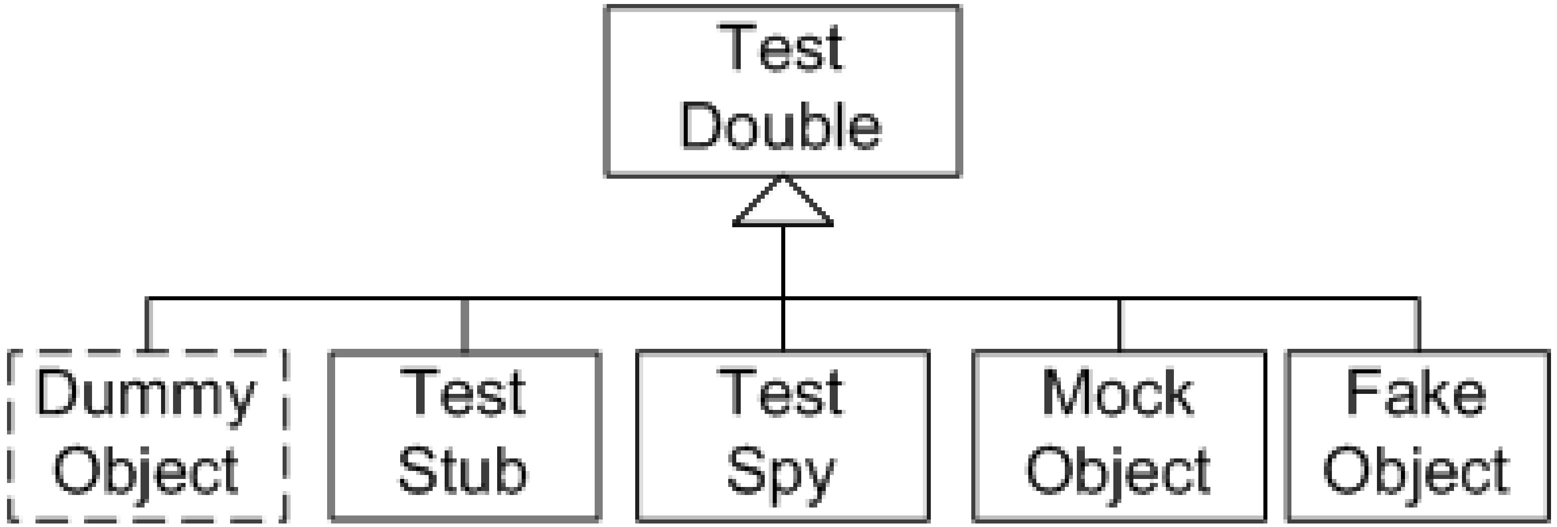
nagarro

Test Doubles

A Test Double is our Stuntman in an Automated Test

Types of Test Doubles



Dummy

- Object dependencies that are passed to the SUT as parameters which are of no interest to the SUT or test validation
  - Can be used in cases where the passed object is required by the SUT's interface but never called when exercising the SUT for a specific test case
- Can be null
- Used as placeholders only

- Example
  - A dummy can replace a database connection class that is never accessed by the SUT in the specific test case but is problematic as it tries to establish a connection to a database its constructor

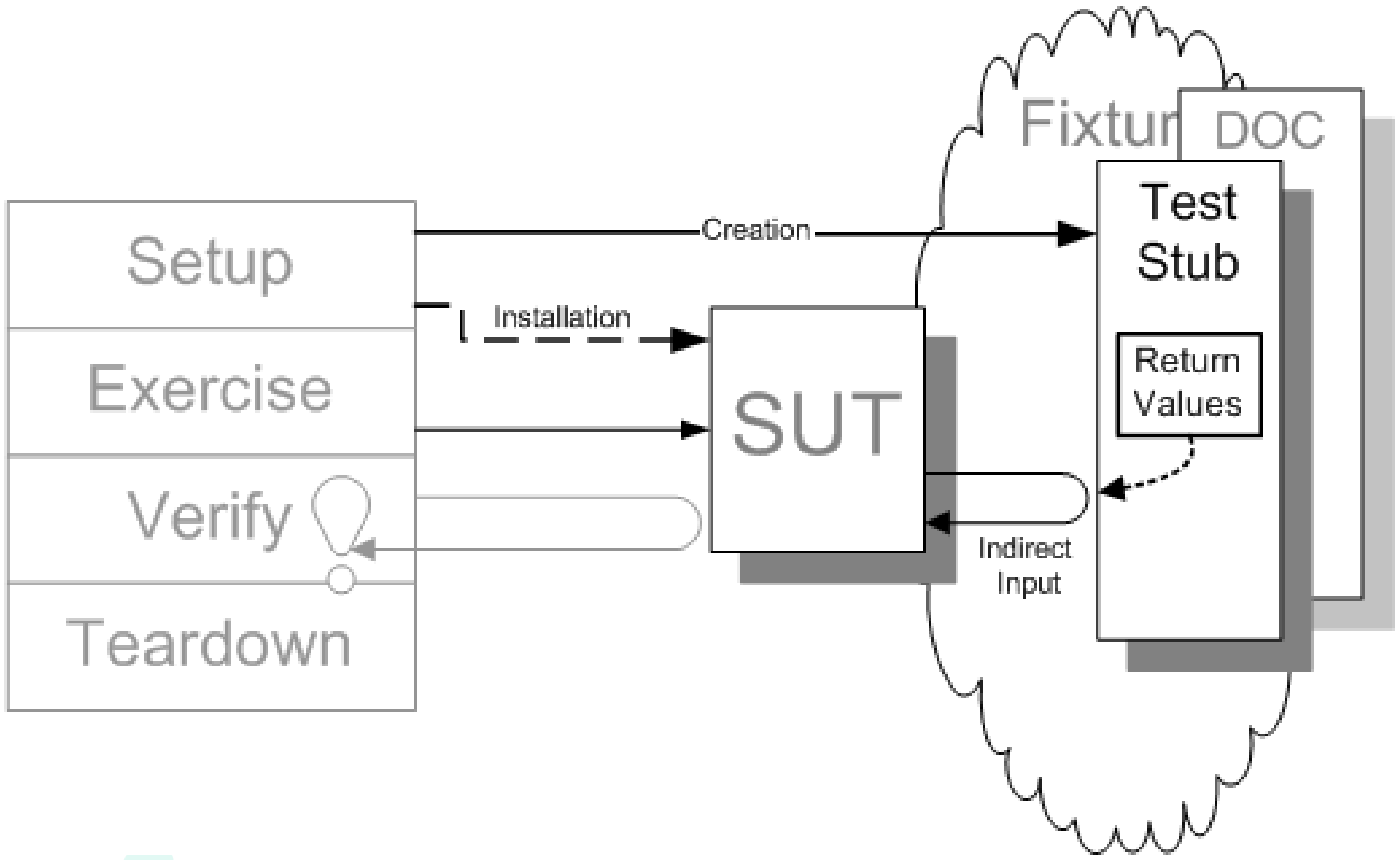
Stub

State oriented: Object that replaces real component used in SUT and provides indirect inputs to the SUT

- Usually it is not necessary to implement complete interface
- Can be used to return unexpected values in order to test SUT for error handling
- In early phases of the SUT a stub can be used as a temporary implementation until the real component is developed

- Example:
  - Replace dependency of the SUT that produces undeterministic outputs (such as a database entry or random value) with a stub
  - Instrument the stub to return a specific known value when a specific method is called to test a specific logic of the SUT

Stub



http://xunitpatterns.com/Test%20Stub.gif

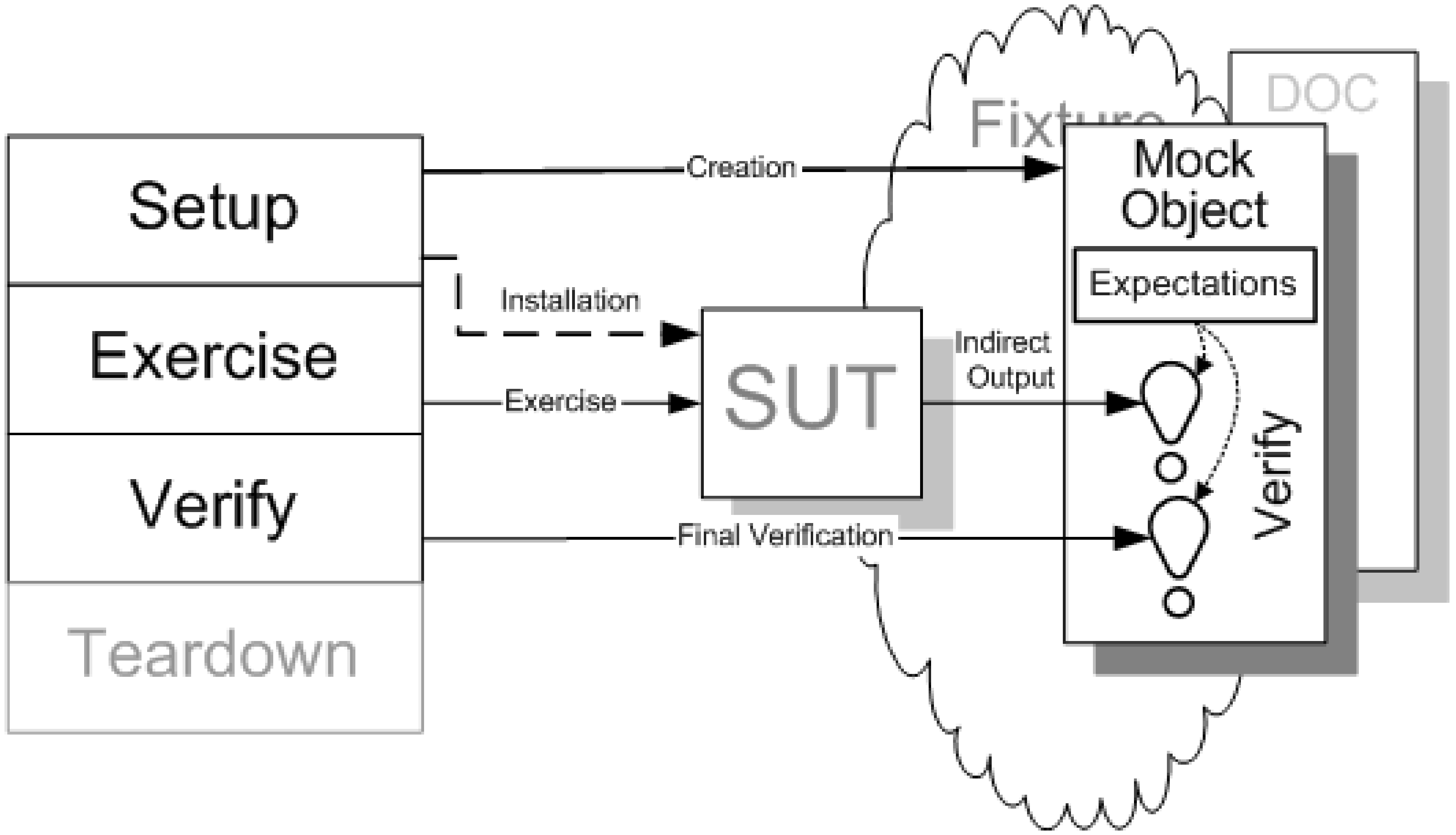
Mock

Behaviour oriented: verifies side effects of the SUT, or interactions from SUT

- Can be used only on predefined methods or properties, otherwise an exception occurs
- Has dynamic behaviour
- Using mocking frameworks is recommended
- Replaces a dependency of the SUT to make sure the SUT made the expected call to that “mocked” dependency

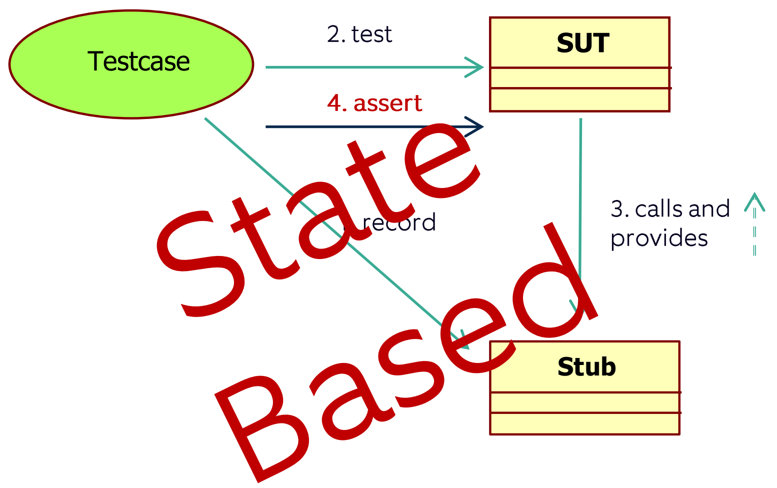
- Example
  - replace database, logging or networking class with problematic dependencies to infrastructure with a mock
  - Instrument the mock to expect a specific call with specific parameters from the SUT
  - Check that the call happened in the verification step

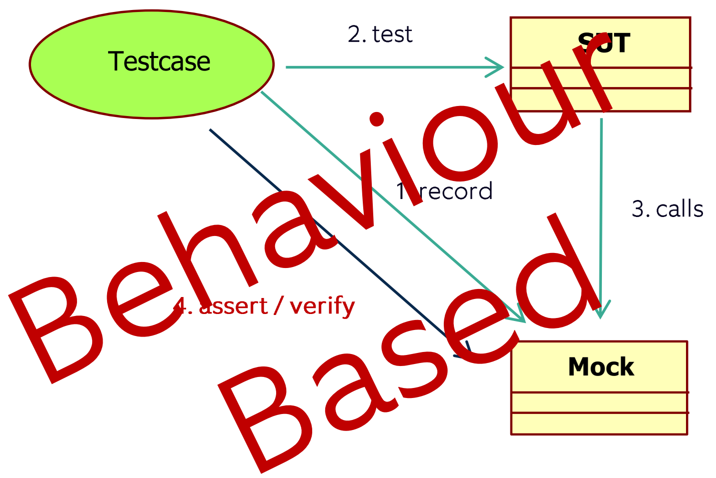
Mock



http://xunitpatterns.com/Mock%20Object.gif

Stub or Mock (both replace a dependency)





- Stub simulates the behaviour
- Provides data to drive the SUT

- Mock verifies the behavior
- Occured the call with the recorded signature?

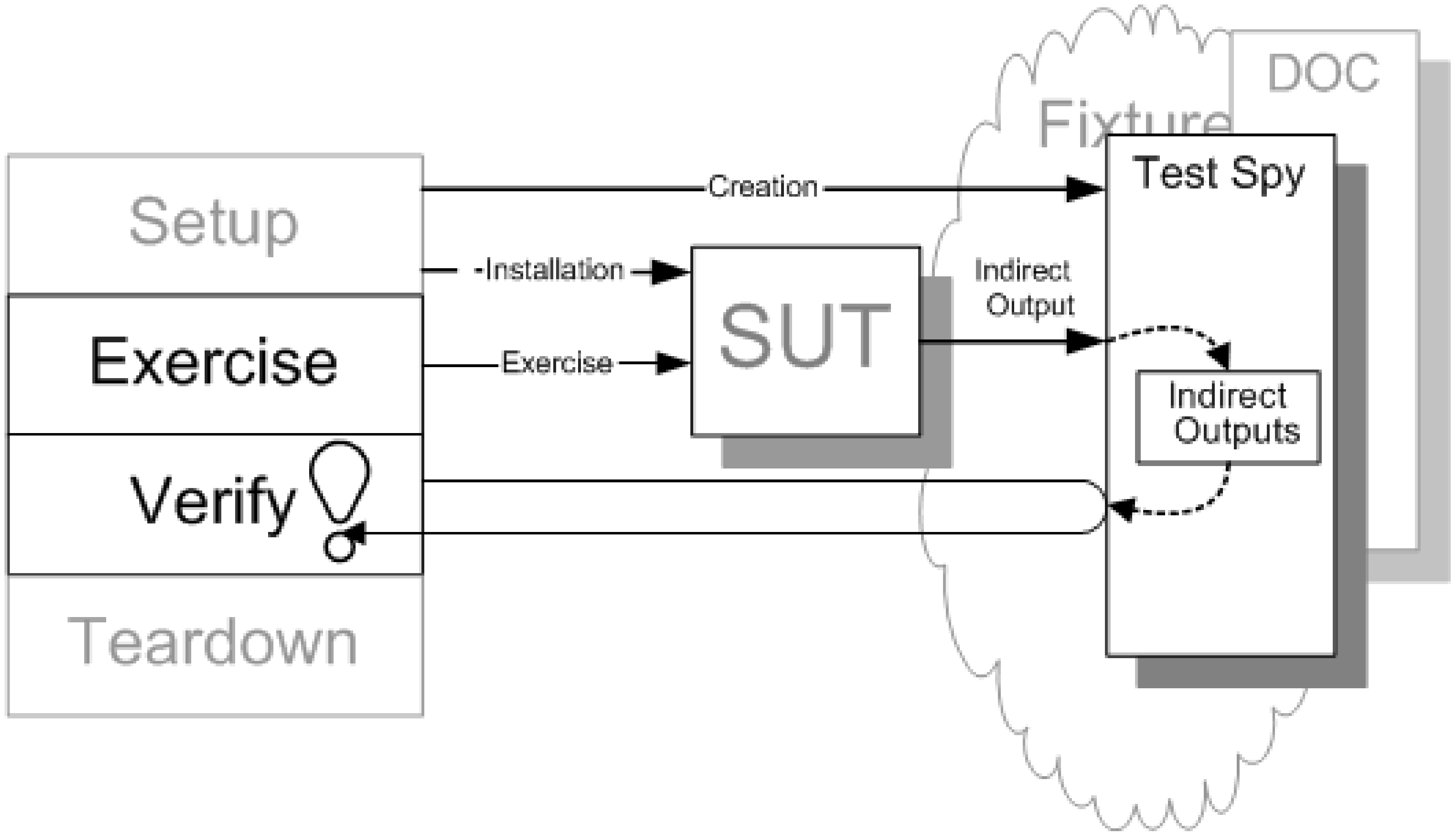
Test Spy

Captures indirect output calls made to another component by the SUT for later verification by the test

- Used when it is not possible to predict value of all attributes of the interaction
- Explicit assertion of recorded method calls reveals more intent of the test
- It does not fail on first deviation from expected behavior , therefore more information can be collected
- Implementation as anonymous inner class, by implementing retrieval interface or using registry for storing information

- Example
  - Audit log that is called by the SUT when exercised without producing direct outputs
  - The spy can replace the audit log and calls to it can be recorded and verified in the test case after exercising the SUT

Test Spy



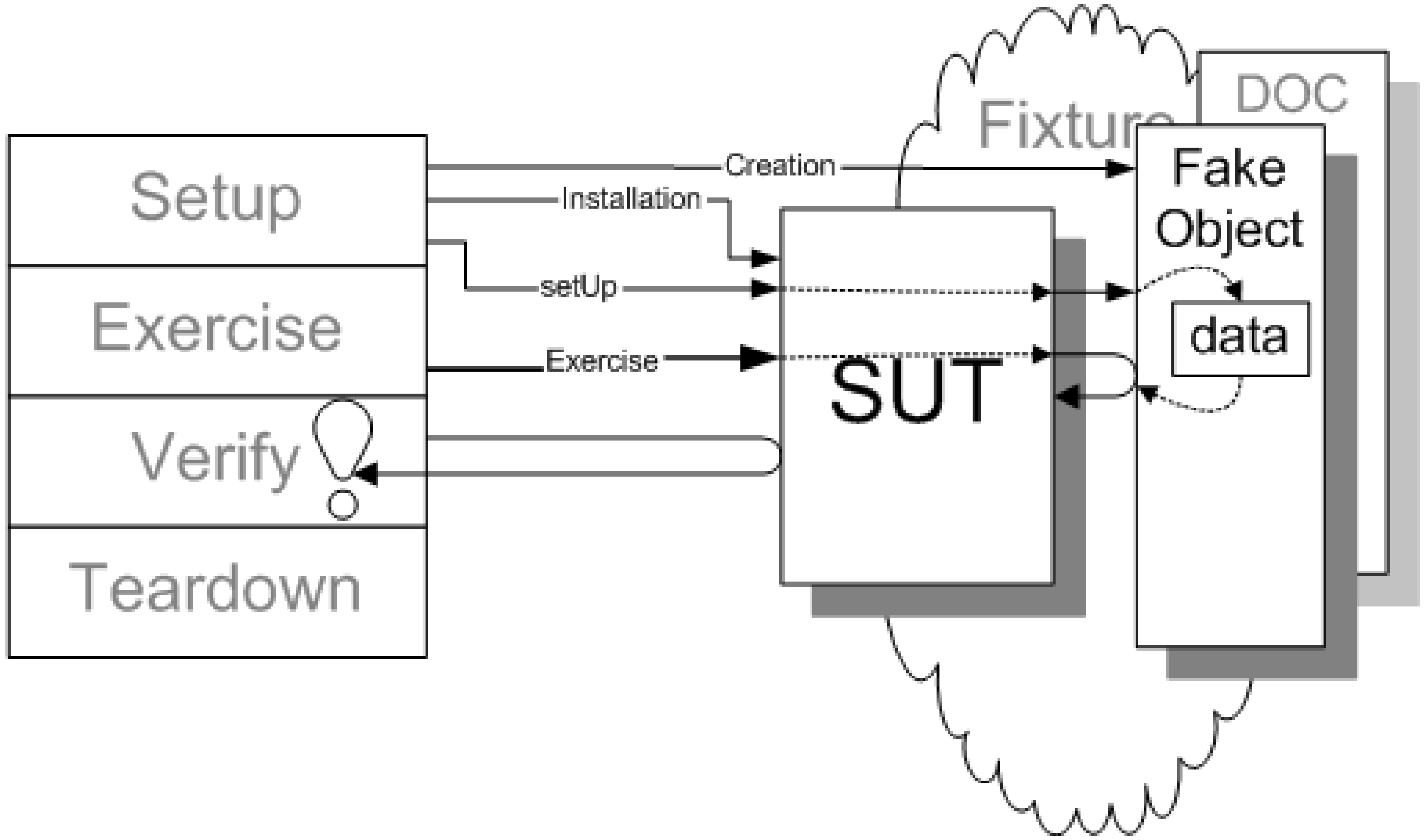
http://xunitpatterns.com/Test%20Spy.gif

Fake

- Simplified implementation of the dependent component, contains logic but is not suited for production
- Usually implements the whole interface but not used to verify inputs or outputs
- Usually used by multiple tests
- Example: In-memory test database

- **Usually used in integration tests**

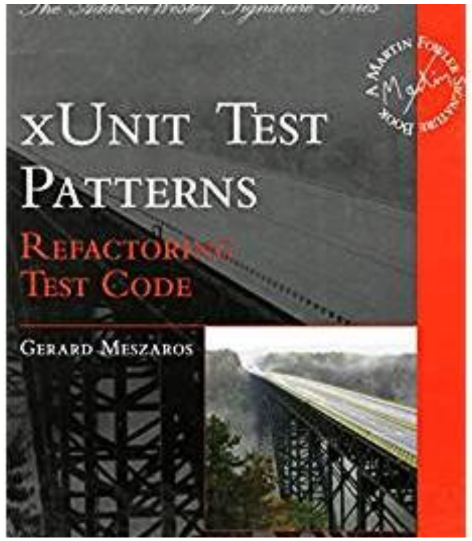
Fake



http://xunitpatterns.com/Fake%20Object.gif

Appendix: References

- xUnit Test Patterns: Refactoring Test Code by Gerard Meszaros
- xunitpatterns.com


The Addison-Wesley Signature Series
xUnit Test Patterns
Refactoring Test Code
Gerard Meszaros
A Martin Fowler Signature Book