# 適切なテストツール (テスト理論パート 2)

この記事は、インテル® デベロッパー・ゾーンに公開されている「The Right Toolset for Testing (Testing Theory Part 2)」の日本語参考訳です。

---


この記事は、ソフトウェア・テストの理論と実践に関するシリーズ (全 2 パート) のパート 2 です。パート 1 では、想定される標準状態、想定される異常状態と障害、想定外の状態について述べました。パート 2 では、実用的なテストツールとそれらを利用してさまざまなシステム状態を検証する方法を説明します。

## 閑話: 不十分なテストは恥となる

最初に、全く異なるお話をしましょう。

私は最近、あきれるほど粗末なテストの見本とも言うべき例を目にしました。ストックホルムの高級ショッピング・モールを散策していたとき、大画面にライブ動画広告が流れていました。「Gladiatorerna」というテレビ番組の新シーズンの開始を知らせるため、広告が流れるたびに、ゼロ (番組の開始時刻の 20:00 CET) に向けてカウントダウンするというもので、よくできていました

20:00 を過ぎてから、その広告が再び流れました。そこで興味深いことが起こったのです。

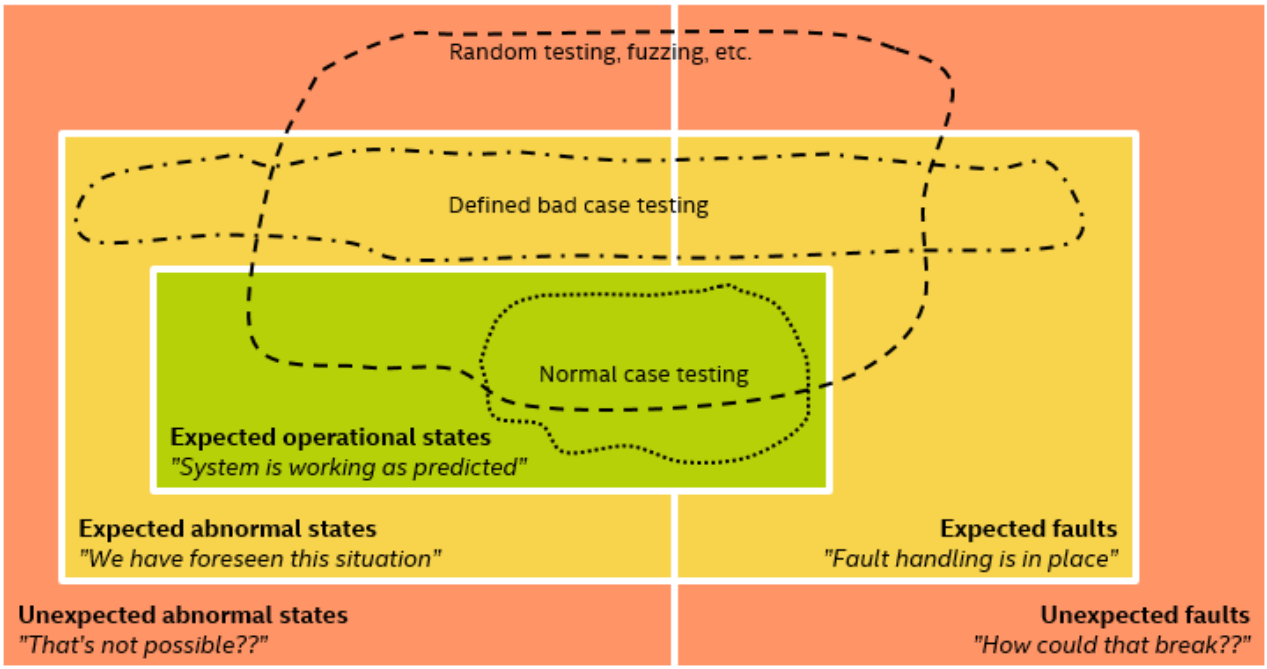
驚いたことに、残り時間がマイナスで表示され、それは非常に分かりづらいものでした。明らかに、"番組開始から約 2 分経過" とすべきところを、"マイナス 1 時間、マイナス 2 分、28 秒" と表示されていたのです。これはバグ以外の何物でもないでしょう。

これが表示されていたということは、カウントダウン (20:00) を過ぎた場合について誰もテストしていなかったということです。スウェーデンの主要テレビ局のプライムタイム番組の宣伝としては、非常にお粗末と言わざるをえません。20:00 までしか放送されないと仮定していたのかもしれません。しかし、カウントダウンを過ぎたケースについてもテストするのは当たり前のことで、これは想定される状態 (緑色のボックス) の範囲であると私は思います。突然 2017 年から 1867 年になったわけではないのですから。しかし、実際には想定外の状態 (赤色のボックス) に分類されました。そして、予期しない、受け入れがたい結果となったのです。

このような過ちは起こらないようにする必要があります。

## ランダムテスト

パート 1 では、創造力に富んだテスターはテスト範囲を拡大し、より多くの想定される状態のセットを拡張して、適切な対応を可能にすると説明しました。優れた人材に加えて、優れたツールを賢く利用することでテストを拡張できます。これは特に、ランダムな特性を持つテストの作成に効果があります。
ブラックチームのようなトップレベルのテスターであっても限界はあります。彼らは、黄色のボックスの範囲を拡大することはできますが、赤色のボックスの範囲には対応できません。そこで、テストの生成、ランダムテスト、ファジングが必要になります。テストを作成するためのルールを設定し、マシンにテストを作成させ、例えどんなに無意味に見えたとしても、これらのテストをシステムで実施します。




上の図に示すように、ランダムテストやファジングは、通常、システムの標準動作を含む、すべてのボックスにまたがるテストを作成します。そのため、さまざまな状態をカバーすることができます。最も重要なことは、テストの作成に人間が直接介入しないため、想定される状態ボックスの外側もカバーしていることです。

実際、ファジングはテストに非常に役立つツールであることが実証されています。セキュリティー違反の多くでは、ランダムに生成された入力データ (Web 要求、メディアファイル、ネットワーク・パケット) がシステムへの侵入経路を見つけるためによく使用されます。適切に使用すれば、ファジングは、人間が思いつかないような入力データとイベントシーケンスを作成します。それこそが、人間の想像力の限界によって見落とされている穴をふさぐために必要です。

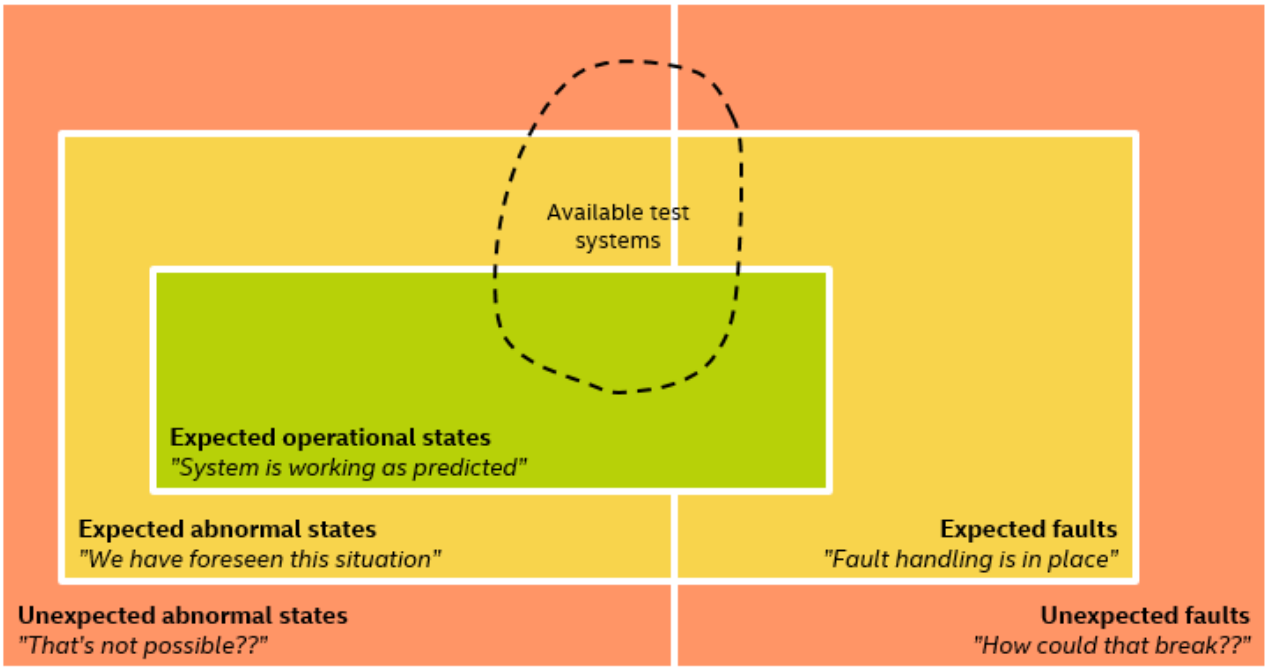
哲学的に言えば、想定外の動作を直接作成するよりも、そのような動作を作成するルールを設定するほうが簡単です。ランダムなルールを使用することで、新しいものを作り出すことができます。

## テストシステム

忘れられがちなテストのもう 1 つの側面として、テストシステムによって実行可能なテストが制限されます。特定の状態や特定の入力に対応していないテストシステムでは、それらの状態や入力をテストすることができません。

実際のテストの実行は、利用可能なテストシステムの表現力によって制限されます。テストしたいものではなく、テストできるものしかテストすることができません。テストしたい項目のリストはあっても、既存のテストシステムとインフラストラクチャーでは実装できないことはよくあります。

この状況を図に示すと以下のようになります (論点を強調するため誇張しています)。

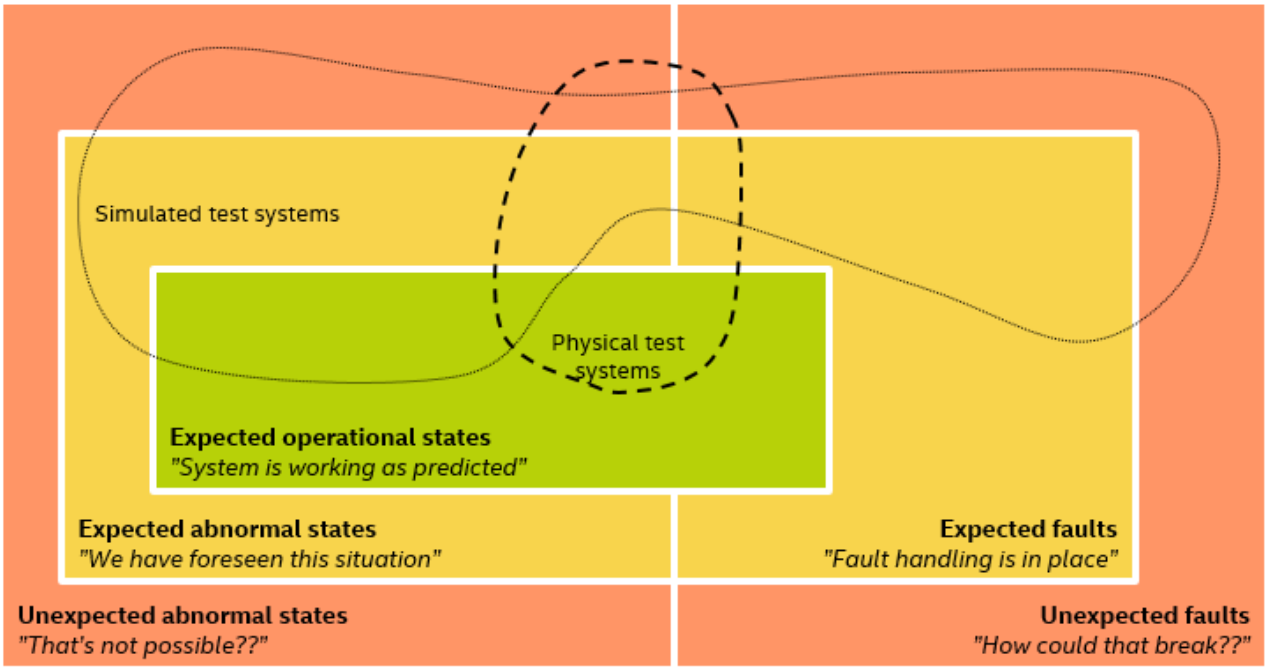



これを前出の利用可能な異なるテストケースの図と組み合わせると、テストは利用可能なテストケースとテスト実行システムがオーバーラップする部分しかカバーできません。利用可能なテストシステムの制限により、思いつくすべてのテストを実装できるとは限りません。また、テストシステムで実行することは可能でも、テストが作成されない場合もあります。

例えば、多数のプロセッサー・コアを利用して並列アルゴリズムのスケーリングと振る舞いをテストする、単純な例について考えてみます。このテストは、テストシステムで利用可能なコア数に制限されます。テストでは、多種多様なすべてのシステムを用意できるわけではないため、テストされていないコア数でソフトウェアを実行するユーザーが必ず存在します。

テストでより多くのケースをカバーするため、テストケースの作成方法だけでなく、テストの実行方法についても革新する必要があります。組込みシステム、特に航空宇宙分野では、通常これに多大な労力が費やされています。スキャパレリの着陸に関する投稿 (英語) で述べた物理テスト装置は、そのようなテストに必要な典型的なもので、定置装置の簡単なテストだけでなく、振動や移動をカバーするようにテストを拡大する大規模な専用装置です。

ここで、強力なテスト実行プラットフォームとして、コンピューター・シミュレーション・システムが登場します。物理的にテストできない標準状態、ハードウェアで再現できない異常状態、ハードウェアの障害をテストします。シミュレーションを利用することで、次に示すように、テストでカバーできる範囲が拡大します。




シミュレーションは、物理テストシステムで可能なテストのスーパーセットを提供するとは限りません。中には、シミュ—レーションでは表現できず、ハードウェアでのみテスト可能なものがあるからです。

シミュレーションの可能性については、以前の記事を参照してください。

- インテル® Memory Protection Extensions (インテル® MPX) の機能を確認する方法に関する Xen* のバグ (英語): 実行プラットフォームが変更された場合にのみ見られるいくつかのバグの例もあります。
- ハードウェアでノード数のスケーリングが利用できない場合の IoT システムのテスト (英語) または非常に大きなメモリーを使用する場合 (英語)
- オペレーティング・システムの堅牢性をテストするためシステムにハードウェア・イベントを挿入する (英語)
- Simics を使用したハードウェアレベルの問題の挿入 (英語): ハードウェアの問題がある場合のソフトウェア動作のテストです。
- シミュレーションを使用したセキュリティー・テストに関する Wind River* のホワイトペーパー (英語)

免責事項: 常に物理ハードウェアでの最終テストが必要です。実行する前にテストし、テストしたものを実行する必要があります。シミュレーションはテスト範囲を拡大し、システムの安定性を高めるのに役立ちますが、物理システムでのテストに取って代わるものではありません。

## 謝辞

テストに関する私の考え方を図解するように助言してくれたエバンジェリスト・チームの Richard Osibanjo 氏に感謝します。数が多くなりましたが、図にまとめることができました。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。