silverthread

Customer Use Case:

Informed Decision Making Through Custom ROI Tool

Abstract

A multinational software company was struggling to identify quality issues in a business-critical legacy system. The development team suspected that the issue was architectural in nature but was unable to find design flaws within modules. When Silverthread scanned the codebase they discovered that while individual modules were architecturally sound, modules were put together in ways that increased complexity and produced difficult to diagnose bugs.

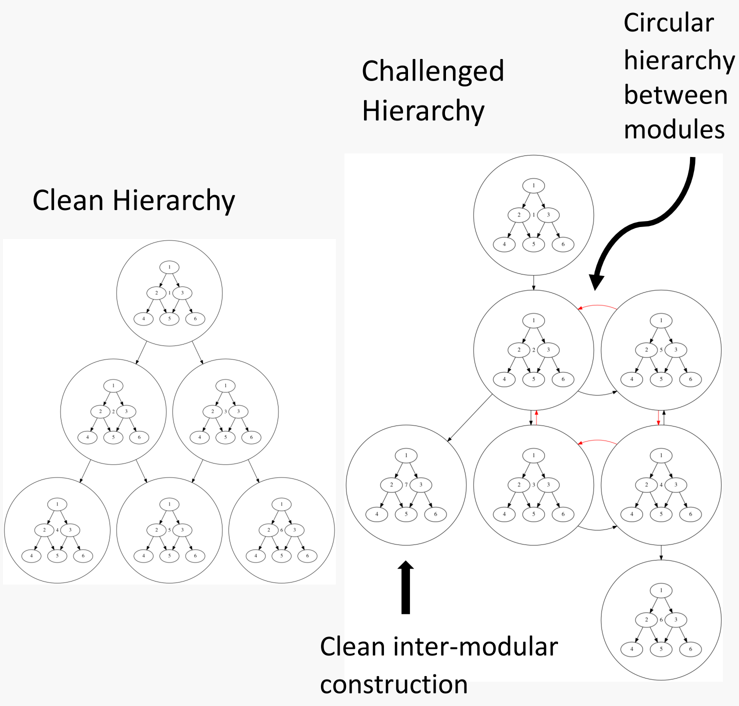
Result

Silverthread exposed the architecture flaws and created a ROI estimate tool to help the business determine which level of refactoring and test coverage was the most profitable for their business.

The Problem

A large, multinational software company was experiencing unexpected delays and costly repairs to a business critical legacy codebase. Software architects at the company attempted to discern the origin of these issues, but were unable to find a cause. Silverthread was contacted to locate and eliminate the source of the delay, and provide economic projections for various refactoring efforts.

Like most software companies working in large codebases, this company had several development teams that each focused on one section of the codebase, called components. Components are designed to be independent of each other with component relationships going through declared and maintained APIs, and with dependencies flowing in one direction (hierarchical). Silverthread scanned the codebase and found that components were circumventing the hierarchy, calling on each other in unannounced back door routines and creating a cyclical relationship. When an engineer in one component team made a change to a cyclical component, the change would ripple through the system and affect other components. Due to the improper component relationship, these changes were unexpected, unaccounted for, and undocumented. An additional barrier to effective code development was the obscurity of the existing codebase to its developers: each developer was familiar only with the files that they had worked on or created.


Circular hierarchy between modules
Challenged Hierarchy
Clean Hierarchy
Clean inter-modular construction

Silverthread's Findings: High Complexity, Low Visibility

Silverthread created both file- and component-level visualizations to act as a visual aid for codebase architecture. These visualizations additionally shed light on change impacts and potential propagation of bugs from updating one area of the codebase. This unplanned chain of edits created hours of engineering rework, a higher likelihood of bugs and a heightened risk of downstream issues. In order to remove this cyclicality the codebase needed to be refactored. Codebase refactoring is expensive, time consuming, and typically delays production for several months. The client was unsure if refactoring the codebase was economically sound or if they should continue business as usual.

Using code quality, design quality, and test coverage information from the CodeMRI® scans, Silverthread created fitted software economic models. These models predicted the amount of future defects in each area of the code. The models were refined through machine learning and trained on Silverthread's benchmark database of over 6,000 systems.

Using these models and visualizations Silverthread identified key coupling points where files were incorrectly attached and created a targeted file list to ensure that refactoring efforts focused only on challenged parts of the code.

silverthread

Customer Use Case:

Informed Decision Making Through Custom ROI Tool

Step 1: **Building Insight**
Determine the cause of the development delay

❯

Step 2: **Learning**
Connect technical and economic development

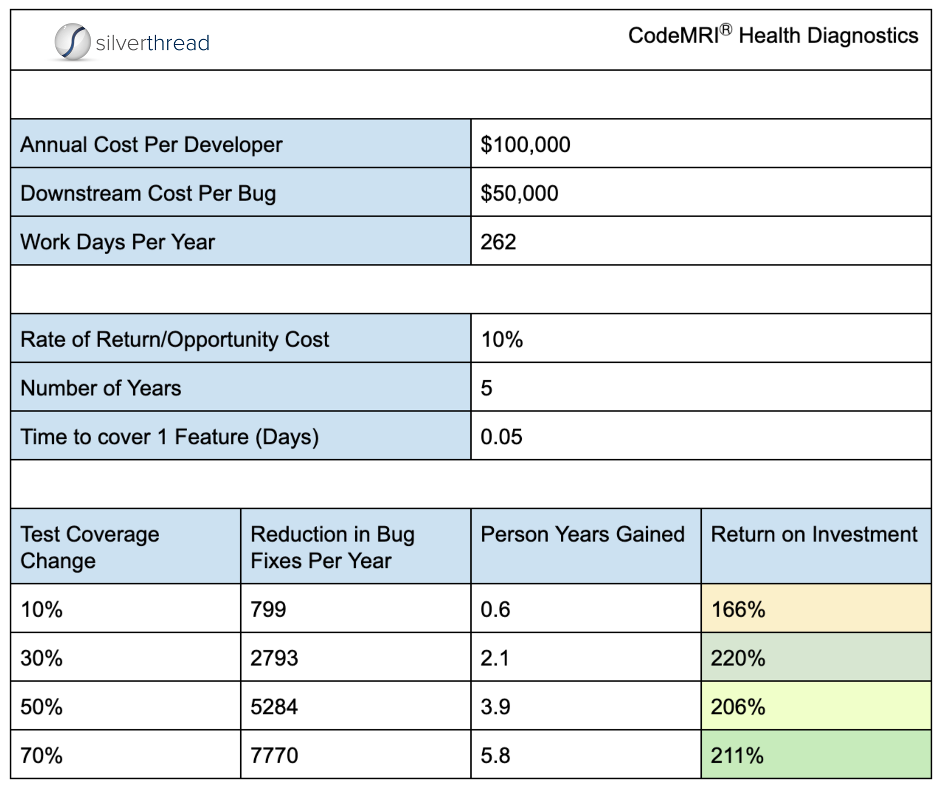
❯

Step 3: **Decision Making**
Create fitted ROI models to determine best option

Custom Predictions and Solutions

After adding task tracking and version control data to create fitted models tailored to this organization’s software, Silverthread created a tool that allowed them to evaluate the economic outcomes of improvements made to the codebase. Regression analysis combined with benchmarking led to the discovery of a high correlation of financial waste with system complexity. Silverthread used these data to create and refine economic models for a variety of software development activities. This provided an estimate of expected time, effort, and expense incurred from refactoring as well as subsequent cost savings.

Silverthread presented the client with an ROI estimation tool that predicted resulting improvements in risk, cost, and optionality based on various initial investments. The client was able to choose the plan that least disrupted their workflow and had the highest ROI.



silverthread — CodeMRI® Health Diagnostics

Annual Cost Per Developer	$100,000
Downstream Cost Per Bug	$50,000
Work Days Per Year	262

Rate of Return/Opportunity Cost	10%
Number of Years	5
Time to cover 1 Feature (Days)	0.05

Test Coverage Change	Reduction in Bug Fixes Per Year	Person Years Gained	Return on Investment
10%	799	0.6	166%
30%	2793	2.1	220%
50%	5284	3.9	206%
70%	7770	5.8	211%

About Silverthread

Silverthread is the market leader in software economics – helping executives take financial control over complex software assets. Based on 15 years of applied research at MIT and Harvard Business School, the CodeMRI® platform of tools allows organizations to translate software architectural health metrics into quantifiable business impacts. We have helped over 100 global commercial and government institutions and programs gain visibility into their software asset health, and dramatically improve operational and financial outcomes.