Software Requirements Metrics Provide Leading Indicators in Measurement-Driven Dashboards for Large-Scale Systems

Richard W. Selby

Head of Software Products, Northrop Grumman Space Technology, One Space Park, R4/2011, Redondo Beach, California, 90278, USA, Rick.Selby@NGC.com, 310-813-5570
Adjunct Professor, Computer Science Department, University of Southern California

Abstract

Measurement-driven dashboards provide a unifying mechanism for understanding, evaluating, and predicting the development, management, and economics of large-scale systems and processes. Dashboards enable interactive graphical displays of complex information and support flexible capabilities for user customizability and extensibility. Dashboards commonly include metrics for software requirements because requirements size, growth, and stability are critical drivers for software projects. This paper focuses on dashboards that have been used on actual large-scale projects as well as example empirical relationships revealed by the dashboards. The ratio of software requirements to source-lines-of-code averaged 1:46 for these projects. Projects that far exceeded the 1:46 requirements-to-code ratio tended to be more effort-intensive and defect-prone during verification. Measurement-driven dashboards increase visibility into large-scale systems and provide feedback to organizations and projects.

1. Introduction

Current and future technological systems are increasingly software-intensive and large-scale in terms of system size, functionality breadth, component maturity, and supplier heterogeneity [10]. Organizations that tackle these large-scale systems and attempt to achieve ambitious goals often deliver incomplete capabilities, produce inflexible designs, reveal poor progress visibility, and consume unfavorable schedule durations. Successful management of these systems requires the ability to learn from past performance, understand current challenges and opportunities, and develop plans for the future. Effective management planning, decision-making, and learning processes rely on a spectrum of data, information, and knowledge to be successful. However, many organizations and projects possess insufficient or poorly organized data collection and analysis mechanisms that result in limited, inaccurate, or untimely feedback to managers and developers. Organization and project performance suffers because managers and developers do not have the data they need or do not exploit the data available to yield useful information.

2. Dashboards

Measurement-driven dashboards provide the foundation for effective and efficient management of organizations and projects that develop large-scale systems [2]. Dashboards provide graphical displays of interactive measurement-driven gauges that depict trends, identify outliers, and support drill-down capabilities to more detailed information [2] [4] [5] [6] [7]. These information-intensive dashboards create value by collecting, analyzing, and synthesizing data so that decision makers can characterize progress, compare alternatives, evaluate risks, and predict outcomes [8] [9]. Decision makers use dashboards to enable data-driven management as well as systematic process improvement and organizational change [3] [11]. Many improvement methods, such as Six Sigma and Capability Maturity
Model Integration (CMMI), utilize data-centric techniques and rely on accurate measurement-based information [1].

Different dashboards address different sets of goals. For example, one dashboard may support the following development-oriented goal areas: requirements, reuse, technology infusion, progress, cycletime, deliveries, pre-delivery defects, and post-delivery defects. Another dashboard may support the following management-oriented goal areas: business scope, customer satisfaction, risk, business development, finance, personnel, process compliance, and process improvement. Figure 1 displays an example dashboard for software development-oriented goals.

Dashboards incorporate a variety of information and features to help managers and developers characterize progress, identify outliers, compare alternatives, evaluate risks, and predict outcomes. The design principles for dashboards include the following:

- Enable different metric sets to support different goals
- Utilize different types of displays or gauges for different types of data
- Enable organization- and project-specific displays
- Define points-of-contact that identify responsible persons
- Define data trends and the date through which the data are current
- Define lower and upper control limits and hyperlinks for displaying outliers
- Enable hyperlinks that display tabular formats of the underlying data
- Support context-specific help
- Enable hyperlinks to view or “drill down” hierarchically to more detailed data
- Highlight overall metric status using red, yellow, or green indicators

Figure 2 redisplays the example dashboard from Figure 1 and identifies key features that support design principles for improving usefulness.
3. Software Requirements Metrics

Dashboards commonly include software requirements metrics because they are leading indicators of project scope, growth, stability, and progress. Software requirements metrics characterize the “problem space” that a project is addressing, as opposed to metrics such as source-lines-of-code that characterize the “solution space.” Software requirements metrics are also available very early in a project and can form the basis for early analyses and predictions of project plans, alternatives, risks, and outcomes. Using software requirements metrics also helps resolve the counting issues associated with reused design or code and whether components are developed in-house or from commercial-off-the-shelf (COTS) suppliers. The project requirements, in terms of functionality and performance, are typically the same regardless of whether the implementation reuses software design and code or incorporates COTS components. Of course, the project requirements, in terms of organization and process, may vary depending on the degree of software reuse and usage of COTS components.

This paper focuses on software requirements metrics from dashboards that have been used on actual large-scale projects as well as example empirical relationships revealed by the metrics. The data analyzed originates from over 10 large-scale projects that use measurement-driven dashboards to actively manage...
their development activities and evolving products. Figure 3 displays the software requirements metrics for the projects.

![Ratio of SLOC to Requirements](image)

Figure 3. Software requirements metrics for the projects.

For these projects, the number of requirements is defined to be the number of “shall” statements in the requirements specification documents. For example, a requirements document may contain the following statement “the system shall determine the three-dimensional location of a vehicle within an accuracy of 0.1 meter.” This “shall” statement would count as one requirement. In order to facilitate consistency within and across projects, requirement specification standards and guidelines need to be defined to enforce the breadth and depth of functionality expressed in a single requirement. Of course, simply counting the “shall” statements oversimplifies the project requirements, but this metric does provide an initial basis for project scope, growth, stability, and progress.

4. Data Analysis

Initial data analysis of the software requirements metrics reveals the following observations:

♦ The ratio of requirements in a system-level parent specification to requirements in a software specification ranges from 1:300 for early projects to 1:6 for mature projects.

♦ The ratio of requirements in a software specification to delivered source-lines-of-code averages 1:81 for mature projects and has a median of 1:35.
♦ When Project #14 is excluded (see Figure 3), the ratio of requirements in a software specification to delivered source-lines-of-code averages 1:46 for mature projects and has a median of 1:33.
♦ Projects that far exceeded the 1:46 requirements-to-code ratio, such as Projects #13 and #14, tended to be more effort-intensive and defect-prone during verification.

5. Conclusions

Our ongoing research investigates principles for measurement-driven dashboards for development and management of large-scale systems. Successful development, management, and improvement of large-scale systems require the creation of dashboards as well as underlying infrastructure to support data collection from organizations, projects, processes, products, teams, and resources. Synergistic integration of dashboards and underlying infrastructure enables numerous goal-driven capabilities, including user-specifiable interactive displays based on flexible automated analyses and customizable measurement breadth, depth, granularity, and frequency. Dashboards and infrastructure can then support different organization, project, and user goals and metrics. Our future research directions include further investigation of dashboards and underlying infrastructure with a complementary focus on leading indicators such as software requirements metrics. Software requirements metrics are available much earlier in the lifecycle than source-lines-of-code, and they contribute to dashboards by providing leading indicators for system scope, growth, and stability.

References


Biography

Richard W. Selby is the Head of Software Products at Northrop Grumman Space Technology in Redondo Beach, CA. He manages a 250-person software organization and has served in this position since 2001.
Previously, he was the Chief Technology Officer and Senior Vice President at Pacific Investment Management Company (PIMCO) in Newport Beach, CA where he managed a 105-person organization for three years. From 1985-1998, he was a Full Professor of Information and Computer Science (with tenure) at the University of California in Irvine, CA (UC Irvine). In 1993, he held visiting faculty positions at the MIT Laboratory for Computer Science and MIT Sloan School of Management in Cambridge, MA, and in 1992, he held a visiting faculty position at the Osaka University Department of Computer Science in Osaka, Japan.

His research focuses on development, management, and economics of large-scale systems and processes. He has authored over 70 referred publications and given over 170 invited presentations at professional meetings. At Northrop, he led the $2.6 billion company to a successful enterprise-wide rating of Capability Maturity Model Integration (CMMI) Level 5 for Software. At PIMCO, he led the $1 billion company to be ranked as the fourth most innovative technology organization in financial services, according to Wall Street & Technology. At UC Irvine, he co-authored an international best-selling book that analyzed Microsoft’s technology, strategy, and management that was entitled Microsoft Secrets: How the World’s Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. The book, written with Michael Cusumano, has been translated into 12 languages, has 150,000 copies in print, and was ranked as a #6 best-seller in Business Week.

He received his Ph.D. and M.S. degrees in Computer Science from the University of Maryland, College Park, MD in 1985 and 1983, respectively. He received his B.A. in Mathematics from St. Olaf College, Northfield, MN in 1981.