

TRACKING PROJECT SIZE ATTRIBUTES TO MONITOR PROJECT PROGRESS

By Neil Potter

Tracking project size attributes provides an early warning system for problems with project progress, project growth and stability. Size data can be used to validate the likely completion date of projects and to provide the team and management with a visual picture of the project's magnitude. In this article, summary examples are provided of how companies track size. Tailor these examples to fit your needs or use them as a starting point to generate your own tracking ideas. When you are ready for more detailed information on project metrics, review the books in the reference section of this article.

In some texts, such as the Software Engineering Institute's Capability Maturity Model Integration (CMMI), size is referred to as a *Work Product Attribute*. This is a generic term that can be applied to any work product, including intermediate work products (such as a design document) and final work products (such as code and assembled hardware). In this article, "size" is one adequate implementation of a work product attribute.

When reading about size, it is tempting to assume that "My group is different," and that no size metric is applicable. It is also common for software developers to assume that size must mean lines of code (LOC), and that LOC is never an applicable metric. If you have this filter, know that size is simply a parallel and early view into the amount of work a project has to do, one that can be used to estimate effort and monitor progress. The absolute accuracy of the metric is not as important, so long as the metric changes proportionally in value when the amount of project work changes. Five example-size tracking metrics are summarized here. Each one provides a different window into the project. They are: *requirements count*, *requirements growth*, *requirements completed*, *cumulative requirements changes* and *units coded*.

Requirements count

The unique requirements of a project can be counted, and changes to that count plotted over time. A prerequisite to counting requirements is to split ambiguous paragraphs of requirements into unique requirement statements. The precision of the count is not as important as knowing the magnitude of the requirements to be implemented and observing changes to the count over time. If the requirements are not uniform in scope, assign a complexity or weighting factor to make the requirements count larger when requirements are complex. In Figure 1, the requirements count line starts at 60 requirements, and increases late in the project. At this time, a re-plan is likely to be needed.

Two other examples are the *use case points metric* [Jalote02] and the number of screens on a user interface. The use case points metric is based on the number of use cases, their complexity and factors including the stability of the requirements and the experience of the team. The metric number is then multiplied by 20 person-hours to derive an initial project effort estimate. This

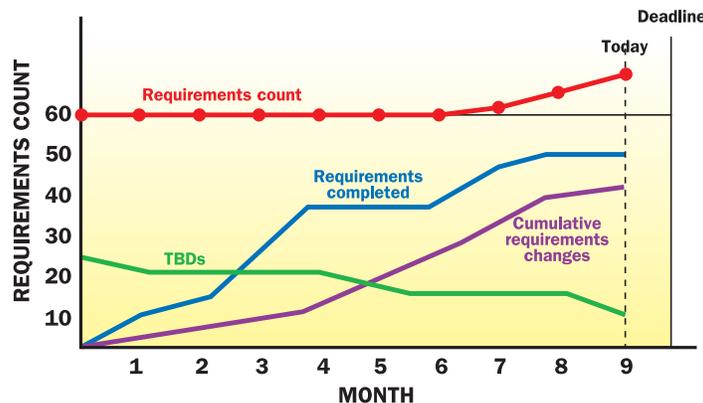


Figure 1 Tracking requirements metrics, based on [SEI92]

number, and the effort expended on the project, is then tracked as the project progresses.

The second metric uses the number of screens in a user interface. A count of the screens provides a rough order of magnitude of project effort (e.g., 1 screen, 10 screens, 100 screens). If the effort required to complete the project increases when the number of screens increases, then the

screen count is a suitable size metric. In the beginning of the project, the metric is used to validate the effort required by multiplying the number of screens by the average effort required per screen. If the number of screens changes, one can expect the effort or schedule to change also. If the screens have a wide variation in complexity, assign a weighting factor.

(Continued on page 3)

CAPABILITY MATURITY MODEL INTEGRATION (CMMI)-BASED APPRAISALS: HOW LARGE SHOULD THEY BE?

By Mary Sakry

Organizational expectations from CMM and CMMI process appraisals have shifted over the years. Seventeen years ago, we would normally have been contacted by an organization interested in appraising to determine what improvements to make. (They assumed that they were probably operating at CMM Level 1.) It was normal to appraise small organizational units that wished to take the findings and make improvements to their development process. At that time, there was no detailed maturity model to compare against.

We have seen a shift toward wanting to pass a test for as large an organization as possible. Appraisals are now often seen as a final gate, rather than an aid in determining needed improvements. With this comes the push for the organization to be perfect before the appraisal (with an incorrect focus on documentation), and a desire to cover a very large organization with one appraisal.

This raises two issues: (1) how useful are the results of these marathon events? and (2) how can the organization feel it is getting an accurate picture of its maturity level?

We maintain the premise that conducting one huge appraisal for 250-1000 people is not recommended. One large appraisal has the following risks:

- The population combined would receive a CMMI rating of Level 1 if any significant piece (or large project) was Level 1.
- Interviewing a small sample from 1000 people (for example, 10-40%) could lead to inaccurate appraisal results. An appraisal needs to determine if CMMI practices are institutionalized across the majority of projects. A single appraisal *could* leave 60% or more of the population untouched. The remaining 600 people, in this example, could claim to be CMMI Level 3 even though they have never been interviewed or evaluated in any way. Institutionalization can only be confirmed by appraising a larger sample of a smaller population at one time (e.g., 50-70% of a population size of 100-250).
- Performing one large appraisal would be a logistical challenge. All interviewees would have to be available at specific times during the appraisal.

- The findings (areas to improve) from a single appraisal might be difficult to address since they would summarize the performance of all areas combined and would not be specific to any one area.
- If the selected projects were rated at Level 1, it would be impossible to know if there were projects actually performing at Level 3 in the remaining population.
- A follow-on appraisal by a government agency (or other potential customer) of a specific project could result in an unexpected maturity rating and loss of contract award.

Findings need to be written in a way that allows interviewees to speak candidly, thus no specific project attribution is ever given. When you add this to the problem the team has in summarizing diverse practice implementations performed across an organization, the appraisal findings can become too generic to be actionable (such as, “The estimation methods vary greatly and are rarely followed.”) Some of the estimation methods could have been working very well, others not as well, and some almost ignored.

We believe that an appraisal is more suitable for a population of up to 200 people. The appraisal could then be repeated for any remaining logical sections of the total population (i.e., 5 appraisals covering 1000 people). If all sections were rated at Level 3, the whole division would be Level 3. Any one section could be rated Level 3 with one or more of the other sections rating lower.

In summary, we are concerned that the current trend is toward ineffective appraisals that cover too much ground and provide a poor reflection of an organization’s true maturity level. In addition, the results of such appraisals are difficult to use for process improvement since they cover such a diverse set of people and development practices.

Therefore, conducting one appraisal of a large organization neither helps the appraisal team be confident in its result, nor helps the organization know where it truly stands.



Requirements growth

Monitor changes in scope by counting the revised number of requirements (the “Requirement count” in Figure 1.) This line provides an early warning signal of a schedule risk if more requirements are added to the project. The project might be able to absorb some new requirements, but at some point, the work associated with these requirements will exceed the established schedule and budget.

The line labeled “TBDs” (To Be Defined) is a count of the placeholders in the requirements document where requirements need to be clarified. Initially, this can be included in the initial count. As the project progresses, this number should decrease, showing that the ambiguity of the project is decreasing.

Requirements completed

Progress can be tracked by counting the number of requirements that have been completely implemented (see the line labeled “Requirements completed” on the graph in Figure 1). Although this is a gross measure of progress, it is one that shows how far along the project is from a customer and release-readiness view. The validity of the metric is dependent on the definition of a requirement being “completed.” A “completed” requirement might be one that has been successfully tested.

Cumulative requirements changes

The line labeled “cumulative requirements changes” monitors the volatility of the project scope. A project team should expect requirements to change as they are clarified. However, many changes late in the project add the risk of the team not being able to complete any feature, or only having limited time to test the features that have been completed. This line can also indicate a situation where there are so many changes, that the team is effectively starting over with a new set of requirements.

Units Coded

“Units coded” is one example metric that provides better visibility when software is being developed. (Similar metrics for other phases would be *designs completed*, *tests completed* and *defects repaired*.) In the graph in Figure 2, the project initially estimates that there will be 60 units of code to create. As the project executes, the number of modules actually completed (here, completed is defined as peer reviewed, tested and under configuration management) is tracked on the graph. Tracking is kept simple in this example by assuming that a straight line reflects the desired progress between the start and end of the project.

At week three, the team members notice that their rate of progress (see the line labeled “Actual”) is approximately half of the initial planned rate. This causes a re-plan and a new baseline.

The graph can also be used as an early warning system of potential schedule problems. For example, if the team members discover new code that needs writing, this new estimate can be added to the graph. The change can be seen in context to the overall project work estimate, and any upwards or downwards trend can be detected.

Selecting metrics

You will not always be able to have one size metric that is valid for the duration of the project. Specific size metrics might be more useful by project phase.

For example, you might measure the number of requirements baselined (requirements that have been defined and agreed upon) during the requirement phase, and the number of repaired test defects during the testing phase.

Earned Value [Humphrey95] is an example metric that tracks tasks completion for any phase of the project. If the tasks in the project schedule are at a similar

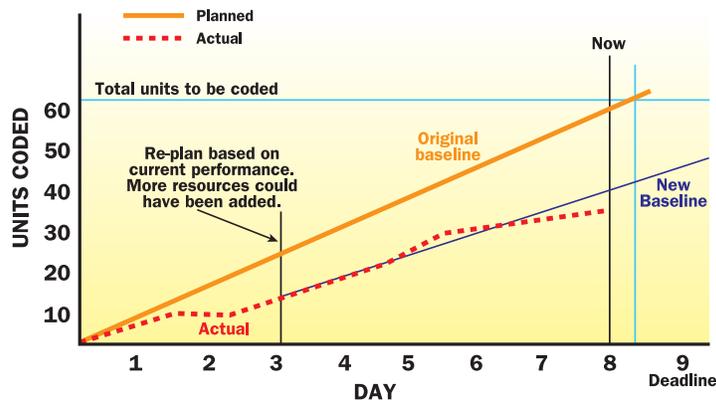


Figure 2 Simple trend graph showing overall project size and progress for the coding phase

level of granularity, the metric provides a uniform method of determining current status. (The difficulty is that each task needs to be clearly defined.) The rate of task completion (called Schedule Performance Index) can be calculated and used to predict the chances of hitting the final deadline.

Summary

Size metrics provide a view into the amount of work to be performed in a project. In the beginning, a size estimate can be used as input into effort and schedule calculations. As the project progresses, size can be monitored and scope changes can be rapidly detected.

[Jalote02] Jalote, P. *Software Project Management in Practice*. PP 58–65, Boston: Addison-Wesley, 2000.

[Humphrey95] Humphrey, W. *A Discipline for Software Engineering*. PP 180-196, Reading, MA: Addison-Wesley, 1995.

[Wiegers03] Wiegers K. *Software Requirements*. P 342, Second Edition, Redmond, WA: Microsoft Press, 2003.

[SEI92] *Software Measures and the Capability Maturity Model*. J. H. Baumert MS McWhinney. Technical Report CMU/SEI-92-TR-025.

Practical Solutions for your Software Development Challenges

❑ Understand customer needs. Clarify product requirements early.

In this workshop, IN SEARCH OF EXCELLENT REQUIREMENTS, software engineers, managers, requirements analysts and user representatives learn how to gather, document, analyze and manage customer requirements for software applications.

❑ Decrease product development time-to-market. Reduce costs.

In this workshop, ACCELERATING PRODUCT DEVELOPMENT FOR SOFTWARE PROJECTS THROUGH LEAN CYCLE TIME REDUCTION, project managers and their teams learn how to accelerate delivery through specialized schedule optimization and “Lean Thinking” techniques.

❑ Manage projects effectively. Meet project deadlines and reduce risks.

In this three-day SOFTWARE PROJECT PLANNING AND MANAGEMENT workshop, project managers and their teams learn how to meet deadlines through better estimation, reduce surprises using risk management, schedule work for better optimization, understand and negotiate project trade-offs, and track progress.

❑ Meet project deadlines. Scope and estimate the project work.

This one-day SOFTWARE ESTIMATION workshop (a subset of Software Project Planning and Management) helps teams develop more accurate estimates.

❑ Avoid schedule delays caused by needless product rework.

Find defects rapidly.

This two-day INSPECTION (PEER REVIEWS) workshop teaches teams to efficiently find defects in code and documentation. (Includes moderator skills.)

❑ Hands-on SEI CMM/CMMI. Perform a CMM/CMMI gap-analysis.

The following workshops are available:

- ❑ SEI CMM: LEVEL 2 (one day), SEI LEVEL 3 (two days), SEI LEVEL 4 (one day)
- ❑ SEI CMMI: Overview (half day), LEVEL 2 (one day), LEVEL 3 (two days)
- ❑ SEI INTRODUCTION TO CMMI (three days)

❑ Identify critical changes to improve organizational results. Benchmark against the CMM/CMMI.

A SOFTWARE PROCESS APPRAISAL examines your organization’s software practices and generates a focused list of the critical areas for improvement. Our SEI-authorized Lead Appraisers conduct customized CMM/CMMI-based appraisals.

❑ Goal/problem-based improvement.

This two-day MAKING PROCESS IMPROVEMENT WORK workshop provides a systematic approach for organizations to improve their development capability. It includes: getting management support, focusing the organization on the critical issues, planning the improvement and effecting change.

❑ Manage your subcontractors.

In this one and one-half-day workshop, SOFTWARE SUBCONTRACT MANAGEMENT, software engineers, managers and subcontract managers learn how to define a software product to be outsourced, write a subcontract management plan, select appropriate vendors and manage the project to completion.

❑ Tailored assistance. Dedicated phone-based assistance.

This service consists of customized education and coaching on your specific problems (e.g., meeting deadlines, quality and cultural change.)

Detailed information on our services is available at www.processgroup.com.

Contact us at 972-418-9541 or help@processgroup.com to discuss your needs.

Come see our book!

www.processgroup.com/tpgbook.htm
Also available in Chinese at
www.china-pub.com

Here is the book’s Table of Contents:

Foreword by Karl Wieggers.

Preface.

Acknowledgements.

Chapter 1. Developing a Plan.

- Scope the Improvement.
- Develop an Action Plan.
- Determine Risks and Plan to Mitigate.
- Chapter Summary.

Chapter 2. Implementing the Plan.

- Sell Solutions Based on Need.
- Work with the Willing and Needy First.
- Keep Focused on the Goals and Problems.
- Align the Behaviors of Managers and Practitioners.
- Chapter Summary.

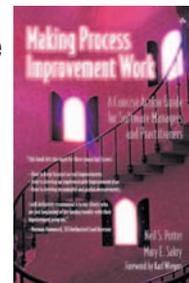
Chapter 3. Checking Progress.

- Are We Making Progress on the Goals?
- Are We Making Progress on our Improvement Plan?
- Are We Making Progress on the Improvement Framework?
- What Lessons Have We Learned So Far?
- Chapter Summary.

Conclusion.

Appendices.

References.



The Process Group

Mailing address: The Process Group
P.O. Box 700012
Dallas, TX 75370

Telephone number: 972-418-9541

Fax number: 972-618-6283

E-mail: help@processgroup.com

Web: www.processgroup.com

POST back issues are on line