Software Architecture Quality Case Studies on Open Source Projects

Ural Erdemir & Sinan Eski
@ CodeMR
CONTENT

- Software Design & Principles
- What is software quality?
- Why software quality?
- Technical Debt
- How to measure software quality?
- Case Studies on Open Source Projects
SOFTWARE DESIGN & PRINCIPLES
Readable
Extendable
Maintainable
“Maintenance typically claims 40-80% of all projects cost”

Barry Boehm
• Design Documents
  ○ Outdated
  ○ Incomplete
  ○ Absent

• High rate of turnover among developers
Developers spend 80% of development time reading code!
Funny comments by previous developers

// When I wrote this, only God and I understood what I was doing
// Now, God only knows

// Dear maintainer:
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 42
//

/*
 * You may think you know what the following code does.
 * But you don't. Trust me.
 * Fiddle with it, and you'll spend many a sleepless
 * night cursing the moment you thought you'd be clever
 * enough to "optimize" the code below.
 * Now close this file and go play with something else.
 */
the source code determines what the actual design really is

software tools would be available that post processed a source code design and generated the auxiliary documentation.
Good vs Bad Design
Design Principles & Patterns

SOLID
YAGNI
DRY

GoF DESIGN PATTERNS
"a code smell is a surface indication that usually corresponds to a deeper problem in the system".

Martin Fowler
Object-Oriented Metrics in Practice
by Michele Lanza, Radu Marinescu
WHAT IS SOFTWARE QUALITY AND WHY DO WE NEED IT?
- Functionality
- Performance
- Reliability
- Usability...

- Coupling
- Cohesion
- Complexity
- Design Structure
- Testability
- Reusability
- ...

[Image of an iceberg with the tip above water and the bulk below water, symbolizing the concept of software development where the visible part represents the functional aspects and the submerged part represents the underlying complex design and code.]
Rio de Janeiro. 3am, 2 of February, 1998 a building called Palace II starts to crumble. The building was finished less than 3 years before that. 44 apartments were completely destroyed, and 8 people died.

Many experts said that the concrete had **too much water and contained beach sand**. Also, they said that there wasn't an engineer overlooking the construction, and that **there was basically no quality control**.
Relation Between Design Quality & Change Cost
REASONS
WE'LL ASK FOR ESTIMATES

AND THEN TREAT THEM AS DEADLINES
“When, due to constraints, I design *quickly and dirty*, my project is loaded with *technical debt*”

Ward Cunningham, 1992
Technical Debt

Martin Fowler
1 October 2003

You have a piece of functionality that you need to add to your system. You see two ways to do it, one is quick to do but is messy - you are sure that it will make further changes harder in the future. The other results in a cleaner design, but will take longer to put in place.

Technical Debt is a wonderful metaphor developed by Ward Cunningham to help us think about this problem. In this metaphor, doing things the quick and dirty way sets us up with a technical debt, which is similar to a financial debt. Like a financial debt, the technical debt incurs interest payments, which come in the form of the extra effort that we have to do in future development because of the quick and dirty design choice. We can choose to continue paying the interest, or we can pay down the principal by refactoring the quick and dirty design into the better design. Although it costs to pay down the principal, we gain by reduced interest payments in the future.
Technical debt
Technical Debt
BROKEN WINDOW THEORY

James Q. Wilson and George Kelling
“Prevention is better than cure”

Desiderius Erasmus

The cost of a defect rises significantly the later in the process it is discovered.
HOW TO MEASURE SOFTWARE QUALITY?
“You can't control what you can't measure”
Tom DeMarco
Internal Quality Attributes

- Coupling
- Size
- Cohesion
- Complexity
**Software metrics** are quantitative measures of the degree to which a software component possesses a given attribute.
- Design Patterns have positive impact on metrics
- Creational Patterns
  - Singleton, Abstract Factory, Builder, Object Pool...
- Structural Patterns
  - Adaptor, Bridge, Facade, Composite...
- Behavioral Patterns
  - Observer, Strategy, Visitor, State...

- Code Smells/Anti-patterns have negative impact
  - Duplicate Code, Long Methods, Large Class
  - Long Parameter List, Divergent Change
  - Shotgun Surgery, Data Class, Refused Bequest
  - Dead Code...
Software Metrics

- Lines of Code (LOC),
- Number of Classes/Entities
- Number of Fields (NOF)
- Number of Methods (NOM)
- Lack of Cohesion in Methods (LCOM)
- Cohesion Amongst Methods (CAM)
- Weighted Methods per Class (WMC)

- Response For a Class (RFC)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOC)
- Coupling Between Object Classes (CBO)
- Afferent/Efferent Coupling
- ...
A LOT OF METRICS
AMBIGUOUS
BIG / SMALL VALUE
THRESHOLD
Why Visualization?

- Limited Time, Limited Resources
  - Which are the critical parts?
  - Where to start improvement?
- An integrated representation of complex and multi-dimensional data
  - Intuitive
  - Catchy and memorable
- Easy to perceive and comprehend
  - Lets users to see the big picture
  - Brings metrics and relationships together
Representation

- For all metrics in charts:
  - Coupling
  - Complexity
  - Size

Diagram:

- NODE REPRESENTATION
- COMPLEXITY
- COUPLING
- SIZE
Measure Code Quality
See the Big Picture
Visualization of the Relations
CASE STUDIES
Apache Kafka
Apache Kafka
JUnit
| Name                  | Complexity | Coupling | Size | Lack of Cohesion | CBD | BRC | SRC | DIT | NDC | WMC | LOC | C/LCC | NOF | NSDF | NOM | N |
|-----------------------|------------|----------|------|------------------|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|-------|-----|-----|
| JUnit4                | low        | low      | low  | low              | 27  | 83  |     |     |     |     |     |     |     |      |     |      |
|                | low-medium | low      | low  | low-medium       | 10 | 24  |     |     |     |     |     |     |     |      |     |      |
|                | low        | low      | low  | low              | 8  | 3   |     |     |     |     |     |     |     |      |     |      |
|                | low        | low      | low  | low              | 3  | 3   |     |     |     |     |     |     |     |      |     |      |
|                | low-medium | low      | low  | low-medium       | 1  | 1   |     |     |     |     |     |     |     |      |     |      |
|                | low        | low      | low  | low              | 1  | 1   |     |     |     |     |     |     |     |      |     |      |
|                | medium-high| low      | low  | medium-high      | 2  | 35  | 1   | 1   | 61  | 106 | 104 | 0   | 0   | 1    | 38  |      |
|                | low-medium | low      | low  | low-medium       | 0  | 3   | 1   | 4   | 4   | 7   | 5   | 0   | 1   | 2    | 1   |      |
|                | low        | low      | low  | low              | 1  | 15  | 3   | 1   | 20  | 49  | 40  | 5   | 3   | 8    | 0   |      |
|                | medium-high| low      | low  | medium-high      | 1  | 19  | 4   | 5   | 0   | 4   | 16  | 11  | 2   | 2    | 4   | 0    |
|                | low        | low      | low  | low              | 1  | 7   | 2   | 1   | 0   | 5   | 14  | 4   | 1   | 0    | 5   | 0    |
|                | low        | low      | low  | low              | 0  | 1   | 0   | 1   | 1   | 2   | 1   | 2   | 0   | 0    | 1   | 0    |
|                | low        | low      | low  | low              | 1  | 2   | 0   | 1   | 5   | 2   | 3   | 2   | 0   | 0    | 2   | 0    |
|                | low-medium | medium-high | low | low-medium-high | 1 | 120 | 55 | 2 | 59 | 132 | 129 | 1 | 0 | 13 | 38 |
|                | low        | low      | low  | low              | 2  | 11  | 3   | 1   | 7   | 19  | 16  | 2   | 0   | 7    | 0   |      |
|                | low        | low      | low  | low              | 2  | 4   | 0   | 1   | 3   | 4   | 5   | 4   | 0   | 0    | 4   | 0    |
|                | low        | low      | low  | low              | 1  | 63  | 42  | 0   | 1   | 6   | 19  | 116 | 116 | 2   | 0   | 22  | 3  |
|                | low-medium | medium-high | low | low-medium-high | 4 | 60  | 29  | 19 | 3 | 59 | 169 | 161 | 1 | 4 | 16 | 13 |
|                | low        | low      | low  | low              | 0  | 6   | 0   | 1   | 0   | 6   | 9   | 6   | 0   | 2    | 6   | 0    |
|                | low        | low      | low  | low              | 2  | 0   | 0   | 1   | 0   | 3   | 6   | 5   | 0   | 0    | 1   | 2    |
|                | low        | low      | low  | low              | 1  | 55  | 26  | 1   | 0   | 23  | 61  | 58  | 2   | 0   | 17   | 0   |      |
|                | low        | low      | low  | low              | 2  | 103 | 27  | 2   | 0   | 30  | 89  | 17  | 1   | 3    | 14  | 4    |
|                | low        | low      | low  | low              | 4  | 107 | 64  | 1   | 0   | 106 | 252 | 251 | 0   | 0    | 1    | 70   |
|                | low        | low      | low  | low              | 3  | 16  | 10  | 1   | 0   | 13  | 26  | 25  | 0   | 0    | 1    | 9    |
|                | medium-high| low      | low  | low              | 1  | 4   | 0   | 5   | 0   | 4   | 13  | 10  | 0   | 1    | 4    | 0    |
Spring
Spring
Spring
Thanks!

Any questions?