Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods

Ganesh J. Pai, *Member*, *IEEE*, and Joanne Bechta Dugan, *Fellow*, *IEEE*

Abstract—We present a methodology for Bayesian analysis of software quality. We cast our research in the broader context of constructing a causal framework that can include process, product, and other diverse sources of information regarding fault introduction during the software development process. In this paper, we discuss the aspect of relating internal product metrics to external quality metrics. Specifically, we build a Bayesian network (BN) model to relate object-oriented software metrics to software fault content and fault proneness. Assuming that the relationship can be described as a generalized linear model, we derive parametric functional forms for the target node conditional distributions in the BN. These functional forms are shown to be able to represent linear, Poisson, and binomial logistic regression. The models are empirically evaluated using a public domain data set from a software subsystem. The results show that our approach produces statistically significant estimations and that our overall modeling method performs no worse than existing techniques.

Index Terms—Bayesian analysis, Bayesian networks, defects, fault proneness, metrics, object-oriented, regression, software quality.

1 Introduction

The notion of a good quality software product, from the developer's viewpoint, is usually associated with the external quality metrics of 1) fault (or defect) content, i.e., the number of errors in a software artifact, 2) fault density, i.e., fault content per thousand lines of code, or 3) fault proneness, i.e., the probability that an artifact contains a fault. To guide the software verification and testing effort, several measures of software structural quality have been developed, e.g., the Chidamber-Kemerer (C-K) suite of metrics [1], [2]. These *internal* product metrics have been used in numerous models which relate them to the *external* quality metrics [3], [4], [5], [6], [7], [8], [9]. Owing to the belief that a high quality software process will produce a high quality software product [10], there are also some models in the literature which relate certain process measures to fault content [11], [12], [13]. The main idea in many of these existing approaches is to build a statistical model that relates the product or process metrics to the quality metrics.

Although one intuitively expects a high quality software development process to yield a high quality product, there is very little empirical evidence to support this belief. There is also sufficient variation in the development process so that faults enter the software from diverse sources. Many of these sources do not yet have established measures to support their inclusion in existing models for quality assessment, so they are subjectively qualified, e.g., conformance of the executed process to a process specification, quality of the development team, quality of the verification process. Consequently, the existing software quality assessment methods are insufficient for including such sources. Furthermore, there does not yet seem to be a standardized set of process measures that have been empirically validated as significant for software quality assessment. Besides these issues, Fenton et al. have identified various shortcomings with existing approaches and indicated the need for a causal model for quality assessment [14], [15], [16], [17].

Thus, there is a need for both 1) empirically validating the relationship of process measures with external quality metrics and 2) building a repertoire of statistical models which can incorporate existing product and process metrics, as well as other sources of evidence that may have been subjectively qualified.

Now, we briefly provide the context which motivates the work described in this paper. One of the broad goals of this work is to build a framework for quality assessment where we use not only the available process and product measurements, but also the evidence available from the diverse sources influencing fault introduction. Elsewhere [18], we have developed such a framework using Bayesian networks (BN) [19], as shown in Fig. 1. In short, our idea is to

1. separately consider *product* measurements as one set of factors that influence software quality,
2. separately consider the available *process* measurements and subjectively qualifiable process properties as another set of factors influencing quality,
3. redefine quality as the likelihood of observing properties of the software product, e.g., fault content, fault proneness, reliability, and
4. build a model capable of relating all the input variables to software quality.

- *G.J. Pai is with the Fraunhofer Institute for Experimental Software Engineering (IESE), Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany. E-mail: ganesh.pai@iese.fraunhofer.de.*
- *J.B. Dugan is with the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, 351 McCormick Road, PO Box 400743, Charlottesville, VA 22904-4743. E-mail: j.b.dugan@ieee.org.*

Manuscript received 6 Feb. 2007; revised 15 June 2007; accepted 19 June 2007; published online 9 July 2007.
Recommended for acceptance by B. Littlewood.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0032-0207.
Digital Object Identifier no. 10.1109/TSE.2007.70722.

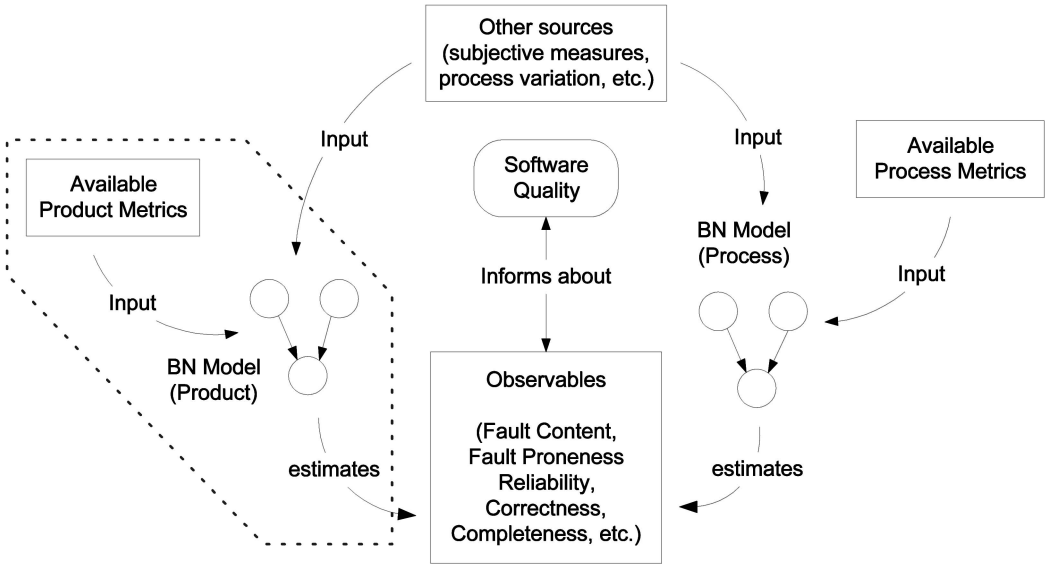

Fig. 1. Quality assessment framework.

In this paper, we mainly describe how Bayesian methods can be used for assessing quality (shown by the dotted box in Fig. 1). Specifically, we consider the C-K suite of metrics among the set of input variables and build a BN to relate them with both fault content and fault proneness. Assuming that the relationship can be modeled using a general linear model, we derive the structural and numeric specification for the BN. Our model can be thought of as a generalization of existing techniques for assessing software quality. Our model produces 1) a probability distribution of the estimated fault content per class in the system and 2) the conditional probability that a class contains a fault. Then, we empirically test our model using a data set from a real software subsystem. The results also show that our model produces these estimations at a statistically significant level.

We make the following contributions through this paper: First, we use a BN which relates software product metrics to fault content and fault proneness. Although there is some existing work in the literature that describes BN-based methods for defect content prediction [16], [17], [20], we believe that this is the first instance where a BN has been used for assessing defect proneness. Second, we assume that a general linear model relates the product metrics to quality; under certain assumptions, we show how multiple linear regression, Poisson regression, and logistic regression can be represented as a BN. This is the underlying functional form used to numerically specify the BN. Third, we use an entirely Bayesian approach for data analysis: Specifically, we use both Bayesian linear regression and Bayesian Poisson regression in fault content analysis. We find, surprisingly, that the linear model is better at describing the data than the Poisson model. However, since our analysis is based on only one data set, we believe that the study should be replicated on different sets of data to generalize our findings. Fourth, we add to the body of empirical knowledge about the relationship between certain measures for object-oriented software and fault content.

Acronyms and terminology. A list of the terminology and acronyms used in this paper is given below:

- r.v.: Random/uncertain variable
- i.i.d.: Independent and identically distributed
- BN: Bayesian network(s)
- pdf: Probability density function
- CPD: Conditional probability distribution(s)
- GLM: General linear model
- BLR: Bayesian linear regression
- BPR: Bayesian Poisson regression
- OLS: Ordinary least squares
- C-K: Chidamber-Kemerer
- $X, Y, Z, \ldots$: r.v., or their corresponding nodes in a BN
- $X = x$: r.v. X assumes state/value x
- $\mathbf{X}$: Set of r.v., i.e., $\mathbf{X} = \{X_i\}$
- $p(x|y)$: Probability that $(X = x)$ given that $(Y = y)$
- $i(A, B|C)$: A is probabilistically independent of B given C
- Pa_A: Parents of node A
- Des_A: Descendants of node A
- ND_A: Nondescendants of node A

The rest of this paper is organized as follows: In Section 2, we introduce Bayesian networks and describe our motivation for their use. Section 3 describes our research method, including model construction, model evaluation criteria, and the data set with which we empirically test our approach. The experimental results of the validation exercise are discussed in Section 4, while the associated threats to validity are discussed in Section 5. Section 6 presents related work in the literature. We conclude with directions for future work in Section 7.

2 Bayesian Networks

A Bayesian network is a concise representation of a joint probability distribution on a set of statistical variables, encoded as an acyclic graph of nodes and directed edges [19]. Consider a finite set of random variables (r.v.)

$\mathbf{X} = \{X_1, X_2 \ldots, X_n\}$. Each X_i can be discrete or continuous; $\mathbf{X}$ can also contain a mixture of discrete and continuous r.v. The set of states $\{x_i\}$ of each r.v. X_i are mutually exclusive. Formally,

Definition 2.1. *A probabilistic network $\mathcal{N} = (\mathcal{G}, \mathcal{S})$ over $\mathbf{X}$ consists of:*

- *A directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; $\mathcal{V}$ is the set of nodes in the graph and there is a one-one correspondence between $\mathcal{V}$ and $\mathbf{X}$. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the set of directed edges, representing conditional independence assumptions, i.e., for each $X_i \in \mathbf{X}$, $i(X_i, ND_{X_i}|Pa_{X_i})$ and $ND_{X_i} = \mathbf{X} \setminus (\{X_i\} \cup Des_{X_i})$.*
- *A set $\mathcal{S}$, of (prior) conditional probability distributions, that specifies $p(X_i)|(Pa_{X_i})$ for each $X_i \in \mathbf{X}$, where Pa_{X_i} represents the set of immediate parents of X_i.*

Once a network is specified over a set of r.v., we can compute their marginal and joint probabilities.[1] Given a BN structure, the joint probability distribution over $\mathbf{X}$ is encoded as

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|Pa_{X_i}). \quad (1)$$

Given this joint probability, the marginal probability of an r.v. X_i is computed as

$$p(X_i) = \sum_{X_j, j \neq i, j=1}^{n} p(\mathbf{X}). \quad (2)$$

Bayes' theorem is used to compute posterior probabilities for the r.v. when evidence[2] is available. The usage of Bayes' theorem is the reason why the network is termed as a Bayesian network. This characterisation of a BN does not include the notion of causality; however, BN are practically applied to model causal influences, where the entities of a system are modeled as the nodes of a BN, while the edges represent the cause-effect relationships between the entities. The qualitative part of a BN is encoded in the structure of the digraph, while the conditional probability distributions for the nodes encode the quantitative portion.

We are motivated to use BN as our (causal) model form since they can easily handle data which are both quantitative and qualitative. They are also are well suited to specifying variable relationships which may be either probabilistic or deterministic. Additionally, once a BN has been specified, the underlying mathematics permit evidence propagation in either direction in the network. Thus, in our context, using a BN to model the relationship between product metrics and quality permits us to explore the drivers of observed good quality; the results can be used to benchmark subsequent development.

3 Research Method

Our research approach is straightforward: We use a BN to model the relationships between the measurable structural properties of a software product, and its quality. This amounts to 1) identifying the observable parameters, which will form the nodes of the BN, 2) formulating a BN structure that captures the conditional independence relationships between the domain variables, and 3) specifying the conditional prior probabilities for the nodes in the BN. Once the BN model has been constructed, we test it empirically.

3.1 Model Parameters

In this paper, the main independent variables are a suite of metrics measuring the structural quality of object-oriented code and design; specifically, we consider the (C-K) suite. Additionally, we include a metric which measures class size. We chose these measures because they are adopted in industrial software development, especially within NASA, which sponsored part of this research. Additionally, there is a large body of empirical evidence which supports their correlation with measures of software quality [2], [4].

1. Weighted methods per class (WMC): The number of methods implemented in a given class.
2. Depth of inheritance tree (DIT): The length of the longest path from a given class to the root class in the inheritance hierarchy.
3. Response for class (RFC): Number of methods implemented within a class plus the number of methods accessible to an object class due to inheritance. Traditionally, it represents the number of methods that an object of a given class can execute in response to a received message.
4. Number of children (NOC): The number of classes that directly inherit from a given class.
5. Coupling between object classes (CBO): The number of distinct noninheritance related classes to which a given class is coupled, i.e., when a given class uses the methods or attributes of the *coupled* class.
6. Lack of cohesion in methods (LCOM): A measure of the degree to which a class represents single or multiple abstractions. There are varying definitions for LCOM; however, in this paper it is measured as suggested by Rosenberg and Hyatt [21], i.e., by computing the average percentage of methods in a given class using each attribute of that class, and then subtracting that percentage from 100 percent.
7. Source lines of code (SLOC): This is measured as the total lines of source code in the class and serves as a measure of class size.[3]

The dependent variables, which serve as surrogate metrics of software quality, are:

1. Fault content (FC): We define fault content as the number of faults per class. The estimation of our model is a (marginal) conditional probability of observing a certain number of faults per class, given the metrics for that class.
2. Fault proneness (FP): The conditional probability that a class contains a fault, given the metrics for that class.

1. We have assumed discrete r.v. in $\mathbf{X}$; continuous r.v. are discretized in our approach.

2. (Hard) evidence refers to the state of an r.v. being known with certainty, i.e., $p(X = x) = 1$.

3. Function points are a more preferred measure of artifact size since SLOC varies with the programming language used. However, the data set that we used for empirical analysis provided neither the measurement of function points nor the source code to extract them.

3.2 Model Construction

Traditionally, the BN (structure and conditional probability distributions) is constructed using a mixture of 1) domain experts, who can provide both a structural model of the independence relationships between the variables of interest, and their corresponding distributions, and 2) mining the available data for relationships, typically using learning algorithms [22]. In our approach, however, we assume that a general linear model (GLM) relates the dependent and independent r.v. and construct a BN structure which represents the GLM.

This assumption is motivated by several factors: The GLM is versatile enough to be able to represent existing linear relationships or, through transformations, a variety of nonlinear relationships. Linear models also provide a relatively simple and parameterized approach to comprehending the dependencies between domain variables. Additionally, despite the (valid) criticisms [14] of using linear models in fault content estimation, there is a large body of existing empirical research [4], [5], [6], [8], [23] to support using linear models and their transformations in software quality assessment.

Consider that a response variable Y varies as some function of a set $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$ of independent predictor variables; in the general linear model, we have

$$E(\mathbf{Y}) = \boldsymbol{\mu} = g^{-1}(\mathbf{X}\boldsymbol{\beta}), \tag{3}$$

where $\mathbf{Y}$ is the set of observations with expected value $E(\mathbf{Y}) = \boldsymbol{\mu}$. $\mathbf{X}\boldsymbol{\beta}$ is the linear predictor with coefficients $\boldsymbol{\beta}$ and g is the link function determined by the distribution of Y (typically, the exponential family). Since $\mathbf{X}$ and $\boldsymbol{\beta}$ are independent r.v., we can express the joint distribution of the r.v. in the GLM as

$$p(Y, \mathbf{X}, \boldsymbol{\beta}) = p(\mathbf{X})\ p(\boldsymbol{\beta}) p(Y|\mathbf{X}, \boldsymbol{\beta}). \tag{4}$$

This results in the BN structure of Fig. 2, where $p(\boldsymbol{\beta})$ is a (prior) distribution over the regression parameters, $p(\mathbf{X})$ is the distribution of the regressor variables, and $p(Y|\mathbf{X}, \boldsymbol{\beta})$ is the conditional probability density (CPD) representing the likelihood of the linear predictor. For discrete r.v., the CPD is represented as a node probability table, representing all possible values that the r.v. can take, given the states of the parents. It is convenient to specify the CPD as a parameterized deterministic (or probabilistic) relationship between the child and the parent nodes if the distribution of the child node is known. Now, we derive the functional form for the CPD when Y models fault content, and fault proneness.

3.2.1 CPD for Fault Content

Linear regression. If we assume in (3) that Y follows a Normal distribution, i.e., $Y \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2)$, and has an identity link function, then we have the functional form of multiple linear regression. In matrix notation,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{5}$$

Here, $\mathbf{y}$ is an $n \times 1$ matrix of observations of Y, $\mathbf{X}$ is an $n \times (k+1)$ matrix of predictor values (representing n data points for the k independent variables and a column vector of ones for the intercept), and $\boldsymbol{\beta}$ is an $(k+1) \times 1$ vector of unknown regression coefficients. The error process is captured by $\boldsymbol{\epsilon}$, which is an $n \times 1$ vector of errors. If we assume i.i.d. Gaussian noise, i.e., $\boldsymbol{\epsilon} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$, then the conditional distribution $(Y|\mathbf{X}, \boldsymbol{\beta}, \sigma^2)$ is also Gaussian [24]. Thus,

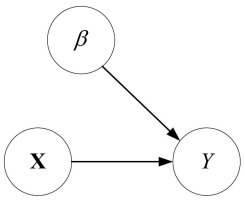

Fig. 2. BN representation of a general linear model.

$$Y|\mathbf{X}, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 I), \tag{6}$$

where σ^2 models noise variance and I is the $n \times n$ identity matrix.

Hence, in Fig. 2, if we assume that Y is a Gaussian r.v., then the conditional distribution for Y is Gaussian, with parameters $(\boldsymbol{\mu}, \sigma)$, and the CPD for Y is given as in (5), i.e., multiple linear regression.[4]

Poisson regression. Now, if we assume that (3) has a log-link function and that Y follows a Poisson distribution, i.e., $Y \sim \mathcal{P}(\boldsymbol{\mu})$, we have the functional form of Poisson regression

$$\ln(E(Y)) = \ln(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{7}$$

where $\mathbf{X}\boldsymbol{\beta}$ is as defined as in (5), but $\boldsymbol{\epsilon}$ is Poisson distributed. As a consequence, the conditional distribution $(Y|\mathbf{X}, \boldsymbol{\beta})$ is also Poisson distributed. Thus,

$$Y|\mathbf{X}, \boldsymbol{\beta} \sim \mathcal{P}(e^{\mathbf{X}\boldsymbol{\beta}}). \tag{8}$$

Hence, in Fig. 2, if we assume that Y is a Poisson r.v., then the CPD for Y is given as in (7), i.e., Poisson regression.

3.2.2 CPD for Fault Proneness

Consider the general functional form of binomial univariate logistic regression for a binary response r.v. Y which varies as a function of a set $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$ of independent predictor variables. We have,

$$p(Y = 1|\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\beta}}}. \tag{9}$$

Now, we assume that, in (3), Y follows a Bernoulli distribution, with outcomes $Y = \{0, 1\}$; also assume that the link function is the *logit* or log-odds function. Hence,

$$(Y|\mathbf{X}, \boldsymbol{\beta}) \sim \mathcal{B}(\boldsymbol{\mu}) \Rightarrow p(Y = 1|\mathbf{X}, \boldsymbol{\beta}) = \boldsymbol{\mu}, \tag{10}$$

$$Logit(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta} \Rightarrow \ln\left(\frac{\boldsymbol{\mu}}{1 - \boldsymbol{\mu}}\right) = \mathbf{X}\boldsymbol{\beta}, \tag{11}$$

$$\boldsymbol{\mu} = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\beta}}}. \tag{12}$$

4. As a consequence, the parameters can be estimated using the so-called least-squares technique [24]; a completely Bayesian approach, on the other hand, places a conjugate prior on the parameter [25].

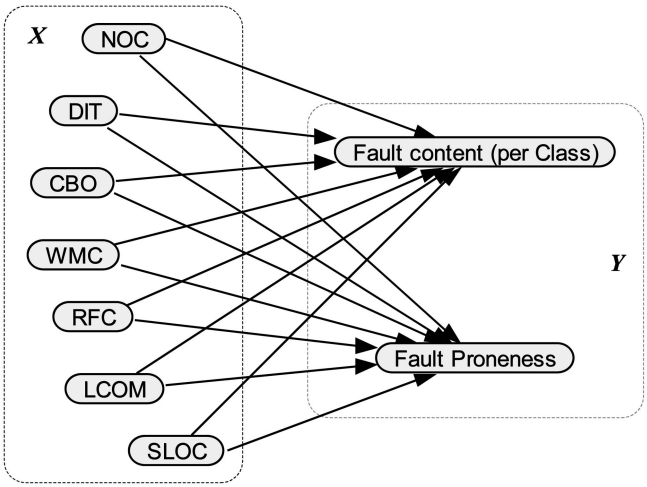

Fig. 3. BN model for defect content and defect proneness analysis.

Hence,

$$p(Y = 1|\mathbf{X}, \beta) = \frac{1}{1 + e^{-\mathbf{X}\beta}}. \quad (13)$$

From (9) and (13), it follows that the BN of Fig. 2 can be used to represent binomial logistic regression, if we assume that Y is a Bernoulli r.v.

3.3 Our Model

We have derived both the BN structure and the numerical specification for our model. Since the parent nodes representing the independent r.v. in the BN are observable, the prior distribution on these nodes is directly specified from available data. Fig. 3 shows our model for fault content and fault proneness assessment using the relevant metrics and dependent r.v. The parent nodes ($\mathbf{X}$) in the model are the suite of metrics, while the child nodes ($\mathbf{Y}$) include the r.v. for fault content and fault proneness

Since the CPD of each of the child nodes is a regression equation, we use traditional correlational analysis to determine which of all of the independent r.v. remain in the model. Linear and Poisson regression are applicable for fault content estimation, whereas logistic regression is useful for fault proneness estimation. The latter is especially useful since classes with faults can be categorized as fault-prone, whereas those without faults are considered to be non-fault-prone. To further reduce the size of the network, the nodes for the regression parameters are collapsed into the functional form of the CPD of the child nodes. All nodes are discretized when the model is solved.

We use the Netica Tool[5] and its underlying Bayesian propagation algorithm to compute the marginal distribution of the child nodes in our model. To derive the exact functional form of the CPD for each child node, we use a completely Bayesian approach for multiple linear regression, Poisson regression, and binomial logistic regression using the BayesX tool.[6] The tool allows the user to choose the family of distributions for the dependent r.v., and assumes diffuse priors[7] for the regression parameters. In the case of fault content assessment, we also apply the traditionally used method of ordinary least-squares (OLS) regression and compare it with the corresponding Bayesian solution.

We note that, by applying learning algorithms, the CPD for the child nodes can be automatically constructed, as opposed to solving the regression models independently, as we have done here. However, the application of learning algorithms to solve BN is left as an aspect of future work.

TABLE 1
KC1 Data Description

Item	Value
Number of classes	145
Number of methods	2107
Size	43K SLOC
Total number of (relevant) faults	640
Total % faulty (and fault-prone) classes	39.31
% Design faults	14.375
% Classes (Design faults)	18.62
% Source code faults	85.625
% Classes (Source code faults)	39.31

3.4 Model Evaluation and Data Description

Using a public domain data set,[8] we empirically evaluate three aspects of our modeling method:

1. Which underlying multiple regression model is a better candidate for specifying the CPD for the DC node in the BN? Alberg diagrams [5] are a simple mechanism to compare the underlying multiple regression models, i.e., the Bayesian approach, with OLS. To construct an Alberg diagram, first order the data and the model estimations in descending order, then plot the cumulative percentage of each to evaluate the usefulness of the models. The best model is the one which most closely follows the data.
2. After choosing the appropriate regression model, how does the BN model perform in describing the relationship between fault content and the product metrics? To compare the model estimates with the actual data, we use the Kolmogorov-Smirnov (K-S) test for significant difference between the data distribution and the estimated distribution.
3. Which underlying logistic regression model is a better candidate for specifying the CPD of the dependent r.v. in the BN? To evaluate the fault proneness estimation capability, we compute *sensitivity* (the proportion of correctly classified fault-prone modules), *Specificity* (the proportion of correctly classified non-fault-prone modules), *Precision* (the probability of correct classification), as well as the rates for false positive and false negative classification.

The data set KC1 (Table 1) is obtained from an object-oriented software subsystem implemented in C++ and is comprised of 2,107 methods implemented in 145 classes,

5. Available at http://www.norsys.com/.
6. Available at http://www.stat.uni-muenchen.de/~bayesx/.
7. A diffuse prior is usually an appropriate noninformative prior [26].
8. Available from the NASA Metrics Data Program: http://mdp.ivv.nasa.gov/.

TABLE 2
KC1 Data Summary Statistics

Metric	N	Min.	Max.	%25	Median	%75	Mean	Std. dev.
CBO	145	0	24	3	8	14	8.317	6.355
LCOM	145	0	100	58	84	96	68.724	36.761
NOC	145	0	5	0	0	0	0.214	0.697
RFC	145	0	222	10	28	44	34.379	36.078
WMC	145	0	100	8	12	22	17.421	17.389
DIT	145	1	7	1	2	2	2.000	1.254
SLOC	145	0	2313	10	108	233	211.248	344.358
FC	145	0	52	0	0	5	4.414	8.952

TABLE 3
Correlational Analysis of KC1 Metrics

Metric	CBO	LCOM	NOC	RFC	WMC	DIT	SLOC	DC
CBO	1							
LCOM	0.041	1						
NOC	-0.129	0.119	1					
RFC	**0.542**	**0.383**	-0.032	1				
WMC	**0.377**	**0.378**	0.101	**0.666**	1			
DIT	**0.470**	**0.249**	-0.112	**0.718**	**0.215**	1		
SLOC	**0.818**	0.010	-0.136	**0.523**	**0.496**	**0.385**	1	
FC	**0.520**	-0.006	-0.156	**0.245**	**0.352**	0.036	**0.560**	1

resulting in over 43KLOC. The data set is organized so as to include

- class to method relationship,
- relationships between methods, faults, and fault priority,
- relationships between faults, fault type and fault location,
- ten measures at the class level (including the C-K suite), and
- metric information at the method level.

We are primarily interested in the metrics available at the class level. The data set also contains both static and dynamic fault information; the former reflecting fault content over the entire life-cycle of the project and the latter reflecting the history of the fault. Of the two, we are concerned with the static fault information: 1,001 faults were reported, of which only 640 were attributed to errors in either the source code or design of the KC1 system. Only these faults were considered in our analysis.

The remainder were either errors in configuration, errors in the operating system or supporting COTS software, not bugs, or were not reproducible. Using the relationships between classes and methods, methods and faults, we compiled a data subset which contains the number of faults identified per class (both source code faults per class and design faults per class), metric information per class, and size of the class. All classes with design faults also had source code faults and all classes which reported at least one fault, were considered to be fault prone.

Table 2 provides the summary statistics for the metrics used to evaluate fault proneness and fault content.

4 Experimental Results

In this section, we evaluate our approach empirically using the metrics from the KC1 data set. First, we examine the association between the dependent and independent r.v. using correlational analysis. Then, in Section 4.1, we describe the results of multiple regression using 1) OLS, 2) Bayesian linear regression (BLR), and 3) Bayesian Poisson regression (BPR). In Section 4.2, we perform Bayesian logistic regression.

Table 3 shows the Spearman correlation coefficients quantifying the strength of the interactions between the different metrics in KC1. The **bold-face** values represent significant ($p < 0.0001$ at $\alpha = 0.05$ significance level) correlation between the variables. There appears to be a moderate level of correlation between the variables, implying that they are not completely independent. All of the metrics of the C-K suite except NOC and LCOM appear to be significantly correlated with class size, whereas only the metrics CBO, RFC, WMC, and SLOC are significantly correlated with fault content.

4.1 Defect Content Analysis

Using the correlational analysis as a starting point and using a backward search, we select those variables which are significantly correlated with fault content as the first linear predictor in the multiple regression analysis. Then, we add additional r.v. into the model and create a second linear predictor to examine whether the performance of the baseline model is improved. Specifically,

$$\eta_1 = (\mathrm{CBO}, \mathrm{WMC}, \mathrm{RFC}, \mathrm{SLOC})$$

and

$$\eta_2 = (\mathrm{CBO}, \mathrm{WMC}, \mathrm{RFC}, \mathrm{SLOC}, \mathrm{LCOM}).$$

TABLE 4
Model Comparison for Predictors

Predictor	Model		
	OLS (Adj. R^2)	BLR (DIC)	BPR (DIC)
η_1	**0.465**	**964.712**	-1329.255
η_2	0.463	966.218	**-1336.4292**

TABLE 5
Multiple Regression Models (Linear)

Model	Parameter					
	Intercept	CBO	RFC	WMC	SLOC	LCOM
OLS	-0.1028	0.183	-0.0396	0.0552	0.0161	0
BLR	-0.0955	0.1813	-0.039	0.0533	0.0162	0
BPR	0.2298	0.1117	-0.0075	0.0187	0.0006	-0.0048

With η_1 and η_2, we perform three sets of multiple regression analysis, i.e., OLS, BLR, and BPR. The better predictor from each set of analyses is now a possible candidate functional form for the BN model. We use 10-fold cross validation to construct the regression models; the data set is divided into 10 equal parts and regression is performed 10 times. Each time one part is chosen as the test set and the remaining nine parts are used to build the model. Table 4 summarizes the regression form, the predictor combination, and the statistic used to discriminate between the predictor combinations.

For the OLS method, the adjusted coefficient of determination (Adj. R^2) statistic compares the relative predictive power of the model, accounting for the sample size and difference in the number of variables; a larger R^2 implies that a larger proportion of variation is explained and is therefore preferred. The deviance information criterion (DIC) used to compare the Bayesian regression models is a natural way to compare models using the trade-off between the goodness-of-fit and the inherent model complexity [27]; a smaller DIC reflects a better model.

From Table 4, it appears that the 4-parameter linear predictor is better than the 5-parameter predictor when OLS and BLR are used; however, the 5-parameter predictor appears to be better in the case of Bayesian Poisson regression. Table 5 shows the models chosen using the goodness-of-fit criteria. These represent the set of candidate functional forms that will be used to build the CPD for the fault content node in our BN model (Fig. 3). From Table 5, it is immediately apparent that the BLR and OLS are very nearly the same. This is not surprising because BLR is equivalent to OLS if we assume diffuse priors for the regression coefficients and that the dependent r.v. has a Gaussian distribution.

Alberg diagrams, which plot the cumulative percentage of faults versus the cumulative percentage of classes, are a convenient graphical mechanism to compare the performance of these candidates. As mentioned earlier, to build an Alberg diagram, we sort the model estimations in descending order. Then, we plot the cumulative percentage of estimated faults against the percentage of modules having them. The better model will more closely follow the data.

Fig. 4 shows the Alberg diagram, which compares both models and yields some interesting observations. First, we notice that the Pareto principle, i.e., the 80-20 rule, holds for the data and that more than 80 percent of the faults are found in 20 percent of the modules. The BLR and BPR model

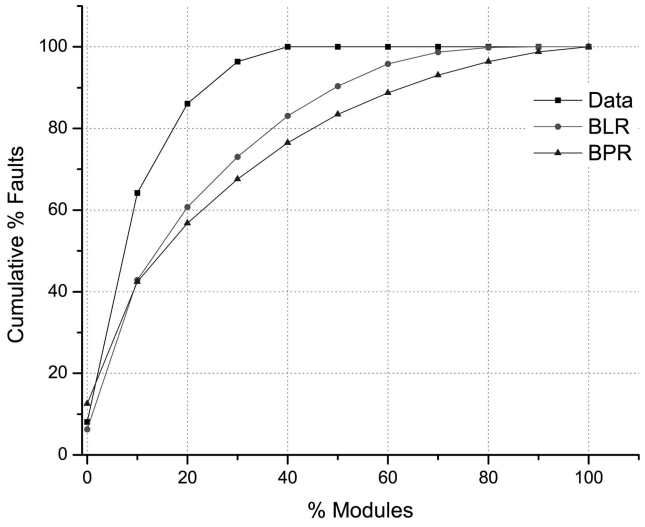

Fig. 4. Alberg diagram comparing BLR with BPR.

perform almost equally for the first 20 percent of the modules (both estimating approx 60 percent of the cumulative fault content), with the BPR model initially overestimating the fault content. After this point, the BLR model follows the data more closely and indicates a better fit to the data. We found this result surprising as the initial expectation was that Poisson regression would better describe a defect arrival process. We cannot generalize this result beyond this experiment; more analysis is required to evaluate the usefulness of Poisson regression versus linear regression.

The Alberg diagram is useful, not only in comparing the models, but also in guiding the effort to be invested in corrective actions, i.e., planning inspections or further testing. For example, if the available resources permit inspection or testing of about 30 percent of the software, then, by choosing the first 30 percent of the modules, i.e., those modules arranged in decreasing order of estimated defect content, we expect to find approximately 70 percent of the defects.

We use the BLR model to specify the CPD for the fault content node in our BN model, which is built in the Netica tool. Fig. 5 shows the results of this analysis. The distributions for the parent nodes are specified directly from data. The model estimation is a distribution of the expected fault content per module, over all the modules. For example, the model estimates that the probability of observing zero to six faults per module is 77.1 percent (Fig. 5).

The result of the K-S test ($p = 0.787$ at $\alpha = 0.05$ level of significance) is accepting the null hypothesis, i.e., the estimated distribution of fault content over the classes in KC1 are not significantly different from the data distribution.

4.2 Fault Proneness Analysis

For fault proneness assessment as well, we choose the two linear predictors η_1 and η_2. We build the models using Bayesian binomial logistic regression with 10-fold cross validation; again, the DIC of the models is used to evaluate their goodness-of-fit. Then, we compute their specificity, sensitivity, and precision to compare their results. The classification threshold for the models is $\tau = 0.5$, i.e., if $\tau \geq 0.5$, then the class is considered to be fault prone.

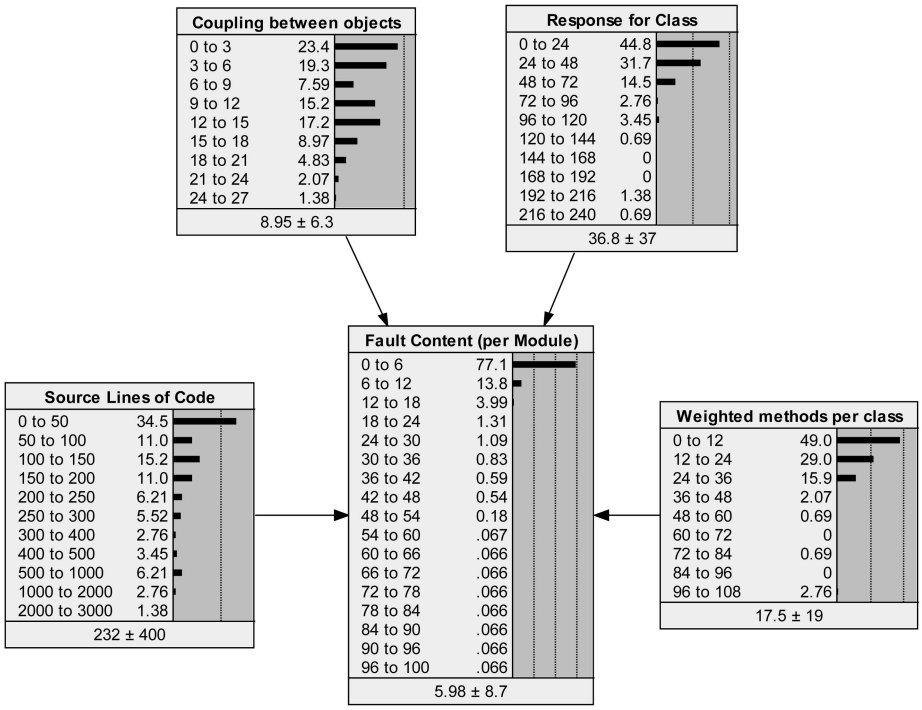

Fig. 5. Defect content estimation from the BN model.

Table 6 shows the model parameters, Table 7 shows the confusion matrices obtained from applying the two regression models, and Table 8 shows the results of evaluating model specificity, sensitivity, precision, and the corresponding rates for false positives and false negatives. From Table 6, we find that η_2 has a lower DIC than η_1. Also from Tables 7 and 8, η_2 is more sensitive to finding fault prone modules, achieves greater precision while having smaller rates of false positive and false negative classification.

Thus, the functional form of the CPD for the fault proneness node in our BN model uses η_2 as the linear predictor. Fig. 6 shows the BN model for fault proneness analysis using this chosen functional form. The estimation of the model is the marginal probability of observing a fault over all the modules. Essentially, this means that we should expect a 37.2 percent chance of finding at least one fault in a class picked at random from the KC1 software system.

4.3 Discussion of Results

One of the goals of this paper is to experimentally evaluate how Bayesian methods can be used for assessing software fault content and fault proneness.

Given the results of performing multiple regression, we find that the metrics WMC, CBO, RFC, and SLOC are very significant for assessing both fault content and fault proneness. Gyimóthy et al. [23] have found that this specific set of predictors is very significant for assessing fault content and fault proneness in large open source software systems. Additionally, their study also finds LCOM and DIT to be very significant for linear regression and NOC to be the most insignificant for both analyses. Our results indicate that neither DIT nor NOC are significant, but LCOM seems to be useful when performing Poisson regression; however, the linear model not containing LCOM was better than the Poisson model containing it. Therefore, depending on the underlying model used to relate the metrics to fault content, LCOM is significant.

We did not have data related to the change in metrics for subsequent releases of the KC1 system. Consequently, we performed 10-fold cross validation to build a BN model that estimates fault content at a statistically significant level. We also used the K-S test to confirm the hypothesis that the estimated distribution of fault content per module is not significantly different from the data. Given these findings, we believe that, once a BN model containing WMC, CBO, RFC, and SLOC measures is built on a given release of a

TABLE 6
Multiple Regression Models (Logistic)

Parameter	Predictor η_1	Predictor η_2
Intercept	-2.2269	-1.757
CBO	0.1646	0.198
RFC	-0.0152	-0.0134
WMC	0.0163	0.0246
SLOC	-0.003	0.0028
LCOM	0	-0.0146
DIC	153.62	**150.71**

TABLE 7
Confusion Matrices

Observed as	Classified as η_1 NFP	η_1 FP	η_2 NFP	η_2 FP	Total
FP	68	20	**75**	13	88
NFP	27	30	23	**34**	57
Total	95	50	98	47	145

TABLE 8
Model Performance

Predictor	Parameter				
	Specificity	Sensitivity	Precision	False Negative Rate	False Positive Rate
η_1	0.6296	0.5263	0.6759	0.4	0.2842
η_2	**0.8523**	**0.5965**	**0.7517**	**0.2766**	**0.2347**

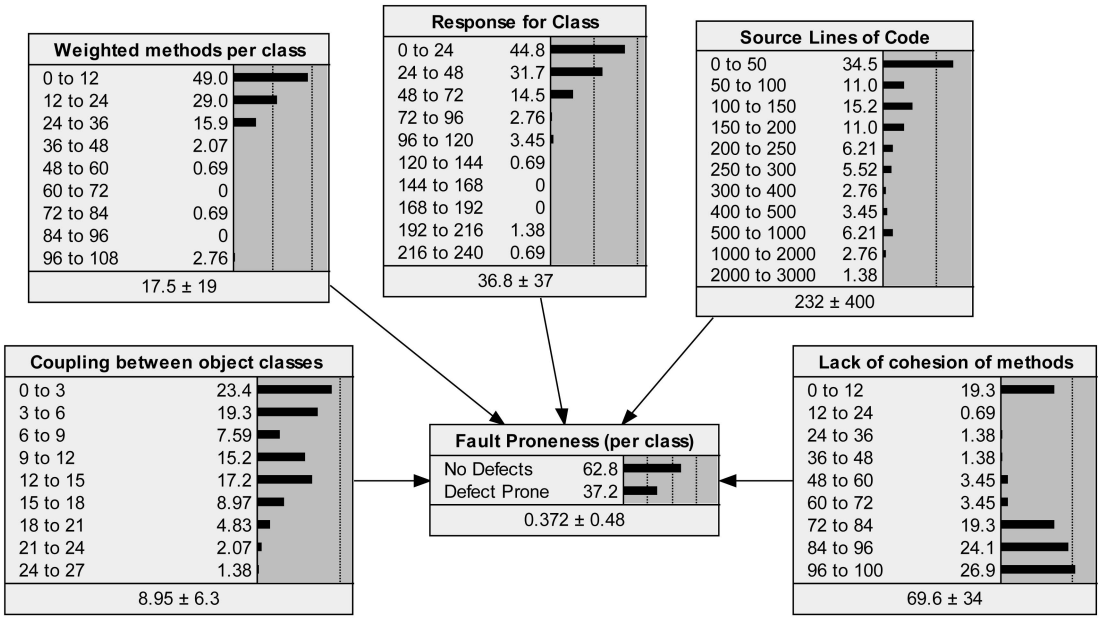

Fig. 6. BN model for fault proneness analysis.

system, it would be useful in providing early estimates of fault content in subsequent releases of the same system. To generalize these findings across systems, more empirical validation is required. However, we believe that a BN model, together with the assumption of a general linear model that relates product metrics to defect content, is applicable across software systems.

There is existing work in the literature by Zhou and Leung [9] which empirically validates the use of the same metrics suite that we have used in this paper, for assessing fault proneness. Consequently, we have not repeated univariate analyses on these metrics. As before, their study also finds WMC, CBO, RFC, LCOM, and SLOC to be very significant for fault proneness analysis, whereas DIT is not sufficiently significant. In the case of NOC, however, their results were inconclusive. In our analysis, adding the LCOM metric added to the performance of the logistic regression model; it had better sensitivity and precision as compared to the model that did not have it. Then, using the BN model, we computed the (marginal) probability that a class is fault prone, i.e., it contains at least one fault. As in the BN model for assessing fault content, we performed a 10-fold cross validation to construct the functional form of the CPD for fault proneness.

Zhou and Leung also applied logistic regression to compute the fault-proneness of the KC1 data set which we used in this paper; however, their predictor had a different subset of metrics from the C-K suite than we chose. The resulting model (Model I′ from reference [9]) had greater sensitivity and precision than ours. To improve the performance of our own approach, we simply replaced η_2 in our BN model with the parameters from Model I′. This updated model performance and the associated probability of finding a fault increased to 49 percent (from 37.2 percent). The main point of this exercise is to demonstrate the flexibility of our approach in 1) including existing models that meet our assumptions and 2) in refining model estimates with the included information.

Since the BN model is parametric, model refinement when new evidence (in the form of raw data or functional forms) is available is straightforward. Once the CPD of the BN variables is known, the BN can answer any probabilistic query on the nodes; this is useful as it quickly shows the variables that could be modified to reduce this probability. Given the broad context of our research goals (discussed in Section 1), we have built an analysis model based on Bayesian methods, i.e., Bayesian networks, Bayesian linear and logistic regression which performs no worse than existing methods. On the other hand, it provides a mechanism with which diverse sources of information can be easily included for model refinement.

5 Threats to Validity

As identified by Briand et al. [2], we identify three threats to validity in this experimental analysis:

1. *Construct validity.* This refers to the degree to which the dependent and independent variables in the study measure what they claim to measure.

Since one of the goals in our work is to examine how Bayesian methods can be used for quality assessment using a suite of product metrics, establishing construct validity is not in the scope of this work. Furthermore, since the independent variables in our work, the C-K suite of metrics, have been empirically validated to an acceptable degree in the literature, these variables can be claimed to be satisfactory.

The dependent variables in our study are measures of software quality, i.e., fault content and fault proneness. The former is measured as the number of faults found per class and the latter is defined as the probability of finding a fault during the activities in the verification process, e.g., acceptance testing. Assuming that these activities were properly performed, the dependent variables are also satisfactory.

2. *Internal validity.* This is the degree to which we can draw conclusions on the causal relationship between the independent and dependent variables.

   In our study, we have assumed conditional independence among the independent domain variables and that the independent and dependent r.v. are related by a general linear model. To explore the validity of these assumptions, we performed correlational analysis between the variables. Some of the independent r.v. have been found to have a statistically significant relationship, both among themselves and with the dependent r.v. The former implies that there is some redundant information between the metrics and possible collinearity, while the latter shows some evidence of a causal relationship.

   Thus, the assumption of conditional independence among the independent r.v. is under threat. Indeed, by using BN-structure learning algorithms [22], it would be possible to find a BN structure which would violate our assumption. From the assumption of the general linear model and the assumption of sampling from a Gaussian distribution, we have shown that the estimated fault content per class is not significantly different from the data distribution. Consequently, the threat to internal validity between the independent and dependent r.v. is reduced. We also found that linear regression performed better than Poisson regression in this case. However, this result cannot be generalized to other systems and more empirical analysis is required.

3. *External validity.* This refers to the degree to which experimental results can be generalized to other research settings, and within the population being studied.

   Since we have performed our empirical analysis on one data set, the external validity of our work is threatened. Specifically, we cannot generalize the results of using the specific set of predictors which we selected (both in fault content and fault proneness analysis), to other systems. However, through the use of k-fold cross validation ($k = 10$ in our experiment), we can be confident that the model results are statistically significant; therefore, the modeling approach would be applicable when dealing with subsequent versions of the same system.

We have shown that the BN model can represent multiple linear and logistic regressions. It is worthwhile to note that, by applying BN parameter learning algorithms, it is possible to estimate the regression parameters β and, consequently, the conditional distribution of the child nodes $p(Y|\mathbf{X})$. By doing so, we can modify our research approach so that CPD for the child node would be automatically generated rather than being computed from OLS, BLR, or BPR, as we have done here. Thus, with the underlying assumption of a general linear model, the BN subsumes existing methods in fault content and fault proneness assessment. Since these techniques have been applied in many empirical studies in the literature [8], the BN methodology with the Bayesian approach is applicable across systems.

6 Related Work

The empirical analysis of object-oriented metrics and their suitability for assessing fault proneness has been performed by numerous researchers, e.g., refer to [2], [4], [7], [8] and the references contained therein. To summarize the results for these studies, the C-K suite has been found to be significant for fault proneness for different systems; nevertheless, the specific subset of predictors from the suite which seem to show a significant relationship to fault proneness differs from system to system. Their significance for assessing both fault proneness and fault content in large open source software systems has been discussed in [23]. Ohlsson and Alberg [5] have used Alberg diagrams and other software product metrics in assessing software defect content using multiple linear regression, while Graves et al. [28] have used general linear models in predicting the distribution of faults incidences using measurements from software change history.

Fenton et al. [14], [15], [16], [17] have extensively advocated adopting a causal approach to software quality assessment and have built associated BN models. Their approach relies on specifying the functional form of the CPD in the BN as a parameterized probability density function; the common approach seems to be to use a truncated normal distribution whose parameters are the weighted sum of the parent nodes. A variety of other distributions can be used if specified using expert opinion or by tuning the model using data. Their approach seems to be most closely related to our work. If we perform regression with a zero intercept, it is possible to interpret the functional form of multiple linear regression in our work as equivalent to their usage of a weighted sum of parent nodes, i.e., the weights in their approach correspond to the parameter vector β in ours.

Gras and Minana [29], [30] also use BN in an industrial context at Motorola to predict fault injection levels in different software development phases. This work is also closely related to ours in using linear regression and principal components analysis as one possible approach to relate the BN nodes. Our approach represents an extension of a part of their work: We consider the general linear model, which is flexible in modeling linear regression, logistic regression, and Poisson regression. However, to the best of our knowledge, our approach is the first to use a BN for fault proneness assessment.

Zhou and Leung [9] use the same data set as ours to evaluate fault proneness models for classifying faults ranked by severity. Our approach differs from theirs in using a Bayesian approach toward fault proneness *and* fault content analysis.

7 Conclusions and Future Work

The broad goal of our research is to build an extensible framework for software quality assessment, one in which we can include available product and/or process metrics and diverse sources of evidence related to fault introduction during software development. Our approach is to separately consider the process and the product in their relationships to software quality and combine them in the unified framework. Bayesian networks provide a robust mechanism to build such a framework.

In this paper, we empirically examined one portion of this framework, i.e., using Bayesian methods for relating software *product* metrics to fault content and fault proneness. In particular, we constructed a BN model whose underlying representation is the generalized linear model. We derived the functional forms for the BN to represent different instantiations of the GLM: More specifically, we used functional forms of linear, Poisson, and logistic regression to specify the CPD for the nodes in the BN. A completely Bayesian approach is used to solve these models, after which the model was tested on a public domain data set; the results from the BN model are 1) an estimated distribution over the fault content per class and 2) the probability of finding a fault in a class. We found that the model estimations do not differ significantly from the data at a statistically significant level. Since our approach represents a generalization of some existing techniques in software quality assessment, the model performs no worse than existing methods. Through the resulting correlational and regression analyses, we contribute to the body of empirical knowledge about the relationship between the C-K metrics and fault content.

The BN models built in this paper were solved using MCMC simulation techniques; thus, approximations in results present a potential problem. One avenue for future work is exploring the use of BN with exact inference algorithms. Another potential problem with the BN model is its scalability when the number of parameters in the model grows. Since BN represent a class of models with state-space explosion problems, another avenue for future work is exploring the use of abstraction and pruning mechanisms in solving the models. There are a few potential solutions: First, partial least squares regression [31] and principal components regression provide a statistical technique to reduce the number of parameters. Second, if sufficient data is available, the BN model can be learned automatically and then optimized using domain knowledge.

A natural extension to fault proneness analysis is fault severity classification using multinomial logistic regression techniques and BN models. Similarly, a natural extension to fault content analysis is exploring how the independent variables can be combined in our model to represent nonlinear relationships.

Finally, as indicated earlier, there have been several empirical studies that have examined the relationship of product metrics, fault content, and fault proneness, but few that have empirically examined the relationship of process data to software quality. The BN framework provides a robust mechanism to include diverse sources of data into the analysis; thus, the second portion of this framework, i.e., the inclusion of process factors in the BN, presents an interesting and, we believe, useful area for future work.

Acknowledgments

The authors would like to thank Mike Chapman and the NASA Metrics Data Program for providing the data set used for the empirical analysis. They also thank the anonymous reviewers who helped to improve the paper with their detailed and constructive comments. This work was partially supported by the NASA Johnson Space Center under cooperative agreement number NNJ05JL56A and the NASA IV&V center. Opinions, findings, conclusions, and recommendations expressed in this paper are not necessarily the views of NASA.

References

[1] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE Trans. Software Eng.*, vol. 20, no. 6, pp. 476-493, June 1994.

[2] L. Briand, J. Wüst, J.W. Daly, and V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems," *J. Systems and Software,* vol. 51, pp. 245-273, 2000.

[3] T. Khoshgoftaar, J.C. Munson, B.B. Bhattacharya, and G.D. Richardson, "Predictive Modeling Techniques of Software Quality from Software Measures," *IEEE Trans. Software Eng.,* vol. 18, no. 11, pp. 979-987, Nov. 1992.

[4] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Trans. Software Eng.,* vol. 22, no. 10, pp. 751-761, Oct. 1996.

[5] N. Olhsson and H. Alberg, "Predicting Fault-Prone Software Modules in Telephone Switches," *IEEE Trans. Software Eng.,* vol. 22, no. 12, pp. 886-894, Dec. 1996.

[6] L.C. Briand, W.L. Melo, and J. Wüst, "Assessing the Applicability of Fault-Proneness Models across Object-Oriented Software Projects," *IEEE Trans. Software Eng.,* vol. 28, no. 7, pp. 706-720, July 2002.

[7] R. Subramanyam and M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE Trans. Software Eng.,* vol. 29, no. 4, pp. 297-310, Apr. 2003.

[8] L. Briand and J. Wüst, "Empirical Studies of Quality Models in Object-Oriented Systems," *Advances in Computers,* vol. 56, 2002.

[9] Y. Zhou and H. Leung, "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," *IEEE Trans. Software Eng.,* vol. 32, no. 10, pp. 771-789, Oct. 2006.

[10] W.S. Humphrey, *A Discipline for Software Engineering.* Addison-Wesley, 1995.

[11] A.P. Nikora, "Software System Defect Content Prediction from Development Process and Product Characteristics," PhD thesis, Dept. of Computer Science, Univ. of Southern California, May 1998.

[12] N. Nagappan and T. Ball, "Use of Relative Code Churn Measures to Predict System Defect Density," *Proc. Int'l Conf. Software Eng.,* 2005.

[13] N. Nagappan, T. Ball, and B. Murphy, "Using Historical In-Process and Product Metrics for Early Estimation of Software Failures," *Proc. Int'l Symp. Software Reliability Eng.,* 2006.

[14] N.E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Software Eng.,* vol. 25, no. 3, pp. 1-15, Mar. 1999.

[15] N. Fenton, P. Krause, and M. Neil, "Software Measurement: Uncertainty and Causal Modeling," *IEEE Software,* vol. 10, no. 2, Mar./Apr. 2002.

[16] M. Neil, P. Krause, and N.E. Fenton, "Software Quality Prediction Using Bayesian Networks," *Software Eng. with Computational Intelligence,* T.M. Khoshgoftaar, ed. Kluwer, 2003.

[17] N.E. Fenton, M. Neil, P. Hearty, W. Marsh, D. Marquez, P. Krause, and R. Mishra, "Predicting Software Defects in Varying Development Lifecycles Using Bayesian Nets," *Information and Software Technology*, vol. 49, pp. 32-43, Jan. 2007.
[18] G.J. Pai, "Probabilistic Software Quality Assessment," PhD thesis, Dept. of Electrical and Computer Eng., Univ. of Virginia, Feb. 2007.
[19] F.V. Jensen, *An Introduction to Bayesian Networks.* Springer-Verlag, 1996.
[20] M. Neil and N.E. Fenton, "Predicting Software Quality Using Bayesian Belief Networks," *Proc. 21st Ann. Software Eng. Workshop,* Dec. 1996.
[21] L. Rosenberg and L. Hyatt, "Software Quality Metrics for Object-Oriented System Environments," Technical Report SATC-TR-95-1001, NASA, 1995.
[22] D. Heckerman, "A Tutorial on Learning with Bayesian Networks," *Learning in Graphical Models,* M. Jordan, ed., MIT Press, 1999.
[23] T. Gyimóthy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Trans. Software Eng.,* vol. 31, no. 10, pp. 897-910, Oct. 2005.
[24] C.M. Bishop and M.E. Tipping, "Bayesian Regression and Classification," *Advances in Learning Theory: Methods, Models and Applications,* J. Suykens et al., eds., vol. 190, pp. 267-285, IOS Press, NATO Science Series III: Computer and Systems Sciences, 2003.
[25] G.E.P. Box and G.C. Tiao, *Bayesian Inference in Statistical Analysis.* John Wiley and Sons, 1992.
[26] J.O. Berger, *Statistical Decision Theory and Bayesian Analysis,* second ed. Springer-Verlag, 1993.
[27] D.J. Spiegelhalter, N.G. Best, B.P. Carlin, and A. van der Linde, "Bayesian Measures of Model Complexity and Fit," *J. Royal Statistical Soc.,* vol. 64, no. 3, pp. 583-639, 2002.
[28] T.L. Graves, A.F. Karr, J.S. Marron, and H. Siy, "Predicting Fault Incidence Using Software Change History," *IEEE Trans. Software Eng.,* vol. 26, no. 7, pp. 653-661, July 2000.
[29] J. Gras, "End-to-End Defect Modeling," *IEEE Software,* vol. 21, no. 5, pp. 98-100, Sept./Oct. 2004.
[30] E.P. Minana and J. Gras, "Improving Fault Prediction Using Bayesian Networks for the Development of Embedded Software Applications," *Software Testing, Verification, and Reliability,* vol. 16, no. 3, pp. 157-174, 2006.
[31] H. Abdi, "Partial Least Square Regression (PLS Regression)," *Encyclopedia of Measurement and Statistics,* N.J. Salkind, ed. pp. 740-744, Sage Publications, 2007.

Ganesh J. Pai received the BE (Hons.) degree in electronics engineering from the University of Bombay (Mumbai), India. He received the MS degree in electrical engineering from the University of Virginia, from where he also received the PhD degree in computer engineering. Currently, he is a research scientist in the Requirements and Usability Engineering Department at the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany. He also has experience as a consultant for probabilistic risk assessment, software engineering, and information-technology development. His primary research interests span the areas of rigorous software engineering, software processes, and software reliability/ quality engineering. Other research interests include dependable computer systems, verification and validation, and hardware/software co-design. He is a member of the IEEE, the IEEE Computer Society, the ACM, and Eta Kappa Nu.

Joanne Bechta Dugan received the BA degree in mathematics and computer science from La Salle University and the MS and PhD degrees in electrical engineering from Duke University. She is a professor of electrical and computer engineering and the director of the Computer Engineering Program at the University of Virginia. Her research focuses on probabilistic assessment of the dependability of computer-based systems. She has developed the Dynamic Fault Tree model, which extends the applicability of fault tree analysis to computer systems. Her research interests include hardware and software reliability engineering, fault-tolerant computing, system health management, and mathematical modeling using dynamic fault trees, Markov models, and Bayesian networks. She is a fellow of the IEEE and the IEEE Computer Society, cited for "contributions to dependability analysis of fault tolerant computer systems," and was awarded the Harriet B. Rigas Award for outstanding woman engineering educator, given by the IEEE Education and Computer Societies.