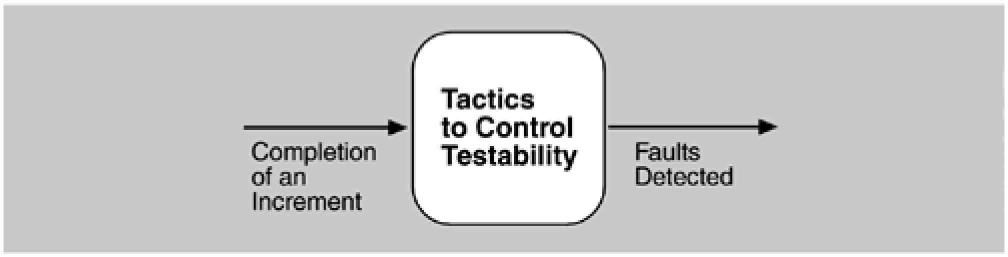
## 5.6 Testability Tactics

The goal of tactics for testability is to allow for easier testing when an increment of software development is completed. Figure 5.10 displays the use of tactics for testability. Architectural techniques for enhancing the software testability have not received as much attention as more mature fields such as modifiability, performance, and availability, but, as we stated in Chapter 4, since testing consumes such a high percentage of system development cost, anything the architect can do to reduce this cost will yield a significant benefit.

**Figure 5.10. Goal of testability tactics**



Although in Chapter 4 we included design reviews as a testing technique, in this chapter we are concerned only with testing a running system. The goal of a testing regimen is to discover faults. This requires that input be provided to the software being tested and that the output be captured.

Executing the test procedures requires some software to provide input to the software being tested and to capture the output. This is called a test harness. A question we do not consider here is the design and generation of the test harness. In some systems, this takes substantial time and expense.

We discuss two categories of tactics for testing: providing input and capturing output, and internal monitoring.

### INPUT/OUTPUT

There are three tactics for managing input and output for testing.

- *Record/playback.* Record/playback refers to both capturing information crossing an interface and using it as input into the test harness. The information crossing an interface during normal operation is saved in some repository and represents output from one component and input to another. Recording this information allows test input for one of the components to be generated and test output for later comparison to be saved.

- *Separate interface from implementation.* Separating the interface from the implementation allows substitution of implementations for various testing purposes. Stubbing implementations allows the remainder of the system to be tested in the absence of the component being stubbed. Substituting a specialized component allows the component being replaced to act as a test harness for the remainder of the system.

- *Specialize access routes/interfaces.* Having specialized testing interfaces allows the capturing or specification of variable values for a component through a test harness as well as independently from its normal execution. For example, metadata might be made available through a specialized interface that a test harness would use to drive its activities. Specialized access routes and interfaces should be kept separate from the access routes and interfaces for required functionality. Having a hierarchy of test interfaces in the architecture means that test cases can be applied at any level in the architecture and that the testing functionality is in place to observe the response.

### INTERNAL MONITORING

A component can implement tactics based on internal state to support the testing process.

- *Built-in monitors.* The component can maintain state, performance load, capacity, security, or other information accessible through an interface. This interface can be a permanent interface of the component or it can be introduced temporarily via an instrumentation technique such as aspect-oriented programming or preprocessor macros. A common technique is to record events when monitoring states have been activated. Monitoring states can actually increase the testing effort since tests may have to be repeated with the monitoring turned off. Increased visibility into the activities of the component usually more than outweigh the cost of the additional testing.

Figure 5.11 provides a summary of the tactics used for testability.

**Figure 5.11. Summary of testability tactics**

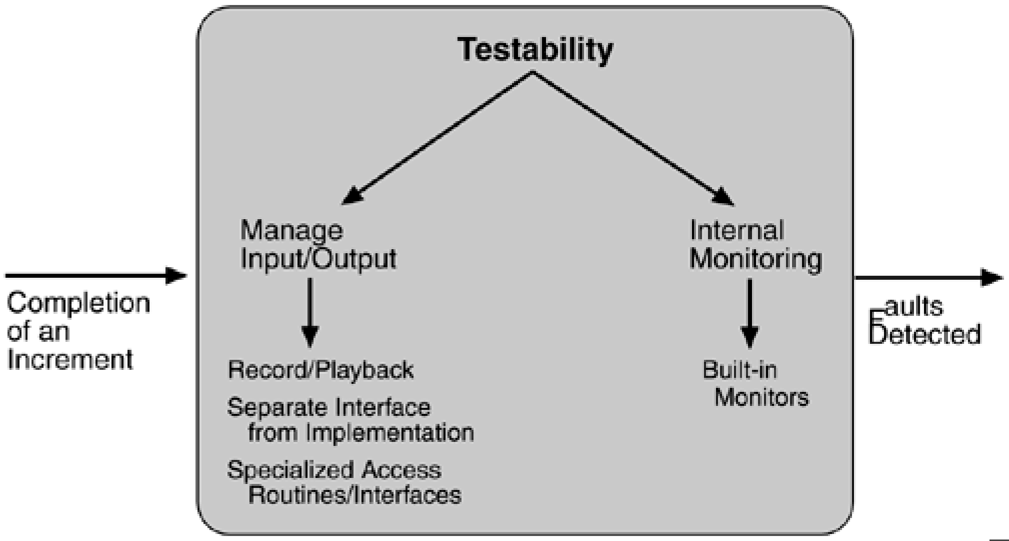

Testability
Manage Input/Output
Internal Monitoring
Completion of an Increment
Faults Detected
Record/Playback
Separate Interface from Implementation
Specialized Access Routines/Interfaces
Built-in Monitors

TABLE OF CONTENTS
◀ PREVIOUS NEXT ▶