xUnit Test Patterns:Test Double Patterns:

Test Double

goyoki

Test Double

- ユニットテストの障害となるDOCとすり替わるもの
  - DOCが使えない・存在しない・必要なインターフェースを持たない場合でも自動テストを実行可能にする

- 由来：Stunt Double
  - 映画製作でリスキーで危険なシーンを撮影する時に、役者とすり替わって演技する人
  - 役者のような演技はできないだろうけれど、撮影シーンが要求する技能を身に付けている

How It Works

- 本当のDOC（not SUT!）とすり替わる
- Four-Phase TestのFixture SetupフェーズでDOCとすり替え
- Hard-Codeしてしまうか、Setupフェーズでの設定でふるまいを定義

How It Works

- 留意事項
  - Test Doubleを使うといっても、DOCのすべてのインターフェースを実装する必要はない
  - テストが必要とする機能のみを提供すればよい
  - 同一のDOCを対象とする場合でも、テストに合わせて複数のTest Doubleを用意することもできる

When to Use It

- 観察ポイント（Observation Point）・操作ポイント（Control Point）の確保や効率化に用いる
  - Behavior Verificationのための観察ポイントをDOCが提供しておらず、Untested Requirementが発生しているとき
  - 間接入力を制御するための操作ポイントをDOCが提供しておらず、Untested Codeが発生しているとき
  - Slow Test問題を抱えていて、テストをもっと軽快かつ頻繁に実行したいとき

When to Use It

- Test Doubleを用いる場合は、製品と異なる構成でテストすることに注意
  - Test Doubleを使わないテストを最低でも1回は実施すべき
  - 検証すべきSUTを部分的にすり替えてしまってないか注意すべき
  - 多用するとOverspecified Software状態となり、Fragile Testを生み出すことがある

Test Doubleのタイプ

- Test Sutb
- Test Spy
- Mock Object
- Fake Object
- Dummy Object

Test Stub

- DOCの代替となる
- SUTの間接入力を操作する操作ポイント(Control Point)を持つ

Test Stub

- 注意事項
  - Test Stubは「テストのための未実装コードの仮実装」という意味でもつかわれる
  - ここでは混乱を避けるためにそれをTemporary Test Stubと呼称する

Test Spy

- DOCの代替えとなるもの
- Test Stubとしても機能する
  Test Stub＋観察ポイント(Observation Point)
- 間接出力を得るための観察ポイントを持つ
  - SUTからのメソッド呼び出しも取得できる
    - Behavior Verificationを実現
  - 間接出力を保持し、**後から**検査できるようにする

Mock Object

- DOCの代替となるもの
- Test Stub＋Assertionを拡張したもの
  - テストが失敗しない場合、Test Stubとしても動作する
- SUTを実行する**中で**間接出力を検査する観察ポイント(Observation Point)として機能

Fake Object

- DOCの機能的な代替となるもの
- DOCと同じように機能する
- SUTの間接入力・間接出力の検査以外の目的で使用
- DOCが遅すぎる/利用できない/許容できない副作用を持つ場合に一般的に使用
  - 例として、「Faster Tests without Shared Fixture」で説明したような、DB関連テストの高速化ができる

Dummy Object

- テストもSUTも気に掛けないオブジェクト
  - SUTのメソッドのシグニチャが、パラメータとして特定のオブジェクトを要求することがある
    テストもSUTも気に掛けない場合、そのオブジェクトとして用いるのがDummy Object
- 本来Test Doubleと区別されるものではない
  - Value Patterns（Literal Value、Derived Value、Generated Value）の代替手段のようなもの

Procedural Test Stub

- 手続き型言語でのTest Double
  - 手続き型言語の世界ではよくTest Stubと呼称される
  - テスト/デバッグを進めるために、まだ利用できないコードの代替として一般的に使用
  - Test Logic in Productionとして実装されることも
    - 例えばDebugging Flagを導入
- Test Double分類下のTest Stubの定義と区別するため、「Procedural Test Stub」と呼称する

Implementation Notes

- 作成する際はいくつかの考慮点がある:
  - 特定のテストで使うべきか、複数のテストで共有すべきか
  - コードの中に組み込むべきか、動的に生成すべきか
  - Test Doubleを使うために、どのようにSUTとやり取りするか

Unconfigurable Test Double

- 設定不要なTest Double
  - Fake ObjectやDummy Object
    - Dummy Objectは使われないのでそもそも不要
    - Fake Objectは（単純だったり軽快だったりする点以外で）本物のDOCとしてふるまうように実装される
- 本来のDOCのようにSUTにTest Doubleを組み込んで利用する

Hard-Coded Test Double

- Hard-CodedされたTest Double
- 単一のテストで特定の値を返したり、特定のメソッド呼び出しを検証したりするTest Stubを実現する方法として、もっともシンプル
- いくつかのフォームを持つ
  - Self Shunt
  - Anonymous Inner Test Double
  - Test Double Classとして実装されたTest Doubles

Configurable Test Double

- 設定で構築可能なTest Double
- 複数のテストで同じTest Doubleを使う場合によく適用する
- xUnit ファミリーの多くがツールキットを提供

Installing the Test Double

- SUTを実行する前に設定
- Four-Phase TestのSetupフェーズで、Test Doubleを組み込むための多くのsubstitutable dependency patternsを使える

- 色々なパターンがあるので、詳細はこれから説明