

# How Much Code Inspection Is Enough?

Robert T. McCann

Lockheed Martin Management and Data Systems

*Given code inspection effectiveness (defects found during inspection/defects found during inspection plus those found during test) as a function of preparation rate (amount of code examined per labor hour), it is then possible to construct a simple cost model that predicts testing labor hours as a function of code inspection preparation rate. This paper develops that model and computes the optimum code inspection preparation rate to minimize total cost (inspection + test). Existing program data (with significant caveats) have been used together with certain rough approximations to show that Fagan-style code inspections obey a simple predictive cost model.*

The purpose of this paper is not to present actual performance data but to demonstrate how such data can be analyzed in the context of a cost model extending over multiple development processes: code inspection, code inspection rework, test, test rework, and regression testing.

The purpose of the model is to demonstrate how to reduce development costs by managing the amount of time spent preparing for software inspections. All program data have been modified to protect the proprietary nature of that data. The conclusions and basic nature of the model are unaffected by these modifications.

This model can be used to optimize performance of future programs using a similar Fagan-style [1, 2] code inspection process with significant cost and schedule savings as well as quality improvement –

in effect, better, faster, and cheaper. Such performance improvements should result in improvements in profitability, competitiveness, and customer confidence in both contract performance and product quality.

Further, the model is easily modified to other work products and cost drivers, e.g., design complexity. Given that code inspection may not be particularly effective in finding design and requirements defects, it may make sense to extend the model to include more development processes, such as requirements development and requirements inspection, design development and design inspection, defect repair and associated inspection, etc.

## Inspection, Test, and Accounting Processes

To perform this kind of analysis, it is necessary for a project to have defined, consistently executed processes for code

inspection, code inspection rework, testing, and testing rework that collect and retain sufficient data. It is especially important in deriving this model that the inspection process was nearly statistically stable with respect to inspection preparation rate (lines of code reviewed per labor hour). Otherwise, the statistical fits would have little predictive value. The cost data are also assumed available and traceable to these processes.

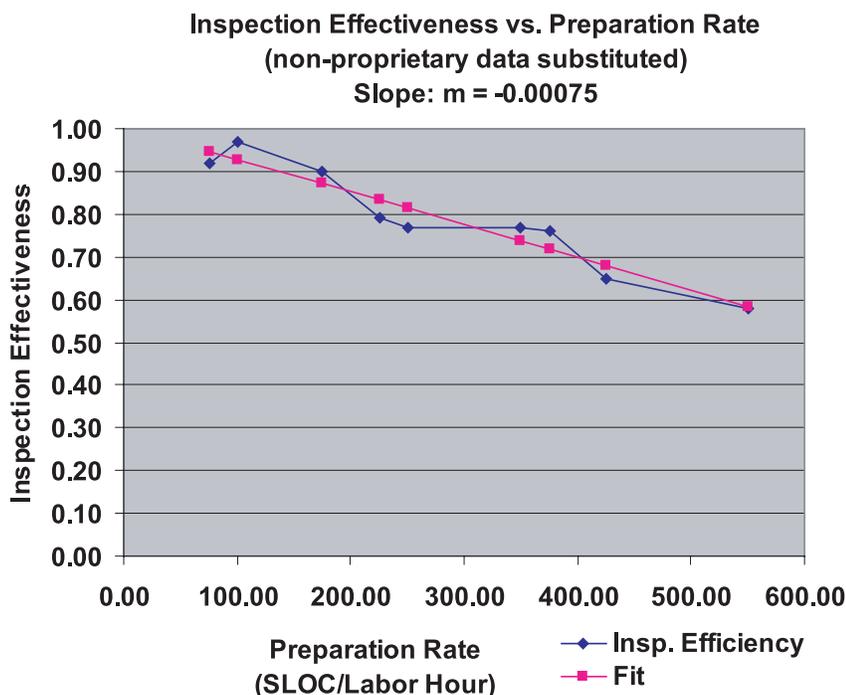
## Inspection Effectiveness

It is not possible to know the true, exact percentage of code defects found by code inspections until all defects have been discovered; furthermore that is not likely ever fully to happen. However, if it is assumed that a significant fraction of major code defects found in testing could have been found in code inspection, then there is value in studying the code inspection effectiveness.

This is the ratio of major code inspection defects<sup>1</sup> divided by the sum of major code inspection defects plus the major code defects found by testing. The code inspection effectiveness ratio has the desirable property of ranging from zero to one even though the relationship between major inspection defects and test defects is many-to-many rather than one-to-one. The model has the further benefit of quantifying how inspection preparation regulates testing costs.

When code inspection effectiveness is averaged by computer software configuration item (CSCI<sup>2</sup>), it is possible to develop a correlation between code inspection preparation rate and code inspection effectiveness. The correlation is well fit by a straight line with a y-intercept of 1.0 (see Figure 1). The F-test [3] shows a confidence level in excess of 99 percent with the linear fit accounting for more than 90 per-

Figure 1 : Least Squares Fit of Inspection Effectiveness



cent of the variation in the data. The F-test is a statistical test to verify that the function used to fit the data did not do so by accident.

Please note, however, that the fit function does not apply for very small preparation rates; at zero preparation rate, the preparation would never end. It is also not wise to extrapolate the fit function too far beyond the data on the right; generating a negative effectiveness is a clear indication of going too far.

The following variations to the data were performed and had only minor effects on the results (the slope changed by up to 10 percent, and the goodness-of-fit parameters changed slightly):

- Restriction of inspection defects to major defects [4] rather than all defects found.
- Removing any one of the data points.
- Use of three separate selection criteria on the test defects:
  - Total test defects.
  - Code-only test defects, including generated code.
  - Code-only test defects, excluding generated code.

In all of these cases, the effectiveness is well fit by a straight line that achieves 100 percent at zero preparation rate and declines by a constant amount for every 100 lines of code per labor hour of inspection preparation review (see Figure 1 on page 9).

### Cost Model

The cost model should include all major costs that are affected by the inspection preparation rate. At the top level, this cost model includes the costs associated with just two development processes:

1. Code inspections with inspection rework.
2. Testing with test rework and regression testing.

A more detailed analysis, including other code development processes or including the effect of code complexity on cost could not be supported due to absence of necessary data in the program databases. Code complexity, both algorithmic complexity and interface complexity, would be expected to affect inspection effectiveness as well as inspection rework labor hours and test rework labor hours. These were excluded from this model due

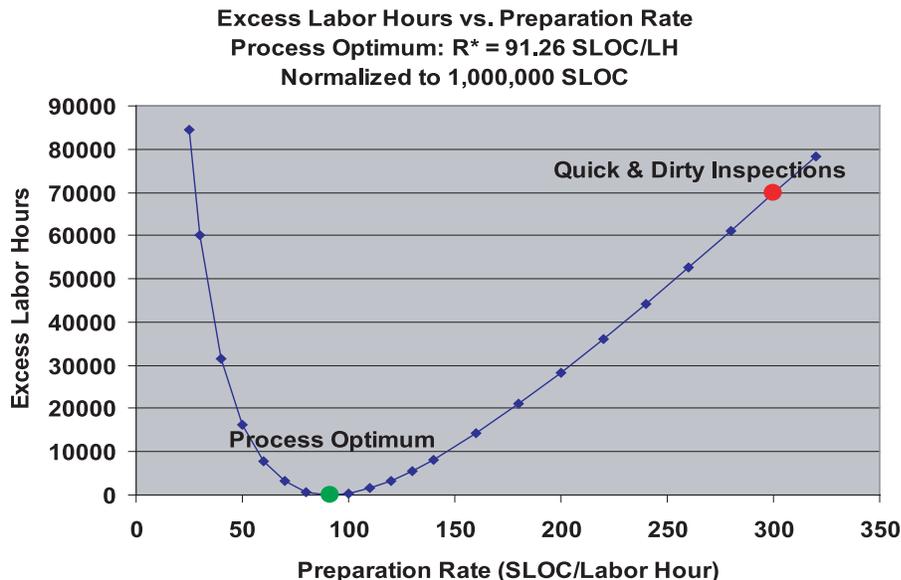


Figure 2: *Murphy's Tongue - Excess Labor as a Function of Inspection Preparation Rate*

to the absence of appropriate supporting data.

In this model, total cost is the sum of the following items:

- Code inspection preparation labor.
- Code inspection meeting labor.
- Code inspection rework labor.
- The labor for running the full test suite once.
- Testing rework labor.
- Regression testing labor.

In deriving the cost function, the following statements are assumed to be approximately true:

- The inspection preparation labor is proportional to the amount of code being reviewed.
- The inspection meeting labor and the inspection rework labor are proportional to the number of major defects found in the inspections.
- The test rework labor and the regression test labor are proportional to the number of major defects found during testing.
- A significant number of the major code defects found by testing could have been found earlier in code inspections.

Using these assumptions, the total excess cost due to discovered defects can be expressed as follows:

$$T_e = C + S * W_{ri} * I + S * D / R + S * I * (-m) * (T_r * E_t - W_{ri}) * R$$

See the Cost Function Derivation sidebar for a more detailed derivation of this cost

function and for variable and terms definitions.

### Cost Optimization

The optimum code inspection preparation rate is obtained by finding the point at which the cost function has a minimum. A minimum is found by setting the first derivative of the total cost with respect to code inspection preparation rate equal to zero, solving for code inspection preparation rate, and verifying that the point is a minimum and not a maximum or an inflection point:

$$I * S * (-m) * (T_r * E_t - W_{ri}) - D * S / R^2 = 0$$

The result is the following formula for the optimum preparation rate:

$$R^* = \text{SQRT}\{D / [I * (-m) * (T_r * E_t - W_{ri})]\}$$

For demonstration purposes, the following values are used:

- D = 4
- E<sub>t</sub> = 1.0 approximately (a very high quality requirement: man rated or species rated)
- I = 0.040 defects/SLOC
- m = -0.00075 (SLOC/labor hour)<sup>-1</sup>
- S = 1,000,000 SLOC
- T<sub>r</sub> = 20 labor hours/defect
- W<sub>ri</sub> = 4 labor hours/defect

Substituting this demonstration data yields the following optimum code inspection preparation rate:

$$R^* = 91.29 \text{ SLOC/labor hour}$$

The relative cost in excess of the cost at the optimum rate is given by evaluating the difference in the total cost at a given rate minus the cost at the optimum rate:

$$T_M = S*D*(1/R - 1/R^*) + S*I*(-m)*(T_r*E_t - W_{ri})*(R - R^*)$$

Since this curve is shaped like a parabola that opens upward (see Figure 2), there must be a unique preparation rate  $R^*$  that minimizes the total cost. Figure 2 is named in memory of Murphy's Law because any variation from the optimum, no matter how well intentioned, will increase development costs.

## The Cost of Variation

It is not enough to know the optimum preparation rate. No human process is without variation, so it is necessary to know the cost of variation. Whether a code inspection is run slightly too fast or slightly too slow, the cost is higher than the minimum cost. This idea is summarized in the following definition: "World class quality is on target with minimum variance." [5]

The cost of variation is approximately parabolic with respect to deviations from the optimum preparation rate. Therefore the Taguchi cost-of-variance formula for this model is the averaged second derivative term in a Taylor expansion of the cost function [6],

$$T_V = (D*S/R^3) * \langle (R - R^*)^2 \rangle$$

where the angle brackets indicate computing the average over the whole dataset. Substituting example data yields the following:

$$T_V = 5.26 * \langle (R - R^*)^2 \rangle \text{ labor hours}$$

This is an approximation to the exact cost behavior for this model, but it shows the salient point that variation itself has a cost that may be worth minimizing.

## Conclusions

If a program collects the right data from the inspection, test, and cost-accounting processes, performance and cost analysis can result in the ability to predict program cost and quality performance in terms of one inspection process control, inspection preparation rate. With the right data, a second driver could be added to the model, e.g., code complexity. This driver would be expected to affect inspec-

# Cost Function Derivation

Code inspection labor consists of preparation labor, inspection meeting labor, and inspection defect rework labor. Preparation labor is just size divided by preparation rate:  $S/R$ . The inspection meeting labor is linearly related to the preparation labor because the meeting time will be driven by the number of candidate defects to be discussed and recorded. The sum of preparation labor and meeting labor is  $C + D*S/R$ , where  $C$  and  $D$  are the linear regression coefficients. Inspection rework is driven by the number of defects found, and that is the discoverable defect insertion rate " $I$ " times the amount of code inspected " $S$ " times the inspection effectiveness " $(1 + m*R)$ " times the labor to fix an average defect " $W_{ri}$ ."

Testing labor includes the cost of testing perfect code " $T_0*S$ " (running through the whole test suite once), rework driven labor, and regression testing. Both regression testing and test rework will be driven by the number of defects detected during testing; the number escaping the inspections " $I*S*(-m)*R$ " times the test effectiveness " $E_t$ " times the labor for fixing and regression testing an average defect " $T_r$ ." Symbolically this can be expressed as follows:

$$T_C = T_0*S + C + D*S/R + W_{ri}*I*S*(1 + m*R) + T_r*E_t*I*S*(-m)*R$$

or

$$T_C = T_0*S + C + S*W_{ri}*I + S*D/R + S*I*(-m)*(T_r*E_t - W_{ri})*R$$

where:

$C$  = Y-intercept of empirical fit of total inspection labor (preparation plus meeting).

$D$  = Slope of empirical fit of total inspection labor vs. preparation rate.

$E_t$  = 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).

$I$  = Discoverable code defect rate = code inspection defect rate + test defect rate.

$m$  = Slope of code inspection effectiveness regression line.

$R$  = 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 one line of perfect code (regression testing not needed).

$T_C$  = Total cost (labor hours).

$T_e$  = Total excess cost due to discovered defects.

$T_r$  = Labor per defect to do test rework and regression testing.

$W_{ri}$  = Labor hours to rework a code inspection defect.

$C+D*S/R$  = 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).

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

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

tion effectiveness as well as inspection rework labor hours and test rework labor hours.

Realistically, one ought to be able to achieve 90 percent code inspection effectiveness by preparing at a cost optimizing rate of about 100 SLOC/labor hour without any other change in the code inspection process. To achieve further improvement, say 99 percent effectiveness without increasing code inspection cost, it would be necessary to improve the code inspection process. There are several ways to do this:

- Use checklists that are improved each time they are used.
- Train each developer to develop and to use an individualized checklist.
- Use Watts Humphrey's Team Software Process/Personal Software Process that does both of the above and more [7].

In the example using the optimal 91.29 SLOC/labor hour preparation rate rather than a *quick and dirty* 300 SLOC/labor hour, there is a savings of nearly 70,000 labor hours/MSLOC (\$ millions/MSLOC at any realistic labor rate). If the code inspection process could be modified to 99 percent efficiency without increasing code inspection costs, then the savings would potentially exceed 107,000 labor hours/MSLOC. Such potential improvement makes clear the value in performing code inspections at a deliberate pace and reliably recording certain key information to enable optimization of program performance. With this kind of analysis and with reliably recorded data, simultaneously working better, faster, and cheaper really is possible!

In conclusion, to get the best value from an inspection, it is more cost effective to prepare for that inspection as if preparing for a final exam in college rather than reading the material as if reading a light novel for entertainment. However, preparation thoroughness consistent with a Ph.D. thesis defense may not be appropriate unless there is a clear business case for exceptional quality, e.g., man-rated or better. ♦

## Acknowledgments

The author would like to thank John Gibson of Lockheed Martin Mission Systems and Pat Dorazio, Earl Pape, and

Bernie Pindell of Lockheed Martin Management & Data Systems, Gaithersburg, Md., for reading this report and making numerous helpful suggestions. Thanks are also due to Dr. Abol Ardalan of The University of Maryland University College for teaching me the value of cost models and how they are built, and to Dr. Gary Kaskowitz of The University of Maryland University College for teaching me basic business statistics.

## References

1. Fagan, Michael G., Design and Code Inspections to Reduce Errors in Program Development, *IBM Systems Journal*, vol. 15, no. 3, 1976.
2. Fagan, Michael G., Advances in Software Inspections, *IEEE Transactions on Software Engineering*, vol. SE-12, no. 7, July 1986.
3. Montgomery, Douglas C. and Runger, George C., *Applied Statistics and Probability for Engineers*, John Wiley & Sons, Inc., NY, 1994, pp. 315-317, 493-495, and 510-513. Also see the Microsoft Excel-97, SR-2 online help for the LINEST function, example 4, using the F and R2 Statistics.
4. See note 1.
5. Wheeler, Donald J. and Chambers, David S., *Understanding Statistical Process Control*, 2<sup>nd</sup> Ed., SPC Press, Knoxville, TN, 1992, pp. 141-147.
6. *Ibid.*, pp. 143-147.
7. Humphrey, Watts S.; Lovelace, Mark; and Hoppes, Ryan, *Introduction to the Team Software Process*, Addison-Wesley Publishing Co., 1999, ISBN: 020147719X.

## Notes

1. The Lockheed Martin Management & Data Systems definition of a major defect is a defect that will cause a malfunction or deviation from requirements or specifications, seriously violates policies or standards, indicates missing function, or makes the system unusable. The defect must be fixed since it may cause some degree of project failure, economic loss, poor customer satisfaction, or contractual or legal breach.
2. A computer software configuration item is a large set of related functionality produced by one team of developers.

## About the Author



**Bob McCann** is a staff systems engineer at Lockheed Martin Management & Data Systems in Gaithersburg, Md. He 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. DOE-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 M&DS Metrics Process Steering Committee and works on improving software development processes, methods, and metrics. He has a bachelor's degree in physics with a concentration in mathematics from Shippensburg University; master's degrees in physics and computer science from University of Maryland and Southwest Texas State University, respectively; and is working on a master's degree 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**

***“It's hard enough to find an error in your code when you're looking for it; it's even harder when you've assumed your code is error-free.”***

**—Steve McConnell**

***“The most important single aspect of software development is to be clear about what you are trying to build.”***

**—Bjarne Stroustrup**