Fraunhofer Institut
Experimentelles
Software Engineering

Using Multiple Adaptive Regression Splines to Understand Trends in Inspection Data and Identify Optimal Inspection Rates

Authors:
Lionel C. Briand
Bernd Freimut
Ferdinand Vollei

IESE-Report No. 062.00/E
Version 1.0
January 1, 2001

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

Inspections have been shown to be an effective means of detecting defects early on in the software development life cycle. However, they are not always successful or beneficial as they are affected by a number of technical and managerial factors. One important aspect is to understand what are the factors that affect inspection effectiveness (the rate of detected defects) in a given environment, based on project data. In this paper we look at management factors such as the effort assigned, the inspection rate, and so forth. We collected data on a number of analysis and code inspections, and performed a multivariate statistical analysis. Because the functional form of effectiveness models is a priori unknown, we use a novel exploratory analysis technique: Multiple Adaptive Regression Splines (MARS). We compare the MARS model with more classical regression models and show how it can help understand the complex trends and interactions in the data, without requiring the analyst to rely on strong assumptions. Results are reported and discussed in light of existing empirical results .

Keywords: Software Inspection, Inspection Effectiveness, MARS

Table of Contents

1 Introduction

Inspections have been shown to be an important defect detection technology [9][13]. However, when one is faced with planning inspections, a number of decisions have to be made. For example, the following questions are considered relevant as they are deemed to have an impact on inspection effectiveness, that is the capacity of inspections to uncover defects:

– What overall effort to devote to the inspection?
– What should be the inspection rate?
– How many participants to involve?
– How should the material to be inspected be broken down?

In order to answer such questions, which will be discussed in further details below, we need to develop models that relate defect detection effectiveness to variables such as effort, number of participants, or the amount of code inspected. To build such effectiveness models, data on inspections need to be collected and multivariate statistical analysis techniques are required to exploit such data and capture the complexity of the phenomena under study.

There is, however, a problem that we face when building such multivariate models. We rely on such models to help us predict and understand the relationships between defect detection and the variables mentioned above. But to do so, common and mature approaches, such as multivariate regression analysis, require that we specify beforehand the functional form of the relationships among model variables. Because there is little knowledge and theory about inspection effectiveness factors [6][2], this is difficult to do without taking the risk to fit an inadequate model to the data.

We are therefore in a typical situation where we need to perform some exploratory analysis in a multivariate context. Not only we are interested in modeling relationships, e.g., between defect detection and effort, but we would like to find out about interactions between variables, that is the way they affect each other's impact on effectiveness, e.g., effort impact on defect detection may depend on the inspection rate, that is the pace followed while inspecting documents.

This paper will first contribute by using a novel exploratory, multivariate analysis technique (MARS [7]) to help us build a defect detection prediction model that is as accurate as possible. As far as we know, MARS has not been used before for building software engineering models. For the purpose of evaluating the gain of using MARS compared to conventional approaches, we will also

build ordinary least squares regression models following classical variable section procedures [5]. We will compare the two types of models in terms of their goodness of fit, predictive power, and their capacity to help us understand the phenomena under study. We will then analyze the MARS multivariate models to gain some understanding regarding a number of common hypotheses regarding inspection defect detection effectiveness and its relationship to various factors such as effort, participants, or inspection rate. From all these results, we then provide general recommendations regarding the construction of such models in other inspection environments.

The paper is organized as follows. We first summarize the current state of knowledge based on our review of the literature. Then, in Section 3, we describe the motivations, the environment in which our study was performed, and the data collection performed. Section 4 then introduces the main modeling technique used: Multivariate Adaptive Regression Splines (MARS). Regression analysis results are then reported in Section 5, followed by MARS results in Section 6. The latter section also performs comparisons with results in Section 5. Section 7 then concludes the paper by summarizing findings and lessons learned.

2 Managing and Improving Inspections

In order to improve and control inspections, it is first necessary to identify the factors impacting inspection effectiveness, that is the number[1] of defects detected. Knowingand understanding these factors will enable us to control them when planning and conducting an inspection, so that a maximum defect detection effectiveness can be achieved.

In this section, we summarize existing empirical results so that we can compare our results and discuss them in light of reported data. In the literature, several factors have been hypothesized and/or shown to affect the effectiveness of inspections. For example, several studies showed that the effort spent on inspecting an artifact has a major impact on the inspection effectiveness [13]. Christenson et al. [2] reported the preparation effort of the inspectors to be correlated with the density of defects found. Ebenau [6] identified the examination rate[2] and the preparation rate[3] as major drivers of inspection effectiveness. In a context where defects are searched during meetings (i.e., examination), spending more effort on preparation (i.e., reading a document) yields a higher understanding of the document to be inspected and hence results in more detected defects during inspection meetings. Spending more effort on examining the document simply increases the thoroughness of the inspection and increases the chances of detecting defects.

Characteristics of the inspected product can have an impact on the effectiveness of inspections as well. Some studies [2] [6] reported the size of the inspected document to impact inspection effectiveness as a larger document usually contain a larger number of defects. Additionally, the "complexity" of a product [2] [8] [9] and its initial quality [10] can have an effect on inspection effectiveness as these factors relate to the defect content of the inspected product.

The characteristics of the inspection team can also show some effect on inspection effectiveness. Porter et al. [11] suggest that an inspection team composed of several inspectors can enable the detection of a wide variety of defects since each inspector is likely to rely on a different expertise. The larger and the more varied the team, the better the coverage of the document, thus resulting in an increased inspection effectiveness. Additionally, the qualification

[1] Using the proportion of defects found would be equivalent as this is the number of detected defects divided by a constant: the total number of defects.

[2] Effort spent examining the document in the inspection meeting per unit of document size (e.g., LOC)

[3] Effort spent reading the document during preparation per unit of document size (e.g., LOC)

of the inspection participants can impact the effectiveness. Inspectors well versed in the application domain can already know about potential defects in the inspected product [11] [15] [13].

Finally, the organization of the inspection process and its infrastructure can have an impact on the effectiveness as well. Porter et al. [11] identify the number of inspections sessions as another factor influencing inspection effectiveness. Additionally, the defect detection technique chosen for an inspection may have an impact on effectiveness as well. For example, more systematic techniques may help inexperienced inspectors [12].

3 Case Study Setting and Data Collection

The work described in this paper took place in a business unit of Siemens AG, Germany, which is developing products and services for mobile communication and intelligent networks. In this particular business unit, inspections are performed throughout the entire life cycle to ensure the quality of all software artifacts. Thus, inspections are performed after each of the development phases: analysis, design, and coding. Due to the substantial investment in software quality through systematic inspections, the quality assurance team's objective is to continuously improve and control the defect detection effectiveness of these inspections.

Because software quality is a major objective in this environment, data is systematically collected regarding the factors that have been shown to affect inspections' effectiveness in the published literature: the number of inspection participants, the type and size of the work product (different size measures are used depending on the type of product), the size of the change from the last version of the work product, the inspection effort, the number of defects found (classed into major and minor defects, where major defects are those that would lead to a fault or failure in subsequent phases), and the estimated rework effort.

Depending on the artifact to be inspected, three different kinds of inspection methods are applied. First, with the so-called "comment technique", the artifact is distributed to many inspectors who simply read the document and send their comments to the author. There is no formal, precisely defined inspection procedure. The second one is an inspection approach, similar to the one described in [15], where inspectors use checklists to identify defects during preparation and where an inspection meeting is held to collect the individual inspectors' defects. Third, there are "intensive" inspections, which are similar to the approach proposed in [18] and enhance the second inspection approach in two ways. First, during the inspection meeting, a reader reads parts of the document, which is then discussed by the participants. In this discussion inspectors also collect the defects they detected during preparation. Thus, with intensive inspection, there is more interaction between authors and inspectors on the content of the inspected document. Second, a discussion takes place on how to prevent the detected defects in the future as

The data collected in this environment and used throughout the analysis is listed in Table 1. In addition, qualitative data regarding the type of inspection performed (as discussed above) and the type of document inspected was col-

lected. The definition of these categorical variables is specific to the environment under study and only meaningful in that context.

In addition, based on the data collected, two composite measures are computed: effort per participant (Effpart), inspection rate (Totrate and Rate as Loc and Iloc per effort unit, respectively).

Variable	Description
Defects	sum of major and minor defects detected in the inspection
Particip	number of participants taking part in the inspection (either in preparation or in the meeting)
Effort	total effort spent by the participants for the inspection (including preparation effort and meeting effort) in person-minutes
Sessions	number of meetings that were performed to completely inspect a document
Dloc	(for code documents) size of the change compared to the last version
Loc	(for code documents) total size of the inspected document.
Iloc	(for code documents) size of document's part actually inspected.
Totpage	(for analysis documents) the total size of the inspected document measured in number of pages
Inspage	(for analysis documents) the size of the inspected part of the document in number of pages. (The documents were structured according to a pre-defined standard for analysis documents)

Table 1: Inspection Measurement

4 Multivariate Adaptive Regression Splines (MARS)

When analyzing and modeling the relationship between fault detection and inspection effort, as well as other potential effectiveness factors mentioned earlier, one of the main issues is that relationships between these variables are expected to be complex (non-linear) and to involve interaction effects. Because we currently know little about what to expect and because such relationships are also expected to vary from one organization to another, analyzing inspection data in order to understand what affects inspections' effectiveness is usually a rather complex, exploratory process.

When using typical regression techniques, the risk to fit the data with models that may be simplistic is rather difficult to avoid. For example, we typically resort to log-linear models to handle non-linear relationships [5]. But this comes with a number of drawbacks such as forcing the model to have a null intercept or making the analysis of interactions impossible (the whole log-linear model is a multiplicative expression). An alternative to model such complex relationships is artificial neural networks [16]. However, such models are difficult to interpret [4] as it is nearly impossible to assess the impact of individual independent variables on the dependent variables and their interactions. Interpretation is key in our context, as the models we build are not just used for prediction purposes but are also used to support decision-making and, from a more general perspective, gain a better understanding of software engineering processes.

MARS is a novel statistical method presented in [7] and supported by a recent tool developed by Salford Systems[4]. At a high level, MARS attempts to approximate complex relationships by a series of linear regressions on different *intervals* of the independent variable ranges (i.e., subregions of the independent variable space). It is very flexible as it can adapt any functional form and is thus suitable to exploratory data analysis. One challenge though is to find the appropriate intervals on which to run independent linear regressions, for each independent variable, and identify interactions while avoiding overfitting the data. This is the purpose of the search algorithms proposed by the MARS methodology. Though these algorithms are complex and out of the scope of this paper, MARS is based on a number of simple principles. They are introduced below in order for the reader to understand the results presented in later sections. It is also interesting to note that the results in [4] show that for datasets of sizes comparable to what we use in this study, MARS models are more likely to be accurate than artificial neural networks.

[4] www.salford-systems.com

Figure 1 illustrates a simple example of how MARS would attempt to fit data, in a two dimension space (where Y and X are the dependent and independent variables, respectively), with piece-wise linear regression splines. A key concept is the notion of *knots*, that are the points that mark the end of region of data where a distinct linear regression is run, i.e., where the behavior of the modeled function changes. Figure 1 shows two knots: x_1 and x_2. They delimit three intervals where different linear relationships are identified. MARS search algorithms identify appropriate knots in an automated way, though a number of search parameters have to be set by the user. Of course, in a case with higher dimensions and interactions between independent variables, the search becomes much more complex but the fundamental principles remain the same. The reader is referred to [7] for further details.

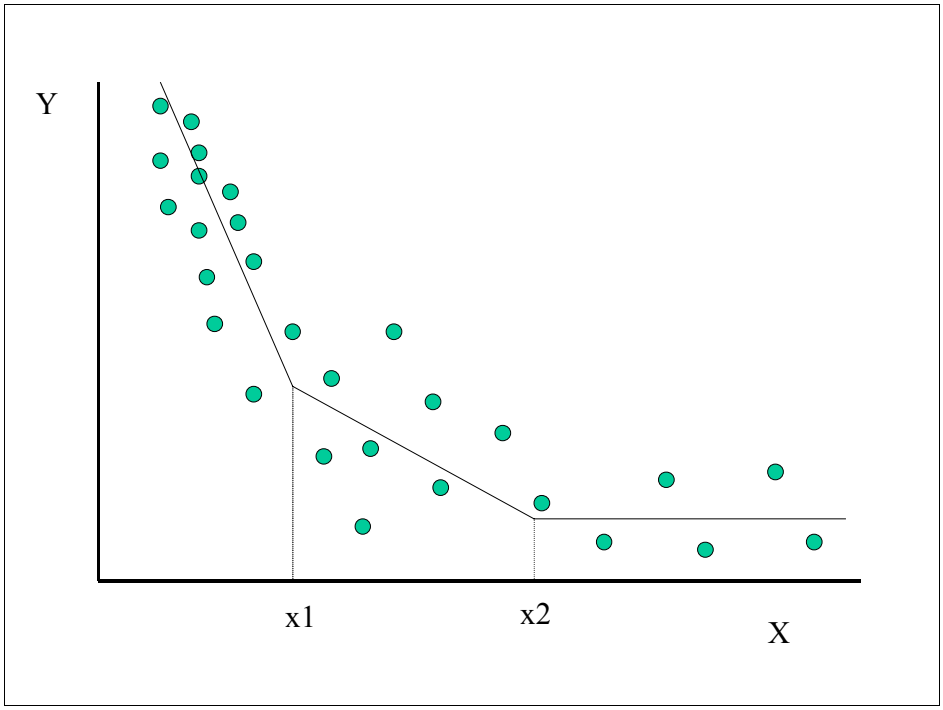


Figure 1: Example Knots in MARS

In order to model the concept of knots and piece-wise linear regression splines, MARS uses the concept of basis function. These are functions of the form:

max(0, X-c), or
max(0, c-X)

where *X* is an independent variable and *c* a constant.

Such basis functions re-express an independent variable *X* by mapping it to new variables, which are of the form described above. For max(0, X-c), *X* is set to 0 for all values of *X* up to some threshold value *c* and is equal to *X* for all values of *X* greater than *c*. By mixing the two types of basis functions presented above and providing adequate values for *c*, it is possible to approximate any functional shape. Determining the right knots (threshold values *c*) is a key challenge addressed by MARS search algorithms. In short, basis functions are used as the new independent variables of our regression estimation models. MARS also looks for interaction terms among basis functions, thus leading to the modeling of the interactions among independent variables.

MARS also provides some insight regarding the importance of variables as predictors of defect detection effectiveness, the dependent variable. MARS refits the model after removing all terms involving the variable to be assessed and calculates the reduction in goodness of fit. All variables are then ranked according to their impact on goodness of fit. An optimal MARS model, in terms of goodness of fit, is the one with the lowest generalized cross-validation (GCV) measure. The function $\hat{f}$ is the MARS prediction model based on basis functions. *Y* is the dependent variable–the number of defects detected in our case–and there are *N* observations in the dataset. *C(M)* is the cost-complexity measure of a model containing *M* basis functions.

$$GCV(M) = \frac{1}{N}\sum_{i=1}^{N}\left[y_i - \hat{f}(x_i)\right]^2 \Big/ \left[1 - \frac{C(M)}{N}\right]^2$$

Besides the usual computation of the squared prediction error, there is a cost incurred per basis function included in the model so as to avoid overfitting, much like adjusted R^2 in least-squares regression. In other words, *C(M)* is used to penalize model complexity, prevent the overfitting of data, and promote the parsimony of models. This is usually defined as *C(M) = M* in linear least-squares regression and this is what we use in this paper. The function $\hat{f}$ is the MARS prediction model based on basis functions. *Y* is the dependent variable–the number of defects detected in our case–and there are *N* observations in the dataset. The loss in *GCV* associated with removing all the basis functions in which a variable is involved is the measure used to assess its importance in a MARS model.

Other measures of goodness of fit can be used to assess regression models from a practical standpoint. In particular, we will use four of them in this paper.

– Absolute Relative Error (ARE): |actual – estimated|

– Magnitude of Relative Error relative to the actual value (MRE):

|actual – estimated| / actual

– Magnitude of Relative Error relative to the estimated value based on a regression model (MRE'):

|actual – estimated| / estimated

– Coefficient of determination of the regression model (R^2 between actual and predicted defects)

Looking at the above measures is relevant, especially when the models are to be used for prediction. However, as discussed below and in Appendix A, they cannot really be used to compare the plausibility of non-nested models, i.e., determine which model fits the closest to reality. Therefore, such goodness-of-fit measurements should be used and interpreted with care.

A few other technical issues need to be considered when using MARS. In [4], simulations and case studies show that MARS is sensitive to outliers (i.e., observations in empty parts of the sample space, , which are more difficult to detect in multidimensional settings) and strong collinearities among independent variables[5]. In the analysis below, we will attempt first to remove outlying, overinfluential observations in the sample space before building any model. However, to retain the objectivity of the analysis results, outliers will be kept during the validation stage of the models (see cross-validation below). These outliers will be identified using the Jackknife Mahalanobis distance (distance from the sample space multivariate mean or centroid) [5]. We will verify whether observations showing a very large Mahalanobis distance have an overinfluential effect on the multivariate models that we build. If this is the case, they should be removed for model building. The main motivation here is to make sure that no one observation will distort the models being built. In the case study presented below, one observation was clearly outlying and hence removed for model building purposes. Regarding collinearity, we will use Principal Component Analysis (PCA) [5] to identify strongly collinear variables belonging to the same principal component. One variable from each principal component will then be allowed to enter the MARS models. In other words, all of the above will help us prevent, to the best extent possible, the computation of spurious results by MARS search procedures. This would otherwise prevent us from building stable, accurate models and understand the inspection processes.

[5] Note that these problems also affect the reliability of variable selection procedures used to build least squares regression models. MARS may, however, be more sensitive to it because of the automated computations of optimal knots.

5 Multivariate Regression Analysis

As mentioned above, we will first use conventional procedures for investigating the inspection data by applying ordinary least squares regression [5]. Our goal is to use the least-squares regression results as a comparison baseline to assess the benefits of using MARS. We will use stepwise regression procedures to select significant covariates and try different functional forms to fit the best regression model possible. Like for MARS models, we will remove the same outlier and make sure we do not allow strongly correlated variables to enter the model[6].

We first identify the factors, among the ones that are measured, which have a significant impact on inspection effectiveness in the studied environment. Following a procedure similar to the one used in previous studies [6] [2], we performed a multivariate regression analysis on data from both analysis and code inspections. Data for design inspections were not investigated as the size of the data set was too small to obtain meaningful results. The total number of detected defects (i.e., the sum of major and minor defects) was chosen as dependent variable for the effectiveness of inspections. Independent variables are: the number of inspection participants (Particip), the inspection effort (Effort), the number of inspection sessions (Sessions), the size of the change from the last version (Dloc, for code only), and the size of the inspected document. For analysis documents, the total size of the inspected document (Totpage) and the size of the document actually inspected (Inspage) were roughly measured in number of pages. Note that the number of pages may be an acceptable measure as analysis documents tend to have a consistent structure. For code documents, the following measures were collected: the total size of the artifact in lines of code (Loc), the size of the change compared to the previous release (Dloc), and the number of lines of code actually inspected (Iloc).

We performed a stepwise multivariate regression to identify which factors, among the ones presented above, have a significant impact on the number of defects detected. In building the regression models two common functional shapes, namely the linear and log-linear relationships, were investigated. Below, we present only the log-linear models because they showed to be more plausible based on goodness of fit values (median MRE) and a standard plausibility test to compare functional forms: the J-Test [3]. Though certainly not perfect, the log-linear model allows us to model non-linear relationships while still using linear regression analysis. Because of its practicality, this functional form

[6] As this tends to make stepwise variable selection procedures unreliable

is commonly used in software engineering data modeling, such as in cost estimation [17].

In the analysis we only considered inspections in which at least one defect was found. The rationale for this selection was that, based on our discussion with the quality management team, we suspected that some of these zero-defect inspections might not have been thoroughly performed. In particular the ones performed according to the rather loose "comment technique" represented 77% of all zero-defect inspections and were particularly suspect. Since we could not collect information regarding the inspection process conformance or the initial quality of inspected documents, we decided it would be more prudent to eliminate these observations from the analysis. To perform such quality checks and decide on the validity of the data for the analysis at hand is usual when collecting data in industrial settings. Our goal here is to make sure we use relatively clean, valid data to identify significant inspection effectiveness factors. The heuristic we used is rough but appeared to be effective at getting cleaner relationships. These inspections detecting zero defects should be carefully investigated as they may be the symptom of a problem.

5.1 Code Inspections

Table 2 provides common descriptive statistics of the investigated variables: mean, standard deviation, maximum value, 75% percentile, median (i.e., the 50% percentile), 25% percentile, minimum value. Data was collected for 237 observations (code inspections) but, as discussed above, one outlier was left out during the statistical analysis. Regarding our categorical variables, 27% of the code inspected was in assembler and the remainder in the programming language CHILL. The proportions for "comment", "intensive", and default (standard checklists) inspections were 43%, 40%, and 17%, respectively. This data, however, was deemed unreliable by the quality management team. This may also explain why we did not find that variable significant in the analysis.

Measure	Mean	StdDev	Max	P75	Med.	P25	Min
Defects	11.7805	17.6198	130	13	6	2	1
Particip	4.66244	2.42427	17	5	4	3	1
Effort	1221.77	1670.65	14100	1560	630	270	30
Effpart	255.365	303.202	3000	310	180	75	7.5
Loc	13197.9	65962.4	94500	10000	3200	1236	30
Dloc	1567.89	3403.83	25029	1624	300	90	4
Iloc	2971.73	5442.24	39188	2830	806	300	10
Rate	7.19578	21.3471	130.626	3.33333	1.251	.4166	.0059
Sessions	1.4641	0.9135	6	1	1	1	1

Table 2: Descriptive Statistics for Code Inspections

As discussed above, in order to prevent the use of independent variables being strongly collinear, we run a Principal Component Analysis (PCA) on all the variables in Table 2. Table 3 presents six principal components (PC1 –PC6) that explain 97% of the variance in the data set. In each principal component, the coefficients associated with each variable (loading), represents its contribution to the principal component. Variables showing high loadings in the same principal component tend to be strongly correlated and may be seen as capturing the same underlying "concept" or dimension. The reader is referred to [5] for more details.

From these principal components, we can see that Effort and Effpart are strongly correlated (high loadings in PC1). This is also the case of Iloc and Rate (high loadings in PC2). In the model below, we will only allow Effort and Rate to enter and leave Effpart and Iloc out. The selection of one high loading variable over the other in PC1 or PC2 does not strongly affect the goodness of fit of the regression model to be built.

	PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**
Eigen-Value	2.6398	2.0142	1.0601	0.8361	0.7115	0.5263
Percent	32.9976	25.1775	13.2514	10.4512	8.8933	6.5792
CumPerc	32.9976	58.1751	71.4265	81.8776	90.7710	97.3502
Particip	-0.099985	-0.098317	**-0.974378**	0.0236156	-0.124721	-0.036022
Effort	**-0.851037**	-0.020853	-0.42085	0.1048765	-0.175142	-0.074648
Effpart	**-0.955408**	-0.014007	0.0962267	0.1030884	-0.075809	-0.169734
Loc	-0.137135	0.0642781	-0.030225	**0.9807216**	-0.026941	-0.115147
Dloc	-0.195486	0.2713266	-0.050201	0.1308865	-0.047655	**-0.927875**
Iloc	-0.142954	0.8857935	**0.0472498**	0.1201449	-0.126869	-0.312002
Rate	0.1514296	0.9613718	**0.0776791**	-0.013592	-0.004964	-0.050908
Sessions	-0.15739	0.0894070	-0.135046	0.0271967	**-0.972457**	-0.045606

Table 3: Rotated Principal Components for Code Inspections

Using the variables selected based on PCA, a multivariate regression analysis was run, using a backward variable selection procedure. The obtained (log-linear) regression model for code inspections has the following form

$$\ln(defects) = a_o + a_1 \ln(effort) + a_2 \ln(rate)$$

Since Effort is selected as a covariate, using Iloc instead of Rate would have led to a fully equivalent model and identical goodness of fit[7]. Using Effpart instead of Effort would have lead to a moderately lower fit. So we are confident to have achieved the best fit possible with the data available. Estimation statistics

[7] This is due to the fact that Rate is defined as Iloc over Effort. Thus, the equivalence is easy to demonstrate.

for the estimated coefficients a_i are shown in Table 4. For each coefficient, we provide: its estimate, the standard error of the estimate, the t-ratio of the coefficient, the statistical significance of the coefficient (i.e., the probability that the coefficient is equal to zero), and the standardized beta coefficient. The standardized beta coefficient characterizes the change in the dependent variable in terms of standard deviations, if the corresponding independent variable is changed by one standard deviation and all other variables are held constant [5]. Thus, the relative impact of the independent variables can be assessed.

Coefficient	Estimate	Std Error	t Ratio	Prob>|t|	Std Beta
a_0	-3.1360	0.3112	-10.07	<.0001	0
a_1	0.7522	0.0470	15.99	<.0001	0.7759
a_2	0.2445	0.0332	7.37	<.0001	0.3573

Table 4: Estimation Statistics for Code Inspections' Regression Model

The fit of the model is characterized by the statistics shown in Table 5. This table provides the coefficient of determination (R^2), the adjusted R^2 (which accounts for the increased number of independent variables in a multivariate regression model [5]) for the multivariate model described above. We can also compute the R^2 and adjusted R^2 in the normal domain, that is considering *Defects* instead of *ln(Defects)*. This is obtained by performing a regression between the number of defects and the predicted number of defects and then adjusting the resulting R^2 by using standard adjustment formula [5]. The results tell us that our regression model explains little more than 50% of the variance in number of defects. This means that important effectiveness factors are still not captured by our data collection.

R^2	Ad-justed R^2	R^2 Normal Domain	Adj. R^2 Normal Domain
0.5258	0.52	0.56	0.56

Table 5: Goodness of fit for Code Inspections

The relative error of the model is shown in Table 6. We report the mean and median values of both the magnitude relative error (MRE) and the absolute error (ARE). This tells us that 50% of the predictions show a relative error of 50% or above and an absolute error of approximately 3 defects or more.

	MRE	MRE'	ARE
Mean	0.78	0.76	6.59
Median	0.51	0.51	2.72

Table 6: Relative Error for Code Inspections

In this model, the inspection effort and rate (or inspected size as they cannot be differentiated) of the document show a significant impact on the number of defects detected. Estimated regression coefficients are significantly lower than one and thereby confirm that the relationships are not linear and show dis-

economies of scale for effort. A straightforward interpretation is that more effort or a higher rate is spent on inspections allows the inspectors to obtain a more thorough understanding of the document, thus resulting in the detection of more defects. The impact of the inspected document size can simply be explained by the fact that, if more of the document is inspected, assuming a somewhat constant defect density, more defects are to be detected. It is important to note that the log-linear model suggests that the number of defects detected does not grow proportionally to effort, rate or inspected Locs. There are several possible interpretations for this. For example, as reported in [18], when inspected document size grows there are fatigue effects resulting in lower effectiveness to find defects.

To compare the relative impact of Effort and Iloc (or Rate) we consider the standardized beta coefficients. They show the change, in terms of standard deviation, of the number of defects when the independent variable changes by one standard deviation. Since the standardized beta coefficients for (log)Effort is the highest (even when taking into account standard estimation error), effort is clearly the most important driver for the code inspection effectiveness. As a first priority, we therefore recommend making sure a sufficient amount of time is scheduled for the inspections in order to reach adequate inspection effectiveness. We will see below that further analysis with MARS will allow us to be more precise in determining optimal inspection rates.

5.2 Analysis Inspections

Table 7 provides the descriptive statistics for each variable based on 177 observations (analysis inspections). However, in the remainder of the analysis, and as discussed above, one outlier will be left out. As previously mentioned, theclassification of inspections as "comment", "intensive", and default (standard checklists) was deemed unreliable by the quality management team and did not show to be a significant factor. We also tried to classify the documents inspected according to different types but, again, it did not turn to be a significant effectiveness factor.

Measure	Mean	StdDev	Max	P75	Median	P25	Min
Defects	18.0451	19.5197	111	22	12	5	1
Particip	12.1299	8.37949	54	17	10	6	1
Effort	1305.59	1840.35	12600	1500	720	360	12
Effpart	111.147	128.651	900	127.5	72.2699	45	2
Totpage	39.5480	57.7933	526	56	15	9	2
Inspage	37.7457	57.1706	510	45	13	9	2
Rate	.116130	.299292	2.349	.06666	.0238	.010569	.000971
Sessions	1.22598	1.23159	10	1	1	1	1

Table 7: Descriptive Statistics for Analysis Inspections

Like for code inspections, we run a Principal Component Analysis (PCA) on the variables in Table 7. Rate and Totrate are strongly correlated, as well as Effort and Effpart (Like for code inspections), and Totpage and Inspage. As for code inspections, we will allow only a subset of these variables to enter the model to prevent strong collinearities among model covariates: Rate, Effort, and Inspage.

	PC1	PC2	PC3	PC4	PC5
EigenValue	2.9897	1.9903	1.3025	0.8915	0.6947
Percent	37.3718	24.8785	16.2816	11.1433	8.6837
CumPercent	37.3718	62.2503	78.5319	89.6751	98.3588
Particip	-0.018065	0.0841435	**0.9635523**	-0.157381	0.1664413
Effort	0.0225728	**0.9061068**	0.3686147	-0.079799	-0.036491
Effpart	0.0808347	**0.9621043**	-0.157936	-0.117316	-0.02408
Totpage	**0.9555919**	0.0771264	-0.072025	0.2205918	-0.036388
Insppage	**0.9540065**	0.0320833	0.0454971	0.2454602	0.0071750
Rate	0.2402744	-0.103891	-0.097406	**0.9592561**	-0.015228
Totrate	0.2473356	-0.103609	-0.113568	**0.9554223**	-0.019016
Sessions	-0.020249	-0.044734	0.1491591	-0.021114	**0.9871586**

Table 8: Rotated Principal Components for Analysis Inspections

Using the variables selected based on PCA, we run again a multivariate step-wise regression, using a backward selection procedure. This yields a (log-linear) regression model with the following form

$$\ln(defects) = a_o + a_1 \ln(effort) + a_2 \ln(insppage) + a_3 \ln(particip)$$

Using Totpage instead of Insppage would have lead, as expected, to a very similar goodness of fit and model. Like the model for code inspections, the fit statistics for the estimated coefficients a_i in the analysis model are shown in Table 9. All the model covariates show to be strongly significant.

Term	Estimate	Std Error	t Ratio	Prob>|t|	Std Beta
a_0	-2.1555	0.3246	-6.64	<.0001	0
a_1	0.5017	0.0575	8.72	<.0001	0.55
a_2	0.1498	0.0551	2.72	0.0072	0.14
a_3	0.3548	0.0956	3.71	0.0003	0.23

Table 9: Estimation Statistics for Analysis Inspections' Regression Model

Table 10 and Table 11 characterize, respectively, the goodness of fit and the relative error of the model for analysis inspections. We report the mean and median values of both the magnitude relative error (MRE) and the absolute relative error (ARE). Those results tell us, once again, that we manage to explain little more than 50% of the variance in number of defects. We can also see results that are comparable to code inspections in terms of predictive accu-

racy. 50% of predictions show a relative error of 46% or worse and an absolute prediction off by 5 defects or more.

R^2	Adjusted R^2	R^2 Normal Domain	Adj. R^2 Normal Domain
0.5494	0.5415	0.5374	0.5348

Table 10: Goodness of fit for Analysis Inspections

	MRE	MRE'	ARE
Mean	0.7687	0.6931	8.9848
Median	0.4555	0.5242	5.290

Table 11: Relative Error for Analysis Inspections

In this model, inspection effort, the total (or inspected) number of pages, and the number of inspection participants show a significant impact on the number of detected defects. Once again, using a J-test [3], it can be shown that the most plausible model is log-linear. Similar to the results in code inspections, higher inspection effort and larger (inspected) documents show larger number of defects detected.

Interestingly, as opposed to code inspections, the number of inspection participants is a significant driver for analysis inspection effectiveness. This may be explained by the fact that in analysis inspections, a larger variance in the number of participants can be observed. As a consequence, this variable shows an impact that is not visible in code inspections. This is visible when looking at Table 2 and Table 7. One tentative explanation, confirmed by the quality management team, is that analysis inspections cover a broader scope and involve many more interfaces with other documents that code artifacts. A higher number of inspectors increases the probability that the right expertise is present and results in a significant impact on inspection effectiveness.

The standardized beta coefficients show, consistent with the result for code inspections, that the inspection effort is the main driver for the number of detected defects. Based on this result we can recommend, once again, that management ensure a sufficient amount of time to be scheduled and was used for the preparation and execution of Analysis inspections. This confirms the well-known observation that inspection planning is key to the success of introducing systematic inspections in an organization. Using predictive models may help making such a planning more accurate. This recommendation will be made more precise in the next section, using the MARS analysis technique.

6 MARS Analysis

We present below the results obtained when performing a MARS analysis to our inspection data. We first start with code inspections and then present similar results for analysis inspections.

6.1 Code Inspections

We will allow the same predictor variables as in the log-linear models to enter the MARS models, making sure no strong collinearities are present among variables that can enter the MARS model. We will first provide the models built by the MARS procedures and their validation results. Then, we will provide an interpretation of the modeling results.

6.1.1 Model Building and Validation

The basis functions identified by MARS search algorithms are described in Table 12. We can right away identify a number of interaction effects and, in particular, interactions between effort (as captured by basis function BF1) and a number of other variables. Such interactions can be seen in the table when basis functions are part of the definition of other basis functions, e.g., BF1 in BF4. Because of many such interactions, Table 12 suggests that the model is far from being additive and that interactions will play an important role in building an accurate model for code inspections. Models and interactions will be further discussed below.

Basis Functions
BF1 = max(0, EFFORT - 30.000);
BF4 = max(0, PARTICIP - 10.000) * BF1;
BF9 = max(0, DLOC - 3000.000) * BF1;
BF11 = max(0, SESSIONS - 4.000) * BF1;
BF14 = max(0, 0.250 - RATE) * BF1;
BF19 = max(0, EFFORT - 3600.000);
BF21 = max(0, EFFORT - 2100.000);
BF24 = max(0, 4200.000 - EFFORT);
BF25 = max(0, PARTICIP - 9.000) * BF24;
BF27 = max(0, LOC - 29.999) * BF19;
BF28 = max(0, DLOC - 4.000) * BF21;
BF30 = max(0, 0.433 - RATE) * BF21;

Table 12 MARS Basis Functions for Code Inspections

Table 13 provides a ranking of the variables by order of importance. Variables with no impact at all are not shown. As described above, this is computed based on the reduction in goodness of fit when the variable is removed (i.e., all the basis functions involving the variable are removed). The loss in GVC is denoted as "-gvc" in Table 13. The column "Importance" shows the relative importance (percentage) of variables as compared to the best one (i.e., effort here). Confirming the regression analysis results, we can see that Effort is by far the most important variable determining defect detection effectiveness. To a lesser extent the change delta in terms of lines of code (Dloc), the number of lines of code of the artifact inspected (Loc), the rate at which inspections are taking place (Rate), and the number of sessions (Sessions) have also a significant effect on defects detected. These results are rather intuitive as the larger the amount of code modified and inspected, the larger the number of defects detected. The impact of inspection rate has been mentioned in a number of articles [1] [6] [2] and is confirmed by our analysis. We will get back to these issues later in this section.

Variable	**Importance**	**- gcv**
EFFORT	100.000	312.942
DLOC	62.006	178.804
LOC	45.308	139.755
PARTICIP	39.235	128.567
RATE	38.770	127.775
SESSIONS	12.632	98.497

Table 13: Relative Variable Importance for Code Inspections

Table 13 shows that the two significant defect detection predictors (Effort, Rate) were already selected in the log-linear regression model. The MARS model is essentially a richer model in the sense that it models additional effects (Dloc, Loc, Participants, Sessions) and automatically identifies relevant interactions. This is what we would expect from such a data mining procedure: to uncover additional information from the data as no restrictive assumptions are made regarding the model's functional form or interactions. We will see below how the goodness of fit and predictive capability improved as a result from using additional covariates and a different functional form for the regression equation. Though the regression model presented in Table 14 is more complex than the log-linear regression model, the number of covariates (9) is still very reasonable as compared to the number of observations (236). We want to ensure that the model generated is stable and will be accurate over other datasets. A typical rule of thumb is to have 10 data points minimum for each covariate in a regression model. MARS parameters have to be set to avoid overfitting with a too large number of covariates, i.e., basis functions. As mentioned in [4], this is relatively easy with the recent MARS tool but is outside the scope of this paper.

N	236	**R^2**	0.793
Mean Dep. Variable	11.754	**R^2 Adjusted**	0.785

PARAMETER	**ESTIMATE**	**S.E.**	**T-RATIO**	**P-VALUE**
Constant	2.218	0.807	2.748	0.006
Basis Function 1	0.008	.71E-03	11.453	.99E-15
Basis Function 4	0.003	.29E-03	10.435	.99E-15
Basis Function 9	.34E-05	.22E-06	15.282	.99E-15
Basis Function 11	-0.005	0.001	-4.400	.16E-04
Basis Function 14	-0.037	0.007	-5.441	.13E-06
Basis Function 25	-0.003	49E-03	-5.375	.19E-06
Basis Function 27	.91E-06	.82E-07	11.051	.99E-15
Basis Function 28	-.46E-05	.42E-06	-10.786	.99E-15
Basis Function 30	-0.033	0.006	-5.549	.80E-07

F-STATISTIC	96.085	**S.E. OF REGRESSION**	8.194
P-VALUE	999E-15	**RESIDUAL SUM OF SQUARES**	15172.826
[MDF,NDF]	[9, 226]	**REGRESSION SUM OF SQUARES**	58056.920

Table 14: Results from Ordinary Least Squares Regression

We can see from Table 15 that the coefficient of determination R^2 (or rather the adjusted R^2, which is adjusted for the number of covariates) of the MARS regression (0.785) significantly outperforms the log-linear regression model (0.559). Table 15 summarizes the comparison of goodness of fit of the two models. Several measures are presented as all provide valid insights into goodness of fit. R^2 is informative about the percentage of defect variance explained by the regression models. Table 15 also shows the results regarding the relative error of the model.

	R^2 (adjusted)	**Mean ARE**	**Mean MRE**	**Mean MRE'**
MARS model	0.78	5.33	1.05	0.58
Log-Linear model	0.56	6.49	0.78	0.76

Table 15: Comparison of goodness of fit for code inspection models

The R^2 clearly shows that the MARS model yields a better goodness of fit. A log-linear model gives more weight to observations with smaller actuals (due to the log transformation), i.e., they weigh more on the estimation of regression coefficients. Since smaller actuals tend to yield higher MREs, lower MREs resulting from log-linear models are in fact more of a mathematical artifact than an evidence of better goodness of fit. Furthermore, regression models optimize R^2, not relative error, and they should be compared on that basis. But it is also a well-known fact that comparing non-nested regression models with R^2 may be misleading [3]. A better way of comparing the plausibility of non-nested regression models is the J-test (See Appendix A for relevant details, [3] for a

complete description, and Appendix B for the results on our data set). This test confirms very clearly that the MARS model is more plausible than the log-linear model and is therefore more appropriate for interpretation purposes.

We also evaluated the predictive power of the log-linear model and the MARS model using cross-validation [16]. We randomly divided up the dataset (including the outlier as discussed before) into 10 subsets. Each subset was in turn used as a test set and the complementary set was used to refit both models. Thus, each observation in each subset was predicted using a model that was built on the other subsets. The goal is to obtain a more realistic picture of the predictive power of the models as the goodness of fit tends to give optimistic results.

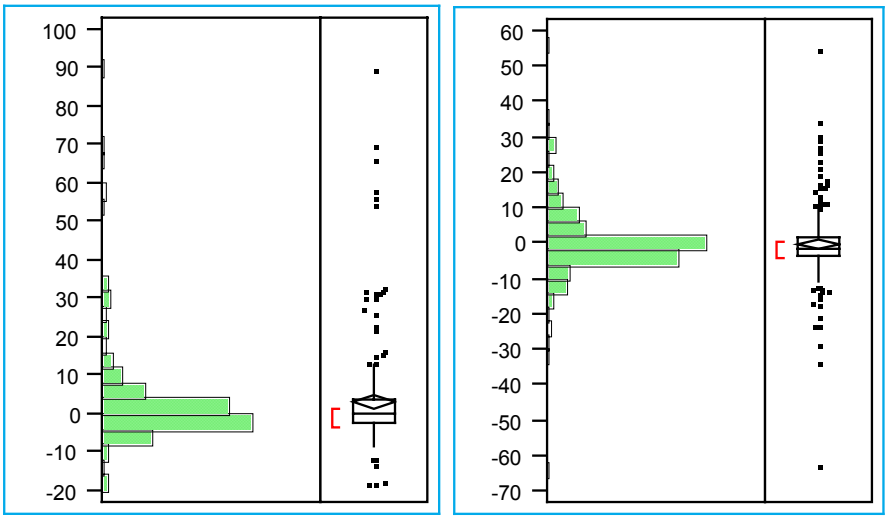
Cross-validation yields a R^2 of 0.532 and 0.647 between actual and predicted defects, for the log-linear and the MARS models, respectively. It is clear that a better goodness of fit is obtained with MARS, based on the exact same data. However, we can see that the MARS R^2 is, as expected, significantly lower that the goodness-of-fit R^2 (0.785) when running a cross-validation. This is not the case of the log-linear model, probably because it is based on less covariates (2 and 9, for the log-linear and MARS models, respectively) and therefore yields more accurate estimated regression coefficients. In general, we have to expect that MARS models show more covariates and this may be a significant drawback, depending on the data set size.

In addition, looking at the distributions of the difference between actual and predicted defects (Figure 2) shows clearly that the log-linear model is biased in the sense that it tends to underpredict the number of defects. For the log-linear model, residuals tend to be more often positive and larger in the upper part of the residual plot, as suggested by the distribution mean (3.47). This bias comes from the log transformation of the dependent variable[8]. The MARS model, on the other hand, does not show such a bias and clearly has smaller residuals. Depending on the application context of the models, bias may have serious practical consequences. For example, if defect predictions are used to plan the overall correction effort resulting from inspections, by summing up defect predictions across many analysis and code artifacts, then a biased model will result into an overall, grossly underpredicted effort and budget. On the other hand, with an unbiased model, artifact prediction errors would tend to cancel each other over all inspections and still result in an overall defect prediction of reasonable accuracy at the project level.

Regarding the difference between actual and predicted defects detected, the box plots in Figure 2 shows the 25^{th} and 75^{th} quantiles, also called quartiles.

[8] Jensen's inequality: For any random variable X, E(X) beings its expected value, if g(X) is a convex function, then $E(g(X)) \geq g(E(X))$

The line across the middle identifies the median sample value. The brackets along the edge represent the most dense 50% of observations. The diamond identifies the mean and the 95% confidence interval about the mean. We can easily see that a number of predictions are really far off and further investigation would be required to determine the cause of such inaccuracies. This could lead to the identification of new factors affecting inspection effectiveness.



a) mean = 3.47 b) mean = 0.12

Figure 2: Quantile Box Plots: actual – predicted defects for (a) log-linear model and (b) MARS model

Based on the discussions above, the MARS model seems to provide a better basis for interpreting the impact of various factors on code inspection effectiveness. This is discussed below where we focus our attention on interactions between factors, as modeled by the MARS model, and their implications in terms of decision making and gaining a better understanding of the code inspection process.

6.1.2 Model Interpretation

To help interpreting the models, we visualize 2-way interactions between independent variables. Figure 3 is a typical example of two-way interactions that can be observed by graphical means from a MARS model. This figure represents the model predicted surface for the dependent variables (i.e., number of defects detected) when only considering the interaction effect of two variables. In other words, the 3-D graph captures only part of the model's effect, i.e., the interaction effect of two variables that contributes to the final model defect prediction. More precisely, in Figure 3, the "contribution" axis is in this case -

0.037*BF14 - 0.033*BF30 and the other axes are simply the variables involved in the interaction terms: Rate and Effort. Values on the contribution axis are shifted by the MARS tool, so that the minimum value is 0. Color codes represent different contribution value intervals. Though the absolute values on the contribution axis are not easy to interpret and of interest here, the shape of the surface modeling the interaction allows us to better understand how the effectiveness factors interplay. Note that the MARS tool only displays the part of the space that is populated by observations and some surfaces may appear truncated.

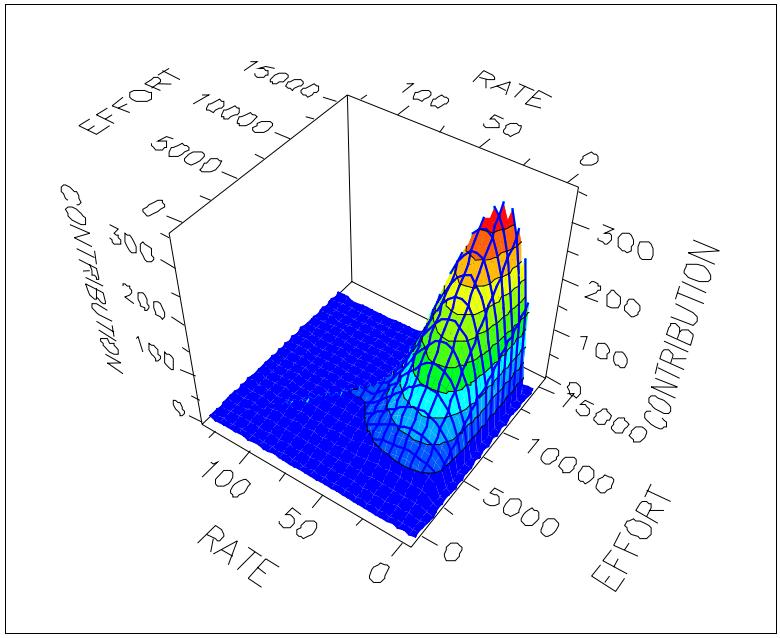


Figure 3: Interaction between Rate and Effort

A grid is also drawn to help the reader get a better sense of the modeled surface in three dimensions. Higher-level interactions may be present in the data, but they are difficult to visualize and interpret. We will not investigate them here. From Figure 3, we can see that the interaction of Rate and Effort indicates an interval (roughly between 25 and 50 lines per hour) on the inspection rate measurement scale that is optimal, i.e., an increase in effort provides a higher pay-off in that interval. It would therefore be desirable that inspection rates remain in that interval for maximum return on investment. However, in practice, other factors may come into play, like the time and people actually available to perform an inspection on an artifact to be certified before a deadline.

It is now interesting to compare our optimal rates with the existing literature. Ebenau [6] reports an optimal rate of 150 lines per hour. It is however reported that the rates were not planned and the smallest observed rates were slightly

above 50 lines per hour. Moreover, he simply used a linear regression between rate and defect density to estimate this "optimal" rate. It is worth noting that the resulting correlation coefficient of this regression was really weak and its significance mainly due to two outliers. Another major difference with our inspections is that the programming language inspected was C. Gilb et al [18] found, based on numerous experiences, that effective individual inspections of software development task products usually lies between 0.5 to 1.5 pages an hour, where a page is roughly around 50 lines. He writes that, though such slow rates shock people initially, we have to remember that inspections include cross-checking against check-lists and other documents and may involve several passes where inspectors focus on different types of faults. In light of the existing reported rates, and considering their level of uncertainty, the optimal rate determined by the MARS analysis is within plausible range.

Ebenau [6] notes that we should expect optimal rates to vary across organizations and inspections as the type of work products, their complexity and size, and the expertise of the inspectors vary significantly. It is therefore important for organizations not to rely on published rates to plan their inspection but to identify their own, optimal rates. MARS can be of assistance in doing this.

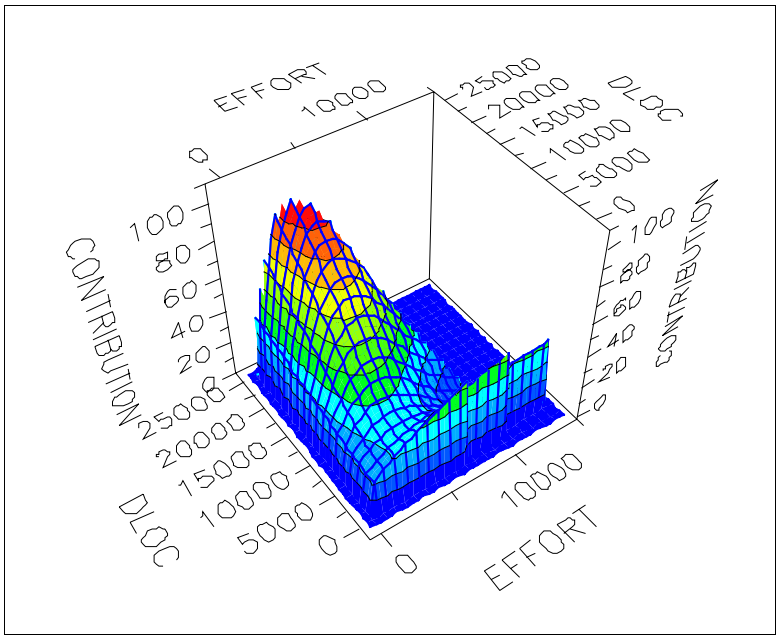


Figure 4: Interaction between DLOC and Effort

In Figure 4 we can see that for a given effort value, the higher DLOC (i.e., LOC changed, added, or deleted), the more defects, and therefore the higher the impact of effort on defect detection. This is visible by observing how the colored contours change as DLOC increases. This is a plausible trend: the larger the change, the higher the number of defects introduced, the more defects

detected for a given effort value. This result reinforces the evidence that the MARS model is a plausible one. Also, one may predict, based on the amount of change in the code, the number of resulting defects to be expected and derive the resulting correction effort. Decisions to implement changes and plan change effort can be based on impact analysis (to estimate Dloc) and the use of the MARS model.

Another interesting observation is that as effort increases, the impact of DLOC on the number of defects increases up to a point, and then decreases. Though not everything can be explained in such an exploratory data analysis, this might reflect the fact that when a certain amount of effort–which is strongly related to effort per participant in our case– goes below or over a certain threshold (roughly below or above 2000 or 6000 man minutes, respectively), inspections tend to be less effective. In the former case, people may not have the time to really understand the documents they are inspecting and find fewer defects. The latter case is somewhat more complex to understand (e.g., may be due to fatigue effects in reported in [18]), is based on fewer observations, and should be the object of further enquiry. A strange surface shows up for high effort values and low DLOCs, but it very likely spurious as we have relatively few observations in that area. That might be, for example, the result of poor quality data collection or an idiosyncrasy of the MARS method, but it is hard to say. In general, regardless of the method used for exploratory analysis, we cannot expect to explain all observed trends at the smallest level of detail.

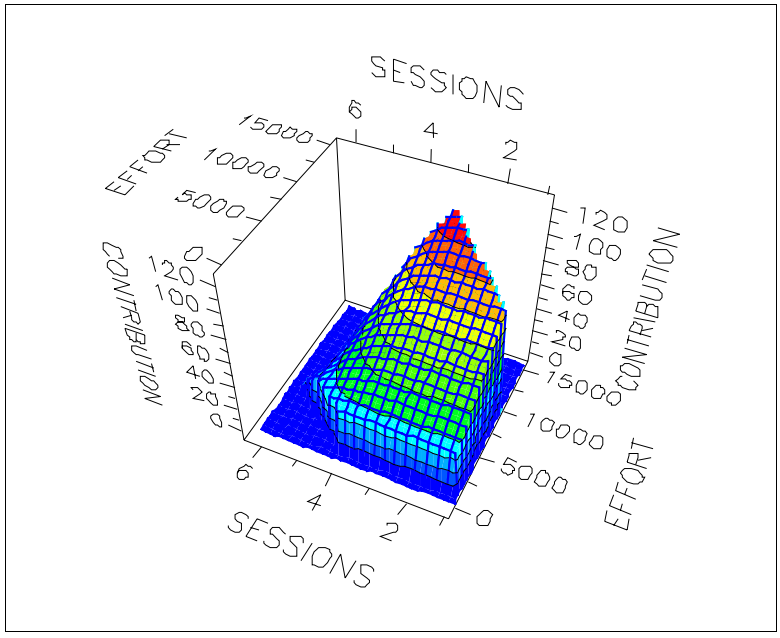


Figure 5: Interaction between Sessions and Effort

Figure 5 shows that when the number of sessions goes beyond a certain point, holding the effort constant, effectiveness starts to decrease. In this environment, an inspection is often broken down into several sessions that focus on different parts of the code to be inspected. It is then possible that, below a certain level, the break down affects the effectiveness of the inspection. From this figure, a maximum of three sessions should probably be recommended to be on the safe side.

From Figure 6, we see that when the number of participants increases, more defects are found at a constant effort level. This is visible from looking at how the color contours bend downward as the number of participants increases. On the range we can observe in our dataset (1 to 17), increasing the number of participants seems to increase inspection effectiveness. As mentioned before, this is likely due to the fact that with a larger number of participants, the inspection is more likely to include people with the right expertise. Now, a cost-benefit analysis, including the cost of additional participants in the equation, would be necessary to make useful recommendations.

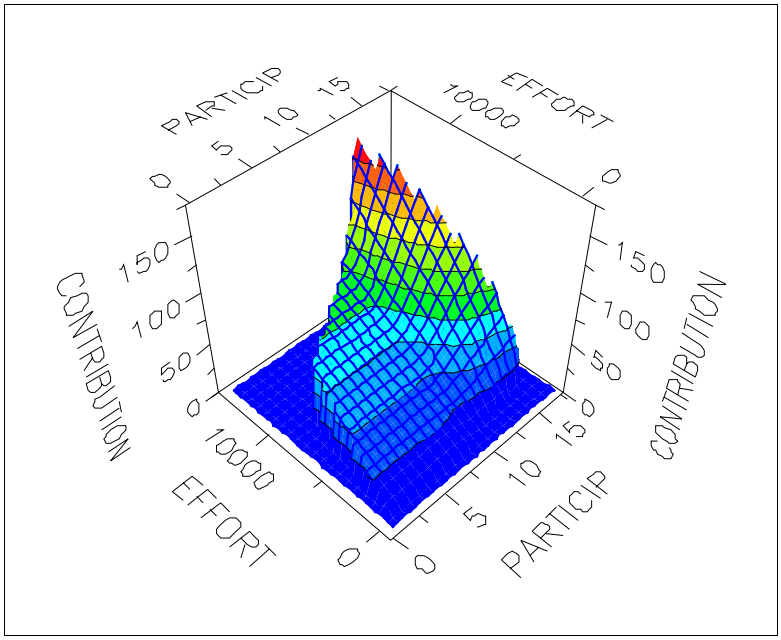


Figure 6: Interaction between Participants and Effort

For looking at the interactions in the figures above, we can first conclude that interactions seem to be key in modeling the effects of typical inspection factors on code inspections. A model that does not account for such interactions, such as log-linear regression models, is likely to be inadequate, as supported by the cross-validation results above. Common sense can also provide us with relatively straightforward explanations for many of these interactions.

It is interesting to note that our MARS model for code inspections supports some previous empirical results and hypotheses in the inspection literature. As expected and reported by Christenson et al [2], inspection effort plays also a very important role. Both the document and change size of the code (Loc, Dloc) plays here a significant role and this is similar to what both Christenson et al. and Ebenau [6] reported. Ebenau estimates inspection size by considering new and changed material and the interfaces to other work products. That latter attribute is missing in our dataset and should probably be part of our future data collection. The number of participants and sessions also shows a significant impact as in the work by Porter et al [11]. The inspection rate shows to be a significant factor as reported by Ebenau [6] and Gilb et al [18].

We have seen above that many of the interactions modeled by MARS may be extremely useful to gain understanding about the inspection process and provide decision support, e.g., to decide about the number of sessions (Sessions), the inspection rate (Rate), or the number of participants (Particip). Such a model is therefore not only useful for planning purposes (e.g., inspection and correction effort planning) but also to help management decisions regarding inspections' effectiveness. The results presented here cannot be systematically generalized to other environments. But similar data collection and analysis procedures should help every organization make its own, optimal decisions.

6.2 Analysis Inspections

Like for the log-linear model, we only allow a subset of the variables to enter the model in order to prevent strong collinearities among them. We first provide the resulting MARS models, their validation results, and provide an interpretation of the results.

6.2.1 Model Building and Validation

We present in this section the MARS results for analysis inspections. Results are reported in an identical fashion to code inspections. From Table 16, we can see that, like for code inspections, inspection effort and rate are key predictors. But their respective importance shows a different pattern from what we observed for code inspections. One possible explanation is that, when reading analysis documents, one will be more likely to miss defects than when reading code if preparation is performed at a too high rate. The quality management team confirmed that analysis documents were more complex and involved numerous interfaces with other documents. As Ebenau [6] reported, interfaces play a significant role in the complexity of inspections.

Rate shows the highest importance rank whereas Effort is second to last. Number of sessions and participants are also among the main predictors of de-

fect detection, like for code inspections. The number of inspected pages has also an important impact. Since this is strongly correlated to the total number of pages, this result is therefore comparable to the high impact of LOC in code inspections. Overall, results are similar across the code and analysis models in terms of the variables selected, though the ranking of variables is sometimes different.

Variable	Importance	-gcv
RATE	100.000	384.534
INSPAGE	63.146	279.110
PARTICIP	59.626	271.531
EFFORT	45.534	245.549
SESSIONS	36.528	232.590

Table 16: Relative Variable Importance for Analysis Inspections

Basis Functions
BF1 = max(0, EFFORT - 2970.000);
BF2 = max(0, 2970.000 - EFFORT);
BF3 = max(0, RATE - .970883E-03) * BF1;
BF4 = max(0, SESSIONS - 1.000) * BF1;
BF5 = max(0, INSPPAGE - 42.000);
BF6 = max(0, 42.000 - INSPPAGE);
BF9 = max(0, PARTICIP - 22.000);
BF11 = max(0, PARTICIP - 23.000) * BF2;
BF13 = max(0, PARTICIP - 7.000) * BF5;
BF17 = max(0, PARTICIP - 5.000) * BF2;
BF26 = max(0, 0.100 - RATE);
BF30 = max(0, PARTICIP - 1.000) * BF26;

Table 17: MARS Basis Functions for Analysis Inspections

Based on Table 18, we can see that the MARS regression fit for analysis inspections is not nearly as good as for code inspections (adjusted R^2: 0.645 versus 0.785). One plausible reason is that precise size measures do not currently exist for analysis documents. There is, therefore, some size effects that are very plausible but not accounted for in our model. The fact that the dataset is smaller for analysis inspections than for code inspections may also partly explain why MARS does not perform as well. Simulations in [4] have shown that complex modeling techniques such as MARS and neural networks reach their full potential when larger datasets are being used[9]. For smaller datasets[10], because the sample space is sparsely populated, these techniques have difficulties

[9] In [4], Deveaux et al. used datasets of 250 and 1000 observations

[10] That is 50 observations in the study of Deveaux et al, which is significantly smaller than our dataset for analysis inspections. But 176 observations is still lower than the dataset sizes (250, 1000) where they show MARS to perform well.

identifying complex patterns in the data, and a simple log-linear regression model may be a good enough approximation.

The regression model based on the basis functions shown above is presented in Table 18. Again, one can notice that we set up the MARS tool tuning parameters so that we have a number of covariates (9 basis functions) that is reasonable when compared to the number of observations (176). Recall a typical rule of thumb when fitting a regression model is to have 10 data points minimum for each covariate.

For the reasons discussed above, the MARS model for analysis inspections shows a more moderate improvement in goodness of fit than for code inspections: adjusted R^2 of 0.645 versus 0.532, for the MARS and log-linear models, respectively. Fit statistics comparing the two models for analysis inspections are provided in Table 19. Running a J-test shows (cf. Appendix B) that the MARS model is more plausible than the log-linear model. The MARS model is therefore more appropriate for interpretation.

N	176	**R^2**	0.663
Mean Dep. Variable	18.125	**R^2 Adjusted**	0.645

PARAMETER	**ESTIMATE**	**S.E.**	**T-RATIO**	**P-VALUE**
Constant	19.58	2.268	8.635	.47E-14
Basis Function 3	0.835	0.177	4.727	.48E-05
Basis Function 4	0.028	0.005	5.464	.16E-06
Basis Function 5	-0.088	0.024	-3.643	.36E-03
Basis Function 6	-0.506	0.074	-6.834	.15E-09
Basis Function 9	-6.970	1.005	-6.937	.85E-10
Basis Function 11	0.005	.93E-03	4.886	.24E-05
Basis Function 13	0.032	0.004	7.397	.65E-11
Basis Function 17	-.53E-03	.10E-03	-5.013	.13E-05
Basis Function 30	19.244	1.953	9.852	.99E-15

F-STATISTIC	36.3	**S.E. OF REGRESSION**	11.649
P-VALUE	0.99E-15	**RESIDUAL SUM OF SQUARES**	22526.5
[MDF,NDF]	[9, 166]	**REGRESSION SUM OF SQUARES**	44334.6

Table 18: Results from Ordinary Least Squares Regression

	R^2 (adjusted)	**Mean ARE**	**Mean MRE**	**Mean MRE'**
MARS model	0.65	8.16	1.07	0.80
Log-Linear model	0.53	8.98	0.77	0.69

Table 19: Comparison of goodness for analysis inspection models

A cross-validation, identical to the one presented for code inspections, shows that the difference in predictive ability between the two models is negligible.

Using a non-parametric correlation measure between predicted and actual defects, Spearman Rho is 0.68 and 0.67 for the log-linear and MARS models, respectively. Furthermore, differences in ARE are small.

The predictive power is expectedly worse than for code inspections. The MARS model, as for code inspections, shows a larger decrease in goodness of fit when running a cross-validation. The most plausible explanation, as mentioned above, is the larger number of covariates of the MARS model (9 and 3, for the MARS and log-linear models, respectively). Consequently, MARS is expected to yield less accurate estimated regression coefficients on a data set of this size. To conclude, the MARS model is not likely to be of much help in terms of building a more accurate predictive model for smaller data sets, unless the number of covariates is significantly smaller, i.e., less significant basis functions.

6.2.2 Model Interpretation

Let us now turn our attention to the interpretation of the MARS model for analysis inspections. Figure 7 shows the effect of number of participants (Particip) for different effort levels. The main result is that beyond a certain number of participants (roughly 20), inspection effectiveness falls sharply. This seems contradictory with code inspection results. However, the range for the number of participants is very different here as compared to code inspections, where the maximum of participants was 17. Such a graph can be used to limit the number of participants to reach optimal effectiveness.

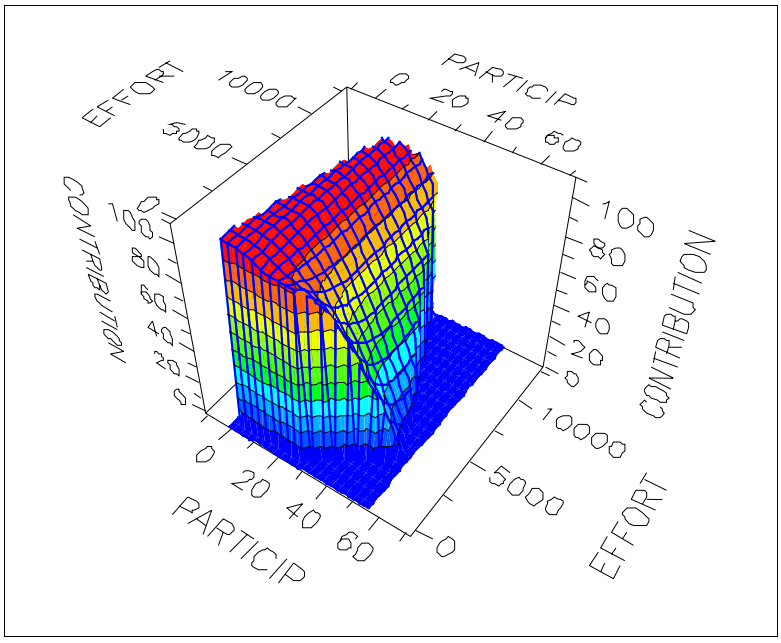


Figure 7: Interaction between Effort and number of participants

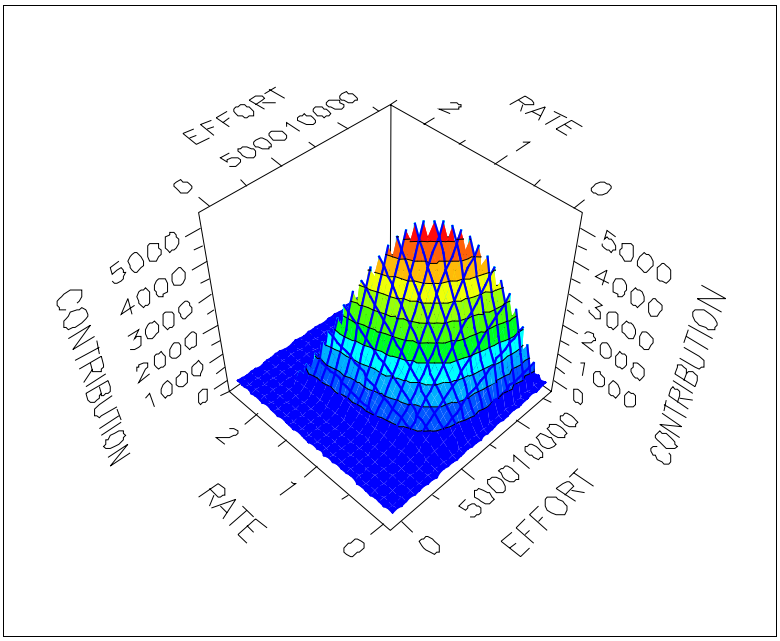


Figure 8: Interaction between Rate and Effort

From Figure 8, we can see that, like for code inspections, inspection rate (i.e., pages inspected per hour) has a strong effect on the impact of effort on defect detection. For analysis inspections to show an optimal pay-off, the inspection rate must be around one page inspected per hour. In the literature, Ebenau [6] reports a rate of 5 pages per hour for analysis documents and Gilb et al. [18] report effective rates to be between 0.5 and 1.5 pages per hour. Our MARS-computed optimal rate is therefore not implausible considering reported rates. But again, for the reasons discussed above, optimal rates are likely to be context-dependent and they should be determined within each organization and for each type of document.

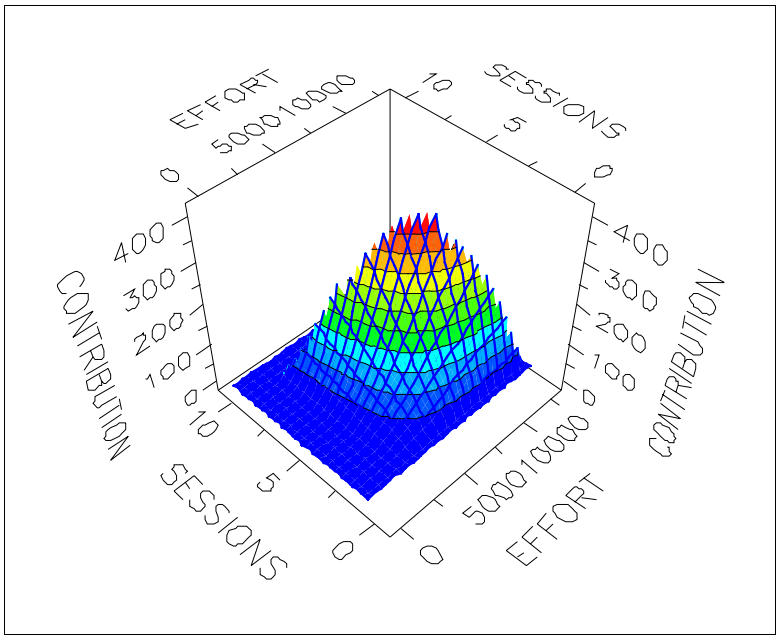


Figure 9: Interaction between Sessions and Effort

Similar to code inspections, there seems to be, based on Figure 9, an optimal range in terms of number of sessions. In that range, 5 to 6 sessions, inspection effort has a higher pay-off.

7 Conclusions

This paper has focused on investigating the impact of a number of inspection factors such as effort or inspection rate and their interactions. We investigated two kinds of inspections: analysis and code inspections. A substantial amount of data was collected and a recent, novel exploratory modeling technique was used to improve our understanding of the underlying structures in the data: Multivariate Adaptive Regression Splines (MARS). Additionally, this novel technique was compared to more conventional ways of building regression models.

The results have shown consistently, across the two types of inspections, that inspection effort and rates were very important defect detection drivers. Their effect is, however, interacting. That is, the impact of effort on effect detected is optimal within a certain code inspection rate (roughly 25 to 50 lines of code an hour) and analysis inspection rate (roughly one page an hour). In addition, for code inspections, we have seen that the size of changes, the size of inspected artifacts, and the number of participants all interact with effort and are also important to predict defects detections.

From a more general standpoint, MARS has helped us better understand and uncover the complex relationships and interactions that exist in inspection data. The MARS model turned out to be a richer model, accounting for more factors, than the linear model we developed. In most cases, simple intuitive explanations could be given for interactions. Though not every result can be readily explained, MARS has nevertheless shown to be a useful exploratory data analysis tool in our context. MARS also seems to be a potentially useful technique to obtain more accurate models than with standard linear regression analysis. However, the dataset has to be large enough and the underlying structure in the data has to warrant such an analysis: strong non-linearity and interactions must be present. For code inspections, the goodness of fit and predictive capability improvements brought by the MARS model are significant. There is, however, no gain in predictive capability for the analysis model. The analysis inspection dataset is significantly smaller than the code inspection dataset. Thus, in this case, a simpler log-linear model may be a good enough approximation. In summary, the ability of MARS to improve model building strongly depends on (1) having a dataset of sufficient size and (2) the presence of non-linear structures in the data. In the context of inspection data, intuition tells us that condition (2) is realistic (see results and discussions above). But fulfilling (1) depends on how many inspections take place in a given organization and its capability to collect data at a sufficient pace. Existing studies, as well as this paper, report thresholds in the realm of 200 observations or so, for MARS to bring a predictive advantage. Though informative, this should be only con-

sidered as an initial rule of thumb or an order of magnitude, until more empirical results are available in the specific context of inspections.

In terms of applications, the models we have presented here can be used for a number of purposes. First, with respect to planning, the number of defects can be predicted for a project's planned inspections. Change effort can then be derived and planned for. Second, in terms of quality control, management may use MARS results to determine reasonable inspection resources and numbers of participants, and rates, thus maximizing inspection effectiveness. Third, such models can help determine the importance of various factors, such as the reading technique employed or the inspection process followed. For example, we have seen, in our case study, that the inspection process followed did not have an impact once inspection effort was accounted for. Since intricate relationships exist between inspection factors and effectiveness, an exploratory technique such as MARS can help better exploit project inspection data.

Though some of the results here are common with other reported studies, they cannot be readily generalized to other environments. However, the type of data collection and analysis that was performed in this study can be reused in any environment where inspections need to be better understood and controlled. Our study, performed in a representative development environment, has shown that it was practically feasible to undertake such measurement and obtain useful, interpretable models. Furthermore, from a practical and subjective standpoint, the feedback we received from practitioners and quality engineers was clearly positive.

8 Acknowledgements

We would like to thank Mr. Karl for his crucial support during the case study. Dan Steinberg, from Salford Systems, has also been very helpful in answering crucial questions regarding the MARS tool. The graphs in this paper were generated using MARS, Version 1.0, from Salford Systems (www. salford-systems.com). Lionel Briand was partially supported by NSERC, the Canadian National Science and Engineering Research Council.

9 References

[1] F. O. Buck, Indicators of Quality Inspection, Tech. Report TR21.802, IBM Corp., 1995

[2] Dennis A. Christenson, Huang T. Steel, and Alfred J. Lamperez, Statistical Quality Control Applied to Code Inspections, IEEE Journal Selected Areas in Communication, vol. 8, pp. 196-200, Feb. 1990

[3] Davidson, R., McKinnon, J., Several Tests for Model Specification in the Presence of Alternative Hypotheses. *Econometrica, vol. 49, no. 3* (1981), 781-93.

[4] R. D. De Veaux, D. C. Psichogios, L. H. Ungar, A Comparison of two Nonparametric Estimation Schemes: MARS and Neural Networks, Computers Chemical Engineering, Vol 17, N 8, pp. 819-837, 1993

[5] W. Dillon, M. Goldstein, Multivariate Analysis: Methods and Applications, Wiley, 1984

[6] Robert G. Ebenau, Predictive Quality Control with Software Inspections, Cross Talk, The Journal of Defense Software Engineering, vol. 7, pp. 9-16, June 1994

[7] J. Friedman, Multivariate Adaptive Regression Splines, The Annals of Statistics, vol. 19, pp. 1-141, 1991

[8] Tzvi Raz and Alan T. Yaung, Factors affecting design inspection effectiveness in software development, Information and Software Technology, vol. 39, pp. 297-305, 1997

[9] Edward F. Weller, Lessons from Three Years of Inspection Data, IEEE Software, vol. 10, pp. 38-45, Sept. 1993

[10] Watts H. Humphrey, A Discipline for Software Engineering, Addison-Wesley 1995

[11] Adam A. Porter, Harvey Siy, and Lawrence G. Votta, A Review of Software Inspections, Software Process, Advances in Computers Series, Academic Press, Marvin Zelkowitz Editor, May 1996.

[12] Oliver Laitenberger and Jean-Marc DeBaud, Perspective-based Reading of Code Documents at Robert Bosch GmbH, Information and Software Technology, vol. 39, pp. 781-791, Mar. 1997

[13] Oliver Laitenberger and Jean-Marc DeBaud, An Encompassing Life-Cycle Centric Survey of Software Inspection, International Software Engineering Research Network (ISERN) Technical Report ISERN-98-14, Fraunhofer Institute for Experimental Software Engineering, 1997

[14] Adam A. Porter, Harvey P. Siy, Carl A. Toman, and Lawrence G. Votta, An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development, IEEE Transactions on Software Engineering, vol. 23, pp. 329-346, June 1997

[15] Priscilla J. Fowler, In-Process Inspections of Workproducts at ATT, ATT Technical Journal, vol. 65, pp. 102-112, mar 1986

[16] Weiss, S., Kulikowski, C. Computer Systems that Learn. *Morgan Kaufmann Publishers, Inc. San Francisco, CA*, (1991).

[17] Norman E. Fenton and Shari Lawrence Pfleeger, Software Metrics - A Practical and Rigorous Approach. International Thomson Computer Press, 2nd edition ed., 1996.

[18] T. Gilb, D. Graham, S. Finzi, Software Inspection, Addison-Wesley, 1993

10 Appendix A: The J-test

There are two kinds of situations in which one may wish to compare the plausibility of regression models. In the case of *nested* regression models, one model has a set of terms which is a subset of the other model's terms. In this case, the adjusted R^2 can simply be used as a means of comparison.

A second, more complicated case that we encounter in our study, is when the two models to be compared are not nested. As described in [3], non-nested models cannot be compared using the R^2 since this one is affected by the use of different variables, showing different spacing within the data. Therefore, as suggested by Davidson and MacKinnon [3], a series of tests can be used to test whether a given model is the most plausible among several alternatives.

Davidson and MacKinnon [3] propose a set of tests that are easier to use in different circumstances. It is recommended to use the J-test when testing the plausibility of linear (or linearized) models. The J-test is easier, more intuitive, and should yield identical results. When comparing two models, the J-test consists of performing the following regression:

$$y_i = (1-\lambda)\times f_i(X_i,\beta) + \lambda \times \hat{g}_i + \varepsilon_i ,$$

$\hat{g}_i = g_i(Z_i,\hat{\gamma})$, where $\hat{\gamma}$ is the estimate of γ ,

where

f and *g* are alternative models whose plausibility is tested, e.g., MARS and log-linear models in this paper

$$H_0 : y_i = f_i(X_i,\beta) + \varepsilon_{0i} ,$$

y_i is the i^{th} observation on the dependent variable, X_i is a vector of observations on independent variables, β is a vector of parameters to be estimated**,** and the error term ε is assumed to be normally distributed.

$$H_1 : y_i = g_i(Z_i,\gamma) + \varepsilon_{1i} ,$$

Z_i is a vector of observations on independent variables, γ is a vector of parameters to be estimated**,** and the error term ε is assumed to be normally distributed. Using the formula above, assuming we wish to test that f_i is the most plausible model, then we test whether λ is equal to zero. If this is the case,

then the alternative model is not needed to explain variations in effort. This test can be performed using the usual two tailed t-test based on the λ estimate and its standard error in order to determine whether the λ estimate is significantly different from zero. To test the plausibility of g_i, the two functions just have to be substituted in the formula above and the t-test performed again**.** If the two t-tests, for the two alternative models, tell us that λ is not significantly different from zero, then both models are plausible. If one t-test shows an λ value significantly different from zero, whereas the t-test for the alternative model does not, then the former model is less plausible than the latter one.

11 Appendix B: J-test Results

In this appendix the results for the J-Test are presented. The λ value discussed in Appendix A is highlighted. It can be easily observed that both for analysis and code, a t-test shows, when f is the log-linear model, an λ value significantly different from zero, whereas the t-test for the alternative model does not. Thus, in both cases the MARS models are more plausible than the log-linear ones.

11.1 Analysis Inspections

***f* = log-linear, *g* = MARS, Parameter Estimates**

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1.583685	0.385636	-4.11	<.0001
lnparticip	0.3261522	0.097406	3.35	0.0010
lneffort	0.3379471	0.082834	4.08	<.0001
lninsppage	0.0891194	0.060165	1.48	0.1404
lnMarsPred	0.2910245	0.114523	2.54	**0.0119**

***f* = MARS, *g* = log-linear, Parameter Estimates**

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	22.666649	3.550067	6.38	<.0001
BF3	1.0046601	0.232042	4.33	<.0001
BF4	0.0302259	0.005446	5.55	<.0001
BF5	-0.090794	0.024299	-3.74	0.0003
BF6	-0.582178	0.100219	-5.81	<.0001
BF9	-7.632173	1.163235	-6.56	<.0001
BF11	0.0049638	0.000997	4.98	<.0001
BF13	0.0348562	0.005122	6.80	<.0001
BF17	-0.000605	0.000127	-4.78	<.0001
BF30	23.094062	3.934433	5.87	<.0001
Log-linearPred	-0.262961	0.233303	-1.13	**0.2613**

11.2 Code Inspections

***f* = log-linear, *g* = MARS, Parameter Estimates**

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1.617007	0.436092	-3.71	0.0003
lneffort	0.3188617	0.101216	3.15	0.0018
lnrate	0.1454935	0.037463	3.88	0.0001
lnPredMars	0.6394232	0.132604	4.82	**<.0001**

***f* = MARS, *g* = log-linear, Parameter Estimates**

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	1.7286707	1.02576	1.69	0.0933
BF1	0.0072715	0.00145	5.01	<.0001
BF4	0.0030065	0.000292	10.31	<.0001
BF9	0.0000033	2.71e-7	12.11	<.0001
BF11	-0.005123	0.001166	-4.39	<.0001
BF14	-0.035416	0.007389	-4.79	<.0001
BF25	-0.002669	0.000497	-5.37	<.0001
BF27	0.0000009	8.727e-8	10.23	<.0001
BF28	-0.000004	4.738e-7	-9.38	<.0001
BF30	-0.031581	0.006453	-4.89	<.0001
Log-linearPred	0.1869056	0.241503	0.77	**0.4398**

Document Information

Title:	Using Multiple Adaptive Regression Splines to Understand Trends in Inspection Data and Identify Optimal Inspection Rates
Date:	January 1, 2001
Report:	IESE-062.00/E
Status:	Final
Distribution:	Public