goto; conference aarhus

# Unit Tests Are Not Enough
## Practical Functional Testing in Java

**Martin Klose**
**klose brothers**

**martin.klose@klosebrothers.de**
**@martinklose**

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

gotocon.com

# Agenda

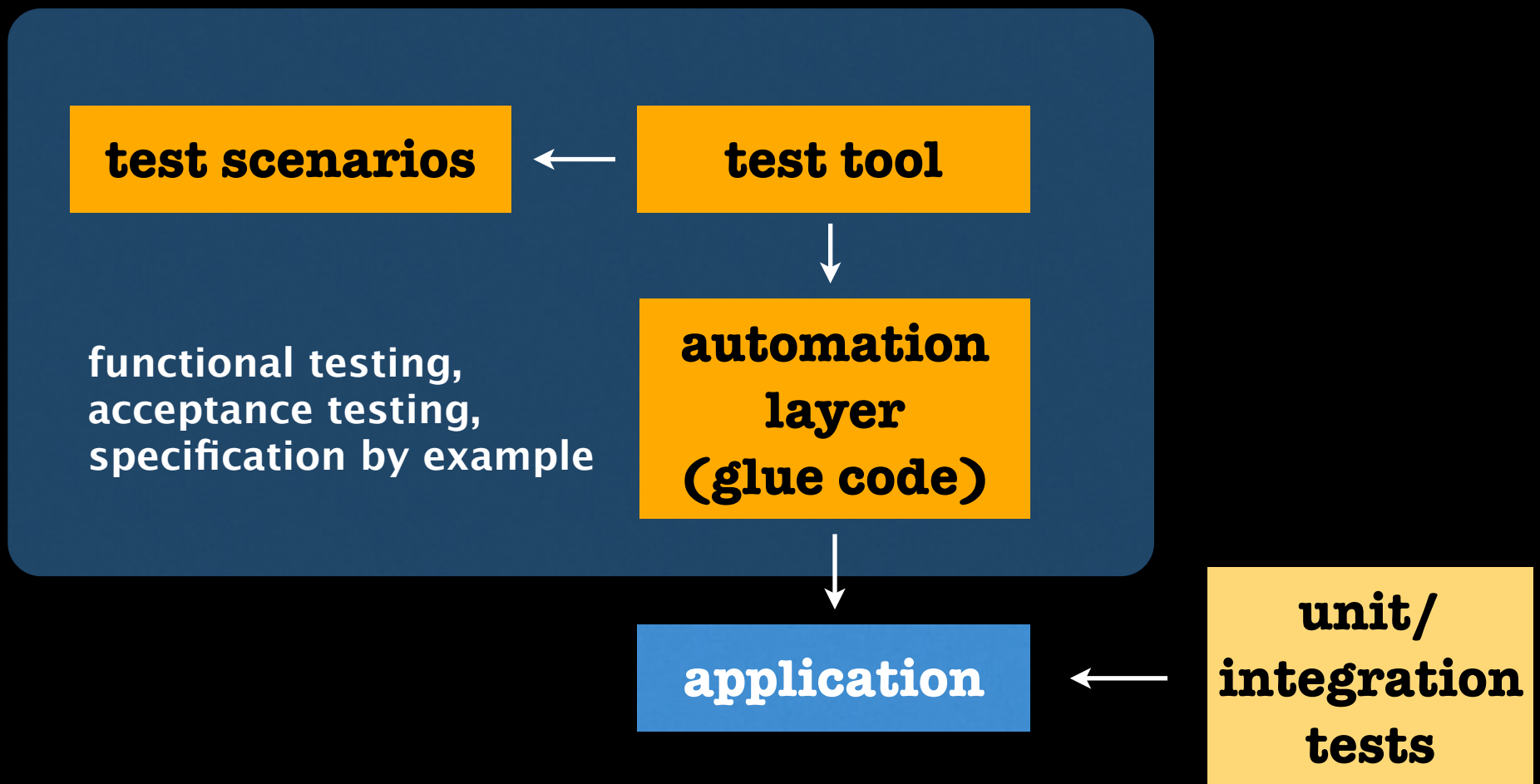
- Why unit tests are not enough
- Characteristics of good tests scenarios
- The vision: an Executable Specification
- The tools we can use
- Tool demo
- Tips for functional testing
- What can go wrong

# A Fairy Tale

- Little John and the Air Balloons

# Customer not satisfied

# What's covered



# Test scenarios should...

- speak the customer's language
- cover only the important aspects
- be complete and realistic examples
- be self explaining
- **NOT** be scripts!

# Test scenarios should...

- speak the customer's language
  - Uses business terms
  - About business functionality not SW design
  - Ideally customer should be able to extend and change
  - Used for communication and clarification
    - Leading to a shared understanding
    - Spot inconsistencies

# Test scenarios should...

- cover only the important aspects
  - Hide technical and implementation details
  - Only business related aspects
  - Do not cover all edge & corner cases
    - We still have testers
    - We have lots of unit tests, too

# Test scenarios should...

- be complete and realistic example
  - Combinations of parameters and expected outputs
  - Using real data helps

# Test scenarios should...

- be self explaining
  - Understandable for users with domain knowledge
  - All & only important information
  - Explaining underlying logic & business rules
  - Make default values explicit when relevant for business logic

# Test scenarios should...

- **NOT** be scripts
  - Describe **what** instead of **how**
  - Don't make the user work backward from single steps to understand what is illustrated
  - Scripts will cost a lot of time in the long term
    - Difficult to understand
    - Reason for test failure is difficult to find
    - Higher maintenance, UI & Workflows change more often than business rules

# The vision: an Executable Specification

- Frequently validated
- Reliable information about system functionality
- Easy to read and understand
- Well organized and easy accessible
- Updated when system functionality changes

# Tools for functional testing should allow...

- to formulate test scenarios in the appropriate abstraction/language
- to connect to the system under test at any layer or API
- to use the VCS of your choice
- to be easily integrated into your CI/ Build

# Tools to consider

- Keyword- / table-centric frameworks
  - FitNesse, Robot Framework, Twist
- Behaviour-Driven-Development tools
  - Cucumber, JBehave, EasyB
- Free text tools
  - Concordion

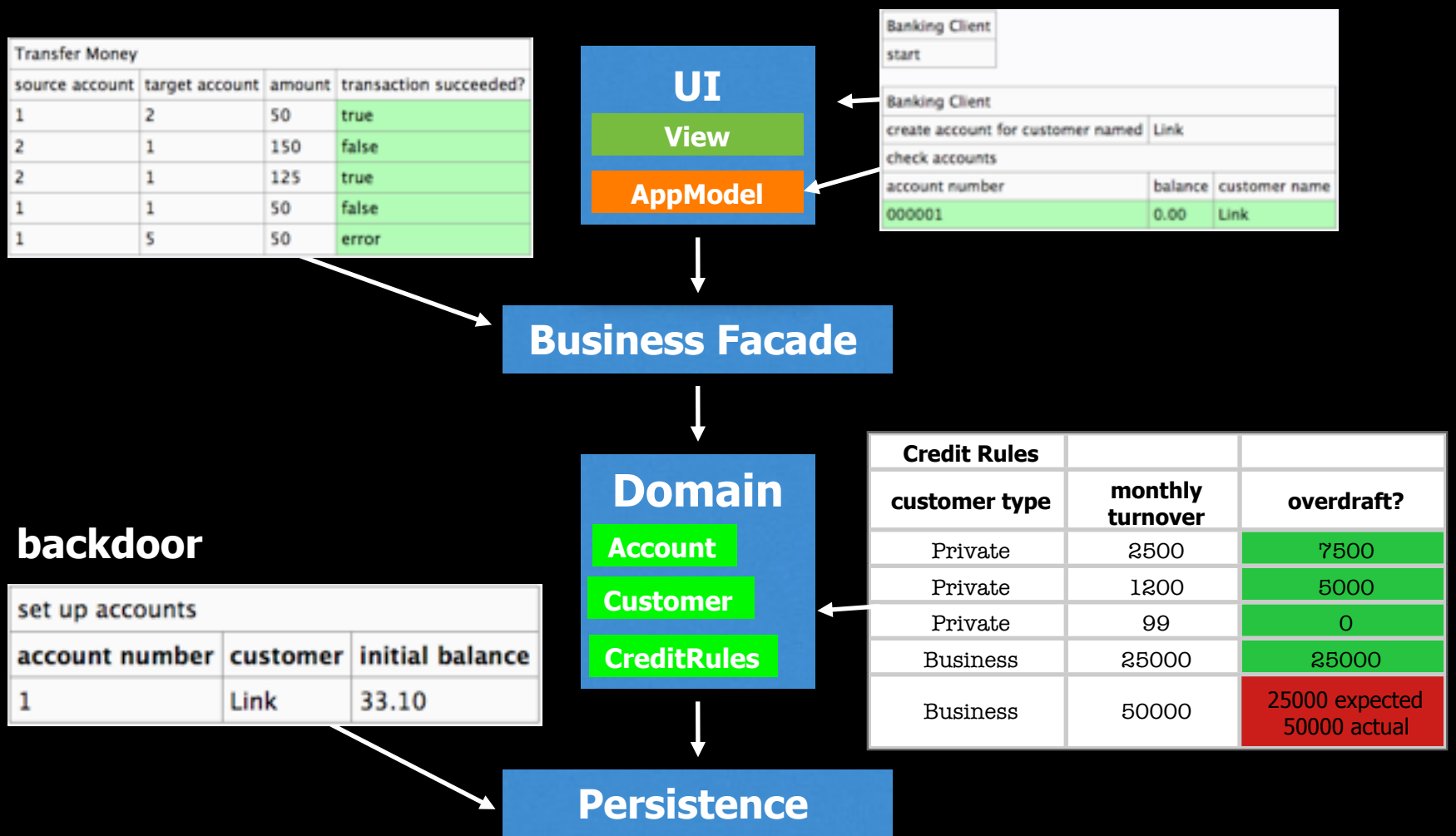
# Demo

- Source code is on github:
  - https://github.com/mklose/goto2011aar

# Hide technical details

- Use the testing tool's set up / tear down / abstraction mechanisms sparingly
- Write dedicated fixture code which does the complicated set up

# Choose the appropriate "point of attac"



# Rules for Automation through the User Interface

- Rule 1: Never automate through the UI
  - UI changes more often => Brittle tests
  - UI automation is technically complex
  - UI automation is slow

# Rules for Automation through the User Interface

- Rule 2: Only automate through the UI in case of emergency
  - Sometimes necessary to build customer trust in test automation
  - Abstract the UI away
  - Be aware of what you're missing out on

# Decouple from systems not under your control

- You cannot reliably include external (test) systems in your regression testing
- Use an additional set of tests for checking real collaboration with external systems
  - Usually started by hand

# Organization of Tests

- Goals
  - I want to see the state of current features at a single glance
  - I want to organize all test scenarios in a consistent manner
- How to reach both goals
  - Organize test scenarios for current work by stories/features
  - Reorganize regression tests by functional area

# What can go wrong ...

- No close collaboration between developers, testers & domain experts
- Abuse of acceptance tests as replacement for thorough unit testing
- Neglecting clean up, refactoring and rework of acceptance tests

➡ It's all about communication!

**Given** a birthday balloon with a face on it
**When** you hang it to the ceiling
**Then** the face is smiling friendly

# Nodes


- Kudos goes to @johanneslink
  - shirt http://605644.spreadshirt.de/

- global-day-of-coderetreat **Sat, Dec 3, 2011**
  https://github.com/coreyhaines/coderetreat/wiki/Cities
  - Bielefeld*, Germany too
    *and yes it exists!

# References


- Gojko Adzic: Specification by Example,How Successful Teams Deliver the Right Software
- Cunningham & Mugridge: FIT for Developing Software: Framework for Integrated Tests