Maintenance and Agile Development:
Challenges, Opportunities and Future Directions

Geir K. Hanssen
NTNU and SINTEF ICT

Aiko Fallas Yamashita
Simula Research Laboratory
University of Oslo

Reidar Conradi
NTNU

Leon Moonen
Simula Research Laboratory
Today’s talk is about…

Software entropy and agile methods

Why code smells and refactoring?

Findings from Case Study

Potential solutions (according to our literature review) and avenues for future research

[ simula . research laboratory ]
Software entropy and agile methods

Why code smells and refactoring?

Findings from Case Study

Potential solutions (according to our literature review) and avenues for future research

[ simularesearchlaboratory ]
As software is modified, its disorder or entropy will increase…

[Jacobson, 1992]
...and unmanageable system complexity impacts the organization as well as the quality of the product.
High-pace of agile development imposes demands on the code base and the developers

Agile characteristics:
✓ High change responsiveness
✓ Fast delivery
✓ Customer-driven development

The interplay between agile methodology and software entropy may be different from other projects due to agile’s characteristics.
What are the challenges that software entropy entails when software maintenance is performed under agile methods?
Software entropy and agile methods

Why code smells and refactoring?

Findings from Case Study

Potential solutions (according to our literature review) and avenues for future research
Agile recommends identification of *code smells* and *refactoring* for ensuring maintainability...
Now it is possible to automate code smells detection up to certain degree...

Code Smells detection/visualization
[Van Emden & Moonen, 2002]

Specification of detection strategies for detecting code smells
[Marinescu, 2002]

Commercial tool is available for detecting some code smells
(based on Marinescu detection strategies) - Borland Together [IBM, 2007]

GodClasses := ((ATFD, TopValues(20%)) and (ATFD, HigherThan(4)))
and ((WMC, HigherThan(20)) or (TCC, LowerThan(0.33)))
Agile recommends identification of code smells and refactoring for ensuring maintainability...
Agile recommends identification of code smells and refactoring for ensuring maintainability…

…but tradeoffs need to be made to ensure cost-effective smell detection and refactoring
Agile recommends identification of code smells and refactoring for ensuring maintainability…

…but tradeoffs need to be made to ensure cost-effective smell detection and refactoring
Agile recommends identification of *code smells* and *refactoring* for ensuring maintainability…

…but tradeoffs need to be made to ensure cost-effective smell detection and refactoring.
What information is there to support decisions?
Which refactorings pay off, which ones are risky, ...

Literature survey

What are potential solutions?
What information is there to support decisions? Which refactorings pay off, which ones are risky, ...

What are potential solutions?
What information is there to support decisions? Which refactorings pay off, which ones are risky, ...

What are potential solutions? What are the problems?
What information is there to support decisions? Which refactorings pay off, which ones are risky, …

What are potential solutions?
Which methods, tools and empirically derived knowledge is available for detecting and analyzing code smells and guiding refactoring decisions?

What are the problems?
Which are the most relevant challenges perceived when dealing with software entropy?
What information is there to support decisions? Which refactorings pay off, which ones are risky, ...

What are potential solutions?
Sources: IEEE Xplore, ACM and ISI
Web of knowledge (2000-2008)

What are the problems?
Source: Architects from CSoft
Motivation: Software entropy and agile methods

Findings from Case Study and Literature

Code smells and refactoring for dealing with entropy

Potential solutions and avenues for future research
CSoft: A Norwegian case of EVO

1997
Creative chaos

Waterfall

2003
EVO (Similar to SCRUM)

2 weeks iteration

Iteration planning

Release

2 weeks iteration

Product Backlog

Sprint Backlog

Sprint

Working Increment of the software

[ simula . research laboratory ]
Characteristics of the product displayed high degree of entanglement in the business layer

“The Blob”

12 ASP.NET App
Core assembly
150KLOC C#
144 namespaces entangled, displaying cyclical dependencies
Approx. 160 assemblies
Approx. 450 KLOC
Dependencies on various technologies (COM+, VB6, J#)
Challenges identified from interviews and workshop can be grouped into four areas:

- **Analyzeability and Comprehensibility**
- **Modifiability and Deployability**
- **Testability and Stability**
- **Organization and Process**
Analyzability and Comprehensibility
Analyzability and Comprehensibility

Steep learning curve
Analyzability and Comprehensibility

Steep learning curve

Lack of documentation
Analyzability and Comprehensibility

Steep learning curve

Lack of documentation

Fear to change code

[simula. research laboratory]
Analyzability and Comprehensibility

Steep learning curve

Lack of documentation

Fear to change code

Duplicated code

[simula, research laboratory]
Modifiability and Deployability
High proportion of Shotgun Surgery

Modifiability and Deployability
High proportion of Shotgun Surgery

Inconsistent update of duplicated code leads to defects

Modifiability and Deployability
High proportion of Shotgun Surgery

Inconsistent update of duplicated code leads to defects

System needs to be always deployed as a whole

Modifiability and Deployability
Testability and Stability
Test coverage is too low due to high functional complexity

Testability and Stability

[ simula, research laboratory ]
Test coverage is too low due to high functional complexity

Tests are inconsistent, complex and hard to maintain

Testability and Stability
Tests are inconsistent, complex and hard to maintain

Test coverage is too low due to high functional complexity

Corrective task effort is high for each modification/refactoring

Testability and Stability
Tests are inconsistent, complex and hard to maintain

Test coverage is too low due to high functional complexity

Corrective task effort is high for each modification/refactoring

Unforeseeable behavior of system increases “The fear”

Testability and Stability

[simula.research.laboratory]
Strong reliance on “The Gurus”

Organization and Process
Strong reliance on “The Gurus”

High “velocity” does not allow to catch all issues
Strong reliance on “The Gurus”

High “velocity” does not allow to catch all issues

Areas of concern defined by the architects is not reflected in the structure of the system
Motivation: Software entropy and agile methods

Code smells and refactoring for dealing with entropy

The CSoft Case Study: Methodology and Findings

Potential solutions and avenues for future research
Analyzability and Comprehensibility

Semi-automatic code inspections
[Van Emden & Moonen, 2002]

Visualization analyses
[Parnin, 2008] [Trifu, 2007] [Van Der Brand, 2007]

Untangle Crosscutting Concerns
[Moonen, 2008]
Analyzezability and Comprehensibility

Semi-automatic code inspections
[Van Emden & Moonen, 2002]
Visualization analyses
[Parnin, 2008] [Trifu, 2007] [Van Der Brand, 2007]
Untangle Crosscutting Concerns
[Moonen, 2008]

Testability and Stability

Improving unit test suits
[Guerra, 2007] [Van Deursen, 2002] [van Rompaey, 2007]
Empirical evidence on error-smells
[Li & Shatnawi, 2007]
Modifiability and Deployability

Analysis and reduction of module dependencies
[Arevalo, 2005] [Leitch, 2003]

Identify Architectural “smells”
[Bourquin, 2007]
Modifiability and Deployability

Analysis and reduction of module dependencies
[Arevalo, 2005] [Leitch, 2003]

Identify Architectural “smells”
[Bourquin, 2007]

Organization and Process

Aspect Oriented System
[Moonen, 2008]

Some Aspect Oriented tools...
[Binkley, 2006] [Marin, 2009]
Our proposal for agile evolution:
Evolution monitoring and progressive refactoring

- **Code Metrics**
- **Detection Strategies**
- **Experience Companies**
- **Empirical evidence**
- **Context information**
- **Symptoms instances**
- **Refactoring candidates**
- **Tradeoffs guidance**
- **Refactoring**
  - Detection effort
  - Maintainability
  - Risk

Knowledge base

[ simula research laboratory ]
The proposed approach applied on EVO

Evolution analysis
[D’Ambros, 2008]

Evolution patterns
[Jermakovics, 2008]

Refactoring prioritization
[Chaabane, 2007]
[Wasylkowski, 2007]

Empirical evidence

Knowledge base

Code Smell
Detection effort
Maintainability
Risk

Refactoring
Effort
Risk

Continuous process monitoring

[ simula . research laboratory ]
Avenues for future research

Empirical evidence that can help to answer:
 ✓ Which code smells should we refactor first?
 ✓ Which combination of refactorings has the best overall effect?
 ✓ Which are critical code smells and for which situations?
 ✓ What are the potential risks of certain refactorings?

Tools and methods evaluations
 ✓ Industry case studies where tools are used
 ✓ Criteria for evaluating tools and methods for refactoring
 ✓ Usability criteria / requirements for refactoring tools
Thank you!

For collaborations, or if you are just curious about Simula:
http://www.simula.no/research/engineering

Contact info available at:
http://simula.no/people/aiko