When organizations make many improvements concurrently, software project managers have no way of determining how much improvement is due to process maturity versus other factors. Using a 161-project sample, this article isolates the effects on effort of process maturity versus other effects, concluding that an increase of one process maturity level can reduce development effort by 4% to 11%.

According to the Software Capability Maturity Model (SW-CMM), the primary intended long-term benefit of high process maturity is high-quality software meeting customer requirements, delivered on time and within budget. Productivity improvement is another important potential benefit. Because changing the software development process in an organization requires a large investment, many software organizations remain at low levels of maturity.

The effects of increasing process maturity alone are not easy to determine, because organizations generally make concurrent improvements in other areas that also result in benefits to the development organization. So, a clearer assessment of process maturity effects is needed. Anecdotal case study evidence shows many short- and long-term benefits to improving process maturity. These studies used different assessment approaches, none of which try to isolate individual factors affecting productivity. Nevertheless, they do indicate that increasing maturity levels generally have positive effects.

The premise of this article 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 of increased process maturity in the context of other influences on software development effort. I have used a mathematical model to segregate process maturity’s influence on effort from that of other factors. The model quantifies the magnitude of this influence and the relationship between process maturity and the other factors. The results here are based on previous research that Barry Boehm and I conducted.

Data Collection

Four areas generally influence software development effort: product factors, project factors, platform factors, and personnel fac-
The data we collected on product factors included size, precedentedness, architecture and risk resolution, required reliability, database size, complexity, whether the product was developed for reuse, and documentation match to life-cycle needs. Data collected on platform factors included time constraints, storage constraints, and development platform volatility. Data collected on personnel factors included analyst and programmer capability, personnel continuity, team application experience, and language and tool experience. Data collected on project factors included development flexibility, team cohesion, tool usage, multisite development, schedule compression, and—the factor of interest here—process maturity. The Cocomo II model describes all the factors in detail.11

I collected data for the Process Maturity (PMAT) factor using two methods. The first selects an overall maturity level based on either an organized evaluation or a subjective judgment (see the middle column in Table 1). The SW-CMM Level 1 lower half is for organizations that rely on “heroes” to do the job. They don’t focus on processes or documenting lessons learned. The SW-CMM Level 1 upper half is for organizations that have implemented most of the Key Process Areas that would satisfy SW-CMM Level 2. It is important to distinguish the groups working their way to a Level 2 rating. A transition from Level 1 lower to Level 1 upper is modeled as a change in a PMAT level. These two Level-1 ratings differ from the SW-CMM’s published definition; the remaining levels follow the SW-CMM.

The second method selects a rating, called the Equivalent Process Maturity Level, based on the percentage of compliance for each Key Process Area goal.1 If the project has undergone a recent CMM assessment, the EPML method uses the percentage compliance for the overall KPA (based on KPA compliance assessment data). If an assessment had not been done, I interviewed someone on the project or familiar with the project. My questions were directed at determining how close a project’s processes met the goals for each KPA. I recorded the levels of compliance using the Likert scale. Figure 1 shows that scale along with one KPA.

Although an organization might be rated at a specific SW-CMM level, the respondents were encouraged to use the second method—that is, to answer all KPA questions considering what actually happened on the project.

Each of the 18 KPAs has seven rating levels:

- **Almost Always**: The goals are consistently achieved and are well established in standard operating procedures (over 90% of the time).
- **Frequently**: The goals are achieved relatively often but sometimes omitted under difficult circumstances (about 60% to 90% of the time).
- **About Half**: The goals are achieved about half the time (about 40% to 60% of the time).
- **Occasionally**: The goals are achieved less often (about 10% to 40% of the time).
- **Rarely If Ever**: The goals are rarely if ever achieved (less than 10% of the time).
- **Does Not Apply**: The respondent has the required knowledge about the proj-

### Table 1

<table>
<thead>
<tr>
<th>Process maturity rating (PMAT)</th>
<th>Overall process maturity level</th>
<th>Equivalent process maturity level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>CMM Level 1 lower half</td>
<td>0</td>
</tr>
<tr>
<td>Low</td>
<td>CMM Level 1 upper half</td>
<td>1</td>
</tr>
<tr>
<td>Nominal</td>
<td>CMM Level 2</td>
<td>2</td>
</tr>
<tr>
<td>High</td>
<td>CMM Level 3</td>
<td>3</td>
</tr>
<tr>
<td>Very high</td>
<td>CMM Level 4</td>
<td>4</td>
</tr>
<tr>
<td>Extra high</td>
<td>CMM Level 5</td>
<td>5</td>
</tr>
</tbody>
</table>
ect or organization and the KPA but feels that the KPA does not apply to his or her circumstances.

- **Do Not Know**: The respondent is uncertain about how to respond.

I computed an EPML as five times the average compliance level of all n-rated KPAs for a single project (Does Not Apply and Do Not Know are not counted, which sometimes makes n less than 18). The compliance level for each KPA is represented by a weight assigned to each level of the Likert scale (100 for Almost Always, 75 for Frequent, 50 for About Half, 25 for Occasionally, 1 for Rarely If Ever). I calculated the EPML as

\[
\text{EPML} = 5 \times \left( \sum_{i=1}^{n} \frac{\text{KPA weight}}{100} \right) \frac{1}{n}
\]

To convert an EPML to a PMAT level, I used Table 1. An EPML of 0 corresponds with a PMAT level of very low. (I discuss this correspondence in more detail later.)

The data used in this analysis is from 161 projects from 18 sources. These sources include the aerospace industry, federally funded research centers, commercial industry, and industry supported by the US Department of Defense. The data was on past, completed projects. Of the 161 projects analyzed, 65% came from projects undertaken in the 1990s. Earlier projects were assessed using SW-CMM levels based on interviews with people familiar with the project. Much of the data is proprietary and was furnished under nondisclosure agreements to the University of Southern California, where I conducted this research.

The data included these statistics:

- Product sizes ranged from 2.6 to 1,264.0 KLOC (thousands of lines of code). The average number of KLOC was 131, and the median was 47 (see Figure 2).
- Project effort ranged from 6 to 11,400 person-months (a person-month is 152 hours). The average number of person-months was 711, and the median was 195 (see Figure 3).

The effort data comes largely from individual time reporting, which is generally accurate to within 15%. Similar variations occur in reported size data across different organizations. Although I requested uncompensated overtime, I did not consistently collect it. These and other factors, such as the interpretation of qualitative ratings, mean that the data is imprecise. Thus, I present the results with a confidence interval.

Even though the data sources varied, the data has selection bias. I did not receive data on unfinished or unsuccessful projects, and no unsuccessful companies contributed data. The data is from successful projects from successful companies: the organizations are mature enough to practice collecting data, and all of the project data is from completed projects. Process maturity level data covered the full range, as Figure 4 shows. High maturity level data was available due to the selection bias mentioned earlier and to the emphasis these organizations placed on data collection and analysis. A total of 31% of the projects provided KPA level data.

The results presented here measure develop-
operation effort in person-months. It includes the software developer’s time, project management time, administrative support time, and project support personnel time (for example, 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).

**Effect on Effort**

The results of data analysis from the 161 projects showed that PMAT is statistically significant, supporting the hypothesis that increasing process maturity decreases the effort to develop a software product. Using Bayesian regression analysis, a full calibration of the Cocomo II model revealed this effect. Equation 2 shows a simplified form of the Cocomo II model used in the analysis:

\[ \text{Person-months} = A \times (\text{Size})^B \times C \]  

(2)

The model’s calibration with the 161 project observations showed that PMAT was stronger at explaining variation in actual project data effort when it was used as an exponent to the project size, the \( B \) in Equation 2.\(^{10} \) As the project’s size increased, the process maturity level affected development effort more.

The bottom row of values in Figure 5 shows PMAT levels converted to numbers for use in Equation 2. The sequence of values assigned to the PMAT rating levels decreases, as the EPML goes from very low to extra high. This tests the premise stated earlier that, as higher PMAT levels are attained, the software development effort decreases.

The analysis results varied for reasons discussed earlier. Figure 6 shows a one-level change in PMAT across different size projects with a green line. The red and blue lines show variation in the results at the 95% confidence level. This figure shows that

\[ \text{with 95% confidence, even after normalizing for the effects of other cost drivers, an increase in process maturity corresponds with a reduction in project effort.} \]

Other research using a different approach recently confirmed these results.\(^{12} \) A one-level change in PMAT—for example, from nominal to high—was applied to a range of project sizes to show the percentage reduction in effort (see Figure 6). At a 95% confidence level, process maturity reduced effort by 3% to 15%, as the blue and red lines in Figure 6 show. The average reduc-
tion, shown by the green line, for 2- to 1,000-KLOC projects was between 4% and 11%.

In addition to analyzing PMAT’s influence on effort, these results showed the relative strength of PMAT among other factors influencing effort. Figure 7 shows the productivity range (the total percentage a factor can influence effort, going from the lowest to the highest factor rating) for all factors. The productivity range for the factors that are used with size (B in Equation 2), of which PMAT is a part, is based on a 100-KLOC project. For this project, PMAT’s productivity range is 1.43. This means it can change the effort to develop a software product up to 43%, from the lowest PMAT level to the highest. PMAT’s productivity range would be 1.62 for a 500-KLOC project and 1.71 for a 1,000-KLOC project. For a typical project of 10 KLOC, the PMAT productivity range would be 1.20.

Depending on product size, PMAT’s influence on effort is less than that of some other factors. This suggests that, if productivity improvement is an organization goal, process maturity should be considered as one of multiple strategies to achieve that goal. In addition to PMAT, the other strategies would include improving “management-controllable” factors such as analyst capability, programmer capability, and personnel continuity. This analysis supports the conclusion that increasing an organization’s software development process maturity will not hurt productivity.

The analysis presented here is a generalization across all levels of process maturity. The percentage reduction in effort is not uniform among levels. More data on process maturity collected at the KPA level would permit quantification of the change between each level. Future work in this analysis area requires collecting more KPA data to assess which KPAs influence effort the most. Implementing the effort-saving KPAs first would offset the costs of implementing other KPAs. 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 is insufficient to establish such assessments with statistical significance.

**Acknowledgments**

I am grateful for the support of the Center for Software Engineering at the University of Southern California, where I conducted this research.
References


About the Author

Bradford Clark is a technical consultant with Software Metrics, Inc. He consults, trains, and provides analysis in software product and process measurement. While attending graduate school, he conducted research on the effects of software process maturity on software development effort. Brad was a Navy Civil Servant for 11 years at China Lake, California, and before that he flew A-6 aircraft off and on small aircraft carrier decks around the world. He currently works with the Cocomo II Project Team at the Center for Software Engineering at the University of Southern California and he is a visiting scientist at the Software Engineering Institute. Brad received his PhD in computer science from the University of Southern California. He is a member of the IEEE Computer Society and the ACM. Contact him at Brad@Software-Metrics.com.