codecentric

Emergentes Design mit TDD

Clean Code Days

David Völkel

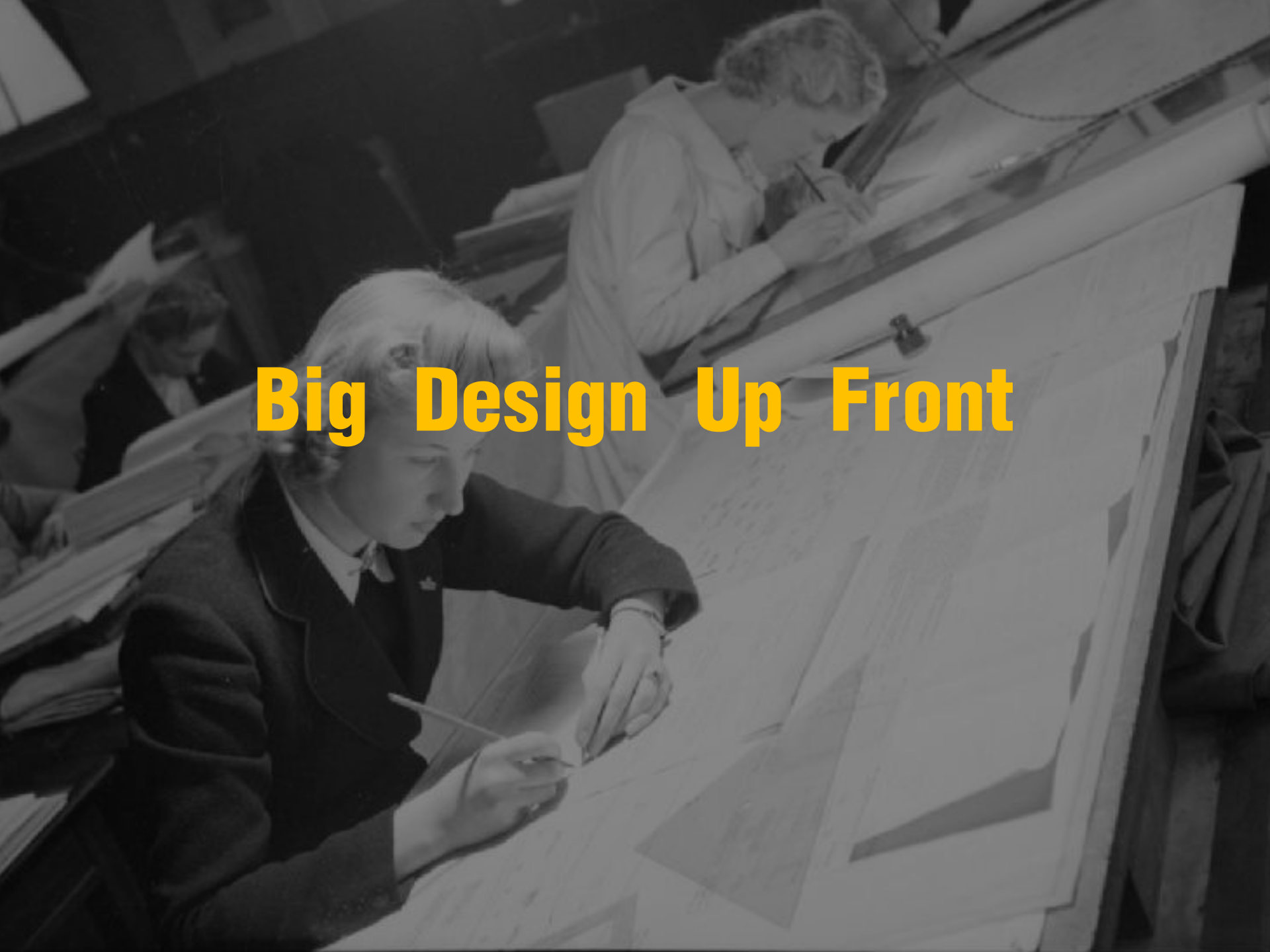
04.11.2015

@davidvoelkel

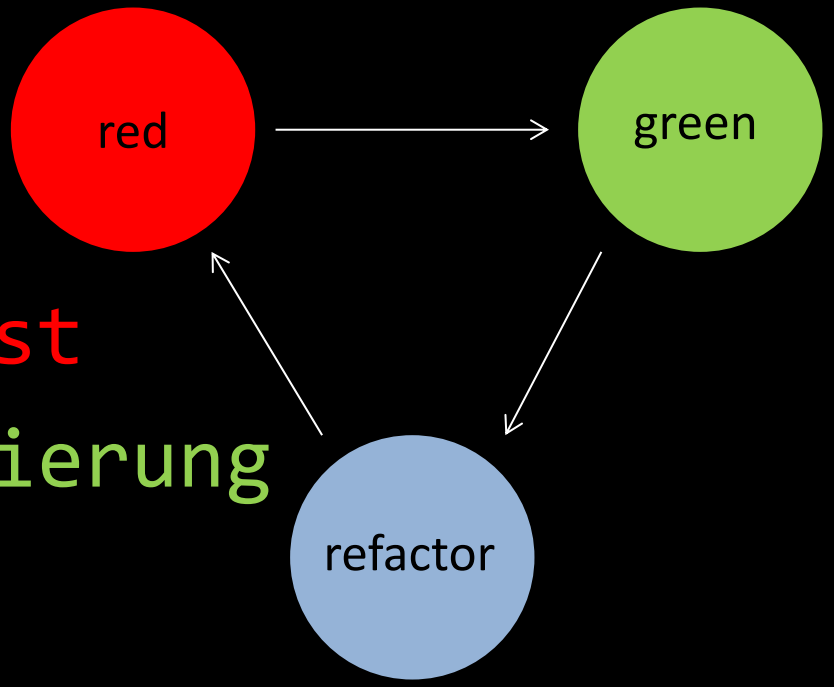
TDD & Design

@codecentric

@softwerkskammer


Big Design Up Front

TDD



Fehlschlagender Test

Minimale Implementierung

Refactoring

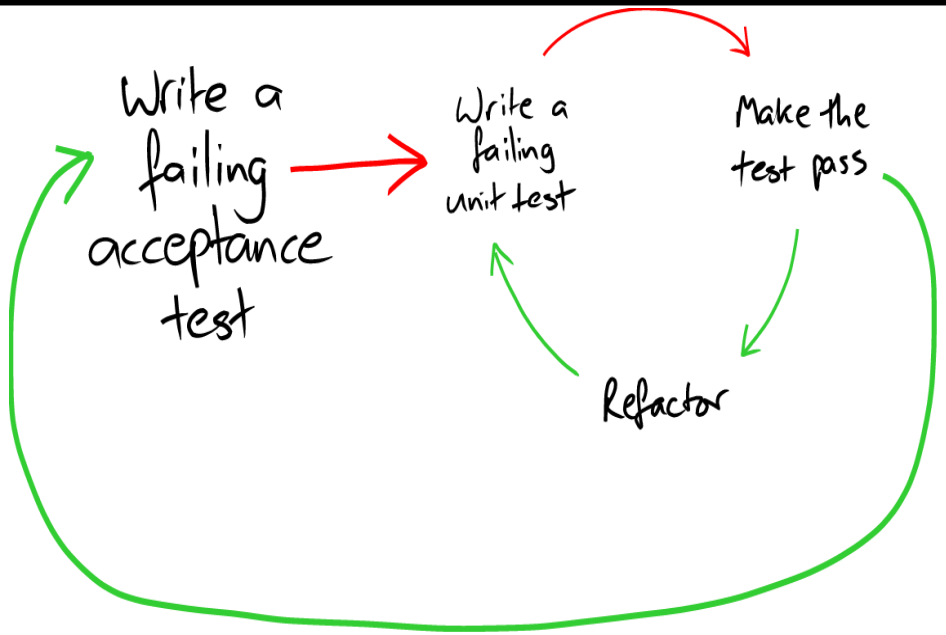
Emergentes Design

Entsteht kontinuierlich

London School TDD

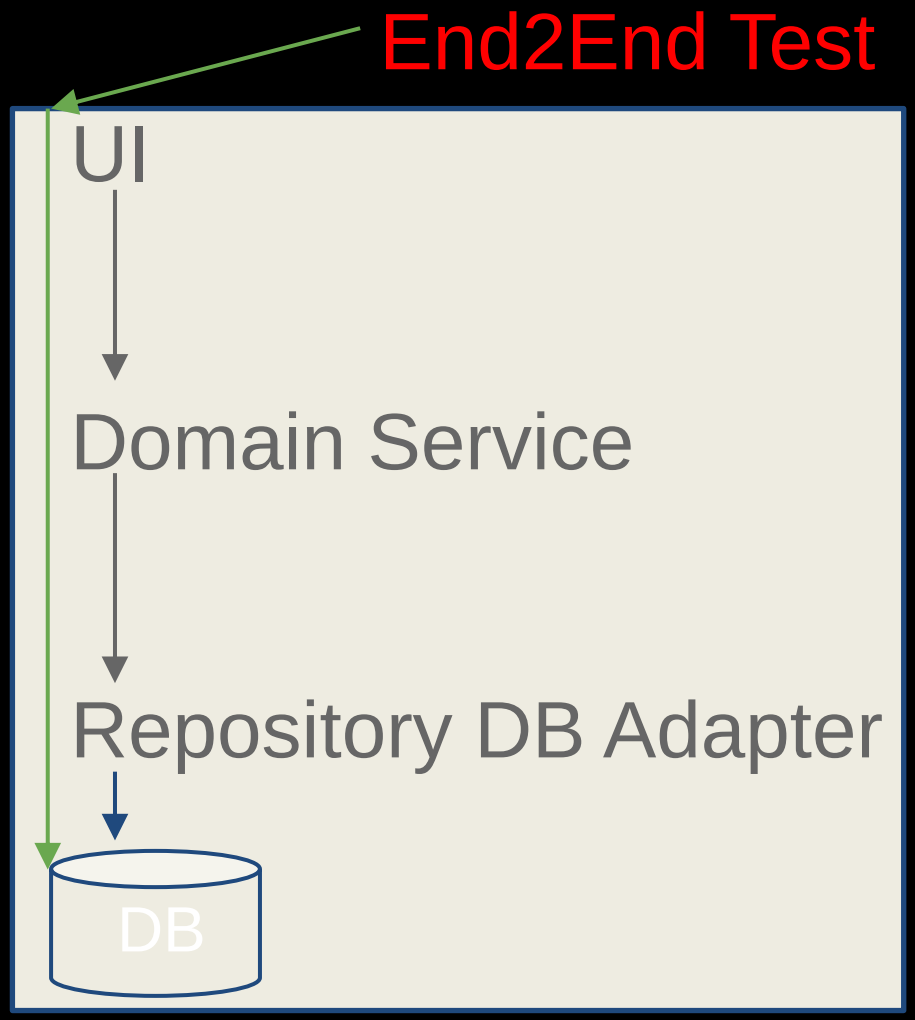
„Mockists“

ATDD



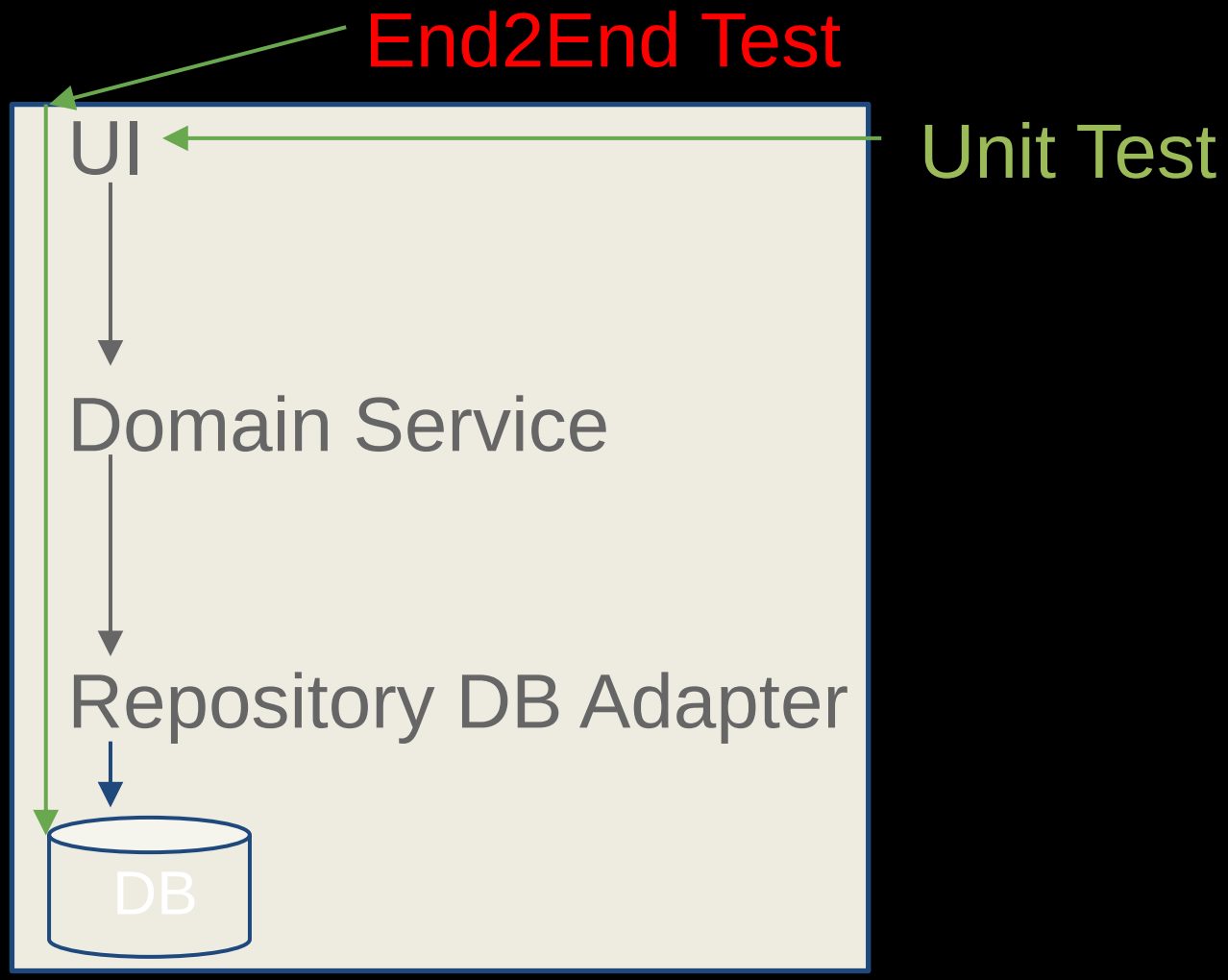
From *Growing Object-Oriented Software* by Nat Pryce and Steve Freeman

London School TDD

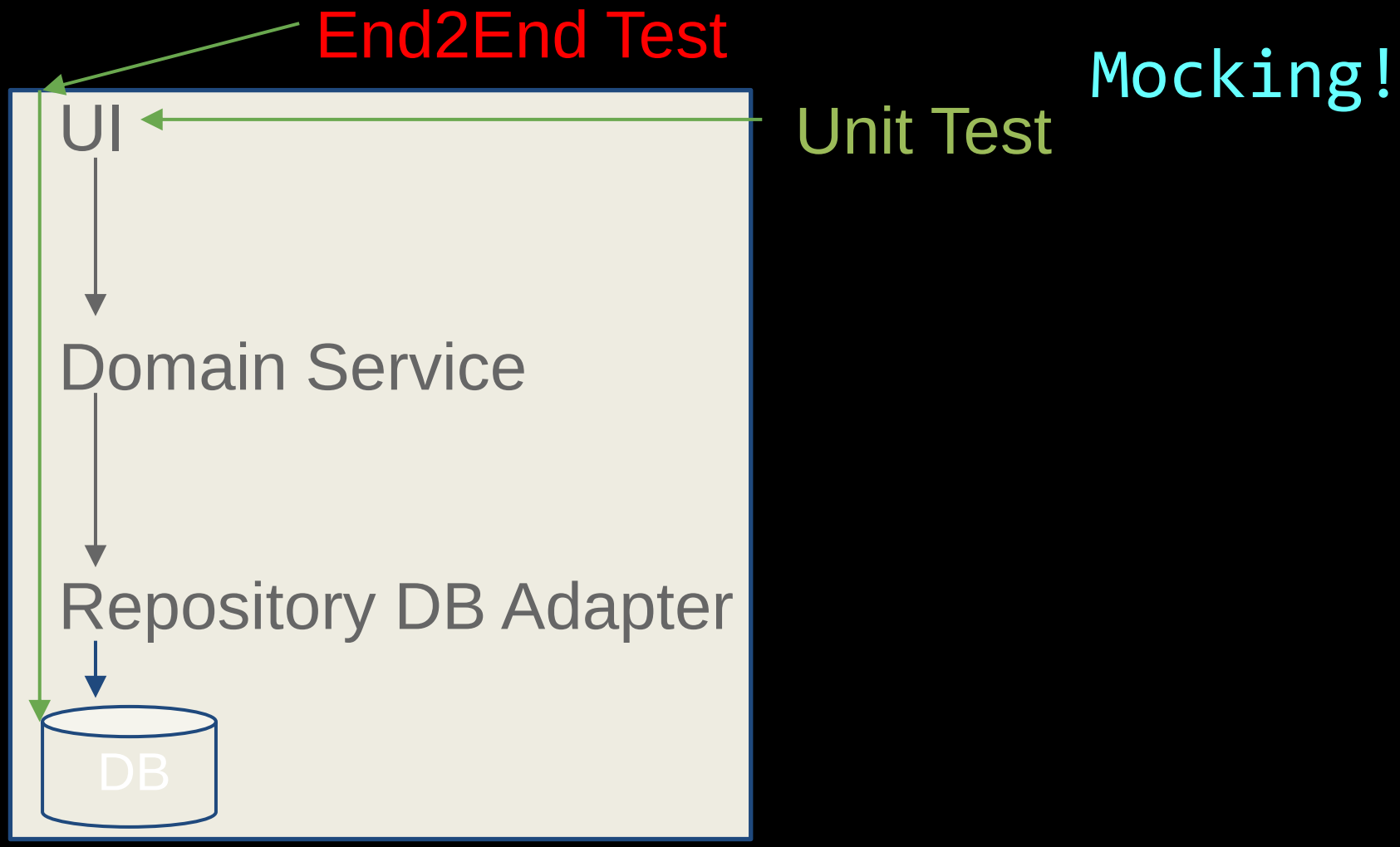


Outside-In Design

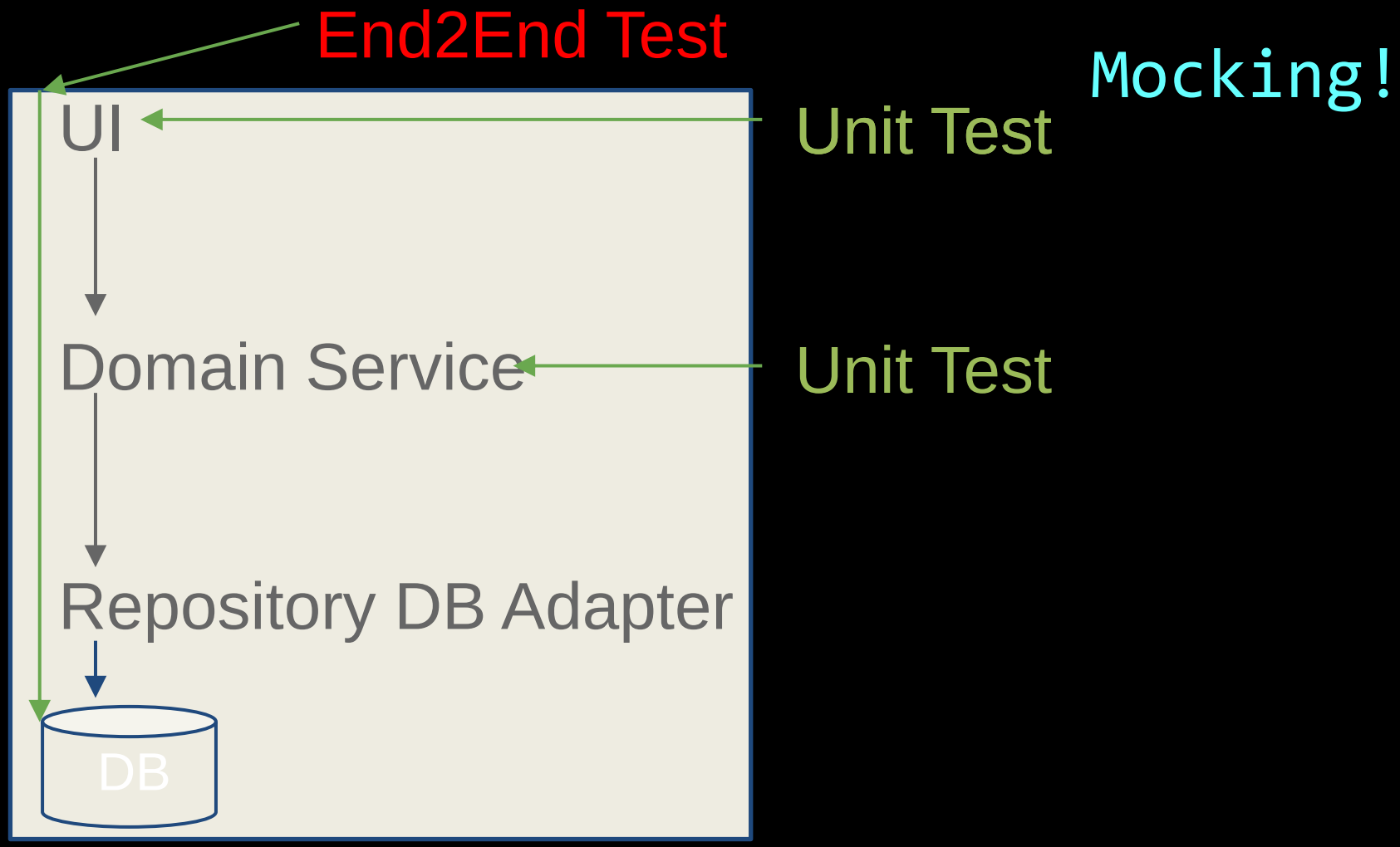
London School TDD



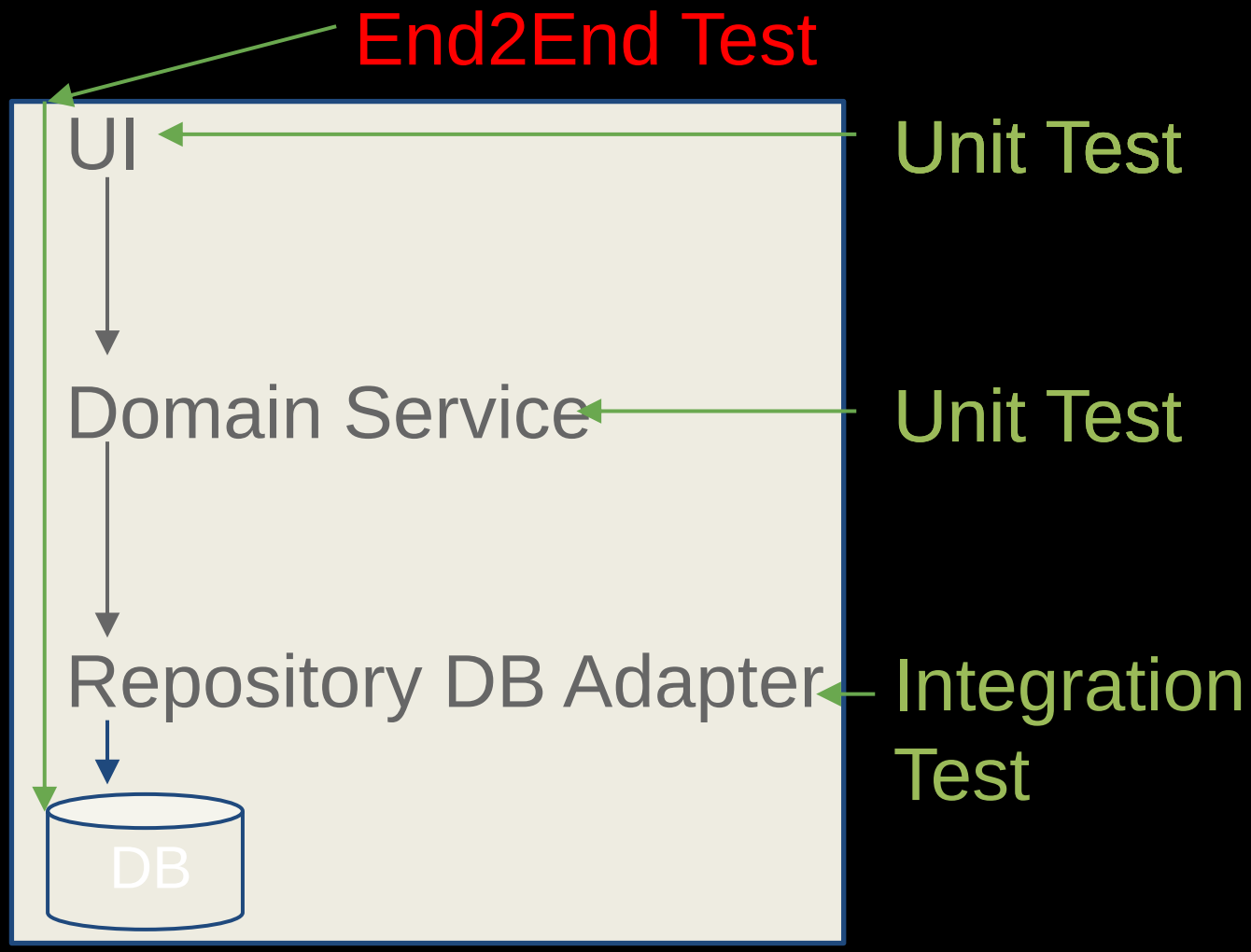
London School TDD



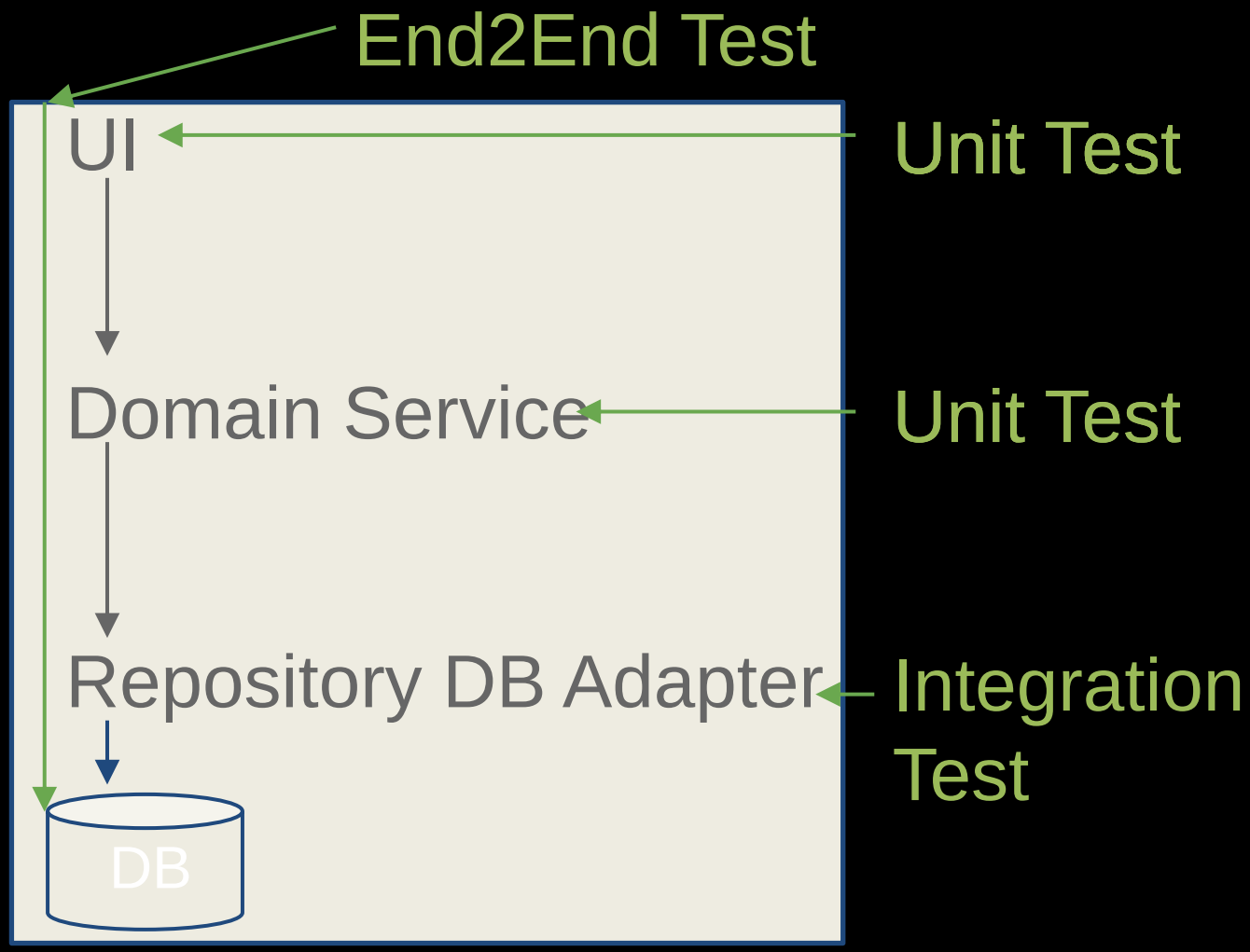
London School TDD



London School TDD



London School TDD



Detroit School TDD

„Classicists“

Mocks nur an Systemgrenzen

(Meist) Inside-Out/Bottom-Up

4 Rules of Simple Design

1. Pass all Tests
2. Clear, Expressive & Consistent
3. No Duplication
4. Minimal Units

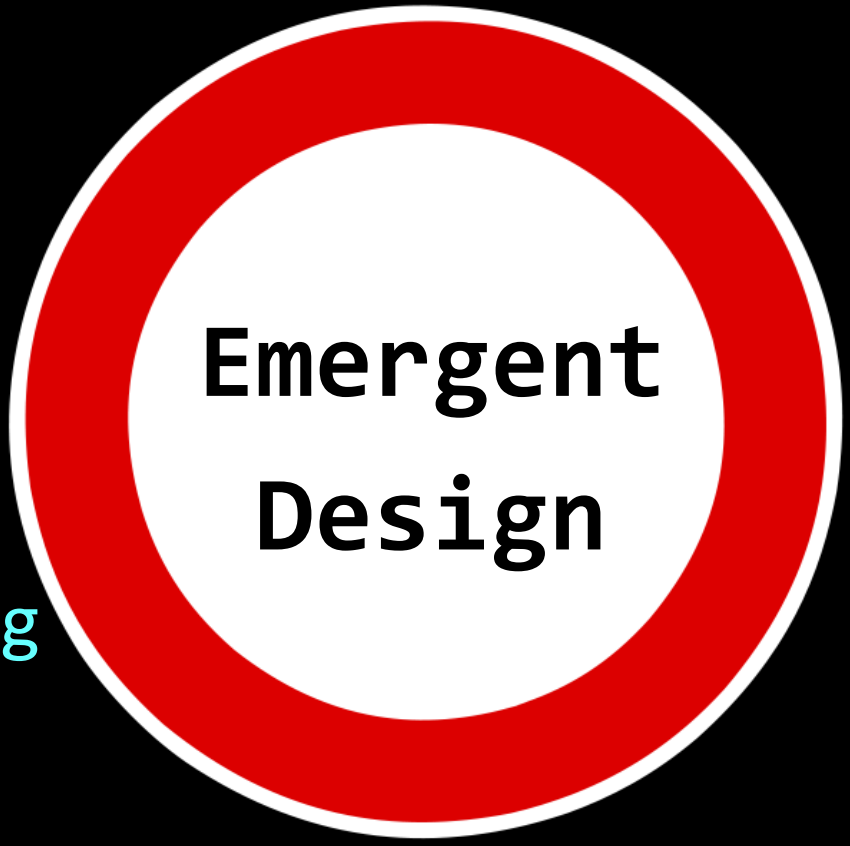
Emergentes Design

Entsteht kontinuierlich

Minimal, aber brauchbar

Gefahren

- Big Picture?
  - Akzeptanztests
- Refactoring
  - Disziplin
  - Designskills nötig



London vs. Detroit Design

Outside-In

Absehbarer

Refactoring & Tests Schreiben

Inside-Out

Unbekannter, „emergenter“

Refactoring

codecentric

Fragen und Diskussion

Lizenz