Tests

Testarten und Testpyramide

Viele Begriffe - ein Ziel

- Test werden in Stufen / Gruppen unterteilt.
- Unterteilung erfolgt nach Granularität der zu testenden Einheit.
- Whitebox, Graybox und Blackbox Tests

Je kleiner die zu testende Einheit Code desto weniger Aufwand kostet das Schreiben eines Tests und desto geringer ist die Testausführungszeit.

Stufen und Arten

- Unit-Test / Modul-Test
- Integrations-Test
- Komponenten- / Functional-Test
- Systemtest / Smoke-Tests
- End-2-End-Test
- Abnahmetest (Acceptance-Test)

Unit-Tests

- Ist ein Test auf der Ebene der einzelnen Module der Software.
- In Java testen der einzelnen Klasse
- Whitebox Test
- Abhängigkeiten werden durch Mocks ersetzt
- Ziel ist die Sicherstellung und Dokumentation der Funktionsweise des Codes
- Ausführung lokal / CI-Server ($ mvn test)

Integration-Tests

- Integrationstest testet die Zusammenarbeit voneinander abhängiger Komponenten.
- Testschwerpunkt liegt auf den Schnittstellen der Komponenten (Klassen) untereinander
- Graybox Test
- In Java mehrere Klassen im Verbund getestet (z.B. Spring ApplicationContext initialisert / Anwendungsteile aktiviert)
- Ausführung lokal / CI-Server ($ mvn verify)

Component-Tests / Functional-Tests

- testet gegen die Schnittstellen der Anwendung (UI / Rest / SOAP / etc.)
- Umgebung der Anwendung (DB / weitere Dienste) sind ge-”mocked”
- Ziel: komplette UseCases zu testen (Blackbox Test)
- Ausführung lokal / CI-Server ($ mvn verify)
- Container / Laufzeitumgebung (z.B. Tomcat) wird hochgefahren

System-Tests / Smoke-Tests

- das gesamte System wird gegen die gesamten Anforderungen (funktionale und nicht funktionale Anforderungen) getestet
- Test findet auf einer Testumgebung statt und wird mit Testdaten durchgeführt
- Blackbox Test
- Testumgebung soll die Produktivumgebung simulieren, d.h. dieser möglichst ähnlich sein
- Ausführung lokal oder CI-Server gegen Testumgebung (z. B. Tests in JAR)

Smoke-Test

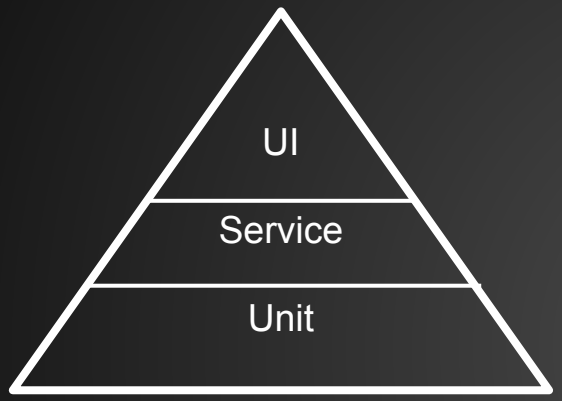
- Untermenge der System-Tests
- sind System-Tests allerdings eingegrenzt auf die 80% der geschäfts- / umsatzrelevanten Anwendungsfälle (Use cases / Workflows)
- Ausführung während Deployment der Produktionsumgebung oder gegen Testumgebung
- Ziel: Nachweis, das die Kerngeschäfts-prozesse funktionstüchtig sind

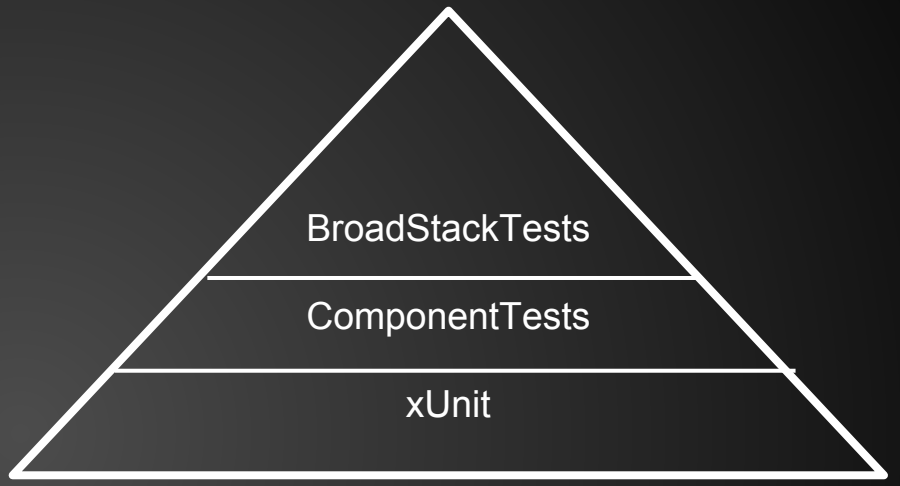
End-to-End-Tests

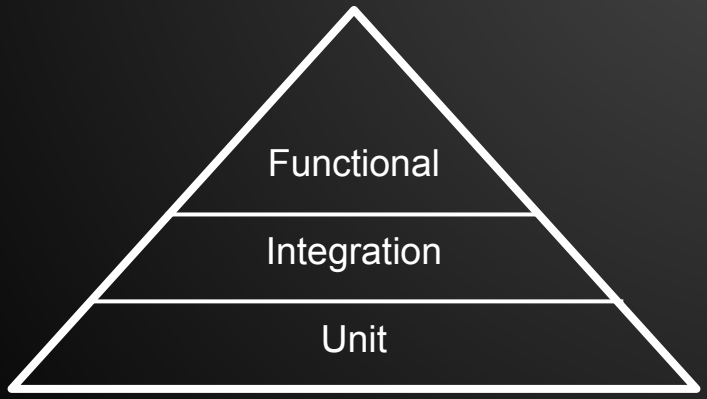
- test kompletter funktionaler Anforderungen unter Einbeziehung aller beteiligter, auch externer, Systeme
- z. B. Firewall, Web, Middleware, DB, externe Dienstleister (Druckaufträge, SMS-Versand, Kartendienstleister)
- schwer oder mit hohem Aufwand automatisierbar daher meist manuell getestet
- oft bei erster Inbetriebnahme neuer Komponenten oder Prozesse

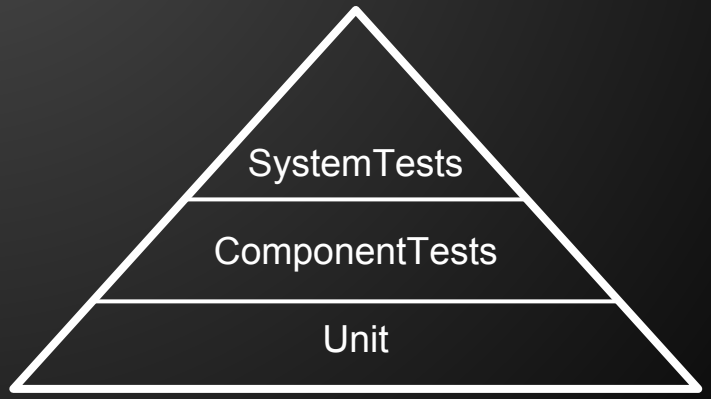
Quelle: http://upload.wikimedia.org/wikipedia/commons/1/18/All_Gizah_Pyramids-2.jpg

Pyramide(n)

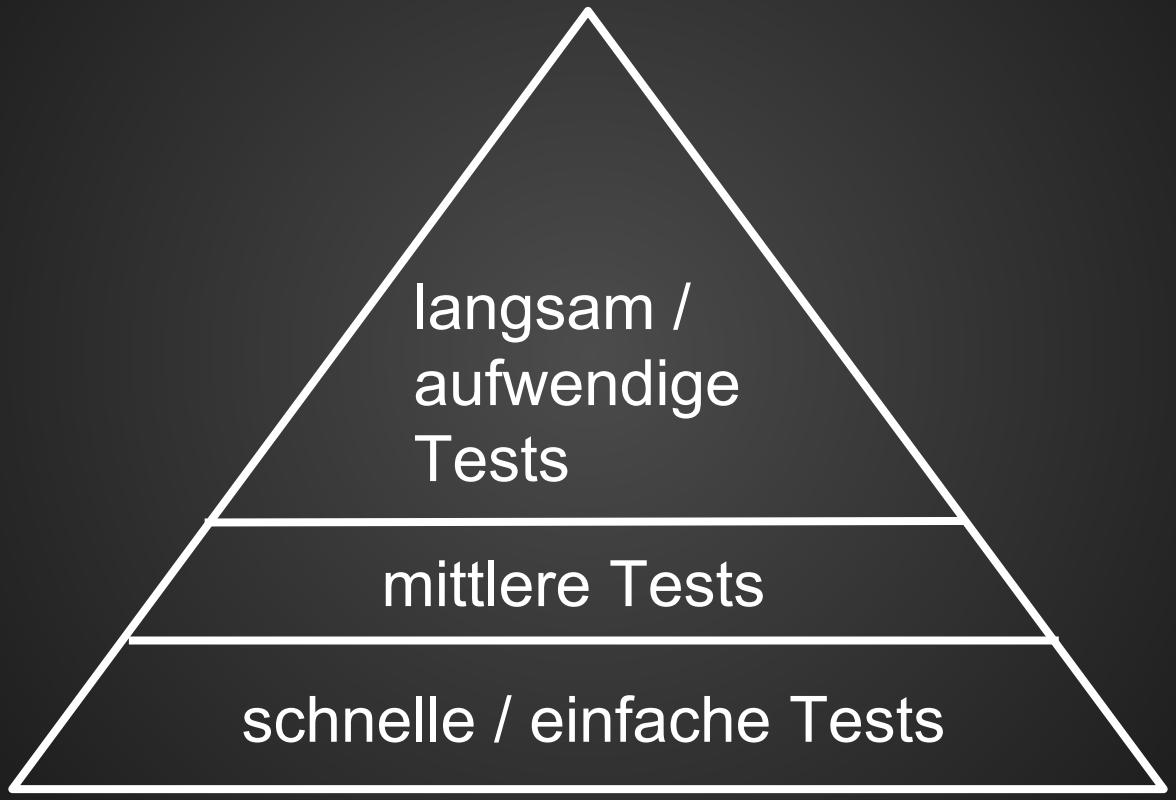

UI
Service
Unit


BroadStackTests
ComponentTests
xUnit

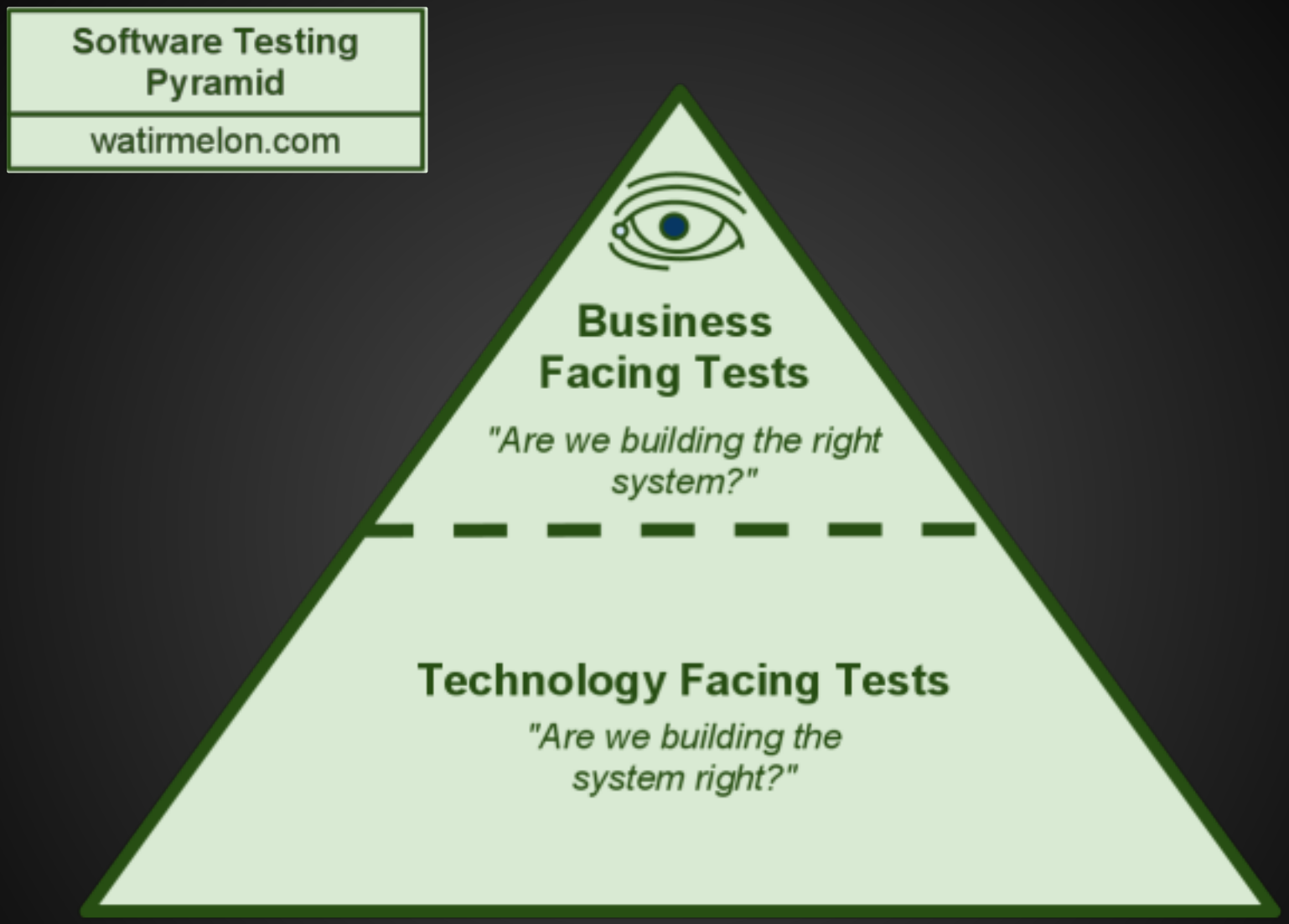

Functional
Integration
Unit


SystemTests
ComponentTests
Unit

Pyramide

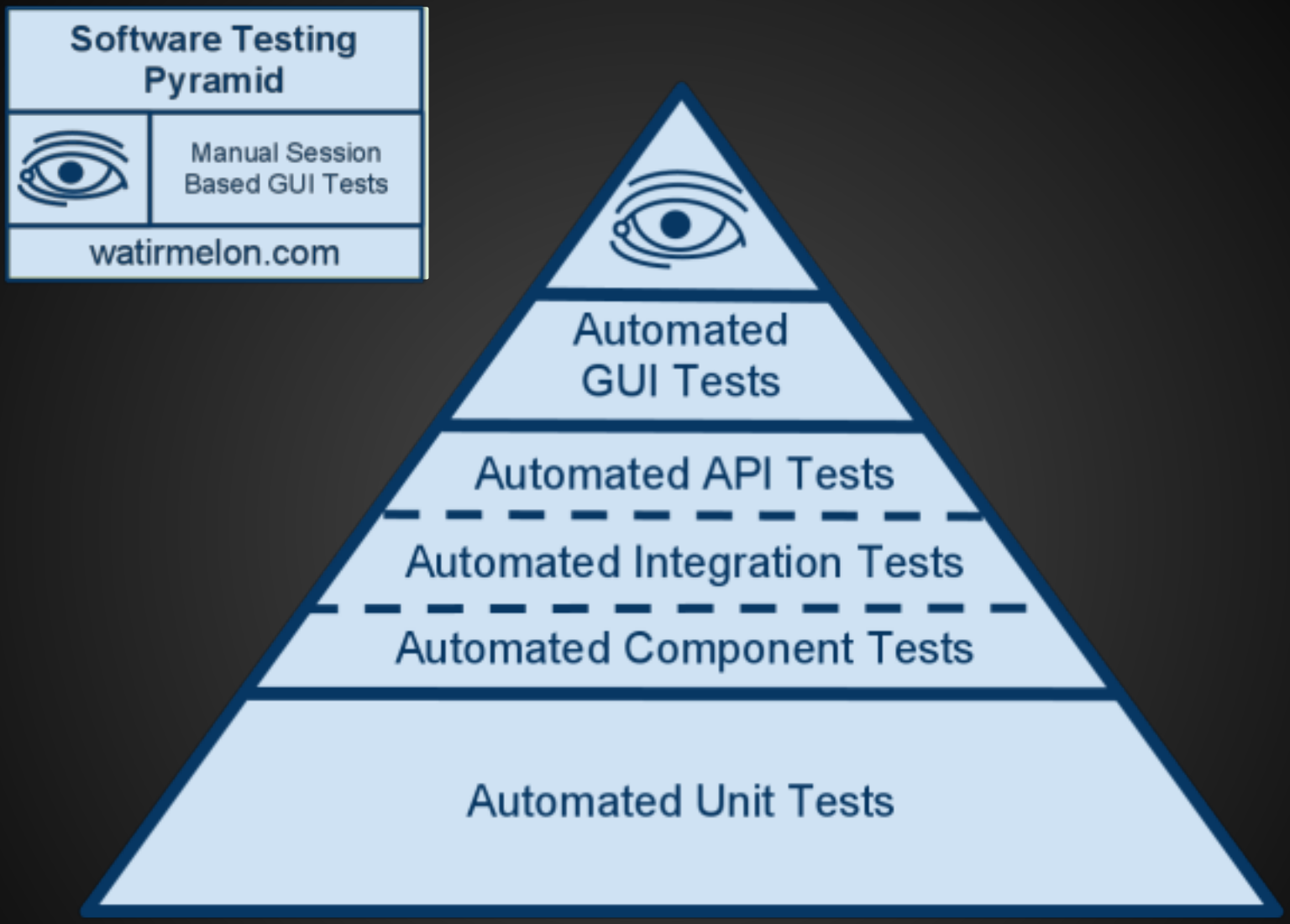


Automation der Testauführung ist das Ziel!



von Alister Scott

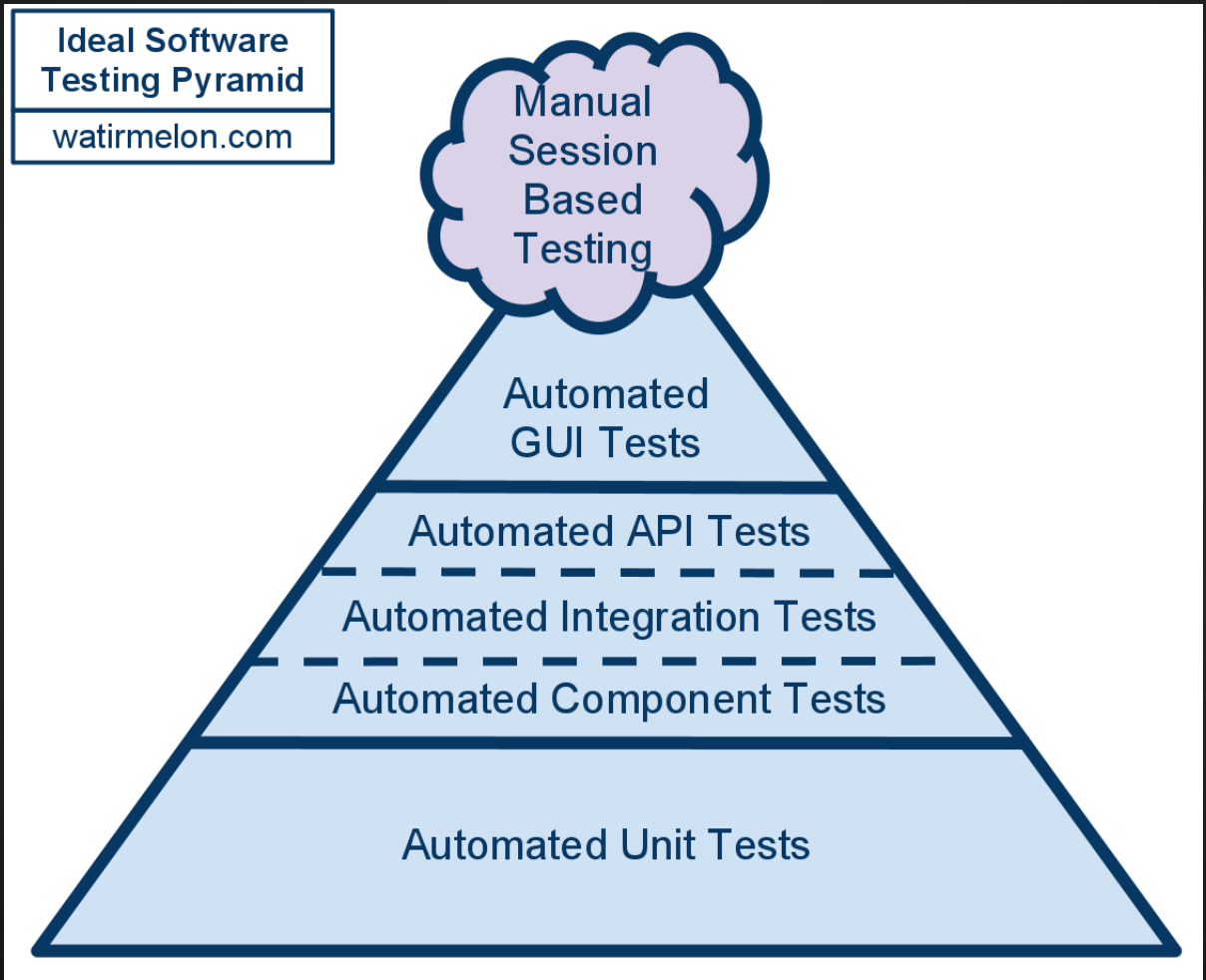
Quelle: http://watirmelon.com/2011/06/10/yet-another-software-testing-pyramid/



von Alister Scott

Quelle: http://watirmelon.com/2011/06/10/yet-another-software-testing-pyramid/

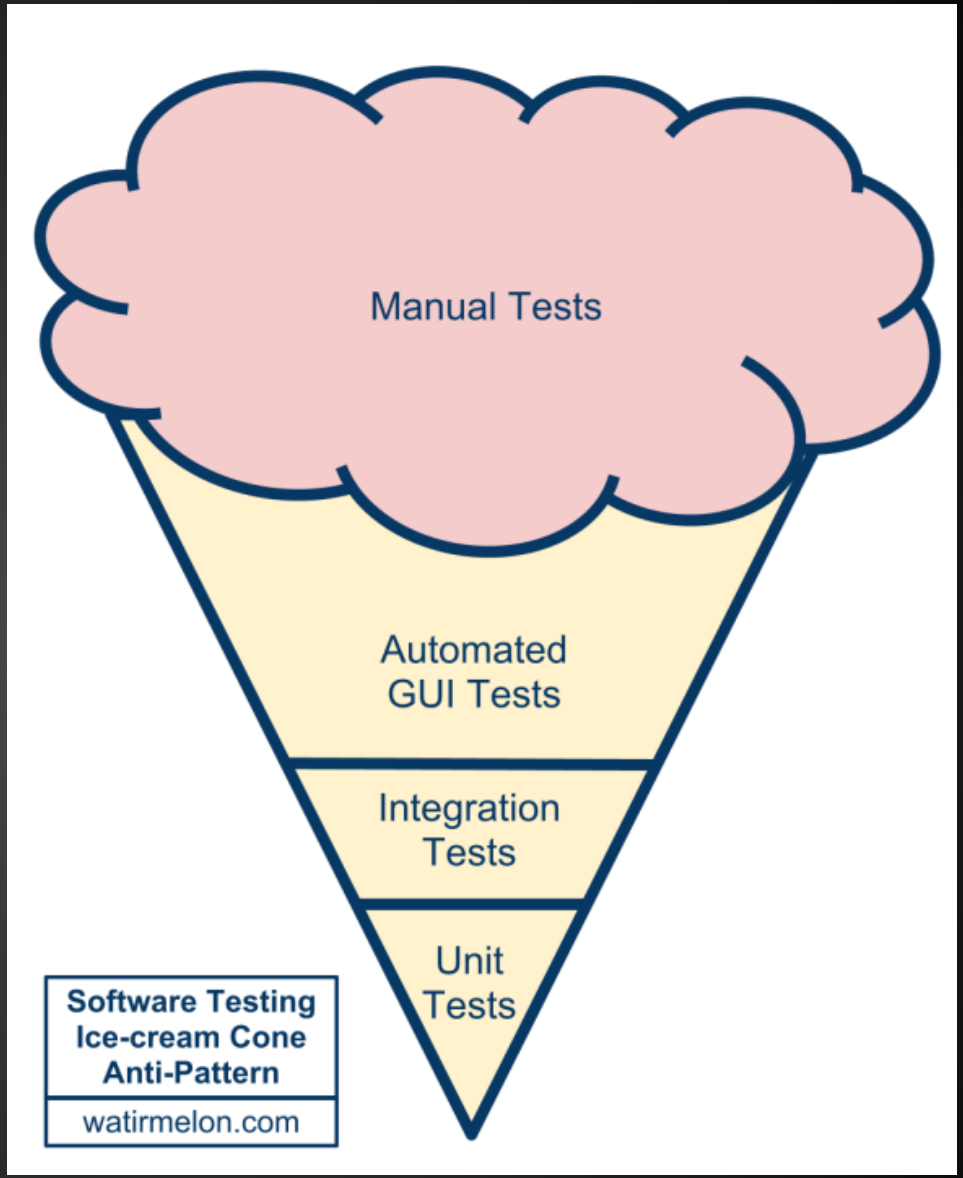
Ideale Testpyramide



Quelle: http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone

Anti-Pattern

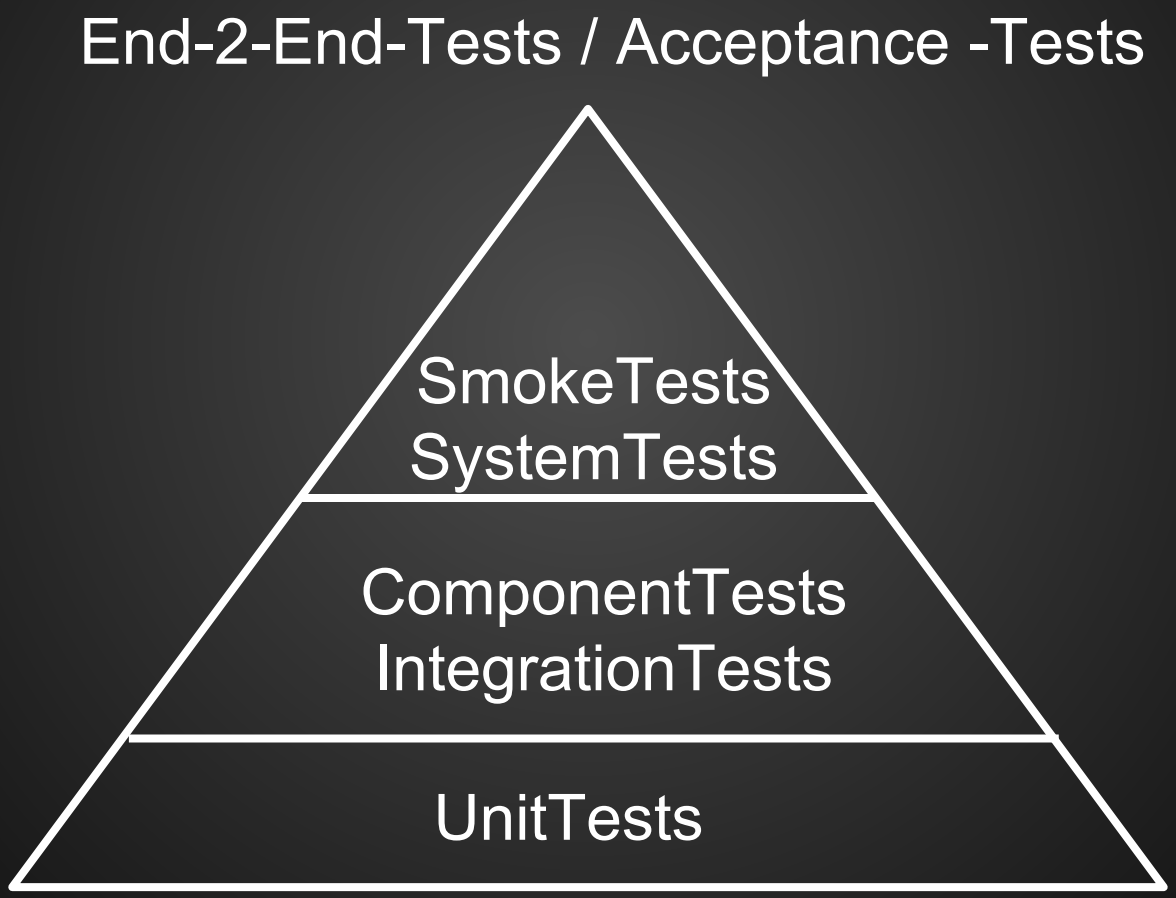
Die Eis-Tüte (!)



von Alister Scott

Quelle: http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone

Zusammenfassung der Testarten in einer Pyramide



Essenz

- Je weiter unten in der Pyramide
  - desto mehr Tests
  - feingranularere Tests
  - desto schneller / einfacher programmierbar
  - je kürzer die Ausführungszeit pro Test
- Je weiter oben zur Spitze der Pyramide
  - desto weniger Tests
  - desto langlaufender in der Ausführungzeit je Test
  - aufwendigerer programierbar