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 30 percent. Using a 161 project sample a change in one level of process maturity resulted in a reduction of development effort from 4 to 10 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 specification, design, source code, and testing. These different artifacts interact with each other where a delay or defect in one affects the completeness 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 government organization misuses taxpayers' money either through budget overruns or decreased user and customer satisfaction. Controling and improving the process used to develop software is seen as the 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 KPA compliance from Level 2 and from Level 3.

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 demonstrated KPA compliance capability in 6 KPA's, see Table 1. 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 successful projects on 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

<table>
<thead>
<tr>
<th>Level</th>
<th>Key Process Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>None</td>
</tr>
<tr>
<td>Level 2</td>
<td>Repeatable</td>
</tr>
<tr>
<td></td>
<td>Requirements Management</td>
</tr>
<tr>
<td></td>
<td>Software Project Planning</td>
</tr>
<tr>
<td></td>
<td>Software Project Tracking and Oversight</td>
</tr>
<tr>
<td></td>
<td>Software Subcontract Management</td>
</tr>
<tr>
<td></td>
<td>Software Quality Assurance</td>
</tr>
<tr>
<td></td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td></td>
<td>Configuration Process Definition</td>
</tr>
<tr>
<td></td>
<td>Configuration Process Planning</td>
</tr>
<tr>
<td></td>
<td>Training Program</td>
</tr>
<tr>
<td></td>
<td>Integrated Software Management</td>
</tr>
<tr>
<td></td>
<td>Software Product Engineering</td>
</tr>
<tr>
<td></td>
<td>Intergroup Coordination</td>
</tr>
<tr>
<td></td>
<td>Peer Review</td>
</tr>
<tr>
<td>Level 3</td>
<td>Defined</td>
</tr>
<tr>
<td></td>
<td>Requirements Management</td>
</tr>
<tr>
<td></td>
<td>Software Project Planning</td>
</tr>
<tr>
<td></td>
<td>Software Project Tracking and Oversight</td>
</tr>
<tr>
<td></td>
<td>Software Subcontract Management</td>
</tr>
<tr>
<td></td>
<td>Software Quality Assurance</td>
</tr>
<tr>
<td></td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td></td>
<td>Configuration Process Definition</td>
</tr>
<tr>
<td></td>
<td>Configuration Process Planning</td>
</tr>
<tr>
<td></td>
<td>Training Program</td>
</tr>
<tr>
<td></td>
<td>Integrated Software Management</td>
</tr>
<tr>
<td></td>
<td>Software Product Engineering</td>
</tr>
<tr>
<td></td>
<td>Intergroup Coordination</td>
</tr>
<tr>
<td></td>
<td>Peer Review</td>
</tr>
<tr>
<td>Level 4</td>
<td>Managed</td>
</tr>
<tr>
<td></td>
<td>Quantitative Process Management</td>
</tr>
<tr>
<td></td>
<td>Software Quality Management</td>
</tr>
<tr>
<td>Level 5</td>
<td>Optimizing</td>
</tr>
<tr>
<td></td>
<td>Technology Change Management</td>
</tr>
<tr>
<td></td>
<td>Process Change Management</td>
</tr>
</tbody>
</table>

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 has added 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 149 key practices. All of the KPA's are described in [2].

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 long-term benefits of high process maturity are software delivered on time, within budget, within customer requirements, and of high quality. Another important benefit would be the effect it has on productivity.

Much has been written discussing the short-term and long-term benefits of increasing maturity levels. 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 assessing 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 affects. 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

Copyright 1998 USC-CSE
Process Maturity on effort and shows the quantified relationship between process maturity and other factors.

**COCOMO II Model**

The COCOMO II model predicts effort (Person Months) and schedule required to complete a software development project [12]. 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 lists the effort multipliers and their associated rating scales.

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

(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. The parameter in COCOMO II for process maturity is labeled PMAT and it is a scale factor in the COCOMO II model.

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 product effort is more than double. 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 PMAT is determined using one of two methods. The first method was by selecting 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 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.

![Figure 2. Overall Maturity Level](image)

The second way of rating Process Maturity is to rate the 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. An example of one KPA is given in Figure 3.

There are eighteen KPAs and each has seven ratings. The ratings 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). This rating was given a weight of 75.
- **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). This rating was given a weight of 25.
- **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 3. KPA Data Example**

PMAT is computed as the average of all rated KPAs for a single project (Does Not Apply and Don’t Know are not counted which sometimes makes n less than 18). After the compliance of each KPA is determined each compliance level is weighted (100 for Almost Always, 75 for Frequently, 50 for About Half, 25 for Occasionally, 1 for Rarely if Ever) and a PMAT percentage is calculated as in Equation 2.

\[
PMAT\% = \left( \sum_{i=1}^{n} \frac{KPA\%}{100} \right) \times \frac{1}{n}
\]

To convert the PMAT percentage, an ordinal value, to a quantified value for use in the model a monotonic sequence of numbers are assigned to each rating, see Figure 7 (these values are established from analysis results discussed later). The percentage is used from the Very Low to Extra High ratings to select the associated quantified value. The sequence of numbers assigned to the PMAT ratings should be decreasing from Very Low to Extra High. The rating values decrease to support the hypothesis stated earlier that as higher levels of Process Maturity are attained (moving towards the Extra High rating) the software development 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 158 and a median of 53, Figure 4.

**Figure 4. KSLOC Distribution**

Project effort ranges from 6 to 11,400 Person Months. PM data has an average of 830 Person Months and a median of 180 Person Months, Figure 5.
There was no data collected on whether the software technology used on the project matched the application complexity. There was no data collected on whether the processes used to develop the software were mismatched to the type of application domain. 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.

While the data sources varied there was selection bias in the data. We were not given data on unfinished or unsuccessful projects nor did any unsuccessful companies contribute data. The data was from successful projects from successful companies. Proof of this is in the fact that these companies were mature enough to practice collecting data and that the project had to finish in order to provide completed data.

A data collection form is used to collect data from projects for calibration of the COCOMO II model. 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.

For Process Maturity respondents had 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. One is a sample of 112 projects from data collected by 1997. 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. This dataset will be called Sample A. The second sample is 161 projects from data collected by 1998. This will be called Sample B. Sample A was used to calibrate COCOMO II using multiple regression analysis. Sample B was used to calibrate COCOMO II using a more sophisticated technique called Bayesian regression analysis. Process maturity levels in both data sets covered the full range. The proportion of Levels 3, 4, and 5 projects is higher than the community-wide distributions shown in Figure 1. This is due to 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.

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 the estimated PMAT coefficients with their 95% confidence limits shown Figure 6. 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 1.15 to 1.97 for Sample B.

These coefficients are used to calibrate the COCOMO II PMAT ratings. The initial PMAT rating values are shown in Figure 7 Bar I. When the Sample A coefficient (4.22) is multiplied, the range of PMAT values is from 0.0 to 0.211, shown in Figure 7 Bar A. When the Sample B coefficient (1.56) is multiplied, the range of PMAT values is from 0.0 to 0.078, shown in Figure 7 Bar B.

Using the PMAT ratings derived from calibration with data from Sample B, we can examine the effect on effort. Recall the PMAT's rating can be selected one of two ways: either overall maturity level or KPA-based percentage. From Figure 7 we observe that selecting Level 1 Upper yields a PMAT rating value of 0.0624 and selecting Level 3 yields a PMAT rating value of 0.0312. It can be observed...
Figure 7. PMAT Scale Values

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 Nominal to High,
is applied to a range of project sizes we can see the
reductions in effort. The PMAT coefficient and the
two confidence limits from Figure 6 Bar B are
applied to the initial PMAT rating values. A one
level change is applied to a range of sizes from 2,000
to 1,000,000 Lines of Code. Analysis shows that at a
95% confidence level, process maturity reduces effort
required to develop software from between 3% to
15%, shown in Figure 8. The mean is between 4% and 11%
reduction.

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]. The
effect of a one level change using the Sample A
derived coefficient over the same projects sizes is a
reduction in effort between 10% to 32%. This result
is an overall reduction of between 50% to 160%.

The lower end of this range appears reasonable and it
overlaps the upper end of the range determined from
Sample B, see Figure 8.

In addition to analyzing PMAT's influence on
effect, the strength of PMAT among other factors
influencing effort can be observed. Figure 9 shows the
effect of the COCOMO II scale factors varying
with size (see Tables 3 and 4 for acronym
explanation). The multiplicative factors are not
affected by size. The PMAT values used in Figure 9
are based on the 1.56 calibration coefficient. Factors
that are stronger than PMAT are Schedule Constraint,
Time Constraint, Complexity, Analyst Capability,
Programmer Capability, and Storage Constraint.

Applied over a range of sizes it can be seen that
PMAT's effect on effort 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. For those
size projects, all of the multiplicative factors are more
influential than PMAT. Yet PMAT is the most
influential of the scale factors.

Figure 8. % Reduction in Effort per PMAT Level

Figure 9. Average Change in Effort Per Rating
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) affecting software development effort. After normalizing for the effects of other effort influences, a one-increment change in the rating of Process Maturity resulted in a 4% to 15% 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. This factor provides a significant assessment of the effects of process on development effort.

Future work needed in this analysis area is collection more KPA data. This could be used to assess which KPAs have the most influence on effort. Implementing the effort saving KPAs first would offset the costs of implementing the other WAS. Based on the KPA results, the model could be refined to capture any nonuniform improvements in going from CMM Level n to Level (n+1).

References


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

<table>
<thead>
<tr>
<th>Scale Factors</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedentedness (PREC)</td>
<td>thoroughly Precedented</td>
<td>largely Precedented</td>
<td>usually Precedented</td>
<td>generally familiar</td>
<td>largely familiar</td>
<td>thoroughly familiar</td>
</tr>
<tr>
<td>Architecture and Risk Resolution (PEAS)</td>
<td>High (20%)</td>
<td>Good (40%)</td>
<td>Average (60%)</td>
<td>generally (75%)</td>
<td>mostly (90%)</td>
<td>full (100%)</td>
</tr>
<tr>
<td>Development Flexibility (PLEX)</td>
<td>rigorous</td>
<td>occasional</td>
<td>relaxation</td>
<td>general</td>
<td>conformity</td>
<td>some conformity</td>
</tr>
<tr>
<td>Team Cohesion (TEAM)</td>
<td>very difficult interactions</td>
<td>some difficult interactions</td>
<td>basic co-operative interactions</td>
<td>largely co-operative interactions</td>
<td>highly co-operative</td>
<td>seamless interactions</td>
</tr>
<tr>
<td>Process Maturity (PMA)</td>
<td>CMM Lvl 1 (Lower)</td>
<td>CMM Lvl 1 (Upper)</td>
<td>CMM Lvl 2</td>
<td>CMM Lvl 3</td>
<td>CMM Lvl 4</td>
<td>CMM Lvl 5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Scale Factors</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedentedness</td>
<td>thoroughly</td>
<td>largely</td>
<td>usually</td>
<td>generally familiar</td>
<td>largely familiar</td>
<td>thoroughly familiar</td>
</tr>
<tr>
<td>Architecture and Risk Resolution</td>
<td>High (20%)</td>
<td>Good (40%)</td>
<td>Average (60%)</td>
<td>generally (75%)</td>
<td>mostly (90%)</td>
<td>full (100%)</td>
</tr>
<tr>
<td>Development Flexibility</td>
<td>rigorous</td>
<td>occasional</td>
<td>relaxation</td>
<td>general</td>
<td>conformity</td>
<td>some conformity</td>
</tr>
<tr>
<td>Team Cohesion</td>
<td>very difficult interactions</td>
<td>some difficult interactions</td>
<td>basic co-operative interactions</td>
<td>largely co-operative interactions</td>
<td>highly co-operative</td>
<td>seamless interactions</td>
</tr>
<tr>
<td>Process Maturity</td>
<td>CMM Lvl 1 (Lower)</td>
<td>CMM Lvl 1 (Upper)</td>
<td>CMM Lvl 2</td>
<td>CMM Lvl 3</td>
<td>CMM Lvl 4</td>
<td>CMM Lvl 5</td>
</tr>
</tbody>
</table>

Copyright 1998 USC-CSE