

# Effects of Process Maturity on Development Effort

Bradford K. Clark  
Center for Software Engineering  
University of Southern California  
Los Angeles, CA 90089-0781

## 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 several analyses in which the effects of process maturity were broken out from the other effects via calibration of the COCOMO II software cost estimation model. Using a 112-project sample a change in one level of process maturity resulted in a reduction of development effort from 10 to 32 percent. 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.

## Introduction

There are many companies and government organizations that develop or maintain software to support their operations or their business products. The development of software includes the creation of specifications, design, source code, and test artifacts. These different artifacts interact with each other so that a delay or defect in one affects the completion of the others. This often results in a software product that is behind schedule, over budget, non-conforming to requirements and of poor quality. The outcome is that the company loses money or the customer experiences budget overruns or decreased user and customer satisfaction. Controlling and improving the process used to develop software is seen as one remedy to these problems [1].

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 [2]. The motivation behind the SW-CMM is that a mature software development process will deliver the product on time, within budget, within requirements, and of high quality.

The SW-CMM is explained in the next section. Section 3 discusses industry experience with the SW-CMM. Section 4 discusses the model used to separate Process Maturity's influence from other influencing factors. Section 5 presents data collection characteristics and analysis. The paper ends with conclusions and future work.

## CMM-based Process Maturity

The Software Capability Maturity Model (SW-CMM) provides a set of requirements that organizations can use in setting up the software processes used to control software product development. The SW-CMM specifies "what" should be in the software process but not "when" or "for how long." The SW-CMM has what is called a process maturity framework [2]. There are five levels of process maturity, Level 1 (lowest) to Level 5 (highest). To be rated at a specific level an Organization has to demonstrate capabilities in a number of Key Process Areas (KPA) associated with a specific SW-CMM level, Table 1. The capabilities demonstrated in transitioning from lower levels to higher levels are cumulative. In other words, a Level 3 Organization must demonstrate compliance with all Level 2 and Level 3 KPA's.

The Process Maturity framework is presented in Table 1. All Organizations start at Level 1. This is called the Initial level. At this level few processes are defined, and success depends on individual effort. This makes the software process unpredictable because it changes as work progresses. Schedules, budgets, functionality, and product quality are generally unpredictable.

To achieve Level 2 the organization demonstrates capability in 6 KPA's, see Table 1. A Level 2 Organization has basic management processes in place to track cost, schedule, and functionality. The project works with its subcontractors to maintain cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. Level 2 is called the Repeatable level.

A Level 3 Organization has demonstrated capabilities in an additional 7 KPA's, Table 1. At this level the software processes for both management and engineering activities are documented,

Software CMM	Key Process Areas
Level 1	None
Level 2 Repeatable	Requirements Management
	Software Project Planning
	Software Project Tracking and Oversight
	Software Subcontract Management
	Software Quality Assurance
Level 3 Defined	Software Configuration Management
	Organization Process Focus
	Organization Process Definition
	Training Program
	Integrated Software Management
	Software Product Engineering
Level 4 Managed	Intergroup Coordination
	Peer Reviews
	Quantitative Process Management
Level 5 Optimizing	Software Quality Management
	Defect Prevention
	Technology Change Management
	Process Change Management

**Table 1. CMM Framework [2]**

standardized, and integrated into a standard software process for the whole organization. Projects tailor the standard software process to develop their own unique defined software process. Level 3 is called the Defined level.

A Level 4 Organization adds 2 more KPA's to its capabilities. At this level detailed measures of the software process and product quality are collected. Both the process and product are quantitatively understood and controlled. Level 4 is called the Managed level.

At Level 5 an Organization has capabilities in 3 more KPA's and is in a continuous improvement state. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. Level 5 is called the Optimizing level.

Each KPA has a set of goals, capabilities, key practices, measurements and verification practices. The goals and key practices are the most interesting of these because they could be used to assess the impact of a KPA on a project's development effort. The goals state the scope, boundaries, and intent of a KPA. A key practice describes "what" should happen in that KPA. There are a total of 52 goals and 150 key practices. All of the KPA's are described in [2].

Process Maturity as specified by the SW-CMM does not address all areas that affect productivity on a software development project such as development methodologies, technologies, standards, and the need for qualified people.

### SW-CMM Experience Data

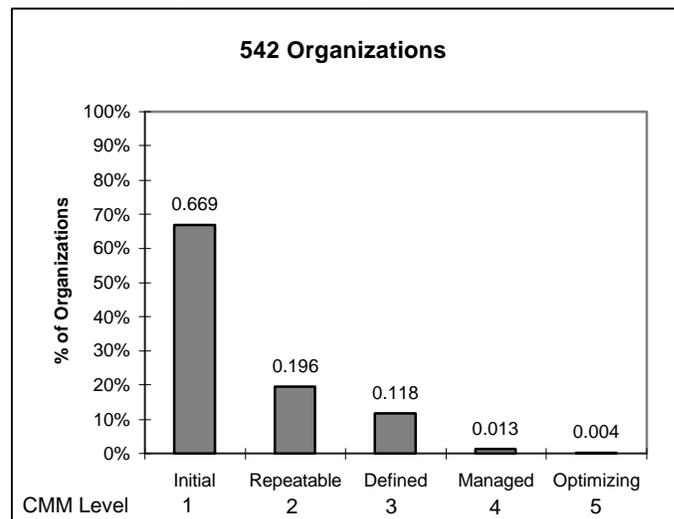
An important question for industry and government is what are the benefits of investing

resources to improve the Organization's 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 [3,4,5,6,7,8,9,10]. 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. Figure 1 shows a distribution of companies assessed using the SW-CMM.

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 hypothesis of the work 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 is 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



**Figure 1. Process Maturity Distribution [11]**

Process Maturity on effort and shows the quantified relationship between process maturity and other effort influencing factors.

## COCOMO II Model

The COCOMO II model estimates effort (Person Months) and schedule required to complete a software development project [12]. The model is also used for analysis. With calibration to real world project data, the model can also be used to analyze the influence that different parameters have on effort. Additionally the calibrated parameters can be used to understand the relative strengths between parameters, i.e. which parameters have the most or least influence. This is why the model is used here to analyze process maturity's effect on effort. The analysis results are discussed later.

There are generally four areas that influence software development effort. They are product factors, project factors, platform factors, and personnel factors. The COCOMO II model has parameters in these four areas. It takes as input the estimated product size, including code developed and discarded. It has parameters that have multiplicative and exponential effects on effort. The set of parameters that influence the model multiplicatively are represented by values called effort multipliers (EM), see Equation 1. There are seventeen effort multipliers in the model. They represent factors that influence effort in the four areas discussed above. Table 2 at the end of this paper lists the multiplicative factors determining the effort multipliers and their associated rating scales.

$$PM = A \cdot (Size)^B \cdot \prod_{i=1}^{17} EM_i \quad (1)$$

where  $B = 1.01 + \sum_{j=1}^5 SF_j$

The other set of parameters in Equation 4 influence the model exponentially and they are represented by values called Scale Factors (SF). The selection of scale factors is based on the rationale that they are a significant source of exponential variation on a project's effort. There are five scale factors in COCOMO II and they are listed in Table 3 along with their rating scales at the end of this paper. The parameter in COCOMO II for process maturity is labeled PMAT and it is a scale factor in the COCOMO II model, i.e. it influences effort exponentially.

Because PMAT is one of the scale factors that comprise B, it is important to understand the influence of B in the COCOMO II model. If  $B > 1.0$ ,

the project exhibits what is called diseconomies of scale. For instance when the product's size is doubled the project effort is more than doubled. This is generally due to two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead. Larger projects will have more personnel and thus more interpersonal communications paths consuming overhead. Integrating a small product as part of a larger product requires not only the effort to develop the small product, but also the additional overhead effort to design, maintain, integrate, and test its interfaces with the remainder of the product.

If  $B = 1.0$ , the economies and diseconomies of scale are in balance. This linear model is often used for cost estimation of small projects.

In the COCOMO II model the Process Maturity factor (PMAT) is determined using one of two methods. The first method selects an overall maturity level based either on an organized evaluation or subjective judgment, Figure 2. 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. These two Level 1 ratings are a departure from the published definition of the SW-CMM. From the large number of organizations assessed at SW-CMM Level 1, see Figure 1, it is important to distinguish the groups working their way to a Level 2 rating. The remaining levels follow the SW-CMM discussed earlier.

CMM Level 1 (lower half)
CMM Level 1 (upper half)
CMM Level 2
CMM Level 3
CMM Level 4
CMM Level 5

**Figure 2. Overall Maturity Level**

The second method selects a rating for Process Maturity as a percentage of compliance for each set of KPA goals. The KPA data is collected at the project level. This level of information is desired so that the effects of Process Maturity can be assessed at the project level. A summary of one KPA and its COCOMO II rating scale is given in Figure 3.

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

**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 3. KPA Data Example**

- **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.

PMAT is computed as the average of all n rated KPA's 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% for Almost Always, 75% for Frequently, 50% for About Half, 25% for Occasionally, 1% for Rarely if Ever). A PMAT percentage is calculated as in Equation 2.

$$PMAT\% = \left( \sum_{i=1}^n \frac{KPA\%_i}{100} \right) \cdot \frac{1}{n} \quad (2)$$

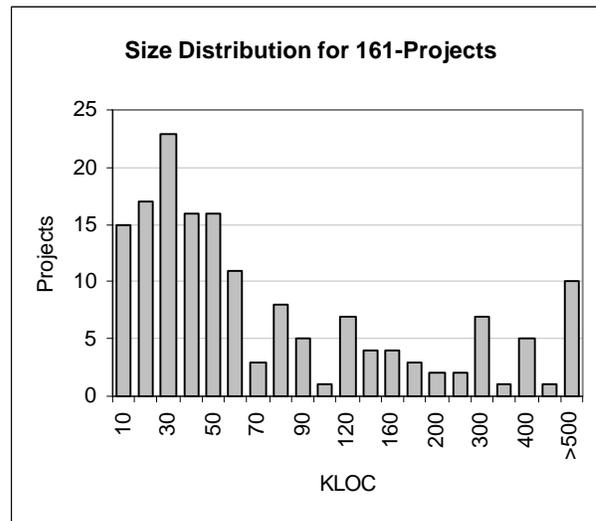
To convert the PMAT percentage, an ordinal value, to a quantified value for use in the model a simple monotonic sequence of numbers is assigned to the rating levels, see row I in Figure 7 (these values are then calibrated to project data as discussed later). The percentage is used from the Very Low to Extra High rating values to select an interpolated quantified

value. The sequence of numbers assigned to the PMAT ratings is decreasing from Very Low to Extra High. The rating values decrease to test the hypothesis stated earlier that as higher levels of Process Maturity are attained (moving towards the Extra High rating) the software development diseconomies of scale, and thus effort, should decrease.

## Data Collection and Analysis

There are project observations used in this analysis from eighteen sources. These sources covered the Aerospace Industry, Federally Funded Research Centers, Commercial Industry, and Department of Defense supported Industry. The data was on past, completed projects. Much of the data is proprietary and furnished to the University of Southern California under nondisclosure agreements.

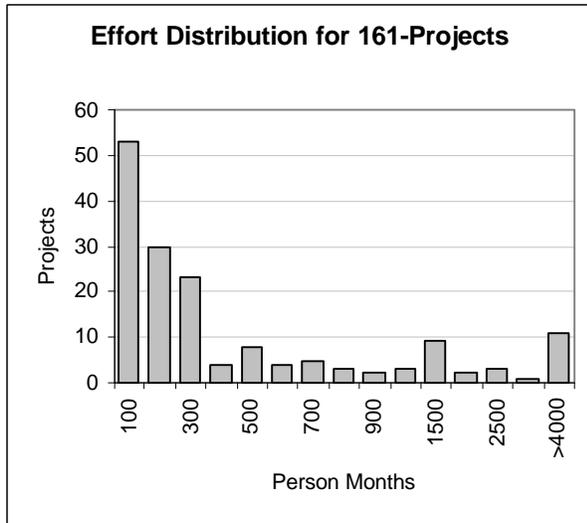
Most of the data came from 1990's projects, although some projects from the 1970's and 1980's are included. Product sizes range from 2.6 to 1,264.0 thousands of lines of code (KLOC). The KLOC size data has an average of 131 and a median of 47, Figure 4.



**Figure 4. KLOC Distribution**

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 5.

The effort data used to calibrate COCOMO II largely comes from individual time reporting, which is generally accurate to no better than 15%. 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

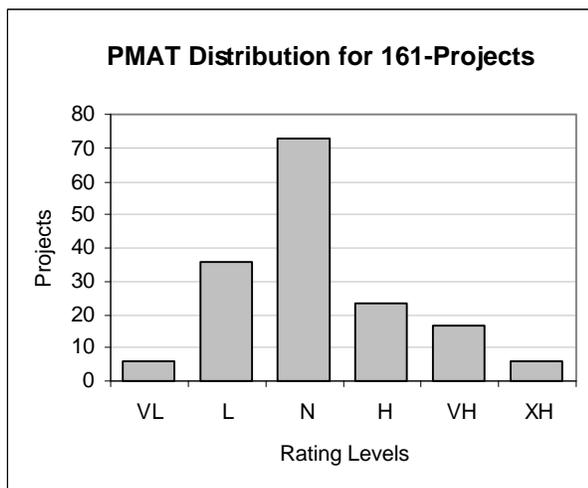


**Figure 5. PM Distribution**

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 6. 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 data.

For the results presented here, development effort is measured in Person Months. A person month



**Figure 6. PMAT Distribution**

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

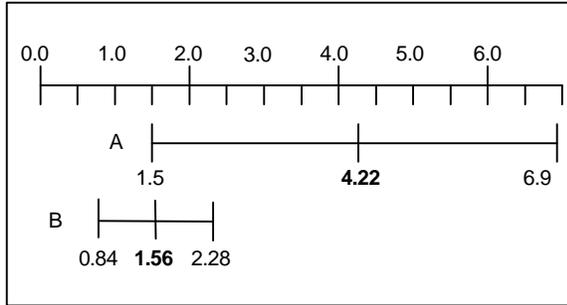
There are two methods of rating a project's process maturity. This was discussed earlier. While an organization may be rated at a specific SW-CMM level, the respondents were encouraged to use the second method, i.e. answer all of the KPA questions considering *what actually happened on the project*.

The results of two data samples are presented. The first sample is data from 112 projects that was collected by 1997. This will be referred to as Sample A. Research was done on this sample to initially demonstrate the influence of process maturity on effort [13]. Partial results from that research are presented here. The second sample is data from 161 projects collected by 1998. This sample includes the 112 projects from Sample A. This will be referred to as Sample B. Sample A was used to calibrate COCOMO II using multiple regression analysis. Sample B was used to calibrate COCOMO II using both multiple regression analysis and a more sophisticated technique called Bayesian regression analysis [14].

The calibration of the COCOMO II model to both sets of project data showed that PMAT is statistically significant in supporting the hypothesis that increasing process maturity decreases the effort required to develop a software product. The calibration results from both datasets produced estimated PMAT coefficients shown in Figure 7 with their 95% confidence limits. Bar A represents Sample A and bar B represents Sample B. Due to noise in the data the actual coefficient for PMAT could appear within the ranges of 1.5 to 6.9 for Sample A and 0.84 to 2.28 for Sample B.

The fact that, for both Samples A and B, the 95% confidence regions do not include zero or negative values, is the most important single conclusion in the paper. It says that, for the 112 and 161 representative projects in the analysis:

*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.*



**Figure 7. PMAT Coefficient Ranges**

**Magnitude of PMAT Effort Reduction**

The PMAT variable appears in COCOMO II as a scale factor in the exponent that relates size to effort. Thus, its effect on effort reduction will be larger for larger-size projects. The magnitude of the PMAT effect is obtained by applying the coefficients resulting from the multiple regression analyses of Samples A and B to the initial PMAT rating values, shown in Figure 8 row I. When the Sample A coefficient (4.22) is multiplied to the initial ratings, the range of PMAT values is from 0.0 to 0.211, shown in Figure 8 row A. When the Sample B coefficient (1.56) is multiplied to the initial ratings, the range of PMAT values is from 0.0 to 0.078, shown in Figure 8 row B.

	Lvl 1 Lower	Lvl 1 Upper	Lvl 2	Lvl 3	Lvl 4	Lvl 5
	Very Low	Low	Nominal	High	Very High	Extra High
I	0.0500	0.0400	0.0300	0.0200	0.0100	0.0000
A	0.2110	0.1688	0.1266	0.0844	0.0422	0.0000
B	0.0780	0.0624	0.0468	0.0312	0.0156	0.0000

**Figure 8. PMAT Scale Values**

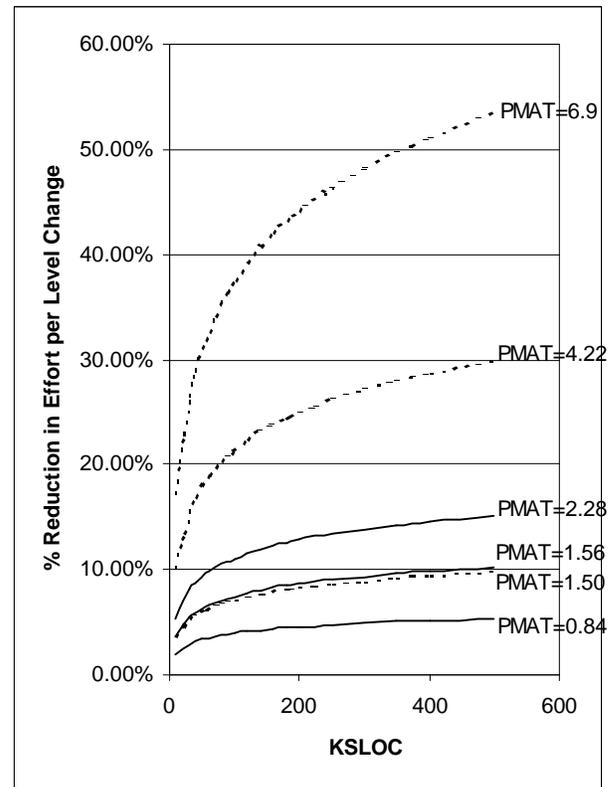
Using the PMAT ratings derived from Sample B, we can examine the effect on effort. Recall that PMAT's rating can be selected one of two ways: either overall maturity level or KPA-based percentage. From Figure 8 row B we observe that selecting Level 1 Upper yields a PMAT rating value of 0.0624 and selecting Level 2 yields a PMAT rating value of 0.0468. It can be observed that a change between any two consecutive levels is a difference in PMAT rating values of 0.0156.

If a one level change, e.g. from Low to Nominal, is applied to a range of project sizes we can see the reductions in effort. The PMAT coefficients and their two confidence limits from Figure 7 bars A and B are applied to the initial PMAT rating values, row I in Figure 8. A one level change is applied to a range

of sizes from 2,000 to 500,000 Lines of Code. Using Sample B analysis shows that, at a 95% confidence level, process maturity reduces effort required to develop software from between 3% to 15%, see the solid lines in Figure 9. The average is between 4% and 11% reduction. Based on the average, the effect of going from Level 1 Lower to Level 5 is an overall reduction in effort of between 15% to 75%.

Compare this to the results obtained from Sample A and used in earlier research [13]. At a 95% confidence level process maturity reduces effort from between 5% to 54%, see the dotted lines in Figure 9. The average is between 10% and 30% reduction. Based on the Sample A average, the effect of going from Level 1 Lower to Level 5 is an overall reduction in effort of between 50% to 160%. The reduction of effort by 160% is unreasonable. However, the lower end of this range, 50%, appears reasonable and it overlaps the upper end of the range determined from Sample B, see Figure 9.

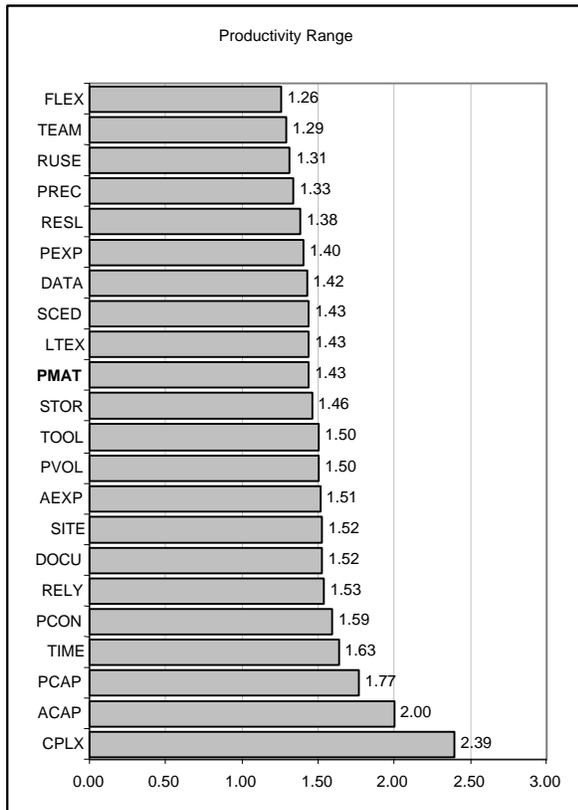
In addition to analyzing PMAT's influence on effort, the strength of PMAT among other factors influencing effort can be observed. Using Sample B, the productivity range (the total percentage a factor can influence effort going from the lowest to the highest rating) of all COCOMO II factors is shown in Figure 10. The productivity range for the scale factors is based on a 100 KLOC project. PMAT is



**Figure 9. % Reduction in Effort per PMAT Level**

the most influential of the scale factors. Its productivity range would increase from 1.43 to 1.50 for a 500 KLOC project. Some of the factors that are stronger than PMAT are Tool Usage (TOOL), Application Experience (AEXP), Required Reliability (RELY), Analyst and Programmer Capability (ACAP, PCAP), and Product Complexity (CPLX).

Applied over a range of sizes it can be seen that



**Figure 10. PMAT Comparison to Other Factors**

PMAT's effect on effort, using Sample B data, is higher for large projects than on small projects. Projects in the size range of 10 to 50 KSLOC showed a reduction in effort of only 4% to 6% for a one level change in rating. Projects in the size range of 500 to 1000 KSLOC show corresponding reductions of 10% to 11%.

## 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 in Sample B, Software Process Maturity was a significant factor (95% confidence level) in reducing software development effort, even after normalizing for the effects of other effort influencing factors. For projects ranging in size from

2 to 1000 KSLOC, a one-increment change in the rating of Process Maturity resulted in a 4% to 11% reduction in effort.

The analysis performed and 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.

## References

- [1] W.S. Humphrey, Managing the Software Process, Addison-Wesley, Reading, Ma. 1989.
- [2] 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.
- [3] 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.
- [4] K. Butler, "The Economic Benefits of Software Process Improvement," Crosstalk, Hill AFB, Ogden, Ut., 1995, pp. 14-17.
- [5] R. Dion, "Process Improvement and the Corporate Balance Sheet," IEEE Software, October 1993, pp. 28-35.
- [6] 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.

- [7] W.S. Humphrey, T.R. Snyder, and RR. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, August 1991, pp. 11-23.
- [8] 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.
- [9] 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.
- [10] H. Wohlwend and S. Rosenbaum, "Schlumberger's Software Improvement Program," IEEE Transactions on Software Engineering, November 1994, pp. 833-839.
- [11] W.C. Peterson, "SEI's Software Process Program," Presentation to the Board of Visitors, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., April 1997.
- [12] B.W. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering, J.D. Arthur and S.M. Henry (Eds.), J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, 1995, pp. 57-94.
- [13] B.K. Clark, "The Effects of Software Process Maturity on Software Development Effort," Ph.D. Dissertation, University of Southern California, Los Angeles, Ca. 90089-0781, August 1987 (available at <http://sunset.usc.edu/~bkclark/Research>).
- [14] S. Chulani, B. Boehm, and B. Steece, "Bayesian Analysis of Empirical Software Engineering Cost Models," Center for Software Engineering Technical Report, USC-CSE-99-513, University of Southern California, Los Angeles, Ca. 90089-0781. (accepted for IEEE-TSE; Special Issue on Empirical Methods).

Multiplicative Factors	Very Low	Low	Nominal	High	Very High	Extra High
Required Reliability (RELY)	slight inconvenience	low, easily recoverable losses	Moderate, easily recoverable losses	high financial loss	risk to human life	
Database Size (DATA)		DB bytes/ Pgm SLOC < 10	10 (D/P < 100)	100 (D/P < 1000)	D/P (1000)	
Complexity (CPLX)	There is another table that establishes this rating based on 5 criteria. [12]					
Developed for Reuse (RUSE)		none	Across project	across pro gram	across product line	across multiple product lines
Documentation Match to Lifecycle Needs (DOCU)	Many life- cycle needs uncovered	Some life- cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life- cycle needs	Very excessive for life- cycle needs	
Time Constraint (TIME)			50%	70%	85%	95%
Storage Constraint (STOR)			50%	70%	85%	95%
Platform Volatility (PVOL)		Major changes every 12 mo.; Minor changes every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.; Minor: 1 wk.	Major: 2 wk.; Minor: 2 days	
Analyst Capability (ACAP)	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Programmer Capability (PCAP)	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Personnel Continuity (PCON)	48% / year	24% / year	12% / year	6% / year	3% / year	
Application Experience (AEXP)	2 months	6 months	1 year	3 years	6 years	
Platform Experience (PEXP)	2 months	6 months	1 year	3 years	6 year	
Lang. & Tool Experience (LEXT)	2 months	6 months	1 year	3 years	6 year	
Tool Usage (TOOL)	edit, code, debug	simple, front-end, back end CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, re use	
Multi-site Development: Collocation (SITE)	International	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
Multi-site Development: Communications (SITE)	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication.	Wideband elect. comm, occasional video conf.	Interactive multimedia
Schedule Constraint (SCED)	75% of nominal	85%	100%	130%	160%	

**Table 2. COCOMO II Effort Multipliers (EM)**

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness (PREC)	thoroughly unprecedented	largely unprecedented	Somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
Architecture and Risk Resolution (RESL)	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Development Flexibility (FLEX)	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Team Cohesion (TEAM)	very difficult interactions	some difficult interactions	Basically co operative interactions	largely cooperative interactions	highly cooperative	seamless interactions
Process Maturity (PMAT)	CMM Level 1 (Lower)	CMM Level 1 (Upper)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5

Or an average of KPA Goal compliance, see Figure 3.

**Table 3. COCOMO II Scale Factors (SF)**