

Cost of Quality: a Key Effectiveness Metric for Software and IT

By Gary Gack

Many different metrics have been proposed and sometimes used, but most organizations have great difficulty agreeing on a core set that provide an overall perspective. This White Paper proposes "Cost of Quality" metrics that get to the heart of *organizational* efficiency and effectiveness in software and IT organizations. These metrics facilitate (but are not a pre-requisite for) effective application of Lean Six Sigma in software and IT organizations. Part 1 of this paper defines Cost of Quality metrics as they apply to software and IT organizations. Part 2 discusses application of these metrics to process improvement.

We deal here with effectiveness from the perspective of execution – selecting the right things to do is certainly important, but is not our topic here. We concern ourselves here with quality, cost, and cycle time.

Part 1: Cost of Quality Metrics for Software and IT

Labor, including both staff and contractors, is nearly always the largest element of cost in software or IT organizations. Hence, understanding and improving the use of labor is fundamental to any improvement strategy. Many software/IT groups have systems and processes in place to track and report labor use against a multi-level "chart of accounts" that may include dozens, hundreds, or even thousands of charge categories. A "typical" chart of accounts for Applications Development and Support is illustrated in the following table:

Applications Development
Line of business
Project (dozens to hundreds)
Life Cycle Phase (e.g., Requirements, Design, etc.; 5-10)
Task (tens to thousands)
Applications Support
Line of business
Application (dozens to hundreds)
Activity (e.g., defect fix, question, data correction, etc.; perhaps 10)

At first glance, we might believe that an organization with this much data would be well positioned for Lean Six Sigma but in practice that is often not so. More often than not there are serious issues with the data, including:

1. Reliability of the data is inversely proportional to the number of charge categories. More categories mean more entries for each individual reporting and more time to do the reporting. Most individuals will enter time weekly or even less often, and cannot accurately remember how time was actually spent. Many individuals will do development and support activities in the same week, work on several development projects, and several phases/tasks within each – commonly one person will spend time on dozens of different charge categories during any given week or month.
2. Many de-facto incentives exist that distort the data – to mention just one, it is common practice to stop charging time to a project when the budget has run out – instead, time is charged to something that has remaining budget, independent of the actual work being performed. Similar distortions

occur between phases. The infamous "hear no evil, see no evil" conspiracy prevails. No one really wants to know the truth.

3. There is usually no feedback to the individuals reporting time, and no decisions typically result – hence, widespread cynicism becomes entrenched – "nobody uses this data, so who cares". Non-compliance and distortion become the norm.

More is not Always Better

From an efficiency improvement perspective (not necessarily from a project management perspective) task level time detail is not useful, because that data is not comparable across projects. What is common to all projects, and far more useful for measuring effectiveness, is a "Cost of Quality (CoQ)" view of time expenditures. For software and IT activities we commonly use a simplified three part CoQ scheme consisting of Value Added, Appraisal (testing, inspections, any activity performed to find defects), and Rework – collectively referred to as "internal" CoQ.

Significant additional costs, known as "external" CoQ, include all costs incurred by the development and/or support organizations associated with diagnosis and correction of defects released to customers. External CoQ should also include the costs experienced by the customer as a consequence of defects, but in practice these are very difficult to measure and are often ignored.

In many instances this approach will mean *less* time reporting, not more – fewer items in the Chart of Accounts. Alternately, if you are committed to more detailed time accounting in order to use critical path method and perhaps earned value, you can structure your charts of accounts so that the detail is explicitly mapped to CoQ categories.

Internal (pre-release) Cost of Quality

Appraisal, the first CoQ category, is all time spent finding defects – in most organizations this is mostly testing, but may also include inspections and reviews of various work products prior to testing. Most organizations devote 30-50% of total pre-release effort to this activity, but usually have no idea if that is enough or too much. Testing usually stops when time runs out, with little or no insight into effectiveness of the effort expended. The essential effectiveness metric for all types of Appraisal is **Cost (Effort) to Find a Defect = (Total Appraisal Effort / Total Defects Found)**.

Understanding this metric for each different type of Appraisal enables us to choose the most cost-effective combination of the various forms for reviews, inspections, and testing. In practice we always find early appraisals such as inspections have significantly lower cost per defect, commonly 3-20 times less than testing. (Defect tracking is a second essential source of metrics – that is addressed below.)

Rework is all time spent **fixing defects** found by any form of appraisal *before* a system is delivered and/or by customers *after* a system is delivered (external CoQ) - typically 30-50% of total effort, but rarely measured. Most organizations cannot separate Appraisal effort from Rework, let alone pre- from post- release rework effort. The essential effectiveness metric for Rework is **Cost to Fix a Defect = (Total Rework Effort / Total Defects Fixed)**. Note: ALL work done to "fix" an application after it has been delivered to the customer is Rework, which may include corrections to features or functions that are incorrect, but also may include *missed requirements* – things the customer expected but did not receive.

Value Added is simply total time spent minus time spent on Appraisal and Rework. When measured, which rarely occurs, Value Added is commonly 30% - 40% of total effort. When cancelled projects and projects delivered but not used are considered, Value Added may be even less.

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

IMPROVING EFFECTIVENESS, SIMPLY PUT, MEANS INCREASING VALUE ADDED! Any and all improvement initiatives can set goals and measure improvements in terms of impact of Value Added. Value Added is the central "Big Y" for all software and IT activities.

Note that this definition of Value Added is offered for the sake of "operational definition" simplicity – it does not exclude the possibility that some effort categorized as Value Added may be redundant or unnecessary. However, let's eat the elephant one bite at a time – when we get Value Added to 50% we can begin to consider refinements of our categories. Getting these three CoQ categories implemented will provide plenty of challenge for at least the first year.

Understanding Defects

Tracking defects means recording EVERY defect found by a customer or by any formal Appraisal process. That *does not* mean we are going to ask individuals to track every change – unit tests and individual code reviews / walkthroughs are usually *not* subject to tracking. Once an author declares a work product "complete" however, and releases it for independent Appraisal by others, all defects found should be tracked.

Every defect found should be identified with the following information:

- Work product in which the defect was found (this may need to be determined by the person doing the fix)
- The Appraisal method used to find the defect (inspection, type of testing, customer etc.)
- The origin of defect – i.e., where it was "inserted" (e.g., requirements, design, coding, etc.) – it is not always possible to determine the origin, but an adequate sample is usually feasible
- Project ID and life cycle phase (if during the project) or application ID (if after release) in which the defect was found
- Other information as necessary to track assigned to, status, closing information (not necessarily needed for process improvement, but typically required for management purposes)
- Defect type – a *short* list of orthogonal categories that make it possible to determine which type of defects are most effectively found by which types of appraisals

In earlier articles (link) I have dealt with use of a defect cost scorecard and Defect Containment Effectiveness (DCE) and Total Containment Effectiveness (TCE) metrics – these tools can be applied to manage differential effectiveness across phase of origin and detection.

Understanding "Size"

We sometimes hear CIOs and others claim that their actual to estimated effort and schedule variance is only 5 or 10% - try as we may to keep a straight face when we hear that, we usually grin in spite of ourselves. Independent industry evidence (e.g., the Standish Group Chaos reports) consistently indicates average project overruns are more like 100%. The secret ingredient to reconcile these divergent perspectives is a measure of SIZE – as estimated/promised, as delivered.

Meaningful assertions about variance MUST include normalization for changes in size/scope – a project that cost twice as much as planned, but delivered twice as much functionality, should be understood to have had zero variance – similarly, an project that is on budget but delivered 50% of the intended functionality should be understood to have had a 100% variance. Very few organizations understand real variance today.

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

Putting it All Together – An Effectiveness Dashboard

These effectiveness indicators are used in three perspectives: Baseline (values when we start), Trend (changes in values over time, reflecting aggregate impact of all interventions), and Pre-Post Intervention (to reflect the impact of a specific change or improvement under reasonably controlled conditions, making an effort to isolate individual effects).

- Appraisal Cost per Defect by Phase and Appraisal type
 - By project and in aggregate
- Rework Cost per Defect by Phase and Appraisal type
 - By project and in aggregate
- Value Added, Appraisal, and Rework % of Effort
 - By project and in aggregate
- Defect Containment Effectiveness
 - By project and in aggregate
- Total Containment Effectiveness
 - By project and in aggregate
- Effort Variance Normalized for Size
 - By project and in aggregate
- Schedule Variance Normalized for Size
 - By project and in aggregate
- Defect Density (Defect per Size) – total "insertion" rate
 - By project and in aggregate
- Effort per Size ("productivity") – note: dangerous ground – must consider variations in "schedule pressure"
- Duration per Size ("cycle time") – note: dangerous ground – must consider variations in "schedule pressure"

Based on experience with sustained application of these metrics it is typically possible for an organization to shift 10-20% of "non-value added" work to Value Added within 1 to 2 years. It's not easy, but the payoff potential is VERY large.

Managing the "Politics" of Implementation

Every organization that attempts improvement based on measurement invariably runs up against resistance. Often that is written off to "politics" and dismissed as an impossible problem to solve – it's just too hard - let's just give up.

Sometimes it's useful to reflect for a moment on the fact that the word "politics" has the same root as the word "polite", and to realize that resistance is frequently a consequence of a lack of careful, tactful, consideration of potential impacts of measurements on those they measure. Imposing any set of metrics, however valid or well intentioned, is almost always a recipe for disaster – understanding and trust must come first.

While every organization is somewhat different, there are perhaps some common themes that facilitate successful implementation of a cost of quality framework in most instances:

- Step 1: Secure an executive sponsor. Sorry, but you'll never get anywhere trying to drive measurement from below or beside those who will collect and use the data. You MUST

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

have executive sponsorship. Worse yet, your executive sponsor must "get it" and be prepared to be patient and supportive over an extended period. Cost of Quality (or any other measurement system) is NOT a silver bullet – it will take time to get real valid results. You need your executive sponsor to communicate her/his sponsorship, intentions, and understanding. Something like the following may work:

To: All IT Colleagues
From: CIO
Re: Cost of Quality

As all of you are aware, our business partners expect continuous improvement in everything we do – we've all heard the "better, faster, cheaper" mantra. We have for some time been searching for a way to measure and demonstrate our improvement that is common across the organization, and have settled on a relatively simple approach known as "Cost of Quality". I believe this approach will work in all areas of IT. Successful implementation of this approach will require your cooperation and good will. I want to assure you that this program will be used in a strictly constructive way – not to find fault with anyone, but to help all of us improve our performance and demonstrate that improvement to our business partners.

Over the next 2 months the quality team will be conducting a series of facilitated sessions designed to develop the details necessary to implement this program in a fair and practical manner. You will be fully informed about the objectives and processes involved and your input will be solicited and will have a major influence on how we execute this.

In broad outline our plan and schedule for implementation of this program will be approximately as follows, subject of course to adjustments if needed:

- Months 1 and 2: Facilitated sessions to provide education on the concepts and goals of the program and to solicit input. The result will be a set of "operational definitions" of the data to be collected and the uses to be made of that data.
- Months 3 - 6: Preparation of data collection methods, data validation processes and any necessary tools, including one or several pilots.
- Months 7 - 12: Collection and validation of baseline data. No actions will be taken on this data, but my expectation is that each group manager will be prepared to indicate at the end of this period that they are confident they have accurate baseline values for the data collected. It is the responsibility of each manager and group to ensure they have an accurate baseline at the end of the baseline period.
- Each group manager, together with his or her team, will be expected to define specific improvement targets for Cost of Quality metrics, with the overall intention that each group will improve their "Value Added" contribution relative to the baselines established during the startup period. These goals will be incorporated in all managers and groups MBO goals for year 2 and beyond.

- Step 2: As suggested by the CIO's memo, spend time educating all concerned on the basic definitions. Allow each group to develop specific examples in their current work. Be prepared to answer some difficult questions and to develop precise yet simple "operational

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

definitions" that are widely accepted. Make sure everyone has a clear understanding of what the data mean before anything is actually collected

- Step 3: Work with those involved to define how data will be collected, recognizing that the overhead involved must be reasonable to have any chance of success. It is ESSENTIAL that you build in data validation from the start – Caper Jones estimates that only 55% of actual project effort is typically included in time accounting systems, excluding factors such as unpaid overtime, manager's time, QA time, DBA time, etc. Perhaps you can live with data that is 80% accurate, but 55% is certainly not good enough. If you don't have a continuous validation process you can be sure the data will not be accurate. Make the local managers responsible for accuracy – after all, they are going to be expected to demonstrate improvement in year 2, so the old "the data isn't accurate" excuse must be off the table.
- Step 4: Pilot the data collection and validation processes and tools – be prepared to make changes to iron out difficulties. Keep it simple.
- Step 5: Collect the data. Publish frequent feedback on the facts you have found. Expect questions and challenges. Prepare an educational program to provide illustrations of how to interpret the data and to provide some ideas on improvements that might be used to drive improvement during year 2. Best results invariably come from initiatives that originate among those "being improved" – it helps to educate them on possibilities, but best for them to make their own choices within the frame of overall objectives set by executive management – tell them what you expect them to achieve, not how to get it done.
- Step 6: Initiate improvement efforts and provide data to show the impact during year 2. (See part 2 below for examples.)

No doubt many will want to make this happen faster. Many have tried, few have succeeded – this is a race the tortoise nearly always wins.

Part 2: Using Cost of Quality Metrics to Drive Software and IT Improvement

This part will illustrate how CoQ metrics might be used to drive improvement. We will use a "case study" of the notorious outsourcing group, SnidelySoft. Snidely management decided 18 months ago to differentiate from competition and gain competitive advantage by aggressive application of Lean Six Sigma. To support that they initiated time accounting using the Cost of Quality framework, combined with defect tracking.

Snidely now has 12 months of solid data for each of their three primary divisions, covering a total of 20-30 projects for each. Table 1 summarizes averages of the key effort related metrics for each division, by life cycle phase (including the first 3 months after delivery).

Table 1: Effort

Metric	Division		
	Peaches	Pears	Plums
Requirements Total effort	4.4%	11.1%	15.3%
Value Added % of effort	100%	100%	85%
Appraisal % of effort	0%	0%	10%
Rework % of effort	0%	0%	5%
Design Total effort	5.5%	12.5%	17.8%
Value Added % of effort	100%	100%	88%
Appraisal % of effort	0%	0%	8%

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

Rework % of effort	0%	0%	4%
Build Total effort	15.4%	15.3%	25.4%
Value Added % of effort	92%	93%	78%
Appraisal % of effort	5%	4%	14%
Rework % of effort	3%	3%	8%
Test Total effort	28.0%	30.6%	26.3%
Value Added % of effort	-	-	-
Appraisal % of effort	72%	73%	78%
Rework % of effort	28%	27%	22%
Post-Release Total effort (3 months)	46.7%	30.6%	15.3%
Value Added % of effort	-	-	-
Appraisal % of effort	-	-	-
Rework % of effort	100%	100%	100%
Grand Total Effort	100%	100%	100%
Value Added % of effort	24%	38%	48%
Appraisal % of effort	28%	23%	27%
Rework % of effort	48%	39%	25%

Table 2 summarizes defect related data.

Table 2: Defects

Metric	Division		
	Peaches	Pears	Plums
Requirements			
Containment Effectiveness (%)	0%	0%	62%
"Find" hours per defect (hours)	-	-	.75
"Fix" (Rework) hours per defect	-	-	.25
Design			
Containment Effectiveness (%)	0%	0%	48.7%
"Find" hours per defect (hours)	-	-	1.6
"Fix" (Rework) hours per defect	-	-	.6
Build			
Containment Effectiveness (%)	8.4%	10%	52%
"Find" hours per defect (hours)	3.3	3.2	3
"Fix" (Rework) hours per defect	1.1	1.2	1.1
Test			
Containment Effectiveness (%)	62.6%	78%	82.6%
"Find" hours per defect (hours)	20	18.8	19.2
"Fix" (Rework) hours per defect	6	6.1	5.8
Post-Release (3 months)			
Containment Effectiveness (%)	71.6%	80.2%	91%
"Find" hours per defect (hours)	-	-	-
"Fix" (Rework) hours per defect	42	43.7	38.4

Summary Observations

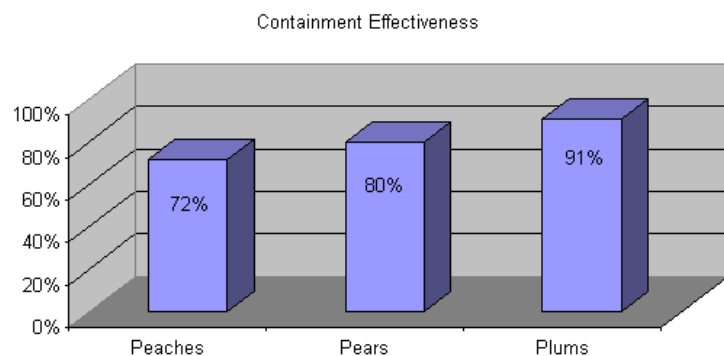
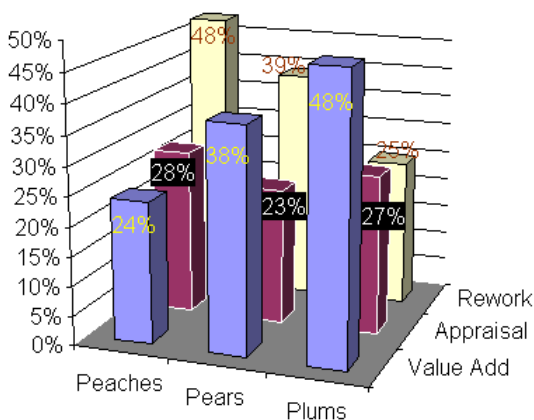
- Effort required to find and fix defects in a given phase does not vary significantly across divisions.

© 2007 Process-Fusion.net

www.Process-Fusion.net ♦ info@Process-Fusion.net

+1 (904) 579-1894

- Effort to find and fix defects increases significantly in all divisions as they progress through the life cycle.
- Peaches devoted a significantly lower percentage of effort to early phases than did the others. They also devoted less effort to Appraisals prior to the test phase, delivered the lowest quality (post-release containment effectiveness), had the lowest overall Value Added % and highest Rework %.
- Pears devoted relatively more effort to Requirements and Design than did Peaches, but also relied primarily on testing to remove defects. Their results in terms of delivered quality and total Value Added were better than Peaches, but significantly below Plums.



Moving From Data to Action

It's very obvious from this data that the Plums are far ahead in the effectiveness race, but why is that? What actions should Peaches and Pears take to close the gap? What can Plums do to keep their lead?

Peaches

- Devote more time to Requirements and Design – put the cowboy hats back in the closet and slow down at the beginning in order to finish fast – use this data to convince your management and customers that you need to devote more time up front, and you need more of their time as well to get the requirements right.
- Devote more time to Appraisal earlier in the life cycle – higher cost per defect later in the life cycle means there is a lot of leverage in devoting time to Appraisal efforts such as inspections earlier in the life cycle.

Pears

- Time distribution to Requirements and Design looks pretty good, but devote more time to early Appraisals.

Plums

- Focus on improving the effectiveness of Appraisal efforts – improve containment rates in order to reduce overall cost per defect
- Experiment with devoting more effort to Appraisals early in the life cycle by carefully monitoring effort required to find a defect in each phase – find the point of diminishing returns for each appraisal method in each phase (how much appraisal is "enough" in each phase?)

Some of you may be thinking, "ok, but we've know that for years – why do we need metrics?" It's true we've known these are actions we should take for a very long time – *it's also true that most organizations, and the industry as a whole, haven't acted on this understanding!*

The real value of metrics like those proposed here is that they help us move beyond understanding to ACTION – they help us convince management and customers that changes in processes and time allocation really do benefit everybody. "Knowing" is not enough – we need "proof".