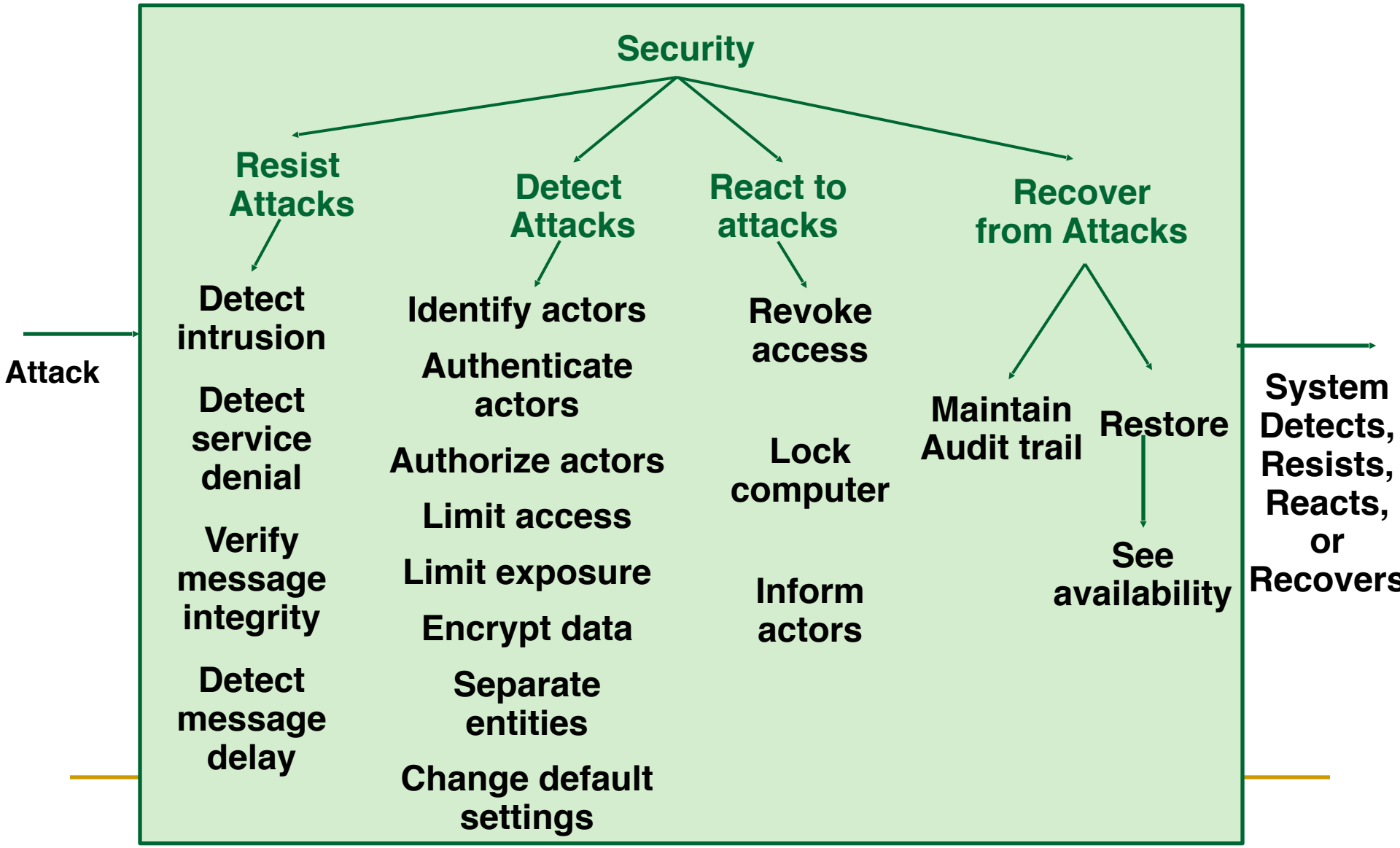
Tactics for Security (2)

主讲教师：王灿
Email: wcan@zju.edu.cn
TA: 李奇平 liqiping1991@gmail.com
Course FTP: ftp://sa:sa@10.214.51.13

Some Proverbs

- No fortress is impregnable
- A chain is only as strong as its weakest link
- The easiest way to capture a fortress is from within

Security Tactics Hierarchy



Recover from Attacks

- Systems attacked need to recover:
  - Restoring data and service
    - Tactics from availability: backups, passive redundancy, checkpoints etc.
  - Attacker identification
    - Audit trails for
      - Trace the action of the attacker
      - Help identify intruder and support non-repudiation
      - Should be protected/hidden

Architectural Design Support for Security

- We check the architectural design and analysis process for security from the following 7 aspects:
  1. Allocation of responsibilities
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Choice of technology

Allocation of Responsibilities

- Identify responsibilities need to be secure and ensure the following supports for them:
  - Identify/authenticate/authorize/grant access to/ revoke access from actors
  - Record & notify
  - Encrypt and verify data
  - Recovery

Coordination Model

- For system communication and coordination:
  - Authenticate/authorize actors for connections
  - Encrypted transmission
  - Monitor and identify connections with unexpectedly high demands for resources, and capability to restrict or terminate these connections

Data Model

- Identify sensitive data fields and for these data:
  - Ensure that data of different sensitivity is separated and has different access
  - Audit trail for access
  - Encrytion with keys stored separately
  - Can be restored

Mapping among Architectural Elements

- Check how alternative mappings may affect security by checking:
  - Data/service access
  - Audit trail
  - Recognition of anomaly
  - Ensure the responsibilities for security support

Resources Management

- Determine the system resources for
  - Identify/monitor/authenticate/authorize an actor
  - Encrypt data
  - Log & notify
- Ensure shared resources are not used for passing sensitive data

Binding Time Decisions

- Determine cases where an instance of a late-bound component may be untrusted and ensure security check for them

Choice of Technology

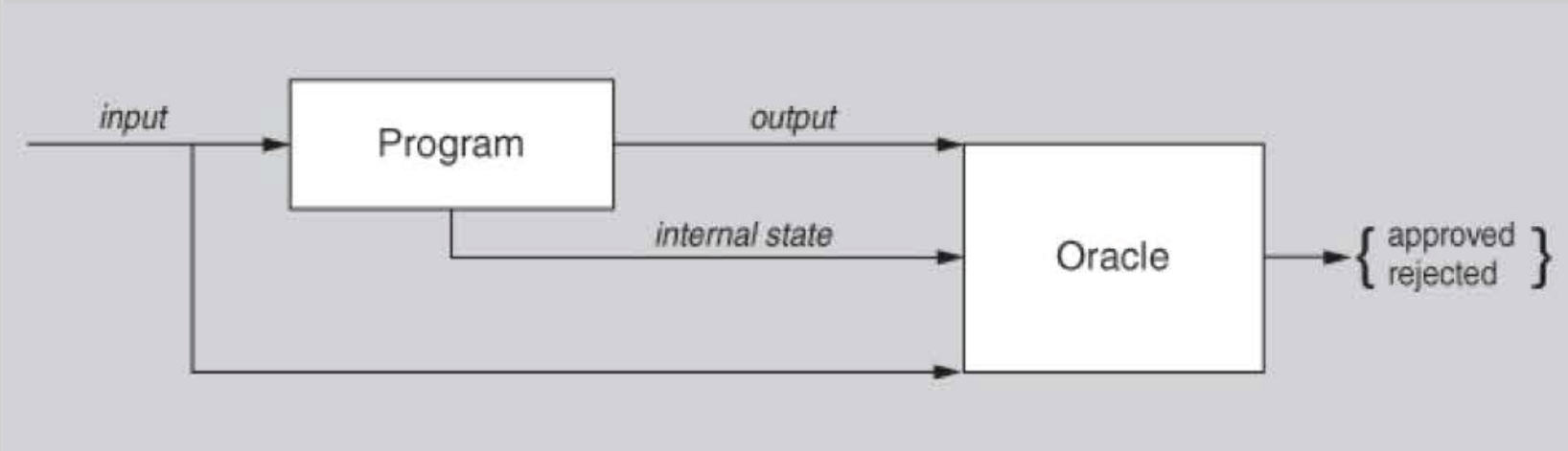
- Your choice of tech. shall support authentication, data access rights, resource protection, and data encryption.

Tactics for Testability

Software Testing

- 40% of the cost of developing systems is taken up by testing
  - If the architect can reduce this cost through design then the payoff is substantial
- Software testability refers to the ease with which the software can be made to demonstrate its faults through testing
  - Probability that a fault will be revealed on its next execution

A Model of Testing



- Output = computational results + some QAs (e.g. response time)
- The ***internal state*** provides the oracle more insights

Testability

- **Controllability**
  - Be able to control each component's inputs
  - Sometimes also need to manipulate its internal state
- **Observability**
  - Be able to observe its outputs
  - Sometimes also need to observe its internal state
- Operability, decomposability, stability, understandability
- Testing is usually performed using a ***test harness***

Testability General Scenario (1)

- Stimulus
  - A milestone in the development process is met
    - The completion of a coding increment (e.g. a class layer or service), integration completed, system delivered etc.
- Source of stimulus
  - The testing is performed by unit testers, integration testers, system testers, acceptance testers, or end users
- Environment
  - Design/ development /compile /deployment / integration/ run time

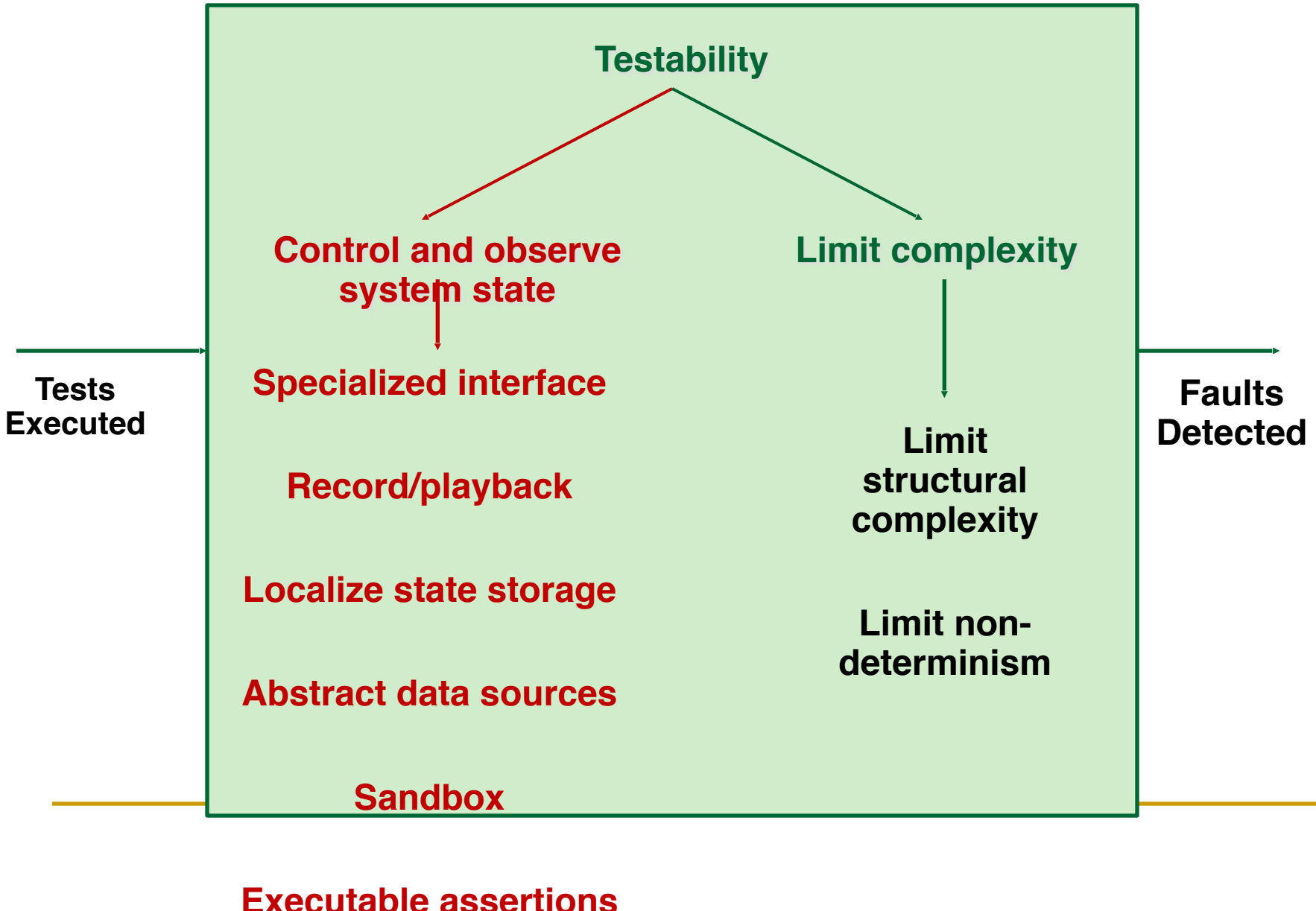
Testability General Scenario (2)

- Artifact
  - The portion of the system being tested: design, a piece of code, or the whole system etc.
- Response
  - Execute test suite and capture results, capture activity that resulted in the fault, control and monitor the state of the system
- Response Measure
  - Coverage
  - Time to prepare the test environment and perform tests
  - Length of the longest dependency chain in a test

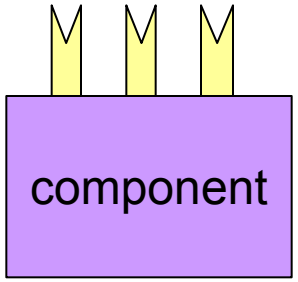
Testability Tactics

- Architectural techniques for improving testability
  - Control and observe system state
    - Adding controllability and observability to the system
  - Limit complexity
    - Limiting complexity in the system's design

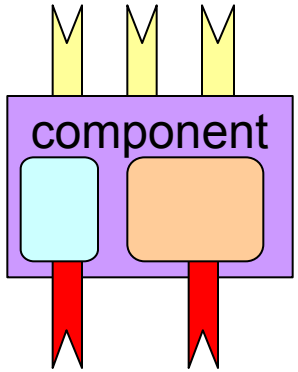
Testability Tactics Hierarchy



Specialized Interfaces (1)



A component with normal interface to provide required functionality



A component with specialized interface

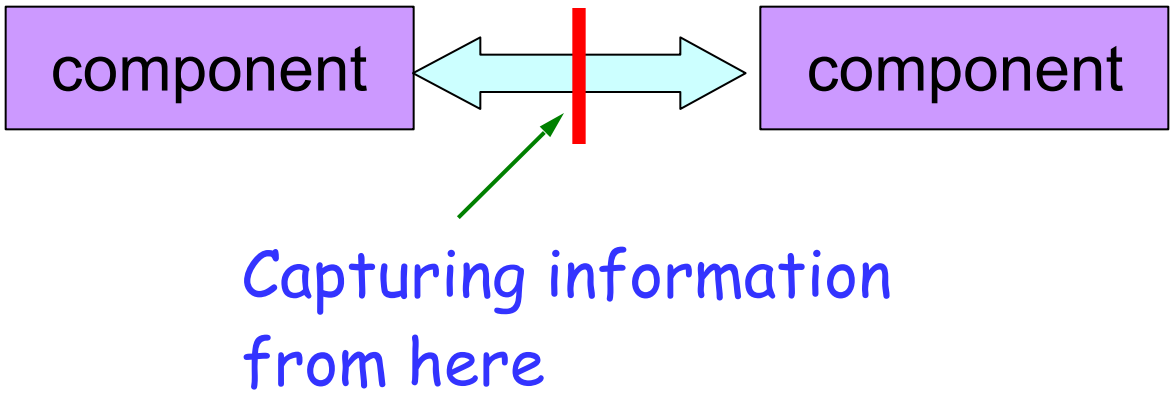
- Allows the capturing or specification of variable values for a component through a test harness or through its normal execution.

Specialized Interfaces (2)

- Some common examples:
  - A ***set*** and ***get*** method for important variables, modes, or attributes
  - A ***report*** method that returns the full state of the object
  - A ***reset*** method to set the internal state
  - A method to turn on verbose output, various levels of event logging, performance instrumentation, or resource monitoring

Record/Playback

- The state that caused a fault is often difficult to re-create
  - Recording the state when it crosses an interface allows that state to be used to "play the system back" and to recreate the fault



Localize State Storage

- If the state control is distributed for a module (or system), the state setting in testing will be difficult
  - Localize state storage thus makes testing convenient
  - May use a state machine object to more effectively manipulate, track and report a module state

Abstract Data Sources

- Flexibly controlling input data will make testing easier
- Abstracting the data source interfaces lets you substitute test data easily
  - E.g. a system with a database of customer transactions, flexible data source connection means it can easily connects to files of test data

Sandbox

- Sandboxes are a specific example of virtualization that isolates untested code changes and experimentation from the production environment or repository
  - E.g. The Spring framework supports running tests as a "transaction", which is rolled back at the end
- The sandbox typically provides a tightly controlled set of virtualized resources for running tests
  - Network access, the ability to inspect the host system or read from input devices are usually disallowed or heavily restricted

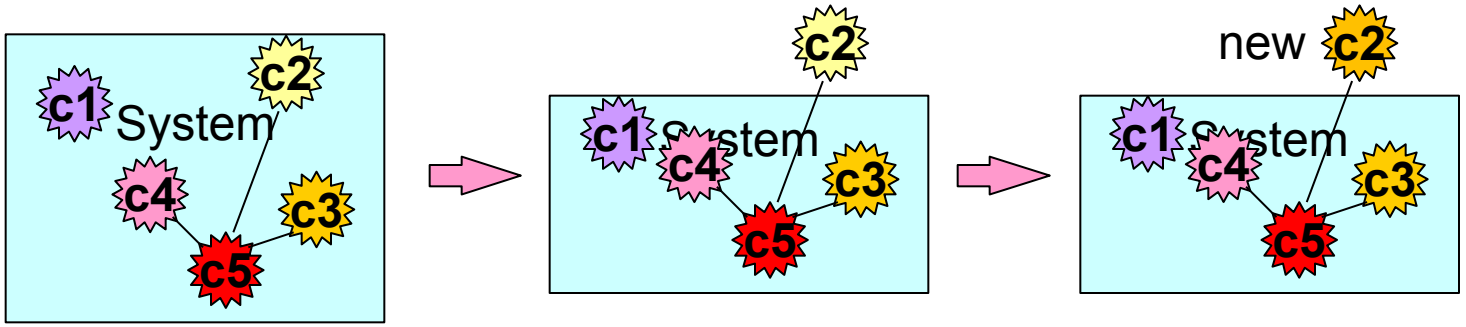
Executable Assertions

- An assertion is a predicate (a true–false statement) placed in a program, where the checking code is automatically executed, and warnings are generated if any conditions are violated

```
x = 1;
assert (x > 0);
x++;
assert (x > 1);
```

Replacement Techniques for Testability

- Component replacement: (used with "separate interface from implementation") replace component with a different implementation



- Preprocessor macros: when activated, will enable state-reporting or probe statements that return or display information, or return control to a testing console

Reading Assignment

- Read Chapter 10 & 11 of the textbook.