

Quantifying the Effects of Process Improvement on Effort

Bradford K. Clark, Ph.D.
Software Metrics, Inc.

Abstract

There is a good deal of anecdotal case-study evidence that organizations that have improved their SEI-CMM process maturity level have been able to reduce the amount of effort it takes them to produce a given software product. However, these case-study projects have concurrently realized other improvements in such areas as tool use, domain knowledge, component reuse, and office efficiency. As a result, managers have had no way of determining how much improvement is due to process maturity versus the other factors.

This paper reports on an analysis in which the effects of process maturity were broken out from the other effects on effort. Using a 161-project sample, a change in one level of process maturity resulted in a reduction of development effort from 4 to 11 percent.

SW-CMM Experience

The Software Engineering Institute has published a Software Capability Maturity Model (SW-CMM) that can be used to assess an organization's software process maturity [1]. An important question is what are the benefits of investing resources to improve an Organization's software development process maturity. The primary intended long-term benefit of high process maturity is predictability: software delivered on time, within budget, within customer requirements, and of high quality. Another important benefit would be to improve productivity.

Much has been written discussing the short-term and long-term benefits of increasing maturity levels [2,3,4,5,6,7,8,9]. It requires a large amount of dollar investment by an organization to change the software development process within the organization and to realize an increased level of maturity. Many software organizations are at low levels of maturity.

The effects of increasing process maturity alone are not easy to determine, as organizations are generally making concurrent improvements in other areas that result in benefits to the development organization. However there is a need for a clearer assessment of Process Maturity effects. The case studies show that there are many benefits to improving Process Maturity. The conclusions in the case studies used different assessment approaches, none of which attempt to separate out individual

factors that affected productivity. Even with this incomplete analysis, the indication is that increasing maturity levels has generally positive effects.

The premise of the results presented here is that increasing process maturity decreases the effort required to develop a software product (effort is a fundamental component of productivity). The challenge is determining the effect that increasing process maturity has on effort within the context of other factors that influence software development effort. A mathematical model was used in the analysis presented here to segregate process maturity's influence on effort from other influencing factors. The model quantifies the magnitude of the effect of Process Maturity on effort and shows the quantified relationship between process maturity and other effort influencing factors. The results here are based on work presented in [10,11].

Data Collection

There are generally four areas that influence software development effort. They are product factors, project factors, platform factors, and personnel factors. Data collected on product factors were size, precedentedness, architecture and risk resolution, required reliability, database size, complexity, whether the product was developed for reuse, and documentation match to lifecycle needs. Data collected on platform factors were time constraint, storage constraint, and development platform volatility. Data collected on personnel factors were analyst and programmer capability, personnel continuity, team application experience, and language and tool experience. Data collected on project factors were development flexibility, team cohesion, tool usage, multi-site development, schedule compression, and, the factor of interest here, process maturity (PMAT). All of the factors are fully described in detail in the COCOMO II model. [11]

Data for the Process Maturity factor (PMAT) was collected using one of two methods. The first method selects an overall maturity level based either on an organized evaluation or subjective judgment, Table 1. The selection for SW-CMM Level 1 (lower half) is for organizations that rely on "heroes" to get the job done. There is no focus on processes or documenting lessons learned. The SW-CMM Level 1 (upper half) is for organizations that have implemented most of the KPA's that would satisfy

SW-CMM Level 2. It is important to distinguish the groups working their way to a Level 2 rating and a transition from Level 1 Lower to Level 1 Upper is modeled as a change in a PMAT level. These two Level 1 ratings are a departure from the published definition of the SW-CMM. The remaining levels follow the SW-CMM discussed earlier.

Table 1. PMAT Rating Levels

PMAT Rating	Maturity Level	EPML
Very Low	CMM Level 1 (lower half)	0
Low	CMM Level 1 (upper half)	1
Nominal	CMM Level 2	2
High	CMM Level 3	3
Very High	CMM Level 4	4
Extra High	CMM Level 5	5

The second method selects a rating, called Equivalent Process Maturity Level (EPML), based in the percentage of compliance for each of the Key Process Area (KPA) goals taken from [1]. If the project has undergone a recent CMM Assessment, then the percentage compliance for the overall KPA (based on KPA Key Practices compliance assessment data) is used. If an assessment has not been done, then the levels of compliance to the KPA's goals are used (with the Likert scale as shown in Figure 1) to set the level of compliance. The goal-based level of compliance is determined by a judgement-based averaging across the goals of each Key Process Area. See [1] for more information on the KPA definitions, goals and activities.

While an organization may be rated at a specific SW-CMM level, the respondents were encouraged to use the second method, i.e. answer all of the KPA questions considering *what actually happened on the project*. A summary of one KPA and its compliance levels is given in Figure 1.

There are eighteen KPA's and each has seven rating levels. The rating levels are defined as:

- Almost Always: When the goals are consistently achieved and are well established in standard operating procedures (over 90% of the time).
- Frequently: When the goals are achieved relatively often, but sometimes are omitted under difficult circumstances (about 60 to 90% of the time).
- About Half: When the goals are achieved about half of the time (about 40 to 60% of the time).
- Occasionally: When the goals are sometimes achieved, but less often (about 10 to 40% of the time).
- Rarely If Ever: When the goals are rarely if ever achieved (less than 10% of the time).

- Does Not Apply: When you have the required knowledge about your project or organization and the KPA, but you feel the KPA does not apply to your circumstances.
- Don't Know: When you are uncertain about how to respond for the KPA.

Requirements Management KPA: involves establishing and maintaining an agreement with the customer on the requirements for the software project.

Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Goal 2: Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Almost Always
Frequently
About Half
Occasionally
Rarely if ever
Does not apply
Do not know

Figure 1. KPA Data Example

An Equivalent Process Maturity Level is computed as five times the average compliance level of all n-rated KPAs for a single project (Does Not Apply and Don't Know are not counted which sometimes makes n less than 18). After each KPA is rated, the rating level is weighted (100 percent for Almost Always, 75 percent for Frequently, 50 percent for About Half, 25 percent for Occasionally, 1 percent for Rarely If Ever). The EPML is calculated as in Equation 1.

$$EPML = 5 \times \left(\sum_{i=1}^n \frac{KPA\%_i}{100} \right) \cdot \frac{1}{n} \quad \text{Eq. 1}$$

To convert EPML to a PMAT level use Table 1. An EPML of 0 corresponds with a PMAT rating level of Very Low in the rating scales. This is discussed in more detail later.

The data used in this analysis is on 161 projects from eighteen sources. These sources cover the Aerospace Industry, Federally Funded Research Centers, Commercial Industry, and Department of Defense supported Industry. The data was on past, completed projects. Of the 161 projects analyzed, sixty five percent came from 1990's projects. Earlier projects were assessed using SW-CMM levels based on interviews with people familiar with the project. Much of the data is proprietary and furnished to the University of Southern California under nondisclosure agreements where this research was conducted.

The data had the following statistics:

- Product sizes range from 2.6 to 1,264.0 thousands of lines of code (KLOC). The KLOC size data has an average of 131 and a median of 47, Figure 2.
- Project effort ranges from 6 to 11,400 Person Months. PM data has an average of 711 Person Months and a median of 195 Person Months, Figure 3.

The effort data largely comes from individual time reporting, which is generally accurate to no better than fifteen percent. Similar variations occur in reported size data across different organizations. While it was requested, uncompensated overtime was not consistently collected. These and other factors, such as the interpretation of qualitative ratings, mean that the data are imprecise. The results are presented with a confidence interval.

Even though the data sources varied there is selection bias in the data. We were not given data on unfinished or unsuccessful projects nor did any unsuccessful companies contribute data. The data is from successful projects from successful companies. Proof of this is in the fact that these companies are mature enough to practice collecting data and that the project data is from completed projects. Process maturity levels in both data sets covered the full range, see Figure 4. This is due to the selection bias mentioned earlier and the higher emphasis on data collection and analysis at the higher process maturity levels by organizations that contributed data. Thirty one percent of the projects provided KPA level data.

For the results presented here, development effort is measured in Person Months. A person month is 152 hours. It includes the software developer's time, project management time, administrative support time, and project support personnel time, e.g. configuration management and quality assurance. The period measured on a project was from completion of requirements analysis to the end of integration and test (again differences in interpreting these boundaries provide another source of data imprecision).

Existence of PMAT Effort Reduction

The results of data analysis from 161 projects showed that PMAT is statistically significant in supporting the hypothesis that increasing process maturity decreases the effort required developing a software product. How this result was achieved is discussed in this section.

Process Maturity's effect on effort was analyzed in a full calibration of the COCOMO II model using Bayesian regression analysis. A simplified form of the model is shown in Equation 2. The calibration of

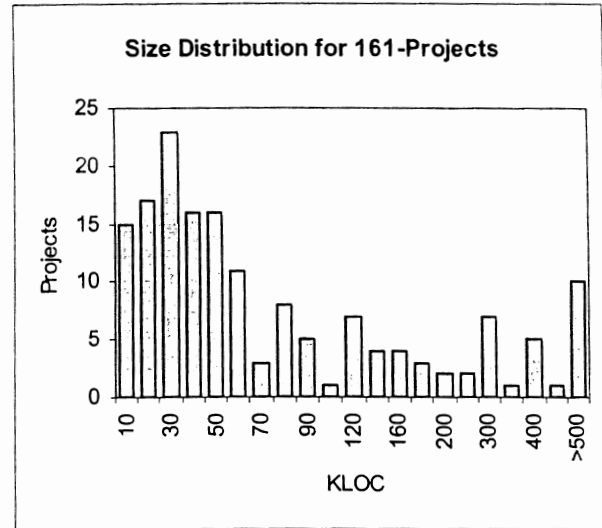


Figure 2. KLOC Distribution

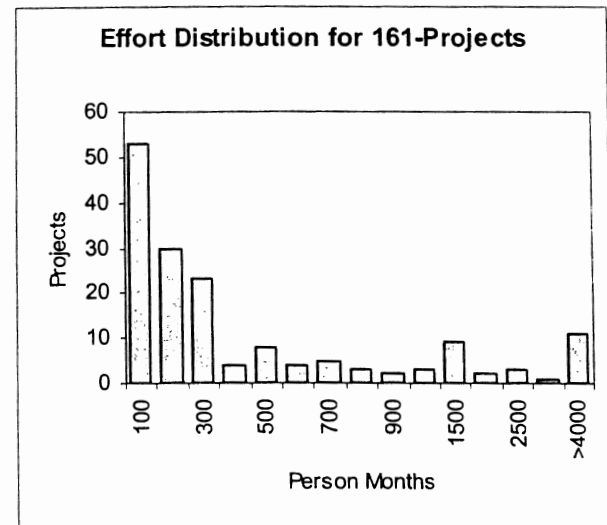


Figure 3. PM Distribution

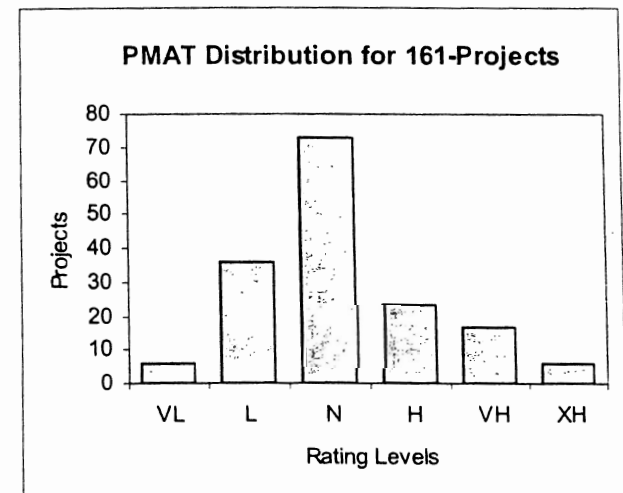


Figure 4. PMAT Distribution

the model with the 161 project observations showed that PMAT was stronger at explaining variation in actual project data effort if it was used as an exponent to the project size, the B in Equation 2 [10]. As the size of the project increased, the level of process maturity made more of an impact on development effort.

$$\text{Person Months} = A \times (\text{Size})^B \times C \quad \text{Eq. 2}$$

As can be seen in the bottom row of values in Figure 5, a PMAT level is converted to a number for use in Equation 2. The sequence of values assigned to the PMAT rating levels decreases from Very Low to Extra High to test the premise stated earlier that as higher levels of Process Maturity are attained (moving towards the Extra High rating) the software development effort decreases.

There was variation in the analysis results. Figure 6 shows a one level change in PMAT. The

Maturity Level	Lvl 1 Lower	Lvl 1 Upper	Lvl 2	Lvl 3	Lvl 4	Lvl 5
EMPL	5	4	3	2	1	0
	Very Low	Low	Nominal	High	Very High	Extra High
Values	0.0780	0.0624	0.0468	0.0312	0.0156	0.0000

Figure 5. PMAT Scale Values

upper and lower dotted lines show the variation in the results at the 95% confidence level. The dotted lines do not include zero or negative values which confirm the conclusion:

With at least 95% confidence, even after normalizing for the effects of other cost drivers, an increase in Process Maturity corresponds with a reduction in project effort.

These results were recently confirmed in other research using a different approach [12].

A one level change in PMAT, e.g. from Nominal to High, is applied to a range of project sizes to show the percentage reduction in effort, Figure 6. At a 95% confidence level, process maturity reduces effort required to develop software from between 3% to 15% as shown by the lower and upper dotted lines in Figure 6 respectively. The average is between 4% and 11% reduction shown by the solid line.

In addition to analyzing PMAT's influence on effort, the strength of PMAT among other factors influencing effort can be observed. The productivity range (the total percentage a factor can influence effort going from the lowest to the highest factor rating) for all factors is shown in Figure 7. The

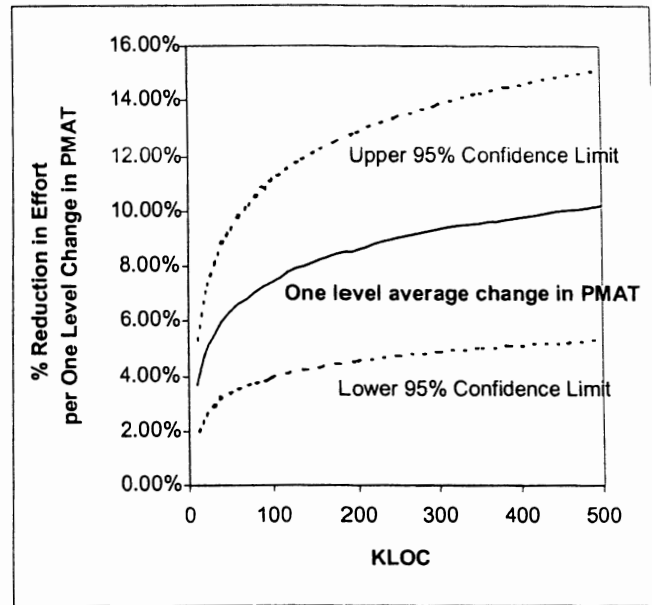


Figure 6. % Reduction in Effort per PMAT Level

productivity range for the factors that are used with size, B in Equation 2 and in which PMAT is a part, is based on a 100 KLOC project. For a 100 KLOC project, PMAT's productivity range is 1.43. This means it can change effort to develop a software product up to 43% from the lowest PMAT level to the highest. PMAT's productivity range would increase from 1.43 to 1.62 for a 500 KLOC project and to 1.71 for a 1000 KLOC project. If your typical project size is 10 KLOC the PMAT productivity range will be 1.20.

Conclusions

This paper examines process maturity's effect on effort, which is a fundamental component of productivity, within the context of other factors. For the 161 projects, Software Process Maturity was found to be statistically significant in reducing software development effort, even after normalizing for the effects of other effort influencing factors. For projects ranging in size from 2 to 1,000 KSLOC, a one-increment change in the rating of Process Maturity resulted in a 4% to 11% reduction in effort, Figure 6.

Depending on product size, the amount of influence PMAT has on effort is less than some other factors influencing software development effort. This would suggest that if productivity improvement were an organization goal, process maturity should be considered as one part of multiple strategies to achieve that goal. In addition to PMAT, the other strategies would include improving the condition of "management controllable" factors such as analyst

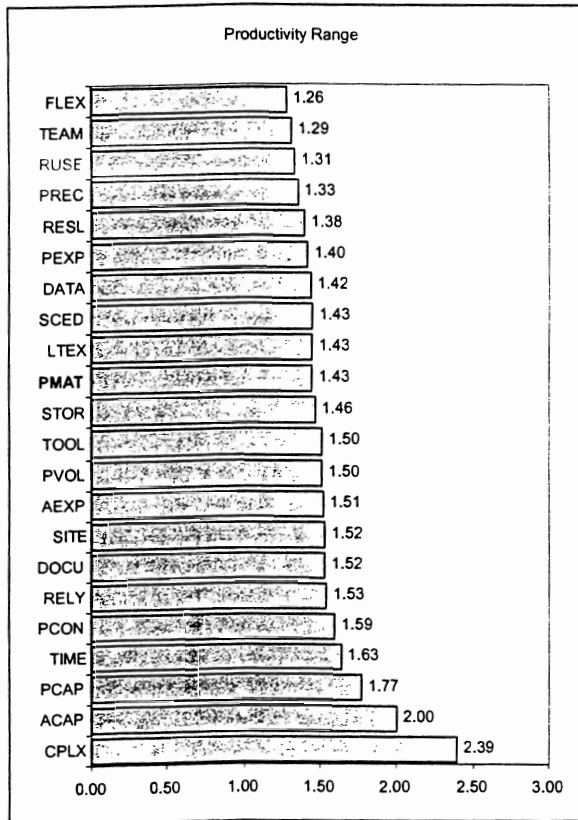


Figure 7. PMAT Comparison to Other Factors

capability, programmer capability, and personnel continuity. This analysis supports the conclusion that increasing an organization's software development process maturity will not cost productivity.

The analysis presented here is a generalization across all levels of process maturity. There is reason to believe that the percentage reduction in effort is not uniform between levels. More data on process maturity collected at the KPA level would permit quantification of the change between each level.

As a result of studying Process Maturity's effect on effort, it seems reasonable to suggest that it is an important input to software cost estimation models. The SEI Capability Maturity Model is well defined. It establishes criteria to evaluate processes used to develop software. The Process Maturity factor provides a significant assessment of the effects of process on development effort.

Future work needed in this analysis area requires collection of more KPA data. This would be used to assess which KPA's have the most influence on effort. Implementing the effort saving KPA's first would offset the costs of implementing other KPA's. Based on the KPA results, the model would be refined to capture any nonuniform improvements in going from CMM Level n to Level $(n+1)$. The 161-project COCOMO II data sample has not been

sufficient to establish such assessments with statistical significance.

Acknowledgements

The author wishes to recognize the support of the Center for Software Engineering at the University of Southern California where this research was conducted.

References

- [1] M.C. Paulk, C.V. Weber, B. Curtis, and M.B. Chrissis. The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading, Ma., 1995.
- [2] J.G. Brodman and D.L. Johnson, "Return on Investment (ROI) from Software Process Improvement as Measured by US Industry," Software Process Improvement and Practice, John Wiley & Sons Ltd., Sussex, England and Gauthier-Villars, 1995, pp. 35-47.
- [3] K. Butler, "The Economic Benefits of Software Process Improvement," Crosstalk, Hill AFB, Ogden, Ut., 1995, pp. 14-17.
- [4] R. Dion, "Process Improvement and the Corporate Balance Sheet," IEEE Software, October 1993, pp. 28-35.
- [5] J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, D. Zubrow, "Benefits of CMM-Based Software Process Improvement: Initial Results," CMU/SEI-94- TR-13, Software Engineering Institute, Pittsburgh, Pa., 1994.
- [6] W.S. Humphrey, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, August 1991, pp. 11-23.
- [7] T. McGibbon, "A Business Case for Software Process Improvement," Contract Number F30602-92-C-0158, Data & Analysis Center for Software (DACs), Kaman Sciences Corp., Utica, NY, 1996.
- [8] B. Springsteen, B. Brykczynski, D. Fife, R. Meeson, and J. Norris, "Policy Assessment for the Software Process Maturity Model," IDA D-1202, Institute for Defense Analyses, Alexandria, Va., 1992.
- [9] H. Wohlwend and S. Rosenbaum, "Schlumberger's Software Improvement Program," IEEE Transactions on Software Engineering, November 1994, pp. 833-839.
- [10] B.K. Clark, "The Effects of Software Process Maturity on Software Development Effort," Ph.D. Dissertation, University of Southern California, Los Angeles, Ca. 90089, August 1997 (available at <http://sunset.usc.edu/~bkclark/Research>).

- [11] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, Upper Saddle River, NJ, June 2000.
- [12] D. Harter, "The Life Cycle Effects of Software Process Maturity," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, Pa., April 2000.