FIRST-Regel

Ein gutes Werkzeug zur Identifizierung von guten Unit Tests ist FIRST, auch als FIRST Properties von Unit Tests bekannt. FIRST steht dabei für:

- Fast
- Isolated
- Repeatable
- Self-Validating
- Timely

Fast: Ein Unit Test soll schnell in der Ausführung sein. Warum? Nun, so granular wie Unit Tests sind, reden wir bei der Durchführung der Tests nicht von einem oder zwei Unit Tests, sondern im Laufe der Zeit wird die Anzahl eher auf ein paar Hundert Unit Tests ansteigen, vielleicht auch ein paar Tausend. Und der große Vorteil einer guten Codeabdeckung durch automatisierte Tests ist, dass wir bei Änderungen quasi durch einen Mausklick überprüfen können, ob noch alles funktioniert. Wenn der Durchlauf aller Tests aber ein oder zwei Stunden dauert, oder vielleicht noch länger, dann wird die Motivation abnehmen, diese Testläufe in kurzen Intervallen durchzuführen. Das geht wieder auf Kosten der Qualität, weshalb jeder einzelne Unit Test schnell sein muss, zwei bis drei Sekunden maximal. Und noch schneller wäre noch besser.

Isolated: Unit Tests sind unabhängig voneinander. Das bedeutet, die Durchführbarkeit eines Unit Tests hängt weder davon ab, dass ein anderer Test bereits gelaufen ist, noch dass ein anderer Test vorher ein bestimmtes Ergebnis erzielt hat. Jeder Unit Test erzeugt im Arrange (siehe auch AAA-Regel) selber die für ihn notwendige Startkonstellation.

Repeatable: Ein Test muss beliebig oft wiederholbar sein. Und zwar ohne dass zwischen zwei Testläufen manuell oder auch durch einen anderen Prozess (z.B. ein anderer Test) die im ersten erzeugten Zustände zurückgesetzt werden müssen. Oder anderes formuliert: Ein Unit Test hinterlässt keine persistenten Zustandsänderungen im System. Entweder erzeugt er erst gar keine solchen Zustandsänderungen oder er räumt sie selber zum Abschluss des Tests wieder auf, indem er z.B. Werte aus der Datenbank wieder löscht oder zurücksetzt. Das steht in einem engen Zusammenhang mit den Punkt Isolated.

Self-Validating: Ein Unit Test kann immer selber eineindeutig bestimmen, ob er erfolgreich war oder fehlgeschlagen ist. Für diese Beurteilung ist definitiv keine manuelle Bewertung durch einen Menschen erforderlich. Klingt blöd und eigentlich selbstverständlich. Und doch finden wir das immer mal wieder vor, dass erst einer der Entwickler oder Tester nochmal darauf schauen muss, um zu beurteilen, ob der Testlauf jetzt ok war oder eher nicht. Wenn das der Fall ist, dann stimmt mit der Grundidee der entsprechenden Test etwas nicht.

Timely: Unit Tests sind nicht besonders gut geeignet, um sie irgendwann mal später zu schreiben, Wochen oder Monate nachdem der Sourcecode geschrieben wurde. Quasi nur, um nachträglich noch eine gute Testabdeckung zu erreichen. Wer schon richtige Unit Test geschrieben hat, der hat schnell gemerkt, dass alleine das Schreiben der Unit Tests einen großen Einfluss darauf, wie der Sourcecode strukturiert wird. Deshalb gehören die Erstellung von Sourcecode und Unit Tests unmittelbar zusammen.