

## **Leveraging a Software-Estimating Model to Ease the Climb up the CMM Ladder**

**Arlene F. Minkiewicz**

I am frequently asked how a commercial software model will add value to an organization's attempts to increase software process maturity. I like this kind of question because there is a pretty simple answer up front but there's also a lot of details that make for a rich discussion about software process improvement and software cost models. The simple answer is that project management is a very important part of any solid process improvement effort and being able to perform accurate size, cost and schedule estimates is vital to good project management.

The more interesting answer requires some digression into the complicated world of process improvement. I like to frame this discussion in the context of Carnegie Mellon University's Software Engineering Institute (SEI) Capability Maturity Model (CMM) for Software. I do this not because this is the only vehicle for software process improvement, but because this is an effort that encapsulates years of research into the software related processes that must be in place and properly understood and executed in order to have a world class software development organization. I will talk briefly about software process maturity and the CMM – what it is and why it is beneficial. This will be followed by a discussion of software-estimating tools – what they do and how they work. Then I will delve into some of the specific process areas where software estimating model will help your organization achieve process maturity successes.

### **Software Process Improvement and the CMM**

In the late 80's, the federal government made the determination that their job of building software systems would be simplified if they could perform quantitative evaluations of the capability of the subcontractors competing to build these systems. They shared this realization, along with some funding, with Carnegie Mellon University's Software Engineering Institute (SEI) and began the project that led to the development of what we call today the SEI Capability Maturity Model (CMM) for Software. Since its introduction in the early 90's, hundreds of software development organizations have used the CMM not only to assess the maturity of their existing processes but as a framework to guide their climb to higher levels of process maturity. The CMM has gained tremendous popularity in the industry, so much so that many organizations are finding they must achieve certain levels in order to win software contract awards.

So what is the CMM? It is a model through which a quantitative assessment of an organization's software process maturity can be made. The CMM document for software published by the SEI describes the process areas that should be addressed and gives guidance on the activities required to get those processes in place. This model is based on the premise that real process improvement involves the entire software development organization, not just the groups that build the software. It requires commitment throughout the entire organization. A CMM assessment at an organization results in the assignment of a ranking from 1(Initial) to 5 (Optimizing) depending on how many of the 18 Key Process Areas have been successfully ingrained in the organizational software development process.

### **Benefits of Software Process Improvement**

Clearly the climb from level one to level five is a long and expensive journey. Why are so many companies willing to do this? There are many ways in which process improvement benefits the software development organization – both qualitative and quantitative. Some are fairly hard to measure well. Even those that are easy to measure are often undervalued because it is not until an organization reaches some beginning level of process maturity that they have the mechanisms in place to measure – making comparisons to the worst case hard.

Improved processes result in higher quality products. While product quality is a very hard thing to quantify – even though we may count defects per line of code (or other size measure) in the released product it is still hard to quantify those missed requirements or features that failed to meet any client need. Certainly focus in level two on Requirements Management and Software Quality Assurance begins to address better analysis and early defect detection (before the product is released rather than after). It's the move to level 3 that really starts to make an impact on overall product quality. The introduction of Software Product Engineering and Intergroup Coordination results in products that deliver the right functionality in a low defect package. The introduction of Peer Reviews starts the process of preventing defects before they begin. This improvement in product quality has the added benefit of lowering maintenance costs.

Improved processes not only result in better products, they lead to better products that can be built in less time for less cost per line of code. There are all kinds of studies that support this – Productivity increases from 60 to 100%, Cycle time decreases from 25 to 75% - and other numbers all over the map, but these numbers need to be viewed in the context of how the studies were done, where the measurements started and what assumptions were made. Many factors contribute to the increased productivity and reduced cycle time. Processes that focus on forethought, inclusion of all interested parties from the beginning, commitment from all levels of the organization, peer reviews and training all contribute to working smarter and getting the most out of each hour.

In addition to the dollar and cents incentive, process improvement leads to a software development environment that people are happy to work in. Mature organizations offer environments where creativity can thrive within the confines of process. The process areas are meant to constrain the management and execution of the projects, not the content. The mature organization is proficient at predicting cost and schedule, so project plans are realistic. The mature organization is proactive rather than reactive, so developers spend their time writing excellent code instead of putting out fires.

### **Software estimating tools benefit your software development organization in many ways**

Long before software process improvement and the CMM were common vocabulary in the software world, there was wide spread recognition that software project managers needed better ways to estimate the costs and schedules of software development projects. In the early 70's two concurrent research efforts resulted in two parametric software cost-estimating models available to the software development community (COCOMO and PRICE S). Since then many new models have been developed. While mainly derivatives or expansions of the original offerings, each new model has evolved differently and thus offers different benefits and contains different shortcomings. The following discussion attempts to describe in general what a software cost estimating tool does and how it works without delving in the details of specific models.

Software cost-estimating tools solicit input from the users describing their software project and from these inputs the tool will derive a cost (and usually schedule) estimate for that project. The process that drives inputs to outputs is either cost estimating relationships derived from regression of actual data, analogies comparing input parameters to existing knowledge bases, algorithms derived from theoretical research, or some combination of these methodologies. At a minimum, the cost estimating tools ask for the user to describe;

- the size of the software (either in source lines of code (SLOC), Function Points (FPs), or some other sizing metric)
- the anticipated amount of reuse
- the type of software being developed (real time, operating systems, web development, IS, etc.)
- the operating platform of the software (commercial or military; ground, air, space, or desktop)
- a quantification of the organization's software development productivity

Most tools offer tables, wizards or knowledge bases to assist in the selection of proper inputs for the novice user or the user moving into new products, platforms or technologies. The tools will

also gather input about the software development environment (programming language, tools, etc.) and the software development experience of the team. This information is then used as the basis of the cost and schedule estimates.

Although cost and schedule estimates are the main deliverable of the software-estimating tool, there are many other needs the right estimating tool can address for your organization. Software project planning is really a balancing act between cost, schedule, quality and content and the right software-estimating tool can help optimize this balance. Many tools offer the capability of estimating latent defects in the delivered product and then use this information to predict maintenance costs. This offers the project manager the capability to make trade-offs based on the total cost of ownership, rather than just development costs. Most tools have other trade-off and analysis features as well – allowing the user to set a baseline and vary different parameters to optimize cost and schedule. Another important feature that most cost-estimating tools deliver is the ability to perform a Risk Analysis, so that a confidence level can accompany your estimate. Your organization can use a software cost-estimating tool to help derive a common language (the tools input parameter set) to describe and compare software development projects and a common productivity measurement to make reasonable comparisons between projects that have technical and operational differences.

There are, of course, limitations to every software cost-estimating tool which are important to understand. Each tool was created and is maintained using a certain set of data and research that does not include all types of projects, platforms or technologies. It is important to understand the limits of the tool you use and know when you are estimating on the edge or outside of these limits. Many organizations find it best to include in their estimating processes, the use of more than one tool or methodology – one for performing an estimate and another to use as a sanity check. Doing multiple estimates helps ensure that you won't miss a case where one particular tool is weak. It is also very important, no matter what tool or method you use for your software estimates, to understand what the tool is looking for when it asks for particular parameter values. As with everything else, if your input is not well thought out your output will suffer.

### **Software-estimating tools can play an important role in your process improvements**

What part will your software-estimating model play in your organization's drive to a higher process maturity level. There are processes at every level that require the standardization of estimating, data collection, quality control, measurement, and analysis. These are all areas where an estimating tool can help add structure around which processes can be defined. The following paragraphs address some specific KPAs that have direct requirements that an estimating tool will help meet.

#### **Level 2 KPAs**

One of the goals for the Software Project Planning KPA is that 'Software Estimates are documented for use in planning and tracking software projects'. The software plan is expected to include size estimates for all of the work products along with cost and schedule estimates. A software-estimating model, with capability to estimate software size, cost and schedule, provides an excellent tool for institutionalizing these estimating practices.

The main value a software-estimating tool can add at this point in your process improvement venture is the ability to estimate software cost and schedule consistently and logically. The software-estimating model acts as the lowest common denominator, aiding the process of putting software projects into a common framework so that information can be learned from each project and applied to many others. Imagine that you are a software project manager who has just delivered a software project late and over budget. You are gearing up for the next project and would like to learn from this past experience, but how do you evaluate where you went wrong. Using your software-estimating tool as a guide, you can determine values for the input parameters that would have led you to the right answer based on the actual experiential data you have collected. Once you have done this, you can, based on what you know about this new

project, determine which of these input parameters will need to change and how the input values might change. Using a commercial model for estimating facilitates the creation of a common language for discussing project cost drivers and aids in the development and implementation of a documented process.

One of the goals for Software Project Tracking and Oversight involves taking and managing corrective actions when results deviate from the plan. A software-estimating model can be a useful tool in reaching this goal as well. When a well thought out estimate turns out to be incorrect, this is generally because assumptions made about the project were incorrect. Incorrect assumptions lead to wrong inputs. The software project team can review the original inputs to the model and compare to actual information to date. This review helps highlight where incorrect assumptions have been made and offers the opportunity to re-plan the remaining portion of the project based on more accurate versions of assumptions. Suppose you are involved in an effort to estimate a software project that represents a new market for your organization. In developing your original estimate you made and documented (through an input wizard in the tool) assumptions about your software development team's capability to learn and apply domain knowledge for this new market. You also assumed that the team was proficient in the technologies employed. The critical design review arrives (with the project late and over-budget) and you are tasked with determining what went wrong and what the real cost and schedule should be. You look back at the original estimate and compare it to what you know about the project. First of all, the team took much longer than expected to obtain domain knowledge. Additionally, once the project got underway, they determined that the incorporation of a new development technology was required to achieve all of the goals of the project. You modify the inputs to your estimating tool and regenerate the estimate (leaving all other parameters the same) only to discover that now the critical design review cost and schedule is much closer to the actual. You now have an estimate to complete with a new level of credibility. Care should be taken to make sure that use of an estimating tool does not lead you to overlook factors outside the scope of the tool. You may find that there was nothing wrong with your original estimate, but that the missed deadline was due to unforeseeable organizational changes that no tool could account for.

Another process area where a software-estimating tool can be included is the Software Subcontract Management KPA. Not only does there need to be a process for planning software developed on site, there needs to be a process for reviewing and checking the plans provided by a subcontractor. These plans are also based on size, cost and schedule estimates. The language that a commercial tool offers for talking about those things that drive cost can easily be used by subcontracting organizations as well.

### **Level 3 KPAs**

As an organization begins to attack those processes that are required to move from repeatable to defined processes, the focus shifts from the project level to the organizational level. One of the goals of the Organization Process Focus KPA is that process activities are coordinated at the organizational level. An important part of the achievement of this goal is focused on the organization's software process database that collects process and project data. The parameters or inputs used within a commercial estimating model contribute to the development of a generic framework for describing product and project characteristics like functionality, quality, size, complexity, and reuse in such a way that comparisons across the organization are possible. Groups within the organization can use this language to compare unlike projects in ways that provide useful analysis to feed the improvement process. For example, imagine you work for an organization that develops avionics for both military and commercial applications and the military side of the house is attempting to evaluate the cost impact of incorporating a new technology that the commercial side of the house has been using for some time. Using the data learned from the commercial avionics development, the software estimator need only change the information directly related to the operating platform to apply lessons learned to evaluate how this new technology would change costs on a military application. The input parameters for the model help focus the evaluation on what's different and what's the same when performing this type of comparison and help remove noise from the comparison.

As processes begin to be defined at an organizational level, the real power of a commercial model becomes clear. Part of the Integrated Software Management KPA requires that an organization use data in the software project database for software planning and estimating. This is a process that we call calibration and it is automated as part of the many good estimating tools. As noted earlier, in implementing a commercial tool, the organization has already committed to storing data in the process database in a way that makes comparisons possible at an organizational level. The tool then has the capability to look at this historical data, describing the actual past performance of the organization, and use it to improve the estimates made from that point.

Another question frequently asked is how one might need to change the inputs that they provide to an estimating model as they progress to higher levels of maturity. The beauty of a process improvement approach such as that dictated by the CMM is that once an organization has reached a new level of maturity they already know the answer to this question. The increased data collection and analysis with more mature organizations provides the information required for calibrating and tailoring the inputs to reflect organizational maturity accurately. The value of the tool itself is improved tremendously by the processes that utilize it.

#### **Level 4 and beyond**

Most software-estimating models contain features that can help meet process needs for the Software Quality Management KPA. A model that contains a sub-model to estimate defects per size measure provides the basis for a process that allows for the control and management of a quality plan through trade-off analyses between quality, schedule, and content goals. This sub-model can be calibrated through use of organizational data from the process database. Suppose you have as an organizational goal, the reduction of latent defects in you delivered software by 20%. At level 4 , you have enough data in the software process database to evaluate defects delivered in the past and calibrate the estimating tool's defect estimate. Once you have done this, you can evaluate every proposed project for estimated defects and make corrections to the project plan to make this goal more likely – extending the development schedule or reducing the amount of content expected in the given time frame until the quality goal is met.

The Technology Change Management KPA requires that an organization have a well-kept process database with productivity and quality metrics from past projects and a language and framework for introducing new technology parameters. Your commercial software-estimating model can be an important component in that framework and it's parameters for characterizing software projects constitute an important part of the language. A mature organization has much of the right data about the projects that have been done in the past. The cost estimating relationships or knowledge bases in your tool have encapsulated the impacts of new and emerging technologies. The marriage of these two stores of information offers an organization excellent insight into how their existing capabilities and experiences merge with new target technologies and provides information vital to making the right technology decisions.

#### **Conclusions**

Process improvement is a worthwhile investment for any software development organization. The CMM is an excellent resource for any organization that has a plan to improve their processes, even if the CMM is too grand a scale for their process improvement needs. Review of the CMM and concentration in those areas that will meet organizational goals is a must for any organization seriously considering process improvement because it is thorough, comprehensive and encapsulates a lot of good ideas from many experts in the software process field.

No matter how large or small your process improvement plans are you will find that a commercial software-estimating tool will ease your efforts towards standardization and control. The most value will be obtained if you incorporate it into your organizational practices and make it a

common language for discussions around the things that drive your software costs. It is not until all the projects in the organization can be discussed in a common context that true organizational needs and concerns can be addressed. A commercial model will help provide the framework for these discussions. Not only does it offer a way to make this happen, but as the tool you choose adds value to your software process improvement program, your software process improvement results are making your tool more valuable to your organization.

References:

1. Craig,Rushby, "Software Quality Assurance in a CMM Level 5 Organization", Crosstalk, May 1999, Volume 12, Number 5
2. Paulk, Mark C., "Practices of High Maturity Organizations", *Proceedings of 1999 SEPG Conference*, Atlanta, Georgia, March 1999
3. Paulk, Mark C., "Effective CMM-Based Process Improvement", Software Engineering Institute, Carnegie Mellon University, 1996
4. McGarry, Frank et.al, "Software Process Improvement in the NASA Software Engineering Laboratory, Technical Report. Cmu/SEI-94-TR-22, December 1994
5. Lawlis, Patricia K., "A Correlational Study of the CMM and Software Development Performance", Crosstalk, September 1995, Volume 8 Number 9