Make legacy software testable again

Daniel Krämer, M. Sc.
anderScore GmbH

TOPCONF
Duesseldorf

anderScore
trust in competence

About the speaker

anderScore trust in competence

Daniel Krämer (M.Sc. C.S.)

- Software Engineer
- Interests
  - Java
  - Web engineering
  - Migration and integration
  - Software architecture
  - Clean code
  - Refactoring
  - Testing
- JEE, Spring, Wicket, SOA, …

Table of contents

anderScore
trust in competence

1. Living with legacy 4

2. Refactoring to the rescue 8

3. Development workflow 12

4. Refactoring techniques 14

5. Conclusion 19

1. Living with legacy

- *'Valuables'* inherited from '*ancestors*'

- Grown code base
- Developers no longer present
- Still to be maintained
- Shifting responsibility
- Not replaceable easily

- Often: issues with code quality…

1. Living with legacy

anderScore trust in competence

- Past: **Code & Fix**
  - Source code
    - Badly designed
    - Hardly understandable
    - Poorly documented and tested
    - Modified on occasion
  - Issues in production
    - Recurring
    - Hard to reproduce
    - Difficult to analyze
    - Urgent and expensive to sort out

1. Living with legacy

- Present: ‘**Fear Driven Development**’
  - Application in fact non-maintainable
  - New features to be implemented
  - Bugs to be fixed
  - Butterfly effects
  - Increasing demands
  - “… but don’t you break anything!”
  - Pain for each developer’s everyday life

1. Living with legacy

- Future: **???**

2. Refactoring to the rescue

- Future: **Continuous Refactoring !**
  - Improve understandability
    - Meaningful names
    - Logical structuring
    - Pragmatic documentation
  - Improve code quality
    - Dead code
    - Code smells
    - Patterns and conventions
    - Error handling

2. Refactoring to the rescue

anderScore trust in competence

- Future: **Continuous Refactoring !**
  - **Re-enable testability**
    - Isolate code
    - Refine interfaces
    - Introduce mocks
    - Test every part you touch
    - Unit tests first, integration tests later
    - Goal: continuous growth of test base
  - **Re-enable maintainability**
    - Understand and check implementation
    - Have fun implementing new features
    - Verify desired behaviour

© Raimond Spekking / Wikimedia Commons, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2693604

2. Refactoring to the rescue

anderScore trust in competence

- “Our customers do not pay for refactoring!”
  - … but they pay a lot more for maintenance and new features!
- “The project is going to end in some months anyway!”
  - … but a long maintenance phase is about to follow!
- “Apply Test Driven Development!”
  - … but the application is not testable!
- “Do black box tests at high level!”
  - … but high level (integration) testing can be very complex …
  - … and it is hard to think of all implications!

2. Refactoring to the rescue

- Requirements
  - Collective code ownership
    - **Tested** and **reviewed** code only
    - No private property
    - No single points of knowledge
    - No slave drivers
  - Tool support
    - IDE, refactoring tools
    - No change in business logic
    - Minimal effort possible

3. Development workflow

anderScore trust in competence

Planning

1. Consider additional time for refactoring and testing

Preparation

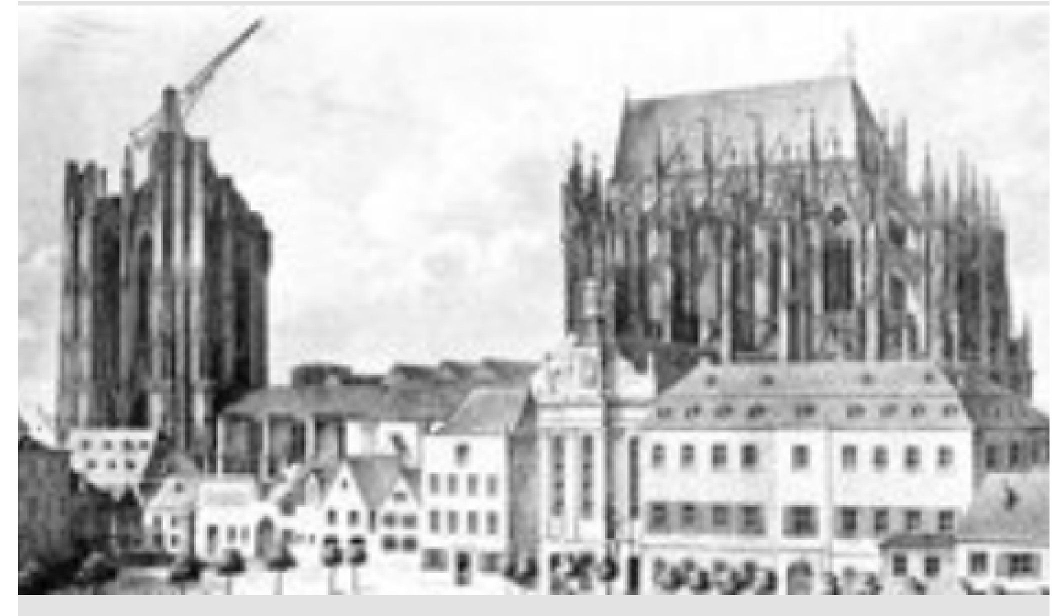
2. Do structural refactoring only
3. Write manageable regression tests
4. Do logical refactoring
5. Verify tests
6. Do code review

Development and testing

7. Add new features
8. Verify and extend tests

4. Refactoring techniques

- Extract code
  - Better understanding
  - ➢ Enables testing in isolation
  - ➢ Enables (partial) mocking
- Apply more functional programming style
  - Explicit interfaces
  - Reduction of unexpected side effects
  - ➢ Enables black box testing
- Reduce external dependencies
  - As much data as possible as input parameters
  - ➢ Enables mocking of input data
  - ➢ Enables multiple test scenarios



4. Refactoring techniques

anderScore trust in competence

- Rethink visibility
  - Methods should be self-contained
  - Multiple interfaces
  - ➢ Visible methods easier to test
- Apply Integration Operation Segregation Principle
  - ➢ Operation: normal UnitTest
  - ➢ Integration: UnitTest with verification (mocks)
- Introduce layers
  - Unfolding spaghetti code
  - Single level of abstraction
  - ➢ Independent testing
  - ➢ Mocking on multiple levels

Example: before

```
private int getMyElementsForTheCalculations (int key2, String key1, List<Customer> list, int mode) {

    // 1. Build SQL
    // ...

    // 2. Execute database call
    // ...

    // 3. Validate input parameters
    // ...

    if (error == 78) {
        return 4711;
    } else if (error != 321 && mode != 445 && Date.isToday("MO")) {
        return -7;
    }
    return 1337;
}
```

anderScore trust in competence

Example: after

```
public List<Customer> getCurrentCustomers (String name, int zipCode) {

    // 1. Validate input parameters
    validateNameAndZipCode(name, zipCode);

    // 2. Build SQL
    String sql = createCustomerSelectSql(name, zipCode);

    // 3. Execute database call
    List<Customer> customers = database.query(sql);

    return customers;
}
```

5. Conclusion

anderScore
trust in competence

- **Problem:** maintaining legacy software as ‘Fear Driven Development’
- **Solution:** continuous refactoring on *design level*
- **Goal:** make legacy software testable (and maintainable) again

- **Requirements:** collective code ownership, tool support
- **Realization:** integration into development workflow
- **Means:** refactoring techniques

- **Limits:** issues related to architecture or technology

Further reading

anderScore
trust in competence

- Refactoring Legacy Code: http://refactoring-legacy-code.net
- Refactoring Catalog: https://refactoring.com/catalog
- Refactoring Guru: https://refactoring.guru
- Mockito: http://site.mockito.org

- Fowler, et. al.: Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional
- Robert C. Martin: Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall

References

- Pictures: Bing image search (several sources; Creative Commons)

Daniel Krämer **www.anderscore.com**

anderScore GmbH

daniel.kraemer@anderscore.com

TOPCONF
Duesseldorf

anderScore
trust in competence