Advanced Software Engineering Course

Beautiful vs. Legacy Code

Sadegh Sulaimany

info@Bioinfotmation.ir

دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

SBNA
Social & Biological Network Analysis Laboratory

Initial assessment

1. What are the clean code principals?

Agenda

- Clear code
- Pitfalls of agile

Software Evolution

- Hardware
  - Hardware designs must be declared finished before they can be manufactured and shipped
  - cost of upgrade for hardware is astronomical
  - Hardware may not continue over time

- Software
  - Grow and evolve over time
  - Initial software can easily be shipped and later upgraded over time
  - cost of upgrade for software is affordable
  - software can achieve a high-tech version of immortality, potentially getting better over time

Drivers of Software Evolution

- fixing faults
- adding new features that customers request
- adjusting to changing business requirements
- improving performance
- adapting to a changed environment

- Customers pay annual maintenance fee

- Comparing your work with novelists
  - software engineers should hope their creations would also be long lasting
    - software has the advantage over books of being able to be improved over time

Legacy Software

- **legacy code**
  - › software that, despite its old age, continues to be used because it meets customers' needs

- software maintenance costs

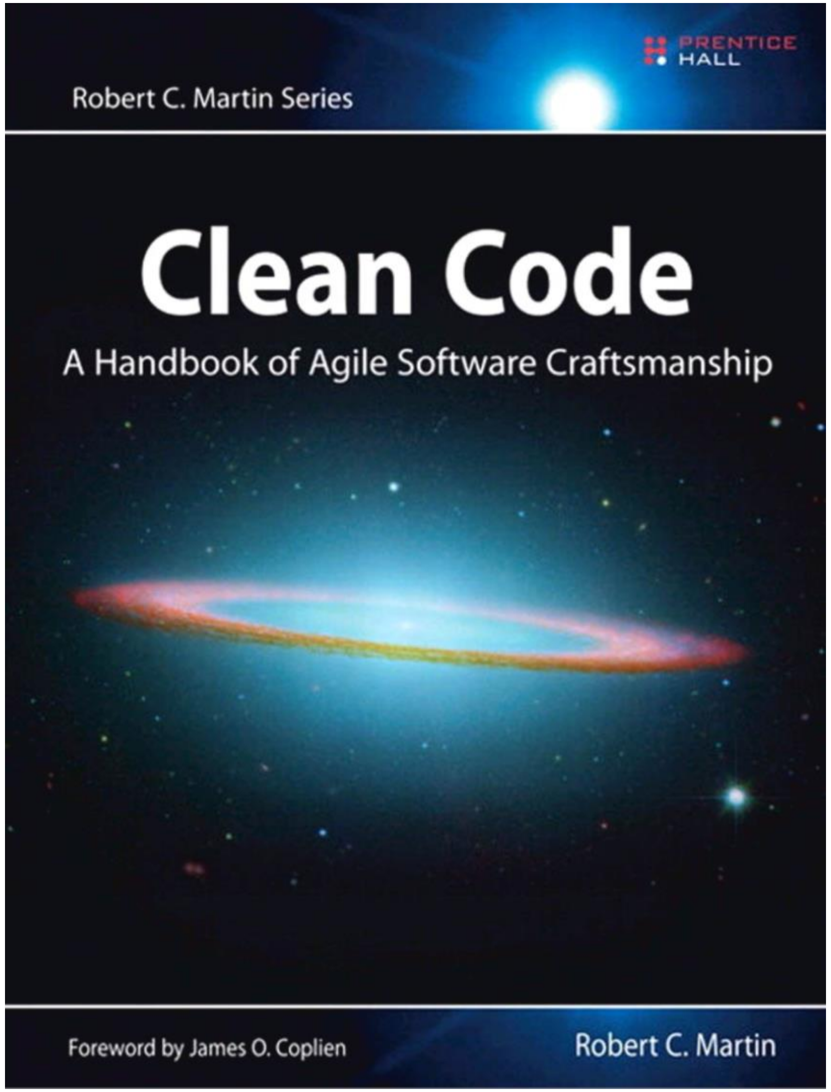
  - adding new functionality to legacy software
    - › 60%

  - fixing bugs
    - › 17%

  - Legacy software is successful software

Clean or beautiful code

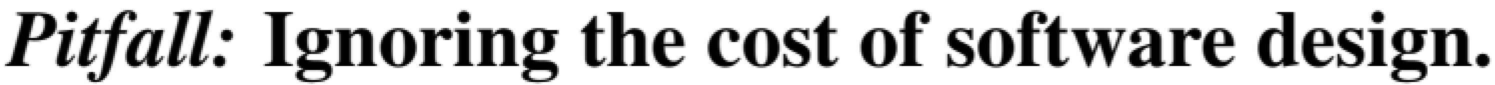
- Vs
  - – unexpectedly short-lived code that is soon discarded because it doesn't meet


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Fallacies and Pitfalls

Fallacy: The Agile lifecycle is best for all software development.

- Agile maybe ineffective producing method for
  - Realtime apps
    - Emergency service dispatch system
    - Fire alarm
  - Projects with strict requirements
    - legal or regulatory projects
  - Projects with predetermined outcomes and timescales
  - Projects with increased risk
    - such as finance

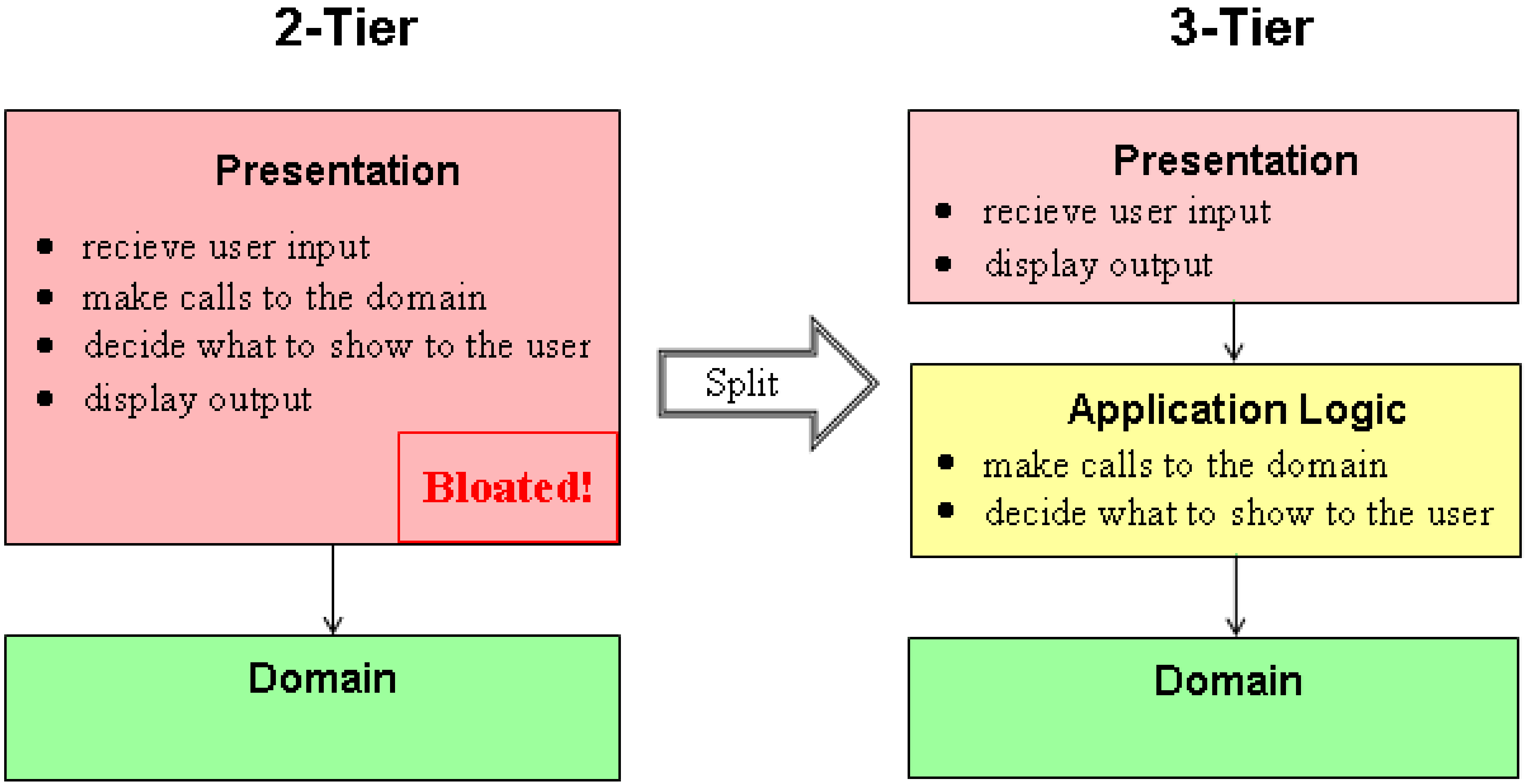


Pitfall: Ignoring the cost of software design.

- Especially for remanufacturing
  - For update
  - For maintenance
  - Design & Test

⚠ *Pitfall:* Ignoring the historical context of software technology.

- Example



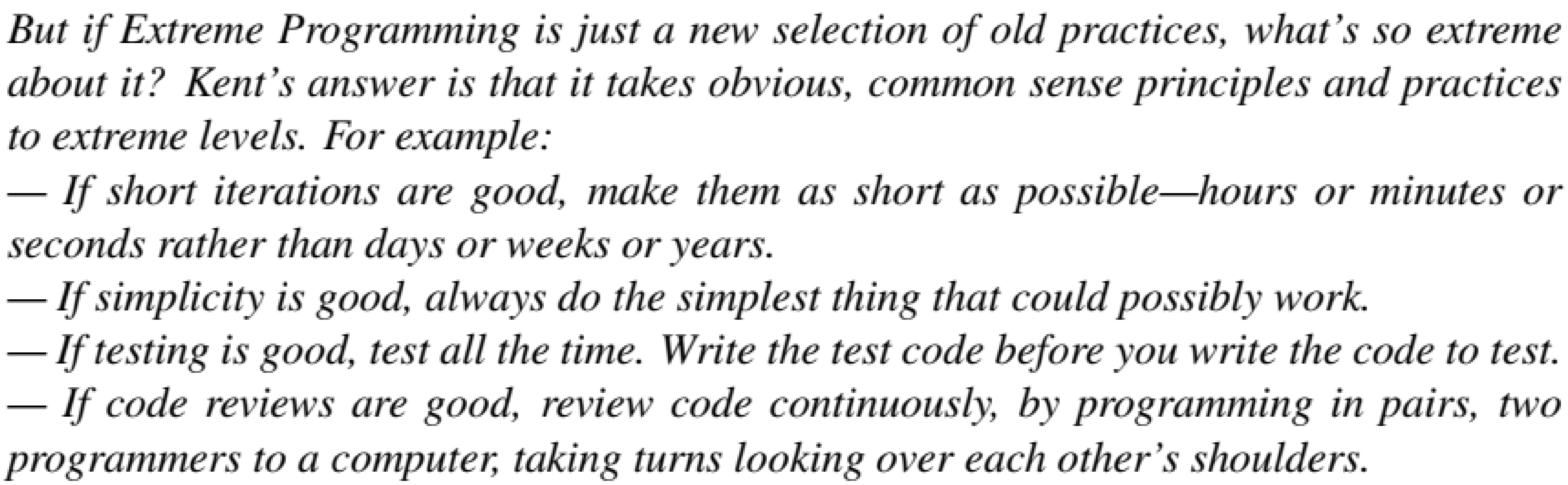
⚠ *Pitfall:* **Being overly focused on learning framework *X* as rapidly as possible.**

- Rapid changes
  - Example
    - Evolution of JavaScript frameworks
      - “hot tech” for building front-end apps has changed from

        Prototype.js
        to jQuery
        to Angular
        to Ember
        to Backbone
        to React, with Vue now another contender

Concluding remarks

- Software Engineering is more than programming
  - Kent Beck



But if Extreme Programming is just a new selection of old practices, what's so extreme about it? Kent's answer is that it takes obvious, common sense principles and practices to extreme levels. For example:

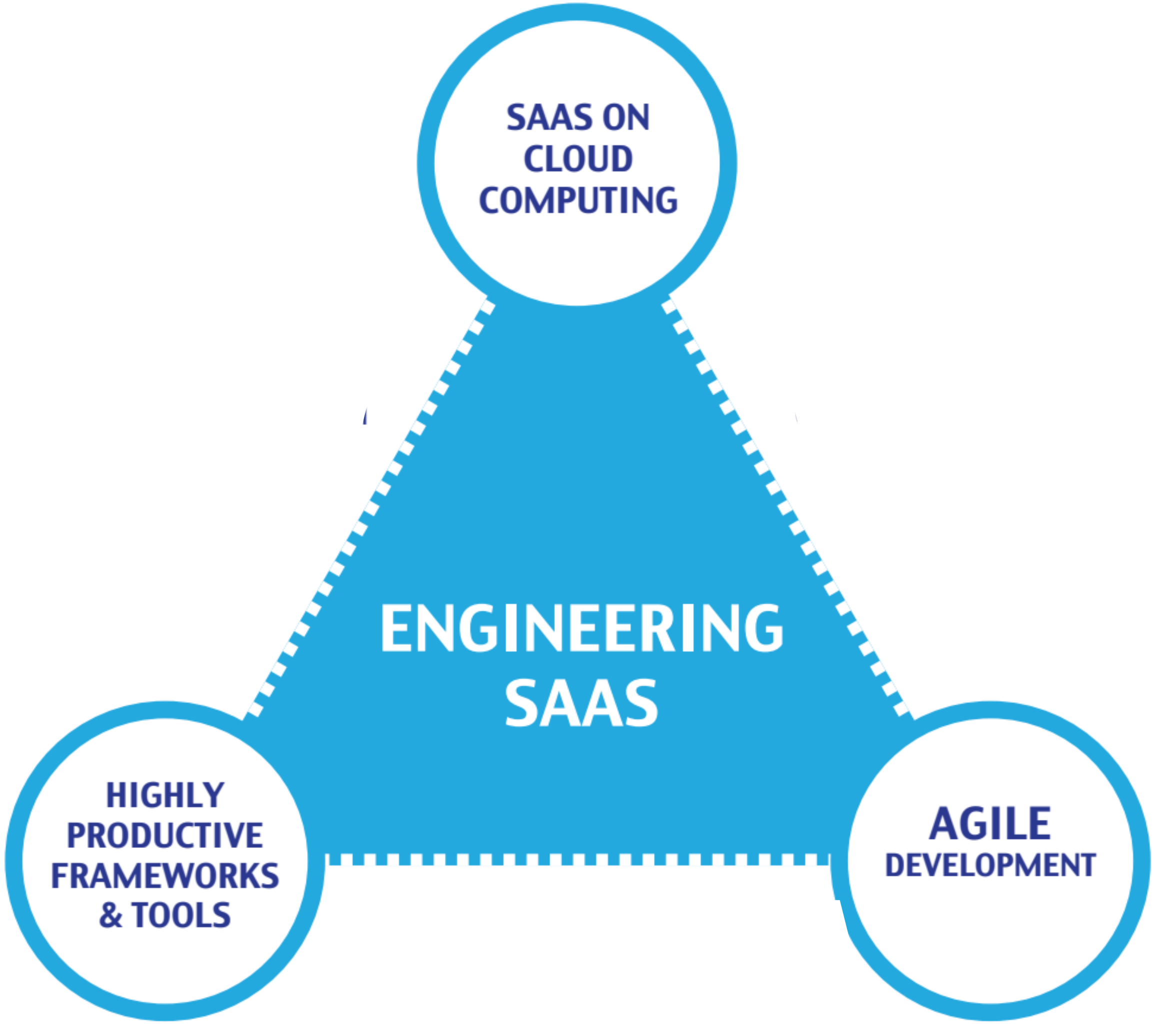
— If short iterations are good, make them as short as possible—hours or minutes or seconds rather than days or weeks or years.

— If simplicity is good, always do the simplest thing that could possibly work.

— If testing is good, test all the time. Write the test code before you write the code to test.

— If code reviews are good, review code continuously, by programming in pairs, two programmers to a computer, taking turns looking over each other's shoulders.

Virtuous triangle



SBNA

Social & Biological Network Analysis Laboratory

Question?

Bioinformation.ir

info@Bioinformation.ir