Testing …

In the real world

Steve Hostettler

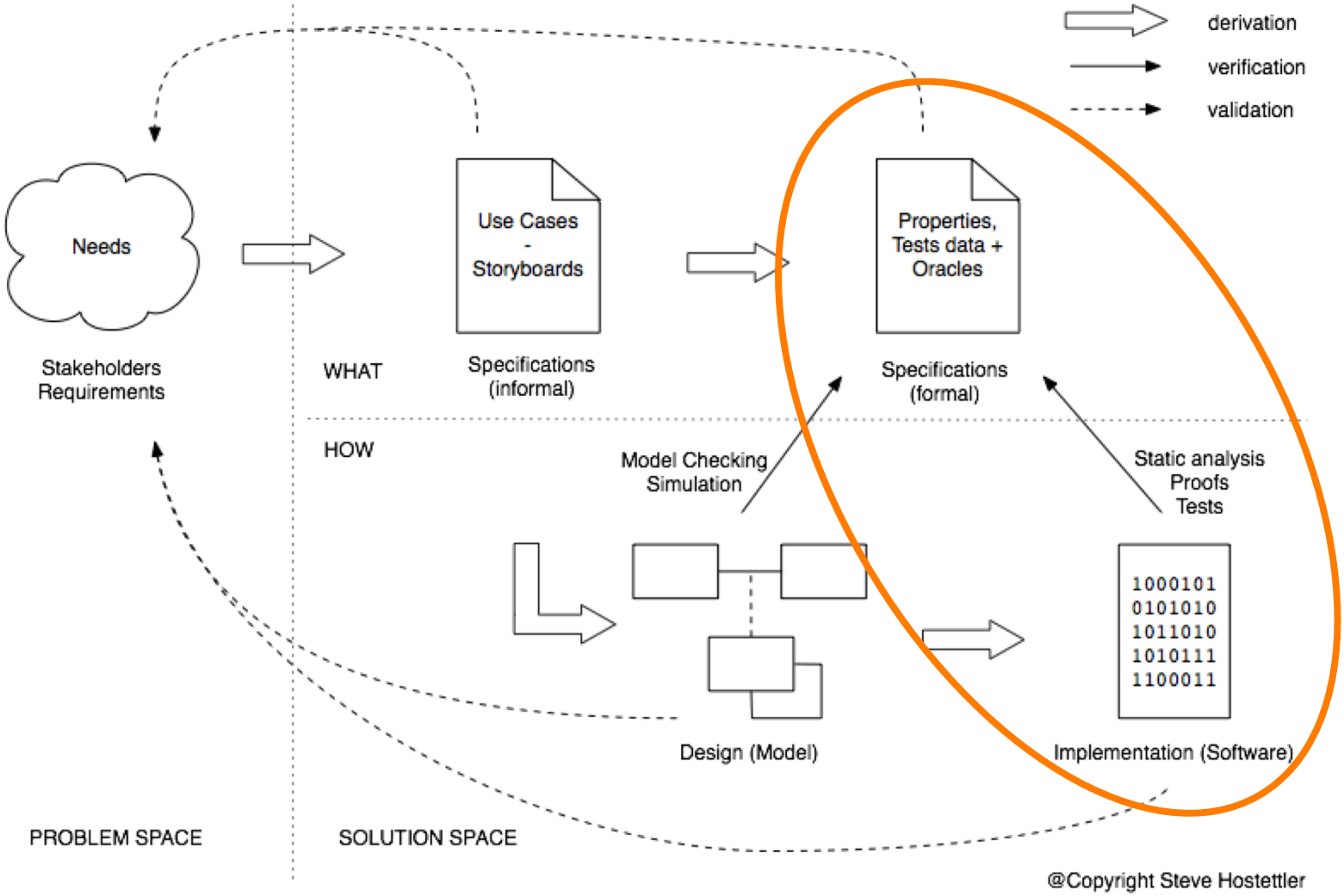
Software Modeling and Verification Group

University of Geneva

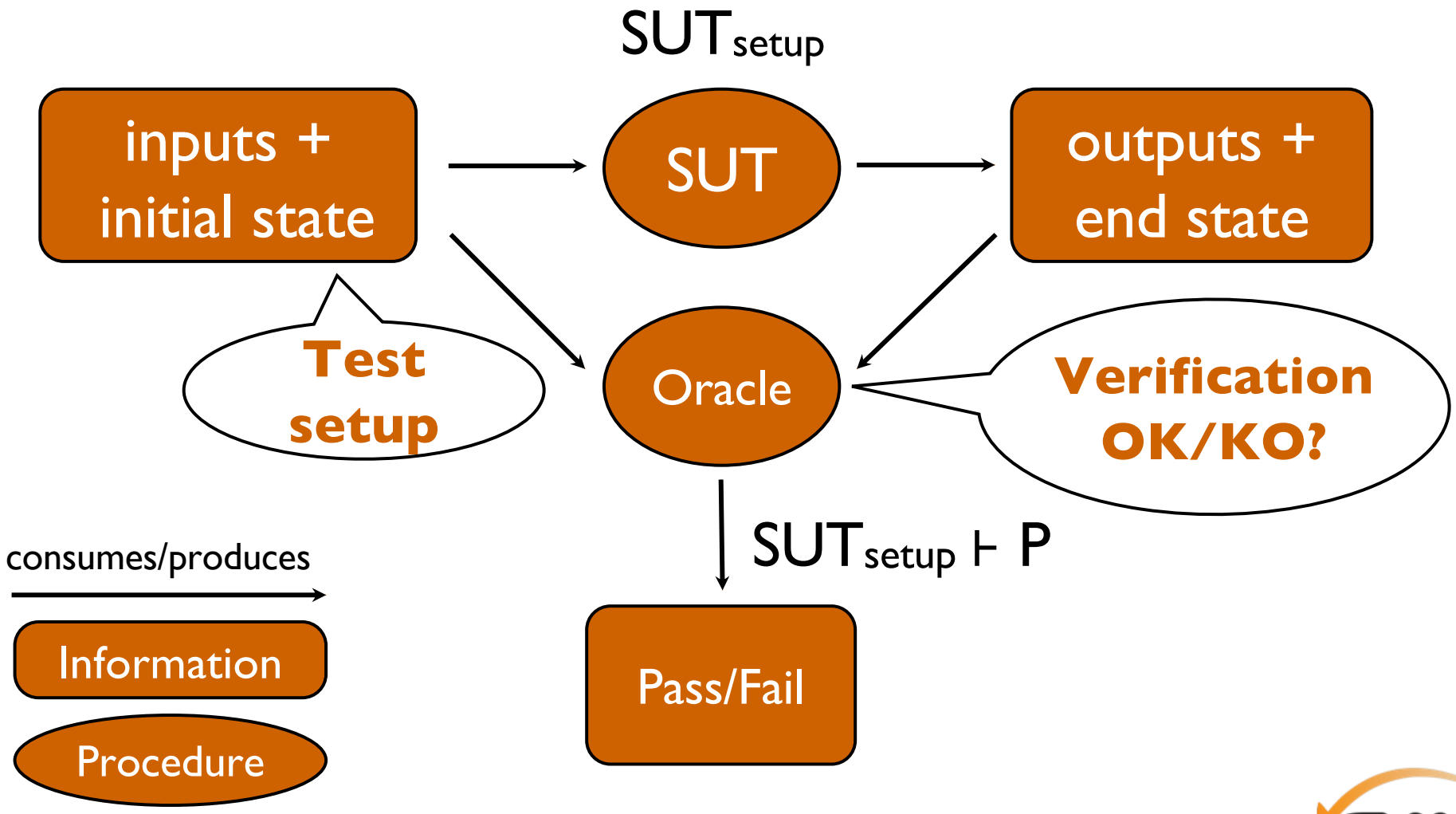
Unit Testing

A short reminder

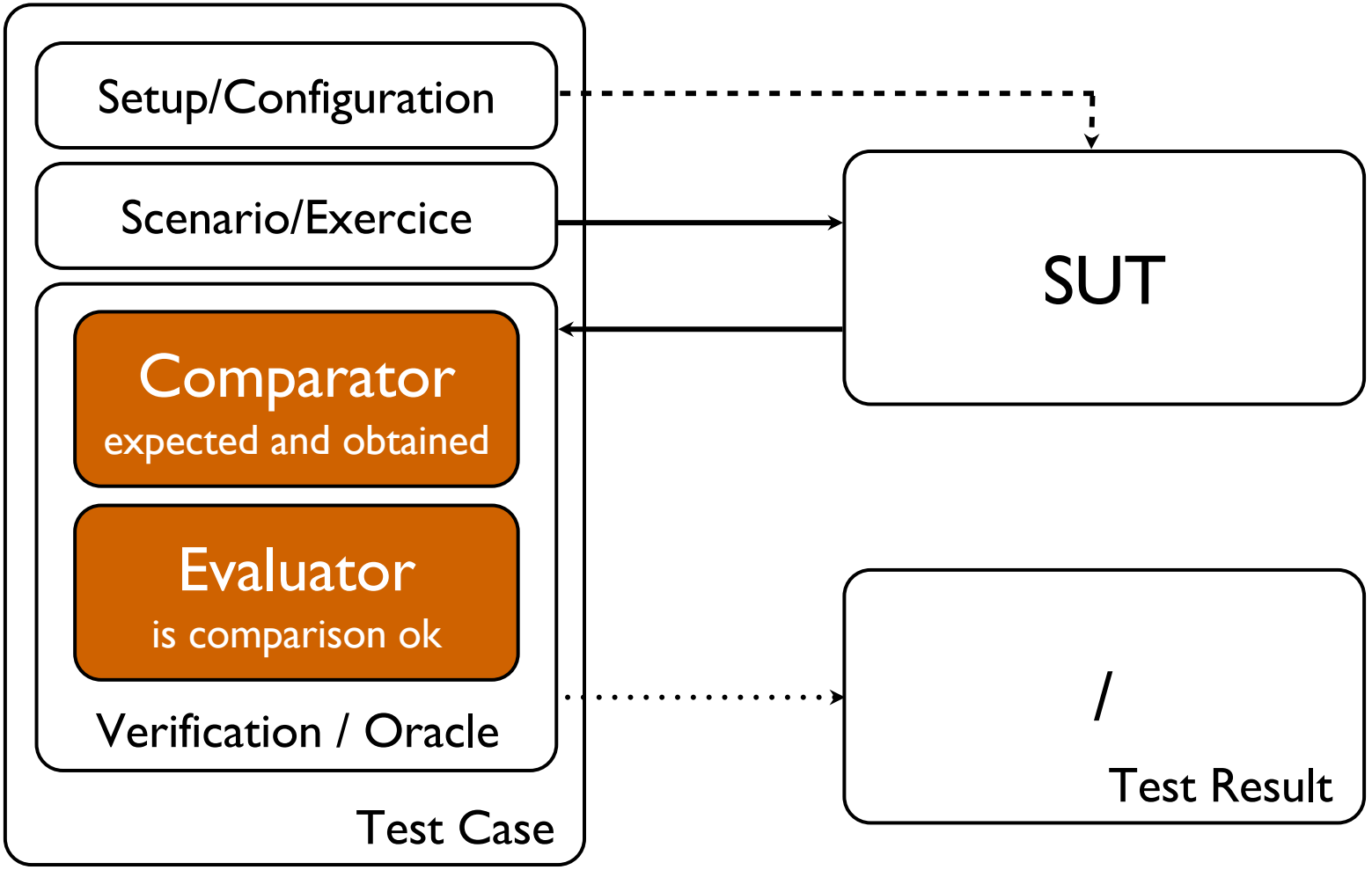
Validation & Verification



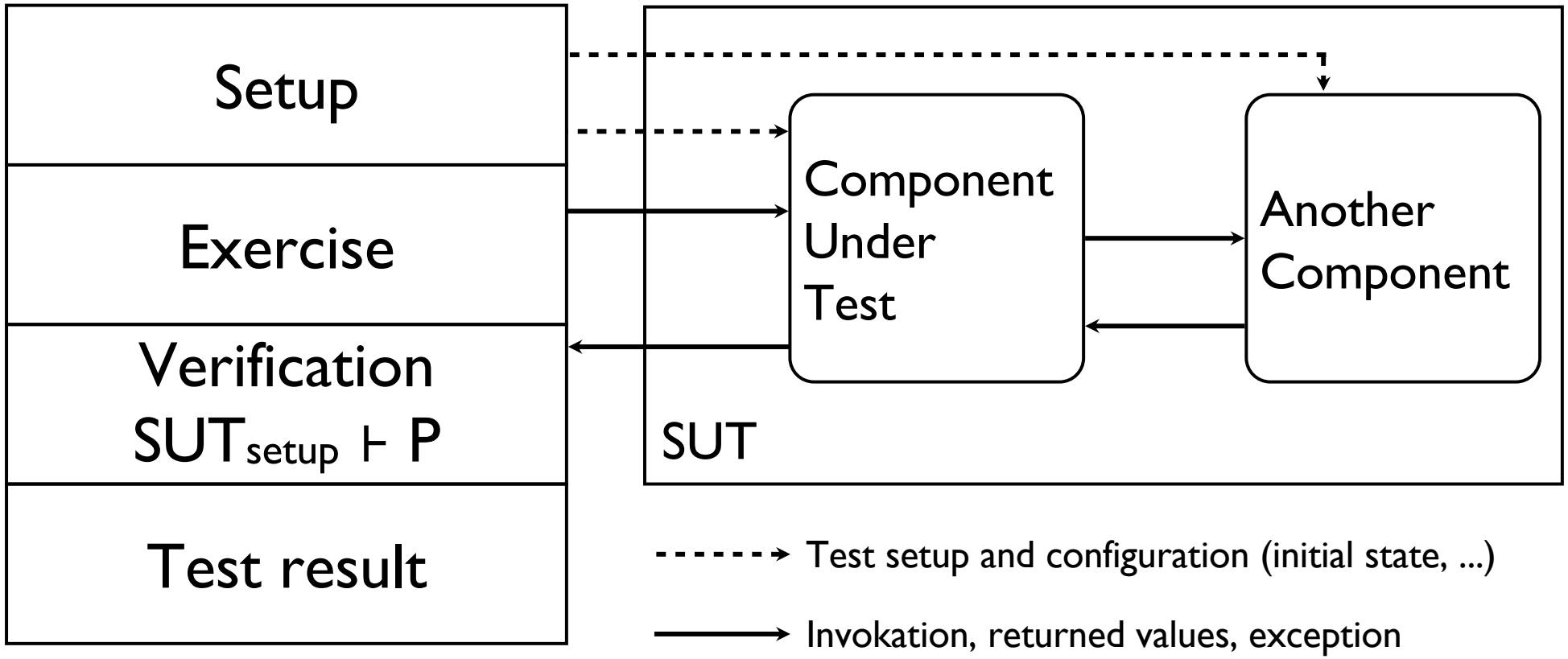
Anatomy of a test



Test Case


Setup/Configuration
Scenario/Exercice
Comparator
expected and obtained
Evaluator
is comparison ok
Verification / Oracle
Test Case
SUT
/
Test Result

Anatomy of a test



Test Levels

Components Testing

Integration Testing

System Testing

Acceptance Testing

Component Testing

a.k.a. unit testing

Component Testing

a.k.a. Unit Testing, Module Testing

Test basis: Component requirements, detailled design, code

Test objects: Components (classes, package, programs, database scripts, ...)

Component Testing

In isolation

Uses mock, stubs, fakes

Tests functional and non-functional requirements

By programmers (low management overhead)

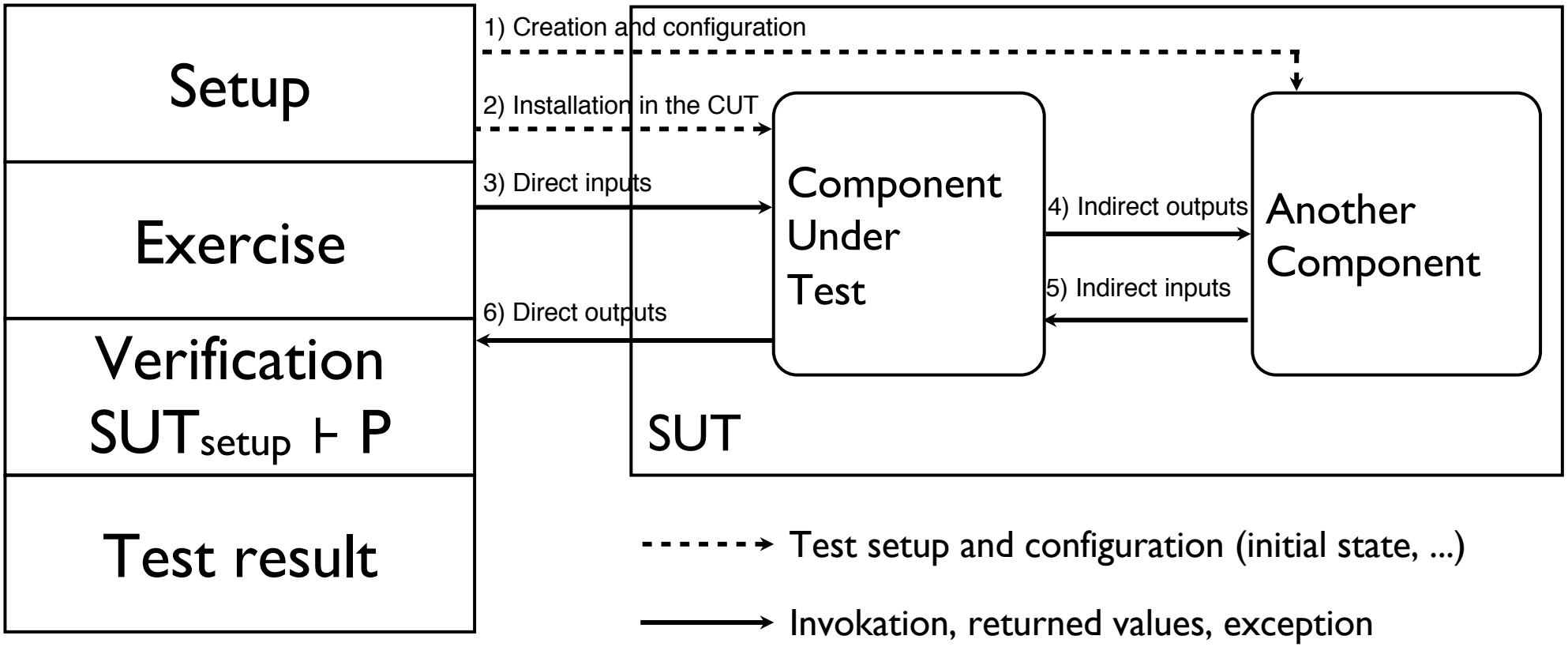
Test doubles

Helps to improve test isolation

Otherwise it would not be a component (unit) test

Many kind of doubles for many different usages

Test doubles



Example

```java
package ch.demo.business.service;
public interface AnotherComponent {
  Integer inc(Integer param);
}
```

```java
package ch.demo.business.service;
public class CUTImpl implements CUT {
 AnotherComponent component;
    public CUTImpl(AnotherComponent c) {
        this.component = c;
    }
  @Override
  public String doBusiness(String param, Integer delta) {
      return component.inc(Integer.valueOf(param)).toString();
  }
}
```

```java
package ch.demo.business.service;
public interface CUT {
  public String doBusiness(String string,
Integer delta);
}
```

Example

```
package ch.demo.business.service;

public class AnotherComponentImpl implements AnotherComponent {

    public Integer inc(Integer param) {

        if (param == null) {

            throw new IllegalArgumentException("Param must be not null!");

        } else if (param == Integer.MAX_INTEGER) {

            throw new IllegalStateException("Incrementing MAX_INTEGER will result in overflow!");

        } else {

            return param + 1;

        }

    }

}
```

```
package ch.demo.business.service;

public class CUTTest {

    public void testInc() {

        Assert.assertEquals("inc(3) != 4", 4, new CUTImpl(new AnotherComponent

    }

}
```

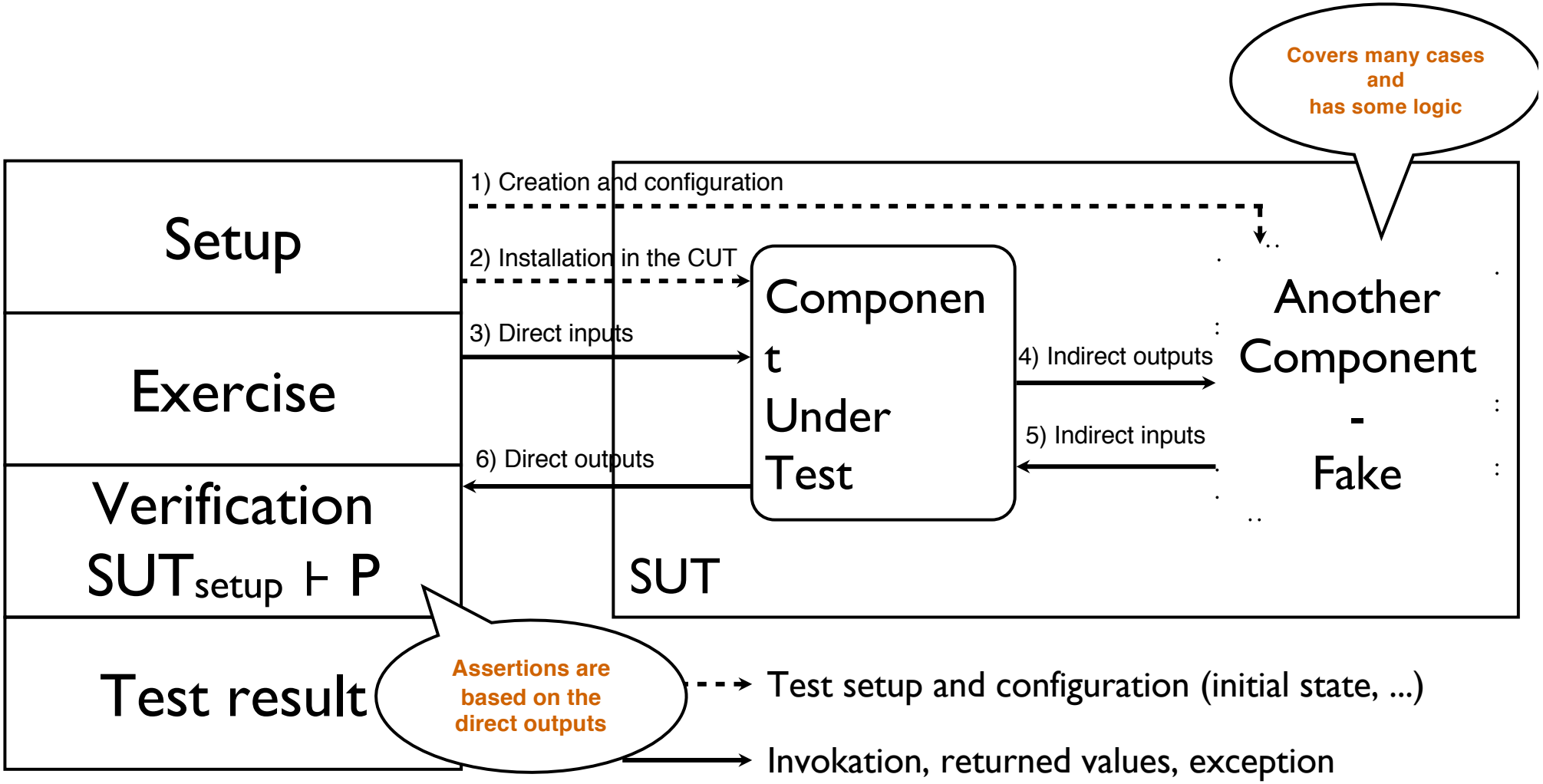
Dummy objects

To satisfy compile time checks

For runtime executions

Parameter delta of the doBusiness can be set to null

Test doubles - Fake

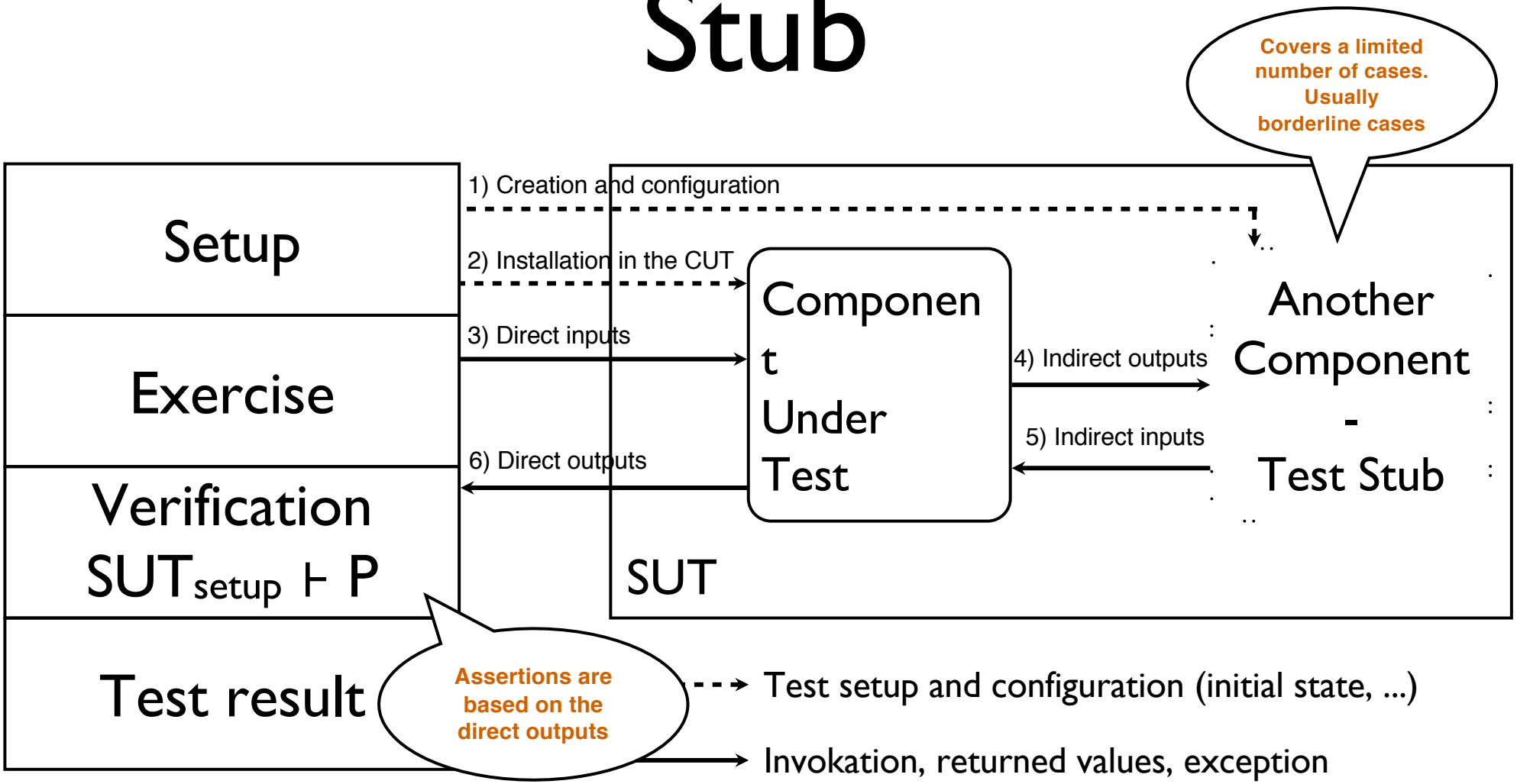


Example

```
package ch.demo.business.service;
public class AnotherComponentFake implements AnotherComponent {
    public Integer inc(Integer param) {
        return param + 1;
    }
}
```

```
package ch.demo.business.service;
public class CUTTest {
    public void testIncWhenAnotherComponentReturnsNull() {
        CUT cut = new CUTImpl(new AnotherComponentFake());
        Assert.assertEquals("inc(Integer.MIN_INTEGER) != Integer.MIN_INTEGER + 1",
Integer.MIN_INTEGER + 1, cut.inc(Integer.MIN_INTEGER, 1));
        Assert.assertEquals("inc(3) != 4", 4, cut.inc(3, 1));
        Assert.assertEquals("inc(123) != 124", 124, cut.inc(123, 1));
    }
}
```

Test doubles - Test Stub

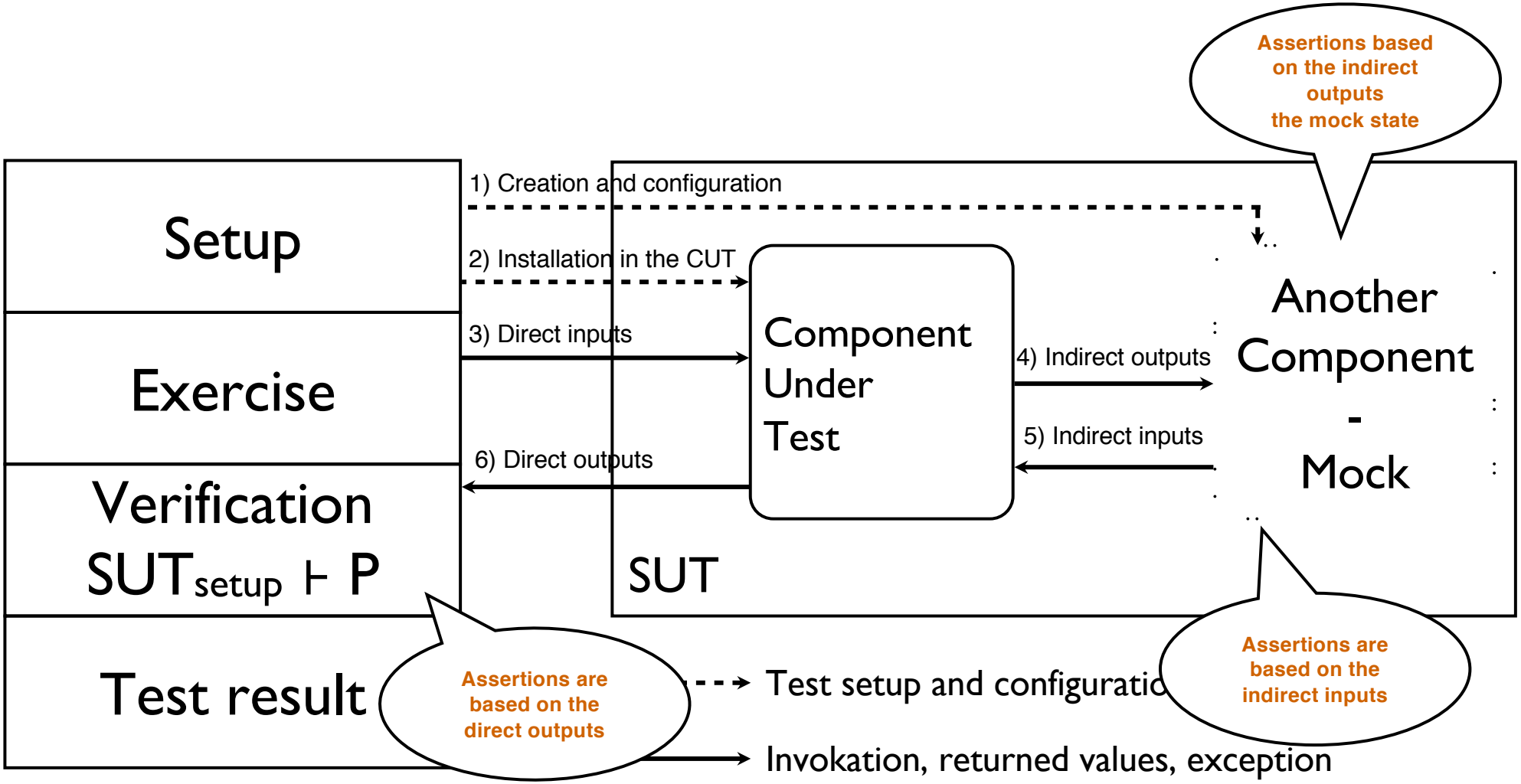


Example

```
package ch.demo.business.service;
public class AnotherComponentStub implements AnotherComponent {
    public Integer inc(Integer param) {
        return null;
    }
}
```

```
package ch.demo.business.service;
public class CUTTest {
    public void testIncWhenAnotherComponentReturnsNull() {
        Assert.assertEquals("inc(3) != 4", 4, new CUTImpl(new
AnotherComponentStub()).inc(3, 1));
    }
}
```

Test doubles - Mock



Example

```
package ch.demo.business.service;


public class CUTTest {

   public void testIncWhenAnotherComponentReturnsNull() {

      AnotherComponent mock = mock(AnotherComponent.class);
         when(mock.inc(Mockito.<Integer>anyObject())).thenReturn(null);

      verify(mock, times(1)).inc(“3”, 1);

      Assert.assertEquals("inc(3) != 4", 4, new CUTImpl(mock).inc(3, 1));

   }

}
```

Test doubles - Test spy