silverthread

Customer Use Case:

Refactoring for Software Economics

Abstract

A Department of Defense (DoD) customer with a legacy Java system was experiencing development delays and budget overruns due to poor design quality and architecture. Silverthread's CodeMRI® Platform identified areas in the system where architecture had degraded and outlined a technical health improvement plan that involved refactoring the codebase over four months using 25% of developer labor, resulting in key software economic outcomes.

Result: Refactoring the legacy system doubled developer productivity and reduced the cost to create a feature by 45%.

Problem

A software development team for a DoD system was consistently missing project deadlines and running over budget due to the degraded health of the legacy codebase they were working in. The software team was spending significantly more time fixing bugs than producing new features. Executives were becoming increasingly frustrated with the software team as deadlines were not being met and budgets being overrun, however, they were not willing to make any further investment until the development team could provide them with data proving that they needed the capital. The development team was having a hard time working with such inefficient code but did not have the design insight to refactor the code themselves or to estimate the financial returns on refactoring vs rewriting the codebase.

Diagnostic Assessment

Silverthread's CodeMRI® Diagnostic reports determined that the slowdown was due to poor design quality and found a large 'Critical Core' in the codebase. A 'Core' is a group of files that break traditional code structure hierarchy, leading to cyclical demands and eventual gridlock.

Silverthread additionally used its ROI Estimator to ascertain whether it was financially prudent to refactor the existing codebase or if it would make more sense to replace the system with a new codebase at a higher architectural standard. This codebase was determined to be an excellent candidate for refactoring, with a high anticipated ROI. Developers and the executive team were finally able to communicate and determined together to being refactoring the codebase.

Solution: Refactor existing system

Using Silverthread's technical health improvement plans, developers were able to identify and attack areas of especially poor quality in order to quickly increase the design quality of the system. The team iteratively selected some recommended changes to act on, fixed them, and generated new improvement plans to check results and explore what to do next.

The tool proposed an order to attempt the refactoring process, but the engineers were free to pick and choose which steps they wanted to work on first. Proposed steps that seemed especially difficult were skipped, and those deemed low-risk were done first. Developers likened the experience to "untangling fishing line" with the benefit of an expert system to guide them. Some knots that were originally very difficult became much more feasible to attack once others around them are removed. CodeMRI® Care also caught and prevented new problems from being introduced that normally would have been missed and left to fester. This included a large amount of faulty code that would have created another Critical Core in the system. CodeMRI® Care caught this immediately and was able to flag and reverse the damage. Over 4 months, the team successfully eliminated the 800+ file Critical Core and all other emerging architectural issues.

Economic Outcome - "Twice as productive"

The resulting refactoring effort produced a number of key results:

- **Revenue/capabilities**: Developers were now "twice as productive"
- **Cost savings**: 52% reduction in wasted software investment

Improving the codebase rapidly led to improved development speed and productivity, and lower cost and risk. The development team reports that the investment in technical health improvement "**paid for itself**" in productivity and quality gains in the span of a few months, with a strongly positive ROI. The development team say that it is now easier to incorporate new developers into the team. Learning curves have been shortened. The team was able to "throw out" a manual for rookies with tips, tricks, and warnings about side-effects that could be triggered in the previously fragile system, as it was no longer needed. Features are developed and shipped twice as fast at half the cost, with more predictability and fewer defects.

 | 200 Portland Street, Suite 500, Boston, MA 02114
www.silverthreadinc.com | info@silverthreadinc.com | Ph: (617) 603-0075

silverthread

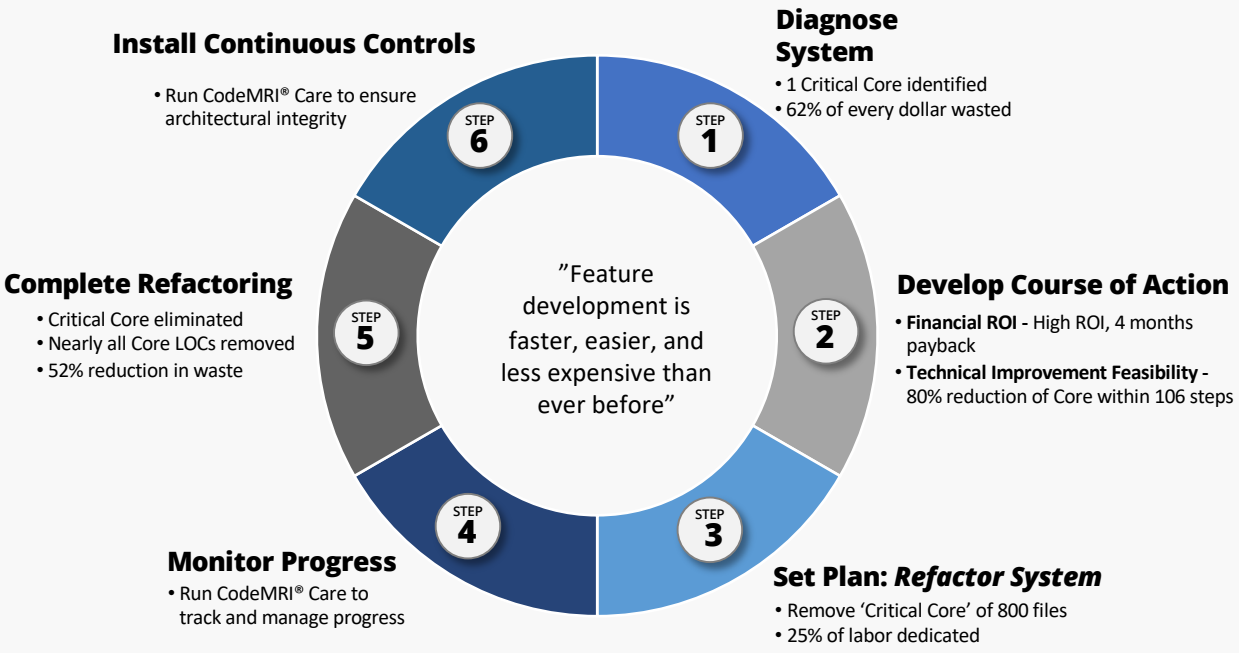
Customer Use Case:

Refactoring for Software Economics

Why Refactor?

Systems are often developed without continuous architectural and design support. As code is continuously added to a database, a lack of architectural guidelines allow code to grow against the design hierarchy, which creates complex and unwieldy codebases that are difficult to work in and prone to bugs. As codebase complexity grows, speed of delivery, agility, quality, and schedule are harmed. Most organizations do not track hidden costs in their codebases such as technical debt and code depreciation. They are often surprised when ill effects of complexity begin to outweigh the benefits of continued use of a system. Rather than scrap a codebase (which typically represents thousands of hours of work), refactoring allows companies to fix these issues of technical debt and code depreciation while still allowing the codebase to continue to run. This provides the least amount of interruption to typical processes and can be much faster and less expensive than starting from scratch.

Customer Journey



Customer Outcomes

TECHNICAL HEALTH

Design Quality

96%
of Core files
eliminated

Code Quality

16%
fewer bugs
released

ECONOMIC OUTCOMES

Capabilities

76%
increase in lines of code per year

Optionality

33%
of developer FTE
optionality

Cost Savings

52%
less waste
per dollar invested

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.