When Is It Cost Effective to Use Formal Software Inspections?

Bob McCann
Lockheed Martin Integrated Systems and Solutions

The purpose of this article is to present a way quantitatively to determine the parametric limits to cost effectiveness of software inspections based on a previously published model. This analysis leads to the conclusion that it is cost effective to inspect both original code and most modifications to the code after initial coding. Any exceptions should be carefully considered based on quantitative analysis of the projected impact of the exceptions. Further, any proposed substitution for rigorous inspections should be carefully evaluated for cost effectiveness prior to replacing or modifying the process.

In this author's experience, the two following issues are discussed qualitatively when program management decides what to inspect and what data to collect:

- Deciding whether or not to inspect work products based on a qualitative understanding of various limiting parameters such as work product size, preparation rate, or expected defect density.
- Deciding whether or not to collect and to analyze inspection data based on a qualitative understanding of the return on investment (ROI) on collecting, analyzing, and using inspection metrics.

This article improves on this practice by estimating quantitatively, from the perspective of an existing quantitative cost model [1], both the parametric boundaries of inspection cost effectiveness and ROI for collecting and analyzing inspection metrics.

It is important to note that there are several different budgetary strategies that may be applied when making process implementation choices. Here are three examples ordered from long term to short term in the planning horizon:

- Minimize total cost of ownership, including post delivery maintenance costs.
- Minimize overall development cost excluding post delivery maintenance costs.
- Assure that inspection overhead costs are exactly balanced by reductions in test rework costs (this does not minimize costs).

These goals may be examined within the context of the existing quantitative cost model. Then, once the inspection process is quantitatively managed, it becomes possible to study the value of various emerging best practices in inspections (see Appendix A).

Although other process models and cost effectiveness analyses are possible, the results are similar [2, 3]. It is possible and reasonable quantitatively to analyze the cost effectiveness of inspections with respect to the model parameters.

The Cost Model

If sufficient metrics are collected during a software development project, it is possible to know the cost structure of the various development processes. That in turn enables modeling of the cost impact of all processes that discover software defects. In particular, the previous article focused on modeling of the impact of software code inspections on the overall cost of developing software [1]. This is an example of how you might model the impact of all quality practices that detect defects. All such practices are data linked to software testing via the defects that escape into the test phase. Indeed you could use the code inspection cost function directly on other work products with only minor reassignment of the meanings of the cost parameters; just compare the cost of finding and fixing a defect with an inspection to the cost of finding and fixing that same defect by another means later in the development lifecycle.

A central result of the previous article is the total cost function for software code inspections plus software testing. For this purpose all software regression testing and rework of test defects were lumped into a single term. That cost function follows:

(total cost) = (inspection fixed cost) + (regression fixed cost)
+ (inspection preparation plus meeting cost)
+ (inspection rework cost)
+ (test rework cost plus regression cost)

or

$$T_c = C_i + C_r + D*S/R + W_{ri}*I*S*[1-(-m)*R] + T_r*E_r*I*S*(-m)*R, \quad (1)$$

Note: m<0, so (-m)>0, (see the sidebar "Symbol Definitions.")

This formula includes two terms derived from regression analysis of inspection data. The first such term, *D*S/R*, combines the inspection preparation labor with the meeting labor because they were found to be co-linear. *D* is the regression coefficient and *S/R* is the preparation labor per inspection (code size *S* divided by the preparation rate *R*). The second term includes the linear regression model for inspection effectiveness as a function of preparation rate, $W_{ri}*I*S*[1-(-m)*R]$, where W_{ri} is the average cost of fixing a defect found in an inspection, I is the density of defects found by all means (this increases linearly throughout the lifecycle), and *m* is the linear regression coefficient relating inspection preparation rate and effectiveness (fraction of discovered defects found by inspection).

The last term models the testing cost due to defects that escaped into the test process because they were missed by the inspection, $T_r*E_r*I*S*(-m)*R$, where T_r is the average cost of finding and fixing one defect in testing, E_r effectiveness of the testing process at discovering defects, *I*S*(-m)*R* is the number of (eventually) discovered defects not found by software inspection.

Method

For each model parameter there may be a corresponding boundary value that distinguishes cost effective process operation from that which is not. In each case, you compare the inspection costs with the test costs that may be prevented by conducting inspections. When the benefit exceeds the cost, it is cost effective to conduct inspections. You may calculate the ROI by subtracting the cost from the benefit and dividing the result by the cost. Positive ROI justifies conducting inspections.

The original model is based on analysis of inspection, test, and cost data for a proprietary project at Lockheed Martin Management and Data Systems. The data used in this paper are simulated data having the same characteristics as the real data; this substitution is required to protect the proprietary nature of the original program data and does not change the conclusions that flow from the model analysis.

Results

The cost model is reviewed and extended to include post delivery marginal maintenance costs. Then a cost effectiveness model is developed and applied to several process model parameters: inspection preparation rate, inspection package size, defect insertion density, the significance of inspection effectiveness correlation, and the ratio of total effort to preparation effort. The model produces a simple inequality that, if true, predicts inspections to be cost effective.

Cost Minimization

Once the decision has been made to inspect a work product, that inspection should be managed so that it minimizes the total applicable cost function. If regression testing is not 100

percent effective, and minimizing the total cost of ownership is the goal, then equation (1) needs cost terms to model the cost of fixing defects found after delivery to the customer: $C_R + T_R*E_R*(1-E_r)*I*S*(-m)*R$ where it is expected that $T_R > T_r$. This results in a linear interpolation of the probable costs in testing and customer use:

$$T_c = C_i + C_r + C_R + D*S/R + W_{ri}*I*S*[1-(-m)*R] + T_{eff}*I*S*(-m)*R \qquad (2)$$

where

$$T_{eff} = [T_r*E_r + T_R*E_R*(1-E_r)].$$

In either case, the same mathematical procedure is used to find parameter values that minimize the cost function.

Preparation rate is the only model parameter that causes the cost function to achieve a cost minimum, and it does so in the region of applicability of the linear regression fit for inspection effectiveness. This regression fit model is thus bounded between zero and one (100 percent effective inspections):

$$0 <= [1-(-m)*R] <= 1.$$

Substituting the model data from the previous article [1] gives:

$$0 < R <= 1/(-m) = 1/0.00075=1329 \text{ SLOC/Labor Hour (LH)}.$$

If the preparation rate approaches either boundary, a non-linear regression model will be more appropriate, provided sufficient data exist to validate the model.

Product size *S* and defect insertion rate *I* both have a positive slope in the cost function (the slopes are given by derivatives with respect to model parameters):

$$dT_c/dS = D/R + W_{ri}*I*[1-(-m)*R] + T_{eff}*I*(-m)*R > 0$$

and

$$dT_c/dI = W_{ri}*S*[1-(-m)*R] + T_{eff}*S*(-m)*R > 0.$$

Clearly, both *S* and *I* drive up costs linearly, so any scheme that reduces either total size or defect insertion rate will drive down development costs provided size and defect insertion rate are independent – more about that later.

The effect of the slope of the regression fit *m* is determined by the relative effect of the cost of finding and fixing defects at various points in the development lifecycle:

$$dT_c/d(-m) = T*I*S*R > 0$$

where

$$T = (T_{eff} - W_{ri}).$$

It is expected that this will be strongly positive. Therefore, any process change that improves the effectiveness of finding defects at higher preparation rates will reduce costs. For instance, automating some of the checks in the inspection process reduces the labor spent finding those classes of defects.

The minimum process cost occurs at a preparation rate for which the slope of the cost function is zero. Thus the cost optimizing preparation rate is found by setting to zero the first derivative of the cost function with respect to preparation rate:

$$dT_c/dR = -D*S/R^2 + T*I*S*(-m) = 0$$

$$R^{*} = \sqrt{D/[T * I * (-m)]} \quad (3)$$

where

$$T = (T_{eff} - W_{ri})$$

and

$$T_{eff} = [T_r * E_r + T_R * E_R * (1-E_r)].$$

Cost Effectiveness

With a cost effectiveness strategy, it is necessary to derive an expression that shows how inspections can be expected to pay for their own overhead assuming, on average, that they will be conducted at the cost optimizing preparation rate. For this example, the cost of finding and fixing defects found by the customer is not included. This expression can be derived by comparing the average cost of finding a defect in an inspection with the associated cost savings in reduced test rework and regression testing. The cost payback is given by the following condition:

(Inspection Costs) <= (Regression Test and Rework Costs Prevented)

or

$$C_i + D*S/R + W_{ri}*I*S*[1-(-m)*R] <= C_r/N + T_r*E_r*I*S*[1-(-m)*R], \quad (4)$$

or, presuming the inspections will be conducted at the optimum preparation rate,

$$(C_i - C_r/N) + D*S/R^{*} <= (T_r*E_r - W_{ri})*I*S*[1-(-m)*R^{*}]. \quad (5)$$

The right hand side of equation (5) is the average test rework cost per defect prevented minus the average inspection rework cost per defect found times the number of defects found by the inspection, *I*S*[1-(-m)*R*]*. C_i is the average overhead cost per inspection, C_r is the average overhead cost per test rework and regression test cycle, and *N* is the average number of inspections bundled per regression cycle. Given equations (4) and (5), it is now possible to investigate the parametric boundaries or *clip levels* of cost effectiveness with respect to the cost model parameters: preparation rate, inspection size, defect injection rate, repair costs, and inspection effectiveness regression coefficient. By making the following substitutions:

$C = C_r/N - C_i$	(net average fixed costs)	
N	(average inspections per regression suite)	
$T = T_r*E_r - W_{ri}$	(net average repair costs)	
$R^{*} = \sqrt{D/[I*(-m)*T]}$	(optimum preparation rate)	(3)
$X = \sqrt{D*(-m)}$	(regression coefficients)	
$Y = \sqrt{I*T}$	(Y^2 is the net burden rate for missed defects)	

the governing relationship, equation (5), for the limits of cost effective operation may be simplified:

$$2*X <= Y + C/(S*Y). \quad (6)$$

The following sections will explore the implications of equations (4), (5), and (6) for each of the model parameters.

Cost Balancing Preparation Rate

Solving equation (4) for *R* gives the preparation rate that assures payback of overhead costs in reduced regression testing and rework prior to delivery to the customer:

$$D*S - [C + T*I*S]*R + T*I*S*(-m)*R^2 <= 0 \qquad (7)$$

where it is expected that $T=(T_r*E_r - W_{ri})>0$ and $C=(C_r-C_i)>0$. This is of the form $a*R^2 + b*R + c <= 0$ which has two bounding roots (small and large):

$$R_S = -2*c*\text{sign}(b)/(|b| + \sqrt{b^2 - 4*a*c})$$

and

$$R_L = -\text{sign}(b)*(|b| + \sqrt{b^2 - 4*a*c})/(2*a)$$

where

$$a = T*I*S*(-m)$$
$$b = -[C + T*I*S]$$
$$c = D*S.$$

Any preparation rate between these two roots will yield cost effective inspections:

$$R_S <= R <= R_L.$$

The condition for real valued solutions is that $(b^2 – 4*a*c) >= 0$. The two real roots will bound the region of interest – the left hand side of equation (7) is negative. Using the data from [1] and finding C_i as the y-intercept for small inspections [1]:

$$C_r=0,\ C_i=2,\ m=-0.00075,\ I=0.040,\ T_r=20,\ W_{ri}=4,\ D=4,\ S=200$$

and assuming $E_r = 1.0$ gives $R_S = 6.38$ SLOC/Labor Hour (LH), and $R_L = 1306$ SLOC/LH.

By comparison, the cost optimizing preparation rate 91.29 SLOC/LH is equal to the geometric mean of the bounding values:

$$R^* = \sqrt{R_S * R_L}$$
$$= \sqrt{c/a}$$
$$= \sqrt{D/[T*I*(-m)]}.$$

The preparation rate that provides the most negative value for equation (7) is found by setting the first derivative of equation (7) with respect to *R* equal to zero:

$$–(C + T*I*S) + 2*T*I*S*(-m)*R = 0$$

or

$$R_p = (C + T*I*S)/[\,2*T*I*S*(-m)\,].$$

Using the above parameters, $R_p = 656$ SLOC/LH, which is well below $1/(-m) = 1329$ SLOC/LH for regression model validity but well above the overall cost optimizing value 91.29 SLOC/LH. Please note: if $C_i = C_r$ then $R_p = 1/(-2*m) = 664.5$ SLOC/LH, which is the exact center of the region of applicability of the linear regression model.

While the parametric analysis of the cost effectiveness strategy establishes the existence of cost effectiveness boundaries for the inspection process, optimum cost performance is not obtained on the boundaries. Rather the most cost effective operation for the process is obtained when the optimum preparation rate is used. Figure 1 demonstrates this recommended practice.

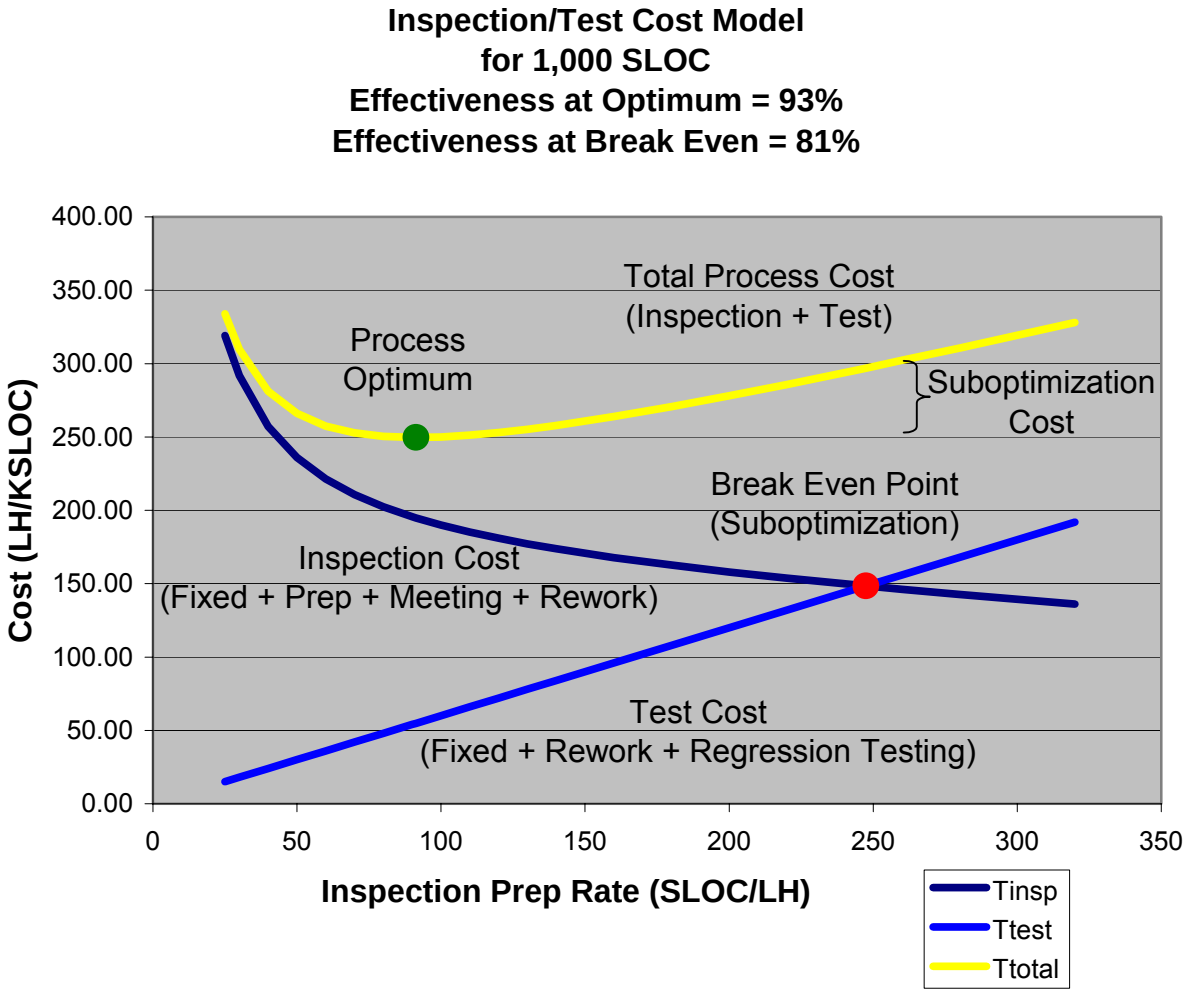

Figure 1: *Inspection/Test Cost Model*

It is worth noting that the inverse of the Total Process Cost function in Figure 1 is the test phase productivity.

Figure 2 displays this function and demonstrates the effectiveness of inspections in controlling test phase productivity.

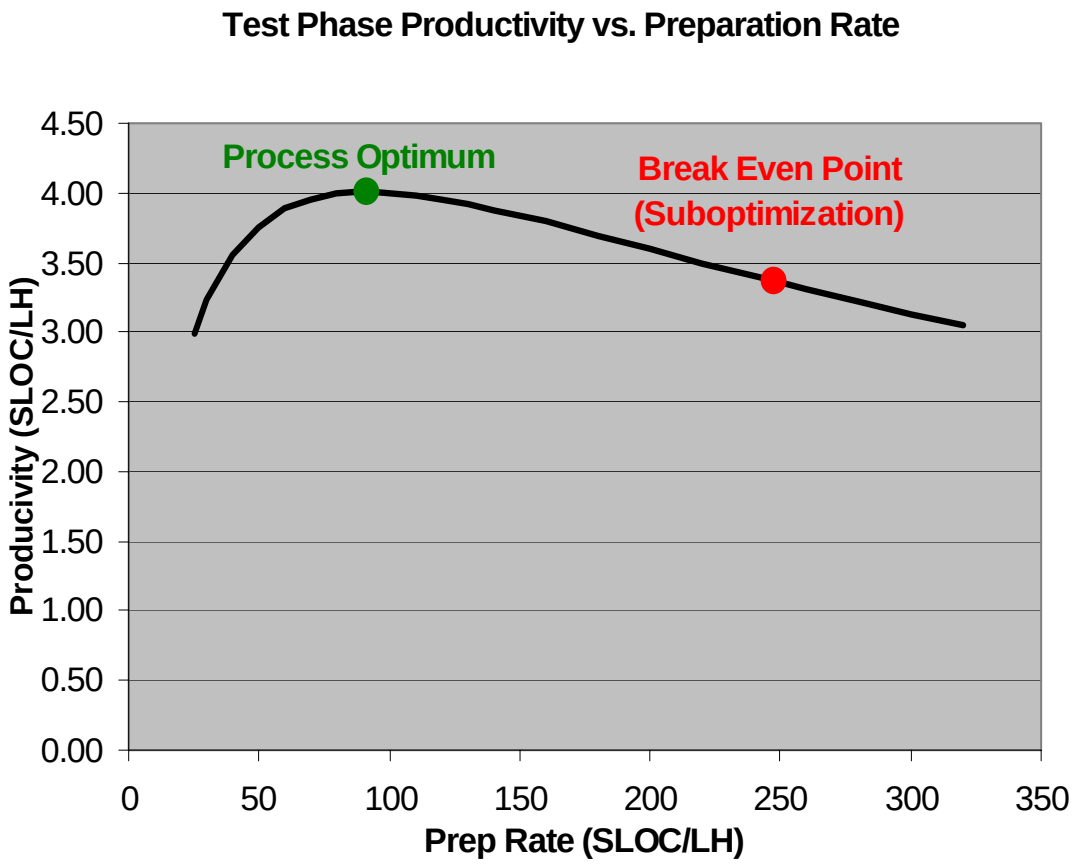

Figure 2: Test Phase Productivity Versus Preparation Rate

The Smallest Size for a Cost Effective Inspection
Solving equation (6) for equality gives:

$$S_c = C/[\ Y*|2*X-Y|\ ] \qquad (8)$$

where $S<=S_c$ when $2*X>Y$ and $S>= -S_c$ when $2*X<Y$.

Please note that there are two singularities: $2*X=Y$, where the net probable marginal rework costs equals twice the sum of meeting plus preparation costs; and $Y=0$, where inspection rework costs equal marginal regression testing and rework costs. Using the above data:

$$C_r=0,\ C_i=2,\ m=-0.00075,\ I=0.040,\ T_r=20,\ W_{ri}=4,\ D=4,\ S=200$$

and assuming $E_r = 1.0$ and using the cost optimizing preparation rate $R*=91.29$ SLOC/LH gives a constraint on size:

$$S >= S_c = 3.62 \text{ SLOC.}$$

That assumes the testing costs through System Level Test (SLT). Including the costs of Operation Release Level (ORL) testing increases the costs significantly to, say, approximately $T_r = 40$ and changes the cut off:

$$S_c = 1.53 \text{ SLOC.}$$

Please note that both values are very small. Very few defect fixes affect so little code. Indeed, if testing fixed costs exceed inspection fixed costs (for the same amount of code), $C_r/N > C_i$, and marginal testing costs prevented exceeds twice the marginal inspection cost incurred, $Y > 2*X$, then $-S_c < 0 < S$, and all code inspection sizes are cost effective!

Defect Density Clip Level
Beginning with equation (6):

$$2*X <= Y + C/(S*Y) \qquad (6)$$

and solving for Y gives two bounding conditions:

$$Y_S = (C/S)/[X + \sqrt{X^2 - C/S}\ ]$$

and

$$Y = \sqrt{T*I}\ > Y_L = X + \sqrt{X^2 - C/S} > Y_S \qquad (9)$$

Using the above simulated project data and the cost optimizing preparation rate gives:

$$I >= \{\ [\sqrt{0.003}\ +\ \sqrt{0.003 + 2/S}\ ]/\sqrt{T}\ \}^2$$

where T = 16 (SLT costs) or 36 (ORL costs).

Figure 3 shows that the defect insertion rate clip level rises for smaller code packages. It is important to know that defect insertion rates increase for smaller packages. Indeed, as demonstrated by Figure 4, the defect rate increases as the size decreases[1]. This is data from the Space Telescope Grant Management System (STGMS), which is a large Java application of over 200,000 non-commented source statements. In the following chart *LN* is the natural logarithm.

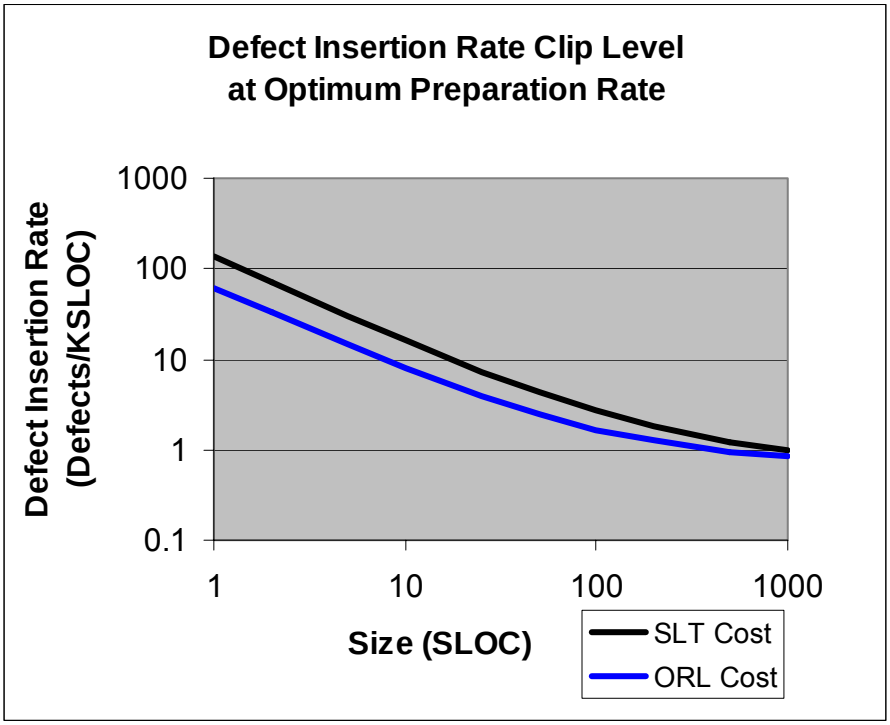

Figure 3: *Defect Insertion Rate Clip Level Versus Optimum Preparation Rate*

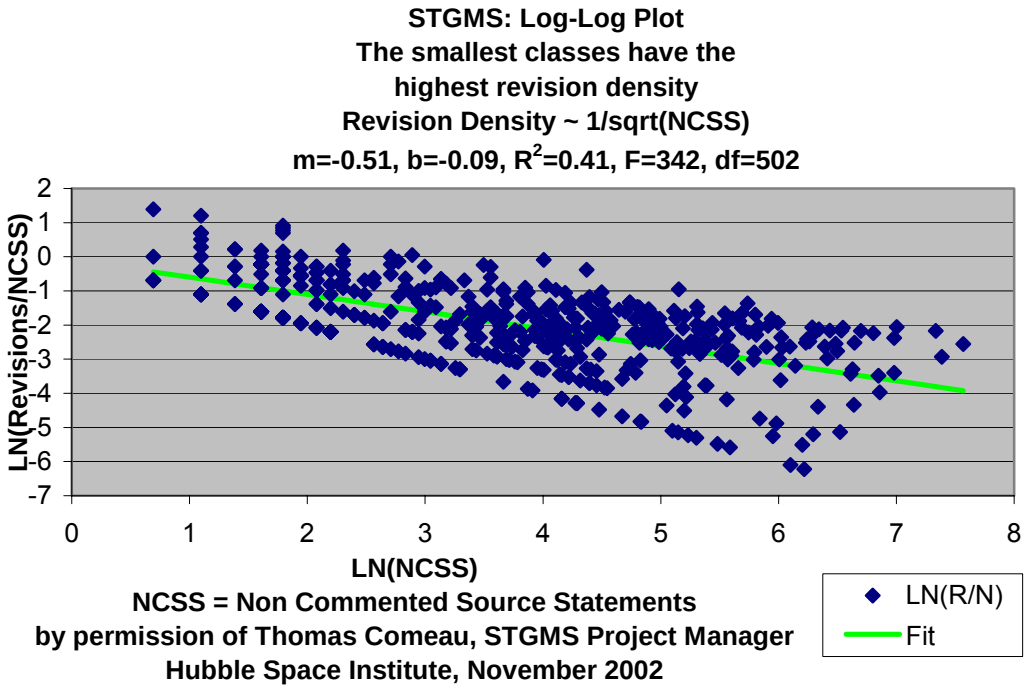

Figure 4: *Space Telescope Grant Management System Revision Density Plot*

This trend is not unique. The same trend may be seen on Figure 5, the public domain Jetty project (a Java server written in Java)[2]. One might hypothesize that the slope of the regression line is explained by the observation that large classes get edited more often than small ones, and each editing session affords opportunity to review all that has been coded to date. In that case the effectiveness of defect detection by the author would be proportional to the number of opportunities to review the modules in the class. The total number of review opportunities (and the effort to build the class) would scale as the square of the number of modules. Hence the defect density would be roughly proportional to $1/\sqrt{\text{Size}}$ if the modules are of similar size and complexity. The variance in the scaling would be sensitive to the variation in the review habits of the individual developers. As such, both defect detection and defect prevention would be subject to potential process improvement and training. In the following chart *LN* is the natural logarithm.

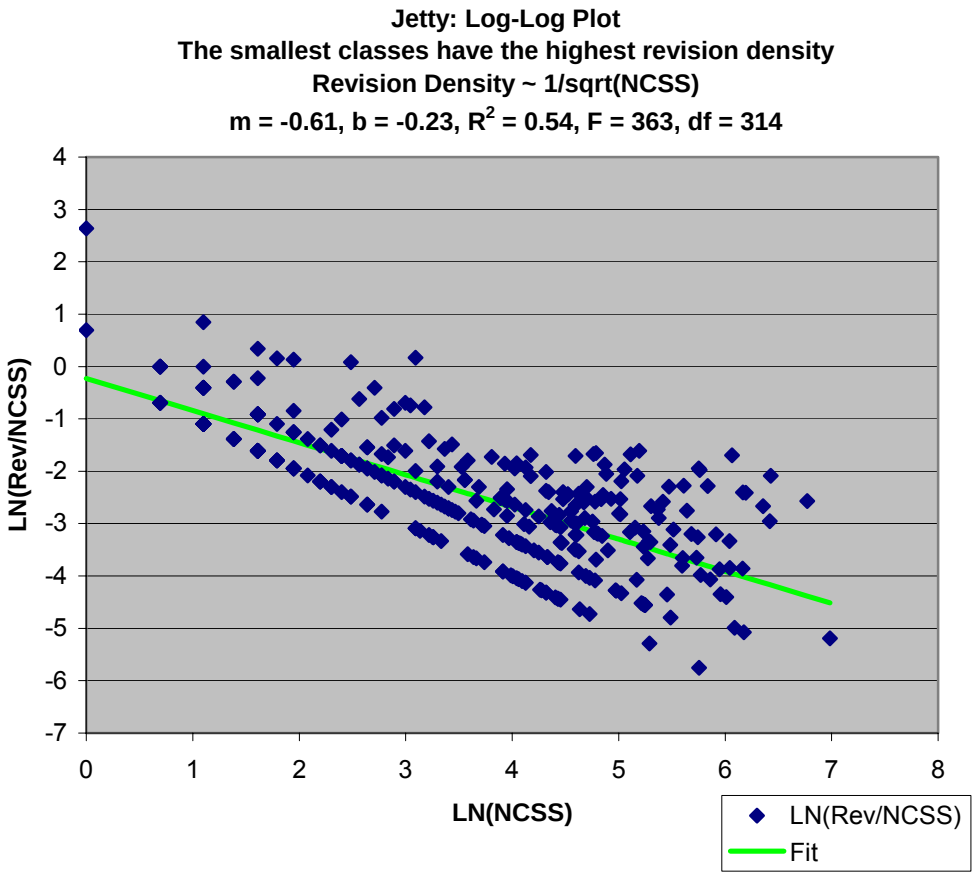

Figure 5: *Jetty Revision Density Plot*

A further consideration is that reworked code is known to have a significantly higher defect insertion rate than initially written code, especially if the maintainer is not the original author. From Capers Jones [4], out of a total average defect load of five defects per function point, 0.4 defects/Function Point is contributed from bad fixes. That says that there is about an 8 percent probability of making a bad fix. That is a defect insertion rate of 80 defects/thousand single lines of code (KSLOC) in code repairs. This is significantly higher than the defect rate for new code that Capers Jones lists as 50 defects/KSLOC. From these charts it is very clear that inspections are justified for even small changes unless the regression cost is very low indeed. Post delivery, the regression costs tend to be even greater, making the case even stronger for inspections.

Clip Level for Net Cost to Fix

One can also solve equation (9) for the net cost to fix a defect:

$$Y = \sqrt{T * I} > Y_L = X + \sqrt{X^2 - C/S} \qquad (9)$$

thus

$$T >= \{[X + \sqrt{X^2 - C/S}\,]\}^2/I.$$

Using the example data generates Figure 6:

Just as with defect density, one can solve equation (6) for the critical linear regression coefficient *(-m)* above which inspection costs dominate regression test costs at the optimum preparation rate:

$$2^* \sqrt{D * (-m)} = 2^*X <= Y + C/(S^*Y). \quad (6)$$

Solving for *(-m)* yields

$$(-m) <= (-m_c) = [Y + C/(S^*Y)]^2/(4^*D). \quad (10)$$

Using the simulated program data gives the following figure:

$$(-m_c) = 0.0189 \text{ for } S=10 \text{ (small changes).}$$

In the example data *(-m)* = 0.00075, which is much less than the critical value. So, inspections are well justified even for small changes. However, there is a size cut off that causes the numerator of equation (10) to become zero:

$$S_c = -C/Y^2$$
$$S_c = 3.1 \text{ (SLT costs)}$$

or

$$S_c = 1.4 \text{ (ORL costs).}$$

Please note that *(-m)* must also be statistically distinct from zero at some acceptable confidence level, say α/2=0.025 for a two tail t-test. For this example, the standard error in *(-m)* is 0.0000387, and *t-critical* is 2.3646. The minimum significant value for *(-m)* is 2.3646*0.0000387=0.0000915, well below the actual value.

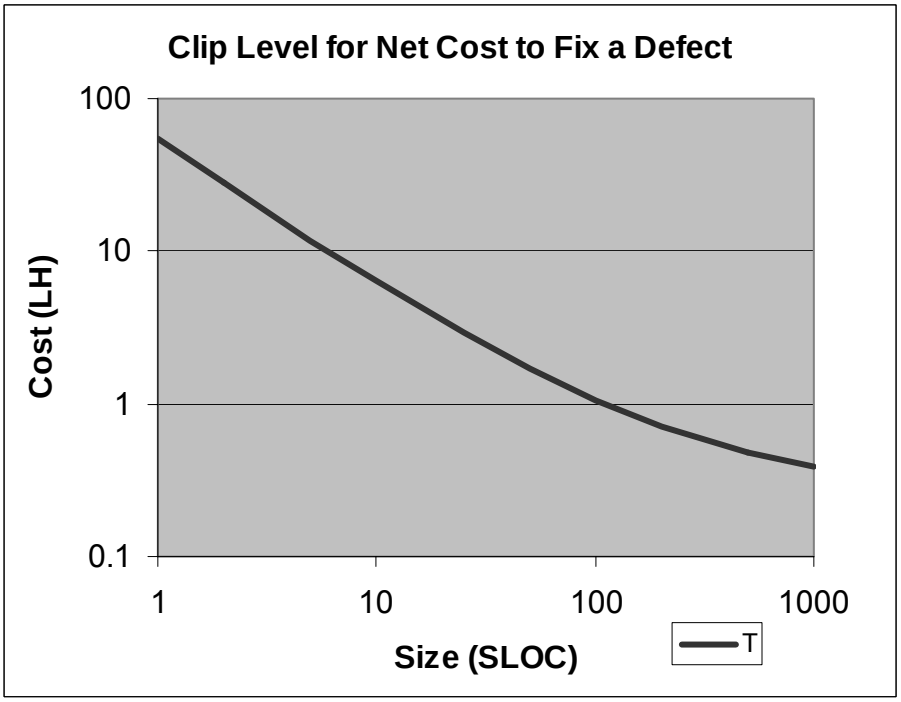

Figure 6: *Clip Level for Net Cost to Fix a Defect Critical Regression Coefficient*

Critical Meeting Rate
By rearranging equation (10) it is possible to specify the meeting rate above that it is not cost effective to conduct inspection meetings:

$$D <= D_c = [Y + C/(S*Y)]^2/[4*(-m)]. \quad (11)$$

For the example data, D_c = 101 for S = 10 and SLT costs and D_c = 356 for ORL costs. These are well above the example value D = 4, so time spent in the meeting does not constrain the cost effectiveness of the process.

Return On Investment for Metrics Analysis – Knowing the Optimum Preparation Rate
First calculate *Total Process Cost* less *Cost at Optimum* evaluated at the unmanaged preparation rate:

$$T_M = S*D*(1/R – 1/ R^*) + S*I*(-m)*T*(R - R^*).$$

Given the simulated data

$$C_r = 0,\ C_i = 2,\ m = -0.00075,\ I = 0.040,\ T_r = 20,\ W_{ri} = 4,\ D = 4$$

with project teams averaging approximately S= 600 SLOC at R= 300 SLOC/LH, and R^* = 91.29 SLOC/LH, the excess process labor at the unmanaged preparation rate is T_M= 41.8 LH/inspection.

So, on average, using the overhead cost of setting up, managing, recording, and analyzing an average inspection (assuming a fair amount of automation and tool support), C_i = 2 LH, the ROI for collecting and analyzing metrics for an average inspection is substantial[3]:

$$ROI = (T_M-C)/C = (41.8-2)/2$$
$$ROI = 20.$$

In this case, on average, every hour spent collecting, recording and analyzing inspection metrics will potentially return about 20 hours in reduced testing and fixing cost. The ROI will be even higher if one includes the added benefits of using the metrics to support defect causal analysis and associated process improvement to prevent future defects of a similar type from occurring [5, 6]. Clearly, it is critical to collect and to analyze the inspection data and to manage the preparation rate accordingly.

Discussion and Recommendations
Within the context of the existing cost model, one can answer the initial question in the title of this article. The parametric boundaries of cost effective operation have been derived and shown to impose relatively benign operational constraints. Further, it has been demonstrated that the return on investment for collecting and analyzing inspection metrics well justifies doing so.

It is well known that the cost of finding and fixing defects increases throughout the development life cycle. Since code is the last development phase prior to testing, and since the cost to fix artifacts increases throughout the development life cycle [7], and since it is reasonable that the cost to fix a test script would be similar to the cost to fix a code defect, it is therefore clear that earlier work products are even more cost effective to inspect than code.

Therefore, because it is cost effective to do so, it is established that performing rigorous inspections should be the default behavior for all software work products that affect the quality

of the delivered code provided there is a documented development process with defined work products in addition to code. It is expected that any organization with an ISO-9001 certification or a rating above Capability Maturity Model® Integration Level 2 would have documented development processes and defined work products that affect the quality of delivered code.

Exceptions to this rule should be granted only after careful quantitative analysis of the cost impact of such exceptions using equations (3) and (6):

$$R^{*} = \sqrt{D/[I*(-m)*T]} \quad (3)$$

$$2*X <= Y + C/(S*Y). \quad (6)$$

where

$X = \sqrt{D*(-m)}$	(regression coefficients)
$C = (C_r + C_R)/N - C_i$	(net average fixed costs)
N	(average inspections per regression suite)
$T = T_r*E_t + T_R*E_R*(1-E_t) - W_{ri}$	(total net average marginal repair costs)
$Y = \sqrt{T*I}$	(Y^2 is the net burden rate for missed defects)
S	(size of inspected product – SLOC, etc.).

Given that inspections are run at (or near) the optimum preparation rate, then the model makes predictions about average achievable inspection effectiveness:

$$E^{*} = 1 - X/Y \quad (12)$$

and average test phase productivity:

$$P_r^{*} = S/T_c = 1/\{ [(C_r + C_R)/N + C_i]/S + I*W_{ri} + 2*X*Y \}. \quad (13)$$

Finally, it is recommended that any proposed substitution for formal inspections should be carefully evaluated for cost effectiveness prior to replacing or modifying the existing process by deriving an equivalent cost model and performing equivalent analysis.

Acknowledgements

The author would like to thank Thomas Comeau, manager of the STGMS project, for providing the software defect data for the Space Telescope Grant Management System and for finding and making available the equivalent data from the Jetty project. Special thanks are due John Gibson of Lockheed Martin Mission Systems for his many reviews and improvement suggestions for this article. Thanks also go to Dr. Louis Blazy of The University of Maryland University College for his review and improvement suggestions. Further special thanks go to Dr. Michael A. Spencer for detailed composition review and resulting suggestions to improve both clarity and impact.

References

1. McCann, Robert T. “How Much Code Inspection is Enough?” Crosstalk July 2001 <www.stsc.hill.af.mil/crosstalk/2001/07/mccann.html>.
2. Paton, Keith A. Cost-Effectiveness of Manual Code Inspection. Proc. of the 11th International Workshop on Software Measurement, Montreal, August 2001.
3. Paton, Keith A. Should You Test the Code Before You Test the Program? Proc. of ESCOM 2001, London, April 2001.
4. Jones, Capers. “Software Cost Estimation in 2002.” Table 3. Crosstalk June 2002 <www.stsc.hill.af.mil/crosstalk/2002/06/jones.html>).

5. Mays, R.G., C.L. Jones, G.J. Holloway, and D.P. Studinski. "Experiences with Defect Prevention." IBM Systems Journal 29.1 (1990).
6. Gale, J.L., J.R. Tirso, and C.A. Burchfield. "Implementing the Defect Prevention Process in the MVS Interactive programming organization." IBM Systems Journal 29.1 (1990).
7. Boehm, Barry W. Software Engineering Economics. Prentice-Hall, 1981: 40 Fig. 4-2.
8. Glass, Robert L. "Inspections – Some Surprising Findings." Communications of the ACM 42.4 (April 1999): 17-19.
9. Radice, Ron. High Quality Low Cost Software Inspections. Andover, MA: Paradoxicon Publishing, Jan. 2002: 14-402-14-403.
10. Radice, et al. One on One Inspections. Proc. 2001 Software Technology Conference, Salt Lake City, UT, May 2001 <www.stc-online.org/stcdocs/stc2001/178/index.htm>.
11. Radice, One on One: 14-416-14-419.
12. McCann, Robert T. "The Efficiency of Using Process Control Charts to Drive the Defect Prevention Process." Software Productivity Consortium's Annual Member Forum, Oct. 1999 <www.software.org/membersonly/forum/1999/proceedings.asp>.
13. Shull, Forrest, et al. "Lab Package for the Empirical Investigation of Perspective-Based Reading." <www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html>.

Notes

1. By permission of Thomas Comeau, manager Space Telescope Grant Management System (STGMS) project. Revision rate is the number of changes per file and is assumed to be representative of the defect rate.
2. The data may be found at <http://sourceforge.net/cvs/?group_id=7322>. The defect density data are not available. It is assumed that the revision density (revisions per file) will have the same character as defect density.
3. This is the ROI just for the metrics collection and analysis using the simulated data. The ROI for the whole process will be different. For instance see any of the following:
   - Don O'Neil. "Setting Up a Software Inspection Program." Crosstalk Feb. 1997 <www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1997/02/softinsp.asp>.
   - Tom Gilb. "Optimizing Software Inspections." Crosstalk Mar. 1998 <www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/03/optimizing.asp>.
   - Don O'Neil. "National Software Quality Experiment: A Lesson in Measurement: 1992-1997." Crosstalk Dec. 1998 <www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/12/oneill.asp>.
4. The company that has achieved repeatable 100 percent effective inspections is TATA Consultancy of India.

Sidebar A:

Symbol Definitions

Please note that subscripts indicate the following:
i refers to inspections.
r refers to rework after testing.
ri refers to rework after inspection.
R refers to regression testing and fixing after delivery.

C_i = Y-intercept of empirical fit of total inspection labor (preparation plus meeting), overhead cost per inspection (labor hours).

C_r = Testing cost overhead per regression test suite, overhead cost per set of test defects (test overhead costs prevented by an inspection) (labor hours).

C_R = Testing cost overhead per regression test suite, overhead cost per set of test defects found by the customer (test overhead costs prevented by an inspection) (labor hours).

D = Slope of empirical fit of inspection labor plus meeting labor vs. preparation labor (dimensionless ratio).

E_r = Test effectiveness (defects found in test/defects found in test plus those found after test completion excluding all defects that cannot be found by testing, e.g., requirements defects when the tests are required to be consistent with documented requirements and the testers are not allowed to challenge the validity of the requirements) (% defects found in this phase).

E_R = Customer test effectiveness (defects found in acceptance test plus those found in user operation/those defects plus estimated remaining defects.) (% defects found by the customer).

I = Discoverable code defect rate = (code inspection defect rate + test defect rate) (defects/SLOC).

LH = Labor Hours (this is much easier to record and to use than monetary cost).

m = Slope of code inspection effectiveness regression line (%/(SLOC/LH)).

R = Code inspection preparation rate (SLOC/labor hour).

R^* = The cost optimizing code inspection preparation rate (SLOC/labor hour).

S = Total SLOC inspected.

$SLOC$ = Non-comment, non-blank, physical lines of code. Any consistently used size measure will work, e.g., executable statements, function points, etc.

T_0 = Labor for testing perfect code (regression testing not needed) (labor hours).

T_c = Total cost (labor hours).

T_M = Total excess cost due to non-optimal inspection preparation (labor hours).

T_r = Labor per defect to do test rework and regression testing (labor hours/defect).

T_R = Labor per defect to do rework and regression testing of defects found by the customer and user community. (labor hours/defect).

W_{ri} = Labor hours to rework a code inspection defect (labor hours/defect).

Term Definitions

C_i+D*S/R = Labor for inspection preparation plus the inspection meeting.

$I*S$ = Defects present at code inspection.

$(1+m*R)$ = Code inspection effectiveness.

$I*S*(-m)*R$ = Defects missed by the code inspection that escape into test.

$I*S*(1+m*R)$ = Defects found by the code inspection.

S/R = Inspection preparation labor.

T_0*S= Total labor for testing perfect code (regression testing not needed).

$T_r*E_r*I*S*(-m)*R$ = Labor to do test rework.

$T_R*E_R*(1-E_r)*I*S*(-m)*R$ = Labor to fix defects found after delivery to the customer.

$W_{ri}*I*S*(1+m*R)$ = Labor for reworking defects found during the inspection.

Appendix A

Even though the practice has existed since the mid 1970's, performing rigorous software inspections (at all) has been and continues to be an industry best practice [8]. Nevertheless, software inspections have room for improvement. Radice claims that 100 percent effective inspections are becoming more common and may some day become much more so[4] [9, 10]. There are several best practices that may make significant contributions to that quest [11]:

- Use of a Quantitative Cost Model to manage the inspection process.
- Managing the inspection process via statistical process control (SPC) on preparation rate.
- Using a defect prevention process in conjunction with inspections to improve both product and process (check lists, tools, etc.).

- Use of SPC on both the preparation rate and the defect rate to trigger use of the defect prevention process [12].
- Use of Tools to automate parts of the inspection.
- Use of Perspective Based Reading techniques to improve the effectiveness of the inspections [13].
- Use of pilot studies to measure the effect of proposed process improvements.
- Use of experimental design techniques to improve the cost effectiveness of pilot studies of proposed process improvements.

Given a validated model for the cost effectiveness of inspections, it is possible to compare the regression line for cost effectiveness of proposed variations to the regression line for the cost effectiveness of the initial process. Standard statistical tests at a predetermined confidence level can then be applied to the difference thus creating an objective standard for determining the cost effectiveness of the process variation.

About the Author

Bob McCann is a staff systems engineer at Lockheed Martin Management and Data Systems in Gaithersburg, Md. He is currently an Institute of Electrical and Electronics Engineers' Certified Software Development Professional and has nearly 20 years of experience in computational physics and high performance computing including nine years at Princeton Plasma Physics Laboratory working in the U.S. Department of Energy-controlled fusion program, as well as about 10 years experience in design and development of relational databases of various kinds. McCann is currently a member of the Lockheed Martin M&DS Metrics Process Steering Committee and works on improving software development processes, methods, and metrics.

He has a Bachelor of Arts in physics with a concentration in mathematics from Shippensburg University, a Master of Science in physics from University of Maryland, a Master of Science in computer science from Southwest Texas State University, and is working on a Master of Science in computer systems management/software development management at the University of Maryland University College.

Lockheed Martin Management and Data Systems
700 North Frederick Avenue
Gaithersburg, MD 20879
Phone: (301) 240-4273
Fax: (301) 240-7190
E-mail: bob.mccann@lmco.com