8-16-1996

Modeling the Performance of Software Processes Quantitatively

David M. Raffo
School of Business Administration, Portland State University, davidr@sba.pdx.edu

Modeling the Performance of Software Processes Quantitatively

David M. Raffo
School of Business Administration
Portland State University
Portland, OR 97207-0751
Internet: davidr@sba.pdx.edu

Despite the fact that software has become one of this countries major industries, difficulties still exist with delivering quality software within budget and on schedule [18]. As a result, improving software processes is a major focus for many organizations. Software project managers are concerned with questions such as:

- What development phases are essential? Which phases could be skipped or significantly minimized in order to reduce costs without sacrificing quality?
- Are inspections worthwhile?
- What is the value of applying tools to support development activities? Will these tools be worth the cost?
- How do we predict the benefit associated with implementing a process change before a substantial commitment of resources and effort is made?

Given a number of potential process changes, how do we prioritize these changes?

These questions are examples of the general questions addressed by our research. These are:

- How do we compare alternative software processes?
- What is the impact of a potential process change on process performance?
- How do we prioritize process changes?

To evaluate these questions, quantitative models which deal with process level issues are required. We review several quantitative approaches that have been used to model software development projects. Although these modeling approaches do support certain project management decisions, process modeling is shown to be superior for addressing the process focused questions posed above. We then present a dynamic simulation based quantitative process model that has been used to predict the impact of a process improvements using a benchmark software process and the results from the study.

Alternative Modeling Approaches

Three distinct approaches have been used to quantitatively model software development projects. They are Analytic Summary Models, Analytic Structural Models, and Process Models.

Most existing quantitative models are **analytic summary models** (as defined in [11]) which focus on high-level quantitative relationships between input and output parameters. These models support management decision making regarding issues relating to project planning and productivity such as expected manpower, schedule, and so forth. Some examples of models in this category are COCOMO [3] and SLIM [20] which are probably the most widely known models of this type. These models view the software process as a "black box" that is not specified in detail. As a result, these models do not capture many important details which are related to process changes.

Analytic Structural Models are another class of models which have been used to capture the interrelations, dependencies, and structure of software development projects at a deeper, more detailed level than Analytic Summary Models. The best known example of this type of model is the systems dynamics model developed by Abdel-Hamid and Madnick [1]. This work richly captures relationships between manpower,

programmer productivity, and work activities completed. Segments of the model are devoted to high level process blocks of software development, quality assurance, and testing. However, no process details are included. If Analytic Summary Models consider the software process as a "black box", then Analytic Structural Models consider the process as "several black boxes making up a system". Other recent work using Analytic Structural Models which has been based on the work by Abdel-Hamid and Madnick includes [15] and [19].

Process Models [3] support process improvement by providing operational guidance regarding the critical sequence of process steps, information flows, and organizational responsibilities. These models are distinct from Analytic Summary and Analytic Structural Models because they delve into the details of the process used to develop software [17], [16], [2], [14], [8] and [6]. Much of this work to date has been in developing tools and methods for modeling software processes. However, process models can be used to quantitatively analyze and assess a variety of issues.

Work conducted by Kellner shows how process models could be used to support management planning and control activities [12] using both deterministic and stochastic cases. The model presented in this paper extends Kellner's work [12], and incorporates analytic and empirical components into process models as well. This creates a synergistic combination which expands the range of issues that can be quantitatively assessed using a process modeling approach. The results obtained from this quantitative assessment can be used to develop a business case to obtain management support and "buy in" of a process change effort.

Evaluation of Alternative Models

In order to quantitatively assess the tradeoffs associated with a change in a software development process, models which capture process level information and details will be required. In addition, the ability to (1) conduct sensitivity analyses for testing different process assumptions in varied operational contexts, and (2) capture complex and uncertain interrelationships found in software processes, will be necessary.

Process modeling using a stochastic simulation based approach offers the desired level of granularity to capture salient features of process dynamics while being able to scope up to a level that is useful for planning and management activities. In addition, process modeling addresses uncertainties related to task durations, effort, number of errors, outcomes of reviews and tests, etc., which are inherent in real world processes.

Although Analytic Summary and Analytic Structural Models are valuable and provide predictive power to support certain management decisions, they do not capture process details. Capturing process level issues is the key ingredient to being able to make decisions which evaluate the effectiveness process alternatives, the value of particular process phases or the impact of a particular process change.

Modeling Approach and Application

Our approach uses dynamic simulation models of each process alternative. The approach features careful modeling of process steps, information flows, and organizational responsibilities. In addition, there is explicit modeling of complex interdependencies among process components. Using this information, the model predicts development costs (effort), project schedule, and quality (remaining defects) that results from the process being used.

The quantitative predictions are made possible using a new technique called Task Element Decomposition which was developed to manage the computation of interdependent operation times in large-scale projects. The Task Element Decomposition technique enables a variety of product and process factors to be taken into account [21]. These factors can include source code size, quality, and complexity and developer skill. Looping and operation time dependencies with uncertainties are constructed to quantitatively measure important performance characteristics of the system and to support extensive sensitivity analyses. This

enables the process model to be used to evaluate alternative processes and to assess the impact of potential process changes.

Once this model is developed, realistic process scenarios can be run to test important process related contingencies. A multi-attribute decision making framework (employing utility functions) is then used to tradeoff among the three performance measures of cost, quality, and schedule in order to compare the overall performance of the process alternatives.

Using this approach decisions such as:

- Comparing process alternatives based on similar process parameters
- Go/no-go on individual proposed changes
- Prioritization and selection from among several proposed changes can be supported. Moreover, the results obtained from this quantitative assessment can be used to develop a business case to obtain management support and "buy in" of a process change effort.

This approach has been applied to the benchmark ISPW-6 Software Process Example developed by Kellner et al. [13]. This process "was carefully designed to incorporate examples of eighteen important process aspects and issues" and contains "a diversity of process aspects encountered in real-world software processes" [5]. The example process contains, "a relatively confined portion of the software change process. It focuses on the designing, coding, unit testing, and management of a rather localized change to a software system"[12]. The main steps of the "original" process include: Schedule and Assign Tasks, Modify Design, Review Design, Modify Code, Modify Test Plans, Modify Unit Test Package, Test Unit, and Monitor Progress. The "alternative" process used for comparison contains the above steps plus steps for a code inspection.

Parameters for the model were developed from sources in the literature [3, 7, 10] to test the impact of inserting a source code inspection process on overall process performance. Dynamic simulations of the process models were run using a variety of process scenarios. The questions which were investigated included:

- Do source code inspections offer an improvement in process performance (using data reported in the literature)?
- What would be the impact if the error detection capability of Source Code Inspections was reduced?
- Would Source Code Inspections still be worth while if the lowest reported error detection capability in the literature was achieved?
- If there was only time to conduct either a Source Code Inspection or a Unit Test, which method would be more effective?
- Would Source Code Inspections still be beneficial if the starting source code had more errors?

Quantitative results were provided in the form of distributions for the three process performance measures studied - cost (effort), quality (remaining defects), and schedule (task duration). The results of the study showed the following:

- Source Code Inspections significantly increase the quality of the resulting code.
- The Source Code Inspection process is an efficient error detection method if implemented correctly.
- Inspections make even greater impact when the starting quality of the code is poor.
- Even if poor Inspection performance is realized as compared to results reported in the literature, the process with Source Code Inspections and Unit Test offers an overall improvement.
- If pressed for time, it is better to reduce Unit Testing rather than reduce Inspections.

- It is worth implementing inspection procedures well in order to obtain the maximum return. Inspections are a high leverage activity.

The results obtained compared well with results reported from actual field studies of source code inspections found in the literature.

Conclusion

A quantitative process modeling approach is being developed. Preliminary tests show the approach to be well suited for analyzing process level issues in software development. Using this process modeling approach, decisions such as:

- Consistent comparisons of process alternatives based on similar process parameters
- Go/no-go on individual proposed changes
- Prioritization and selection from among several proposed changes

can be addressed. The results obtained from this quantitative assessment can be used to develop a business case to obtain management support and "buy in" of a process change effort.

References

(Available upon request via e-mail from the author)

[1] Abdel-Hamid, T. and Madnick, S., "Software Project Dynamics: An Integrated Approach", Prentice-Hall Software Series, 1991, ISBN 0-13-822040-9.

[2] M. Akhavi and W. Wilson "Dynamic Simulation of Software Process Models", Proceedings of the 5th Software Engineering Process Group National Meeting (Held at Costa Mesa, California, April 26 - 29, 1993), Software engineering Institute, Carnegie Mellon University, 1993. Presentation in Session Three, Advanced Track.

[3] Boehm, B., "Software Engineering Economics", Prentice-Hall, 1981 ISBN 0-13-822122-7

[4] Boehm, B.W., "Improving Software Productivity", Computer, September 1987, pages 43-50.

[5] Curtis, B., Kellner, M. I., Over, J., "Process Modeling", Communications of the ACM, Vol. 35, No. 9, September, 1992.

[6] W. Dieters and V. Gruhn, "Software Process Model Analysis Based on FUNSOFT Nets", *Mathematical Modeling and Simulation*, (8), May 1991.

[7] Fagan, M. E., "Design and Code Inspections to Reduce Errors in Program Development", IBM Systems Journal, vol. 15, no. 3, 1976

[8] V. Gruhn, "Software Process Simulation on Arbitrary Levels of Abstraction", *Computational Systems Analysis 1992*, pp. 439-444, Amsterdam, The Netherlands, 1992. Elsevier.

[9] Harel, D., et al., "STATEMATE: A Working Environment for the Development of Complex Reactive Systems", IEEE Transactions on Software Engineering 16,4 (April 1990), pp. 403-414.

[10] Humphrey, Watts, "Managing the Software Process", Addison Wesley, 1989, ISBN 0-201-18095-2

[11] Kellner, M. I., "Modeling the Software Maintenance Process: Analytic Summary Models" *Proceedings of the Conference on Software Maintenance - 1988* (Held in Phoenix, Arizona October 24 - 27, 1988), IEEE Computer Society Press, 1988, pp. 279 - 283.

[12] Kellner, M. I., "Software Process Modeling Support for Management Planning and Control", *Proceedings of the First International Conference on the Software Process* (Held at Redondo Beach, California, USA, October 21-22, 1991) IEEE Computer Society Press, 1991, pp. 8-28.

[13] Kellner, M. I., Felier, P. H. Finkelstein, A., Katayama, T., Osterweil, L. J., Penedo, M. H., Rombach, H. D., "ISPW-6 Software Process Example" in Proceedings of the First International Conference on the Software Process (Held at Redondo Beach, California, USA, October 21-22, 1991), IEEE Computer Society, Washington, DC., 1991, pp. 176-186.

[14] M. Lehman, "Software Engineering, the Software Process and Their Support" IEEE Software Engineering Journal, v6 n5, September 1991.

[15] Madachy, R., "A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment", Ph.D Dissertation, Department of Industrial and Systems Engineering, University of Southern California, December 1994.

[16] N.H. Madhavji and W. Schafer, "PRISM - Methodology and Process-Oriented Environment", IEEE Transactions on Software Engineering, v17 n12, December 1991 pp. 1270 - 1283.

[17] Mi, P, and Scacchi, W., "Modeling Articulation Work in Software Engineering Process", Proceedings of the First International Conference on the Software Process, IEEE Computer Society, Washington, DC., 1991, pp. 188-201.

[18] Mills, H. D., "Software Engineering: Retrospect and Prospect", *The Twelfth Annual International Computer Software & Applications Conference (COMPSAC)*, October, 5 -7, 1988, pp. 89-96.

[19], Nguyen, N. and Ahmed, S., "Dynamic Software Process Modeling", Presentation given at the 7th National Software Engineering Process Group Conference (Held in Boston, MA, May 22-25), 1995.

[20] Putnam, L. H., "Software Cost Estimating and Life-Cycle Control; Getting the Software Numbers, New York: The Institute of Electrical Engineers, Inc., 1980.

[21] Raffo, David "Correlated Operation Time Models in Manufacturing and Service Industries", WP# 1993-15, Graduate School of Industrial Administration, Carnegie Mellon University.