Conceptual Data Model-Based Software Size Estimation for Information Systems

HEE BENG KUAN TAN and YUAN ZHAO
Nanyang Technological University
and
HONGYU ZHANG
Tsinghua University

Size estimation plays a key role in effort estimation that has a crucial impact on software projects in the software industry. Some information required by existing software sizing methods is difficult to predict in the early stage of software development. A conceptual data model is widely used in the early stage of requirements analysis for information systems. Lines of code (LOC) is a commonly used software size measure. This article proposes a novel LOC estimation method for information systems from their conceptual data models through using a multiple linear regression model. We have validated the proposed method using samples from both the software industry and open-source systems.

Categories and Subject Descriptors: D.2.8 [**Software Engineering**]: Metrics—*Product metrics*

General Terms: Measurement

Additional Key Words and Phrases: Software sizing, line of code (LOC), conceptual data model, multiple linear regression model

ACM Reference Format:

Tan, H. B. K., Zhao, Y., and Zhang, H. 2009. Conceptual data model-based software size estimation for information systems. ACM Trans. Softw. Eng. Methodol. 19, 2, Article 4 (October 2009), 37 pages. DOI = 10.1145/1571629.1571630 http://doi.acm.org/10.1145/1571629.1571630

1. INTRODUCTION

It is very important to have good estimation of the required effort for any potential or confirmed software project [Boehm et al. 2000; Canfora et al. 2004;

This article is a revised and extended version of "Estimating LOC for Information Systems from Their Conceptual Data Models" presented at ICSE 2006 in Shanghai, China.
Authors' addresses: H. B. K. Tan (corresponding author) and Y. Zhao, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798; email: {ibktan,PG01874422}@ntu.edu.sg; H. Zhang (corresponding author), School of Software, Tsinghua University, Beijing 100084, China; email: hongyu@tsinghua.edu.cn.

DOI 10.1145/1571629.1571630 http://doi.acm.org/10.1145/1571629.1571630

Jeffery and Walkerden 1999; Ruhe et al. 2003]. Overestimation may lead to the abortion of essential projects or the loss of projects to competitors. Underestimation may result in huge financial losses. It is also likely to adversely affect the success and quality of projects [Boehm and Fairley 2000]. Despite significant effort spent on software estimation research, there is still much difficulty in estimating effort accurately for software projects [Briand and Wieczorek 2002; Molokken and Jorgensen 2003].

The estimation of software size plays a key role in project effort estimation. Line of code (LOC) and function points (FPs) are the size measures most commonly used in existing software effort estimation models. Despite the existence of well-known software sizing methods such as the function point method [Albrecht and Gaffney 1983; COSMIC 1999; Garmus and Herron 2000; Jeffery et al. 1993; Jones 1997] and its variants tailored for object-oriented software [Costagliola et al. 2005; Laranjeira 1990], many practitioners and project managers continue to produce estimates based on ad hoc or so-called “expert” approaches [Armour 2002; Miranda 2000]. The main reasons cited are the lack of required information in the early stage of a project and the need of domain specific methods [Miranda 2000]. However, the accuracy of ad hoc and expert approaches has inherent problems that often results in upsets over project budgets and schedules [Miranda 2000]. It also affects the success of many projects.

The entity-relationship (ER) model originally proposed by Chen [1976] is generally regarded as the most widely used tool for the conceptual data modeling of information systems [Ghezzi et al. 2003]. In ER modeling, an ER model is constructed to depict the ideal organization of data, independent of the physical organization of the data and where and how the data is used. An ER model is specified in a diagram called an *ER diagram* [Teorey et al. 1986]. Class diagram is an evolution of the ER diagram [Blaha and Premerlani 1998; Hay 2002]. In this article, for the purpose of following the latest notation, we shall use the term *class diagram* instead of ER diagram for conceptual data modeling. Before proceeding further, we shall define a few terms.

In a class diagram, a *non-many-to-many binary relationship type* or *non-many-to-many binary association type* refers to a binary relationship type or binary association type, respectively, in which at least one multiplicity is “one” or “zero or one.” The term *conceptual data model* refers to class diagram constructed in the following way to model the logical organization of data:

(1) Represention of the information of business entities and concepts that are to be maintained in a database as classes without operations.
(2) Represention of the relationships between objects of these classes as non-many-to-many binary associations, aggregations and generalizations between the classes. No other associations are required for the representation as any other association can be decomposed into non-many-to-many binary associations by introducing a class to represent the original association itself and a non-many-to-many binary association to associate each associated class to the class introduced [Snoeck and Dedene 1998].

Information systems constitute one of the largest software domains. This article proposes a novel method to estimate the LOC for an information system from its conceptual data model. It is an expanded version of the method proposed in Tan et al. [2006]. Based on samples collected from both industry and open-source systems, we have validated the proposed method for systems that are developed using several programming languages. We have also empirically compared the LOC estimation from the proposed method with the well-known function point method and the use case point method.

The article is organized as follows. Section 2 gives the background information. Section 3 presents the proposed method for LOC estimation. Section 4 reports our validation of the proposed method. Section 5 empirically compares LOC estimation of the proposed method with the well-known function point and use-case point methods. Section 6 discusses the use of the proposed software sizing method in software effort estimation. Section 7 compares the proposed method with related work. Section 8 concludes the article.

2. BUILDING A MULTIPLE LINEAR REGRESSION MODEL

Regression analysis is a classical statistical technique for building estimation models. It is one of the most commonly used methods in econometric work. It is concerned with describing and evaluating the relationship between a dependent variable and one or more independent variables. The relationship is described as a model for estimating the dependent variable from independent variables. The model is built and evaluated through collecting sample data for these variables. The following multiple linear regression model that expresses the estimated value of a dependent variable y as a function of k independent variables, $x_1, x_2, \ldots x_k$, is a commonly used method for regression analysis:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_k x_k,$$

where $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k$ are the coefficients to be estimated from a random sample.

We apply the following two commonly used methods for building and validating multiple linear regression model:

(1) *Holdout method*. The holdout method uses separate datasets for model building and validation. Any model built from a random sample should be validated against another random sample. As a rule of thumb, the size of the dataset for building the model should not be less than five times of the total number of independent variables. The size of the latter sample should not be less than half the size of the former sample.

(2) *10-Fold cross-validation method*. The 10-fold cross validation is the most commonly used k-fold cross-validation (by setting the integer k to 10). K-fold cross-validation uses only one dataset [Briand et al. 1999]. It divides the dataset into k approximately equal partitions. Each partition is in turn used for testing and the remainder is used for training. That is, the $(1-1/k)$ portion of the dataset is used for training and the $1/k$ portion of the dataset is used for testing. The procedure is repeated k times so that, at the end, every instance is used exactly once for testing. From this, the best-fitted

parameters are chosen to build and validate the model based on the whole dataset.

The following tests that can be computed from most statistical packages are usually carried out in building a model [Kennedy 2003; McClave and Sincich 2003; Neter et al. 1996; see also the SAS/STAT User's Guide[1]]:

(1) *Significance test*. This is usually based on a 5% level of significance. An F-test should be done for the overall model. If the value of $(\Pr > F)$ is less than 0.05, then it indicates that the overall model is useful, that is, there is sufficient evidence that at least one of the coefficients is nonzero at the 5% level of significance. Furthermore, a t-test is conducted on each $\hat{\beta}_j (0 \leq j \leq k)$. If all the values of $(\Pr > |t|)$ are less than 0.05, then there is sufficient evidence of a linear relationship between y and each $x_j (1 \leq j \leq k)$ at the 5% level of significance.

(2) *Fitness test*. The basic method to measure how well a regression model built describes the set of data in the sample collected is by the multiple coefficient of determination R^2. The values of R^2 always fall between 0 and 1; $R^2 = 0$ implies a complete lack of fit of the model to the data, while $R^2 = 1$ implies a complete fit with the model passing through every data point. However, R^2 always increases as more variables are added to the regression equation even though these new variables add little new independent information. It also gives a misleading result if the number of data points is not substantially larger than the number of parameters. To address this problem, an adjusted R^2 called the *adjusted multiple coefficient of determination* R_a^2 has been proposed. If R_a^2 for the model is higher than 0.75, then it implies that the least-squares model has explained more than 75% of the total sample variation of y values, after adjusting for sample size and number of independent variables.

(3) *Multicollinearity test*. Multicollinearity should not exist between independent variables. Variance inflation factors (VIFs) and condition indices are used to test multicollinearity. The VIF of an independent variable measures how much the variance in the estimate of its coefficient is "inflated" by the fact that the other independent variables contain information redundant with the variable. Values of the VIF exceeding 10 imply signs of serious multicollinearity. Values of the VIF exceeding 5 but less than 10 also warrant investigation. If the condition indices corresponding to each independent variable are less than 10 (the threshold value), then there is no significant evidence of the existence of multicollinearity between independent variables. A condition index over 15 indicates a possible multicollinearity problem. A condition index over 30 suggests a serious multicollinearity problem.

(4) *Extreme case test (outlier)*. Ideally, there should be no extreme cases (i.e., outliers). Extreme cases can distort coefficient estimation. For each data point, a residual analysis is conducted. If the absolute value of its

[1]`http: //www.id.unizh.ch/software/unix/statmath/sas/sasdoc/stat.`

studentized residual (RStudent) is below 2 (a usual threshold for identifying the large influence of observation on the parameter estimates), then there is no significant evidence of a large influence on the parameter estimation caused by the data point. That is, there is no significant evidence that the data point is an extreme case. If the absolute value of the RStudent of a data point exceeds 2, then it warrants a look at the data point. If the absolute value of the RStudent of a data point exceeds 3, then it warrants a serious investigation of the data point. If the data point itself is wrong, it should be recollected if possible [Kennedy 2003]. In total, there should be about 5% of the data points in the sample with their absolute values of RStudent exceeding 2 but all these values should be less than 3. Therefore, a small percentage of data points with their absolute values of RStudent above 2 and below 3 are expected [Glantz and Slinker 2001] in a good model.

MMRE and PRED(0.25) are the most commonly used measures for model validation in software engineering research. Lowest MMRE is preferred. PRED(0.25) is the ratio of the number of cases in which the estimates are within the 25% of the actual values divided by the total number of cases. The main criterion for validation lies in obtaining acceptable values of MMRE and PRED(0.25). Their acceptable values are not more than 0.25 and not less than 0.75, respectively.

3. THE PROPOSED SOFWARE SIZE ESTIMATION

Before proceeding to the proposed LOC estimation method, we shall discuss the rationale of using a conceptual data model to estimate LOC and our earlier exploration of this subject.

3.1 The Rationale of Using a Conceptual Data Model

In this article, *information system* refers to a database application that supports business processes and functions through maintaining a database using a standard database management system (DBMS) such as Oracle, etc. Information systems are data-intensive. From our observation, a large class of information systems has the following properties:

(1) They provide a graphical user interface (GUI) for maintaining business entities and the concepts and the relationships between them in a database.
(2) They implement business logic through referencing to information (maintained in a database) about business entities and concepts via the associated relationships.
(3) They deliver information (maintained in a database) on business entities, concepts, and their relationships, interactively or through printed reports.
(4) For each business entity, concept, and relationship type, they provide programs to create, reference, and remove their instances (to maintain their complete lifecycle).
(5) They contain simple data analysis functions such as sum, count, average, minimum, and maximum.

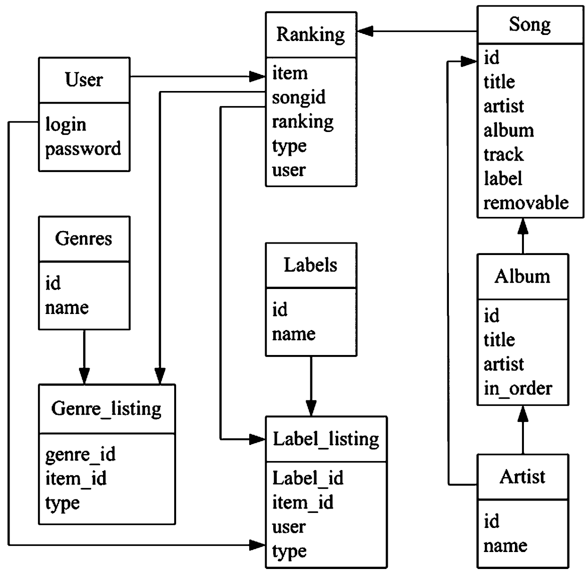

Fig. 1. Conceptual data model for the mp3cattle syatem.

(6) They provide error correction programs for canceling extraneous input instances submitted by users.

Therefore, the size of the source code of an information system in this class depends basically on its database, which in turn depends on its conceptual data model. In fact, the derivation of program structure from the structure of the data that the program processes in the Jackson structured program design method (JSP) implies this characteristic [Burgess 1988].

For example, Figure 1 is a conceptual data model for the mp3cattle system from the open-source Website SourceForge[2] that falls under the class just mentioned. Arrows show the direction of navigation paths between related classes. We constructed the data model through a reverse engineering process that will be discussed in Section 4.1.

The preceding observation suggests that the size of an information system is well characterized by its conceptual data model. As a conceptual data model for an information system can be more accurately constructed than obtaining the information required by existing software sizing methods in the earlier stage of software development, we have explored using it in LOC estimation. In fact, the construction of a conceptual data model in the early stage of software development is quite commonly practiced in industry.

3.2 Our Earlier Exploration

A preliminary method using a multiple linear regression model to estimate the LOC for information systems from their conceptual data model was proposed

[2] `http://sourceforge.net/`.

in Tan and Zhao [2004, 2006]. This method is based the entity-relationship (ER) model. In this method, the LOC for an information system is estimated from the total number of entity types (E), total number of relationship types (R), and total number of attributes (A) in an ER diagram that represents the system's conceptual data model. We used the method to build and validate LOC estimation models for two programming languages: Visual Basic with SQL; and Java with JSP, HTML, and SQL. The models were built and validated from limited datasets collected from industry. The model that we built for Visual Basic-based systems is as follows:

$$Size = 6.788 - 0.062E + 2.169R - 0.007A.$$

The model that we built for Java-based systems is as follows:

$$Size = 4.678 + 1.218E + 0.028R + 0.023A$$

In both models, size was measured in thousand lines of code (KLOC).

3.3 The Proposed LOC Estimation Method

To continue the work on estimating the LOC for information systems from conceptual data models, further examinations were carried out on the two models that we built for the Visual Basic- and Java-based systems in Tan and Zhao [2004, 2006]. We observed that the coefficients for E and A in the model for the Visual Basic-based systems were negative. Furthermore, from the coefficients in the two LOC estimation models, one may expect to see some consistency on relative order of influence by E, R, and A on LOC. However, there was no such consistency reflected. In the model for the Visual Basic-based systems, the coefficient for R was the highest, but the coefficients for both A and E were negative. But, in the model for Java-based systems, the coefficient for E was the highest, that for R was the second highest, and that for A was the lowest. To address these issues, we performed more statistical analysis on the datasets and also investigated the accuracy of the datasets.

It is well known that multicollinearity [Belsley et al. 2004; Kennedy 2003] can have harmful effects on multiple regression models, both in the interpretation of the results and in how they are obtained. To investigate the above-mentioned observations, we ran a multicollinearity diagnostics for the two models. From the result computed by SAS, we noticed that some condition indices for the Visual Basic-based model exceeded the threshold value (10) and some condition indices for the Java-based model exceeded 15. We further noticed that all VIFs for both models exceeded 4 and two VIFs for the Java-based model exceeded 15. From the model building criteria discussed earlier, this suggests that some multicollinearity may exist among E, R, and A in both models.

Most organizations in the software industry hold the view that their systems are confidential and should not be accessed by external parties. Therefore, most of the data in the datasets presented in Tan and Zhao [2004, 2006] was supplied by users directly and we have no means of verifying it. After much effort, we managed to convince two organizations to allow us to check the accuracy of the data provided. The checking revealed that three out of the four VB-based systems provided by one of them were wrong. All three Java-based systems provided by the other organization were also wrong.

Based on the findings, we decided to recollect datasets to rebuild and revalidate models. This time, for the industry datasets, we collected data only from those systems which were verifiable by us. And, because of this, it was extremely difficult to collect larger datasets from the industry. To address this problem, we collected data from open-source systems through reverse engineering in addition to the industry datasets.

In our new empirical studies, samples were collected from systems that fell under the class of information systems discussed in Section 3.1. All open-source systems were collected randomly by students from this class of information systems. The industry systems were collected from those organizations that allowed us to validate the data provided through verifying their documentation. All the data was extracted by one person and verified by another person.

Through much experimentation, for a programming language or development environment, we propose the following multiple linear regression model to estimate the KLOC (thousand lines of code) for an information system in the above-mentioned class, developed using the programming language or environment, from its conceptual data model:

$$KLOC = \hat{\beta}_0 + \hat{\beta}_C C + \hat{\beta}_R R + \hat{\beta}_{\overline{A}} \overline{A}.$$

KLOC is based on physical lines of code excluding comment and blank lines. Static text files for displaying on screen were excluded from KLOCs. In the multiple linear regression model, we used the following three independent variables to characterize the conceptual data model:

—C, the total number of classes in the conceptual data model;

—R, the total number of unidirectional relationship types in the conceptual data model. It was counted according to the type of relationship as follows:
  (i) *Non-many-to-many binary association*. Each navigation direction of a non-many-to-many binary association was counted as 1. That is, a bidirectional non-many-to-many binary association was counted as 2.
  (ii) *Aggregation*. For each part class in each aggregation, each navigation direction between the part class and the assembly class was counted as 1.
  (iii) *Generalization*. For each subclass in each generalization, each navigation direction between the subclass and the superclass was counted as 1.

—$\overline{A}$, the average number of attributes per class in the conceptual data model, that is, $\overline{A} = A/C$, where A was the total number of attributes that are defined explicitly in the conceptual data model. Note that, in counting A, for a subclass in a generalization, attributes of the subclass inherited from its superclass were not counted.

$\hat{\beta}_0$, $\hat{\beta}_C$, $\hat{\beta}_R$, and $\hat{\beta}_{\overline{A}}$ were the coefficients in the multiple regression model to be estimated from samples.

The use of the proposed LOC estimation method centers on the construction of a conceptual data model using a class diagram, for a target system. From the objective of a target system, the activities to be supported by the system, and its other requirements, a conceptual data model should be constructed as follows:

(1) Identify all the business entities and concepts required to be tracked or monitored by the system. That is, identify all business entities and concepts required to be maintained in a database. Derived entities and concepts should be excluded. Data entities that only facilitate the implementation of the system should also be excluded. Each resulting entity and concept is represented as a class.
(2) Identify all the interactions between business entities, between concepts, and between the business entities and concepts represented. All the interactions are represented as non-many-to-many binary associations and aggregations. Type and subtype relationships between business entity types and between concept types are represented by generalizations between classes.
(3) For each business entity and concept represented as a class, identify the required attributes to describe the entity or concept, and represent them as the attributers of the class.

Many system development methodologies build conceptual models using UML notation in the requirement analysis stage. If such a model is available, then the required conceptual data model for the proposed method can be derived from it by including only persistent classes that model nonderivable business entities and concepts. All nonpersistent classes that model processing logic, implementation data, or concepts should also be excluded. Once the conceptual data model is constructed, the values of the required parameters in the proposed model can be counted easily from their definitions.

4. VALIDATION OF THE PROPOSED LOC ESTIMATION METHOD

We have validated the proposed method using the two methods for building and validating multiple linear regression model that were discussed in Section 2—the holdout method and the 10-fold cross-validation method.

Both methods were applied to the following five pairs of datasets categorized into three groups according to the programming language used—each pair had two datasets labeled as *dataset A* and *B*—that were collected independently from software industry and open-source systems:

(1) *Industry VB-based system*. This pair of datasets is shown in Tables A.I and A.II in the Appendix. They were collected from the industry. The systems in this pair were developed using Visual Basic with SQL. Both datasets had 16 systems. These datasets were collected from a number of different organizations in the software industry. They contained systems from a variety of application domains including shipment management, auction management, finance, administration, logistics management, business management, medical information systems, and donation management.
(2) *Open-source PHP-based system*. This pair of datasets is shown in Tables A.III and A.IV in the Appendix. They were collected from two

open-source Web sites: SourceForge (see footnote 3) and Freshmeat.[3] The systems in this pair were developed using PHP with HTML and SQL. Dataset A had 32 systems. Dataset B had 31 systems. These datasets contained systems from a variety of application domains including content management, resource scheduling, inventory management, and entertainment.

(3) *Java-based system*. The following three pairs of datasets were collected from systems developed using Java with JSP, HTML, and SQL:

  (i) *Industry Java-based system*. This pair of datasets is shown in Tables A.V and A.VI in the Appendix. Both datasets had 16 systems. They were collected from a number of different organizations in the industry. These datasets contained systems from a variety of application domains including food-supply management, business management, project management, schedule management, demand chain management, freight management, and booking management.

  (ii) *Open-source Java-based system*. This pair of datasets is shown in Tables A.VII and A.VIII in the Appendix. They were collected from two open-source Web sites: (see footnote 3) Freshmeat (see footnote 4). Dataset A had 30 systems. Dataset B had 24 systems. These datasets contained systems from a variety of application domains including content management, project organization, member management, job scheduler, and entertainment.

  (iii) *The combined industry and open-source Java-based system*. This pair of datasets was formed by combing the corresponding datasets from the last two pairs of datasets. Therefore, dataset A had 46 systems. Dataset B had 40 systems.

For each pair of datasets, an LOC estimation model was built and validated using both the holdout and 10-fold cross-validation methods. In applying the holdout method, dataset A was used for model building and dataset *B* for model validation. The sizes of all the model building and validation datasets satisfied the requirements discussed in Section 2. In applying the 10-fold cross-validation method, dataset A and B were combined into one to which the cross validation was then applied.

All the required statistics for our model building and validation were computed by the statistical packages from SAS and open-source software WEKA (http://www.cs.waikato.ac.nz/ml/weka/).

4.1 Data Collection

The main objective for collecting data from open-source systems was to have larger datasets. Many organizations in the software industry did not construct proper conceptual data models in the early stage of their software development. Furthermore, because of confidentiality concerns, many of them did not allow us to access their documentation to verify the accuracy of the data supplied by

[3] `http://freshmeat.net.`

them. To ensure the accuracy of our own investigations, we did not use such data. Moreover, data collected from open-source systems also has the advantage of allowing others to verify it.

In the datasets collected from the industry, all the conceptual data models used were extracted from requirement specifications (to reflect the early stage of software development). All data supplied by users was verified by us to ensure its correctness. All questionable data was excluded.

In the datasets collected from open-source systems, all the sample systems were randomly drawn from SourceForge (see footnote 3) and Freshmeat (see footnote 4)—the world's two largest open-source software development Web sites. For each system sampled, the conceptual data model of the system was manually reverse engineered from its database schema and program source code. The software tool, Code Counter Pro 1.21, was used to count the LOCs of the system automatically. Next, we shall discuss the reverse engineering process and give some details on how LOC is counted before proceeding to the procedure for validating the data collected.

We analyzed the transformation of class diagrams into relational database designs discussed in Chapter 13 of Blaha and Premerlani [1998] to design our reverse engineering process. For each system sampled, the following two steps were carried out to reverse engineer a conceptual data model from its database schema and program source code:

(1) *Recovery of classes and attributes*. First, the database schema defined by SQL statements (typically stored in a .SQL file) for the system was extracted. Next, tables in the schema that represented domain entities and concepts were identified. System testing and code review were performed to aid in understanding of the meaning of a table. Then, each table identified was represented as a class in the conceptual data model. The attributes of the table were extracted from the schema to form the attributes of the class. Class and attribute names had to follow the corresponding table and attribute names, respectively, for data validation purposes. Tables that were solely for implementation purposes and did not represent any domain entities and concepts were excluded.

(2) *Recovery of relationship types*. In this step, for all the tables represented as classes in Step 1, each navigation path from table S to table T was identified and represented as a directed relationship type as follows. First, from the database schema, all the candidate navigation paths between these tables were identified through the following rules:

(a) Each table T with a foreign key X that referenced to the primary key of a table S *strongly suggested* a candidate navigation path from S to T.

(b) Each table T with an attribute X having a name very close to the name of the primary key of a table S or the name of a table S with a primary key *strongly suggested* a candidate navigation path from S to T.

(c) Each table T with an indexed attribute X having a name very close to the name of an attribute of a table S *suggested* a candidate navigation path from S to T.

Next, all the SQL statements in the program source code were studied and analyzed and system testing also conducted, to examine the candidate navigation paths identified and to detect further navigation paths. Finally, each confirmed navigation path from a table S to table T was represented as a directed relationship type from the class that represented S to the class that represented T.

As an example, Figure 1 shows a conceptual data model reengineered for the mp3cattle system from SourceForge. From Figure 1, the number of classes, attributes, and relationship types of this system are 9, 31, and 10, respectively. Therefore, the three parameters in the proposed model, C, R, and $\overline{A}$ for this system, are 9, 10, and 3.444, as shown in Table A.VIII.

For each system sampled, the LOC of its source code were counted automatically by the software tool, Code Counter Pro 1.21, based on physical lines of code. Comment and blank lines, and static text file for displaying on screen, were excluded in the counting. More specifically, for VB-based systems, the counting was based on .frm, .frx, .bas, and .sql files. For PHP-based systems, the counting was based on .php, .sql, .css, .html, and .xml files. For Java-based systems, the counting was based on .java, .sql, .js, .jsp, .html, and .xml files. For systems that cater to multilanguages, the above tasks were performed on those files required by the English language. As non-English source code is only for fixed descriptions (e.g., button names and field label descriptions), which are stored in separate files with clearly indicated descriptions, they could be excluded without any difficulty.

The data collection was carried out by four final-year students in their final-year projects. These students were divided into two groups with two students each. Both students in the first group collected the data independently for all the open-source PHP-based and Java-based systems in dataset A. Both students in the second group collected the data independently for all the open-source PHP-based and Java-based systems in dataset B. The data collected were validated for accuracy through the following procedures:

(1) For each system, the data was collected independently by two students and compared. The data from both students had to have an exact match before acceptance could be considered.
(2) The data for the first three systems collected by each student were thoroughly verified by us regardless of the matching results. This served as a kind of training. It also helped to resolve any misunderstandings. Subsequently, if the data collected by the two students for the same system did not match, we carried out a check to verify the data from both students and to correct the mistakes. For the identical data collected by the two students, for the next 20 systems, 40% was selected randomly and still verified by us. For the remaining systems, 20% was selected randomly and still verified by us.
(3) After the first three systems, the comparison and verification were carried out on a weekly basis. Students were briefed about their mistakes before any further data collection was undertaken. This was to improve the accuracy of the subsequent data collected by them.

4.2 Model Building and Validation

This section reports the model building and validation for each pair of datasets using both the holdout and 10-fold cross-validation methods.

4.2.1 *Using the Holdout Method.* Applying the holdout method, we built the following five models from the five pairs of datasets collected:

(1) *Industry VB-based system*. The following model was built from the dataset shown in Table A.I and validated using the dataset shown in Table A.II (16 systems each):

$$KLOC = -11.546 + 1.374^{*}C + 1.126^{*}R + 0.311^{*}\overline{A}.$$

(2) *Open-source PHP-based system*. The following model was built from the dataset shown in Table A.III (32 systems) and validated using the dataset shown in Table A.IV (31 systems):

$$KLOC = -13.223 + 1.241^{*}C + 1.672^{*}R + 0.652^{*}\overline{A}.$$

(3) *Java-based system*.
   (i) *Industry Java-based system*. The following model was built from the dataset shown in Table A.V and validated using the dataset shown in Table A.VI (16 systems each):

$$KLOC = -10.729 + 1.324^{*}C + 1.254^{*}R + 0.889^{*}\overline{A}.$$

   (ii) *Open-source Java-based system*. The following model was built from the dataset shown in Table A.VII (30 systems) and validated using the dataset shown in Table A.VIII (24 systems):

$$KLOC = -10.121 + 1.201^{*}C + 1.439^{*}R + 0.726^{*}\overline{A}.$$

   (iii) *Combined industry and open-source Java-based system*. The following model was built from the combination of the two datasets shown in Tables A.V and A.VII (46 systems) and validated using the combination of the two datasets shown in Tables A.VI and A.VIII (40 systems):

$$KLOC = -10.576 + 1.258^{*}C + 1.392^{*}R + 0.754^{*}\overline{A}.$$

The test results for the models built are shown in Table I. In the significance tests, all the tests—(Pr $> F$), and (Pr $> |t|$) for intercept, C, R, and $\overline{A}$—fell well within acceptable levels (less than 0.05). In the fitness tests, all R_a^2 showed good fit (more than 0.75). In multicollinearity tests, the variance inflation factors (VIFs) and condition indices of all the independent variables were less than 5 and 10, respectively (the thresholds). Therefore, there was no evidence of the existence of multicollinearity between independent variables. In the extreme case tests, except for the following systems, the absolute values of studentized residuals (RStudent) for all the systems were below 2:

(1) In the model building dataset for the open-source Java-based system (Table A.VII), the studentized residuals (RStudent) for northstarbbs and xc-ast were 2.2146 and 2.3948, respectively. We checked the data for these two systems and found them correct. Thus, no adjustment to the dataset

was made. These systems accounted for about 6.7% (2 out of 30) of the systems in the dataset with their absolute RStudent above 2 and below 3. This was very close to the expected 5% in a good model.

(2) In the model building dataset for the combined industry and open-source Java-based systems (the combination of Tables A.V and A.VII), the studentized residuals for jcv, northstarbbs, Sacash, and xc-ast were −2.0230, 2.7722, −2.2772, and 2.2337, respectively. We checked the data for these four systems and found them correct. Thus, no adjustment to the dataset was made. These systems accounted for about 8.7% (4 out of 46) of the systems in the dataset with their absolute RStudent above 2 and below 3. This was not far from the expected 5% in a good model. We believe that the slightly higher percentage could be due to the combination of the two datasets with different accuracies in their conceptual data models—the conceptual data models for the open-source datasets were more accurate than the conceptual data models for the industry datasets as they were constructed through reverse engineering from database schemas and program source code.

Therefore, all the test results were affirmative.

All the measures for model validation are shown in the bottom region of Table I. The MMRE and Pred(0.25) values for all the cases fell well within acceptable levels (not more than 0.25 and not less than 0.75, respectively). Therefore, the validation result strongly supports the validity of the respective LOC estimation model built.

4.2.2 *Using the 10-Fold Cross-Validation Method.* Applying the 10-fold cross-validation method, we built the following five models from the five pairs of datasets collected:

(1) *Industry VB-based system*. The following model was built and validated using a single dataset of 32 systems formed by the combination of the two datasets shown in Tables A.I and A.II:

$$KLOC = -4.5866 + 1.204^{*}C + 1.1711^{*}R.$$

(2) *Open-source PHP-based system*. The following model was built and validated using a single dataset of 63 systems formed by the combination of the two datasets shown in Tables A.III and A.IV:

$$KLOC = -12.0275 + 1.3562^{*}C + 1.4302^{*}R + 0.6074^{*}\overline{A}.$$

(3) *Java-based system*.

(i) *Industry Java-based system*. The following model was built and validated using a single dataset of 32 systems formed by the combination of the two datasets shown in Tables A.V and A.VI:

$$KLOC = -8.8106 + 1.3522^{*}C + 1.1317^{*}R + 0.7573^{*}\overline{A}.$$

(ii) *Open-source Java-based system*. The following model was built and validated using a single dataset of 54 systems formed by the combination of the two datasets shown in Tables A.VII and A.VIII:

$$KLOC = -10.3849 + 1.0221^{*}C + 1.5895^{*}R + 0.1251^{*}\overline{A}.$$

Table I. Summary of Statistics from Holdout Method for the Proposed LOC Estimation

Dataset			Industry VB-based	Open-source PHP-based	Industry Java-based	Open-source Java-based	Combined Java-based
Tests for model building							
Significance test	(Pr > F)		<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
	(Pr > $\|t\|$)	intercept	0.0032	<0.0001	0.0008	0.0017	<0.0001
		C	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
		R	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
		$\overline{A}$	0.0491	0.0467	0.0178	0.0499	0.0051
Fitness test	R_a^2		0.9737	0.9438	0.9910	0.9587	0.9737
Multicollinearity test	VIF of indep. variable		All <5	All <5	All <5	All <5	All <5
	Condition index of indep. variable		All <10	All <10	All <10	All <10	All <10
Extreme case test	|RStudent| for system		All <2	All <2	All <2	All <2 except 2 systems (6.7 %) above 2 but below 3	All <2 except 4 systems (8.7 %) above 2 but below 3
Measures for model validation							
MMRE			0.18	0.14	0.15	0.21	0.18
Pred(0.25)			0.81	0.81	0.81	0.79	0.78

(iii) *Combined industry and open-source Java-based system*. The following model was built and validated using a single dataset of 86 systems formed by the combination of the four datasets shown in Tables A.V, A.VI, A.VII and A.VIII:

$$KLOC = -10.2506 + 1.1594^{*}C + 1.3858^{*}R + 0.0667^{*}\overline{A}.$$

The test results for the models built are shown in Table II. In the significance tests, except that (Pr $>|t|$) for $\overline{A}$ for the industry Java-based system is slightly above 0.05 (0.0783), all the tests—(Pr $> F$), and (Pr $>|t|$) for intercept, C, R, and $\overline{A}$—fell well within acceptable levels (less than 0.05). In the fitness tests, all R_a^2 showed a good fit (more than 0.75). In the multicollinearity tests, the variance inflation factors of all the independent variables were less than 5. Except the condition index of the independent variable R for the industry VB-based system (which was 11.36, slightly higher than the first-level threshold 10, but lower than the second-level threshold 15), the condition indices of all the independent variables were less than 10 (the first threshold). Therefore, there was no evidence of the existence of multicollinearity between the independent variables. In the extreme case tests, except for the following systems, the absolute values of studentized residuals (RStudent) for all the systems were below 2:

(1) In the dataset for the open-source PHP-based system (the combination of Tables A.III and A.IV), the studentized residuals (RStudent) for Quantum_Star_SE, Infocentral, e107, and Commerce were −2.144, 2.219, 2.202, and 2.266, respectively. These systems accounted for about 6% (4 out of 63) of the systems in the dataset with their absolute RStudent above 2 and below 3. This is very close to the expected 5% in a good model.

(2) In the dataset for industry Java-based system (the combination of Tables A.V and A.VI), the studentized residuals (RStudent) for Systems 8 and 11 from Table A.VI are −2.419 and −2.272, respectively. These systems accounted for about 6% (2 out of 32) of the systems in the dataset with their absolute RStudent above 2 and below 3. This is very close to the expected 5% in a good model.

(3) In the dataset for open-source Java-based system (the combination of Tables A.VII and A.VIII), the studentized residuals (RStudent) for abaguibuilder, imcms, Openjms, and sqlunit were 2.656, −2.018, 2.692, and 2.397, respectively. These systems accounted for about 7% (4 out of 54) of the systems in the dataset with their absolute RStudent above 2 and below 3. This is close to the expected 5% in a good model.

(4) In the dataset for combined industry and open-source Java-based system (the combination of Tables A.V, A.VI, A.VII, and A.VIII), the studentized residuals (RStudent) for System 8 in Table A.VI, abaguibuilder, imcms, Openjms, and sqlunit, were −2.313, 3.010, −2.212, 3.101, and 2.7130, respectively. These systems accounted for about 5.8% (5 out of 86) of the systems in the dataset (very close to the expected 5% of data points in the sample exceeding 2 in a good model). Except that the absolute values of RStudent for abaguibuilder and Openjms were slightly above 3, the

remaining three systems were all below 3. The latter could be due to the combination of industry and open-source datasets that have different accuracies in their data models, as explained earlier in the use of the holdout method.

Therefore, the overall test results were affirmative.

All the measures for model validation are also shown at the bottom of Table II. Except for the open-source PHP-based system, the MMRE and Pred (0.25) values for all the cases fell well within acceptable levels (not more than 0.25 and not less than 0.75, respectively). The MMRE and Pred(0.25) values for the open-source PHP-based system were 0.308 and 0.651, respectively. Though these values did not fall within acceptable levels, they were close to those levels. Therefore, the overall validation results were supportive.

4.3 Summary of the Results

The overall result of model building and validation supports the proposed LOC estimation model, although there are two minor setbacks that we will discuss shortly.

All the MMRE and Pred(0.25) values from the holdout and 10-fold cross-validation methods are shown at the bottoms of Tables I and II, respectively. The MMRE and Pred(0.25) values from both the holdout and 10-fold cross-validation methods for four out of five pairs of datasets fell well within acceptable levels (0.25 and 0.75, respectively). Such results have been reckoned as good by other researchers [Briand and Wieczorek 2002; Lai and Huang 2003]. The MMRE and Pred(0.25) values for the open-source PHP-based system using the 10-fold cross-validation method were 0.308 and 0.651, respectively. This is the only case where the MMRE and Pred(0.25) values did not fall within acceptable levels. It was the first setback. Though these values did not fall within acceptable levels, they were quite close to those levels (0.25 and 0.75, respectively). Furthermore, the MMRE and Pred(0.25) values for the open-source PHP-based system using the holdout method fell well within acceptable levels. Another setback was the exclusion of $\overline{A}$ (the average number of attributes per class) in the model built by the 10-fold cross-validation method for the industry VB-based system. We carried out a study on the two setbacks. Our study discovered the following characteristics for the two pairs of datasets involved:

(1) In the PHP-based datasets, there were many small systems with small KLOC—40% of the systems had sizes less than 10 (25 out of 63). Therefore, when using MMRE as a validation measure, the deviation was bigger. This problem of using MMRE as a validation measure has been reported in Foss et al. [2003].

(2) In comparing the datasets, $\overline{A}$ was much higher in VB-based datasets. Furthermore, there were 5 out of 32 systems with $\overline{A}$ greater than 20. Classes of these systems were mainly used to store parameters to make the systems more flexible, and some of the parameters were not used. Among the 149 systems in other pairs of datasets, none of their $\overline{A}$ values were greater than 20.

Table II. Summary of Statistics from 10-Fold Cross-Validation Method for Proposed LOC Estimation Method

Dataset			Industry VB-based	Open-source PHP-based	Industry Java-based	Open-source Java-based	Combined Java-based
Tests for model building							
Significance test	(Pr > F)		<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
	(Pr > $\|t\|$)	Intercept	<0.0001	<0.0001	0.014	<0.0001	<0.0001
		C	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
		R	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001
		$\overline{A}$	—	<0.0001	0.0783	<0.0001	<0.0001
Fitness test	R_a^2		0.946	0.947	0.965	0.921	0.937
Multicollinearity test	VIF of indep. variable		All <5	All <5	All <5	All <5	All <5
	Condition index of indep. variable		All <10 except for R = 11.36 (<15)	All <10	All <10	All <10	All <10
Extreme case test	|RStudent| for system		All <2	All <2 except 4 systems (6%) above 2 but below 3	all <2 except 2 systems (6%) above 2 but below 3	All <2 except 4 systems (7%) above 2 but below 3	All <2 except 5 systems (5.8%) above 2 but below 3
Measures for model validation							
MMRE			0.128	0.308	0.106	0.184	0.153
Pred(0.25)			0.938	0.651	0.938	0.759	0.826

We also observed that, in most of the models built, the magnitude of the coefficient for $\overline{A}$ was the lowest among the coefficients for the three independent variables. This is consistent with the intuitive understanding. Much complexity in information systems is caused by the maintenance of class instances and the navigation between these instances. The number of attributes in classes does not affect the complexity as much as the number of classes and relationship types. Therefore, C and R have much greater influence than $\overline{A}$ on LOC.

4.4 Threats to Validity

Many organizations in the software industry did not construct formal conceptual data models for information systems developed in the early stage of software development. Furthermore, to ensure that the data collected from a system was correct, we had to verify the data with the system documentations ourselves. Because of confidentiality concerns, many industry organizations did not allow us to access their system documentation. This made data collection from industry sources extremely difficult. As a result, the sizes of our industry datasets just managed to meet the minimum criterion (15 data points). This may have affected the accuracy of our validation. To address this issue, in addition to the industry datasets, we also collected some datasets from open-source systems to build and validate models.

The sizes of the samples collected from open-source systems were much better. Furthermore, the datasets collected from open-source systems also had the benefit of allowing other people to verify their accuracy. However, the conceptual data models for these systems had to be constructed from reverse reengineering of their database schemas and program source codes. Therefore, they were usually more accurate than those models constructed in the early stage of software development.

Clearly, it was not possible for our samples to cover the full range of possible LOC values. So the models we built are not appropriate for systems with LOC values that are significantly larger or smaller than the LOC values in the respective samples. The current model validation criteria on using MMRE and PRED(0.25) are also not perfect [Stensrud et al. 2002].

In summary, we do not claim that we have built models ready for general use. However, in the worst case, our work has shown that it is useful and promising to experiment with the proposed method on a much larger scale to address the critical software sizing problem in industry. Clearly, such experimentation requires much larger organized cooperation between industry and research communities on a long-term basis.

4.5 Further Investigation

Overall, our empirical studies have supported the proposed LOC estimation method—the use of a conceptual data model for the estimation of LOC for information systems. The models built in this study can be used to estimate the LOCs of systems with similar characteristics.

To address the general applicability of the proposed estimation method and the models built, we have recently investigated the common characteristics of the systems in each pair of datasets. We have also investigated other characteristics, in addition to programming language, of a project, which may call for different models to be built. Next, we shall discuss our investigation.

In the function point method, 14 general system characteristics are used as adjustment factors to incorporate pervasive general factors that may not be sufficiently represented by function points [Garmus and Herron 2000]. The 14 factors are data communications, distributed data processing, performance, heavily used configuration, transaction rate, online data entry, end-user efficiency, online update, complex processing, reusability, installation ease, operational ease, multiple sites, and facilitate change.

The pervasive general factors represented by these 14 general system characteristics may also not be sufficiently represented by the conceptual data model for an information system. Based on these, in our investigation, we examined the characteristics of the systems in our datasets. The following four additional characteristics were discovered in the course of the investigation:

(1) *Reuse level*. This describes the degree to which a system is developed through reuse of code from existing resources including an external library. Note that reusability in terms of the 14 general system characteristics describes the reusability of a target system's code.

(2) *Security*. This describes the degree of security required by a system. Security is an important aspect for some Web-based information systems. The security requirement includes the implementation of algorithms for encryption and user authentication purposes. It also includes the implementation of additional input validation on both client and server programs to validate any input received from a client program and processed by the server program. The latter is to prevent security violations such as SQL injection, etc.

(3) *GUI requirement*. This describes the complexity of GUI-related requirements including the input data structure, decoration, help and guidance feature, etc. The GUI requirement ranges from simple form-based GUIs to very complicated GUIs needed to deal with complex input data structure, to guide users or to support other specific needs.

(4) *External report and inquiry*. This describes the quantity of reports and inquiries required by a system. Some systems may need a much larger number of reports and inquiries to present similar data in different formats. Note that external reports and inquiries are two of the factors that the FP method is directly based on. The proposed approach does not represent them directly in the estimation model. If a system requires a large number of reports and inquiries by presenting similar data in different formats, such a requirement may not be fully represented by the conceptual data model on which the proposed method is based.

All the systems in our five pairs of datasets were information systems with the general properties discussed at the beginning of Section 3.1 (our sampling criteria). In addition, within each pair of industry datasets, most systems are similar in respect to the 18 characteristics (14 from the FP method and four newly introduced by us). Within each pair of open-source datasets, many systems exhibited moderate differences in a few characteristics compared with the majority of the systems in the datasets.

Across the pairs of datasets, systems in our industry datasets exhibited significant differences on transaction rates, end-user efficiencies, and complex processing, compared to systems in our open-source datasets. The industry systems were significantly more complex in regard to these characteristics.

In conclusion, our investigation suggests that, in addition to programming language, major differences on the above-mentioned 18 characteristics may call for a different model to be built. Alternatively, the 18 characteristics could be used to adjust/fine-tune the proposed method. We believe that calibrating the proposed method through using a measure that quantifies the impact of each of these characteristics may help develop a more generic-sized estimation model. This could be a further research area.

5. COMPARISON OF THE PROPOSED METHOD WITH EXISTING METHODS

We compared the proposed approach with the FP method and the use-case point method as they are the most well-known existing software sizing methods. The FP method was originally proposed to estimate development effort in two steps: (1) based on FP to estimate LOC; (2) based on LOC to estimate development effort [Albrecht and Gaffney 1983]. The use-case point method was designed to estimate the development effort directly.

Due to the fact that all available existing FP linear regression models for estimating LOC are built using the holdout method, in the comparison, all the multiple linear regression models were built and validated using this method. The comparison was based on the open-source datasets only because the required information was not available for industry datasets.

5.1 FP Method

Though FP also has problems and limitations [Briand and Wieczorek 2002], it is still the most widely used sizing technique in industry. One major problem of the FP method is the difficulty in obtaining the required information in the early stage of software development. Since the proposed method is based on a conceptual data model, it has a clear advantage over the FP method in having the required information more readily available in the early stage of software development. Therefore, the main objective of this comparison is to see whether the accuracy of the proposed estimation method is comparable to the FP method.

In the comparison, for the FP method, we used adjusted function point (FP): $FP = VAF \times UFP$. A value adjustment factor (VAF) was calculated as follows: $VAF = (TDI \times 0.01) + 0.65$. The TDI was calculated as the total of all the degrees of influence (each of them ranked from 0 to 5) by the 14 general system characteristics [Garmus and Herron 2000]. We counted all UFPs and VAFs through reading the system documents and conducting testing. The counting was carried out by a student in his master of science dissertation project.

5.1.1 *Open-Source PHP-Based System.* For PHP-based systems, since there is no existing formula available for estimating LOC directly from the function point, we built multiple linear regression models in the same way as we did for the proposed method (a simple linear regression model in this case as there was only one independent variable, FP). Therefore, UFPs and VAFs were counted for all the open-source PHP-based systems in both the datasets discussed earlier. The FPs and related data for the dataset that built the proposed model are shown in Table A.IX in the Appendix. In the same manner as building the proposed model, we built the following linear regression model from the dataset shown in Table A.IX for estimating KLOC from FP:

$$Size = -2.485 + 0.055FP.$$

The model was tested as discussed in Section 2. In the significance test, (Pr $>$ F) $<$ 0.0001. (Pr $>$ |t|) for intercept and FP were $<$ 0.0001 and $<$ 0.0001, respectively. In the fitness test, $R_a^2 = 0.99$. In extreme case test, the absolute values of the studentized residual (RStudent) for all the systems were below 2 (the threshold). Multicollinearity tests were not required in this case as we only had one independent variable. Therefore, all the test results were affirmative.

Following the same dataset for validating the proposed model, the LOC estimation model from the FP method was validated using the data shown in Table A.X. The MMRE and Pred(0.25) computed values were 0.14 and 0.77, respectively. Note that, for the LOC estimation model built from the proposed method, the MMRE and Pred(0.25) values computed for the same validation dataset were 0.14 and 0.81, respectively. Based on MMRE and Pred(0.25) as measures for accuracy, there was no clear difference between the accuracy of the proposed method and the FP method. The details of the comparison between the two methods are shown in Table A.X.

5.1.2 *Open-Source Java-Based System.* For the Java-based language, the formula for converting FP to LOC is available [Boehm et al. 2000]. The estimated LOC is function point (FP) multiplied by 53. Therefore, UFPs and VAFs were counted for only the validation dataset. The details of the comparison between the two methods are shown in Table A.XI.

Following the same dataset for validating the proposed model, the LOC estimation model from the FP method was validated using the data shown in Table A.XI. The MMRE and Pred(0.25) computed values were 0.17 and 0.75, respectively. Note that, for the LOC estimation model built from the proposed method, the MMRE and Pred(0.25) values computed for the same dataset were 0.21 and 0.79, respectively. Based on MMRE and Pred(0.25) as measures for accuracy, there was also no clear difference between the accuracy of the proposed method and the FP method.

5.1.3 *Conclusion.* Both comparisons have shown that, in terms of estimating LOC, the accuracy of the proposed method is comparable to the FP method. The advantage of the proposed method is that it is based on a conceptual data model which is more readily available in the early stage of software development and yet can obtain an LOC estimate comparable to the FP method in terms of accuracy. As the FP method can also estimate development effort directly, we cannot draw any general conclusion about the accuracy between the two methods from this comparison. However, the comparison does show some potential on the usefulness of the proposed method for software sizing.

5.2 Use-Case Point Method

The use-case point method has been explored to estimate software effort during the early stage of software development [Mohagheghi et al. 2005]. To extend the comparison of the proposed LOC estimation method with existing methods, we recently compared the LOC estimation from the proposed method with the LOC estimation based on use-case points. For the latter estimation, we built a linear regression model based on use-case points.

We collected the samples from open-source systems in 2005. When we revisited these systems recently, some of them had been removed and become proprietary systems. Thus, they were no longer accessible. Therefore, for the model building and validation using use-case points, we could only use those systems that were still accessible.

5.2.1 *Open-Source PHP-Based System.* For open-source PHP-based systems, the use-case points for the systems in the dataset that built the proposed model and were still accessible, are shown in Table A.XII. Using the holdout method, the following linear regression model was built from these systems for estimating LOC from use-case points (UCP):

$$Size = 4.2857 + 0.1049^{*}UCP.$$

The model was tested as discussed in Section 2. In the significance test, (Pr $>$ F) $<$ 0.001, and (Pr $>$ |t|) for the intercept and UCP were 0.21 and $<$ 0.001, respectively. In the fitness test, $R_a^2 = 0.603$. Therefore, none of the test results were affirmative.

The use-case points for the open-source PHP-based systems in the dataset that validated the proposed model and were still accessible are shown in Table A.XIII. The above model was validated using this dataset. The MMRE and Pred(0.25) computed values were 0.871 and 0.154, respectively. Both were far from acceptable levels. Therefore, the use-case point method failed to build a reasonable model from the systems in our datasets for estimating LOC.

5.2.2 *Open-Source Java-Based System.* For open-source Java-based systems, the use-case points for the systems in the dataset that built the proposed model and were still accessible, are shown in Table A.XIV. Using the holdout method, the following linear regression model was built from these systems for estimating LOC from use-case points (UCP):

$$Size = 23.7394 + 0.09^{*}UCP.$$

The model was tested as discussed in Section 2. In the significance test, (Pr > F) = 0.057, while (Pr > |t|) for the intercept and UCP were 0.003 and 0.057, respectively. In the fitness test, $R_a^2 = 0.102$. Therefore, none of the test results were affirmative.

The use-case points for the open-source Java-based systems in the dataset that validated the proposed model and were still accessible are shown in Table A.XV. The above model was validated using this dataset. The MMRE and Pred(0.25) computed values were 1.131 and 0.27, respectively. Both far from acceptable levels. Therefore, the use-case point method failed to build a reasonable model for estimating LOC from the systems in our datasets.

5.2.3 *Conclusion.* In contrast to the proposed method, both the open-source PHP-based and Java-based datasets failed to build acceptable LOC estimation models from the use-case point method. Furthermore, we also found that, in the use-case point method, it was not easy to estimate information required (such as the complexity of the use-cases) in the early stage of software development. As the use-case point method is designed to estimate development effort directly, we cannot draw any general conclusion about the accuracy of the two methods from this comparison. However, the comparison does show some potential on the usefulness of the proposed method for software sizing.

6. FROM SIZE TO EFFORT ESTIMATION

Software effort estimation is a crucial task in the software industry. Estimated effort and market factors are keys for developers to price a software development project. Project plans and schedules are also based on the estimated effort. Therefore, the impact of effort estimation on the success of a software project is crucial.

Considerable research work has been spent on software effort estimation. In terms of the basic estimation methods used, many are algorithmic in nature. However, there have also been explorations into the use of machine learning or nonalgorithmic methods. The major drivers used in effort estimation models are size, team experience, technology, reuse resources, and required reliability. Among these, size is a main driver. The basic measure of effort is the man-hour.

In existing software effort estimation models, the most commonly used size measures are LOC and FP. Many of them can use both LOC and FP as size measures. Effort estimation models that can use LOC as the size measure include COCOMO II, SLIM, SEER-SEM, etc. COCOMO II is an enhanced model of COCOMO (Constructive Cost MOdel) [Boehm et al. 2000]. Both the original and enhanced models were proposed by Boehm. They are well-known cost and effort estimation models, and can be used to estimate the effort required by software projects at both the early design stage and the postarchitecture stage with effort drivers structured at different levels of detail. SLIM was proposed by Putnam [Putnam and Myers 1992]. It is based on the analysis of the lifecycle in terms of the Rayleigh distribution of project personnel level versus time. SEER-SEM (System Evaluation and Estimation of Resources—Software Estimation Model) is a commercial tool based on the original Jensen model [Jensen 1983]. The parametric-based modeling equations used in SEER-SEM are proprietary.

For a large class of information systems, the proposed approach can be used to estimate the LOC of these systems. The estimated LOC can then be used as the size input to the above-mentioned LOC-based effort estimation models. Therefore, the use of the proposed method to supply the size estimate for these models is straightforward.

A good size estimate is very important for effort estimation, and the estimation of size is still challenging [Boehm et al. 2000]. As the proposed software size estimation model is based on a conceptual data model that can be more accurately constructed in the early stage of software development, it has the advantage over the existing methods in terms of the derivation of the information required. Therefore, the proposed method addresses an important problem in existing software sizing methods: required information that is difficult to predict accurately in the early stage of software development. Consequently, it may help in software effort estimation through providing better software size estimation.

7. RELATED WORK

Existing software sizing methods have the problem of using information that is difficult to derive in the early stage of software development. Since the proposed method estimates LOC from a conceptual data model, all the information required by the method is fully available at the end of requirements analysis stage. Furthermore, the derivation of the data required from the conceptual model for the proposed method is simple and not subject to further judgment, decision, and interpretation. As such, the proposed method addresses the problems faced by existing software sizing methods.

Estimation of software size is an important task in software effort estimation. The most commonly used size measures are line of code (LOC) and function point (FP). Due to the difficulty of obtaining the information required by existing sizing methods in the early stage of software development, many practitioners and project managers continue to produce estimates based on ad-hoc or so called “expert” approaches [Armour 2002; Miranda 2000]. Despite the problems, the function point method has been used widely to estimate the LOC of business systems and also to directly predict the effort and cost of software projects [Costagliola et al. 2005]. Some variants of the FP method have also been proposed. In addition to the above-mentioned major problem, although originally conceived to be independent of methodology used to develop the system under measurement, the application of the FP method turns out to be rather unnatural when it is applied to object-oriented (OO) systems [Costagliola et al. 2005]. As a consequence of using use-cases to model software requirements in the OO approach, the use-case point method has been proposed to estimate software effort [Smith 1999]. However, information such as the complexity of use-cases and the details of the scenarios required by this method are difficult to predict in the early stage of software development. Recently, the class point (CP) method that generalizes the FP method for OO systems was also proposed [Costagliola et al. 2005]. The CP method is based on information from design documentation. Clearly, much of the information required by the CP method is available only when the system design has been completed.

As the proposed method is based on a conceptual data model, in contrast to the above-mentioned methods, the information required by the proposed method is more readily available in the early stage of software development. In the worst case, all the information required can also be fully available when the requirements analysis has been completed (i.e., before the design begins). The proposed estimation method shares the use of a class diagram with the CP method. However, the CP method requires much detailed design information about the classes. The proposed method does not require such information. In terms of domain of applications, the proposed method is domain specific. It is for information systems. The use of a domain-specific method has been identified as a key to improving software size estimation. The above-mentioned related methods are for general use.

8. CONCLUSION

We have proposed and validated a novel method for estimating LOC for a large class of information systems as defined in the beginning of Section 3.1 from their conceptual data models. We have also conducted experiments comparing the proposed method with the well-known FP method and the use-case point method on estimating LOC. Our empirical comparison shows that the proposed method is comparable with the FP method for estimating LOC in term of accuracy.

Information systems constitute a large software domain in the software industry. There is still much problem in using existing methods for estimating the sizes of these systems in the industry [Miranda 2000]. As the proposed method is based on a conceptual data model, the information required by the proposed method can clearly be more accurately obtained than by the existing software sizing methods in the early stage of software development. Therefore, we believe that the proposed method is promising in providing a key to addressing this crucial problem in the software industry. Based on our experience with the FP method, there is no doubt that much work is still required to fine-tune the proposed method in order to put it into practical use. It is also highly probable the proposed method can estimate project effort and cost directly based on the three independent variables used (C, R, and $\overline{A}$) in a similar way as the FP method. These are important research areas to be pursued further.

APPENDIX

Tables A. I–A.XV in this Appendix show all the datasets collected.

Table A.I. Dataset A from Industry VB-Based Systems

System no.	Actual size (KLOC)	C	R	$\overline{A}$
1	37.54	27	8	8.000
2	14.723	8	6	25.375
3	24.667	12	10	16.917
4	42.1	19	25	16.526
5	87.23	35	38	8.343
6	31.445	14	21	6.214
7	67.04	35	27	16.829
8	30.79	17	20	6.176
9	22.402	13	14	5.769
10	69.713	28	31	9.571
11	16.17	6	9	27.333
12	90.854	37	39	21.270
13	64.35	27	33	5.481
14	27.076	13	16	8.000
15	20.933	10	10	6.500
16	40.341	22	20	5.818

Table A.II. Dataset B from Industry VB-Based Systems

System no.	Actual size (KLOC)	C	R	$\overline{A}$
1	27.217	16	8	6.188
2	14.53	7	6	16.286
3	65.872	28	24	13.179
4	41.435	24	21	14.417
5	72	36	37	15.000
6	29.52	16	16	27.563
7	52.76	31	24	10.645
8	46.92	29	24	6.931
9	97.88	42	44	33.524
10	38.764	18	12	11.000
11	31.665	12	14	8.000
12	77.52	30	32	7.533
13	16.81	10	7	9.500
14	59.332	32	26	14.781
15	107.8	41	45	16.561
16	53.66	24	18	9.125

Table A.III. Dataset A from Open-Souce PHP-Based Systems

System	Actual size (KLOC)	C	R	$\overline{A}$
Bannerex3a	3.038	5	2	10.600
Castor	22.599	17	7	7.000
Cmsmadesimple	32.243	21	13	4.524
Comendar	16.164	13	11	7.077
Commerce	83.862	35	24	6.571
Coppermine	24.22	13	9	8.077
EclipseBB	63.929	35	19	8.029
Jdcms	2.543	5	3	9.400
Linkbase	6.697	5	5	7.000

(*Continued on next page*)

Table A.III. Table (*Continued*)

System	Actual size (KLOC)	C	R	$\overline{A}$
Linpha	55.537	25	14	8.640
Lotgd	55.752	39	10	9.077
Mailwatch	62.602	30	17	7.000
Mantis	67.111	23	22	14.957
Mundimail	2.552	3	1	8.333
opendocman	12.17	10	5	3.700
Openrating	12.757	13	9	5.000
php4Flicks	5.695	7	3	8.429
Phpalumni	7.744	9	6	9.222
Phpcollegeex	7.514	4	1	8.000
Phpman	11.054	9	9	3.667
phpmylibrary	29.77	17	15	3.412
phpstudentcenter	11.653	9	8	8.778
phptimesheet	6.847	5	4	3.600
Poppawid	13.389	7	5	11.714
Refbase	14.45	12	6	16.583
Replex	4.414	6	3	3.667
Timeclock	2.102	3	1	3.333
Uccass	42.819	20	18	3.500
Uma	4.077	4	2	9.000
Vallheru	57.408	33	14	9.242
WebFileSystem	7.428	7	3	7.000
Winventory	8.947	15	5	4.000

Table A.IV. Data Set B from Open-Source PHP-Based System

System	Actual Size (KLOC)	C	R	$\overline{A}$
ackerTodo	9.47	8	5	5.000
Bookmark4u	19.381	10	10	5.800
ByteMonsoon	4.219	6	2	8.000
Core	17.065	8	5	8.125
e107	65.649	34	12	7.912
eFiction	12.443	9	6	6.222
Fdcl	3.856	5	2	11.600
Gallant	10.225	6	4	11.333
Infocentral	47.734	19	13	8.053
Jamdb	4.554	4	5	8.500
Openrealty	16.846	9	8	6.333
phpESP	33.645	16	15	5.438
Phpnews	6.532	6	5	6.167
Phpollster	1.818	4	3	7.000
PhpScheduleIt	16.006	6	9	8.667
Phpsera	15.846	6	6	7.000
Phpwims	10.507	6	4	9.333
Phpwscookbook	7.607	5	5	10.200
Plume	15.875	11	7	6.455
Quantum_Star_SE	57.643	30	23	9.533
Rasmp	9.452	10	4	6.600
Rimps	10.81	10	6	6.500
so-net	38.781	20	14	10.100

(*Continued on next page*)

Table A.IV. Table (*Continued*)

System	Actual Size (KLOC)	C	R	$\overline{A}$
Supersurf	1.489	4	3	7.000
Usebb	6.118	6	3	13.333
Videodb	24.227	14	10	4.643
Webaddressbook	8.363	3	2	20.000
Wikiwig	23.457	15	9	5.800
Yabbse	22.475	11	10	12.818
Zebraz	8.59	6	5	7.833
Ztml	11.724	10	3	9.600

Table A.V. Dataset A from Industry Java-Based Systems

System no.	Actual size (KLOC)	C	R	$\overline{A}$
1	30.02	22	6	8.182
2	37.288	16	18	6.188
3	60.102	23	26	9.261
4	46.27	24	18	6.583
5	85.009	42	26	5.738
6	12.62	6	5	6.167
7	80.014	40	25	6.675
8	47.658	20	20	5.200
9	20.53	10	11	7.500
10	56.48	24	25	7.792
11	33.602	14	15	7.286
12	63.1	20	26	14.2
13	92.841	45	26	11.622
14	22.08	10	13	6.400
15	14.89	7	6	5.714
16	100.213	52	27	9.481

Table A.VI. Dataset B from Industry Java-Based Systems

System no.	Actual size (KLOC)	C	R	$\overline{A}$
1	84.89	49	22	5.898
2	20.446	12	7	4.250
3	17.29	8	6	7.750
4	34.075	16	12	5.125
5	40.113	18	14	6.389
6	28.722	17	14	7.412
7	71.37	31	27	6.871
8	60.34	30	29	10.300
9	20.45	11	11	5.091
10	93.27	35	36	8.486
11	31.69	19	20	4.789
12	77.52	30	33	6.267
13	45.384	21	24	7.714
14	52.1	24	28	6.000
15	19.36	7	9	4.857
16	56.744	18	26	5.944

Table A.VII. Dataset A from Open-Source Java-Based Systems

System	Actual size (KLOC)	C	R	$\overline{A}$
Chatterbox	11.717	8	6	4.250
churchinfo	47.52	23	19	9.565
cream	84.01	26	40	11.462
dlog4j	26.999	15	14	8.933
dynasite	41.72	20	15	5.900
e-library	13.015	5	6	12.400
elips	30.402	18	7	6.611
Forumnuke	29.159	23	10	6.957
GeneaPro	53.443	28	25	4.179
Ibatis	18.694	13	9	6.615
i-tor	26.384	16	6	5.125
Itracker	38.721	19	16	6.579
jcv	75.643	26	30	6.154
jwordnet	46.72	21	24	6.048
jwp	6.413	7	5	4.143
kaon	79.534	20	37	4.850
Kbvt	36.343	18	17	5.333
malbum	59.684	22	31	6.182
northstarbbs	50.454	15	20	11.600
Personalblog	3.055	4	1	7.000
planeta	63.257	34	17	3.971
racetrack	91.28	35	28	13.571
roller	32.707	11	17	7.545
s2j-0.94	11	5	5	3.600
Sacash	5.543	6	4	3.833
Sixqos	22.686	12	11	6.667
Storyserver	3.911	3	2	6.667
tinapos	20.841	14	7	3.000
Webcockpit	9.269	6	5	3.500
xc-ast	7.732	7	2	11.143

Table A.VIII. Dataset B from Open-Source Java-Based Systems

System	Actual size (KLOC)	C	R	$\overline{A}$
abaguibuilder	83.176	29	23	7.345
Art	17.67	14	7	6.929
Bofhms	10.507	12	4	4.333
Contineo	21.067	11	11	7.364
Dspace	47.57	32	20	4.250
Ejen	4.437	5	3	4.600
Imcms	91.424	59	30	4.712
Itracker	38.721	19	14	6.579
jdbforms	50.72	19	15	6.789
Jmbase	19.723	8	8	5.250
Jwma	63.228	22	19	8.045
Jbooks	12.817	11	7	3.727
Jfolder	29.288	10	8	12.800

(*Continued on next page*)

Table A.VIII. Table (*Continued*)

System	Actual size (KLOC)	C	R	$\overline{A}$
Jgossip	28.047	14	9	5.714
Jpo	8.54	5	5	5.800
Jwnl	6.913	7	5	3.571
mp3cattle	14.598	9	10	3.444
Openjms	59.72	17	16	6.824
Openhre	9.932	9	4	3.444
Quartz	22.954	11	11	4.364
Sqlunit	74.089	26	20	7.808
tau_lastest	48.039	14	16	8.000
Testsuite	30.076	15	9	5.067
Tesuji	8.01	2	1	16.000

Table A.IX. The FP Data for Model Building Dataset for PHP-Based Open-Source Systems

System	UFP	VAF	FP
Bannerex3a	126	1.01	127
Castor	476	1.01	481
Cmsmadesimple	634	1.02	647
Comendar	337	1.02	344
Commerce	1517	1.03	1563
Coppermine	447	1.02	456
EclipseBB	1190	1.02	1214
Jdcms	99	1.01	100
Linkbase	172	1.03	177
Linpha	1003	1.02	1023
Lotgd	1045	1.01	1055
Mailwatch	1183	1.01	1195
Mantis	1240	1.03	1277
Mundimail	90	1.01	91
opendocman	246	1.01	248
Openrating	265	1.02	270
php4Flicks	119	1.01	120
Phpalumni	201	1.02	205
Phpcollegeex	183	1.01	185
Phpman	249	1.03	256
phpmylibrary	549	1.03	565
phpstudentcenter	244	1.03	251
phptimesheet	140	1.02	143
Poppawid	304	1.02	310
Refbase	283	1.01	286
Replex	103	1.01	104
Timeclock	81	1.01	82
Uccass	782	1.03	805
Uma	120	1.01	121
Vallheru	1065	1.01	1076
WebFileSystem	210	1.01	212
Winventory	212	1.01	214

Table A.X. Comparison of Proposed LOC Estimation Method with FP Method for PHP-Based Open-Source Systems

System	Actual KLOC	Proposed method estimated KLOC	FP method			
			UFP	VAF	FP	Estimated KLOC
ackerTodo	9.47	8.325	205	1.02	209	9.016
Bookmark4u	19.381	19.689	421	1.04	438	21.596
ByteMonsoon	4.219	2.783	129	1.01	130	4.681
Core	17.065	10.363	256	1.02	261	11.877
e107	65.649	54.193	1023	1.01	1033	54.343
eFiction	12.443	12.035	245	1.02	250	11.260
fdcl	3.856	3.889	121	1.01	122	4.237
galant	10.225	8.300	182	1.01	184	7.625
Infocentral	47.734	40.136	684	1.01	691	35.511
jamdb	4.554	5.643	132	1.03	136	4.993
openrealty	16.846	15.451	312	1.03	321	15.190
phpESP	33.645	35.258	659	1.03	679	34.847
Phpnews	6.532	6.604	193	1.03	199	8.448
phpollster	1.818	1.321	87	1.02	89	2.396
PhpScheduleIt	16.006	14.922	274	1.04	285	13.188
Phpsera	15.846	8.819	257	1.01	260	11.791
Phpwims	10.507	6.996	218	1.02	222	9.745
Phpwscookbook	7.607	7.992	169	1.02	172	6.996
plume	15.875	16.340	337	1.02	344	16.421
Quantum_Star_SE	57.643	68.679	898	1.02	916	47.893
Rasmp	9.452	10.178	224	1.01	226	9.958
Rimps	10.81	13.457	262	1.01	265	12.069
so-net	38.781	41.590	747	1.02	762	39.422
Supersurf	1.489	1.321	75	1.03	77	1.764
usebb	6.118	7.932	163	1.01	165	6.570
videodb	24.227	23.898	446	1.02	455	22.536
Webaddressbook	8.363	6.884	177	1.02	181	7.445
Wikiwig	23.457	24.222	438	1.02	447	22.087
Yabbse	22.475	25.505	405	1.03	417	20.458
Zebraz	8.59	7.690	156	1.02	159	6.267
Ztml	11.724	10.462	211	1.01	213	9.236

Table A.XI. Comparison of Proposed LOC Method with FP Method for Java-Based Open-Source Systems

System	Actual KLOC	Proposed method estimated KLOC	FP method			
			UFP	VAF	FP	Estimated KLOC
abaguibuilder	83.176	63.137	1602	1.02	1634	86.602
Art	17.67	21.796	426	1.02	435	23.055
Bofhms	10.507	13.193	223	1.02	227	12.031
Contineo	21.067	24.265	337	1.05	354	18.762
Dspace	47.57	60.177	923	1.03	951	50.403
Ejen	4.437	3.541	57	1.06	60	3.18
Imcms	91.424	107.329	1767	1.04	1838	97.414
Itracker	38.721	37.620	554	1.06	587	31.111

(*Continued on next page*)

Table A.XI. Table (*Continued*)

System	Actual KLOC	Proposed method estimated KLOC	FP method UFP	VAF	FP	Estimated KLOC
jdbforms	50.72	39.212	794	1.02	810	42.93
jmbase	19.723	14.811	298	1.04	310	16.43
jwma	63.228	49.483	1249	1.02	1274	67.522
Jbooks	12.817	15.869	174	1.02	177	9.381
Jfolder	29.288	22.694	388	1.04	404	21.412
Jgossip	28.047	23.793	367	1.03	378	20.034
Jpo	8.54	7.290	131	1.06	139	7.367
Jwnl	6.913	8.074	142	1.03	146	7.738
mp3cattle	14.598	17.579	277	1.05	291	15.423
Openjms	59.72	38.274	854	1.05	897	47.541
Openhre	9.932	8.945	201	1.04	209	11.077
Quartz	22.954	22.087	291	1.04	303	16.059
sqlunit	74.089	55.553	1583	1.03	1630	86.39
tau_lastest	48.039	35.525	682	1.05	716	37.948
testsuite	30.076	24.523	482	1.03	496	26.288
Tesuji	8.01	5.336	122	1.02	124	6.572

Table A.XII. Use-Case Points for Open-Source PHP-Based Systems in Dataset A

System	UCP
Bannerex3a	66
castor	17
Cmsmadesimple	219
comendar	96
Coppermine	389
EclipseBB	641
jdcms	36
Linkbase	61
linpha	436
mantis	229
Mundimail	33
opendocman	89
php4Flicks	43
Phpalumni	98
Phpcollegeex	68
phpman	71
phpmylibrary	114
phpstudentcenter	122
phptimesheet	93
Poppawid	112
refbase	241
Replex	48
Timeclock	28
Uccass	99
Uma	43
vallheru	386
WebFileSystem	76
winventory	88

Table A.XIII. Use-Case Points for Open-Source PHP-Based Systems in Dataset B

System	UCP
ackerTodo	76
Bookmark4u	246
ByteMonsoon	43
Core	194
e107	354
eFiction	159
fdcl	36
galant	104
Infocentral	514
jamdb	48
openrealty	114
phpESP	111
Phpnews	68
phpollster	18
PhpScheduleIt	154
Phpsera	149
Phpwims	179
Phpwscookbook	83
Quantum_Star_SE	744
Rasmp	101
Supersurf	58
usebb	81
Webaddressbook	99
wikiwig	134
yabbse	359
zebraz	61

Table A.XIV. Use-Case Points for Open-Source Java-Based Systems in Dataset A

System	UCP
Chatterbox	67
cream	493
churchinfo	169
dlog4j	144
e-library	122
elips	56
Forumnuke	140
GeneaPro	386
Ibatis	23
i-tor	68
Itracker	156
jcv	84
jwordnet	63
jwp	120
kaon	73
Kbvt	67

Table A.XIV. Table (*Continued*)

System	UCP
malbum	67
northstarbbs	120
Personalblog	57
racetrack	67
roller	227
Sacash	115
Sixqos	33
Storyserver	94
tinapos	188
Webcockpit	18
xc-ast	17

Table A.XV. Use-Case Points for Open-Source Java-Based Systems in Dataset B

System	UCP
abaguibuilder	476
Art	152
Bofhms	61
Contineo	198
Dspace	302
Ejen	18
Imcms	387
jdbforms	336
jwma	162
Jbooks	137
Jfolder	302
Jwnl	47
mp3cattle	97
Openjms	367
Quartz	97

ACKNOWLEDGMENTS

The authors would like to thank Singapore Computer Systems Pte. Ltd., IPACS E-Solution (S) Pte. Ltd., NatSteel Ltd., Great Eastern Life Assurance Co. Limited, JTC Corporation, Adroit Pte. Ltd., the Institute of Technical Education of Singapore, and the National Institute of Education and National Computer Systems Pte. Ltd. for providing their project data. Without their support, this work would not have been possible. The authors also thank their MSc. student, Di Wang, and their final-year BEng. students, Han Chong Tan, Pua Chyuan Koh, Teck Leong Ng, Chye Yang Tan, Chiang Fong Lee, and Yeow Meng Ng for collecting the data from the open-source systems. The authors especially thank the anonymous reviewers for their insightful and helpful comments.

REFERENCES

ALBRECHT, A. J. AND GAFFNEY, J. E. JR. 1983. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Trans. Softw. Eng. SE-9*, 6 (Nov.), 639–648.

ARMOUR, P. 2002. Ten unmyths of project estimation: Reconsidering some commonly accepted project management practices. *Commun. ACM 45*, 11, 15–18.

BELSLEY, D. A., KUH, E., AND WELSCH, R. E. 2004. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley, New York, NY.

BLAHA, M. AND PREMERLANI, W. 1998. *Object-Oriented Modeling and Design for Database Applications*. Prentice-Hall, Englewood Cliffs, NJ.

BOEHM, B. W. AND FAIRLEY, R. E. 2000. Software estimation perspectives. *IEEE Softw. 17*, 6 (Nov./Dec.), 22–26.

BOEHM, B. W. ET AL. 2000. *Software Cost Estimation with COCOMO II*. Prentice-Hall, Englewood Cliffs, NJ.

BRIAND, L. C., EMAN, K. E., SURMANN, D., WIECZOREK, I., AND MAXWELL, K. D. 1999. An assessment and comparison of common software cost estimation modeling techniques. In *Proceedings of the International Conference on Software Engineering*. 313–322.

BRIAND, L. C. AND WIECZOREK, I. 2002. Resource modeling in software engineering. *Encyclopedia of Software Engineering*, J. Marciniak, Wiley, Ed. New York, NY, 1160–1196.

BURGESS, R. S. 1988. *Structured Program Design Using JSP*. Hutchension, London, U.K.

CANFORA, G., CERULO, L., AND TROIANO, L. 2004. An experience of fuzzy linear regression applied to effort estimation. In *Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering*. 57–61.

CHEN, P. P. 1976. The entity-relationship model—towards a unified view of data. *ACM Trans. Database Syst. 1*, 1, 9–36.

COSMIC. 1999. *Full Function Points—Release 2.0*. September. Engineering Management Research Lab, Montreal, P.Q., Canada.

COSTAGLIOLA, G., FERRUCCI, F., TORTORA, G., AND VITIELLO, G. 2005. Class point: An approach for the size estimation of object-oriented systems. *IEEE Trans. Softw. Eng. 31*, 1, 52–74.

FOSS, T., STENSRUD, E., KITCHENENHAM, B., AND MYRTVEIT, I. 2003. A simulation study of the model evaluation criterion MMRE. *IEEE Trans. Softw. Eng. 29*, 11, (Nov.), 985–995.

GARMUS, D. AND HERRON, D. 2000. *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison Wesley, Reading, MA.

GHEZZI, C., JAZAYERI, M., AND MANDRIOLI, D. 2003. *Fundamentals of Software Engineering*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ.

GLANTZ, S. A. AND SLINKER, B. K. 2001. *Primer of Applied Regression & Analysis of Variance*. 2nd ed. McGraw-Hill, New York, NY.

HAY, D. 2002. *Requirements Analysis: From Business Views to Architecture*. Prentice-Hall, Englewood Cliffs, NJ.

JEFFERY, D. R., LOW, G. C., AND BARNES, M. 1993. A comparison of function point counting techniques. *IEEE Trans. Softw. Eng. 19*, 5 (May), 529–532.

JEFFERY, D. R. AND WALKERDEN, F. 1999. An empirical study of analogy-based software effort estimation. *Empir. Softw. Eng. 4*, 2, 135–158.

JENSEN, R. 1983. An improved macrolevel software development resource estimation model. In *Proceedings of the 5th ISPA Conference*. 82–92.

JONES, T. C. 1997. *Applied Software Measurement*. McGraw-Hill, New York, NY.

KENNEDY, P. 2003. *A Guide to Econometrics*, 5th ed. Blackwell, London, U.K.

LAI, R. AND HUANG, S. J. 2003. A model for estimating the size of a formal communication protocol application and its implementation. *IEEE Trans. Softw. Eng. 29*, 1, (Jan.), 46–62.

LARANJEIRA, L. A. 1990. Software size estimation of object-oriented systems. *IEEE Trans. Softw. Eng. 16*, 5 (May), 510–522.

MCCLAVE, J. T. AND SINCICH, T. 2003. *Statistics*, 9th ed. Prentice-Hall, Englewood Cliffs, NJ.

MIRANDA, E. 2000. An evaluation of the paired comparisons method for software sizing. In *Proceedings of the International Conference on Software Engineering*. 597–604.

MOHAGHEGHI, P., ANDA, B., AND CONRADI, R. 2005. Effort estimation of use cases for incremental large-scale software development. In *Proceedings of the International Conference on Software Engineering*. 303–311.

MOLOKKEN, K. AND JORGENSEN, M. 2003. A review of surveys on software effort estimation. In *Proceedings of the International Symposium on Empirical Software Engineering*. 223–230.

NETER, J., KUTNER, M. H., NACHTSHEIM, C. J., AND WASSERMAN, W. 1996. *Applied Linear Regression Models*. McGraw-Hill/Irwin, New York, NY.

PUTNAM, L. AND MYERS, W. 1992. *Measures for Excellence*. Yourdon Press Computing Series. Prentice-Hall, Englewood Cliffs, NJ.

RUHE, M., JEFFERY, R., AND WIECZOREK, I. 2003. Cost estimation for Web applications. In *Proceedings of the International Conference on Software Engineering*. 285–294.

SMITH, J. 1999. The estimation of effort based on use cases, Rational Software white paper. IBM, Somers, NY.

SNOECK, M. AND DEDENE, G. 1998. Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types. *IEEE Trans. Softw. Eng. 24*, 4, 233–251.

STENSRUD, E., FOSS, T., KITCHENHAM, B., AND MYRTVEIT, I. 2002. An empirical validation of the relationship between the magnitude of relative error and project size. In *Proceedings of the IEEE Symposium on Software Metrics*. 3–12.

TAN, H. B. K. AND ZHAO, Y. 2004. ER-based software sizing for data-intensive systems. In *Proceedings of the International Conference on Conceptual Modeling*. 180–190.

TAN, H. B. K. AND ZHAO, Y. 2006. Sizing data-intensive systems from ER model. In *IEICE Trans. Inform. Syst. 89-D*, 4, 1321–1326.

TAN, H. B. K., ZHAO, Y., AND ZHANG, H. 2006. Estimating LOC for information systems from their conceptual data models. In *Proceedings of the International Conference on Software Engineering*. 321–330.

TEOREY, T. J., YANG, D., AND FRY, J. P. 1986. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Comput. Surv. 18*, 2 (June), 197–222.

Received June 2006; revised June 2007, November 2007, February 2008; accepted April 2008