Is your software healthy? Process measures tell only part of the story.

Quantifying design quality helps to steer projects and improve software economics.

Software projects are complex and complicated.
Software systems are too complicated for individual human beings to understand. No single person can understand them in their entirety. Evolving complex and durable software assets is challenging. Many software code bases are decades old, contain millions of lines of code, and are increasingly complex. They are so mature that *geriatric* is a better description. Unnecessary complexity results in highly unpredictable outcomes.

Design quality has more impact on agility than process does.
Design quality in a software system is the single most important factor impacting business agility. When design quality is good, developers can understand the code, teams can communicate architectural intent effectively, and software delivery is efficient. When design quality of a code base degrades, waste, rework, and unnecessary overhead destroy efficiency and unintended consequences degrade quality. Eliminating unnecessary complexity yields substantial rewards in efficiency, effectiveness, and staff morale.

Systems and software leaders need more honest measures.
Every software project aims to build and sustain resilient designs, maintainable code, and thorough, automated tests. Traditional project steering is overly focused on supporting process artifacts such as plans, requirements, progress reports, and documentation. Measuring these secondary artifacts yields only subjective guesses of progress and quality, and creates significant overhead. Factual measures of design quality come from the primary artifacts, namely the evolving code base, and these sources enable a more honest assessment. With these more objective insights, project leadership can:

- Locate the root-cause complexity tumors
- Benchmark and understand quality trends
- Forecast economic outcomes more predictably

Leaders can steer the software delivery *process* more honestly by measuring and understanding the evolving *product* artifacts. For example, such measures are a necessity when deciding whether a legacy code base should be ported, refactored, replaced, or migrated to the cloud.

What is design quality and how do we measure it?
A software system is well designed when its code base adheres to certain principles that enable agility, maintainability, and understandability. These include modularity, layering, hierarchy, inheritance, and reuse. When designed well, they have properties that allow individual parts to be changed separately without overwhelming and unintended consequences.

Quantifying design quality is one of the software industry's holy grails. Our research and field applications demonstrate valuable insights that can be realized through code scans and visualized through design structure matrices (DSMs). A DSM is a visual representation of the network of entities and relationships that make up a software system. Silverthread has pioneered the use of DSMs to visualize and quantify design principles. Ideal DSMs, like the one illustrated in Figure 1, have a horizontal layer of hierarchical control dependencies from a well-structured control component, with very few other component-to-component dependencies. They also tend to have a vertical column of dependencies for shared utilities and reusable services, APIs, classes, and data elements.

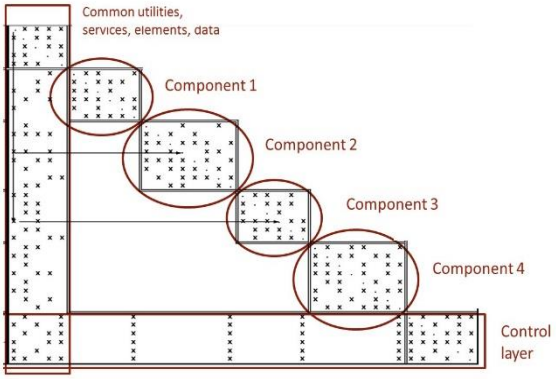

Figure 1: DSM of an ideal code base

After 15 years of Harvard/MIT research capturing a diverse spectrum of various systems, we found recurring patterns of good and bad quality. Figure 2 illustrates two typical DSMs at opposite ends of the quality spectrum.

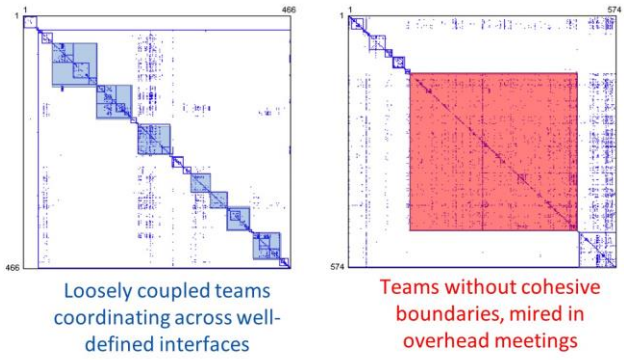

Figure 2: DSMs help us visualize design quality principles

The modular DSM on the left exhibits locally tight coupling within components but relatively loose coupling among components. The alarming structure on the right exhibits tight coupling across a large, dominant core component (red), with smaller peripheral components. The "core" is the component that is most complex and least hierarchical. Our research has substantiated that a wide variance in economic outcomes is significantly correlated to wide variations in design quality.

Figure 3 is a visualization of System X from a Silverthread CodeMRI® report. The system was developed by a system integrator, then handed off to a government organization for maintenance and operation. The red portion of the diagram shows a core of 11,000 files that are circularly interdependent on each other, directly or indirectly. Encapsulation, dependencies, and APIs are absent or have degraded over time.

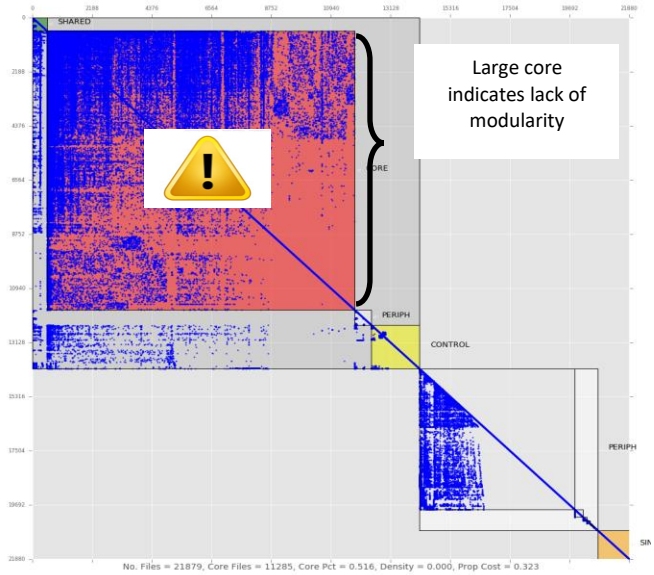

Figure 3: A code base with design quality problems

Predicting waste and overhead in software efforts

The Silverthread team has established a large body of empirical software data and published research that statistically links design quality to software economic outcomes. This body of data includes defect density, developer productivity, bugs deployed into production, and other key performance drivers used to make software economics predictions. Figures 4 and 5 show predictions of team productivity based on thousands of similar systems.

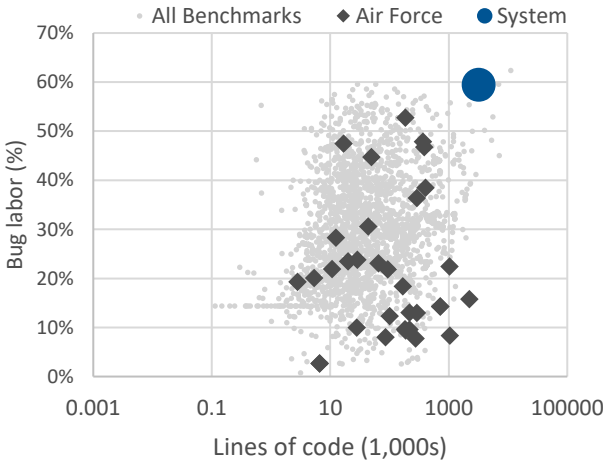

Figure 4: Percentage of effort expected in fixing defects

Econometric models were applied to predict the balance of labor expected between fixing defects (playing defense) versus adding new features (playing offense) in a code base. Figure 4 shows code base size (X=axis) versus projected bug labor % (Y-axis) for thousands of systems, including System X (blue dot). The chart predicts that >60% of effort in System X might be dedicated to bug-fix activity due to complexity and design quality challenges.

Predicting efficiency of software delivery

Figure 5 illustrates the predicted number of labor-days required to develop and debug a new 1,000-line feature. A typical developer in System X is predicted to require more than 80 days: 30 to complete development and 50 to test. In contrast, developers working in systems ranked in the top 10% of our benchmarks can deliver a 1,000-line feature in 15 days.

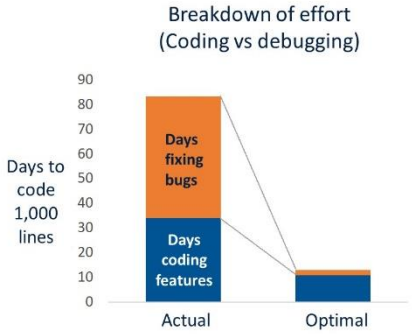

Figure 5: System X will have high defect rates.

Enabling more honest conversations and building trust

System X has experienced significant maintenance difficulties. Developers feel choked by waste, rework, complexity, unanticipated side effects, and low morale. Program leadership suffers from an inability to meet user expectations within a reasonable timeframe and the loss of credibility in forecasting cost, quality, and release targets.

When complexity grows and measurements are largely guesswork, trust between development teams and program leadership dissolves. Quantifying design quality directly from the evolving code base delivers a critical quid pro quo: less overhead for practitioners and more insightful dynamic control for management. When practitioners and managers use the same measures, trust grows. Increasing trust enables leaner production by reducing sources of overhead, unnecessary rework, and waste. Trust is the currency of lean engineering efficiency.

Contact Us

Silverthreads's mission is to advance the state of software measurement practice by quantifying complexity and design quality. Our measurement know-how can establish a more trustworthy foundation for improving software economics.
http://silverthreadinc.com