Maintaining Architectural Quality in Software Teams

Evelyn van Kelle
@EvelynvanKelle

Yiannis Kanellopoulos
@ykanellopoulos
12 million professional software developers write 120 billion lines of code per year. Of which 15% needs to be changed each year.

Is the rapid digitalization of our modern information society sustainable?
The secret for high quality software: Listen to your people.
The way you architect your software mirrors the way your organization thinks.
Plan versus Reality
Plan versus Reality

What we want (or what we designed)

Website
- Accounts
- Savings
- Loans
- Batch processing
- Storage
- External systems

What we ended up with (or what we implemented)

Website (and data validation)
- Accounts
- Login
- Savings
- Loans
- Pension
- Batch processing
- Storage
- External systems
Where did it go wrong?

- Designed architecture
- Mental models & Communication
- Implemented architecture

@EvelynvanKelle
@ykanellopoulos
The future is all about people.
Top 10 causes of Unhappiness, Categories, and Frequency

<table>
<thead>
<tr>
<th>Cause</th>
<th>Category</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Being stuck in problem solving</td>
<td>Software developer’s own being</td>
<td>186</td>
</tr>
<tr>
<td>Time pressure</td>
<td>External Causes -&gt; Process</td>
<td>152</td>
</tr>
<tr>
<td>Bad code quality and coding practice</td>
<td>External Causes -&gt; artifact and working with Artifact -&gt; code and coding</td>
<td>107</td>
</tr>
<tr>
<td>Under performing colleagues</td>
<td>External Causes -&gt; people -&gt; colleague</td>
<td>71</td>
</tr>
<tr>
<td>Feel inadequate with work</td>
<td>Software developer’s own being</td>
<td>63</td>
</tr>
<tr>
<td>Mundane or repetitive task</td>
<td>External Causes -&gt; Process</td>
<td>60</td>
</tr>
<tr>
<td>Unexplained broken code</td>
<td>External Causes -&gt; artifact and working with Artifact -&gt; code and coding</td>
<td>57</td>
</tr>
<tr>
<td>Bad decision making</td>
<td>External Causes -&gt; Process</td>
<td>42</td>
</tr>
<tr>
<td>Imposed limitation on development</td>
<td>External Causes -&gt; artifact and working with Artifact -&gt; code and coding</td>
<td>40</td>
</tr>
<tr>
<td>Feel inadequate with work</td>
<td>Software developer’s own being</td>
<td>39</td>
</tr>
<tr>
<td>Personal issues – Not work related</td>
<td>Software developer’s own being</td>
<td>39</td>
</tr>
</tbody>
</table>


@EvelynvanKelle

@ykanellopoulos
A higher software product quality leads to:

- The faster implementation of improvements and the solution of defects
- The throughput rate improves by factor 3.5 to 4.0 between 2 and 4 stars

Time needed to implement enhancements

<table>
<thead>
<tr>
<th>Stars</th>
<th>0</th>
<th>7</th>
<th>14</th>
<th>21</th>
<th>28</th>
<th>35</th>
<th>42</th>
<th>49</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time needed to fix bugs

<table>
<thead>
<tr>
<th>Stars</th>
<th>0</th>
<th>7</th>
<th>14</th>
<th>21</th>
<th>28</th>
<th>35</th>
<th>42</th>
<th>49</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: “Faster issue resolution with higher technical quality of software”, Software Quality Journal, 2011
COMMUNICATION

The artifacts of your ideas
COMMUNICATION

If it goes well...

If it goes well...
Feedback Loops
Feedback

Developer(s)

System

Input

Architect
Value Congruence
Non-functionals
Without explicit focus, non-functionals become an afterthought

Ideal approach: continuous understream to ensure non-functionals remain at desired level

Too much pressure on functionality without focus on non-functionals eventually erodes quality and causes entire iterations dedicated to “restoration”, losing the momentum

@EvelynvanKelle

@ykanellopoulos
Software Maintainability
The Vicious Cycle of Unsustainable Software Development

- Fear to touch existing code
- System grows in size and complexity
- New developers are needed
- Workarounds (hacks, copy-paste, ...)

@EvelynvanKelle @ykanellopoulos
Why maintainable software?

Initial Development

Maintenance

Initial effort

Recurring effort (7 years - ∞)

Maintenance starts after lunch on the first day of development
How to measure software quality?
### Giving feedback on software products: personal versus tool-based

<table>
<thead>
<tr>
<th>Personal feedback</th>
<th>Tool-based feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific for your project</td>
<td>Faster feedback loop</td>
</tr>
<tr>
<td>More sensitive to context</td>
<td>Allows for scalability by iteration</td>
</tr>
<tr>
<td>Concrete suggestions to improve</td>
<td>More objective</td>
</tr>
</tbody>
</table>

**So which one is better?**

There is a false dichotomy between full automation and human intervention. Successful quality control combines tool-based measurement with manual review and discussion.
Pitfalls in using measurements

One-track metric

Metrics Galore

Treating the metric

Metric in a bubble
Static code analysis challenges

What are the biggest pitfalls of code quality tools

- Too many false positives: 40% of respondents
- Too many warnings: 35% of respondents
- High price: 30% of respondents
- Difficult to configure: 25% of respondents
- No actionable recommendations: 20% of respondents
- Lack of broadly accepted quality standard: 15% of respondents

Percentage of respondents (T=899)
10 guidelines for future-proof code

**Code**
- Write small units of code
- Write simple units of code
- Write code once
- Keep unit interfaces small

**Architecture**
- Separate concerns in modules
- Couple architecture components loosely
- Keep architecture components balanced
- Keep your codebase small

**Way of working**
- Automate tests
- Write clean code

@EvelynvanKelle
@ykanellopoulos
Use measurable standards: make guidelines quantified and actionable

**Code**
- Limit units to 15 lines of code
- Limit branch points per unit to 4
- Do not copy code longer than 6 lines
- Limit parameters per unit to 4

**Architecture**
- Avoid modules larger than 400 lines of code
- Hide classes from other components, no cycles
- Aim for 6-12 top-level components
- Keep codebase below 200,000 lines of code

**Way of working**
- Write automated tests that cover all code
- Stick to the seven “boy scout rules”
How to build effective software teams?
10 best practices for effective software development

- Derive Metrics from Your Measurement Goals
- Make Definition of Done Explicit
- Control Code Versions and Development Branches
- Control Development, Test, Acceptance, and Production Environments
- Automate Tests
- Use Continuous Integration
- Automate Deployment
- Standardize the Development Environment
- Manage Usage of Third-Party Code
- Document Just Enough

@EvelynvanKelle @ykanellopoulos