Welcome Crappy Code
The Death of Code Quality

ACCU 2009, Oxford

Off-Topic FAQ
Will there be a new Edition of the C++ Library Book and if so when?

- Yes
- Don’t know
Off-Topic FAQ

- Contract signed
  - Goal: Mid 2010
- Got a feature list by Alisdair
- Compared old vector with new vector
- Currently:
  - waiting for feature freeze
  - trying to find compilers
  - wondering about the relevance of concepts

Once Upon a Time we thought

- requirements never change
- systems should be homogeneous
- work is done by experts

These times are gone ...
Darwinism

“It is not the strongest of the species that survive, nor the most intelligent, but the ones most responsive to change.”

Charles Darwin

The Context of the 21st Century

- Market Economy
- Globalization
- Pressure
- Complexity
- Mission impossible
- Chaos

©2009 by IT-communication.com
The Context of the 21st Century

Market Economy
↓
Pressure
↓
Mission impossible

Rule of Global Market Economy

Telco Development Manager according to Jim Highsmith:

"You can rant and rave all you want about software quality (or lack there of), but the marketing guys run the world and they want market share now... Period, end of discussion. My job is to deliver on time on budget, with the ‘appropriate’ quality metrics."
When Controllers Control

- "We can have other Java programmers for a cheaper price"
- "Let's outsource to India"
- "Offer cheap, we will get revenue with support"
- "She might play an important role, but I don't see any real reason to keep her in the project"
- "Let's replace the expensive old guy by a cheap young guy coming from university"
- "And, if you still don't want to deploy a final release due to some obscure quality problem just because Uncle Bob told you so, you can immediately leave the company (and take Uncle Bob with you)"

=> The quality of the average programmer shrinks
The Context of the 21st Century

Globalization

Complexity

Chaos

Distributed Systems

Frontends

Thin Client

Smart Client
(Call Center Desktop)

Mobile Phone / Fax

Processes

Order Fulfillment

Backends

CRM

Billing

Logistics
Distribution over Multiple Companies

Wireless Business Unit

Insurance Company

Wired Business Unit

UpdateAddress(…)

“We moved houses”
“Our new Address is: …”
Distribution in Reality

- 50-100 people involved
  - A&D, development, QA&Test
  - operation, incident management
- For each release:
  - Multiple requirements
  - for different customers
  - with different systems involved
- For example:
  - new Attribute
  - new functionality
  - process modification
  - system replacement
  - company merger
  - outsourcing

System Development

- Maintenance of System Landscapes
- Multi-Project Management
Comunication/Management Overload

Many people involved

- Solution Manager
- System Analysts
- System Designers
- System Implementers
- System Testers
- Solution Integrators
- Solution Testers
- Delivery Managers
- Incident Managers

Testing Overload

- For proper quality of distributed business processes we face a lot of problems:
  - To test, all systems must be up and running
  - Providing consistent distributed test data
  - Testers can't test a business process as a whole
    - Testers have to verify the desired effects
      (=> “Test process slip”)
  - Any change can have impacts on other business processes
  - Automation (and TDD) is almost impossible
The Satir Change Model

- Originally applied to humans
- but due to my experience also true for systems

Chaos Overload

- Replacing a System
- New Release
- New Major Feature
- Outsourcing
- New Release
- Bugfix
- New Major Feature
- Replacing a System
- Bugfix
Let's Summarize

- **We have**
  - Pressure due to market economy
  - Improving complexity
  - Chaos & Overload

- **So, we don’t and won’t have**
  - time for quality
  - money for quality
  - support for quality

- **And it gets worse and worse**

- **Obviously, code quality is not possible**

- **We can’t win the fight for quality**
  as a general goal

- **Therefore:**
  - Accept that bad code quality is common
  - React accordingly
How to Deal with Bad Code Quality?

- Praise Copy&Paste
- Refactor asynchronously
- Use experts carefully
- ...

Versioning without Copy&Paste
Praise Copy&Paste

- The simplicity of Copy&Paste strikes everything else

- Common rules do not apply:
  - Three strikes and you automate
  - Three strikes and you refactor
  - DRY (don't repeat yourself)

- Therefore
  - fight for triviality
  - fight against self-fulfillment
  - fight against CV-driven software
  - fight for the boring way of programming
Asynchronous Refactoring

- We separate:
  - “Feature Teams” implement changes (quick&dirty)
  - “Refactoring Teams” clean up asynchronously

Feature Teams:
- make it run
- make it fast
- deploy

Refactoring Team:
- make it right
- deliver

Release

Use Experts Carefully

- The few experts you have use them with care
  - Architecture
    - Care for overall picture
  - Design
    - Care for operability
  - Task Forces
    - Care for holding schedules
  - Refactoring
    - Care for maintainability
    - Care for the sources of copy&paste

Welcome the Mob n’ Bucket Brigade
Software Development in the 21st Century

- Disenchanted?
  Welcome to the reality of the 21st century!

- Note, now we only see the effects we already had in other areas:
  - Mass production has replaced quality (clothes, food, ...)
  - Where are the craftsmen like shoemakers or tailors?
  - Doctors and hospitals care for money instead of people
  - ...

Sounds worse than it is, because we currently learn

- Accepting that requirements change => Agility
- Accepting that systems are heterogeneous => SOA
- Accepting that code quality is not possible => “???”