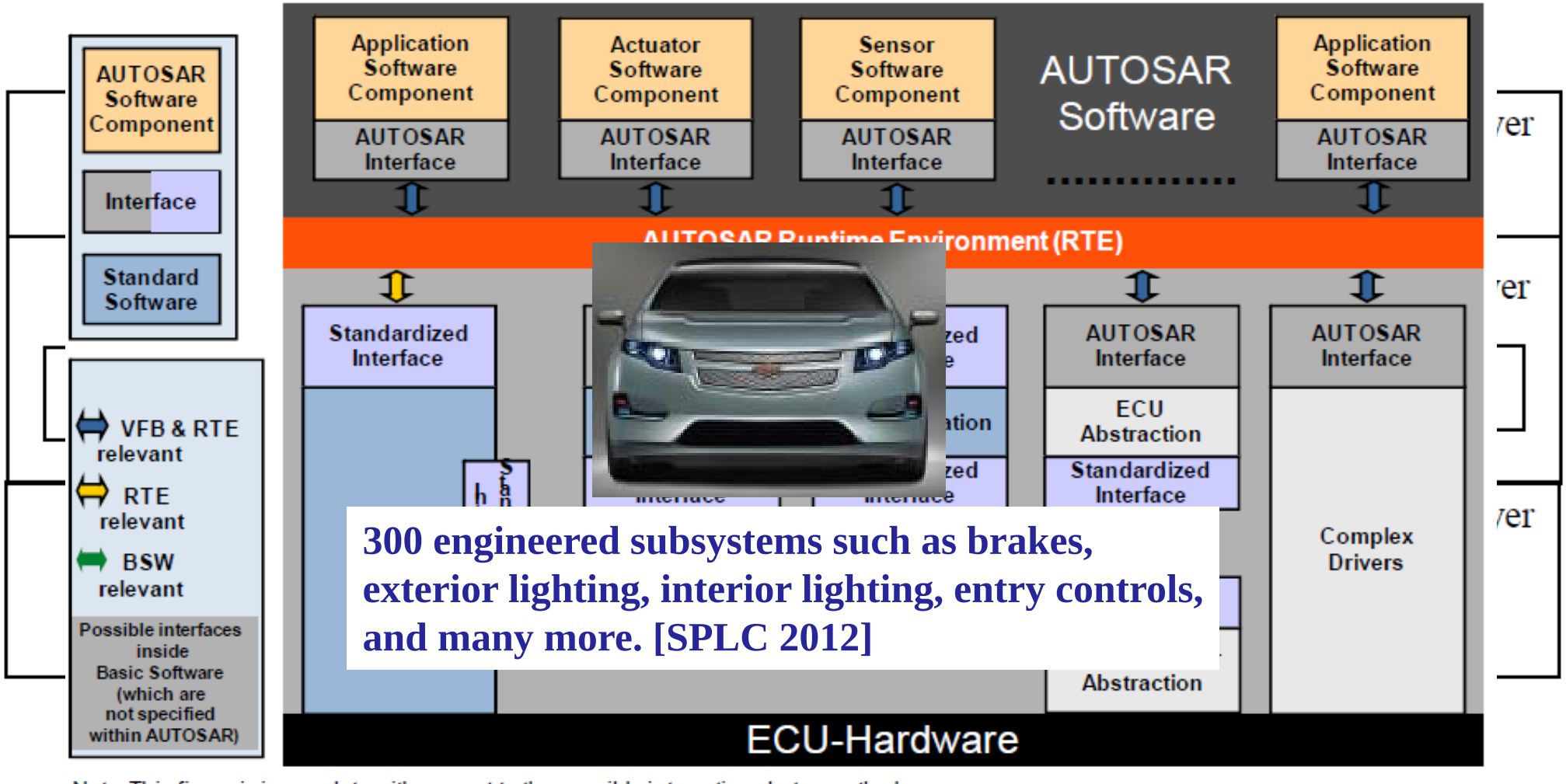
10 million LOCs



Microsoft Dynamics

Several million LOCs

Testing in the Today's SW



300 engineered subsystems such as brakes, exterior lighting, interior lighting, entry controls, and many more. [SPLC 2012]

Note: This figure is incomplete with respect to the possible interactions between the layers.

Other Challenges in Testing SW

- Continuous changes/evolution of SW;
- To reduce cost, reuse is another important issue;
- Testability
  - ✓ Issues on observability & controllability
  - ✓ Methodologies decreasing testability
    - OO development, Web applications

Testability:
If a SW has a fault, how difficult will it be to find a test that causes a failure ?

Outline

1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example test architecture
6. Summary

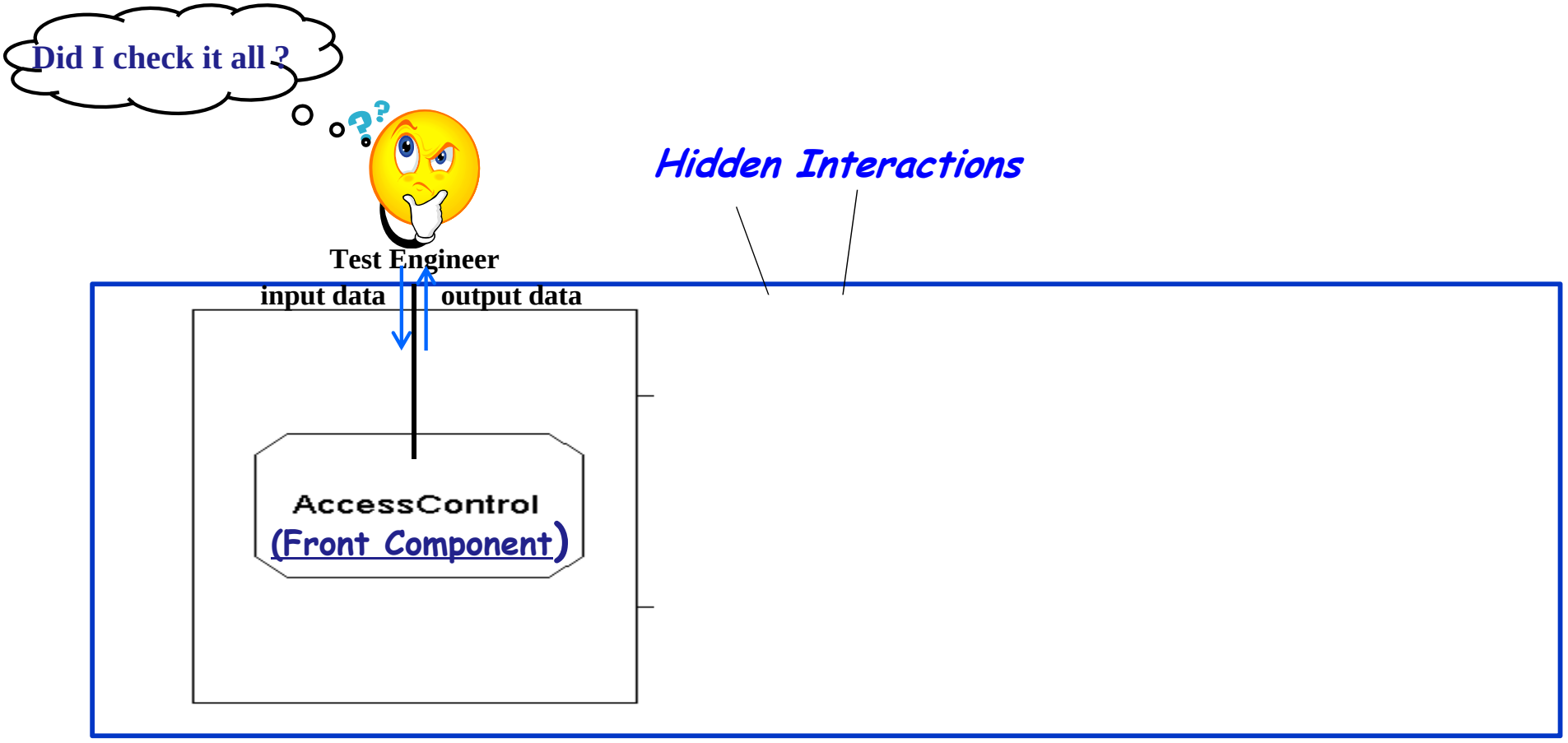
Why Test Architecture? – size, complexity, reuse, evolution, …

- Software architecture
  - ✓ "**designing and specifying the overall system structure [Garlan and Shaw, 1993]"**
  - ✓ a set of important early design decisions
  - ✓ guidelines and puts constraints on implementation
  - ✓ improving the productivity, reusability, and maintainability of software development

If we have something like SW architecture in SW testing, can we expect the same effects as in SW architecture?

Why Test Architecture? - testability

- Integration testing
  - ✓ Limited Observability
    - Only observe test results through the externally visible component
    - Cannot observe hidden behind the front component

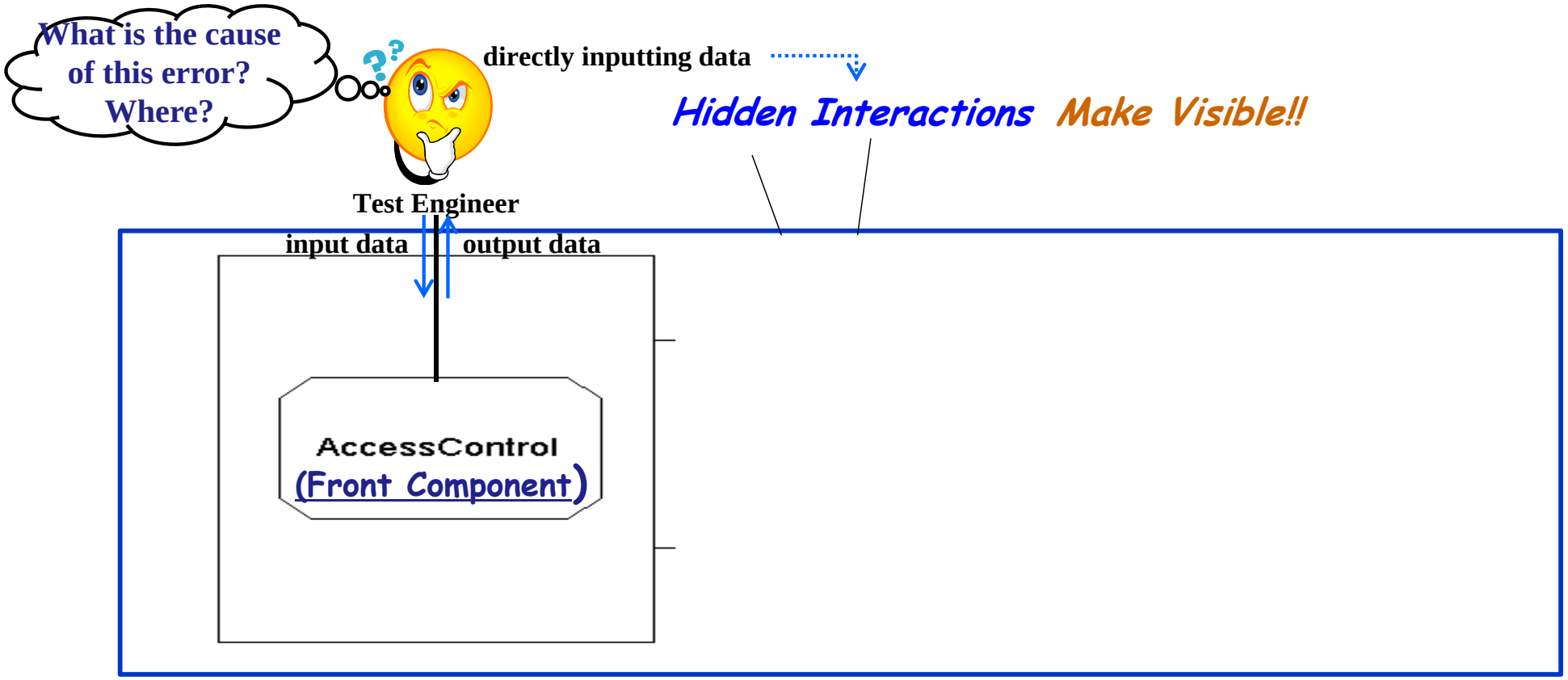


Why Test Architecture? - testability

- Integration testing
  - ✓ Limited Controllability

Test Architecture, First-Class Notion!!

- Cannot control by just sending a sequence of input messages through the front component



Outline

1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example test architecture
6. Summary

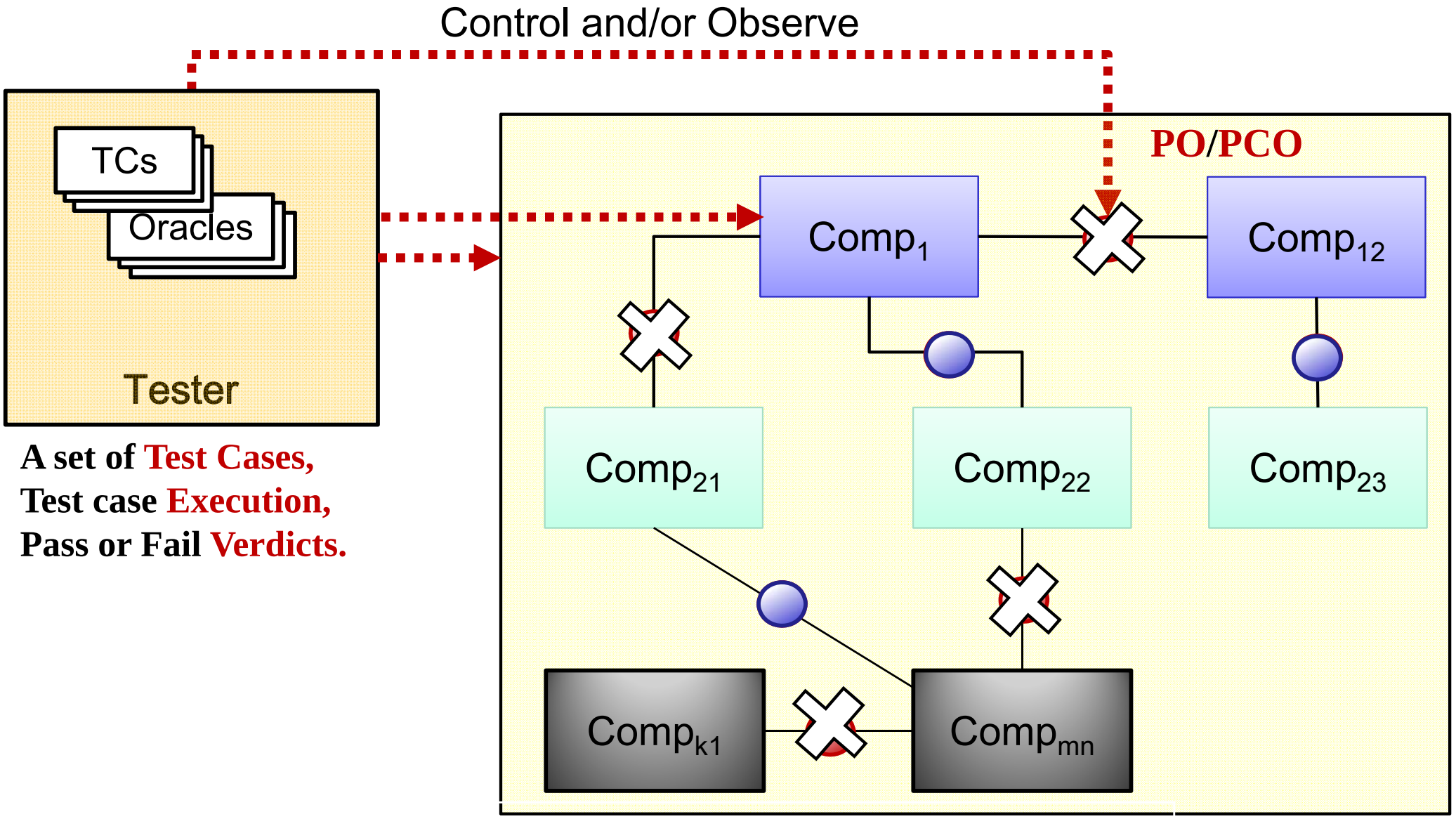
Fundamentals of Test Architecture

- Test Architecture
  - ✓ An overall structure for the testing SUT
  - ✓ Guiding and constraining test derivation and execution

  => play the equivalent role for software testing that software architecture does for software development

- Definition of Test Architecture
  - ✓ A set of Test elements and;
    - Tester
    - PO (Point of Observation)
    - PCO (Point of Control and Observation)
  - ✓ Relationships among test elements, and between test elements and SW architecture

Fundamentals of Test Architecture

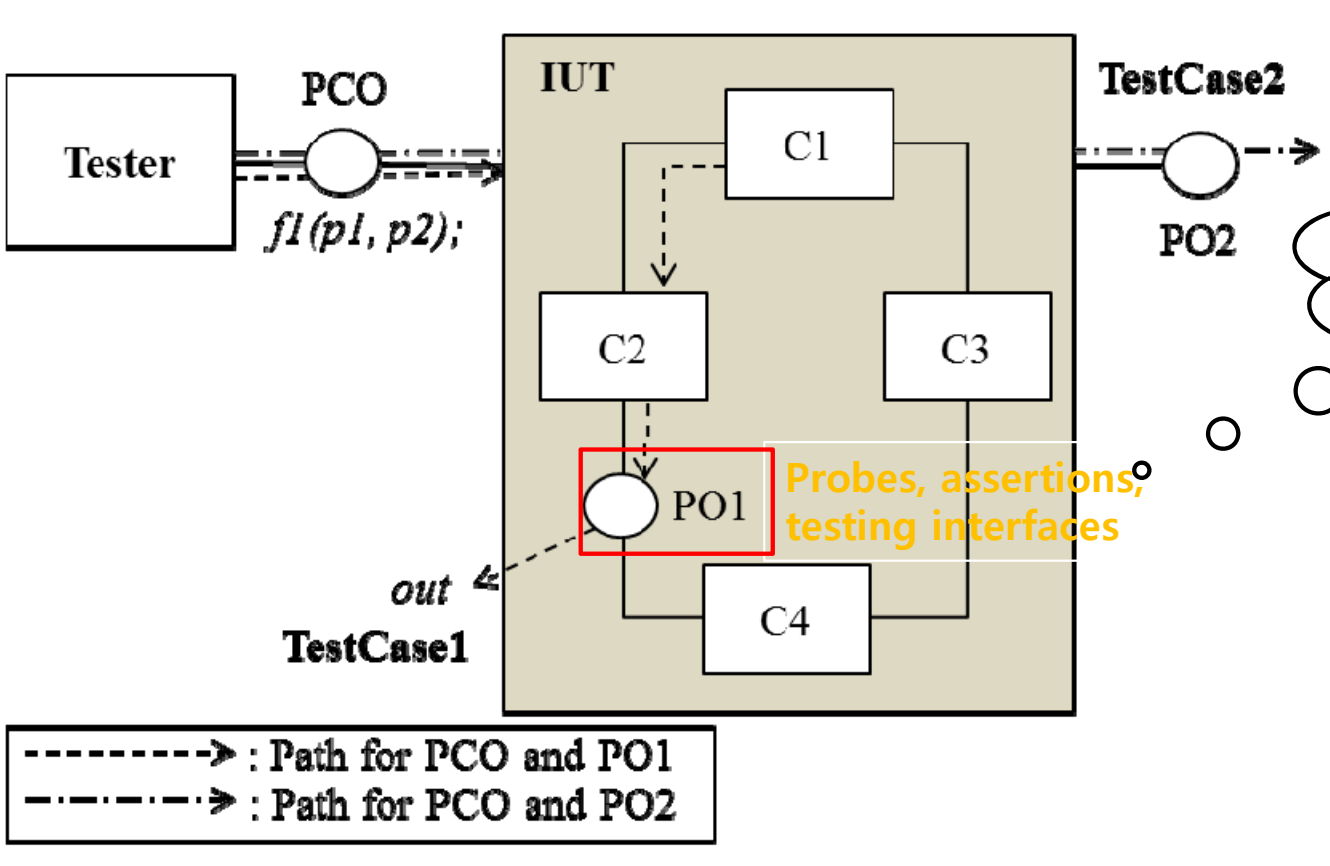


A set of Test Cases,
Test case Execution,
Pass or Fail Verdicts.

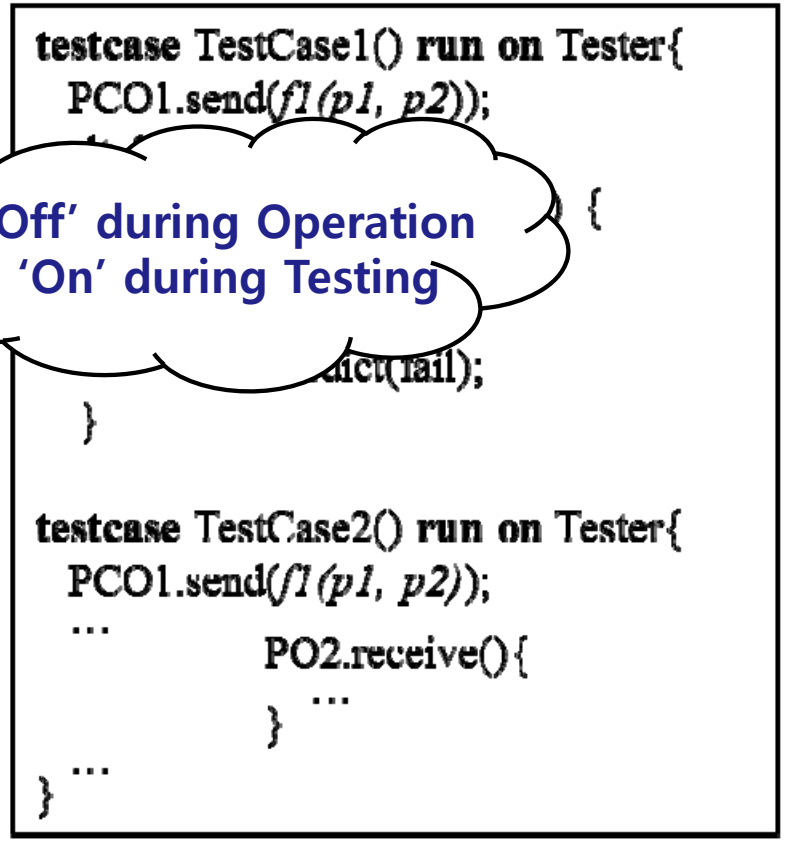
Different test coverage leads to different test architectures!!

Fundamentals of Test Architecture – Types, Implementation

IUT Integrated Test Architecture



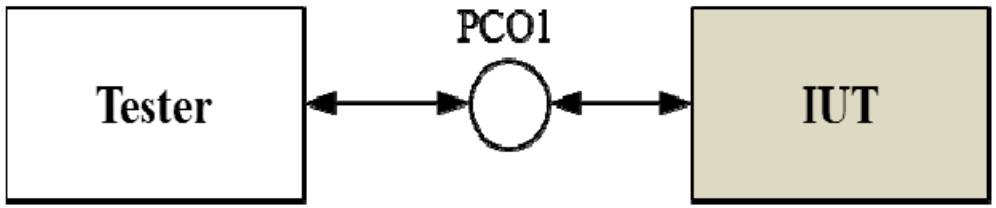
(b-1) IUT integrated TA



(b-2) IUT integrated TA implementation

Fundamentals of Test Architecture – Types, Implementation

IUT Independent Test Architecture



(a-1) IUT independent TA

```
type port PCO1 message{
        inout MyMessage
}

module TestModule{
        ...
        control { execute ( IUTtestcase( ) );
}

testcase IUTtest1( ) runs on Tester {
        ...
        PCO1.send (IUTRequest(...));
        ...
}
```

(a-2) IUT independent TA implementation

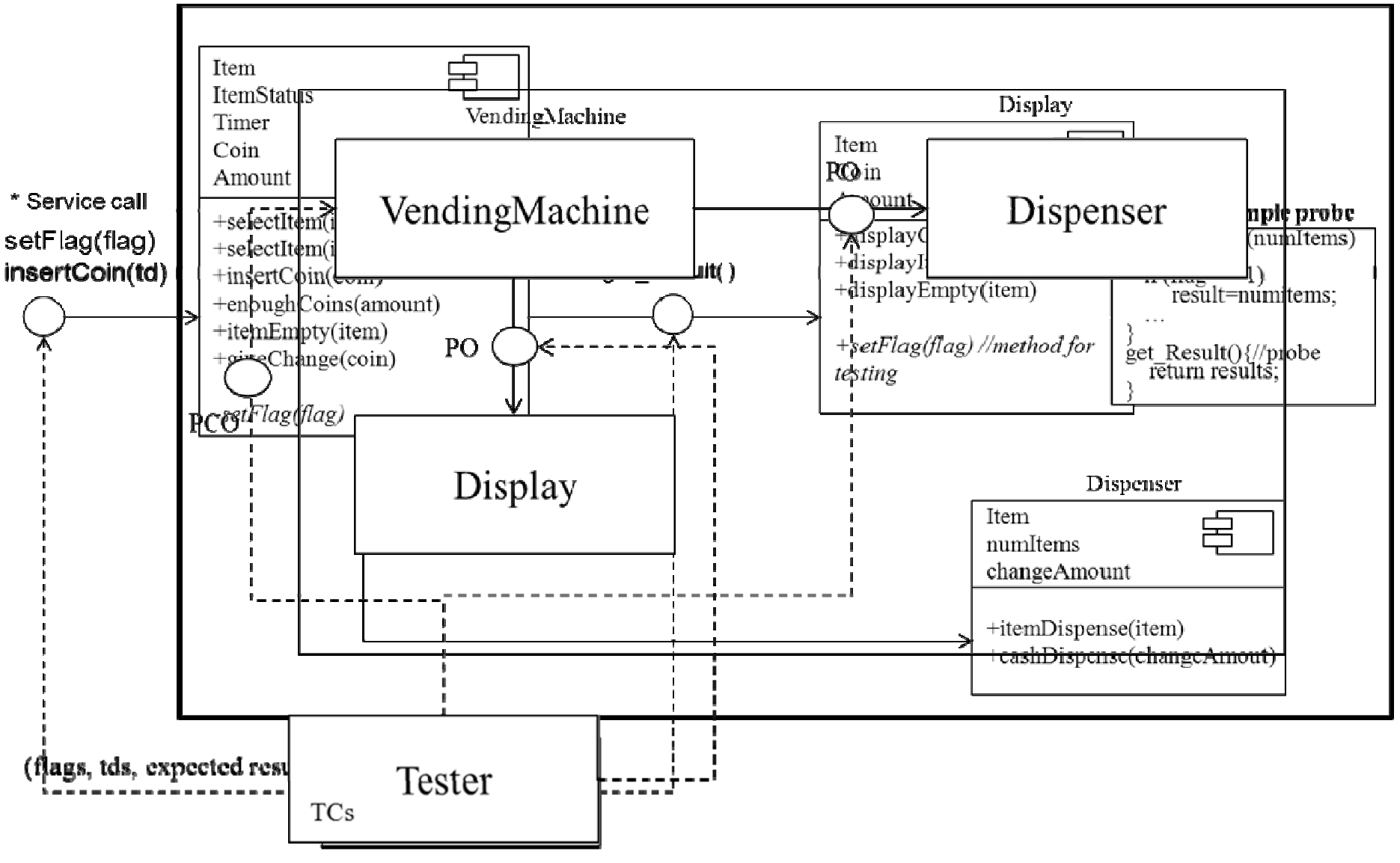
Fundamentals of Test Architecture – Utilizing TA

- Decisions, Guidelines for
  - Structure of a test architecture
    - By adding POs/PCOs to the software architecture at appropriate locations;
    - By defining the interactions among test elements; and
    - By defining interactions between the test elements and the components of the software architecture
  - Test coverage
    - Differentiating locations and the number of POs/PCOs
    - Estimating test efforts based on the test architecture by calculating the costs devoted to adding POs/PCOs and creating test cases together with test data in accordance with the POs/PCOs
  - Interaction mechanisms
    - between test elements; and
    - between the test elements and the components

Outline

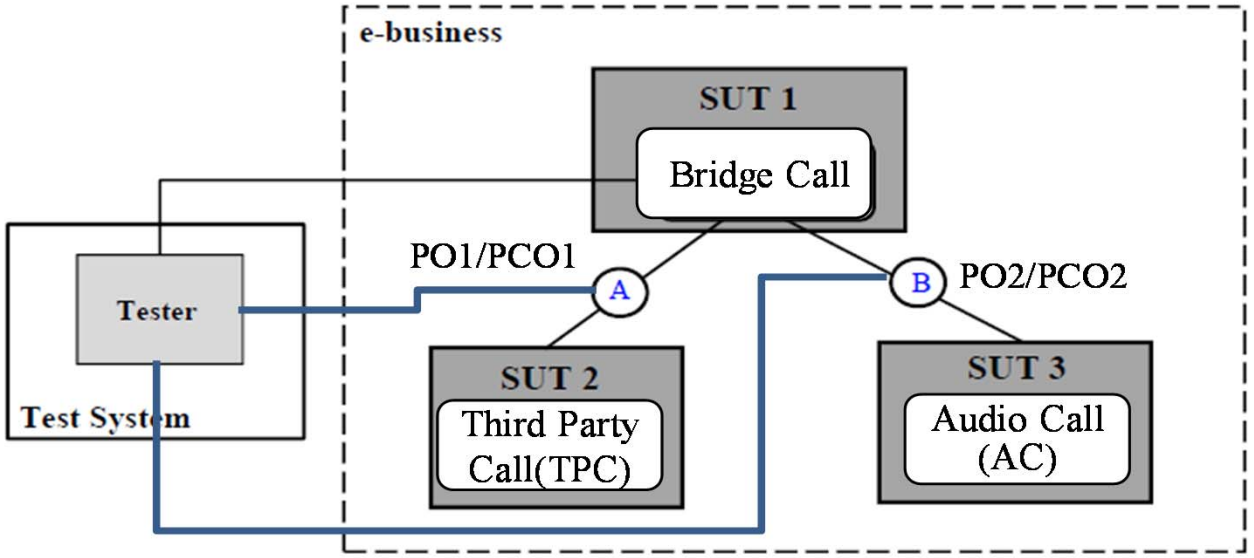
1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example test architecture
6. Summary

Example - IUT Integrated Test Architecture



(b) Test architecture implementation for the vending machine

Example - IUT Independent Test Architecture



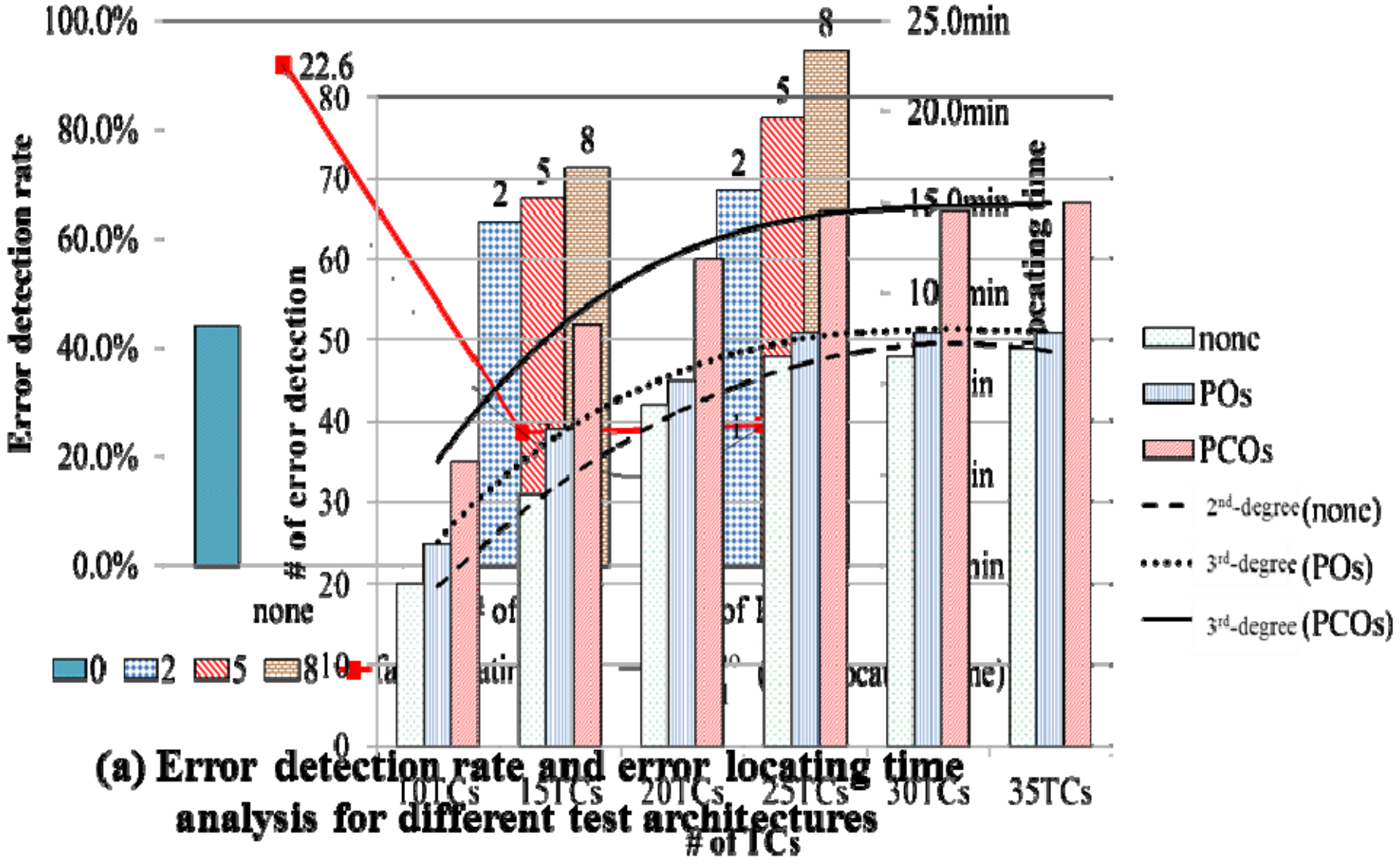
(a) Part of TA for CRM application

Tester!StartCallSessionRq(phone, name) → Tester!MakeCallSessionRq(phone, name) → *PO1?MakeCallSessionRq(phone, name)*→ *PO1?MakeCallSessionRp(callSessionId)* → Tester?MakeCallSessionRp (callSessionId) → Tester!PlayTextMessageRq(callSessionId, "msg") → *PO2?PlayTextMessageRq(callSessionId, "msg")*→ *PO2?PlayTextMessageRp(correlator)* → Tester!GetMessageStatusRq(correlator) →*PO2?GetMessageStatusRq(correlator)* →*PO2?GetMessageStatusRp(Status)* →Tester? GetMessageStatusRp(Status)

(b) Test architecture baseded test scenarios with POs

Tester!StartCallSessionRq(phone, name) → Tester!MakeCallSessionRq(phone, name) → *PCO1?MakeCallSessionRq(phone, name)*→ *PCO1!MakeCallSessionRq(phone', name')* → *PCO1?MakeCallSessionRp(callSessionId)* → *PCO1?MakeCallSessionRp(callSessionId')* → Tester?MakeCallSessionRp(callSessionId) → Tester!PlayTextMessageRq(callSessionId, "msg1") → *PCO2?PlayTextMessageRq(callSessionId, "msg1")*→ *PCO2!PlayTextMessageRq(callSessionId', "msg2")* → *PCO2?PlayTextMessageRp(correlator)* → *PCO2?PlayTextMessageRp(correlator')* → Tester!GetMessageStatusRq(correlator) →*PCO2?GetMessageStatusRq(correlator)* → *PCO2?GetMessageStatusRp(Status)* →Tester? GetMessageStatusRp(Status)

(c) Test architecture based test scenarios with PCOs



(a) Error detection rate and error locating time analysis for different test architectures

(b) Error detection rates as the number of test cases changes

Outline

1. Software Faults
2. What makes software testing difficult?
3. Why test architecture?
4. Fundamentals of test architecture
5. Example test architecture
6. Summary

Summary

“We have entered a golden age for software testing,,

- Still remaining the hardest problems in testing
  - ✓ Testability
  - ✓ Reliability
  - ✓ Security …
- Test Architecture
  - ✓ A systematic way of thinking on SW testing
  - ✓ Pros.
    - Grasps the overall images of testing
    - Facilitates test coverage control by making POs/PCOs explicit
    - Solves test related quality problem, testability

Summary

- Test Architecture
  - ✓ Cons.
    - Requires more efforts to implement POs/PCOs (especially PCO is more expensive)
    - For increasing observability
      - inserting code for test probes, assertions, and/or testing interfaces at strategic positions required