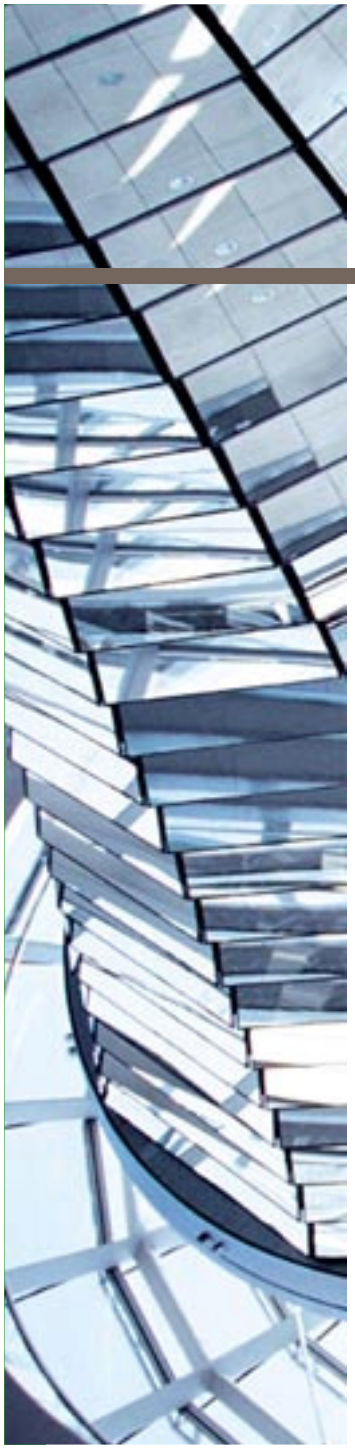


11 Design Patterns and Refactoring

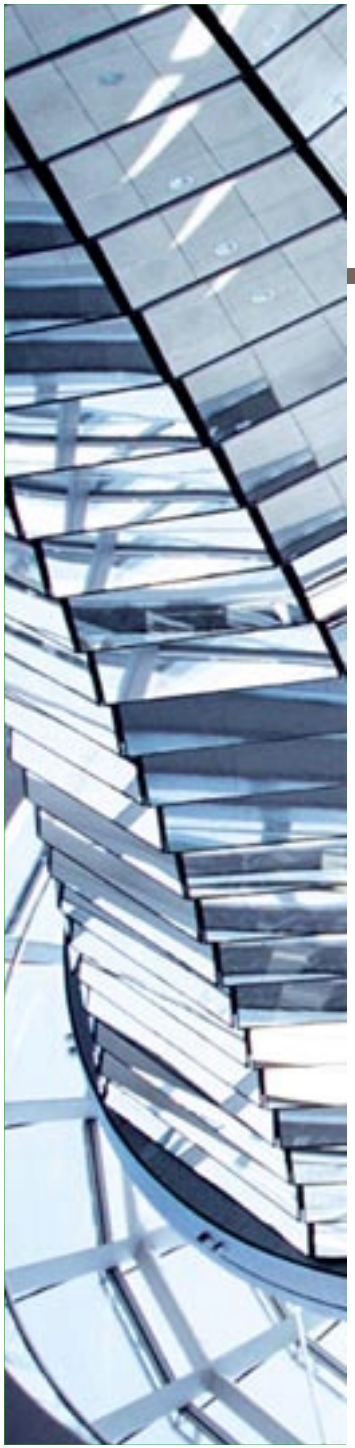
Reference Architectures and Patterns

Winter Semester 2008 / 2009
Prof. Dr. Bernhard Humm
Darmstadt University of Applied Sciences
Department of Computer Science

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

The lecture in the context of the entire course

1. Introduction
2. A reference architecture for business information systems
3. Application kernel
4. Persistence and transaction
5. Authorization
6. Client architecture
7. Exception handling
8. Business Intelligence
9. Systems integration
10. Service-oriented architecture
11. Design patterns and Refactoring
12. Design for testability



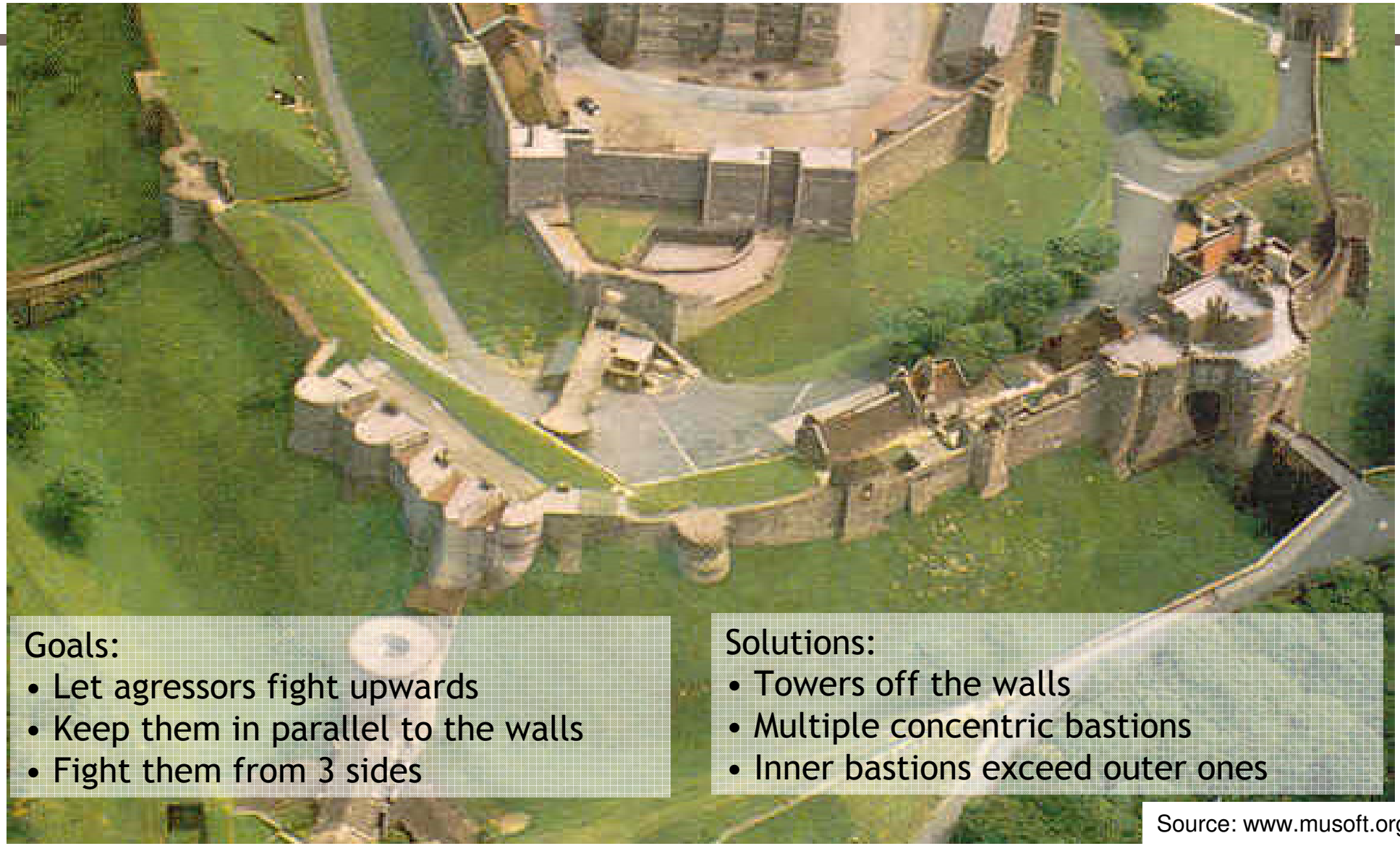
Agenda

➔ **Design Patterns**

Refactoring

Literature

Reminder: Patterns in building architecture



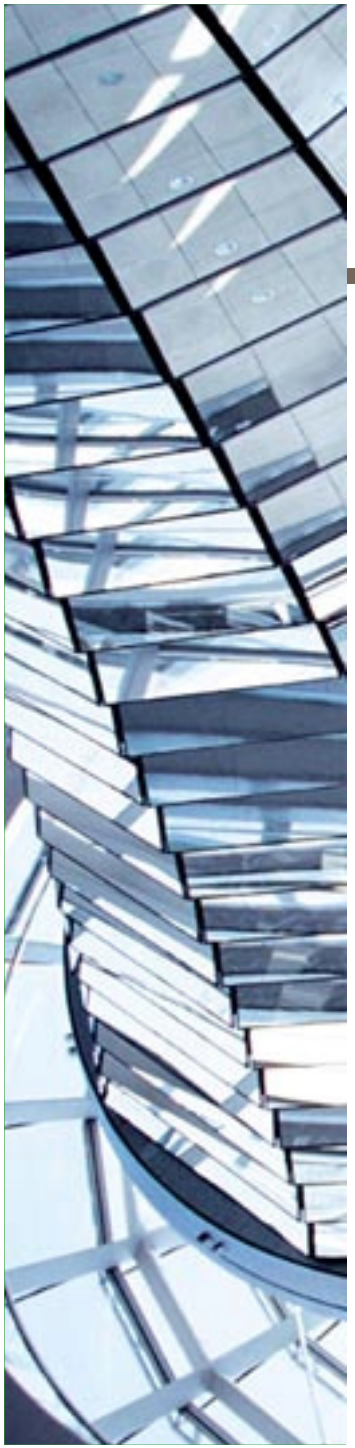
Goals:
- Let agressors fight upwards
- Keep them in parallel to the walls
- Fight them from 3 sides

Solutions:
- Towers off the walls
- Multiple concentric bastions
- Inner bastions exceed outer ones

Source: www.musoft.org

Selected Design Patterns

- Use Slides from Bob Tarr, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County
  http://userpages.umbc.edu/~tarr/dp/spr06/cs446.html
- Introduction To Design Patterns
- Some OO Design Principles
- The Observer Pattern
- Factory Patterns
- The Composite Pattern
- The Adapter Pattern



Agenda

Design Patterns

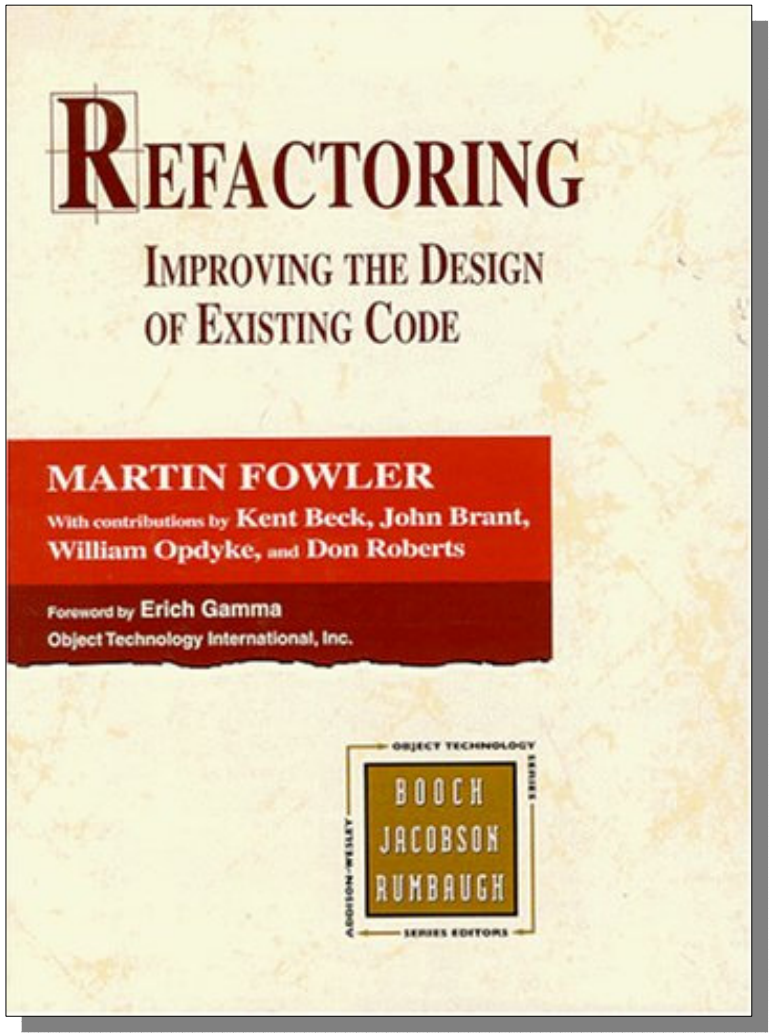
➔ **Refactoring**

Literature

Any *fool* can write code that a *computer* can understand.

Good *programmers* write code that humans can understand.

Martin Fowler: *Refactoring*



Bad Code

```
#include <stdio.h>
#define y for(c=0,i=54;i>=0;c=(s[i]+=a[i]+c)/11,s[i]=s[i]%11,i--);
#define z for(i=0
main(){char s[55],a[55];int c,i,d;for(;;){z;i<55;s[i++]=0);for(;(d=getchar())!='\n';)
(d=d>58?(d>77?d-87:d-55):d-48;z;i<55;a[i]=s[i],i++);y z;i<54;s[i]=s[i+1],i++);
s[54]=0;y z;i<53;a[i++]=0);a[53]=d/11;a[54]=d%11;y)z;i<54&&s[i]==0;i++);
for(;i<55;i++)printf("%c",s[i]<10?s[i]+48:32);printf("\n");}}
```

Winner of the "20C7: 20th Chaos Communication Congress Compact C Coding Contest"
http://www.ulm.ccc.de/old/shortest/20c7/

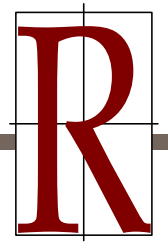
What is the semantic of this program?

Solution: see
http://www.ulm.ccc.de/old/shortest/20c7/aufgabe.html

Catalog for refactorings by Martin Fowler: http://www.refactoring.com/catalog/

- Add Parameter
- Change Bidirectional Association to Unidirectional
- Change Reference to Value
- Change Unidirectional Association to Bidirectional UPDATED
- Change Value to Reference
- Collapse Hierarchy
- Consolidate Conditional Expression
- Consolidate Duplicate Conditional Fragments UPDATED
- Convert Dynamic to Static Construction *by Gerard M. Davison* NEW
- Convert Static to Dynamic Construction *by Gerard M. Davison* NEW
- Decompose Conditional
- Duplicate Observed Data
- Eliminate Inter-Entity Bean Communication *(Link Only)*
- Encapsulate Collection
- Encapsulate Downcast
- Encapsulate Field
- Extract Class
- Extract Interface
- Extract Method
- Extract Package *by Gerard M. Davison* NEW
- Extract Subclass
- Extract Superclass
- Form Template Method
- Hide Delegate
- Hide Method
- Hide presentation tier-specific details from the business tier *(Link Only)*
- Inline Class
- Inline Method
- Inline Temp
- Introduce A Controller *(Link Only)*
- Introduce Assertion
- Introduce Business Delegate *(Link Only)*
- Introduce Explaining Variable
- Introduce Foreign Method
- Introduce Local Extension UPDATED
- Introduce Null Object
- Introduce Parameter Object UPDATED
- Introduce Synchronizer Token *(Link Only)*
- Localize Disparate Logic *(Link Only)*
- Merge Session Beans *(Link Only)*
- Move Business Logic to Session *(Link Only)*
- Move Class *by Gerard M. Davison* NEW
- Move Field
- Move Method UPDATED
- Parameterize Method UPDATED
- Preserve Whole Object
- Pull Up Constructor Body
- Pull Up Field
- Pull Up Method
- Push Down Field
- Push Down Method
- Reduce Scope of Variable *by Mats Henricson* NEW
- Refactor Architecture by Tiers *(Link Only)*
- Remove Assignments to Parameters
- Remove Control Flag
- Remove Double Negative *by Ashley Frieze and Martin Fowler* NEW
- Remove Middle Man
- Remove Parameter
- Remove Setting Method UPDATED
- Rename Method
- Replace Array with Object
- Replace Assignment with Initialization *by Mats Henricson* NEW
- Replace Conditional with Polymorphism
- Replace Conditional with Visitor *by Ivan Mitrovic* NEW
- Replace Constructor with Factory Method UPDATED
- Replace Data Value with Object
- Replace Delegation with Inheritance
- Replace Error Code with Exception
- Replace Exception with Test
- Replace Inheritance with Delegation
- Replace Iteration with Recursion *by Dave Whipp* NEW
- Replace Magic Number with Symbolic Constant UPDATED
- Replace Method with Method Object UPDATED
- Replace Nested Conditional with Guard Clauses
- Replace Parameter with Explicit Methods
- Replace Parameter with Method
- Replace Record with Data Class
- Replace Recursion with Iteration *by Ivan Mitrovic* NEW
- Replace Static Variable with Parameter *by Marian Vittek* NEW
- Replace Subclass with Fields
- Replace Temp with Query UPDATED
- Replace Type Code with Class UPDATED
- Replace Type Code with State/Strategy
- Replace Type Code with Subclasses
- Reverse Conditional *by Bill Murphy and Martin Fowler* NEW
- Self Encapsulate Field
- Separate Data Access Code *(Link Only)*
- Separate Query from Modifier
- Split Loop *by Martin Fowler* NEW
- Split Temporary Variable
- Substitute Algorithm
- Use a Connection Pool *(Link Only)*
- Wrap entities with session *(Link Only)*

Selected refactorings



Composing Methods	Extract Method
Moving Features between Objects	Move Method
Organizing Data	Replace Type Code with State/Strategy Replace Magic Number with Symbolic Constant
Simplifying Conditional Expressions	Replace Conditional with Polymorphism
Making Method Calls Simpler	Separate Query from Modifier Parameterize Method
Dealing with Generalization	Replace Inheritance with Delegation Pull Up Field , Pull Up Method Extract Interface Collapse Hierarchy