Lessons Learned from the ISBSGs Database

1.0 Introduction

The International Software Benchmark Standards Group (ISBSG) was formed in 1997 with a mission “To improve the management of IT resources by both business and government, through the provision and exploitation of public repositories of software engineering knowledge that are standardized, verified, recent and representative of current technologies.”[1] They are a not-for-profit organization that has built, grows, maintains and exploits two repositories of historical IT data to help improve the management of IT projects. One of the repositories is focused on Software Development and Enhancement Projects and contains close to 6000 projects; the other is a newer repository focused on Software Maintenance and Support and contains over 500 projects. The ISBSGs is regarded as one, if not the, largest global and independent source of data and analysis for the IT industry. ISBSG partners represent the following countries: Australia, China, Finland, Germany, India, Italy, Japan, Netherlands, Spain, Switzerland and the USA. The ISBSG database represents data from many industry sectors including Banking, Communications, Defense, Manufacturing, Government, Financial, Wholesale and Retail. If you would like to learn more about ISBSG visit their website www.isbsg.org

While the ISBSGs method of software sizing uses a Functional Size measure, the database contains projects measured using IFPUG Function Points, Mark II Function Points, COSMIC Function Points, NESMA Function Points and several other less popular functional size measures. There is a place to submit Source Lines of Code information as well but no validation is done on this part of the submission. Submitters use a Data Collection Questionnaire to provide the ISBSG with information about their projects. There are over 100 questions asked about the application, industry, size, effort, schedule, quality, architecture, documentation, techniques, and other project information. Data is reviewed, quality checked and normalized before being included in the database. Some of the data in the database comes directly from the submission and some is derived based on the questionnaire responses.

PRICE Systems has recently partnered with ISBSG with licenses to both the data repositories. Although we cannot distribute the data to those without subscriptions, there is no reason we can’t use analysis of this data to provide guidance to the cost estimation community, specifically with respect to how this data can be used to fine tune or calibrate cost estimation models. One effort focused on developing calibrated software estimation templates for TruePlanning for Software® based on various scenarios across industry sector, application type, development type (new or enhancement) and language type (3GL,4G). While the effort was accomplished using a specific cost estimation tool, the methodology presented in this paper can be applied to any commercial or home grown software estimation model. The paper begins with an overview of the methodology. It then describes the categories of cost drivers that can be gleaned from the ISBSG data points that may be relevant for estimation purposes. Following this is a discussion of the criteria for setting specific scenarios for evaluation. Finally the process of calibration is described and the results are presented.

2.0 Overview of the methodology

The goal of this particular project was to identify specific scenarios from the ISBGS database and create templates for cost estimation model inputs that can be used to support cost/effort estimates for new projects similar to the one described by a particular scenario. An example of a specific scenario would be a new development project for the communications industry developing an Information System using a fourth generation development environment. The first step of the methodology involved applying a set of filters to the ISBSG data in order to eliminate data points with suspect quality or inadequate information for effort estimation analysis. Then a set of filters was applied iteratively to identify scenarios for which an adequate number of data points exist to support reasonable analysis. The data was then translated from the ISBSG format into applicable cost model inputs and calibration of the estimation model was performed to use actual effort to determine numeric values of model inputs not directly stipulated in the ISBSG data base. From these input values templates were created for each of the scenarios identified.

3.0 Typical Software Estimation Model Cost Drivers

The following categories of cost drivers are important to software estimation. The actual analysis and calibration for this project was accomplished using TruePlanning for Software®. A brief description of this tool along with information about the specific inputs involved in this analysis can be found in Appendix A.

Type of Application – Different types of applications require that the code that is developed be more or less complicated. For example basic data processes functions are less complex, thus easier and cheaper to develop, than real time software developed for a weapons system. Most cost models have one or more inputs that quantify this code complexity.

Operating Platform - This value describes the intended operating environment and defines the degree of reliability, portability, structuring, documentation, and testing that the customer requires. Most cost models have an input that indicates whether the software is intended to operate on the ground, in the air or on the water.

Code Size – Code size is generally specified in several categories – new, reused, modified, deleted, auto generated, etc. This input or set of inputs is intended to inform the model of how much code is being developed and how much code is being integrated to deliver the final software product. This input category may also represent percentages to indicate the extend of modifications

Team Experience – Another important cost driver for most software estimation models requires some quantification of the development teams experience in the industry, with the particular project being developed and with the development tools being employed.

Calibration Constant – This particular input classification is more relevant in commercially available general purpose estimation models than in ones that are developed by a specific organization. In the cases where a general purpose model is being used, it is important for an organization to be able to fine tune it to accurately represent the practices, policies and history of the specific organization. This isn't intended to model development team productivity but rather the proficiency and efficiency with which an organization can deliver software to the marketplace. Home grown models specific to an organization have this built in, but general purpose models really need this type of control.

Language – this value indicates what the primary programming language for the component is.

4.0 Methodology for creating template scenarios

With close to 6000 data points, each with up to 120 different characteristics or metrics, the first step was to determine appropriate filters [2]. The analysis first zeroed in on the ISBSG determined quality rating. ISBSG assigns a quality value from A through D where A and B represent data items that appear to be of high integrity and C and D represent data items whose integrity is questionable. This analysis focused only on the A and B rated data items. A decision was made to limit analysis to those projects that performed Function Points Counts using either the IFPUG or NESMA methodologies because this provided the maximum number of potential data points. NESMA Function Points were included because of their very close alignment to IFPUG Function Points

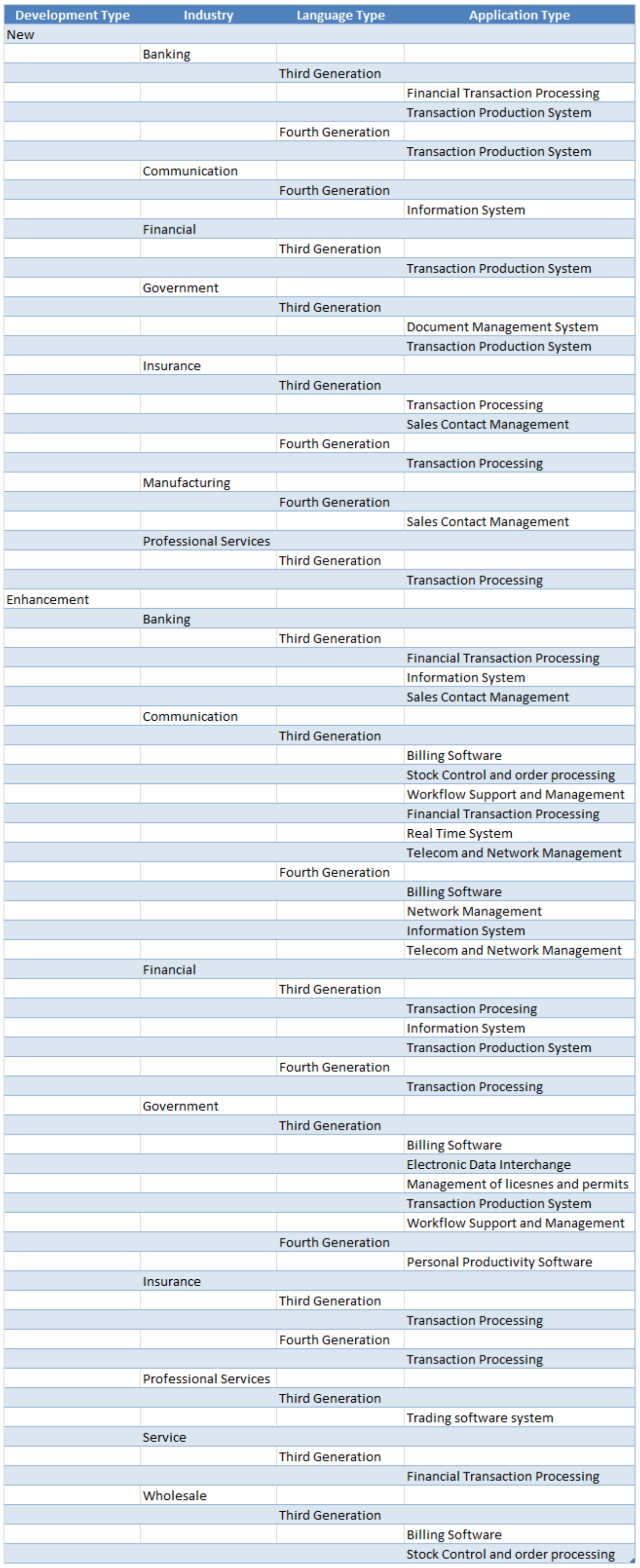
[3]. Data points that were missing either a Functional Size measure or an Effort value were discarded. Finally only projects with a Resource Level of 1 were included. Resource Level 1 indicates that the effort only includes that of the activities directly associated with development team effort (exclusive of support or overhead functions).

The application of the above filters left 2586 data points for calibration and template creation. At this point it was time identify the best calibration approach. The first step to successful calibration is to make sure that the data within the database is normalized so that cost driver, cost and effort data are all in the same units and relative to the same set of activities. This normalization is part of the value that the ISBSGs personnel add to the data. Since effort rather than cost is collected it is unnecessary to worry about escalation, exchange rates or other economic factors. What did need to be addressed was the fact that not every project reported covered the same set of software development activities. Submitters are asked to indicate which activities their reported effort includes. The ISBSGs uses this information along with standard distributions they have learned from their data analysis, to create a normalized effort value to make apples to apples analysis possible.

From this point, the exercise entered into an iterative mode to create calibrated templates for as many scenarios as possible. The goal of this analysis was to create scenarios that were detailed enough to be relevant but for which there were enough data points to give them meaning. The choices for which fields should be used to filter were driven by analysis of the data and application of expert judgment. To create scenarios the following process was followed (all column references are relative to the November 2011 release [2]

1. Data was filtered for an Industry Sector, for example Banking (Column E)
2. Data was filtered for a particular Application Types, for example Billing. This step required judgment because it appears to be a freeform entry and many submitters listed several different application types so assumptions were required to create this filter (Colum L)
3. Data was filtered for Development Type – either Enhancement or New Development (Column AM)
4. Data was filtered on Language Type – either 3GL or 4GL (Column J)
5. Review Normalized Level 1 PDR (Product Delivery Rate) – examine the statistics of this column and identify and eliminate data points that are apparent outliers.
6. Review the data points remaining and if there are at least 5, proceed to the calibration study. If not undo the filters and start the analysis over for the next Industry Sector, Application Type, Development Type, Language Type quartet.

Table 1 shows the scenarios for which there was enough data to create templates



Development Type	Industry	Language Type	Application Type
New			
	Banking		
		Third Generation	
			Financial Transaction Processing
			Transaction Production System
		Fourth Generation	
			Transaction Production System
	Communication		
		Fourth Generation	
			Information System
	Financial		
		Third Generation	
			Transaction Production System
	Government		
		Third Generation	
			Document Management System
			Transaction Production System
	Insurance		
		Third Generation	
			Transaction Processing
			Sales Contact Management
		Fourth Generation	
			Transaction Processing
	Manufacturing		
		Fourth Generation	
			Sales Contact Management
	Professional Services		
		Third Generation	
			Transaction Processing
Enhancement			
	Banking		
		Third Generation	
			Financial Transaction Processing
			Information System
			Sales Contact Management
	Communication		
		Third Generation	
			Billing Software
			Stock Control and order processing
			Workflow Support and Management
			Financial Transaction Processing
			Real Time System
			Telecom and Network Management
		Fourth Generation	
			Billing Software
			Network Management
			Information System
			Telecom and Network Management
	Financial		
		Third Generation	
			Transaction Procesing
			Information System
			Transaction Production System
		Fourth Generation	
			Transaction Processing
	Government		
		Third Generation	
			Billing Software
			Electronic Data Interchange
			Management of licesnes and permits
			Transaction Production System
			Workflow Support and Management
		Fourth Generation	
			Personal Productivity Software
	Insurance		
		Third Generation	
			Transaction Processing
		Fourth Generation	
			Transaction Processing
	Professional Services		
		Third Generation	
			Trading software system
	Service		
		Third Generation	
			Financial Transaction Processing
	Wholesale		
		Third Generation	
			Billing Software
			Stock Control and order processing

Table 1: Cost object template scenarios

5.0 Calibration Process

The selected data points are used to build software component models for the estimation tool by filling in the relevant inputs. The process and assumptions follow:

1. Application Type – leave at default as this is a calibration target
2. Operating Platform - Select value indicating Commercial Software (1.0 for our analysis)
3. Calibration constant – leave at default as this is a calibration target
4. Size Units = Function Points
5. New Code Size = 75% of Functional Size from ISBSG if project is New Development
6. Adapted Code Size = 75% of Functional Size from ISBSG if project is an Enhancement
7. Reused Code Size = 25% of Functional Size from ISBSG
8. Percent Design, Code and Test Adapted = 25%
9. Language = the Primary Programming Language from ISBSG.
10. Start Date = January 1st of the Start Year from the ISBSG
11. Team Experience = a quantification based on analysis of experience of Bas and IT personnel (Columns FF through FN). In cases where these columns were blank, the nominal value for team experience was applied

The percent assumptions above may seem somewhat arbitrary, but it has been this authors experience that few projects in this day and age are comprised completely of new or modified code, there is always some level of reuse even if this is not formally stated. While it is true that amount of reused capability is not really relevant in a benchmark study, this information is fairly important in an estimation exercise. This assumption was partially validated in the fact that it made calibration results within an application type much more consistent. Having said this, the risk exists that this assumption is not representative in all cases of the actual project conditions. This risk can be somewhat mitigated by the fact that the templates carry with them the same percentages as the analysis used.

The calibration process applied for this exercise was slightly different than a typical calibration exercise in that it focused on two input variables at once. The goal of this calibration was not to find the best Calibration Constant and Application Type for each data point but rather to find the pair that best fit the entire data set for the scenario being evaluated. One way to accomplish this would have been with an average value for each input based on individual calibration results. The non-linear nature of the Calibration Constant in the True S model made the average appear to be a poor choice. Automation was created that iterated through thousands of combinations and created statistics for the data set in the form of both an r-Square value and a Pred (50) value. (Pred (50) indicates how often the results applying a particular pair of Calibration Constant and Application Type come within 50% of actuals within the data set – Pred (30) was originally used but the data set was too noisy for that to be affective). The statistics were examined and the pair that appeared to be the best 'model' was selected.

This process was applied to each scenario identified in Table 1.

6.0 Building Scenario template cost objects

The final step in this exercise involved building the actual cost object in TruePlanning. One cost object was created for each scenario by setting the inputs as follows:

1. Application Type = value determined by the calculation
2. Operating Platform = value indicating commercial software
3. Calibration Constant= value determined by the calculation
4. Size Units = IFPUG Function Points
5. New Code Size = 75% of average Functional Size for the data set if Development Type is New, otherwise 0
6. New Code Non Executable = 15%
7. Adapted Code Size = 75% of average Functional Size for the data set if the Development type is Enhancement , otherwise 0
8. Adapted Code Non Executable = 15%
9. Percent Design, Code and Test Adapted = 25%
10. Language = most frequently occurring language in the data set.
11. Team Complexity = nominal value

A risk range was then assigned to the inputs for Application Type, New Code Size or Adapted Code Size and Calibration Constants inputs. The results of the statistical analysis were used to determine the proper risk range. Any analysis that resulted in an r-Square greater than or equal to .8 **and** with a Pred(50) value greater than or equal to 70% was deemed lower risk and the risk range assessment was aligned with the risk that might be applied to a project in the Detailed Design phase. Any analysis with an r-Square greater than or equal to 0.8 **or** with a Pred(50) greater than or equal to 70% was considered medium risk and the risk range was assessed along the lines of a project that is in the Preliminary Design Phase where there are more unknowns. All others were assessed with the risk one might find in the Early Concept stage when very little is known about a project thus resulting in rather wide uncertainty distributions.

The final step in completing the scenario templates involved documentation. For each scenario the following items were included in the software component notes:

1. Sample Size

2. r-Square value
3. Pred(30)
4. Pred(50)
5. Most frequently occurring language in the data set
6. List of the ISBSG id Numbers of the data items used in the sample set (for easy identification for those with access to ISBSG data)

These templates were created in a spread sheet format easily importable into the TruePlanning estimation framework.

7.0 Conclusions

These templates provide a data driven approach to estimating certain types of software. The lesson one learns when examining data of any kind is that there is almost always noise in the data. While the ISBSG data is one of the best sources for well verified, clean data, there still seems to be a fair amount of noise. This is bound to happen when we are relying on surveys from multiple sources – not everyone is going to interpret all of the questions the same way and no amount of documentation is going to change that. Additionally the freeform way some of the fields are populated required another level of subjectivity to be injected into the process.

This being said, the creation of such templates will still provide software estimators who don't have historical data with a valuable tool for their estimation exercises. The estimator is provided with a set of templates that should be used as a starting place for their estimates. Clearly the average Functional Size input used is unlikely to match the specific requirements for any specific estimate, but it adds value in two ways. It provides a useful range from similar projects in the past and it gives the estimator one way of evaluating whether this template is a good match for his particular estimate. (If the estimator is estimating a project expected to be 50 Function Points and the average value in the template is closer to 50,000 Function Points – the analysis may very well not be relevant). The Calibration Constant and Application Type values provide a good starting point for an estimate where historical information is unavailable. The documentation embedded in each template cost object along with the default risk ranges should help an estimator determine how well this template supports their estimating requirements, as not all estimates require the same fidelity. While these templates do not purport to be the definitive solution to all estimating challenges, they add another weapon to the estimator's arsenal to support defendable estimates.

References

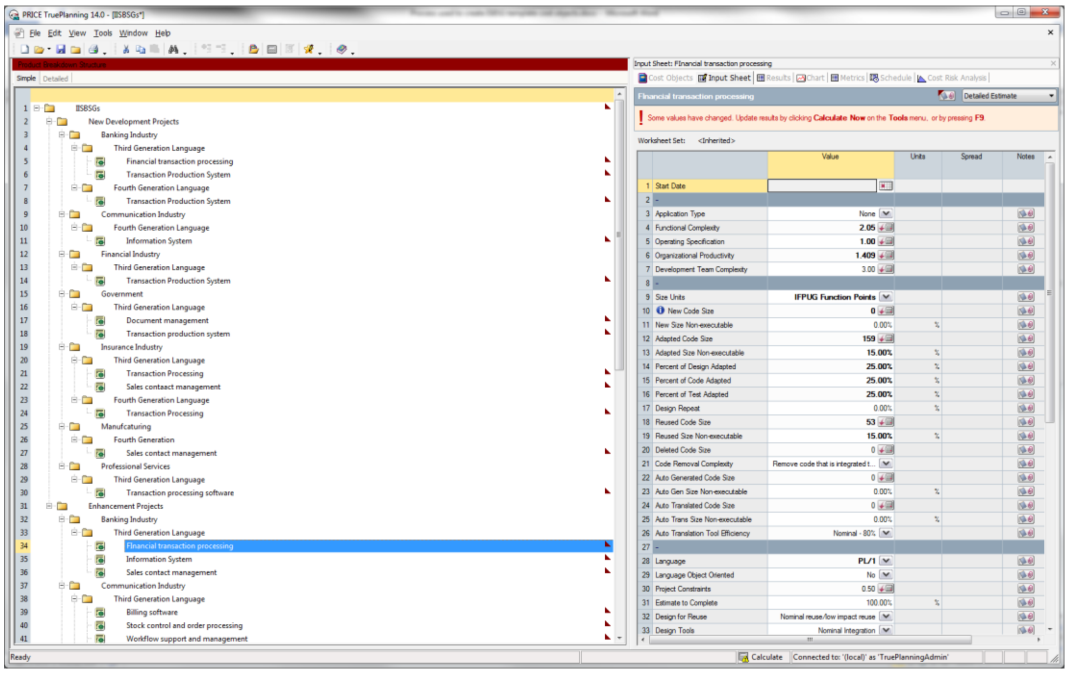
[1] www.isbsg.org

[2] ISBSG Corporate Data Release 291011.xlsx (available from ISBSG)

[3] FPA According to NESMA and IFPUG; the present situation, version August, 2011, available at http://www.nesma.nl/download/artikelen/FPA%20according%20to%20NESMA%20and%20IFPUG%20-%20the%20present%20situation%20(vs%202011-08-01).pdf, (viewed May 2013)

Appendix A – Overview of TruePlanning for Software

The TruePlanning for Software model estimates cost, effort and schedule for software projects. It combines time tested parametric cost and effort cost estimating relationships (CERs) into an activity based framework for convenient planning and budgeting. True S contains many cost and effort drivers. The right hand side of Figure 1 contains a screen shot of the True S input sheet.



```
1  IISBSGs
2    New Development Projects
3      Banking Industry
4        Third Generation Language
5          Financial transaction processing
6          Transaction Production System
7        Fourth Generation Language
8          Transaction Production System
9      Communication Industry
10       Fourth Generation Language
11         Information System
12     Financial Industry
13       Third Generation Language
14         Transaction Production System
15     Government
16       Third Generation Language
17         Document management
18         Transaction production system
19     Insurance Industry
20       Third Generation Language
21         Transaction Processing
22         Sales contaact management
23       Fourth Generation Language
24         Transaction Processing
25     Manufcaturing
26       Fourth Generation
27         Sales contact management
28     Professional Services
29       Third Generation Language
30         Transaction processing software
31   Enhancement Projects
32     Banking Industry
33       Third Generation Language
34         FInancial transaction processing
35         Information System
36         Sales contact management
37     Communication Industry
38       Third Generation Language
39         Billing software
40         Stock control and order processing
41         Workflow support and management
```

FInancial transaction processing

Some values have changed. Update results by clicking **Calculate Now** on the **Tools** menu, or by pressing **F9**.

Worksheet Set: <Inherited>

		Value	Units	Spread	Notes
1	Start Date				
2	-				
3	Application Type	None			
4	Functional Complexity	**2.05**			
5	Operating Specification	**1.00**			
6	Organizational Productivity	**1.409**			
7	Development Team Complexity	3.00			
8	-				
9	Size Units	**IFPUG Function Points**			
10	New Code Size	**0**			
11	New Size Non-executable	0.00%	%		
12	Adapted Code Size	**159**			
13	Adapted Size Non-executable	**15.00%**	%		
14	Percent of Design Adapted	**25.00%**	%		
15	Percent of Code Adapted	**25.00%**	%		
16	Percent of Test Adapted	**25.00%**	%		
17	Design Repeat	0.00%	%		
18	Reused Code Size	**53**			
19	Reused Size Non-executable	**15.00%**	%		
20	Deleted Code Size	0			
21	Code Removal Complexity	Remove code that is integrated t...			
22	Auto Generated Code Size	0			
23	Auto Gen Size Non-executable	0.00%	%		
24	Auto Translated Code Size	0			
25	Auto Trans Size Non-executable	0.00%	%		
26	Auto Translation Tool Efficiency	Nominal - 80%			
27	-				
28	Language	**PL/1**			
29	Language Object Oriented	No			
30	Project Constraints	0.50			
31	Estimate to Complete	100.00%	%		
32	Design for Reuse	Nominal reuse/low impact reuse			
33	Design Tools	Nominal Integration			

Connected to: '(local)' as 'TruePlanningAdmin'

Figure 1: TruePlanning for Software Input Screen

The specific True S inputs that equate to the cost drivers mentioned in Section 3.0 are:

Type of Application – Functional Complexity

Operating Platform – Operating Specification

Code Size – New Code Size, Reused Code Size, Adapted Code Size, Percent Design Adapted, Percent Code Adapted, Percent Test Adapted

Team Experience – Development Team Complexity

Calibration Constant – Organizational Productivity

Programming Language – Language