Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2000 Proceedings

International Conference on Information Systems (ICIS)

December 2000

Measuring Software Volatility: A Multi-Dimensional Approach

Evelyn Barry
Carnegie Mellon University

Sandra Slaughter
Carnegie Mellon University

MEASURING SOFTWARE VOLATILITY: A MULTI-DIMENSIONAL APPROACH[1]

Evelyn Barry
Sandra A. Slaughter
Carnegie Mellon University
U.S.A.

Extended Abstract

The only thing constant is change. This is certainly more true of software systems than almost any phenomenon. Not all software systems change in the same way or at the same rate. Some constantly undergo major modifications and others remain untouched for years at a time. Identification and understanding of these differences in dynamic software system behavior (i.e., *software evolution*) can improve software engineering and systems management. Measurement is key to understanding any phenomenon. Software volatility, a characteristic of software behavior, describes the changeable nature of software. By rigorously defining, evaluating, and validating a measure of software volatility, we can expand our understanding of the evolutionary processes constantly transforming software systems.

While researchers have measured attributes of software products and processes, few studies measure specific characteristics of software behavior, including software volatility. Practitioners use software versioning to track software changes. However, versioning is not a reliable measure of volatility. By examining the versions of MS Windows NT released between its initial introduction and Windows 2000 released this year, we see uneven incremental growth in system size between versions and inconsistent time intervals between versions (Hamm and Port 1999). Counts of software modifications can measure software volatility (Banker and Slaughter 2000; Butcher 1997). Modification counts say nothing of the size or timing of software changes. Measures of change size examine modifications for individual programs, but not at the system-level of analysis.

Software systems model some portion of the business and economic environment they serve. As these environments change, so must software systems (Pfleeger 1998). Organizational theorists Wholey and Brittain (1989) describe environmental variation with three dimensions: amplitude, frequency, and predictability. The close connection between economic environments and software systems suggests similar dimensions could describe software volatility. Hence, we define three dimensions of software volatility: amplitude, periodicity, and deviation.

Amplitude measures the size of software modifications made to a system. Traditional measures for software size include lines of code (LOC), token counts, function point counts, equivalent size metrics, entity counts, and percentages of changed modules (e.g., Albrecht and Gaffney 1983; Conte et al. 1986). Amplitude can be operationalized using any of these metrics. We define a normalized measure for amplitude as the total size of software modifications divided by the total system size.

Periodicity measures time between software modifications, TSM. Time between software modification is the time (measured as days, weeks, or months, etc.) elapsed since the previous modification to that program. Mean time between software modification, MTSM, is the mean TSM for any program in the system. To make comparisons of MTSM between applications of different ages, we normalize periodicity by dividing MTSM by system age.

Deviation is the variance of the TSMs. If we were to use only the dimensions of amplitude and periodicity, software volatility could be described by a smooth sine curve similar to that describing physical systems, e.g., sound waves. However, software systems are rarely as well behaved as physical systems. Therefore, we use a third dimension, deviation, to indicate how closely

[1]This research was funded in part by National Science Foundation grants CCR-9988227 and CCR-9988315 and a Research Proposal Award from the Center for Computational Analysis of Social and Organizational Systems, NSF IGERT.

system behavior follows the cyclical pattern described by periodicity. We normalize deviation by dividing variance of the TSMs by system age squared.

Newly defined measurement functions should be rigorously evaluated and validated to see that they logically behave in a manner consistent with the real world phenomena they model. We approach this task in two steps. First, we evaluate the measurement functions defined for amplitude, periodicity and deviation for appropriate logical and mathematical properties. Then we validate these measures with empirical data from a software portfolio of legacy application systems. This validation includes convergent, discriminant and predictive validity.

A set of logical and mathematical evaluation criteria are built from measurement theory and evaluation criteria used for other software measurement studies (e.g., Chidamber and Kemerer 1994; Fenton and Pleeger 1997; Weyuker 1988). Software volatility measures are aggregate measures of longitudinal behavior. If all system programs are stable, system-level software volatility measures should reflect their combined volatility or stability. Software volatility measures should discriminate between different levels of volatility throughout a system's lifecycle (i.e., non-coarseness). To allow for maximum flexibility in application of these metrics, software volatility measures should be bounded above and below, be scale invariant and technology independent. The normalized measurement functions defined for amplitude, periodicity, and deviation comply with all these criteria.

Empirical measurement validation serves as a reality check to see that a measure accurately characterizes the real world behavior it claims to measure (Fenton and Pfleeger 1997). We demonstrate convergent validity with substantial correlations between each of our measurement functions and a logically comparable measure. Discriminant validity is demonstrated by weak correlations between a dimensional measure of software validity and each of the other two. We empirically demonstrated both convergent and discriminant validity with data from a 20 year log software modifications for 23 systems.

Predictive validity is demonstrated by showing that a new measurement is an explanatory variable in predicting future outcomes. We use a simple predictive model of software complexity for this purpose. Our model uses software volatility dimensions of $amplitude_{t-1}$, $periodicity_{t-1}$ and $deviation_{t-1}$ as explanatory variables to predict software $complexity_t$. The dependent variable was operationalized by six different software complexity metrics each normalized by total system size. Regression analyses were estimated using ordinary least squares procedures. Strong evidence of predictive validity was demonstrated by the significance of the estimated regressions (maximum adjusted R-square 0.8742). Estimates for these same regressions using the three comparable measures as explanatory variables were able to achieve adjusted R-squares no higher than 0.3007.

By defining software volatility measures this work provides researchers with objective metrics to investigate software evolution. Viewing software volatility as a system-level attribute and developing a multi-dimensional picture of this activity provides a direct measure of software behavior. Rigorous evaluation and validation of these measures establishes their credibility and lays the groundwork for theory building. The longitudinal application of these measures gives researchers a more complete picture of the dynamic nature of software lifecycle behavior.

References

Albrecht, A. J., and Gaffney, J. E., Jr. "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering* (9:6), November 1983, pp. 639-648.

Banker, R. D., and Slaughter, S. A. "The Moderating Effects of Structure on Volatility an Complexity in Software Enhancement," *Information Systems Research*, September 2000.

Butcher, G. *Addressing Software Volatility in the System Life Cycle*, Unpublished Doctoral Dissertation, Colorado Technical University, 1997.

Chidamber, S. R., and Kemerer, C. R. "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering* (20:5), June 1994, pp. 476-493.

Conte, S., Dunsmore, H., and Shen, V. *Software Engineering Metrics and Models*, Menlo Park, CA: Benjamin/Cummings, 1986.

Fenton, N. E., and Pfleeger, S. L. *Software Metrics: A Rigorous Approach* (2nd ed.), New York: Chapman & Hall, 1991.

Hamm, S., and Port, O. "The Mother of All Software Projects," *Business Week*, February 22, 1999, pp. 69-76.

Pfleeger, S. L. *Software Engineering: Theory and Practice*, Upper Saddle River, NJ: Prentice Hall, 1998.

Weyucker, E. "Evaluating Software Complexity Measures," *IEEE Transactions on Software Engineering* (14), 1988, pp. 1357-1365.

Wholey, D. R., and Brittain, J. "Characterizing Environmental Variation," *Academy of Management Journal* (32:4), 1989, pp. 867-882.