From Assessment to Reduction: How Cutter Consortium Helps Rein in Millions of Dollars in Technical Debt

Israel Gat
Cutter Consortium
37 Broadway
Arlington, MA 02474 USA
+1 781 648 8700
igat@cutter.com

John D. Heintz
Gist Labs
4230 Canyon Glen Circle
Austin, TX 78732 USA
+1 512 633 1198
john@gistlabs.com

ABSTRACT
Technical debt assessments often follow a similar pattern across engagements. In contrast, technical debt reduction projects can vary significantly from one company to another. In addition to exposing unexpected technical challenges, technical debt reduction projects bring business issues, organizational consideration, and methodical questions to the fore. This paper outlines how all these aspects are being dealt with in a Cutter Consortium engagement with a client that struggles to wrestle technical debt to the ground.

Categories and Subject Descriptors
D.2.9 [**Software Management**]: Productivity, Life Cycle

General Terms
Management, Measurement, Performance, Economics, Human Factors

Keywords
Technical debt, cost-benefit analysis, design decision trade-off, large-scale system development, software metrics, agile metrics, agile, devops, operations, development, code deficit

1. SITUATION
Following a Cutter webinar, a client approached us seeking guidance on dealing with two issues: technical debt and the challenges of slow delivery. This client had already realized that technical debt is a primary cause of its slow development and QA cycles and that an agile development method is required.

We would soon also learn that the absence of automated testing and a deeply branched configuration management strategy is significantly impacting its ability to release software. This successful, international, many-hundred-person company has gone past the point of relying on staff members to be individually smart or hard working and recognized that it struggles with the basics of successful delivery. This company has a legacy codebase consisting of Java, Java Server Pages, Javascript, and Flex.

MTD'11, May 23, 2011, Waikiki, Honolulu, HI, USA

2. WHAT WE FOUND
Our first engagement was to provide an initial assessment of the client's technical debt. During this short engagement, in addition to delivering the initial numbers described below, we drew the following preliminary conclusions:

- Some of its code modules were overwhelmed with technical debt, averaging more than $10 per line of code.
- This client had already begun the process of replacing one such module, the primary front-end user interface, with a new streamlined framework.

Cutter's technical debt assessment is an automated analysis of code deficits, both dynamic (unit testing and code coverage) and static (rule conformance, code complexity, duplication of code, documentation, and design characteristics) [1]. All discovered deficits are converted to a time effort and summed to produce a total effort to redress, measured in both person-days and dollars.

The results of the initial technical debt assessment show several distinct categories of code:

Monolithic back-end: The majority of back-end code (over 1 million lines as shown in Figure 1) was managed as a single-source repository with moderately high technical debt on average and some particularly high regions. The initial static technical debt analysis shows $3.9M in technical debt (see Figure 2).

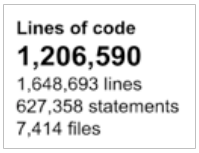

Figure 1: Server Lines of Code

Figure 2: Server Technical Debt, unadjusted

We adjusted this figure in two ways:

1. We removed the measured technical debt from one of the code modules, subtracting $0.7M from the total. This module was the result of automated code translation from a prior library and, while still the responsibility of the client to maintain, was not authored code managed by the teams.
2. We added $2.4M after hand-calculating the code coverage implication of uncovered complexity.

This resulted in an average of $4.75 technical debt per line of code for the primary back-end code. Code complexity, unit test coverage, and duplicate code were judged the most important concerns to address first.

Modularized back-end: Some back-end code had already been encapsulated into standalone modules. These modules were significantly smaller and had on average $3 technical debt per line of code. This represented a step in the right direction and unit test coverage was the primary impact on technical debt cost.

Legacy front-end: The majority of the front-end code (more than 750,000 lines as shown in Figure 3) was implemented via a legacy technology and style.

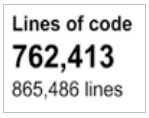

Figure 3: Legacy front-end,

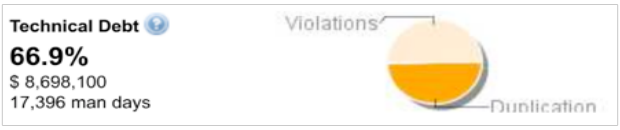

Figure 4: Server Technical Debt, unadjusted

The initial static technical debt analysis showed a significant $8.7M of technical debt, caused by an even contribution from both code duplication and rule violations (see Figure 4).

These results showed an average of $11.40 technical debt per line of code.

New front-end framework: The client had already begun creating a new front-end implementation and was beginning incremental migration. Initial analysis of this framework showed overall much lower estimated technical debt. One interesting detail emerged however: the amount of code duplication was significantly high (~40%) and tracking upwards to the same levels as the previous generation of UI code.

In order to perform a technical debt assessment of the new front-end code, written in Javascript, we developed a collection of analysis tools and integrated them together to perform the same calculations as the other implementation language analysis. This collection of Javascript tools is now being used in multiple clients sites across the US and India.

3. PROPOSED DIRECTION

From our measurements, observations, and comparison with data in the Cutter repository, we drew several conclusions. First, the amount of technical debt in all components of the system must be reduced. Second, moving to an Agile or Lean development method with test automation-based practices was a positive step for this client to take to begin changing the development style of creating technical debt. We proposed an initial plan consisting of the following major points:

- **Technical Debt Reduction.** A technical debt "SWAT" team, led by one of the company's most senior architects, tasked with learning how to reduce the technical debt and then rolling that knowledge out to the rest of the development staff, should be established. This team should focus first on unit testing and any necessary build management changes to enable widespread adoption and execution of those tests.
- **Offload Manual Testing.** The "long tail" of manual testing is one of the primary causes of delayed delivery. We recommend augmenting manual testing with "Test as a Service" supporting companies.
- **Technical Debt SWAT Team as Agile Team.** We recommend running the technical debt SWAT team as an Agile team. The backlog for this team is not user stories but rather units of identified technical debt.
- **Initial Agile Training.** We recommend the technical debt SWAT team, key leaders, and likely future team members attend a two-day initial Agile training. The goal of this training is to enable participants to begin "sprinting" immediately after training.
- **Governance.** We propose tying together cost, technical debt, and value to accomplish effective project governance. This would apply across the entire development pipeline: from new feature development to internal and external brands to maintenance work and infrastructure improvements.

- **Devops.** After an initial examination, we recommend delaying significant focus on progressing toward a devops integration and continuous deployment. Once cycle time across all the teams from beginning development to QA verification is short enough (not yet defined), this thread will be revisited.

4. PRELIMINARY PROGRESS

Since our initial recommendation, progress and adjustments have been made. The Agile rollout, training, and pilot teams have begun to expand, the technical debt SWAT team has redefined its direction, and other functional groups within the organization have become more involved and are starting to coordinate with these efforts.

The initial scope for the Agile training has been expanded to a full rollout of Agile across the entire product management, development, and QA groups. This commitment has met some skepticism, but the increasing swell of activity, progress, and momentum is beginning to build trust that the organization is actually making changes. This systemic approach to Agile adoption, tempered by a strategy for dealing with the interim mixed mode of operation, will provide the client with technical and business capabilities to both redress its technical debt and make progress on new and expanded business ventures.

A second pilot Agile team, with a third starting to form, has begun sprinting. This team is a traditional Agile team, focusing directly on creating small increments of customer value features. Team members assume the responsibility for testing (with automations) their own code and features. Moreover, the team will spend 20%-30% of its time carrying out technical debt reduction tasks.

The technical debt SWAT team is currently focusing on the following steps:

- Beginning to do some actual technical debt reduction work, focusing on unit testing and reducing complexity.
- Providing build/infrastructure tools to make its and everyone else's development easier. Examples include modifying the build scripts to include easy unit test execution and planning for a simpler branch/merge configuration management model.
- Evangelizing its work and the path to technical debt reduction to the rest of the company. Members have hosted "introduction to unit testing" seminars and have begun coordinating with other teams on how to implement the simpler branch/merge strategy and migration.

In short, this team is becoming the center of excellence in all things related to technical debt. It will be the primary advisor providing expertise and guidance for other teams to leverage. As a matter of policy, any Agile team that will be launched in the future will have 20-30% of its backlog populated with technical debt reduction stories.

5. CONCLUSION

To become actionable, a technical debt assessment plan must be followed by a technical debt reduction project. The project implemented in the Cutter client engagement reported herein builds on the following elements:

- SWAT team
- Evangelism
- Agile methods
- Technical debt items in the backlog of every team that converts to Agile

These elements serve as the blueprint for other technical debt reduction engagements carried out by the Cutter Consortium [1].

6. REFERENCES

[1] Gat, I. (ed.) 2010. *How to settle your technical debt--a manager's guide.* Cutter Consortium, Arlington Mass.