Maximising data retention from the ISBSG repository

Kefu Deng and Stephen G. MacDonell
SERL, School of Computing and Mathematical Sciences, Auckland University of Technology, Private Bag 92006, Auckland 1142, New Zealand
stephen.macdonell@aut.ac.nz *(corresponding author)*

Abstract

BACKGROUND: In 1997 the International Software Benchmarking Standards Group (ISBSG) began to collect data on software projects. Since then they have provided copies of their repository to researchers and practitioners, through a sequence of releases of increasing size.
PROBLEM: Questions over the quality and completeness of the data in the repository have led some researchers to discard substantial proportions of the data in terms of observations, and to discount the use of some variables in the modelling of, among other things, software development effort. In some cases the details of the discarding of data has received little mention and minimal justification.
METHOD: We describe the process we used in attempting to maximise the amount of data retained for modelling software development effort at the project level, based on previously completed projects that had been sized using IFPUG/NESMA function point analysis (FPA) and recorded in the repository.
RESULTS: Through justified formalisation of the data set and domain-informed refinement we arrive at a final usable data set comprising 2862 (of 3024) observations across thirteen variables.
CONCLUSION: a methodical approach to the pre-processing of data can help to ensure that as much data is retained for modelling as possible. Assuming that the data does reflect one or more underlying models, such retention should increase the likelihood of robust models being developed.

Empirical software engineering, ISBSG repository, data formalisation, effort prediction, regression, FPA

1. INTRODUCTION

In order to understand and improve software engineering research and practice we must observe, measure (in a broad sense), record, analyse, interpret and act on data related to software engineering artefacts arising from software engineering activities. There has been particularly enduring interest in utilising such data for predictive modelling – that is, to use data from completed actions and artefacts to estimate the characteristics of future actions and artefacts. In fact, software engineering organisations have been encouraged (even implored) to implement and sustain software metrics programmes in order to systematise the above sequence.

Much of this work in research and practice has been directed to the prediction of software development effort, using the data collected from completed projects. The challenges of data collection and management for organisations practising software engineering are significant, however [1,2]. First, software projects tend to be long running processes – over months or years – hence completion is a relatively infrequent event. Second, some projects do not actually make it to completion, meaning that the data are either lost or not included in associated data sets. Third, project data are seen as commercially sensitive and therefore confidential, leading to reluctance to share information across organisational boundaries. Fourth, relatively few organisations devote sufficient effort to the systematic collection and organisation of project data. Finally, with fast changing technology, past project data can quickly become irrelevant in terms of future endeavours.

In order to provide software engineers with an alternative means of accessing such data several groups have positioned themselves as the keepers of repositories of data collected from multiple organisations. These repositories have been utilised extensively in research –

many studies have reported using versions of the ‘Finnish’ data set [3] and the International Software Benchmarking Standards Group repository (ISBSG) [4] (for example, see [5-7]), among others. While commercial users are normally required to buy the latter repository, it is provided free to researchers on application. As a result, the ISBSG data set has been extensively used and cited by previous research. It is at present the largest multi-organization software engineering repository available. The data is collected using questionnaires in accordance with the benchmarking group’s pre-defined terms and metrics, with records dating back to 1997. These questionnaires enable the collection of data in a section-based structure that addresses: the Submitter, the Project, the Process, the Technology, the People and their Work Effort, the Product, the Project’s Functional Size, and Project Completion. Further details can be found from the ISBSG website (isbsg.org), while some of the relevant details of the variables collected are discussed in later sections.

With respect to the ISBSG repository, however, questions over data quality and completeness have meant that much of the data potentially available has not actually been used in the analyses performed. For instance, all three primary studies reviewed by Kitchenham et al. [1] that had used the ISBSG repository had discarded data – in some instances significant proportions of the original data – if there were missing values in observations. While this step is sometimes mentioned, it is not always explained in detail – there seems to be a view that this is a necessary but relatively incidental element of data preparation. In other instances observations have been discarded if they did not have a sufficiently high ISBSG quality grade (see Section 2). (On submission, all of the information on a project is reviewed and rated in terms of data quality (A to D) by ISBSG experts. In particular, the experts look for omissions and inconsistencies in the data that might suggest that its reliability could be questioned.) This is a relatively ‘blunt’ approach to data set refinement. Furthermore, studies do not then consider the impact of such filtering on the population represented by the remaining observations. For the reader, if it is not clear what records have been discarded then it is difficult to know what the retained data actually represents.

Similarly, in spite of these software engineering repositories often comprising very large numbers of variables, the actual number of variables retained and used in the generated predictive models has generally been small. For example, the ISBSG data set contains more than 80 variables but just four were used in the final model generated in the work of Mendes et al. [8]. It is of course entirely reasonable to discard data in certain circumstances – as models get larger (in numbers of variables) they become increasingly intractable to build, and unstable to use. Furthermore, if accuracy is not significantly lower then a smaller model is normally to be preferred over a larger alternative, as in the main it would be easier to understand. However, the *process* of discarding data, as an important step in the data handling process, should be driven not just in response to missing values, or variables with lower correlation to the target variable, but also in relation to software engineering domain knowledge.

Bearing in mind the issues just described, it is contended here that the data selection/discarding process should be transparent and robust. That is, decisions taken with respect to data handling should have clearly stated and justified rationale, taking into account software engineering domain knowledge as well as indicators of statistical importance. Our specific interest here is in using such data for the early-phase prediction of full-project development effort. This paper therefore demonstrates a process through which the retention of data from a release of the ISBSG repository can be maximised with a view to supporting least-squares regression-based prediction of project effort based on previously completed projects sized using IFPUG/NESMA function point analysis (as a typical means of building prediction models).

The remainder of this paper is structured as follows. The next section outlines in general terms the use of the ISBSG repository in software engineering research, and provides a non-systematic illustration of the treatment of the repository in relation to data retention. The data analysis process undertaken for this research is reported in Section 3, comprising discussion and demonstration of the two steps of data set formalisation and refinement. The paper is then concluded in Section 4.

2. RELATED WORK

As stated above, the ISBSG repository has been used quite extensively as a data source for empirical software engineering research. A document available from the ISBSG website lists 44 past or ongoing research projects using the repository. (Note that the projects are listed up to 2007, and some known studies (including this one) are not included, so the true number is likely to be higher than 44.) A further rough indicator of the level of research activity utilising the repository can be drawn from the data returned from a query of Google Scholar – searching for “isbsg” returns 489 hits (as at March 2008), around 450 if citations are excluded. While this does also include items such as conference reports and calls for papers, a large proportion of these entries are published papers authored by those in the empirical software engineering research community.

We now consider a selection of these papers in addition to those reviewed in [1] and mentioned briefly above – but we acknowledge that this is a limited review. The intent is to provide an indicative sense of the various ways in which the repository data have been pre-processed by the research community, rather than a comprehensive discussion of all such treatments.

A study considering the influence of method and CASE tool usage on development effort using Release 8 of the ISBSG repository was reported by Cuadrado-Gallego and colleagues [9]. After starting with 2028 observations the researchers removed those will null or "invalid numerical values" for effort or size. Projects not sized using any of the FPA variants of interest were also removed. This appears to result in a data set of just under 1950 projects, although this is not stated. Clustering is then performed on the entire database, which we take to mean the 1950 projects clustered in two-dimensional space considering size (in function points) and effort. Mention is made of the removal of "a small number of outliers" but no further detail is given. Further analysis proceeds with much smaller data subsets corresponding to the use or non-use of methods and CASE tools.

Abran et al. [10] describe a comparison of approaches to building size-effort models for projects developed in a range of programming languages. They provide a useful description of the filtering required in order to utilise relevant data in their analysis. After starting with 789 records (from the 1999 release of the repository) they removed observations for small (lower effort) projects and those for which there was no data on the programming language used. They further removed observations for languages with two few samples to obtain significance, leaving a total sample of 371 records. In their modelling they used one predictor variable (having taken language into account) – size in function points. Further reduction of the data set after initial analysis involved the removal of 72 (undisclosed) outlier observations.

A lesser degree of detail regarding data filtering (perhaps due to page restrictions) can be seen in the work of Adalier et al. [11]. The study begins with the 3024 observations available in Release 9 of the repository but immediately discards observations rated B through D for quality (in contrast to the Abran et al. study [10] which appears to treat all observations in the 1999 release as valid). Observations containing missing values are also dropped, resulting in a data set of 112 records. Of the many potential predictor variables available, only a function point count, source lines of code and normalised productivity rate are utilised. Of note in terms of effort estimation (rather than model fitting) is that the latter two are available only after a project has been completed.

Gencel and Buglione [12] cite the fact that the repository contains many nominal variables "on which mathematical operations cannot be carried out directly" as a rationale for splitting the data into subsets for processing in relation to the size-effort relationship for software projects. An alternative approach would be to treat such attributes as dummy variables in a single predictive model. On the basis of two prior studies, they took two such attributes into account – application type and business area type – but subsequently dropped the latter variable along with a measure of maximum team size because the values were "missing for most of the projects in ISBSG Release 10". They also utilised the quality ratings as a filter, retaining those observations rated A, B or C.

Only projects rated A and B were used in the analysis of Release 8 of the repository reported by Xia et al. [13]. Further filters were applied in relation to FPA-sizing method, development type, effort recording and availability of all of the components of function point counting (the fifteen unadjusted function point components and fourteen general system characteristics). All other variables appear to be discarded. As a result the original collection of 2027 records is reduced to a set of 184 for further processing.

The same quality rating filter is applied by Pendharkar et al. [14] in their use of the Release 7 iteration of the repository to investigate the link between team size and software size, and development effort. Further, they removed observations for which software size, team size or work effort values were missing. This led to the original set of 1238 project records being reduced to 540 observations.

Release 10 of the repository was used as the source of an analysis of the relationship between software size and development effort reported by Jiang et al. [15]. In this case only size in IFPUG/NESMA function points and effort in total hours are utilised. No 'quality' filtering is performed. Consequently a large proportion of records are retained for modelling – 3433 of 4106.

The above studies appear to reflect a general tendency to exclude many of the categorical variables from consideration, with analyses relying solely on one or a few indicators of software size. In some cases this is entirely justified, in that the other variables are simply not relevant to the issue at hand (e.g. in [14]), but in others it seems that potential difficulties in processing may have discouraged their use. In contrast, Sentas et al. [7] provide what is in our view the most comprehensive description of preparation and use of the ISBSG repository (and other data sets). Central to their work is the pre-processing of several categorical variables in the data set, addressing aspects such as the development platform and the application type. Their intent, as is also the case here, appears to be to keep for consideration as many variables as possible. A differentiating factor is their use of ordinal regression which is more readily able to consider categorical data. They then filter the data set based on quality ratings (A and B only), FPA counting method (IFPUG), and aspects of effort recording. This led to a provisional data set of 556 observations (from 1239 (or 1238 [14]) in the release.) Their discarding of observations with missing values, however, saw more than 500 of these records removed, resulting in a data set of just 52 projects.

There is no question that missingness of data is a problem in empirical software engineering, as it is of course in other fields. "One of the problems often faced by statisticians undertaking statistical analysis in general, and multivariate analysis in particular, is the presence of missing data" [16]. It is commonly regarded as acceptable if the extent of missingness is small (for example, 5% or less) because it is normally inevitable to encounter missingness in the world of inferential statistics, which by its nature attempts to portray the characteristics of a population from those of an inherently incomplete sample. Furthermore, missingness becomes almost more inevitable (if there is such a thing!) when statistics are applied in a domain such as software engineering because the information being collected and analysed could be considered by submitters as commercially sensitive. The fact is that the majority of the observations in the ISBSG repository will contain missing values, due to the design of the data collection procedure and the nature of software engineering practice. For example, one of the data elements (or potential variables) collected in the ISBSG data set is the number of lines of software source code (LOC). If the sizing approach used for a particular project is IFPUG FPA, the LOC variable is purposely ignored. A further example is 'Organization Type' – some organisations might choose to ignore the associated data collection question for this variable in an effort to retain confidentiality (even though this is assured by the ISBSG).

A degree of missingness can be treated through the use of missing data techniques (MDTs) which attempt to impute the missing values, these methods having received some methodological attention in the empirical software engineering literature [17-19] and having been used in [8]. However, in order to be effective the degree of missingness needs to be less than a certain proportion of the potential data source – evidence to date suggests that 40% or 50% would be an upper limit for a data item in repository such as that provided by the ISBSG [17,19].

Finally we note here the work of Liebchen et al. [20] that specifically addressed the issue of data quality in software engineering data sets and the need to clean, filter and polish such data. We agree that greater attention needs to be directed to the pre-processing of data and concur with their view that such work has been limited to date.

3. DATA ANALYSIS

The particular version of the ISBSG repository used in this analysis is Release 9, made available between 2004 and 2007. It includes data on 3024 projects.

The data analysis process described here is composed of two major steps:

- Data Set Formalisation: Examination of the data reveals that it is neither appropriate nor sensible to perform ordinary least-squares (OLS) regression against the raw ISBSG repository as a whole. As a result, the raw ISBSG data set must be formalised to be of reasonable size and content. The rules and the rationale behind the data set formalisation are therefore explained.
- Data Set Further Refinement: Even when a formalised and 'full' data set is acquired, it is still not entirely appropriate to perform OLS regression analysis against it because of the intended objective of the prediction, the computational complexity of regression analysis and the expensiveness of acquiring project data for individual organizations. The methodology, rules and rationale of the data set further refinement process are explained in the second subsection.

3.1 Data Set Formalisation

Despite the fact that OLS regression is reasonably robust, the characteristics of the ISBSG repository mean that it is not feasible to perform an OLS regression analysis against the raw data set. Nor is this even advisable without some consideration of the need for pre-processing: cleaning, filtering and polishing the data [20], in light of the objectives of the particular analysis being undertaken, should always precede the analysis itself.

There are a number of issues related to the raw ISBSG data that require consideration and, in some cases, action:

- *Some variables are de-normalized.* Some of the variables in the raw ISBSG data set are simply descriptive strings rather than pre-defined categories. For example, for the variable 'Project Activity Scope', one observation reads: "Planning;Specification;Build;Test;Implement;" and another observation reads: "Specification;Build;Test;Implement;". This makes the number of distinct levels of the categorical variable very large and it is neither practical nor sensible to perform regression analysis against it. To make it (and other similar variables) usable in an OLS regression analysis, separate dummy variables have to be created with different levels of 'Project Activity Scope'.

- *Some variables are not recorded in a consistent format.* For example, in the variable 'Implementation Date', different formats exist for different observations, for instance "9-Nov-00" and "prior to Feb-2004". This makes it impossible for statistical software to recognize 'Implementation Date' as either a continuous or discrete variable.

- *Some variables have too many distinct levels.* The variable 'Application Type' comprises 105 different and distinct levels. If this categorical variable were to be included in a predictive regression model as is, 104 dummy variables would have to be included in the final equation.

- *Some variables are a mixture of different contexts.* For example, 'Development Techniques' in the raw ISBSG data set contains values such as "Waterfall" and "Object-oriented design". "Waterfall" is a generic development process model description whereas "Object-oriented design" is one of the activities undertaken if a project is implemented using object-oriented methods. Another example is the 'Intended Market' variable whose values inconsistently describe either the location of the development (whether developed in-house or externally) or the actual intended market (whether developed for an internal or external business unit).

- *Some variables are aggregated from other variables.* For example, the 'Adjusted Function Points' is calculated from variables such as 'Input Count', 'Output Count', and 'File Count'. Including both basic *and* aggregated variables contravenes the desire to have only necessary and orthogonal factors in a model.

- *Some variables are irrelevant or unavailable for predictive modelling.* Almost all of the observations in the repository were submitted after the relevant project was completed. As a result, some variables that are not known at the initialization/specification stage of projects are also included; for example, indicators of software quality ('Major Defects', 'Minor Defects'), 'Project Inactive Time' and some productivity variables.

- *Some numerical variables have too few values.* For example, 'Maximum Team Size' has 1918 records out of 3024 missing. In this case, even the best MDTs cannot be applied because this is far below the minimum proportion of non-missing value requirement for MDTs.

A straightforward solution to these issues would be to drop the variables affected. In doing so, however, we may be throwing away predictive capability. Instead, some elements of the raw data set can be formalised through a range of variable transformations. Table 1 lays out the rationale for the formalisation of each variable in the raw ISBSG data set, with the variables shown in shaded boxes in column two being retained for further consideration. (Detailed descriptions of the variables can be found at the ISBSG website isbsg.org) Note that this represents a sample treatment of the data – a different formalisation may be applied if the research objective is different to the one of interest here (that is, estimation of project-level development effort). However, the underlying goal of data retention and adherence to the principle of transparent pre-processing still hold.

TABLE 1: Formalisation rationale explained for variables in the raw ISBSG data set

Raw ISBSG Variable	**Destination Variable in Formalised Set**	**Rationale for the transformation**
Project ID	Project ID	No change – label.
Data Quality Rating	Data Quality Rating	No change – could be used for later filtering.
UFP rating	N/A	Data quality indicator. Not directly related to software effort estimation.
Count Approach	Count Approach	No change – will be used for later filtering.
Functional Size	N/A	Only 'Adjusted Function Points' is used; this variable is a component of 'Adjusted Function Points'.
Adjusted Function Points	Adjusted Function Points	No change.
Value Adjustment Factor	N/A	Only 'Adjusted Function Points' is used; this variable is a component of 'Adjusted Function Points'.
Summary Work Effort	Summary Work Effort	No change, **the response variable.**
Normalised Work Effort	N/A	Only 'Summary Work Effort' is used; while potentially useful, the extrapolation performed to produce this variable is arbitrarily applied and relies on 'Project Activity Scope' being recorded, and recorded accurately. Median difference between 'Summary Work Effort' and 'Normalised Work Effort' is zero.
Reported PDR (afp)	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Project PDR (ufp)	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Normalised PDR (afp)	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Normalised PDR (ufp)	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Project Elapsed Time	N/A	Unrelated to software effort estimation at the

Raw ISBSG Variable	Destination Variable in Formalised Set	Rationale for the transformation
		initialization phase of a project.
Project Inactive Time	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Implementation Date	'Implementation Start Year' and 'Implementation Start Year Range'	Potentially unrelated to software effort estimation at the initialization phase of a project. The raw data set is not in constant format. Furthermore, about 20% of the data are missing. Review data and form two variables (details below).
Project Activity Scope	N/A	De-normalised, providing too much information in one variable. Moreover, this data is primarily useful in terms of determining 'Normalised Work Effort', a variable that is not to be retained in this case.
Effort Plan	N/A	Only 'Summary Work Effort' is used; furthermore (i) this can be expensive to get for software organisations and (ii) there could be some question over its accuracy.
Effort Specify	N/A	Same as previous.
Effort Design	N/A	Same as previous.
Effort Build	N/A	Same as previous.
Effort Test	N/A	Same as previous.
Effort Implement	N/A	Same as previous.
Effort Unphased	N/A	Same as previous.
Minor defects	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Major defects	N/A	Same as previous.
Extreme defects	N/A	Same as previous.
Total Defects Delivered	N/A	Same as previous.
Development Type	Development Type	No change.
Organisation Type	N/A	Too many (more than 100) distinct levels. Mixed in context with 'Application Type' and 'Business Area Type'. This could involve some level of arbitrariness.
Business Area Type	Business Area Type	Re-categorize missing values to "Unspecified" and rename the labels according to standard sector descriptors (details below).
Application Type	N/A	Same issues as for 'Organisation Type', therefore it is omitted
Package Customisation	Package Customisation	Formalise "Unknown" and missing values in the raw data to "Unspecified" (details below).
Degree of Customisation	N/A	Too many missing values (2937 out of 3024, 97%)

Raw ISBSG Variable	Destination Variable in Formalised Set	Rationale for the transformation
		and too many distinct levels (19).
Architecture	Architecture	Re-categorize missing values to "Unspecified" and rename the levels according to current mainstream standards (details below).
Client Server?	N/A	Too many missing values (1346 out of 3024, 45%). Considered in 'Architecture'.
Client roles	N/A	Too many missing values (2968 out of 3024, 98%).
Server roles	N/A	Too many missing values (2970 out of 3024, 98%).
Type of server	N/A	Too many missing values (2821 out of 3024, 93%).
Client/server description	N/A	Too many missing values (2291 out of 3024, 76%). Considered in 'Architecture' and 'Web development'
Web development	Is Web	Re-categorize missing values to "Unspecified" (details below)
Plan documents	N/A	Too many missing values (2896 out of 3024, 96%).
Specify documents	N/A	Too many missing values (2897 out of 3024, 96%).
Specify techniques	N/A	Too many missing values (2961 out of 3024, 98%).
Design documents	N/A	Too many missing values (2907 out of 3024, 96%).
Design techniques	N/A	Too many missing values (2970 out of 3024, 98%).
Build products	N/A	Too many missing values (2895 out of 3024, 96%).
Build activity	N/A	Too many missing values (2951 out of 3024, 98%).
Test documents	N/A	Too many missing values (2896 out of 3024, 96%).
Test activity	N/A	Too many missing values (2984 out of 3024, 99%).
Implement documents	N/A	Too many missing values (2909 out of 3024, 96%).
Implement activity	N/A	Too many missing values (2957 out of 3024, 98%).
Development Techniques	'Main Development Process Model' and 'Object Orientation'	Too many different contexts are explained in this one variable. In this case, two variables are extracted from the raw variable with formalised values (details below).
Functional Sizing Technique	N/A	Unrelated to software effort estimation at the initialization phase of a project.
FP Standard	N/A	Same as previous.
FP Standards all	N/A	Same as previous.
Reference Table Approach	N/A	Same as previous.
Development Platform	Development Platform	Re-categorize missing values to "Unspecified" (details below).
Language Type	Main Language Type	Re-categorize missing values to "Unspecified"

Raw ISBSG Variable	Destination Variable in Formalised Set	Rationale for the transformation
		(details below).
Primary Programming Language	N/A	Mixture of contexts. For example some values are "3GL" or "4GL" which is language type and some others are "IIS" which is the name of a type of web server. Some values have version numbers while some others do not. In such a case, it is very hard to get a sensible category from this variable. Furthermore, given the 'Language Type' the specific programming language cannot provide too much extra information.
1st Hardware	N/A	Mixture of contexts. For example: "Client/Server", "Unix" … which are not descriptions of hardware.
1st Operating System	Main Operating System	Distinct values in raw dataset to be extracted and re-categorized. Levels are minimized according to the mainstream operating systems (details below).
1st Language	N/A	Same as for 'Primary Programming Language'.
1st Data Base System	Main Database System	Same rationale and process as for 'Main Operating System' (details below).
1st Component Server	N/A	Too many missing values (3001 out of 3024, 99%).
1st Web Server	N/A	Too many missing values (3003 out of 3024, 99%).
1st Message Server	N/A	Too many missing values (3017 out of 3024, 100%).
1st Debugging tool	N/A	Too many missing values (2786 out of 3024, 92%).
1st Other Platform	N/A	Too many missing values (2189 out of 3024, 72%).
2nd Hardware	N/A	Too many missing values (3024 out of 3024, 100%).
2nd Operating System	N/A	Too many missing values (2995 out of 3024, 99%).
2nd Language	N/A	Too many missing values (2963 out of 3024, 98%).
2nd Data Base System	N/A	Too many missing values (3009 out of 3024, 100%).
2nd Component Server	N/A	Too many missing values (3022 out of 3024, 100%).
2nd Web Server	N/A	Too many missing values (3023 out of 3024, 100%).
2nd Message Server	N/A	Too many missing values (3022 out of 3024, 100%).
2nd Other Platform	N/A	Too many missing values (2993 out of 3024, 99%).
CASE Tool Used	Case Tool Used	Re-categorize missing values to "Unspecified" (details below).
Used Methodology	N/A	Unrelated to software effort estimation at the initialization phase of a project.
How Methodology	N/A	Same as previous.
Acquired		
User Base - Business Units	N/A	Too many missing values (2434 out of 3024, 80%).
User Base - Locations	N/A	Too many missing values (2384 out of 3024, 79%).
User Base - Concurrent Users	N/A	Too many missing values (2408 out of 3024, 80%).
Intended Market	'Developed Inhouse' and 'Intended Market'	The 'Intended Market' in the raw ISBSG dataset explains two aspects of the software development process. One is the location where the project is developed. Another is the actual intended market. Here the variable is separated (details below).
Recording Method	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Resource Level	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Max Team Size	N/A	Too many missing values (1836 out of 3024, 61%).
Average Team Size	N/A	Too many missing values (1918 out of 3024, 63%).
Ratio of Project Effort: non-project Effort	N/A	Unrelated to software effort estimation at the initialization phase of a project.
% of uncollected Work Effort	N/A	Unrelated to software effort estimation at the initialization phase of a project.
Input count	N/A	An element of the 'Adjusted Function Points' already included. This variable does not provide additional value given the focus of most studies on project-level estimation.
Output count	N/A	Same as previous.
Enquiry count	N/A	Same as previous.
File count	N/A	Same as previous.
Interface count	N/A	Same as previous.
Added count	N/A	Same as previous.
Changed count	N/A	Same as previous.
Deleted count	N/A	Same as previous.
Lines of code	N/A	Lines of code approach, an alternative sizing method, is not included in the research target. Unrelated to software effort estimation at the initialization phase of a project.
LOC not Statements	N/A	Same as previous.

Table 2 lists the variables drawn from the raw ISBSG data set that comprise the 'full' scope from which further variable selection could be reasonably undertaken. For each of the (reordered) variables, the type of the variable (continuous, ordinal or categorical) and the distinct levels of the categorical variables are listed. The extraction of these variables is informed by the information available in the ISBSG data set and ISBSG supplied demographics as well as by prior studies and the intended analyses in terms of project-level prediction.

Once the set of candidate variables has been selected, a rule is defined in order to actually perform the data set formalisation. Before establishing the rules, each distinct value for each of the retained variables in the raw ISBSG data set is carefully examined in order to minimize confusion of concepts and to maximize both consistency and numbers of responses for each level. During the formalisation process, missing or ambiguous categorical values are treated as "Unspecified". As the whole process of defining the formalisation rules is manual, some of the definitions and rules could be considered to be arbitrary. However, significant effort has been expended in an effort to ensure that each decision is defendable and each rule applied consistently. Furthermore, it is contended here that retaining data, even if achieved using potentially arbitrary rules, is to be preferred over the similarly arbitrary dropping of those significant numbers of observations that have missing values. Such a decision would see a very large proportion of the data dropped – observations without missing values in the estimation-related variables account for just 20% of the data. The formalisation rules and their rationale are shown in Table 3.

Note 1: Due to constraints on space it is not possible to replicate here the table that details the formalisation of 'Business Area Type', which shows the transformation of nearly 100 non-standard descriptions to standard business sector labels. The following examples should illustrate the process and outcomes, however: Telecommunications, Computer Systems and IT, Network card administration were all formalised as "IT & T"; and Product Distribution, TRANSPORT and Transport/Shipping were all formalised as "Logistics".

The two tables following (4 and 5) define the specific formalisation rules for raw variables 'Intended Market' and 'Architecture' based on the principles described in Table 3. Note that instances of 'Architecture' with the value "Multi-tier/Client server" are coded to the more complex "Multi Tier" value in the formalised version of the data set, to ensure that the *potential* complexity is accounted for (as it is considered preferable to be conservative).

At the conclusion of the formalisation process the data set comprises one label variable (Project ID), one response variable (Summary Work Effort) and 16 potential predictor variables, with 2862 complete records (i.e. records for IFPUG/NESMA FPA-sized projects, with no missing values).

TABLE 2: Extracted candidate variables from the ISBSG data set as the 'full' data set

Candidate Variable	**Variable Type**	**Notes**
Project ID	Nominal	The unique identifier of the project being described.
Adjusted Function Points	Continuous	Data is complete. The function point value of each project must be included for all projects using this as a counting approach.
Development Type	Categorical	Data is complete, all levels for this variable are provided.
Business Area Type	Categorical	Levels: Health, Insurance, Banking, IT & T, Manufacturing, Accounting, Transport, Government, Sales and Marketing, Other, Unspecified
Package Customisation	Categorical	Levels: Yes, No, Unspecified
Developed Inhouse	Categorical	Levels: Yes, No, Partly, Unspecified
Intended Market	Categorical	Levels: Internal, External, Both, Unspecified
Implementation Start Year	Ordinal	But no validity in its use as a numeric predictor.
Implementation Start Year Range	Categorical	Levels: 1989-1994, 1995-1999, 2000-2004, Unspecified.
Architecture	Categorical	Levels: Stand Alone, Multi Tier, Client Server, Unspecified
Is Web	Categorical	Levels: Yes, Unspecified
Development Platform	Categorical	Levels: PC, Mid-range, Main-frame, Multi-platform, Unspecified
Main Language Type	Categorical	Levels: 2GL, 3GL, 4GL, 5GL, APG, Unspecified
Main Database System	Categorical	Levels: Oracle, DB2, SQL Server, Other, Unspecified
Main Operating System	Categorical	Levels: Mainframe, DOS, Windows, Solaris, Unix, Other, Unspecified
CASE Tool Used	Categorical	Levels: Yes, No, Don't know, Unspecified
Main Development Process Model	Categorical	Levels: Waterfall, Iterative, Other, Unspecified
Object Orientation	Categorical	Levels: Yes, Unspecified
Summary Work Effort	Continuous	The response, with the unit 'person hours'.

TABLE 3: Formalisation rules applied to the raw ISBSG data set

Formalised Variable	**Formalisation Rule**	**Rationale of the Formalisation**
Adjusted Function Points	No change	No change needed as the variable is compulsorily required by ISBSG for FPA-based records.
Development Type	No change	No change needed as the variable is complete and has only 4 distinct levels.
Business Area Type	See Note 1 below	This variable is a mixture of organisation type, application type and business area type. Detailed formalisation

Formalised Variable	Formalisation Rule	Rationale of the Formalisation
		rules are utilised, explained in brief below to form 11 levels.
Package Customisation	Same as 'Package Customisation'. If the value is "Don't know" or null then "Unspecified"	To make it more appropriate for regression analysis by categorising empty data into a value called "Unspecified".
Developed Inhouse	See Table 4 'Intended Market'	The 'Intended Market' variable addresses two aspects of the software development process. One is the type of the physical location in which the project is developed, the other is the actual intended market. This variable specifies "type of the physical location where the project is developed".
Intended Market	See Table 4 'Intended Market'	This variable addresses the actual intended market as indicated from the original variable 'Intended Market' by stripping off the information regarding 'Developed Inhouse' (Please see previous row for more information.)
Implementation Start Year	Manually convert 'Implementation Date' to the specified year e.g. "24/03/1999" to 1999	Manually extract the year because values need to conform to a constant format. (Used only to determine next variable.)
Implementation Start Year Range	Convert 'Implementation Start Year' to the appropriate year range: 1989-1994, 1995-1999, 2000-2004	There are more than 500 missing values out of 2800 observations in 'Implementation Start Year'. Furthermore, while it might be useful in terms of time series analysis it is not sensible to use start year as a numeric predictor variable. Given that the variable may have potential worth a categorical version can be included.
Architecture	See Table 5 'Architecture'	
Is Web	If 'Web Development' is "Web" then "Yes" else "Unspecified"	The raw data set only contains "Yes" and null values for this variable. If the value is null, no assumption can be made about the project, thus "Unspecified" is used in its place.
Development Platform	Same as 'Development Platform'. If the value is null then "Unspecified"	
Main Language Type	Same as 'Language Type'. If the value is null then "Unspecified"	

Formalised Variable	Formalisation Rule	Rationale of the Formalisation

Main Database System	If 'First Database System' contains "Oracle" then "Oracle"
	else If 'First Database System' contains "DB2" then "DB2"
	else If 'First Database System' contains "SQL Server" OR "SQL-Server" OR "MS SQL" OR "MSDE" then "MS SQL"
	else If 'First Database System' is null then "Unspecified"
	else "Other"
		Same reasoning as for 'Main Operating System'.

Main Operating System	If 'First Operating System' contains "Windows" OR "win" OR ".net" OR "SQL-server" OR "NT Server" OR "NT" OR "XP" then "Windows"
	else If 'First Operating System' contains "Mainframe" then "Mainframe"
	else If 'First Operating System' contains "DOS" then "DOS"
	else If 'First Operating System' contains "Solaris" then "Solaris"
	else If 'First Operating System' contains "Unix" then "Unix"
	else if 'First Operating System' is null OR contains "client/server" OR "custom" OR "not assessed" OR "not recorded" then "Unspecified"
	else "Other"
		Distinct values in the raw data set are extracted and re-categorised. The levels are minimized according to the mainstream software development operating systems.

CASE Tool Used	Same as 'Case Tool Used'. If the value is null then "Unspecified"	
Main Development Process Model	If 'Development Techniques' contains "Waterfall" then "Waterfall" else If 'Development Techniques' contains "RAD" OR "Rapid Application Development" OR "Prototype" then	'Development Techniques' in the raw data set is a mixture of 'Main Development Process Model' and 'Object Orientation' which are two entirely different kinds of context with different criteria. Some of the values are actually explaining the

Formalised Variable	Formalisation Rule	Rationale of the Formalisation
	"Iterative" else If 'Development Techniques' is null then "Unspecified" else "Other"	detailed steps/activities in software development processes rather than the development process on its own.
Object Orientation	If 'Development Techniques' contains "Object oriented" OR "Object-oriented" OR "OO" then "Yes" else "Unspecified"	
Summary Work Effort	Same as 'Summary Work Effort'	

TABLE 4: Formalisation rules applied to 'Intended Market' in the raw ISBSG data set

Raw ISBSG Values(Intended Market)	Formalised ISBSG Values (Developed Inhouse)	Formalised ISBSG Values (Intended Market)
Outsourced for internal business unit	No	Internal
Customer & users 1 org, team in another	No	External
Customer, users, team in different orgs	No	External
In-house for internal business unit	Yes	Internal
Partly outsourced and partly inhouse	Partly	Partly
Customer, users & team in same org	Yes	Internal
In-house for all internal business units	Yes	Internal
Customer & team 1 org, users in another	Yes	External
In-house for external business unit	Yes	External
In-house for internal business unit; In-house for external business unit	Yes	Both
Dev in-house for use by ext agent req to rept to us	Yes	External
External for external business unit	No	External
Inhouse for bank customers	Yes	Internal
NULL (value is missing)	Unspecified	Unspecified

TABLE 5: Formalisation rules applied to 'Architecture' in the raw ISBSG data set

Raw ISBSG Values	Formalised ISBSG Values
Multi-tier / Client server	Multi Tier
Multi-tier	Multi Tier
Multi-tier with web public interface	Multi Tier
Stand alone	Stand Alone
Client server	Client Server
NULL (value is missing)	Unspecified

3.2 Further Refinement of the Data Set

To this point, all the appropriate and *potentially* useful variables and observations in terms of project-level software effort estimation have been pre-processed from the raw ISBSG data. Some values have been modified with justification in order to produce sound categorical variables for regression analysis and/or to deal with missing values. The result is a formalised and 'full' data set created from the raw ISBSG data repository. The following issues then need to be considered:

- Taking into account the principles and conventions of software engineering, it is not meaningful to include some of the available variables in an effort estimation model. Weisberg [21] argued that "the single most important tool in selecting a subset of variables for use in a model is the analyst's knowledge of the substantive area under study." He then criticised the action of including all variables in multiple regression models as "throwing everything in the hopper" simply because they are available.

- It is also difficult to make statistical inference from an overly-complicated regression model because it becomes difficult to explain and anticipate the impact of the overall model given certain input conditions. As a result the model may be problematic to utilize in a production environment. It is also difficult to explain the relationship between the response and the many independent variables, given that there may be interaction effects among the independent variables.

- The calculation of a regression model can be computationally expensive. To illustrate, if all 16 predictor variables with all the interactions were to be included in a model, the number of potential components in the final model equation could be (15! + 1). Formulating such a model over the potentially large number of observations (in this case a data comprising more than 2800 observations) would challenge the processing limitations of current desktop PCs.

While there is no definitive suggestion as to the maximum number of candidate variables that should exist in a full data set, there is an accepted trade off between accuracy and parsimony. Finding an optimum model should be informed by software engineering principles and relevant personal experience. With this in mind, all the variables retained so far are considered to decide whether they should be kept in the full data set for further study. Two principles inform the decision to keep or drop a variable at this point:

1. A variable should be dropped if too great a degree of effort has to be expended in order to decide the value of it in the process of software/systems development, given that estimates of effort are often first needed in the very early stages of development

2. A variable should be dropped if there is no conceptual justification for its contribution to a predictive model of software development effort.

In light of the above Table 6 describes the relevant 'Keep/Drop' decisions and the rationale for each, for the 16 potential predictors.

TABLE 6: Final decisions regarding retention of potential predictor variables

Variable	**Action**	**Explanation**
Adjusted Function Points	Keep	Indicator of project scale. Available quite early, prior evidence of relationship with effort.
Development Type	Keep	Indicator of project type. Available early, prior evidence of relationship with effort.
Business Area Type	Keep	Indicator of project domain. Available early, prior evidence of relationship with effort.
Package Customization	Keep	Indicator of project type. Available early, possibly related to effort.
Developed Inhouse	Keep	Indicator of project structure. Available early, possibly related to effort.
Intended Market	Keep	Indicator of project structure. Available early, possibly related to effort.
Implementation Start Year Range	Keep	Indicator of project context. Can be estimated early, possibly related to effort.
Architecture	Drop	In reality, software developers can expend substantial effort in order to reach a decision as to which architecture to use, by investigating the solution domain and the availability of current technology. Therefore, at the time when effort estimates are first needed, decision makers may not have decided on the architecture to use.
Is Web	Drop	The levels of this variable are only "Yes" and "Unspecified". In reality, a project could be a combination of web and other types depending on the chosen architecture.
Development Platform	Keep	Indicator of project technology. Available early, possibly related to effort.
Main Language Type	Keep	Indicator of project technology. Available quite early, possibly related to effort.
Main Operating System	Keep	Indicator of project technology. Available quite early, possibly related to effort.
Main Database System	Drop	To make the decision as to which database system to use, a significant amount of effort would normally be expended. For example, comparing the performance capabilities, benchmarking and proof-of-concept documentation. In reality, organisations tend to favour one or more particular DB systems (as per Architecture) but even this varies over time and (for bespoke systems) depends on customer needs.
CASE Tool Used	Keep	Indicator of project technology. Available quite early, possibly related to effort.
Main Development Process Model	Keep	Indicator of project process. Available early, possibly related to effort.
Object Orientation	Drop	This decision would normally be made at the design phase.

This step represents the end of the data set refinement process. At this point there exists a refined data set that is usable in terms of OLS regression analysis, comprising one continuous response variable and twelve predictor variables (one continuous and eleven categorical). In our processing of Release 9 of the ISBSG repository this resulted in the provision of a complete data set still including 2862 observations, comprising 673 rated A quality, 2006 rated B, 106 rated C and 77 rated D.

4. CONCLUSIONS

We believe that there is a need for greater clarity in describing and justifying the pre-processing, discarding and retention of data from software engineering data sets. In this paper we have illustrated how such clarity can be achieved through an example, filtering, formalising and refining the data in Release 9 of the ISBSG repository in line with an intent to build a predictive model of project-level development effort for FPA-sized projects.

Note that the above outcomes in terms of the variables retained or discarded are not intended to be conveyed as 'correct'. Rather, it is intended to be indicative of the outcomes that might be achieved given a particular research objective while keeping in mind the need to be transparent and to retain as much data as possible. Of course there may well be a need for further processing of the above data in order to build a specific predictive model. For instance, researcher or practitioner interest may be in enhancement-type projects – in that case only those projects with a value of "Enhancement" for 'Development Type' would likely be considered. Even if that were the case, however, the above process would have ensured that as many of these relevant observations as possible were available for such an analysis, along with a range of potentially influential variables. Assuming that the retained data does indeed reflect one or more underlying models, such outcomes should increase the likelihood of robust models being produced.

ACKNOWLEDGEMENT

We thank the ISBSG for providing the data that enabled this work to be carried out.

6. REFERENCES

[1] Kitchenham, B., E. Mendes and G.H. Travassos, "A systematic review of cross- vs. within-company cost estimation studies", *Proc. 10th Intl Conf Empirical Assessment in Soft Eng* 2006.

[2] MacDonell, S.G., and M.J. Shepperd, "Comparing local and global software effort estimation models – reflections on a systematic review", *Proc. 1st Intl Symp Empirical Soft Eng & Measmt* 2007, 401-409.

[3] Experience Pro - Software Technology Transfer Finland, http://www.sttf.fi/eng/indexEnglish.htm

[4] International Software Benchmarking Standards Group, http://www.isbsg.org

[5] Briand, L.C., K. El Emam, D. Surmann, I. Wieczorek and K. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques", *Proc. 21st Intl Conf Soft Eng* 1999, 313-322.

[6] Jeffery, R., M. Ruhe and I. Wieczorek, "Using public domain metrics to estimate software development effort", *Proc. 7th Intl Soft Metrics Symp* 2001, 16-27.

[7] Sentas, P., L. Angelis, I. Stamelos and G. Bleris, "Software productivity and effort prediction with ordinal regression", *Information and Software Technology* 47, 2005, 17-29.

[8] Mendes, E., C. Lokan, R. Harrison and C. Triggs, "A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database", *Proc. 7th Intl Soft Metrics Symp* 2005.

[9] Cuadrado-Gallego, J.J., M.-A. Sicilia, M. Garre and D. Rodriguez, "An empirical study of process-related attributes in segmented software cost-estimation relationships", *Journal of Systems & Software* 79, 2006, 353-361.

[10] Abran, A., I. Ndiaye and P. Bourque, "Evaluation of a black-box estimation tool: a case study", *Software Process Improvement and Practice* 12, 2007, 199-218.

[11] Adalier, O., A. Uğur, S. Korukoğlu and K. Ertaş, "A new regression based software cost estimation model using power values", IDEAL 2007, *Lecture Notes in Computer Science* 4881, 2007, 326-334.

[12] Gencel, C., and L. Buglione, "Do different functionality types affect the relationship between software functional size and effort?", *Proc. 2007 Intl Workshop Soft Measmt*, 2007, 235-246.

[13] Xia, W., D. Ho and L.F. Capretz, "Calibrating function points using neuro-fuzzy technique", *Proc. 21st Intl Forum COCOMO and Softw Cost Modeling*, 2006.

[14] Pendharkar, P.C., J.A. Rodger and G.H. Subramanian, "An empirical study of the Cobb–Douglas production function properties of software development effort", *Information and Software Technology*, In Press.

[15] Jiang, Z., P. Naudé and B. Jiang, "The effects of software size on development effort and software quality", *Proc. World Academy Sci, Eng and Tech* 23 2007, 363-367.

[16] Everitt, B.S., and G. Dunn, *Applied Multivariate Data Analysis*, Second Edition, 2001.

[17] Twala, B., M. Cartwright and M. Shepperd, "Comparison of various methods for handling incomplete data in software engineering databases", *Proc. Intl Symp Empirical Soft Eng* 2005, 105-114.

[18] Jonsson, P., and C. Wohlin, "An evaluation of *k*-nearest neighbour imputation using Likert data", *Proc. 10th Intl Soft Metrics Symp* 2004, 108-118

[19] Li, J., A. Al-Emram and G. Ruhe, "Impact analysis of missing values on the prediction accuracy of analogy based software effort estimation method AQUA", *Proc. 1st Intl Symp Empirical Soft Eng & Measmt* 2007, 126-135.

[20] Liebchen, G., B. Twala, M. Shepperd, M. Cartwright and M. Stephens, "Filtering, robust filtering, polishing: techniques for addressing quality in software data", *Proc. 1st Intl Symp Empirical Soft Eng & Measmt* 2007, 99-106.

[21] Weisberg, S., *Applied Linear Regression*, John Wiley & Sons. New York, 1985.