Sustainable Software Development for Agile Teams

Alexander v. Zitzewitz

a.zitzewitz@hello2morrow.com

hello2morrow, Inc.
Software Development Teams Struggle to Keep Up

Study (Oct 2013) by Forrester Research

Only 39% of teams are on-time, on-budget

Reasons

- 56% cite ever changing requirements
- 50% cite trying to do too much at once
- 34% lack of clear executive direction
- ...

Are we doing it the right way?

How much of that is related to “Technical Debt”?

Shouldn’t we be better at that after 6 decades of software development?
Technical Debt as a Metaphor

Ward Cunningham first defined the term in 1992:

“Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite… The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise.”
Agile Development and Sustainability

- The agile approach does not automatically create maintainable software. Often the opposite is true.
- Ongoing management of Technical Debt is considered to be a critical success factor for high quality and maintainable software systems by promoters of the agile approach.
- That challenges the idea that software development should almost exclusively be driven by business value.
- Project size has obviously an important influence.
How about Technical Quality?

- Do you have binding rules for code quality?
- Do you measure quality rule violations on a daily base?
- Is your architecture defined in a formal way?
- Do you measure architecture violations on a daily base?
- Does quality management happen at the end of development?
- Does your current QM lead to sustainable results?
Erosion of Architecture – Symptoms (Robert C. Martin)

- Rigidity – The system is hard to change because every change forces many other changes.
- Fragility – Changes cause the system to break in conceptually unrelated places.
- Immobility – It's hard to disentangle the system into reusable components.
- Viscosity – Doing things right is harder than doing things wrong.
- Opacity – It is hard to read and understand. It does not express its intent well.

Overall: “The software starts to rot like a bad piece of meat”
Erosion of Architecture – Reasons

- Most projects don’t measure quality on a regular base
- Management considers software as a black box
- System knowledge and skills are not evenly distributed
- Complexity grows faster than system size
- Unwanted dependencies are created without being noticed
- Coupling and complexity are growing quickly. When you realize it, it is often too late
- Quality measurement is done at the end of development
- Time pressure is always a good excuse to sacrifice structure
- The Law of Entropy
Cost of Structural Erosion / Technical Debt
Technical Quality and Sustainability
How to Define Technical Quality?

“Technical quality of software can be defined as the level of conformance of a software system to a set of rules and guidelines derived from common sense and best practices. Those rules should cover software architecture, programming in general, testing and coding style.”

- Technical quality cannot be achieved by testing only
- Technical quality manifests itself in every line of code
- Four aspects of technical quality:
  - Architecture / Dependency-Structure
  - Software metrics
  - Programming rules
  - Testability and test coverage
- Measuring of Technical Debt is done by counting rule violations and weighing them properly
- That requires automated static code analysis
### Categories of Technical Debt

<table>
<thead>
<tr>
<th>Category</th>
<th>Repair Cost</th>
<th>Visible Impact</th>
<th>Maintainability Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>Low</td>
<td>Medium</td>
<td>Mostly low</td>
</tr>
<tr>
<td>Testing</td>
<td>Potentially High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Metrics</td>
<td>Low</td>
<td>Low</td>
<td>Low / High</td>
</tr>
<tr>
<td>Architecture</td>
<td>Very High</td>
<td>Low</td>
<td>Very High</td>
</tr>
</tbody>
</table>

02.02.15
© 2005-2008, hello2morrow
Typical Example for Structural Erosion
How to Implement Sustainability
Preconditions for Sustainability

- Nothing can be delivered that does not meet the standards defined for technical quality. (Exceptions apply)
- Rules and guidelines are documented and checked in an automated way.
- Quality criteria are a core component of development guidelines.
- Sustainability as a goal must be supported by all management levels.
- Invest about 20% of all development and maintenance effort into code hygiene and architecture.
You need an Architectural Model

- Step 1: Cut horizontally into Layers
- Step 2: Cut vertically into vertical slices by functional aspects
- Step 3: Defines the rules of engagement

[Diagram of User Interface, Business Logic, and Data Access layers with natural subsystems]
Improvements Require Transparency
Six Sigma for Software

Define → Control → Measure → Improve → Analyze → Define

© 2005-2014, hello2morrow
Some Simple Rules for Sustainable Projects

- Rule 1: Define a logical architecture down to the level of subsystems and a strict and consistent package/namespace naming convention
- Rule 2: No cyclic dependencies between different packages/namespaces
- Rule 3: Keep coupling low (use coupling metrics like ACD)
- Rule 4: Limit the size of source files (900 LOC is a reasonable value)
- Rule 5: Limit the McCabe complexity of methods (e.g. 15)
- Rule 6: Limit the size of a packages/namespaces (e.g. less than 50 types)
Quality Control Dashboard Example

**Short Term Trend**

- Structural Debt Index (SDI)

**Long Term Trend (24h Moving Average)**

- Structural Debt Index (SDI)

**Violating Type Dependencies**

**Highest ACD**
DZone’s “Designing Quality Software” Refcard

Refcard #130: http://refcardz.dzone.com/
Relevant White-Papers:

1. Project Sanity Checklist
2. The Value of Architecture

Download from www.hello2morrow.com
Also check blog.hello2morrow.com