

Agile Software Development

Produced
by

Eamonn de Leastar (edelestar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

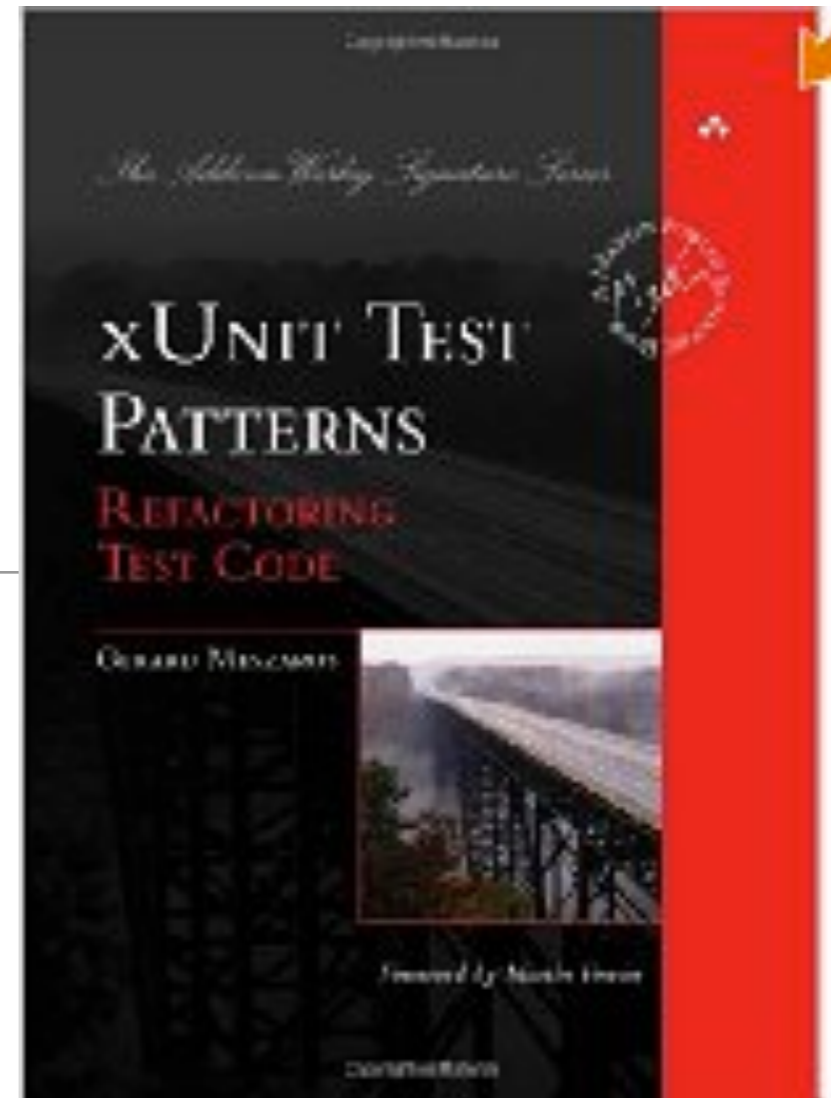
<http://elearning.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

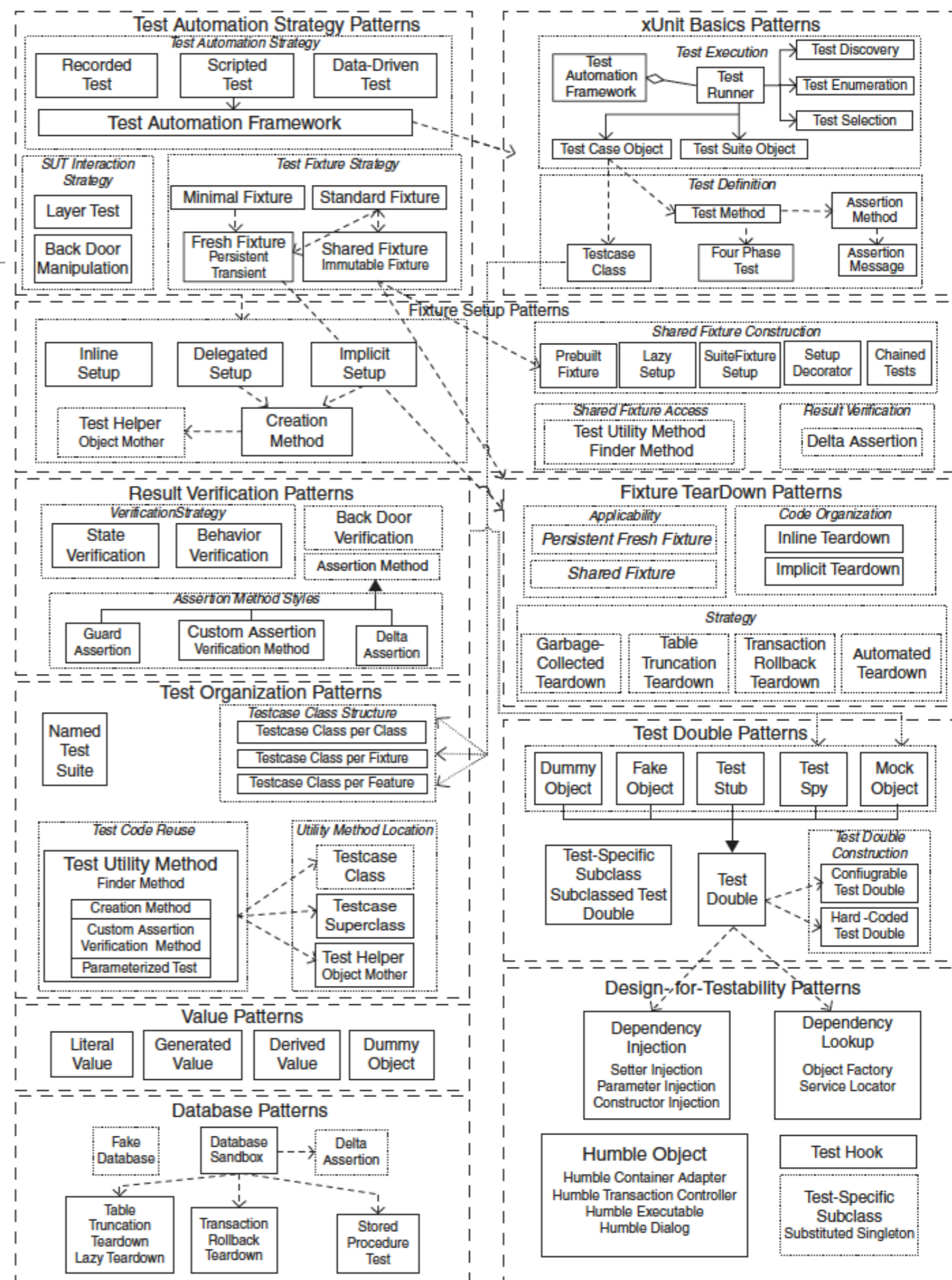


<http://xunitpatterns.com/>



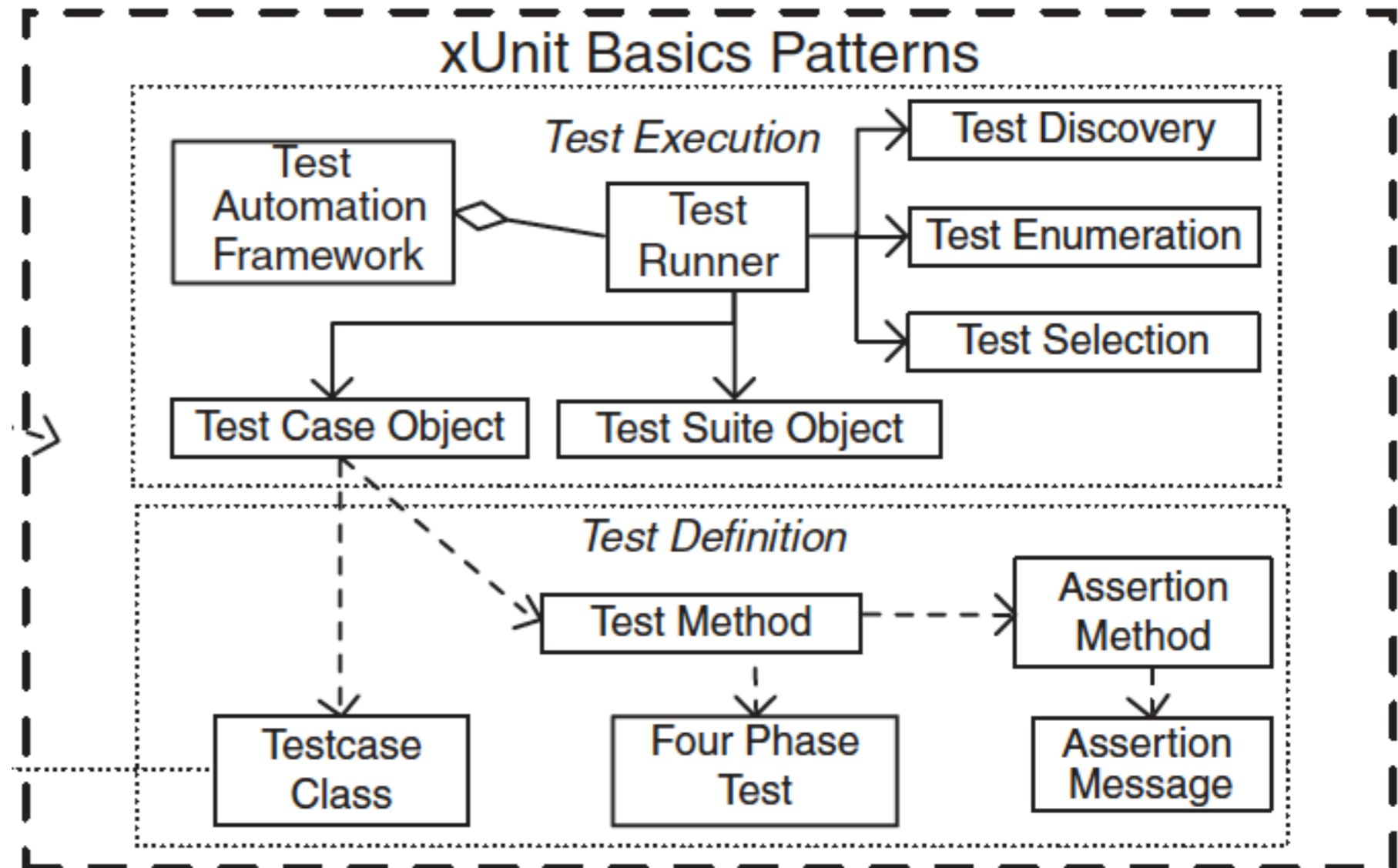
The Patterns

- Comprehensive and exhaustive catalog of test Patterns covering
 - Basics
 - Automation
 - Fixture setup & teardown
 - Result Verification
 - Organisational Structure
 - Database
 - Test Doubles
 - Design-for-Testability



Basics

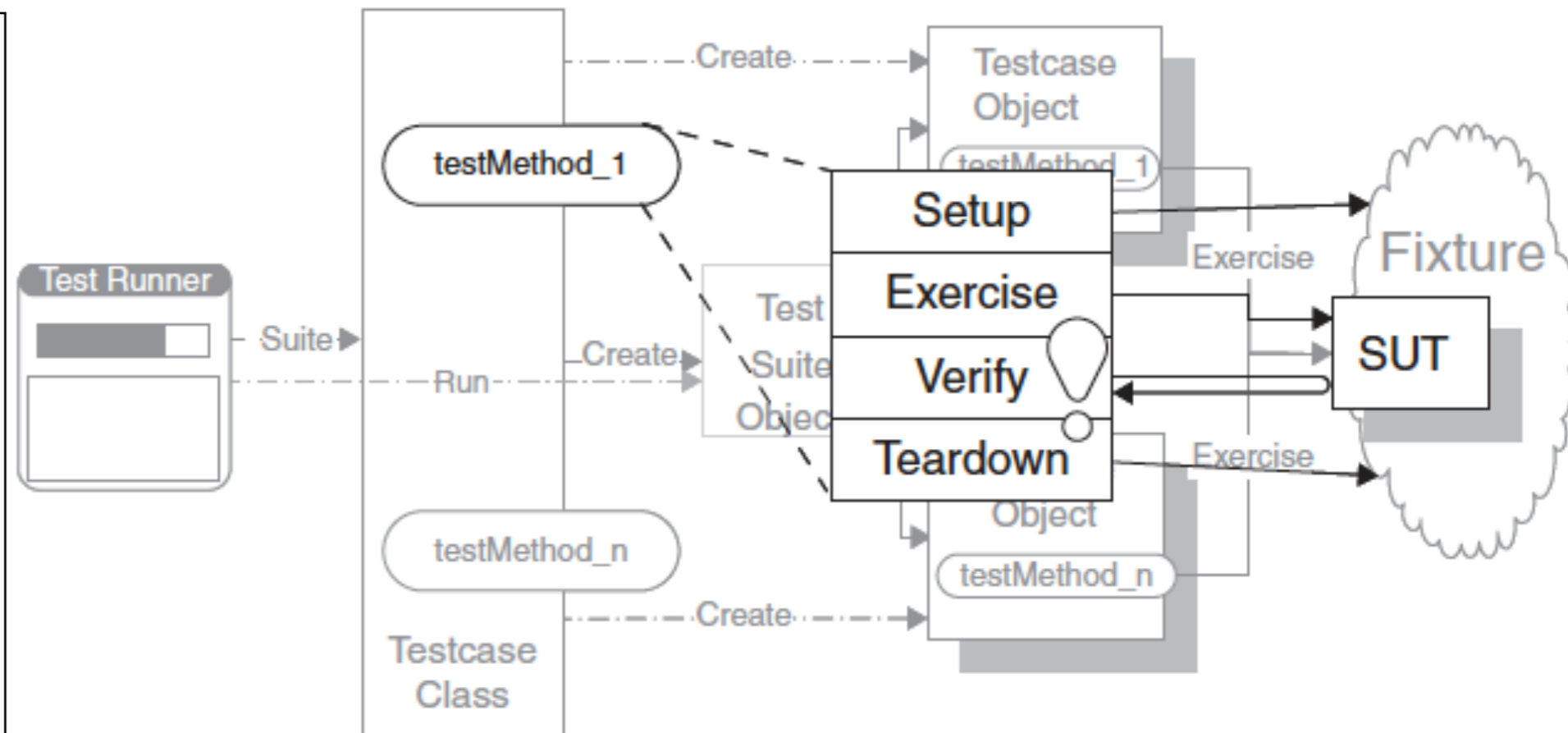
- Features of the xUnit framework
- Largely implemented by JUnit - and automatically integrated into:
 - IDE (Eclipse)
 - Build System (maven)



Four Phase Test

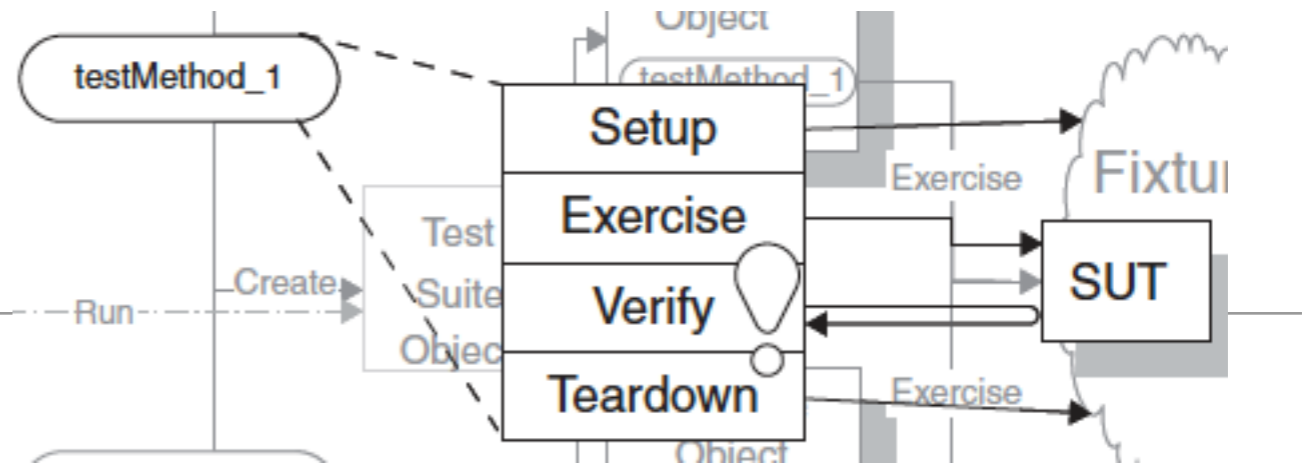
How do we structure our test logic to make what we are testing obvious?

We structure each test with four distinct parts executed in sequence.



SUT = System Under Test

How it works



Phase 1: Setup

- In the first phase, we set up the test fixture (the “before” picture) that is required for the SUT to exhibit the expected behavior as well as anything you need to put in place to be able to observe the actual outcome.

Phase 2: Exercise

- In the second phase, we interact with the SUT.

Phase 3: Verify

- In the third phase, we do whatever is necessary to determine whether the expected outcome has been obtained.

Phase 4: Teardown

- In the fourth phase, we tear down the test fixture to put the

4 Phase Example

```
@Test
public void testXMLSerializer() throws Exception {

    Serializer xmlSerializer = new XMLSerializer(new File("datastore.xml"));
    PacemakerAPI pacemaker = new PacemakerAPI(xmlSerializer);
    populate(pacemaker);

    pacemaker.store();

    PacemakerAPI pacemaker2 = new PacemakerAPI(null);
    pacemaker2.serializer = xmlSerializer;
    pacemaker2.load();
    pacemaker.getUsers()
        .forEach(user -> assertTrue(pacemaker2.getUsers().contains(user)));

    deleteFile("datastore.xml");

}
```

Example Phases



4 Phase Example

```
@Before
public void setup() {
    pacemaker = new PacemakerAPI(null);
    users.forEach(
        user -> pacemaker.createUser(user.firstName, user.lastName, user.email, user.password));
}

@Test
public void testAddActivityWithSingleLocation() {

    User marge = pacemaker.getUserByEmail("marge@simpson.com");
    Activity testActivity = margeActivities.get(0);
    String activityId = pacemaker.createActivity(marge.id, testActivity.type,
        testActivity.location, testActivity.distance, parseDateTime(testActivity.starttime),
        parseDuration(testActivity.duration)).id;

    pacemaker.addLocation(activityId, locations.get(0).latitude, locations.get(0).longitude);

    Activity activity = pacemaker.getActivity(activityId);
    assertEquals(1, activity.route.size());
    assertEquals(0.0001, locations.get(0).latitude, activity.route.get(0).latitude);
    assertEquals(0.0001, locations.get(0).longitude, activity.route.get(0).longitude);

}

@After
public void tearDown() {
    pacemaker = null;
}
```

4 Phase Example

*Phase 1:
Setup*

```
@Before
public void setup() {
    pacemaker = new PacemakerAPI(null);
    users.forEach(
        user -> pacemaker.createUser(user.firstName, user.lastName, user.email, user.password));
}
```

*Phase 2:
Exercise*

```
@Test
public void testAddActivityWithSingleLocation() {

    User marge = pacemaker.getUserByEmail("marge@simpson.com");
    Activity testActivity = margeActivities.get(0);
    String activityId = pacemaker.createActivity(marge.id, testActivity.type,
        testActivity.location, testActivity.distance, parseDateTime(testActivity.starttime),
        parseDuration(testActivity.duration)).id;

    pacemaker.addLocation(activityId, locations.get(0).latitude, locations.get(0).longitude);
```

*Phase 3:
Verify*

```
    Activity activity = pacemaker.getActivity(activityId);
    assertEquals(1, activity.route.size());
    assertEquals(0.0001, locations.get(0).latitude, activity.route.get(0).latitude);
    assertEquals(0.0001, locations.get(0).longitude, activity.route.get(0).longitude);
}
```

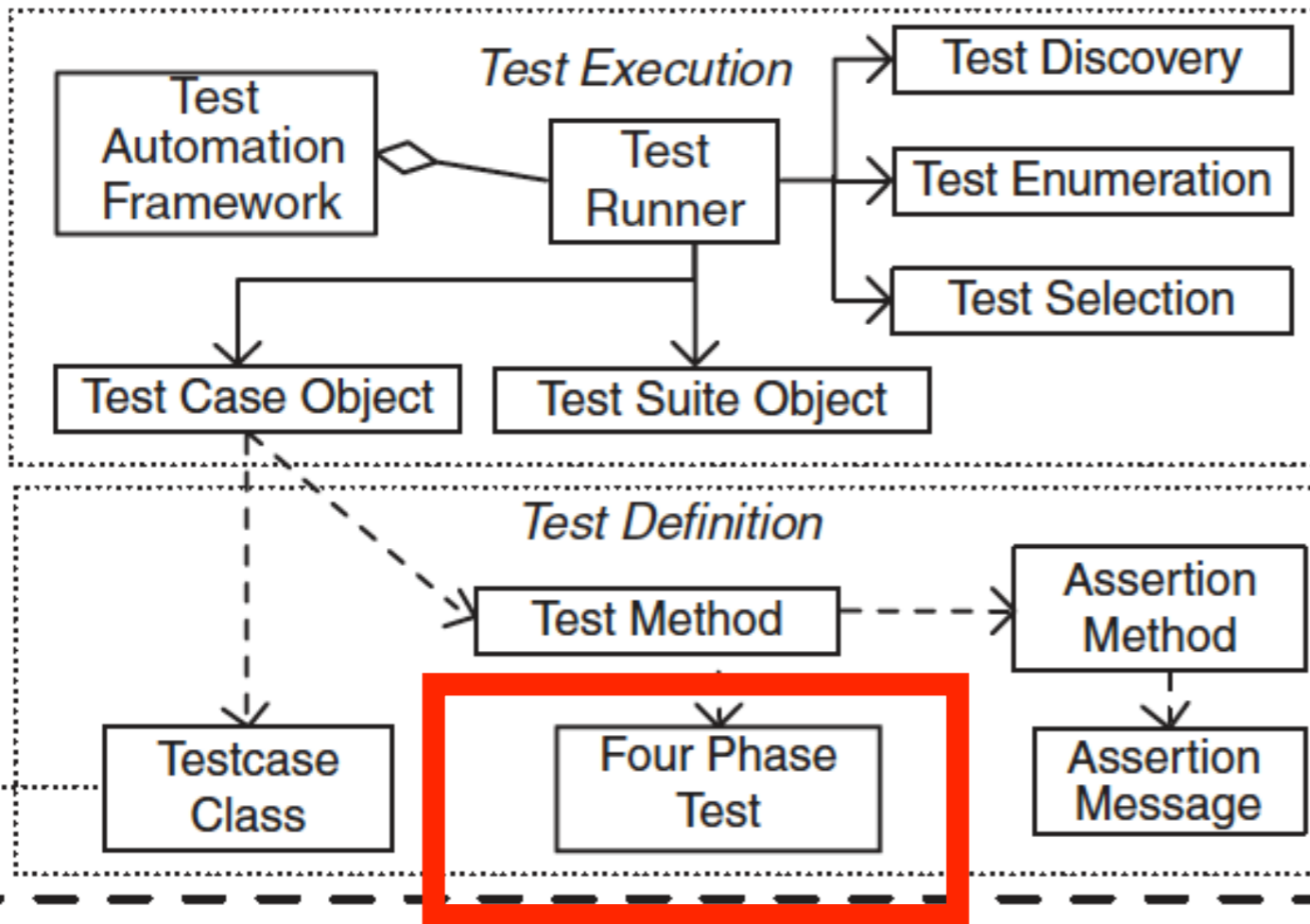
*Phase 4:
Teardown*

```
@After
public void tearDown() {
    pacemaker = null;
}
```

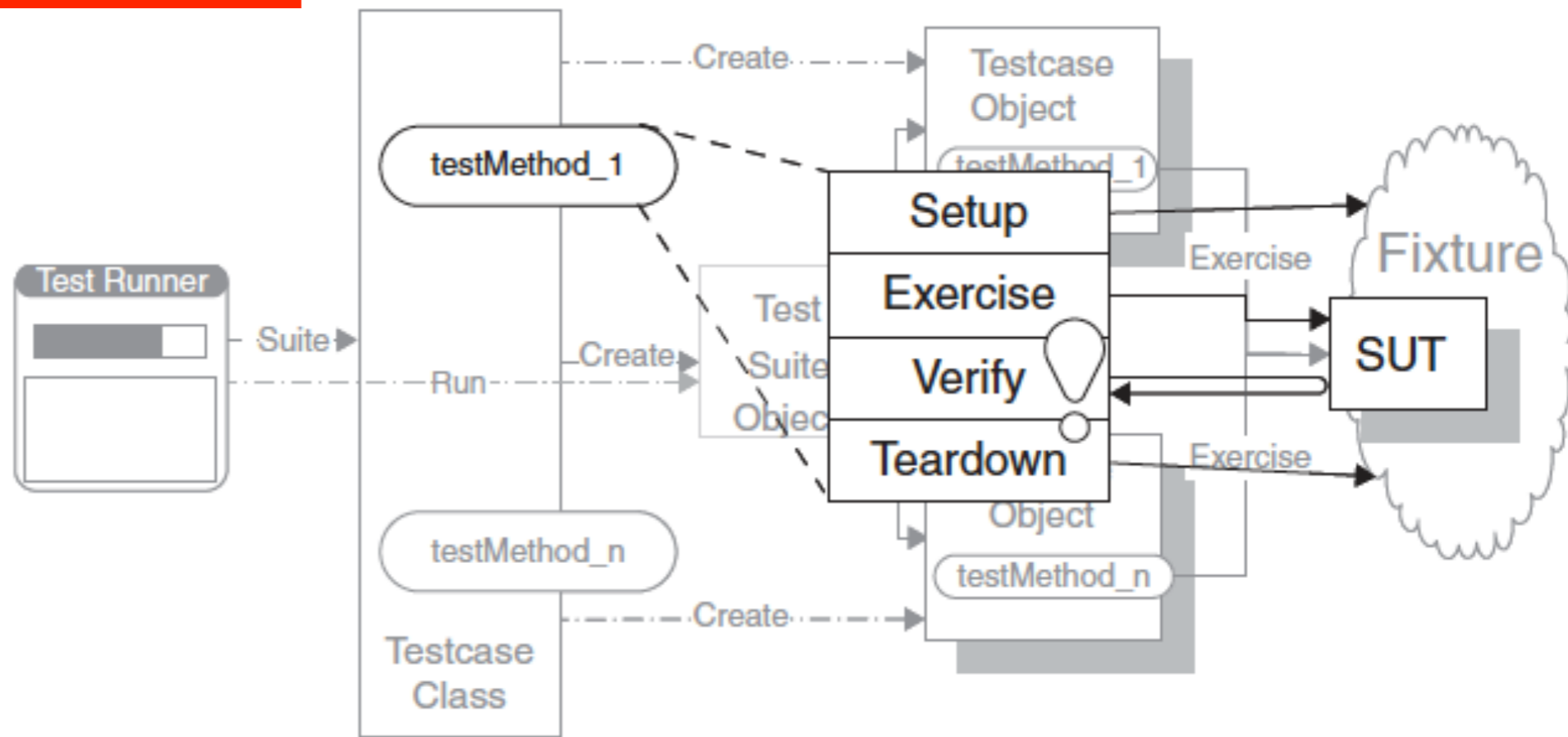
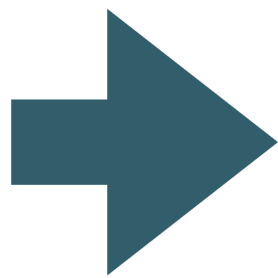
Why we do this

- The test reader must be able to quickly determine what behavior the test is verifying.
- It can be very confusing when various behaviors of the SUT are being invoked —some to set up the pre-test state (fixture) of the SUT, others to exercise the SUT, and yet others to verify the post-test state of the SUT.
- Clearly identifying the four phases makes the intent of the test much easier to see.
- Avoid the temptation to test as much functionality as possible in a single Test Method because that can result in Obscure Tests
- It is preferable to have many small Single-Condition Tests

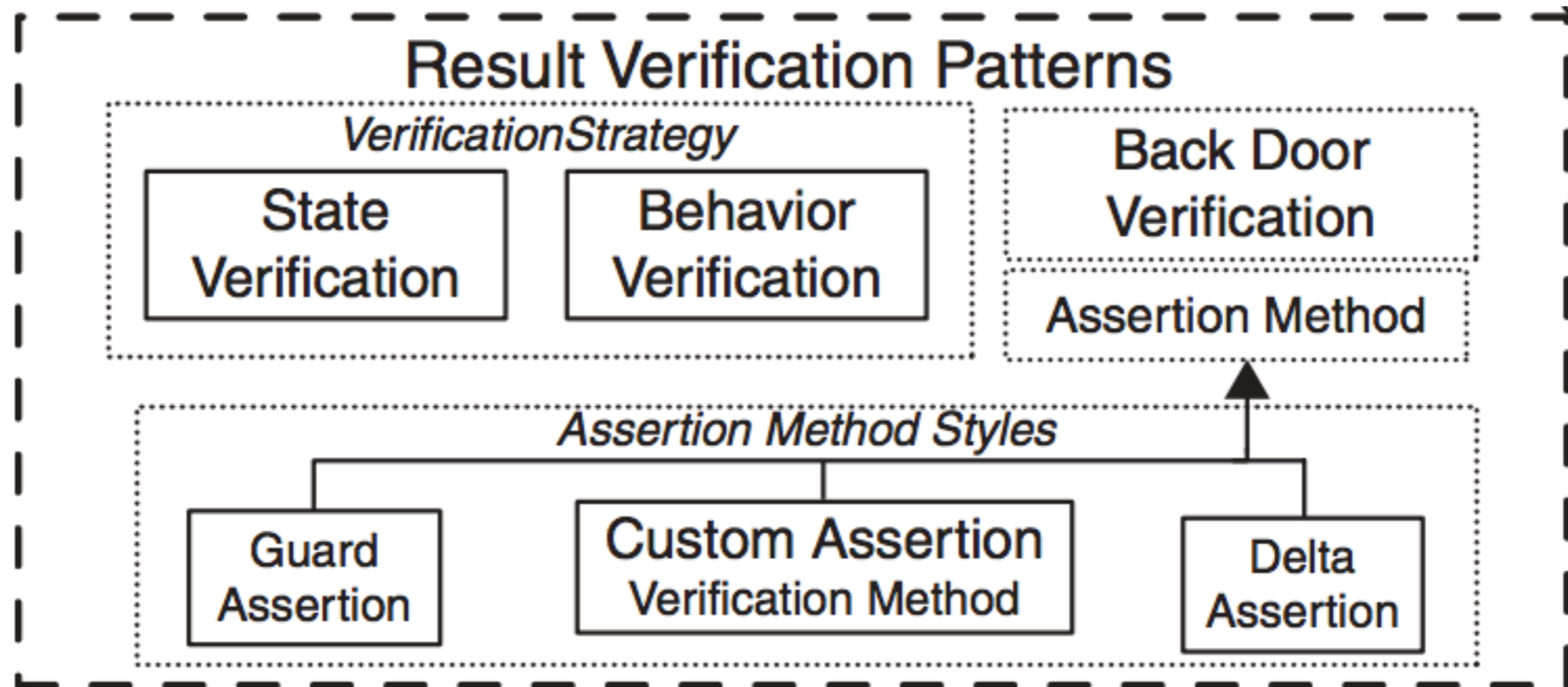
xUnit Basics Patterns



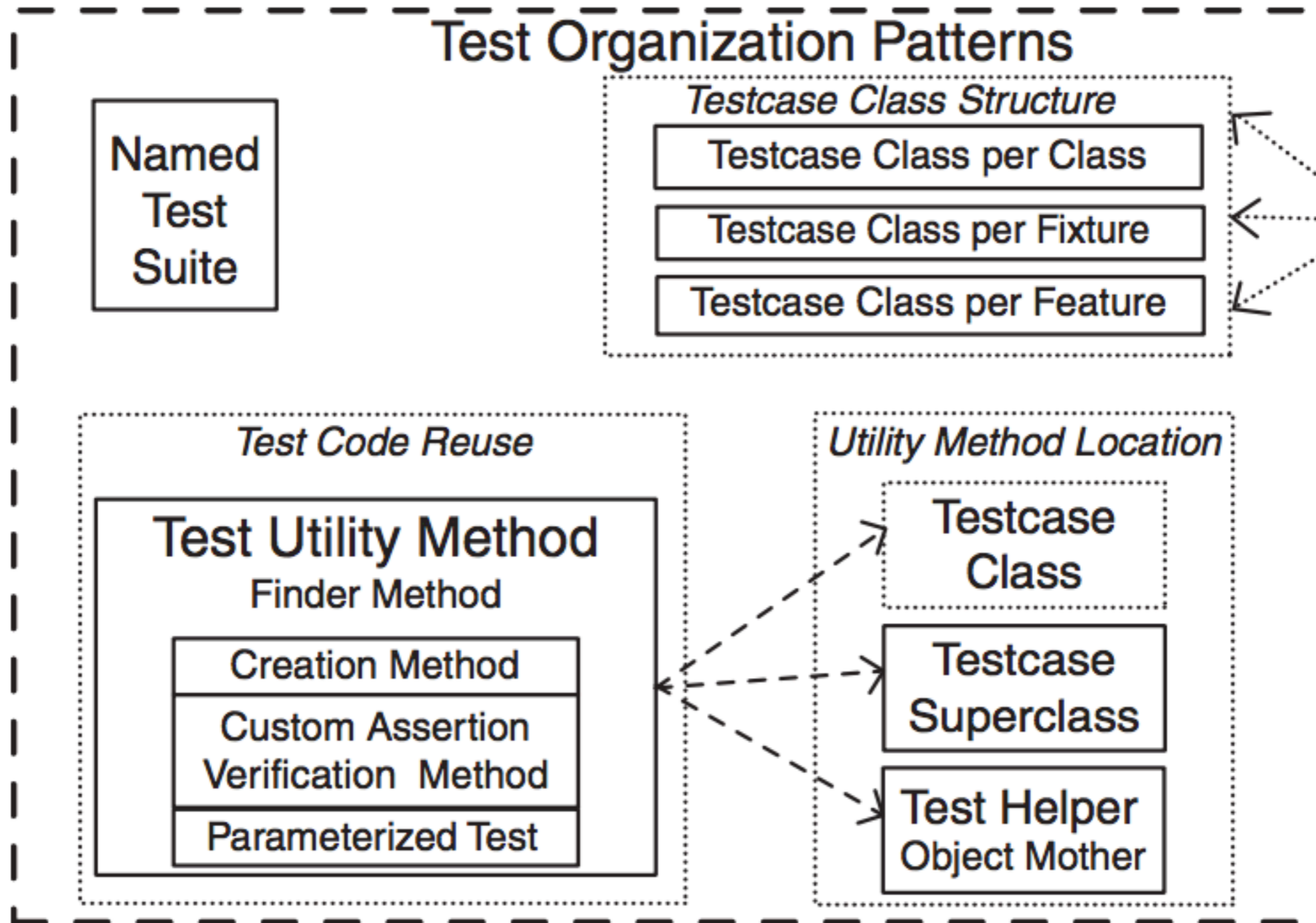
Four Phase Test



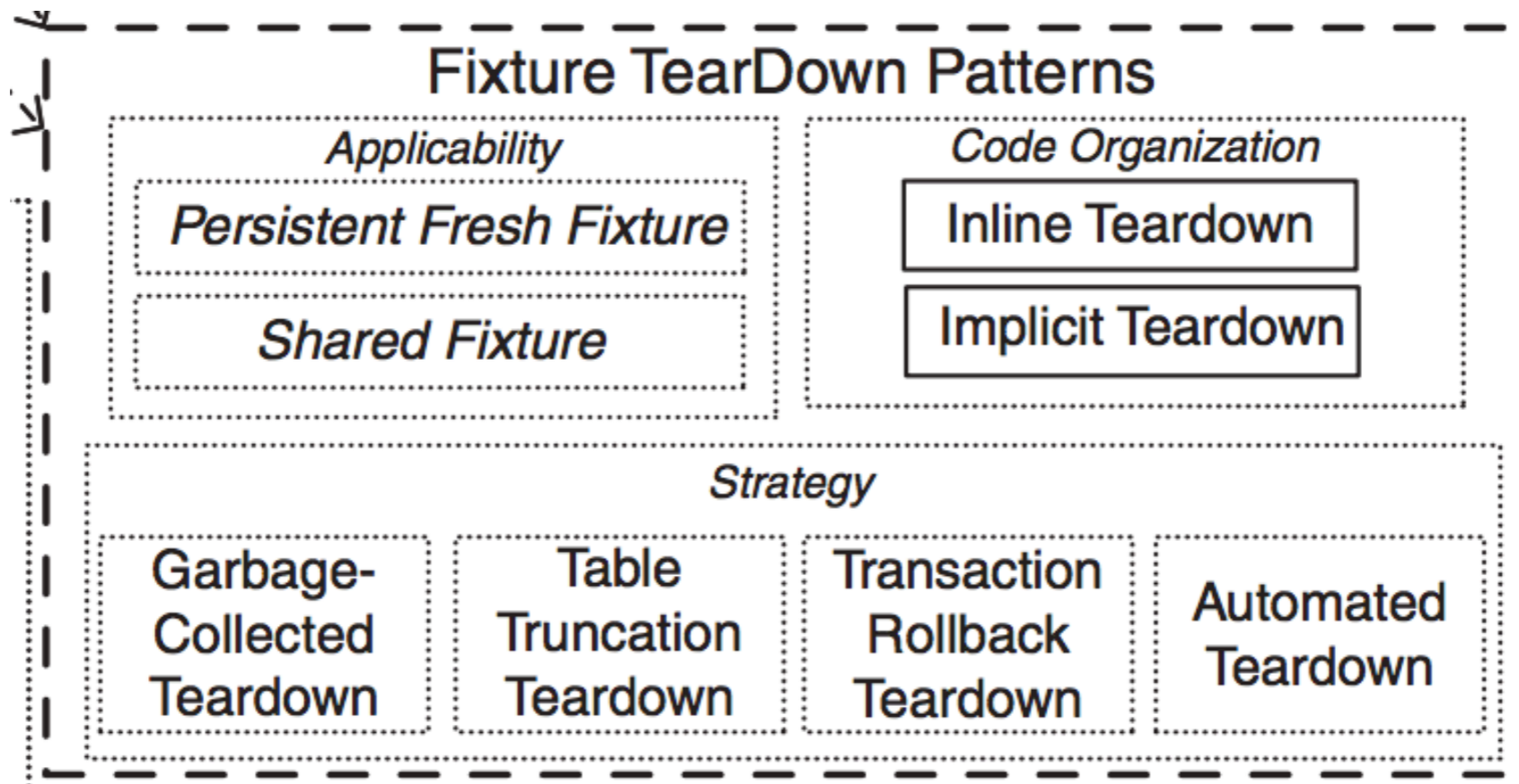
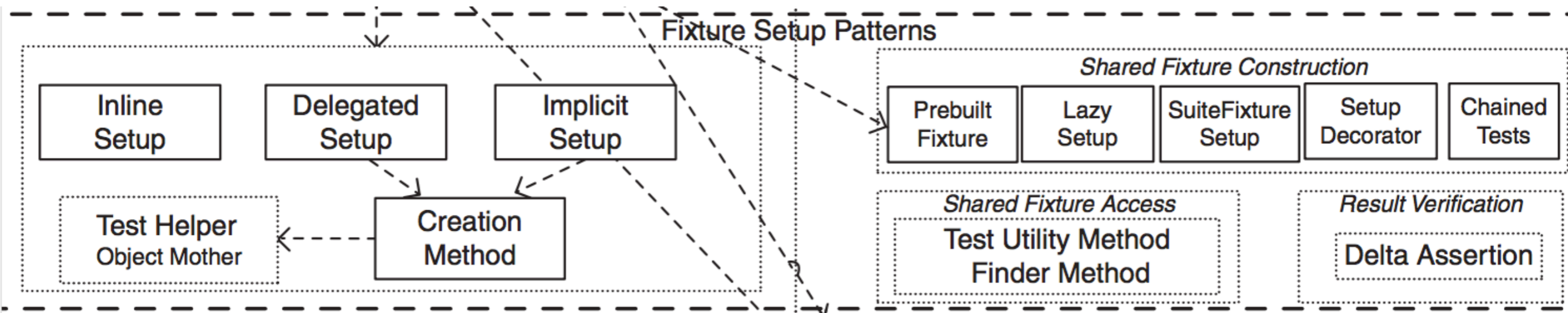
Other patterns... Result Verification



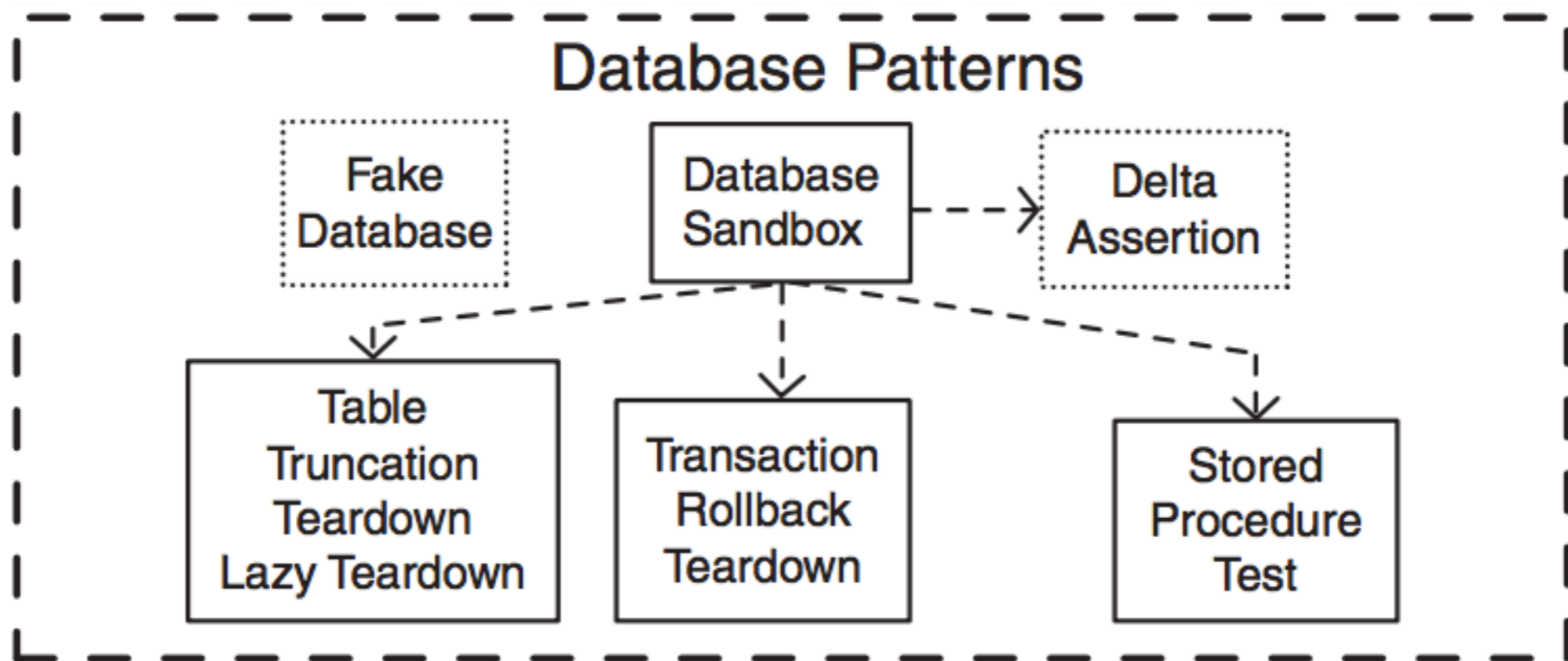
Other patterns... Test Organisation



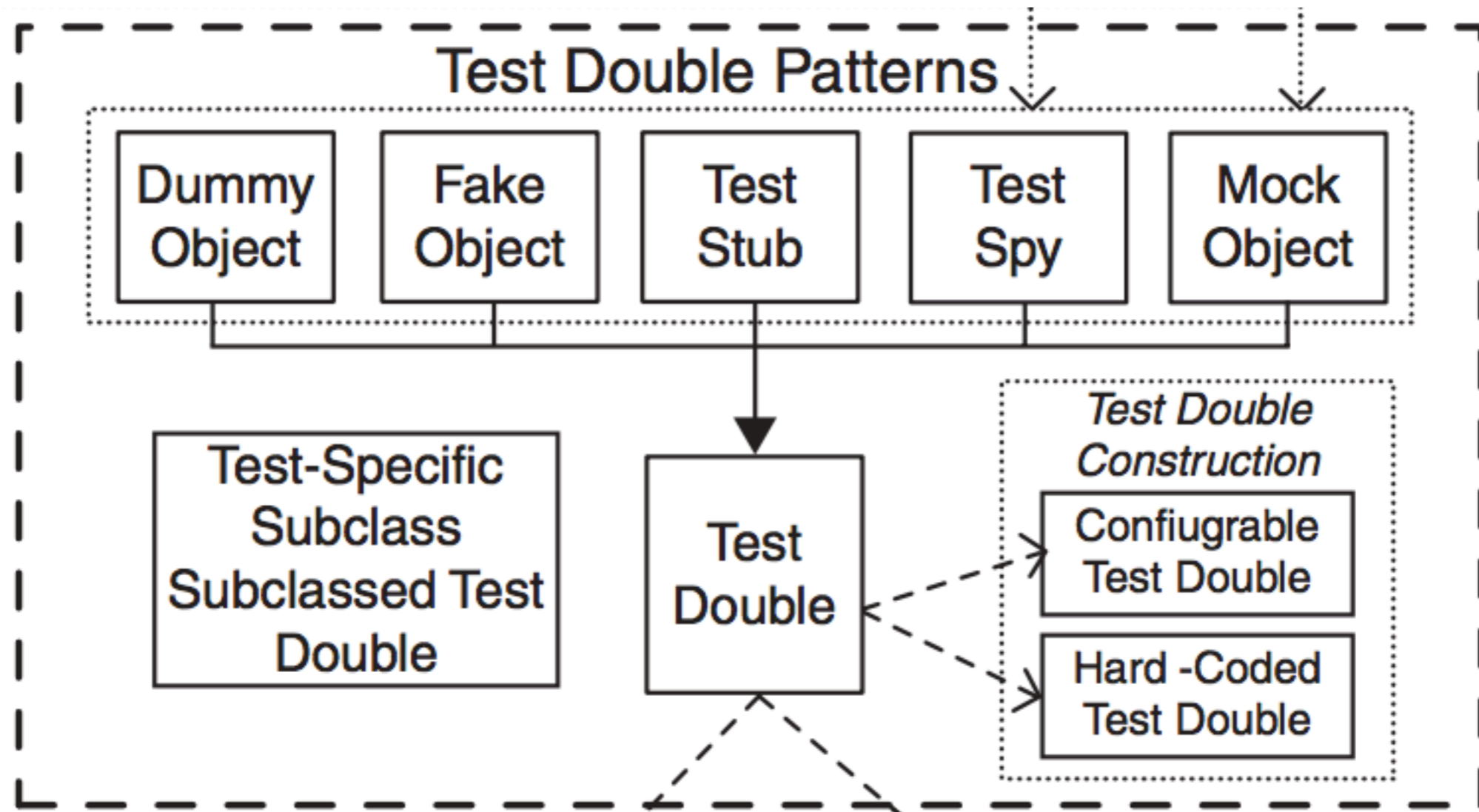
Other patterns... Fixtures



Other patterns... Database

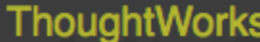




Other patterns... Test Doubles



<https://martinfowler.com/books/meszaros.html>

MARTINFOWLER.COM

[Intro](#) [Videos](#) [Design](#) [Agile](#) [Refactoring](#) [FAQ](#) [About Me](#) [All Sections](#)   

xUnit Test Patterns

Refactoring Test Code

by Gerard Meszaros

If you go to junit.org, you'll see a quote from me: "never in the field of software development have so many owed so much to so few lines of code". JUnit has been criticized as a minor thing, something any reasonable programmer could produce in a weekend. This is true, but utterly misses the point. The reason JUnit is important, and deserves the Churchillian knock-off, is that the presence of this tiny tool has been essential to a fundamental shift for many programmers. A shift where testing has moved to a front and central part of programming. People have advocated it before, but JUnit made it happen more than anything else.

It's more than just JUnit, of course. Ports of JUnit have been written for lots of programming languages. This loose family of tools, often referred to as xUnit tools, have spread their way far beyond the java roots. (And of course the roots weren't really in Java, as Kent Beck wrote this code for Smalltalk years before.)

XUnit tools, and more importantly the philosophy, offer up a huge opportunity to programming teams. An opportunity to write powerful regression test suites that enable teams to make drastic changes to

See on

amazon

informIT

