Module Size Distribution and Defect Density

Yashwant K. Malaiya and Jason Denton
Computer Science Dept.
Colorado State University
Fort Collins, CO 80523
+1 970 491 7031
malaiya|denton@cs.colostate.edu

ABSTRACT
Data from several projects show a significant relationship between the size of a module and its defect density. Here we address implications of this observation. Does the overall defect density of a software project vary with its module size distribution? Even more interesting is the question- can we exploit this dependence to reduce the total number of defects? We examine the available data sets and propose a model relating module size and defect density. It takes into account defects that arise due to the interconnections among the modules as well as defects that occur due to the complexity of individual modules. Model parameters are estimated using actual data. We then present a key observation that allows use of this model for not just estimation the defect density, but also potentially optimizing a design to minimize defects. This observation, supported by several data sets examined, is that the module sizes often follow exponential distribution. We show how the two models used together provide a way of projecting defect density variation. We also consider the possibility of minimizing the defect density by controlling module size distribution.

Keywords

Module size, defect density, reliability, module size distribution

1 INTRODUCTION

The defect density is one of the most important of the software reliability attributes. It is often one of the measures used to ascertain release readiness. There are two factors that control the defect density at release. One of them is the extent and effectiveness of the testing and debugging effort [1]. The other is the initial defect density present at the beginning of testing [2]. A study of the factors that influence the initial defect density is important due to two reasons. First it provides a quantitative method of identifying possible techniques for reducing the occurrence of defects. Secondly it allows one to estimate initial defect density, which can be used to plan the testing effort required.

There are several models that attempt to estimate the defect density. Some of them consider only a single factor while others take multiple factors into account. Models can be additive where influence of several factors is added [3], or they can be multiplicative. Multiplicative models allow a submodel for each factor to be developed independently. Multiplicative modeling is standard for hardware failure rate estimation. Such models for software cost estimation have been widely used. Multiplicative models to estimate software defect density include the RADC model [4,5] and the ROBUST model [6,7]. The sub-models considered in the past include the effect of the maturity of the development process, the skill of the programmers involved, and the complexity of the program. The problem including the impact of requirement volatility has been studied [8].

Development of such models is needed for estimation of defect densities. They also offer an interesting possibility. They can allow an organization to assess the possibility of controlling defect density even before testing begins.

This paper considers the problem of developing a model to account for variation in module size distribution, which can be used as a submodel in a multiplicative model. A common simplifying assumption is that the defects are distributed randomly in a software system. However we can intuitively reason that size of a module may in some way influence the defect density. It has indeed been supported by several studies.

This work was supported in part by a BMDO project funded by ONR and also by an ONR AASERT project.

It is popularly believed that decomposing a software system into small modules improves the design. Surprisingly the projects studied show exactly the opposite for a large range of module sizes. Basili and Perricone [9] studied a project with 90,000 lines of code. They studied 370 modules divided into 5 groups based on module size with increments of 50. They observed, contrary to their expectation, that larger modules were less error prone. This was true even when the larger modules were more complex as measured by cyclomatic complexity.

Shen et al. [10] studied three IBM software projects, with three separate releases of one of them. The sizes ranged from 7 to 326 thousand lines. They give a plot of defect densities for 108 modules. While they did not provide scales for the plot, they mention that for 24 modules with sizes exceeding 500 lines, the program length did not influence the defect density, which remains relatively constant. For the rest of 84 modules, the plot clearly shows that defect density declines as size grows. They also suggest a simple quantitative model for defect density in terms of module size.

Banker and Kemerer [11] have presented a hypothesis that for any given environment there is an optimal module size. For lesser sizes, there is rising economy, and for greater sizes the economy declines due to rising number of communication paths.

Withrow [12] examined the data for 362 ADA modules with total 114,000 lines of code. She divided the modules into 8 groups and gave a plot between module size and error density. This plot shows a remarkable minimum for modules with sizes 161-250, after which the defect density starts increasing with module size. Her results thus support the hypothesis by Banker and Kemerer.

Hatton [13] gave plots of data from a NASA Goddard project along with Withrow's data. He suggested two different models for the two regions. For sizes up to 200 lines, he suggested that the total number of defects grow logarithmically with module size, giving a declining defect density. For larger modules, he suggests a quadratic model.

In contrast, Rosenberg [14] has argued that the observed decrease in defect density with rising module sizes is misleading. We examine his argument and show that his observations can be restated to confirm with a model we propose. Fenton and Ohlsson [15] have studied randomly selected modules from a large telecommunications project. They did not observe a significant dependence. We will see a reason of their observation.

In the next section we propose a composite defect density model that takes into account both declining and rising defect density trends. We then apply it to actual data to obtain parameter values.

This model would be of little value if we did not know the module sizes vary in a project. We present a pleasant surprise. For several projects examined, module sizes distribution is quite similar. This observation is used to obtain an expression for the total defect content in a project with many modules. This allows us to examine the influence of module size distribution to the overall defect density. We discuss how module size distribution can be characterized in a defect density model that takes several factors into account. Finally we consider the intriguing possibility that defect density may be reduced simply by controlling module sizes.

2 A COMPOSITE DEFECT DENSITY MODEL

Here we construct a model that explains the data presented in the literature. A software system is built using a number of modules, which are themselves built using a number of instructions. There are two mechanisms that give rise to defects. Some faults termed *module-related* are related to how the project is partitioned into modules and how the modules interact. Other faults termed *instruction-related* are associated with the lower level building blocks. These faults arise because of imperfect interaction of instructions within a module and their individual implementations. We first obtain models of each of the two fault-types.

A. **Module-related faults**: We can term these *interface* faults because these will primarily be associated with parameters passed among the modules. However some of these may be related with assumptions made by modules regarding each other. They may also be associated with handling of global data. We assume that such faults are uniformly distributed among the modules.

If a module has size s, its defect density D_m for module-related faults is given by

$$D_m(s) = \frac{a}{s} \tag{1}$$

where the minimum possible values of s is one and a is a suitable parameter. In terms of defect density, such defects represent overhead that proportionately declines as module size grows. The model of equation one is consistent with the model given by Shen at al. [10].

Here it is interesting to examine Rosenberg's analysis [14]. He assumes that two random variables X and Y are statistically independent. He gives a simulated scatter plot of Y/X against X, which looks similar to the defect density versus module size plot given by Shen et al. [10]. However his assumption implies that the total number of defects in module is not related to its size, i.e. the defect density is inversely proportional to size. His basis assumption is thus equivalent to the model given in Equation 1. As we see soon, such behavior will be overcome by another factor in large modules.

B. **Instruction-related faults**: These are the faults that will dominate larger modules. We can term these faults *bulk* faults [8]. Let us assume that the probability that an instruction is incorrect has two components. The first component is a constant *b*. The other component depends on the number of other instructions a given instruction may interact with. We can assume that the second component is proportional to the module size s. We can then express the defect density D_i due to instruction-related defects as

$$D_i(s) = b + cs \qquad (2)$$

where *c* is another parameter.

Using Equations (1) and (2) we can express the total defect density *D(s)* as

$$D(s) = D_m(s) + D_i(s) = \frac{a}{s} + b + cs \qquad (3)$$

The model given in (3) specifies that the defect density tends to decline due to the first term as s increases. The third linear term will cause the defect density to rise. The middle term represents defect density that remains unaffected. To locate the minimum, we take the derivative of the RHS and equate it to zero. We get

$$-\frac{a}{s^2} + c = 0$$

giving the module size s_{min} for minimum defect density

$$s_{\min} = \sqrt{\frac{a}{c}} \qquad (4)$$

and the minimum defect density is given by

$$D_{\min} = (2\sqrt{ac} + b) \qquad (5)$$

It is possible to have a model more complex than in Eq. (3) using additional parameters. However that will require us to make further assumptions that will require justification.

It should be noted that the model implies two different regions.

Region A: For modules with s< s_{min}

Region B: For modules with s>s_{min}

In region A, defect density declines with rising module size and in region B the defect density rises.

3 ANALYSIS OF MODULE SIZE-DEFECT DENSITY DATA

Here we will analyze the available data given in the tables below. We apply the model given in (3) to the data to determine the parameter values.

The data given by Basili and Perricone [9] shows a declining defect density. This is in spite of the fact that the larger modules were more complex. The region of rising defect density is not encountered. As Withrow [12] points out, this is because there are only three modules larger than 200 lines. In this case, we had set parameter *c* equal to zero for curve fitting. The observed and fitted values are shown in Fig. 1. The data points all appear to be from region A, as mentioned above.

Table 1: Basili data [9]

Module Size (max)	Module count	Cyclomatic Complexity	Defect Density (/KLOC)
50	258	6	16
100	70	17.9	12.6
150	26	28.1	12.4
200	13	52.7	7.6
225	3	60	6.4

The Withrow data given in Table 2, [12] for Ada modules is plotted in Fig. 3. The data exhibits both declining and rising defect density trends. There is a noticeable jump from the third to the fourth data point in the plot. A possible explanation is that Withrow's study includes data from the test phase. It is possible that larger modules were not tested as thoroughly tested as the smaller modules resulting in relatively higher defect density.

Table 2: Withrow data [12]

Source lines	Modules	Defect Density
4-62	93	5.4
64-97	39	4.9
103-154	52	3.4
161-250	53	1.8
251-397	46	5.2
402-625	31	5.6
651-949	22	6.8
1050-5160	26	8.3

The Columbus Assembly data given by Hatton [13] is plotted in Fig. 2 along with fitted curve as given by our model. The defect density drops sharply until the module size of about 400 and starts rising gradually. The data fits the model very well.

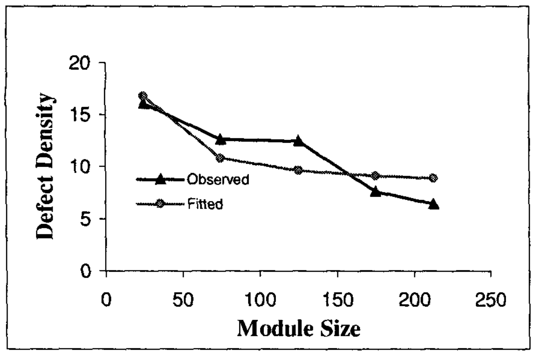

Figure 1: Defect density variation for Basili data.

Table 3 gives the values of the parameters obtained. The second column gives the approximate value for S_{min}, the module size corresponding to the minimum defect density. Since the data available only gives ranges, it should only be regarded as an approximate round number. The parameter *a* is controlled by the defect density of small modules. The parameter *c* accounts for the rise in defect density in larger modules. Its value is found to be quite small for the Columbus and Withrow data, and for Basili data there were no sufficiently large modules. The parameter *b* is largely influenced the minimum defect density observed, as we would expect.

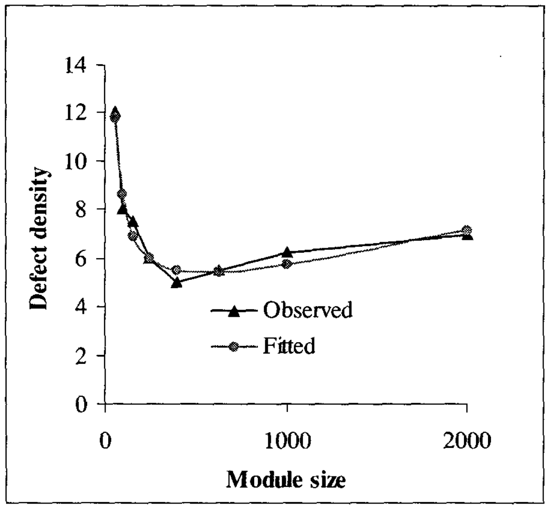

Figure 2: Defect density variation for Columbus data.

Fitting a model with three parameters to experimental data can be difficult because often one of the three can be used to compensate the effect of another one. Depending on the initial estimates, the estimated parameter values can converge to different combination of values. In this case, that can be avoided by initially setting the parameter *c* to zero while the other two are adjusted. After *a* and *b* have converged to specific values, *c* can be made non-zero for fitting.

In these three data sets, most of the available data points correspond to the declining defect density, where parameter *c* plays little role. The opposite is true for the data presented by Fenton and Ohlsson [15]. In their Table 5, the first data point groups all the modules with sizes lass than 500 LOC. They did not observe the initial declining trend, which is not surprising since the trend reverses around size 200-300 lines. Most of their modules were significantly larger than those in other studies. Thus they had very little data from region B. For their project, the data for release n shows a slowly rising trend, as in the Columbus and Withrow data sets. For release (n+1), the data does not show a clear trend.

It should be noted that a very accurate fit is not required since in any given project there will a range of module sizes. For Withrow data we note that the model does not fit with the sharp minimum. However overestimation of the defect density in some modules with be compensated by underestimation for slightly smaller and larger modules. It is possible to obtain a better fit by using a model with more parameters however generally fewer parameters provide better interpretation of the process.

Table 3: Parameter values for the three data sets

Data	S_{min}	Parameter values		
		a	b	c
Basili	-	220.9	7.83	0
Columbus	400	223.79	4.73	0.0013
Withrow	200	121.19	1.76	0.0063

4 DISTRIBUTION OF MODULE SIZES

To know the impact of module size variation within a project, we not only need to know the module-size defect-density relation, but also the distribution of module sizes for the project. One might think that there is a preferred module size and thus we may see a cluster of size values around the average with a Gaussian-like distribution. Surprisingly, there is evidence that it is usually not so.

Fig.4 shows the distribution of module sizes for the Basili data. Small sized modules are the most common. There are only a few modules with large sizes. The distribution curve drops exponentially with increasing module sizes.

Unlike Basili data, Withrow data includes many larger modules. Still as we see from Fig. 5, it has a similar module size distribution. The plots by Shen et al. suggest the same thing. We also examined module size distribution for Gnu C library with 792 modules and again found the same distribution. This surprising preference for smaller modules may either be due to programming practices or a natural tendency of the programming problems to be divisible into segments with such a distribution.

We can use an exponential function to arrive at a simple model for such a distribution. Let the density function for the module size distribution be given by this equation.

$$f_s(s) = g.e^{-gs} \quad (6)$$

Thus the module size distribution plots are described by

$$msd(s) = M.g.e^{-gs} \quad (7)$$

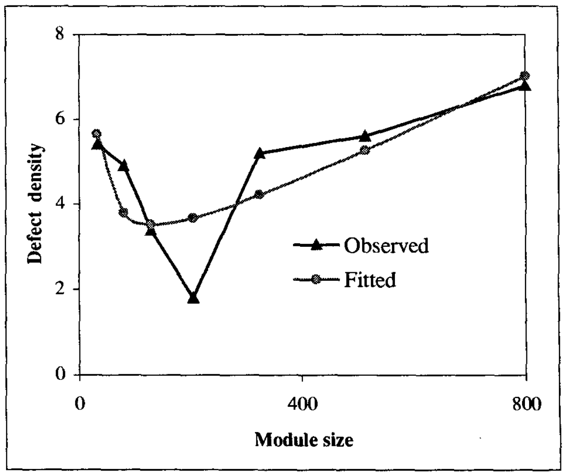

Figure 3: Defect density variation for Withrow data.

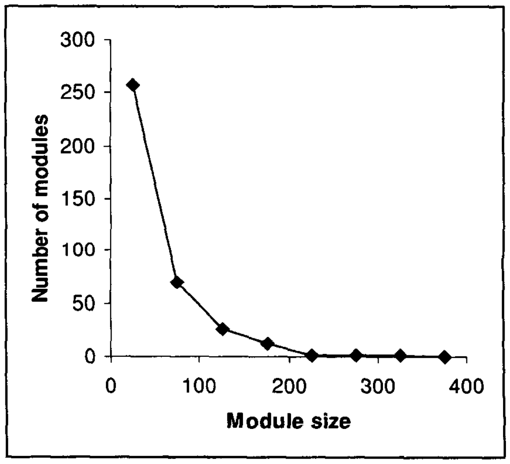

Figure 4: Module size distribution for Basili data.

In this paper, we will use some rule-of-thumb approximations to obtain some simplified expressions. These approximations are not necessary when dealing with an actual data set since a closed form algebraic expression and numerical values can always be obtained. However the simplified expressions allow us to interpret the results, which can be used for rule-of-thumb calculations. In an actual case we will know the size of the smallest and the largest module. However to obtain simple results we will sometimes take the minimum size to be one and the maximum size to be infinity. We have numerically verified that the approximations are reasonable.

Note that the parameter M represents the total number of modules since

$$\int_1^{\infty} msd(s) = \int_1^{\infty} Mge^{-gs} \approx M \tag{8}$$

The available data is all in the form of grouped data, which gives the number of modules m_i that lie in the range (s_i, s_{i+1}). We can estimate the value of $msd(s_i)$ using

$$msd(s_i) = \frac{m_i}{(s_i - s_{i+1})} \tag{9}$$

The Table 4 gives the values of the parameters M and g. The value of M is taken directly to be the total number of modules. If the value of M is obtained by using curve fitting, it will be slightly different. The value of parameter g is within the same range of magnitude; a larger value implies fewer large modules.

Table 4 includes a row for the Gnu C Library which includes a wide range of common functions. The size distribution of functions, shown in Figure 5 serves as a good indicator of the naturally occurring size distribution. Thus it is not surprising that we see the same distribution for Withrow data in Fig. 6.

The exponential distribution is not dependant on the language used. Our observation that the module-size is exponentially distributed for these projects has a significant implication. It allows a way of estimating the total number of defects for a project with different sized modules. Why the exponential distribution arises requires further investigation.

Table 4: Module size distribution parameters

Data	Language	M (total modules)	Parameter g
Basili	Fortran	370	0.0054
Withrow	ADA	362	0.0041
Shen	PL/S	108	0.0029
Gnu C Library	C	792	0.0097

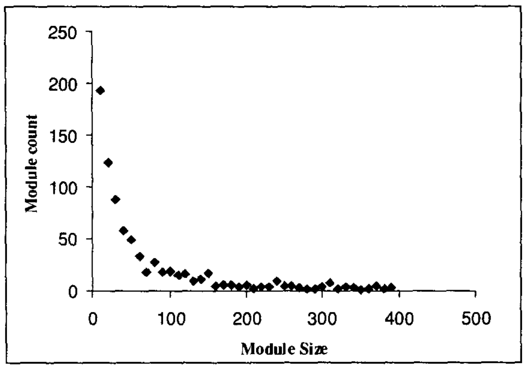

Figure 5: Gnu C Library size distribution

For Fenton data [15], the module size distribution appears exponential for all the data points except for the first one in their Table 1 with LOC <1000. Having very few small modules was perhaps a good choice since it reduced the number of very small modules that can exhibit high defect density.

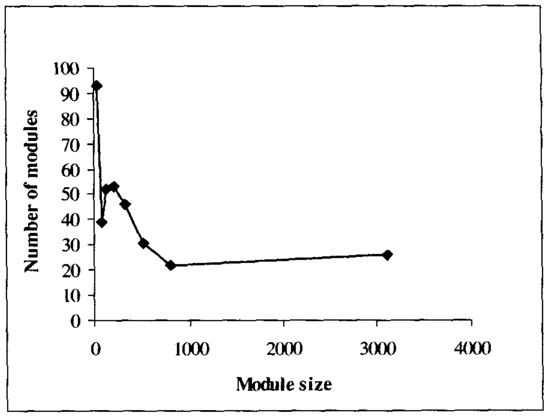

Figure 6: Module size distribution for Withrow data.

5 TOTAL DEFECT CONTENT

The total number of defects in a software system is found by adding up the defects in different modules. Since we know both the module size distribution and the dependence of defect density on module size, we can calculate the total number of defects N given by the following equation.

$$N = \int_1^{s_{\max}} Mge^{-gs}\left(\frac{a}{s} + b + cs\right).10^{-3}.s.ds \tag{10}$$

where s_{max} is the size of the largest module. Because of the exponential function, the number of large modules will be small. An approximate value can be obtained by setting s_{max} to be infinity. Because of the decaying exponential term, the result is not very sensitive to variation of s_{max}. The factor 10^{-3} is needed because the defect density is generally stated in terms of defects per 1000 lines of code.

The overall defect density is then given by

$$D = \frac{\int_1^{s_{max}} Mge^{-gs}(\frac{a}{s} + b + cs).10^{-3}.s.ds}{S_T} \qquad (11)$$

where S_T the total size of the project with all the modules. Equation (11) can be solved easily to get a closed form expression. Since the resulting expression is quite long, it is given in the Appendix.

Example 1: For a software system, there are 400 modules. The module size is exponentially distributed with g=0.004 in Eq. (7). The defect density is related to module sizes as given by (3), with a=120, b=1.8 and c=0.006. The largest module size is 2000 lines.

For this system the module size that will have the minimum defect density is obtained using (4),

$$s_{\min} = 141.42$$

The total number of instructions is given by

$$S_{tot} = \int_1^{s_{max}} Mge^{-gs}.s.ds = 100{,}000 \quad lines$$

The total number of defects given by (10) is

$$N = 941$$

and the overall defect density is found by (11)

$$D = 7.09 \ per \ KLOC$$

6 VARIATION OF MODULE SIZE DISTRIBUTION

For exponential module size distribution, the parameter *g* may vary due to either process variation or due to decisions deliberately made. Assuming that the overall size of the system is the same, how will the variation in *g* influence defect density?

Since overall size of the project S_T is fixed, we have

$$M = g.S_T \qquad (12)$$

Substituting for M in (11), we have

$$D = \int_1^{s_{max}} g^2 e^{-gs}(\frac{a}{s} + b + cs).10^{-3}.s.ds$$

This expression can be approximated to

$$D \approx 0.001(ag + b + 2\frac{c}{g}) \qquad (13)$$

This provides an optimal value of the parameter g given by

$$g_{opt} = \sqrt{\frac{2c}{a}} \qquad (14)$$

Note that from (12) we note that *1/g* represents the size of an average module. If all the modules were of equal sizes, the minimum defect density would occur when each of them has the size given by s_{min} from (4). On the other hand with a realistic exponential distribution, the optimal size s_{opt} of an average module is obtained using (14),

$$s_{opt} = \sqrt{\frac{a}{2c}} = \frac{s_{\min}}{\sqrt{2}} \qquad (15)$$

Equation (15) represents a surprising result. If modules of size 250 have the minimum defect density, the lowest overall defect density would occur when the average module size is about 177. That is because the asymmetric distribution of module sizes results in smaller modules having more impact on the overall defect density.

Example 2: If we allow the value of the parameter *g* to vary in Example 1, the optimal value of g is found from (13) to be

$$g_{opt} = 0.01$$

which yields a defect density of 4.2 per KLOC. Note that this is significantly less than the overall defect density 7.09 when the usual exponential distribution is present. This suggests that defect density may be reduced by breaking modules larger modules and combining smaller modules so that resulting modules have sizes close to

S_{opt}.

7 CHARACTERIZING MODULE SIZE DISTRIBUTION

The values of the parameters a, b, and c depend on the programmers' capabilities, maturity of the process and the extent of testing in prior phases. The effect of the module size variation is reflected in the parameter *g* above, where an exponential distribution is assumed. The exponential distribution was observed in most of the data set we examined. It arises due to natural reasons that need to be explored further.

The total defect density is influenced significantly as the plot in Fig. 7 shows. At g=0.01, the overall defect density is about 4.2 per KLOC compared with 7.1 at g=0.002. This behavior is dependent on the parameter values a, b and c as given in (14). This allows us provides us with a model to take into account the module size distribution. The multiplicative factor F_{ms} that takes into account module size distribution can be written as

$$F_{ms} = (Ag + B + \frac{C}{g}) \qquad (16)$$

The parameters A, B and B will need to be estimated from a similar project, such that for a default value of g, F_{ms} is unity.

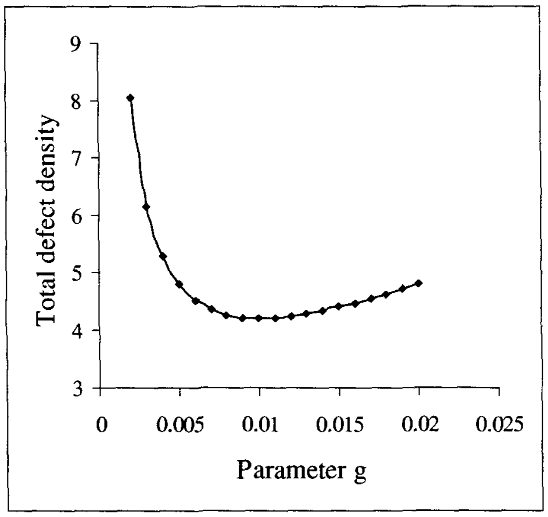

Figure 7: Variation of defect density with parameter g.

Example 3: If the values of a, b and c are as used in the above examples above, and if the typical value of g is 0.005, the model of (16) will be

$$F_{ms} = (25g + 0.375 + \frac{2.5.10^{-3}}{g}) \qquad (17)$$

When *g* is unknown, the default value of F_{ms} will be unity, as required [7].

An interesting possibility is provided by the fact that there is an optimal module size. It is a common recommendation to break very large modules into smaller ones. If there is a *magic module size*, say 200 LOC, at which inherently the defect density is likely to be lower, that would reduce the overall defect density. This would approximately correspond to the HP policy reported by Grady [16] that a cyclomatic complexity greater than 16 is undesirable. In many projects there can be a number of modules on the lesser side of the magic size. It would make sense to minimize the number of very small modules, say those smaller than 100 LOC. A possible approach can be to examine very small modules and attempt to coalesce them into larger modules. It can potentially reduce the overall defect density significantly provided the newly created modules contain fairly cohesive code.

The Ericsson Telecom data reported by Fenton and Ohlsson [15] suggests that there were very few small modules among those randomly chosen for the study. For their releases n and (n+1), the smallest modules were 37 and 196 lines of code. Reducing the number of very small modules would minimize the number of surface defects. Specifically adjusting the module size distribution will require the exponential distribution assumption to be modified. A possible approach to use Weibull distribution, which generalizes the exponential distribution. In cases where extensive module resizing is done, a discrete module size distribution may need to be used.

All the data sets used in this and previous studies came from actual industrial or space projects where objective was to produce a working system, rather than to collect data. The number of defects in a module could have been influenced by a number of factors. Some modules could have gone through more careful inspection and testing. Modules having been reused from previous releases with little modification would have lower defect density than new modules or those, which have been extensively modified. It would be desirable to collect data where such variations are carefully controlled. However since the data sets come from different projects and different organizations, they support the observations of the researchers. We can see that some of the differences in observations for different data sets are explained by the fact that some data sets cover only region A and some only region B. A clear trend may not be seen if the number of modules is small, one needs to use grouped data to observe a pattern.

8 CONCLUSIONS

The paper presents a model giving influence of module size on defect density based on data that has been reported. It provides an interpretation for both declining defect density for smaller modules and gradually rising defect density for larger modules. We observe that for several projects, distribution of module sizes is given by an exponential expression. We analyze the combination of the two to address how the overall defect density for a project with many modules can vary. We identify the condition for optimal distribution. A model for characterizing variation of defect density due to module size variation has been obtained which can be used as a sub-model for a multi-factor defect density model.

The exponential distribution occurs naturally in many software projects, for reasons that are yet to be studied. When module are specifically broken or coalesced to bring them closer to the size that is expected to give the minimal defect density, the exponential distribution may no longer be applicable. If small modules can be combined into optimal sized modules without reducing cohesion significantly, than the inherent defect density may be significantly reduced.

REFERENCES

[1] J. Musa, Software Reliability Engineering, McGraw-Hill, 1999.

[2] J. C. Munson and T. M. Khoshgoftar, “Software metrics in reliability assessment,” in *Handbook of Software Reliability Engineering*, Ed. M.R. Lyu, IEEE-CS Press/McGraw-Hill, 1996.

[3] M. Takahashi and Y. Kamayachi, " An empirical study of a model for program error prediction," Proc. of 8th International IEEE Conference on Software Engineering, pp. 330-33, Aug. 1985.

[4] Methodology for software reliability prediction and assessment. Technical Report RL-TR-95-52, Vol. 1 and 2, Rome Labs, 1992.

[5] W. Farr, “Software reliability modeling survey,” in *Handbook of Software Reliability Engineering*, Ed. M.R. Lyu, IEEE-CS Press/McGraw-Hill, 1996.

[6] N. Li and Y.K. Malaiya, “ROBUST: A Next Generation Software Reliability Engineering Tool” Proc. IEEE Int. Symposium on Software Reliability Engineering, pp. 375-380, Oct. 1995.

[7] Y.K. Malaiya and J.A. Denton, “What do software reliability parameters represent?,” Proc. International Symposium on Software Reliability Engineering, pp. 124-135, Nov. 1997.

[8] Y.K. Malaiya and J.A. Denton, “Requirement volatility and defect density,” Proc. International Symposium on Software Reliability Engineering, pp. 285-294, Nov. 1999.

[9] V. R. Basili and B. R. Perricone, "Software errors and complexity," *Comm. ACM,* vol. 27, pp. 42-52, Jan. 1984.

[10] V.Y. Shen, T. Yu, S. M. Thebut, “Identifying error-prone software-An empirical study,” *IEEE Trans. Software engineering,* vol. SE-11, pp. 317-324, April 1985.

[11] R. D. Banker and C. F. Kemerer, "Scale Economies in new software development," *IEEE Trans. Software Engineering*, pp. 1199-1205, Oct. 1989.

[12] C. Withrow, "Error density and size in Ada software," IEEE *Software,* pp. 26-30, Jan. 1990.

[13] L. Hatton, "Reexamining the fault density-component size connection," *IEEE Software*, pp. 89-97, March 1997.

[14] J. Rosenberg, “Some misconceptions about lines of code,” Proc. Int. Software Metrics Symposium, pp. 137-142, Nov. 1997.

[15] N.E. Fenton and N. Ohlsson, ‘Quantitative analysis of faults and failures in a complex software system,” IEEE Trans. Software Engineering, to appear.

[16] R.B. Grady, Practical software metrics for project management and process improvement, Prentice-Hall, 1992.

9 APPENDIX

Equation (11) above gives an expression for the overall defect density as

$$D = \frac{\int_1^{s_{\max}} Mge^{-gs}(\frac{a}{s} + b + cs).10^{-3}.s.ds}{S_T}$$

This is easily solved although the resulting expression is complex. The numerator is

$$[\frac{M10^{-3}}{g^2} e^{sg}(cg^2s^2 + sg^2b + 2sgc + ag^2 +$$

$$bg + 2c)]_1^{S_{\max}}$$

and the denominator is given by

$$[-\frac{(sg+1)}{g} Me^{-sg}]_1^{S_{\max}}$$

The approximations mentioned above have been verified using numerical values.