Testautomatisierung in Legacy-Systemen

Usch Wildt

Cebit / 19.03.2015

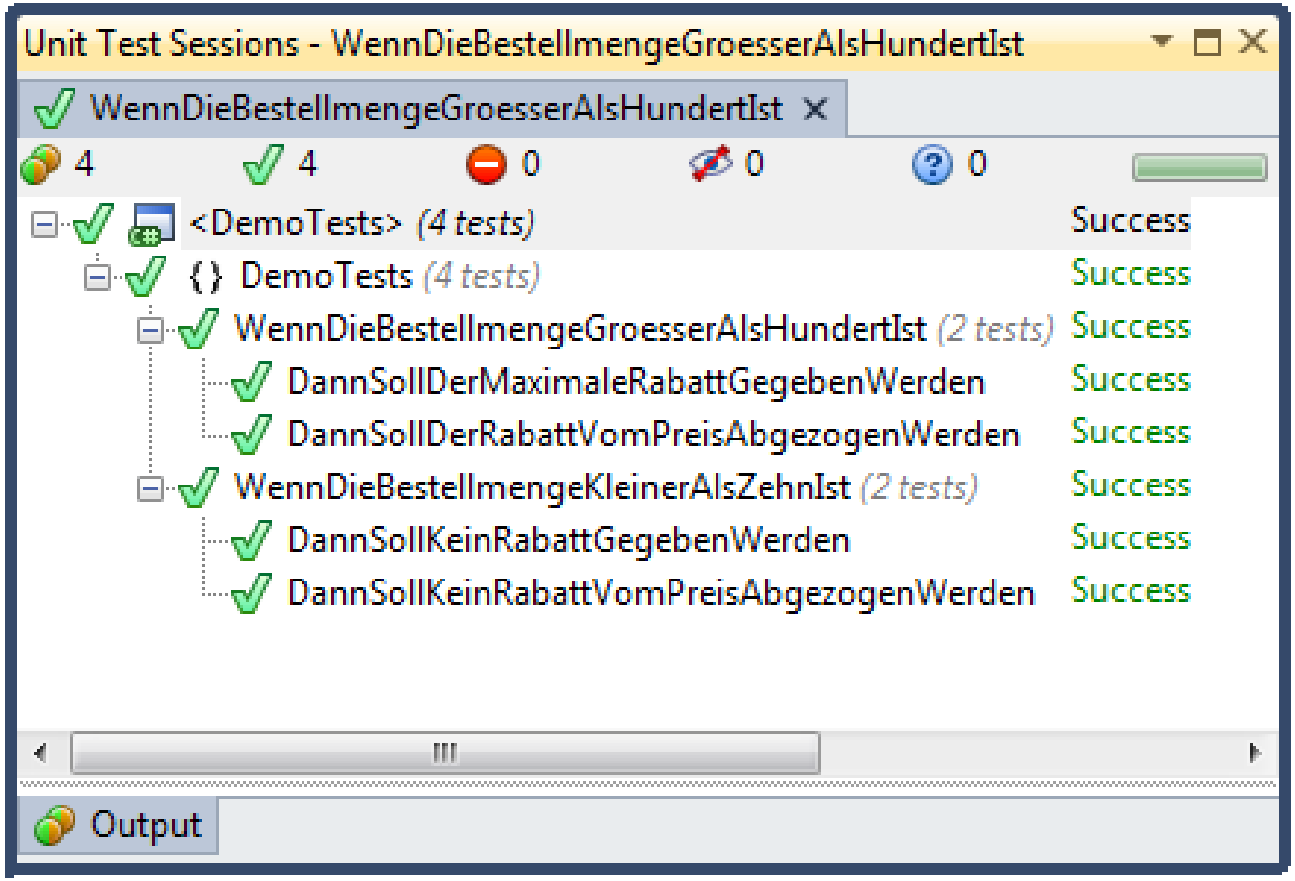
Mein Hintergrund:

- Chemiestudium, IT-Freelancerin
- GF der aldebaran GmbH seit 1998
  Individualsoftware
- Agile Methoden seit ca. 2000
- Schwerpunkte: Prozesse + Testautomatisierung

Automatische Tests

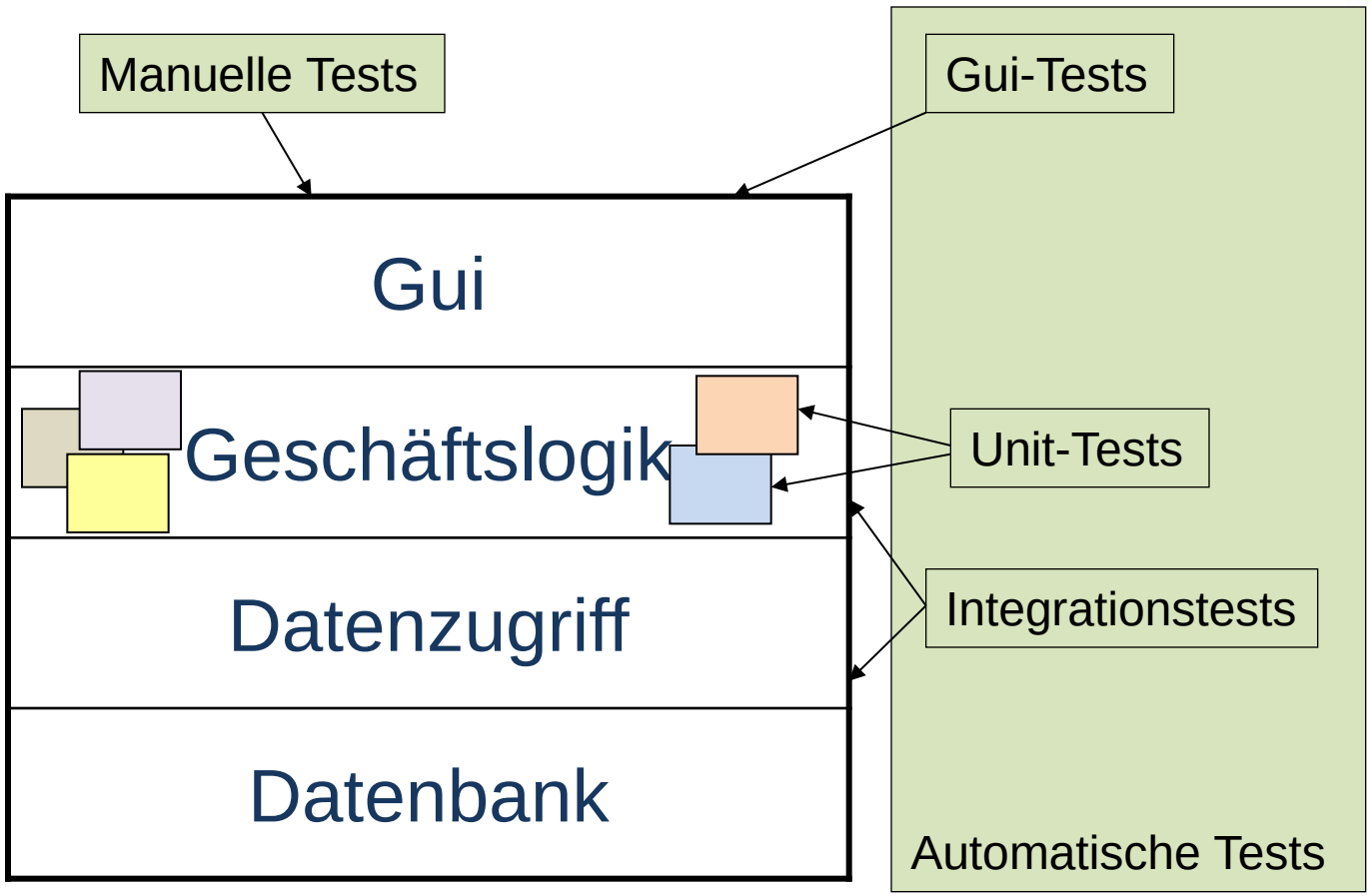
- fleißig, geduldig und pedantisch
- geben schnelles Feedback
- schaffen Vertrauen
- ermöglichen Änderungen
- decken Nebeneffekte auf
- dokumentieren die Anwendung

Automatische Tests dokumentieren die Anwendung:



Architektur im „Greenfield“

Gui
Geschäftslogik
Datenzugriff
Datenbank


Manuelle Tests
Gui-Tests
Gui
Geschäftslogik
Datenzugriff
Datenbank
Unit-Tests
Integrationstests
Automatische Tests

Legacy-Systeme

Änderung notwendig: Hilfe!

➔ dafür Refactoring notwendig

➔ dafür Tests notwendig

➔ dafür Refactoring notwendig

➔ dafür Tests notwendig

➔ dafür …

Was tun???

Projekt mit „besonderen Herausforderungen“:

- monolithische VB6-Anwendung
- 1:1 Migration nach .NET
- keinerlei automatische Tests
- viele Fehler
- viel „erstaunlicher Code“
- fehlende Doku, fehlendes Wissen
- ständiger Zeitdruck
- begrenztes Budget

Architektur im „Brownfield“

Gui

Bug

SQL

SQL

Logik

Gui mit Logik mit SQL

Bullshit

Unlogik

noch ein Bug

SQL

SQL

SQL

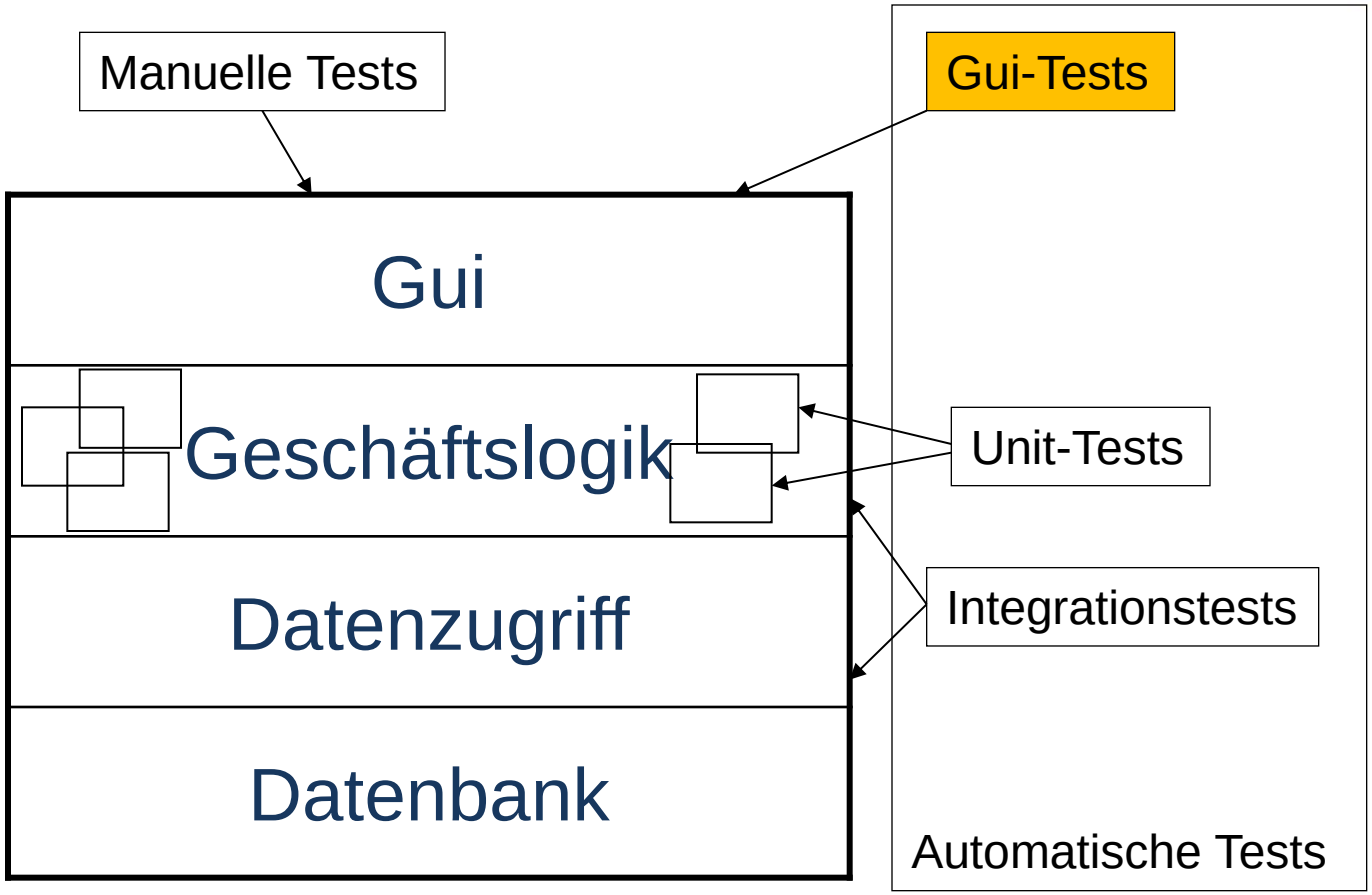
On error resume next

Keine Unit-Tests möglich

- globale Variablen
- Geschäftslogik in Gui-Elementen
- riesige Klassen mit gegenseitigen Abhängigkeiten

Was wir getan haben:

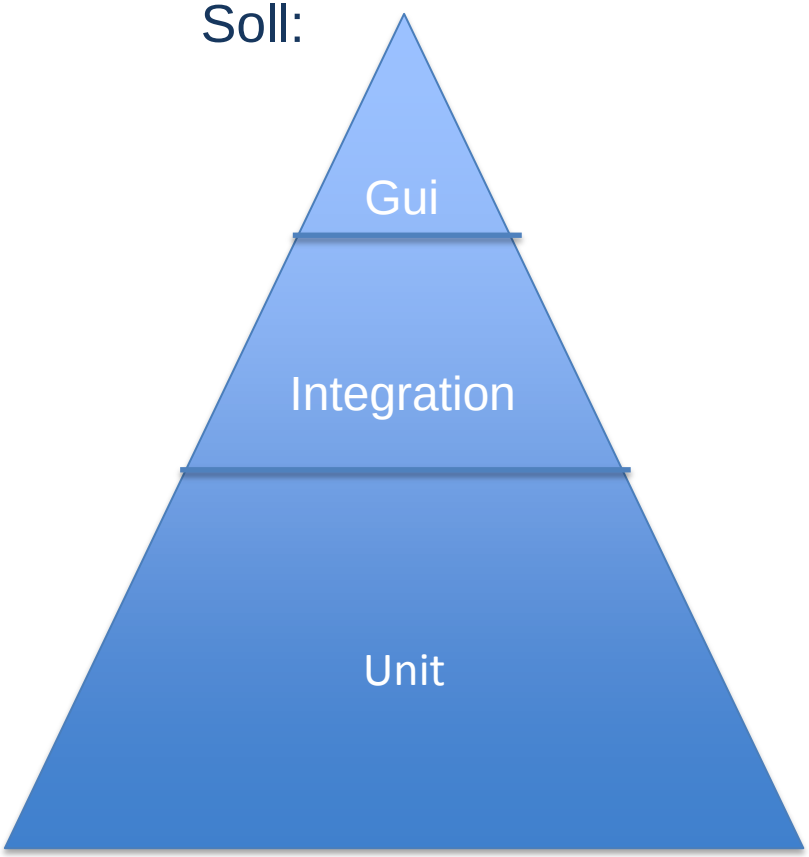
1. Gui-Tests
2. Unit-Tests für neuen Code
3. sukzessive Refactoring und Tests für
   - nennenswerte Erweiterungen
   - Bugs
   - Risikobereiche (aus Kundensicht)
   - wtfs (aus Entwicklersicht)



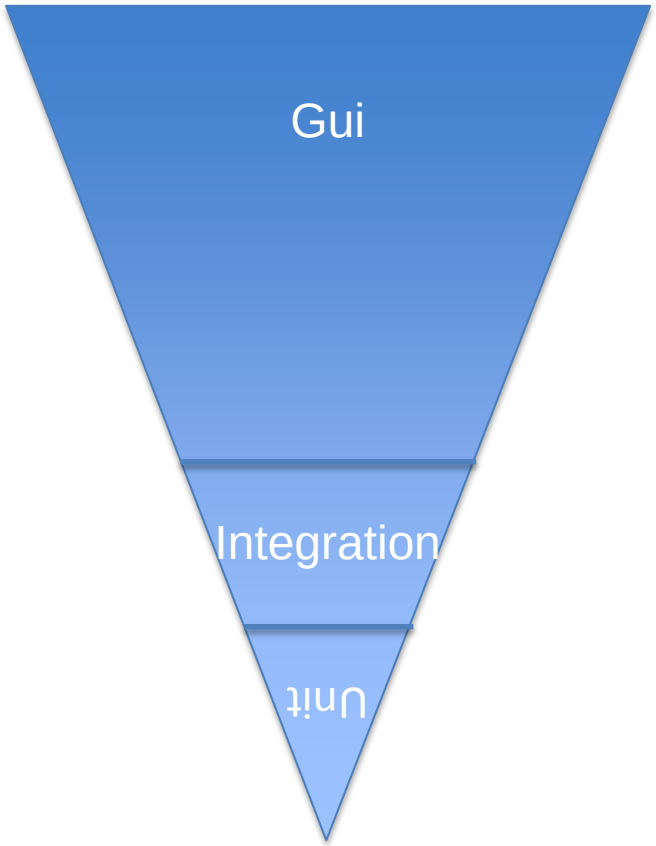
Greenfield - Tests

Gui-Tests

Soll:



Ist:



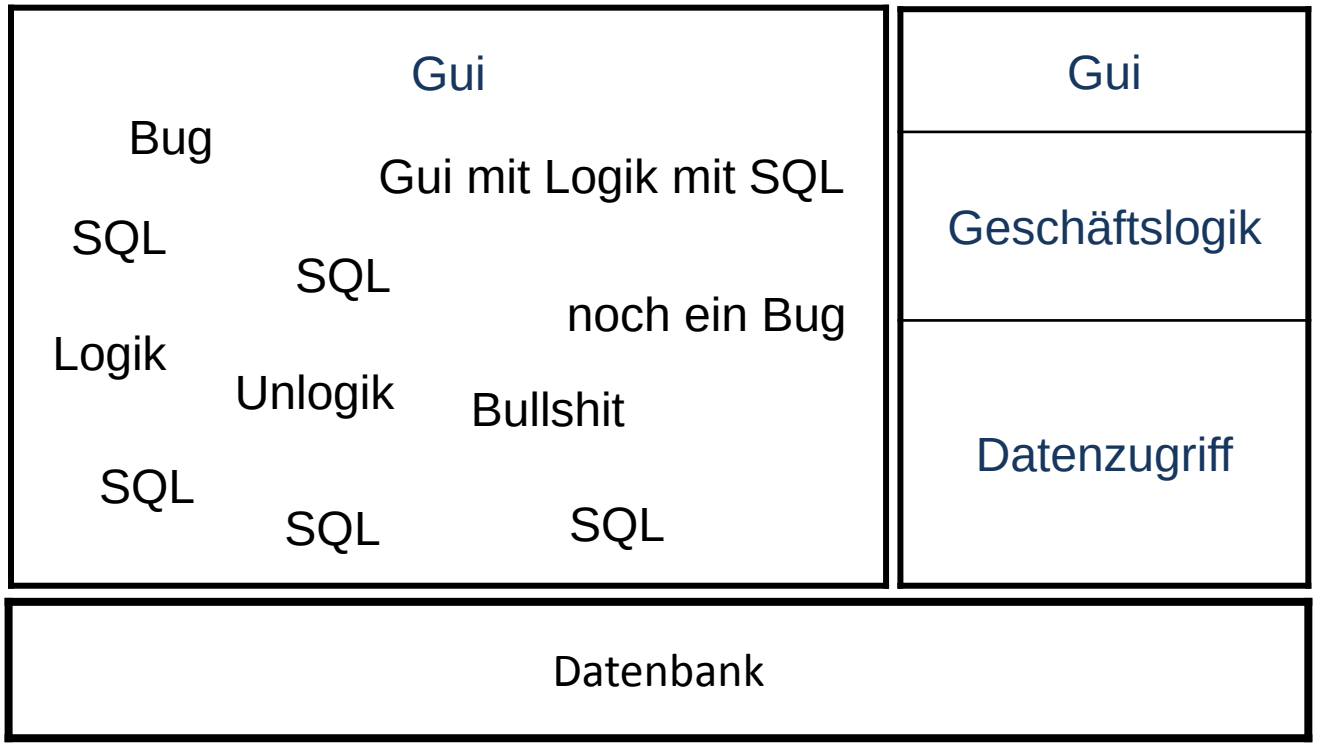
Gui-Tests

- für die wichtigsten Geschäftsvorfälle
- als Sicherheitsnetz
- mit definierten Datenbankständen
- mit (oder ohne) Überprüfung von Ergebnissen

später:

- sukzessive Ablösung durch andere Tests

Struktur und Tests für neuen Code:



Refactoring und (Integrations-)Tests für

- Erweiterungen
- Bugs

Refactoring:

- Code in neue Klassen extrahieren
- Parameter statt globaler Variablen

Refactoring und Tests für

- Risikobereiche aus Kundensicht
- wtfs aus Entwicklersicht: http://engineering.intenthq.com/2015/03/what-is-good-code-a-scientific-definition/

Fragen?