Thought Before Action:

The Advantage of High-Maturity Thinking in a Lower-Maturity Organization

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

James D. McHale
CMMI NDIA 2007

Some Radical Thoughts

Fallacy: Any time spent on the higher maturity level practices while attempting to achieve CMMI ML2 or ML3 is, by definition, wasted effort.

Radical Thought #1: Any time spent implementing policies and practices at ML2 and ML3 that does not support the higher maturity level CMMI practices violates the intent of the model.

- Otherwise serious rework can be required to achieve ML4 and ML5.
- At the extreme, ML2 and ML3 practices are implemented poorly and for all the wrong reasons.

Radical Thought #2: You need to understand ML4 and ML5 concepts before you can properly interpret ML2 and ML3 for your organization.

mprovement?

The phrase ‟process improvement" implies improving the ***performance*** of a given process or set of processes with respect to some objective standard.

- CMMI does not specify performance standards, it only implies their existence.

Improving performance with respect to an objective standard implies that something about the process will be measured.

"If you can not measure it, you can not improve it." – Lord Kelvin

s Performance?

Process Performance: ñA measure of actual results achieved by following a process. It is characterized by both process measures (e.g., effort, cycle time, and defect removal efficiency) and product measures (e.g., reliability, defect density, and response time).ò

Process Performance Baseline [PPB]: ñA documented characterization of the actual results achieved by following a process, which is used as a benchmark for comparing actual process performance against expected process performance.ò

Process Performance Model [PPM]: ñA description of the relationships among attributes of a process and its work products that is developed from historical process-performance data and calibrated using collected process and product measures from the project and that is used to predict results to be achieved by following a process.ò

- from the CMMI Glossary

Performance Model?

A process performance model is used for essential process improvement activities.

- explain past performance (e.g. the PPBs)
- predict future performance (may look like the PPBs in part)
- indicate what (else) to measure
- identify opportunities for improvement

Are these purposes guiding *your* ML2 and ML3 practices?

Can you do these things without the statistical rigor demanded by ML4/5?

ine

In ***Mythical Man-Month***, Fred Brooks gave this gross characterization of effort distribution of programming processes.

Planning and design	1/3
Coding	1/6
Unit Test	1/4
System Test	1/4

This characterization provides a ***baseline*** (although not a complete PPB in the CMMI sense) for process performance at IBM in the late 1960s.

It can help to ***explain*** past process performance, and when combined with an estimate of effort on a future similar project, it can help to ***predict*** future performance of the process (although not yet a PPM).

and What's Missing

Based on this historical benchmark, if I have an enhancement project that I estimate at 100 hours, my predicted performance would be:

Planning and design	33 hrs.
Coding	17 hrs.
Unit Test	25 hrs.
System Test	25 hrs.

Do I have any idea of how relevant this prediction is to me?

Do I have any idea of which activities have the most opportunity for improvement?

Do I have any idea of how to push this in the direction of a true PPM?

Project Data

Planning and Design	27 hrs.
Coding	38 hrs. (until clean compile)
Unit Test	38 hrs. (21 defects found)
System Test	35 hrs. (11 defects found, 3 passes of test suite)
Total	138 hrs. (vs. estimated 100 hrs.)

Do I have any idea of how ñormalòthis may or may not be for me?

What should I be measuring in more detail on future projects?

s I Want to Know

How much time in planning vs. design vs. understanding requirements?

How much time fixing compile/environmental defects?

How many unit test cases? How many passes (partial and complete)?

How much time executing system test suite vs. fixing defects?

rovement Proposal (PIP #1)

Process problems:

1. No way to tell from the gathered data how much time was spent on planning vs. design or other activities in that phase
2. No way to tell how much time in coding was in fixing compile or link defects
3. How much test time in testing vs. fixing

Proposed solutions:

1. Tag all hours with óplanningô, ódesignô, óanalysisô, óotherô
2. Tag all hours in coding with ócodeô or ócompile/fixô
3. Tag all hours in test with átesting <case #>ô or ádefect find/fix <bug #>ô

ugh analysis

After a couple more similar projects:

Phase	Project 1	Project 2	Project 3	Cum. %
Planning & design	27	38	47	25.8
Code	38	26	39	23.5
Unit Test	38	28	43	25.2
System Test	35	29	47	25.5
Totals ï Act./Est.	138 / 100	122 / 110	175 / 120	100.0

From hour tags, understanding requirements is about 1/2 of ṝplanning and designòtime, actual planning and design about 1/4 each.

Defect fix times in UT and ST are about 70-80% of the total test time, more if you count all of the extra passes needed in the test suites.

alysis continued

On average, the phases are fairly equally balanced.

However, looking at individual efforts for ñplanning and designò as a predictor, those were much different (about 1/5, 1/3, 1/4, respectively).

Prediction of the cumulative time measured after ñcodingò seems much more reliable, always about 1/2 the total project time in ñplanning & designò and ñcodingò, the other half in ñunit testò and ñsystem testò.

ñUnit testò is a fairly good predictor of ñsystem testò, even though the tests run are completely different.

Estimates arenôt very good (about 30% average overrun). If only we didnôt have to do ñsystem testòé

Characterizing key relationships and their variation statistically helps to make a PPM.

Process problems:

1. Many defects and multiple test suite passes (waste of time!) are due to not being able to find all defects in the first pass.
2. Effort overruns are creating increased project tracking overhead (i.e. management pressure).

Proposed solutions:

1. Provide inspection training & require inspections of all code; log all inspection effort and defect data.
2. Increase effort estimates by an amount large enough to allow for variation in key performance indicators.

spections

Phase	Project 4	Project 5	Project 6	Cum %
Planning & design	20	40	40	22.9
Coding	25	45	75	33.2
Unit Test	21	33	37	22.2
System Test	24	30	32	21.7
Totals ï Act./Est.	95 / 75	148 / 185	184 / 215	100.0

Inspection time was rolled into ñcodingò since it is the code being inspected, about 1/4 of the total ñcodingò effort.

UT and ST about 42% of total effort, down from about 51%.

Actuals are about 11% under estimates on average, but they would have been about 18-19% over if not for 1/3 ñeffort adjustmentò.

ata Analysis / More Questions?

On this basis (18-19% vs. 30+% over), inspections seem to be working. (Remember to compare apples to apples!)

Defects found in code inspection tend to be simple coding errors, with the occasional design defect.

About 60% of total testing effort still devoted to finding defects and multiple test suite runs. A majority of defects now seem to be design issues (used to be about even between design and coding issues).

ships

ŕPlanning and designò and ŕcodingò effort seem to relate directly to the scope of the project.

E1 (effort before test) = ***f***(scope)

While loosely related to scope, testing effort seems more directly related to the number of defects and the number of test suite passes.

E2 (effort in test) = ***f***(defects) + (effort in 1 pass through UT and ST)*

* - probably related to scope!

Problem description:

1. Effort estimates under management pressure
2. Still lots of ñwastedò time in UT and ST

Process proposal:

1. Reduce the 1/3 ñeffort adjustmentò to 1/5
2. Create more ñinspectableò designs by using design templates or architectural views; inspect for common design defects found in test

Note: Is either proposal ñstatistically soundò? (Probably not.)

What would you do instead? (Hmmm… … .)

Rhetorical Questions

Is the gathering and use of data *by the people doing the job* high- or low-maturity?

Do I have to be ML4 or ML5 to do any of this?

Will this *make* you ML4 or ML5 (or any level) if you do this?

Have you seen control charts? Complex mathematical models?

Do you think that such practices would help speed you on your way to ML4/5?

v Requirements (Constraints)

The Voice of the Business (your boss) tells you that your performance goal for next year is to deliver your projects in 85% of the calendar time that you estimate with fewer defects delivered to the external customer.

Your standard process simply cannot perform to this level.

There are two basic types of response to such pressure.

- Å low maturity (try harder! i.e. more than 40 hours/week)
- Å high maturity (work smarter!)

ity Response -1

Your current process baseline (still not a PPB) looks like this:

Phase	% actual effort	Defect yield*	Notes
Planning and design	30	40%	Based on defects reported from the field. Early yields are from inspections. Single pass of UT and ST ~10% of effort.
Coding	35	50%	
Unit Test	15	40%	
System Test	20	40%	

You need to squeeze 15% out of your average estimated lifecycle effort.

You are still doing multiple passes of extensive (and expensive) testing.

If only you could reduce the number of passes in UT and STé

* Defect yield ï percentage of defects found in phase that were present or injected in that phase

y Response -2

If you could increase yields in the early phases, you could further reduce the number of defects in UT and ST and, more significantly, finally reduce the number of test passes.

You canõt wave a magic wand at inspections and say, ñFind more defects!ò

But youõve heard or read of other methods that drastically reduce the numbers of test defects.

- PSP/TSP
- Correctness by Construction
- Test-Driven Development

Pick one. Investigate. Better yet, get your process group to do it! Or at least pay for the training. (But donõt tell them itõs the ML5 thing to do, it might scare themé)

-Independent) Thoughts

Process is like exercise.

If you arenôt used to it, it hurts.

Once you do get used to it, if it still hurts, you are either

- Å trying to do too much
- Å doing it wrong

It gives you more time and energy to do all the other stuff you know you ought to be doing, so you get more done.

Itôs usually a little easier and a lot more fun when performed in groups.

emember

CMMI is a model that encourages (and ultimately demands) process performance improvement.

While it wonôt get you a ML4/ML5 rating, you can *begin* implementation of high-maturity concepts with very simple models and techniques. (Let the data show the way!)

Significantly improved performance on your projects is achievable now, regardless of maturity level.

jdm@sei.cmu.edu

Software Engineering Institute | Carnegie Mellon