On Optimum Module Size for Software Inspections

Pankaj Jalote
Department of Computer Science and Engineering

Ashok K. Mittal, Ramgopal Prajapat
Department of Industrial and Management Engineering
Indian Institute of Technology Kanpur
Kanpur – 208016; India
(Contact: jalote@iitk.ac.in)

Abstract

Inspection is widely believed to be one of the most cost-effective methods for detection of defects in the work products produced during software development. However, the inspection process, by its very nature, is labour intensive and for delivering value, they have to be properly executed and controlled. While controlling the inspection process, the inspection module size is a key control parameter. Larger module size can lead to an increased leakage of defects which increases the cost since rework in the subsequent phases is more expensive. Small module size reduces the defect leakage but increases the number of inspections. In this paper, we formulate a cost model for an inspection process using which the total cost can be minimized. We then use the technique of Design of Experiments to study how the optimum module size varies with some of the key parameters of the inspection process, and determine the optimum module size for different situations.

Keywords: Software inspections, statistical process control, control charts, software process, software metrics, design of experiments.

1. Introduction

The increasing cost, size, and complexity of software development has led the software engineering community to search for new ways for improving quality and reducing cost through software process improvement. Software inspections are now regarded as a technique that helps improve both the quality and productivity. It is also one of the techniques that have been widely studied. Starting from its original proposal by Fagan [13], the process has been further formalized and detailed [Gilb], and lessons learned in doing inspections have been reported [10, 15]. Experiments have been conducted to study impact of parameters like team size, sessions, process structure on the outcome of inspections [1, 2, 16, 17, 22], and for helping deployment [23].

The effectiveness of inspections depends greatly on how well the inspection process is being deployed – without proper execution and control of the process, inspections can become a time consuming activity with little benefits.

Control charts are a common technique used for monitoring and controlling process in manufacturing [18]. In a control chart, a run chart of the quality attribute of interest for the process being monitored is plotted. Upper and lower control limits are established based on the past performance of the process. If a point falls outside the control limits, it is assumed that the process is out of control which requires some special corrective actions to be performed to bring the process back in control. Though they were proposed originally for manufacturing processes, their use in software processes is growing, particularly for the inspection process [3, 5, 6, 12].

When controlling the inspection process, one of the key control factors determining the cost and effectiveness of inspections is the module size for inspections [16, 17]. What should be the optimum size of the module for inspection is an important practical issue. Barnard has suggested that maximum size of artifact being inspected should be 500 LOC, as larger inspection module size may encourage hasty preparation and less thorough meeting [11]. Another author recommended that the inspection module size should be less than 200 LOC for that particular project to increase defect detection [12].

It should be clarified that the main factors that determine the cost and effectiveness of inspections are factors like the preparation rate, coverage rate in the meeting, the size of the inspection team, etc. However, these factors are orthogonal to module size and a project manager needs to select them regardless of the module size he chooses for inspections. In this paper, we address only the issue of module size. Impact of some of the other factors is discussed in [1, 16, 17].

To determine the optimum module size for inspections, one approach is to collect data on inspections (either from field inspections or from controlled experiments) and then analyze the data to predict the optimum. This approach will require a lot of data that displays some sensitivity to module size such that the impact of module size can be studied. An alternate approach is to build a model of the inspection process and then use the model for determining the optimum size. In this work, we take the latter approach – we first build a cost model for inspections, in which cost of defects not detected in inspections is also modelled. We then use the model to determine the size that will minimize the overall cost.

We then use the methodology of Design of Experiments (DOE) to determine the optimum value of the module size for different situations. The results of these experiments indicate that in most situations the optimum module size is between about 150 LOC and 500 LOC. The optimum module size increases if the inspection process is more stable, i.e. it goes out of control after inspecting a larger amount of code. Actual value of the optimum module size for different situations is given later in the paper.

In the next section we give a brief description of statistical process control and their use in software processes. In section 3 we describe how the software inspection process is controlled using control charts and the role of module size in the overall cost. Section 4 gives the cost model, and section 5 discusses selection of module size for minimizing this cost. Results of the experiment for different values of parameters are given in section 7.

2. Statistical Process Control

The output of any process shows some variation in its key characteristics. The causes of this variation in performance can be classified into two categories – natural causes and assignable causes. The variation due to natural causes is inherent process and is considered due to normal interactions between different inputs of the process. This is small and is considered to be acceptable. The variation due to assignable causes is attributed to external disturbance to the process. This is due to abnormal or sudden behavioural change in the inputs or due to abnormal interaction in the process inputs. This variation is unintended and the amount of this variation is substantial. The variation due to assignable cause is unacceptable and the process is said to be out of control when the process gets influenced by this [5]. The main goal of control charts is to help identify the two situations such that suitable actions can be taken whenever assignable causes are present so that the process can be brought under control. There are various types of control charts, different ones suitable for different situations [7, 18].

In software, control charts are generally used to control various mini-process like the inspection/reviews process [4, 5, 6,12], personal process [19], maintenance [21] and testing [9], though it is also been used to monitor the development process [6, 8]. At the higher maturity level in the CMM framework, use of control charts is expected [26]. As the use of CMM is growing, we can expect the use of control charts for software process to grow [3, 24].

To control inspections, values of individual inspections are generally plotted (as opposed to plotting the means of some samples) [5, 12, 23]. For setting the control limits, the u-charts are perhaps the most suitable, in which the control limits for an inspection will depend on the size of the module being inspected [3, 6, 8, 10]. However, due to varying control limits such a chart is difficult to analyse and interpret [3]. Furthermore, if module size is not too small, the control limits do not change too much and an average can be used [12]. Hence, for simplicity, a fixed control limit may be used which may be treated as the average control limits [12]. We assume that fixed control limits are used when controlling the inspection process.

With this, the inspection process is controlled as follows. Defect density is the performance parameter that is plotted. Based on the data from past inspections, control limits are set on the performance. These control limits are generally set at three-sigma (where sigma is the standard deviation) around the mean in manufacturing, though it has been suggested that it is better to set tighter control limits for software processes [23]. These control limits establish a range on the defect density. When an inspection completes, its defect density is plotted. If the point falls outside the control limits, i.e. the defect density is outside the defined range, then that inspection is carefully examined, perhaps using other supporting data like effort in the different stages of inspection, meeting duration, etc. On examination, either it is determined that the inspection process is working fine and the point has fallen outside due to the natural variation (i.e. it is a false alarm), or that there is some cause for the point falling outside. In the former situation, no further action is necessary. In the latter case, the process is enhanced to remove the cause so that its future performance may fall within the control limits. This enhancement may take form of improving the checklist, training, improving the preparation, etc. An example of this type of process being used, though with different control limits for minor and major defects, is given in [20].

3 Software Inspections and Module Size

When using control charts for monitoring the inspection process, two types of errors are possible [18]:

Type I error: This error corresponds to a point falling outside the control limits when the process is actually under control. When this signal occurs, the analysis is done to search for assignable causes. As there is no assignable cause, this exercise leads to adding up of extra cost to the process. The probability of false alarm is represented by α, and is given by the equation: $\alpha = \Phi\ (\mu_{0,}\text{-}k) + (1\text{-}\Phi\ (\mu_{0,}\ k))$. In this equation, Φ is the cumulative distribution function probability mass function, which has the mean μ_0. The parameter k specifies the control limits (control limits are set at k*sigma around the mean).

Type II error: This error corresponds to a point falling within the control limits when actually the process is out of control. In such case more defects will go to the next phase in the development process and increase the cost of rework. The probability of Type II error is represented by β, and is given by $\Phi\ (\mu_{0,}\ \mu_{1,}\ k)$, where μ_1 is mean number of defects after the arrival of assignable cause, and μ_0, k, and Φ are as before. Further details about this can be found in [23].

These two errors are complimentary to each other. It means that if one increases, the other decreases, so both the errors cannot be made to zero simultaneously [23]. Inspection with smaller inspection module sizes increases the frequency of inspection and so will be more number of false alarms. More number of false alarms incurs unproductive effort to search for assignable cause and also de-motivates the inspectors. When larger inspection module size is taken then if an out of control situation is passed to the next stage, larger number of defects will be passed, incurring more cost. Optimum inspection module size is when total cost of controlling the inspection process is minimized. Optimum inspection module size is a trade off between type I and type II error in statistical terms, but actually represents the trade off between inspection control cost and effectiveness of inspection.

To formulate a model for optimum value of module size, we assume that the inspection process starts from an in-control state, i.e., initially all the variance in the process performance is due to natural causes. After some time, as the assignable causes spring up, the process goes out of control. After performing some inspections in the out of control state, these assignable causes will be detected and suitable action will be taken to eliminate them. This will bring the process back to the normal state. We assume that when some assignable cause arrives in the process, the process mean shift to $\mu_0 \pm \delta\ \sigma$. We also assume that, on an average, D LOC get inspected before the inspection process goes out of control. The average module size for inspection is represented by n, and the control limits are assumed to have been set at $k^*\ \sigma$, where σ is the standard deviation.

Clearly, the cost will depend on the cost of different activities. We require the following costs for modeling the total cost. (All costs are assumed to be in person-hours.)

- C_1 – Average cost of false alarm.

- C_2 – Average cost of fixing a defect in present phase.
- C_3 – Average cost of fixing a defect in the later phases.

4 Inspection Cost

Software inspection has defined steps like overview, preparation, and group meeting to conduct the inspection, followed by rework. There is a cost of conducting these inspections. Then there is the cost of false alarms, and the undetected failure cost. The sum of these costs is the total inspection cost. The inspection module size should be selected that minimizes this total cost.

4.1 False Alarm Cost

A control chart is plotted to control development process. When a point falls outside the control limits, analysis is carried out to find the root cause. To find the root cause, fish bone diagram is plotted and cause effect analysis is done [14, 18]. If there is no assignable cause, then the cost of analysis was unnecessary and forms the cost of false alarm. The cost incurred for each false alarm is denoted by C_1 (person-hour). Tighter the control limit, and smaller the inspection module size, more frequent are the false alarms, and hence higher the false alarm cost. The expected total false alarm cost (FAC) is:

$FAC = C_1 \alpha D / n$,

Where α is the probability of false alarm (or Type I error), D is LOC before process goes out of control, and n is size of inspection module (LOC).

4.2 Defect Leakage Cost/ Undetected Failure Cost

When a control chart indicates that no special cause is present, the process is assumed to be stable. When some assignable cause appears in the process, the process performance changes. Through the use of control charts, this shift is detected when a point falls outside the control limits. But before a point falls outside the limits, some points will fall within the control limits even though they represent inspections executed with the process having some assignable cause. This implies that more than expected number of defects will pass through to the next phases, as the mean number of defects has increased. This will incur extra cost in future. The number of points that will pass through undetected depends primarily on the control limits and the amount of shift in the process. This extra cost incurred due to leakage of defects caused by assignable cause, is called as Leakage Cost/ Undetected Failure Cost.

If the probability of Type II error is β, then the expected number of inspections that will take place before the assignable cause will be detected is $1/(1-\beta)$. When the shift in the mean defect density is δ, and the standard deviation is σ, then the extra defects due to this shift are $\sigma\ \delta$ per unit size. Cost to fix a defect in current phase is C_2 and cost to fix a defect in subsequent phase is C_3.So leakage cost/ Undetected Failure Cost (UFC) is:

$$UFC = (C_3-C_2)\, n\, \sigma\, \delta / (1-\beta)$$

In this the cost model the cost of expected defects are not considered since cost of fixing expected number of defects are cost of process capability.

4.3 Inspection Cost

Software inspection process has defined steps like overview, preparation, Inspection etc. Effort consumed in these steps may, in general, depend on factors like the complexity of code under inspection, training, tools and techniques used, experience of inspection personals. However, the main factor in this cost is likely to be the size of the module being inspected. We assume that the effort consumed in these steps is proportional to the size of module. Under this assumption the inspection cost can be modelled as simple linear model as function of module size, with the proportionality constant depending on the coverage rates in the different stages of the inspection process. For example, if the overview rate is O LOC/person-hour, preparation rate is P LOC/person-hour, and inspection meeting rate is I LOC/person-hour, then the cost of inspecting a module is $a1 + (n/I + n/P + n/O)$, where a1 represents the setup cost for an inspection. In other words, we can represent the cost of an inspection as $(a1 + n*a2)$. The average number of inspection performed before some assignable cause arrive in the process is D/n, where D is the expected LOC after which the process goes out of control, and n inspection module size. The average number of reviews to detect an assignable cause is $1/(1-\beta)$. So in the cycle of assignable arrival and removal a total of $D/n+1/(1-\beta)$ inspections are performed. As this cycle repeats itself and is the time for which we are considering the costs, the inspection cost (IC) is given by

$$(D/n+1/(1-\beta))\,(a_1 + n * a2).$$

5 Optimum Inspection Module Size

It is clear that false alarm cost and the defect leakage cost are two major costs for controlling inspections, while inspection cost is the cost of performing the inspections. These costs depend on module size and the control limits – the parameters that are under the control of a project manager. It is also clear that the false alarm cost increases as the module size decreases (or as the control limits are made tighter), while the defect leakage cost decreases as the module size decreases. The inspection cost is more of a function of the total code size being inspected. Optimum module size is one where the sum of these costs is minimum. This optimum will clearly depend on the control limits as well as the cost parameters of the process. The total inspection cost is the sum of the three costs:

$$\begin{aligned} & FAC + UFC + IC \\ & = C_1 \alpha D / n + (C_3 - C_2)\, n\, \sigma\, \delta / (1-\beta) + (D/n + 1/(1-\beta))\,(a_1 + n * a2). \end{aligned}$$

As we can see, the total inspection cost depends on several parameters. However, parameters like the average cost of fixing defects in different stages, cost of false alarm, mean shift when process goes out of control, amount of shift when the process goes out of control are process parameters that are not in the control of a project manager. The main parameters that are under the control of the project manager are the control limits (which determine the probabilities,) and the inspection module size.

For the cost parameters C2 and C3, it is best if we work with the ratio C3/C2, which represents how many times the cost increases if the defect is detected in later stages as compared to it being detected in the inspection. This ratio is what is more important and is a property of the process. In order to work with this ratio, we also take the ratio of C1 with C2, i.e. work with the ratio C1/C2. (Converting the total cost function to these ratios can be done easily by dividing the equation by C1.) The optimum module size is computed numerically, for which a C program has been developed. Essentially, we compute the cost for different module sizes, starting from 50 LOC and going to 1000 LOC with an increment of 10 LOC, and then take the size at which minimum cost is achieved. One plot of the different costs (including the other costs) and how they change with module size is shown in Figure 3.

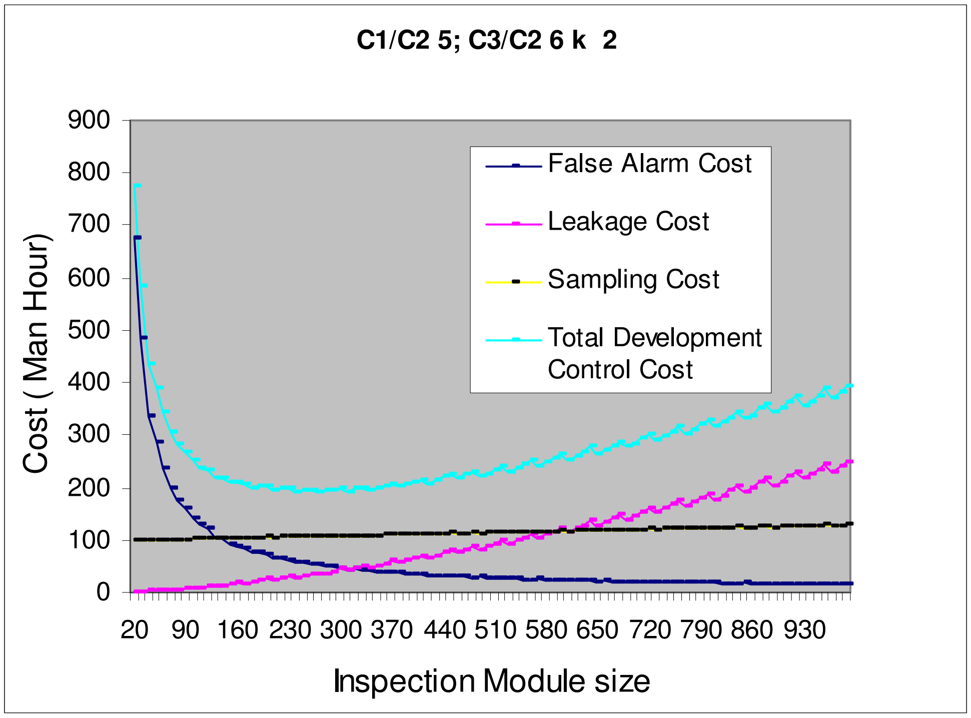

Fig. 3 Optimum Inspection Module size

As we can see from this example, there is an optimum module size for which the total cost is minimum. In the next section, through design of experiments (DOE) we study how this optimum varies with the different parameters and give some guideline for the optimum size for different values of parameters.

6. Design of Experiment for Optimum Module Size

To see how the optimum module size changes with different parameters, we follow the design of experiments approach, and employ the completely factored design [4]. In this approach, for the various input variables, a few levels are selected (e.g. low, medium, and high), and parameter values fixed for these levels. Then the model is applied to different combination of the values. In a completely factored design, all the different combinations are exercised. From the outcome for these different values, the variables that have a significant impact can be determined, and an equation of the dependent variable in terms of the input variables can also be determined [4].

In our model, the main input parameters are: the ratios C1/C2 and C3/C2, D (code amount after which the inspection process goes out of control), k (the control limits), the amount of shift if the process goes out of control, and the module size, n. Out of these parameters, the factors that are under the control of the project manager are the control limits and the module size. The others are essentially process attributes, which may differ from process to process. To reduce the number of parameters, we assume that the shift due to assignable causes is 1.5*sigma, an assumption that is frequently made. (Note, however, this assumption is being made only to make it easier to interpret the results; this parameter can also be incorporate easily, if desired.)

The average cost of fixing a defect in the current phase (i.e. where inspection is being conducted) is generally of the order of about half person-hour to about 2 person-hours – the cost of fixing a defect detected during code or design inspection is usually not very large. The cost C3 depends on the process and the software to be built and can range from a few person-hours to a few person-days or more. For our DOE, we set 3 levels for cost ratio (C_3/C_2) – low at 3, medium at 6, and high at 9. When there is a false alarm, C1 is the cost incurred in examining the inspection. This can be less than an hour (when the project manager discusses with the inspection team to determine that everything is fine), to many hours (when some detailed analysis is done.) For the cost ratio (C_1/C_2) also we select 3 levels – low at 1, medium at 5, and high at 9. For control limits (k) also we set three levels – 1, 2, and 3. For D, also we select three levels – 5 KLOC, 10 KLOC, and 20 KLOC. For these values, the different statistical parameters can be determined and are given in [27] along with ANOVA test results.

In Table 1, we give the optimum module size for different combinations of parameters. To keep the table manageable, we are only giving the values for control limit parameter K being set to 2 (i.e. the control limits are set to 2-sigma.) The value generally chosen in manufacturing is 3, though many people use 2 as "warning limits", and for software it is likely that the limits will be tighter than in manufacturing [23]. Hence we have selected k as 2. As k increases, as we show in Figures 2 and 3, the optimum module size decreases. Generally, the optimum module size for k=1 is about 1.5 times the optimum size for k=2, and the optimum size for k=3 is less than 0.5 times the size for k=2.

Table 1: Optimum module size for different input parameters

C1/C2	**C3/C2**	**Optimum Module Size**		
		D=Lo	***D=Med***	***D=Hi***
Lo	***Lo***	140	170	270
Lo	***Med***	110	140	170
Lo	***Hi***	80	110	170
Med	***Lo***	270	270	460
Med	***Med***	210	240	340
Med	***Hi***	170	240	270

Hi	***Lo***	310	490	540
Hi	***Med***	240	310	380
Hi	***Hi***	210	270	380

As we can see, as the stability of the process increases, reflected by the increase in D, the optimum module size increases. The increase is by about a factor of two from low stability (D=5 KLOC) to high stability (D=20 KLOC). We can also see that as C3/C2 increases, implying the cost of a defect in future stages increases as compared to the current stage, the optimum module size decreases. On the other hand, as the cost ratio C1/C2 increases, implying that the cost of false alarms is increasing, the optimum module size increases. How the optimum module size varies with these two ratios is shown in Figures 2 and 3 – these figures also show how the optimum module size changes with the control limits. The impact of the two cost ratios separately is also shown in the two figures.

For reducing the complexity of the experiment and to understand the impact of the key variables, we have fixed the other variables. Value chosen for some of the variables was mentioned earlier. There are some other parameters also for which we have fixed some reasonable values. For the constant a2, which depends on the rates of the different steps in the inspection process, we have chosen the value 1/50 representing that the total cost of all the stages is about 1 person-hour for 50 LOC. We have currently chosen the setup cost, a1, as zero. We have, however, conducted experiments with values of a1 as 0.5 and 1.0 person-hours, and a2 as 1/100 and 1/200. The value of optimum module size only varies slightly (less than x%). Similarly, though we have fixed C2 as 1 person-hours (representing cost of fixing a defect after inspections), we have experimented with values of C2 as 0.5 and 2.0. Again, our experiments indicate that the optimum module size changes very little with the absolute value of C2, and it depends primarily on the ratios C3/C2 and C1/C2.

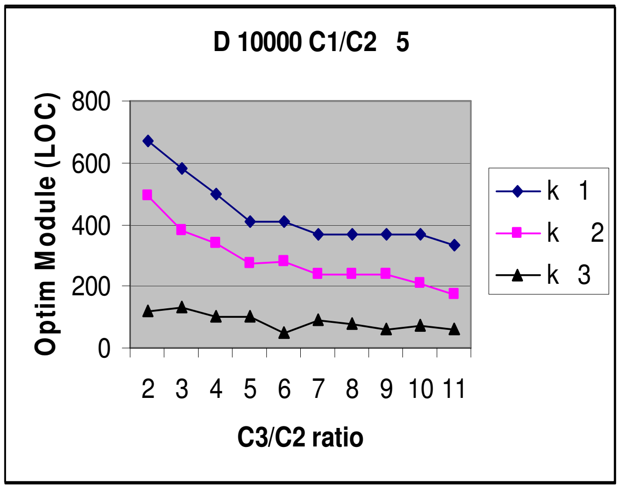

Figure 2: Change in optimum module size with ratio C3/C2

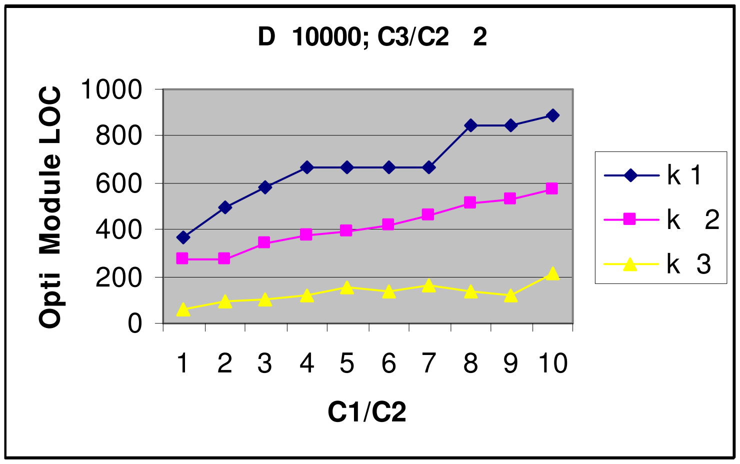

Figure 3: Change of optimum module size with the ratio C1/C2

From the model we can easily study how the optimum module size varies with other parameters – results of other such studies is given in [17]. From these plots, we can determine the dependence of the optimum module size on the different parameters. Though such an understanding is useful, we believe that the main use of this model is to determine the optimum module size for a project, given the characteristics of the process the project is using. For this decision, Table 1 is the main guide. Optimum values for other combination of parameters can be found in [17].

7. Summary

Inspections are widely believed to be an effective method for detecting defects. However, to ensure that inspections are cost effective, they have to be properly controlled. Control charts can be an effective way to monitor and control the inspection process. With control charts, the defect density found is plotted and, based on past performance of the inspection process, some control limits are established. If the result of an inspection falls outside the control limits, it is taken to be a likely sign of some special cause which has to be examined and eliminated.

There are three main costs involved in inspection – the false alarm cost, the defect leakage cost, and the cost of performing the inspection. All these costs depend on module size. In this paper, we have developed a simple cost model based on statistical process control concepts, which can be used to determine the optimum module size for which the total inspection cost is minimum.

We then used the design of experiments technique to understand how the optimum module size varies with the different process parameters. Results suggest that when the cost of fixing defects in subsequent phases is high then it is better to have smaller module size. If, however, the cost of fixing defects later is not much more than cost of fixing the defects now, then the optimum is larger. The optimum increases as the cost of false alarms increases. Mostly, the optimum module size is between 150 and 500 LOC.

We are currently building a web-service using which one can specify the parameters of a process and obtain the optimum module size.

References:

1. A. Porter C. A. Toman H. Siy and L. G, Votta," An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development" IEEE Trans. Software eng., 23(6): pp-329-346, 1997

2. Dewayne E Perry, Adam Porter, Michael W. Wade, Lawrence G. Votta, and James Perpich, "Reducing Inspection Interval in Large Scale Software Development", IEEE Transactions on Software Engineering, vol. 28, no. 7, July 2002.

3. R. Radice, " Statistical Process Control in level 4 and 5 organizations worldwide" 12th Annual Software Technology Conference Salt Lake City, 2000.

4. D. C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, 2001.

5. William A. Florac, Anita D. Carleton, "Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process," IEEE Software, July/August 2000, pp97-106.

6. E.F. Weller, "Practical Application of Statistical Process Control," IEEE Software, May/June 2000, pp 48-55.

7. W.A. Florac and A.D. Carleton, Measuring the Software Process: Statistical Process Control for Software Process Improvement, Addition-Wesley, Reading, Mass., 1999.
8 G.Y. Hong, M. Xie and P. Shanmugan," A statistical Method for Controlling Software Defect Detection Process", Computers & Industrial Engineering 37, 1999, 137-140
9 David Card, "Statistical Process Control for Software," IEEE Software, May 1994, pp 95-97.
10 E.F. Weller, "Lessons From Three Years of Inspection Data," IEEE Software, Sept. 1993, pp 38-45.
11 Jack Barnard and art Price, "Managing Code Inspection Information", IEEE Software, March 1994, pp 59-69.
12 Robert G. Ebenau," Predictive Quality Control with Software Inspections" http://www.stsc.hill.af.mil/crosstalk/1994/06/xt94d06e.asp
13 M.E. Fagan, "Design and Code inspection to reduce errors in program development", IBM System Journal, 1976
14. M.E. Fagan, "Advances in Software Inspections", IEEE Trans. on software Engineering, July, 1986.
15. R.B. Grady and T.V. Slack, "Key Lessons in Achieving Widespread Inspection Use" IEEE software, July, 1994
16. Harvey Siy," Identifying the Mechanisms to Improve Code Inspection Costs and Benefits" www.bell-labs.com/user/hpsiy/research/thesis/main.html
17. Porter Adam, Votta Lawrence, "What Makes Inspections Work?" IEEE Software, pp 99-102, Nov-Dec 1997.
18. D.C. Montgomery, Introduction to Statistical Quality Control, John Wiley and Sons, 1996.
19. Mark C. Paulk, "Applying SPC to the personal software process", Proceedings of the Tenth International Conference on Software Quality, New Orleans, October 2000.
20. Jalote P., CMM in Practice, Addison-Wesley, 2000.
21. E. Weller, "Applying statistical process control to software maintenance", Proceedings of Applications of Software Measurement, 1995
22. Adam Porter, Harvey Siy, Audris Mockus and Lawrence Votta, "Understanding the Sources of Variation in Software Inspections", ACM Transactions on Software Engineering and Methodology, Vol. 7, No. 1, January 1998, Pages 41–79.
23 Jalote P. ,Saxena A., "Optimum Control Limit for Employing SPC in software process" IEEE Transaction on software engineering, vol. 28, no 12, December 2000.
24. P. Jalote, "Use of metrics in high maturity organizations", *Software Quality Professional,* March 2002.

25. T. Gilb and D. Graham, *Software Inspections,* Addison-Wesley, 1993.
26. Software Engineering Institute, *The Capability Maturity Model – Guidelines for Improving the Software Process,* Addison-Wesley, 1995.
27. R. Prajapat, "Optimizing the module size for software inspections", IME Department, IIT Kanpur, 2003.