Improvement Opportunities and Suggestions for Benchmarking

Cigdem Gencel[1], Luigi Buglione[2,3], and Alain Abran[2]

[1] Blekinge Institute of Technology, Sweden
cigdem.gencel@bth.se
[2] Ecole de Téchnologie Superieure (ETS) – Université du Québec à Montreal (UQAM)/
[3] Nexen (Engineering Group), Italy
luigi.buglione@eng.it, alain.abran@etsmtl.ca

Abstract. During the past 10 years, the amount of effort put on setting up benchmarking repositories has considerably increased at the organizational, national and even at international levels to help software managers to determine the performance of software activities and to make better software estimates. This has enabled a number of studies with an emphasis on the relationship between software product size, effort and cost drivers in order to either measure the average performance for similar software projects or to develop estimation models and then refine them using the collected data. However, despite these efforts, none of those methods are yet deemed to be universally applicable and there is still no agreement on which cost drivers are significant in the estimation process. This study discusses some of the possible reasons why in software engineering, practitioners and researchers have not yet been able to come up with reasonable and well quantified relationships between effort and cost drivers although considerable amounts of data on software projects have been collected. An improved classification of application types in benchmarking repositories is also proposed.

Keywords: Benchmarking Repositories, Performance Measurement, Effort Estimation, Cost Drivers.

1 Introduction

Software project management provides a number of challenges in comparison to managing projects in traditional engineering disciplines. Software engineering being a new discipline, the amount of accumulated data and know-how is much less extensive and most is not derived from large sets of controlled experiments.

As of today, there is not yet a software estimation method or model which has a large acceptance in the software community: reliable estimation of budget and duration and allocation of staff and other resources for a new project are still significant management challenges to the software industry.

Over the past decade, software engineering community has identified the need to develop benchmarking repositories (such as the International Software Benchmarking

A. Abran et al. (Eds.): IWSM/Mensura 2009, LNCS 5891, pp. 144–156, 2009.

Standards Group (ISBSG) Dataset[1] [12], the Promise Dataset [23], Laturi/Finnish Software Metrics Association (FISMA) Experience Database [10]) to provide to the community publicly available benchmark data and experience bases[2] that supports reuse of experience and collective learning by understanding, assessing and packaging of the data [2][3].

However, there does not yet exist an international standard on how to develop benchmarking repositories for software: the different repositories were developed independently in different environments, in different countries by using different terminologies, attributes and categories. Therefore, it is difficult to map the attributes of one repository to another in order to compare the results of various empirical studies or to replicate those studies using other repositories to verify if findings could be confirmed and generalized to other contexts.

Some of the issues that are related to standard definitions for the project and product related attributes (e.g. Functional Size, Length of Code, Development Effort, Application Type) as well as the categories associated with the categorical attributes (e.g. Management Information System, Process Control System, etc. for the Application Type attribute) have been worked out, discussed and reviewed extensively by measurement experts from all over the world such as in [6][12] [16][25] and by various measurement associations like the benchmarking repository developers as ISBSG.

Many national or international measurement associations have also been working on improving the benchmarking process, including the Common Software Measurement International Consortium[3] (COSMIC), the International Function Point Users Group[4] (IFPUG), the United Kingdom Software Metrics Association[5] (UKSMA), the Finnish Software Metrics Association[6] (FiSMA) and China Software Benchmarking Standards Group[7] (CSBSG).

However, although considerable amount of effort has been put forth to develop high quality benchmarking repositories, there still exist some improvement opportunities that could bring additional benefits to the software community.

Within the scope of this paper we highlight some improvement opportunities for the benchmarking process and provide some suggestions, especially for the use of benchmarking repositories in performance measurement and in effort estimation. Specifically, we focus on the definition and categorization of the project-related attributes in the benchmarking repositories.

The paper is organized as follows: Section 2 presents the improvement opportunities for benchmarking and estimation purposes. Section 3 presents a proposal for better classification of application types which is one of the significant attributes for performance measurement and effort estimation. Section 4 presents the conclusions and future work of this study.

---

[1] http://www.isbsg.org. The data analysis in this paper is based on ISBSG Dataset v10. However a newer version of the dataset is published recently.

[2] "An information store that contains the evaluation of the information products and the benchmarking procedure as well as any lessons learned during benchmarking and analysis" [8].

[3] http:www.cosmicon.com

[4] http://www.ifpug.org

[5] http://www.uksma.co.uk

[6] http://www.fisma.fi

[7] http://www.csbsg.org

2 Improvement Opportunities for Benchmarking and Estimation

One common approach in industry for estimating effort is to use the average performance of an organization on similar projects completed and to take into account a variety of cost drivers. The cost drivers can be *project*, *product* or *development organization* related attributes. As the benchmarking repositories grow, the assumption is that more accurate estimations can be made.

The organizational know-how is important in making more reliable estimates. However, if organizational data is lacking, which is usually the case, another approach is to use ‘top-down’ estimation models (such as COCOMO II [4], Putnam’s Model/SLIM [18], SoftCost [26], Price-S [24], Galorath SEER-SEM[8] and Cost Xpert[9], etc.) available in the market place or to use industry averages or publicly available benchmarking repositories.

In [9], Cukic claims that the lack of publicly available benchmarking repositories results in poorly validated estimation models, this leading to a distrust regarding many existing estimation models as well as the proliferation of new ones. One of the findings of the Jorgensen and Shepperd systematic review on the research on software development effort and cost estimation [17] is that most of the researchers evaluate estimation methods by picking one or more available repositories and leave it to the readers to decide the degree to which it is possible to generalize the results to other contexts and other sets of projects. They state that this is one of the reasons why systematic aggregation of the research results in this field is still challenging.

In order to make better performance measurements and to develop improved estimation methods or to evaluate the existing ones; it is necessary not only to collect data, but also to collect the data in a format relevant for such purposes.

Figure 1 represent a Root-Cause Analysis (RCA) expressed by a *mind map*, as suggested in [5], analyzing some of the most relevant causes that might lead to unreliable estimates when using benchmarking repositories.

This list can be extended and does not aim to be all-inclusive. Within the scope of this paper we elaborate on some of those causes in order to identify improvement opportunities for the benchmarking repositories available for benchmarking and effort estimation.

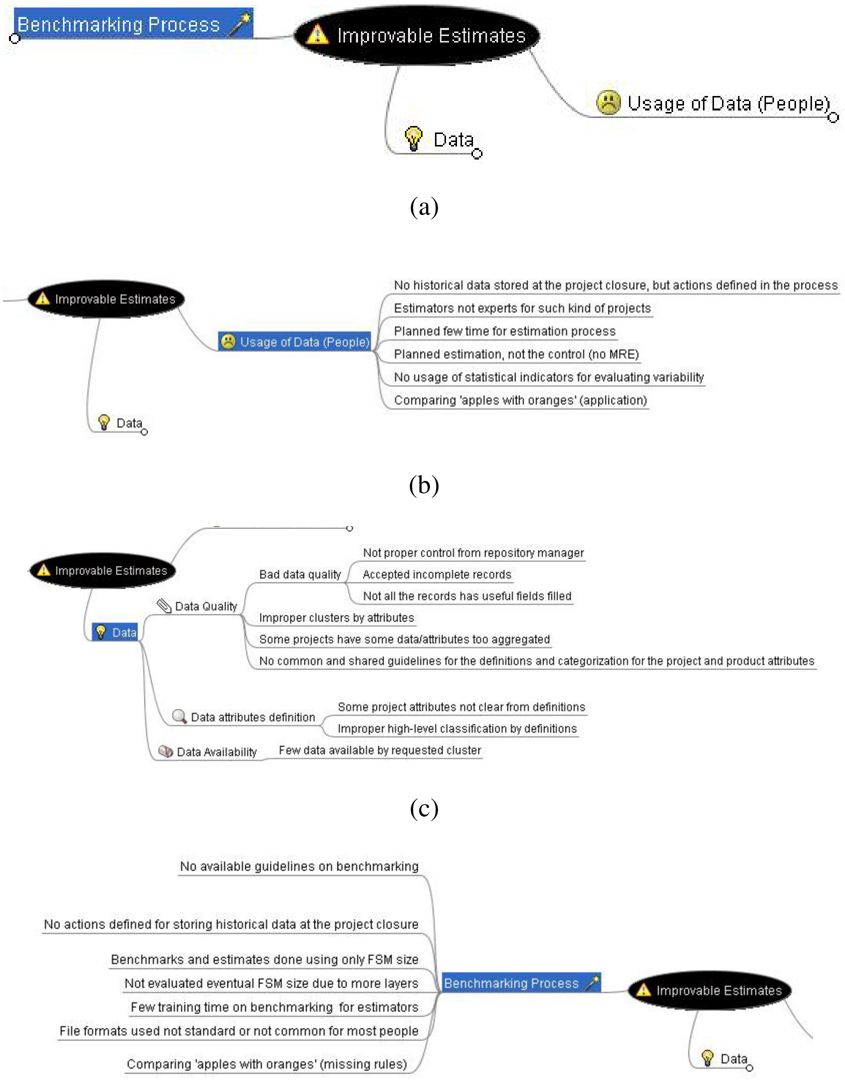
In particular, the issue for benchmarking is two-fold: on one hand, it is necessary to verify the source of data we intend to use in terms of completeness, clearness and consistency of definitions applied, quality of data in a historical data series, etc. On the other hand, it is necessary to have access to common and shared guidelines for the definitions and categorization of the attributes of the entities involved in developing software benchmarking repositories. A guiding principle in any benchmarking activity is to avoid comparing ‘apples to oranges’.

Accordingly, we identified one of the significant improvement opportunities as the development of a standard and unified vocabulary, definitions and categories for the benchmarking repository attributes. This would allow local and international repositories to map their specific definitions and categories to the standard ones

---

[8] http://www.galorath.com

[9] http://www.costxpert.com


Benchmarking Process
Improvable Estimates
Usage of Data (People)
Data
No historical data stored at the project closure, but actions defined in the process
Estimators not experts for such kind of projects
Planned few time for estimation process
Planned estimation, not the control (no MRE)
No usage of statistical indicators for evaluating variability
Comparing 'apples with oranges' (application)
Data Quality
Bad data quality
Not proper control from repository manager
Accepted incomplete records
Not all the records has useful fields filled
Improper clusters by attributes
Some projects have some data/attributes too aggregated
No common and shared guidelines for the definitions and categorization for the project and product attributes
Data attributes definition
Some project attributes not clear from definitions
Improper high-level classification by definitions
Data Availability
Few data available by requested cluster
No available guidelines on benchmarking
No actions defined for storing historical data at the project closure
Benchmarks and estimates done using only FSM size
Not evaluated eventual FSM size due to more layers
Few training time on benchmarking for estimators
File formats used not standard or not common for most people
Comparing 'apples with oranges' (missing rules)

(a)

(b)

(c)

(d)

Fig. 1. Root-Cause Analysis (RCA) for Improvable Estimates: (a) root leaves; (b) Usage of Data (People) leaf; (c) Data leaf; (d) Benchmarking Process leaf

accepted by the international community. This might also help in providing a more rigorous approach to refine and improve the existing definitions and categories in a more organized way.

An example is presented next to illustrate the significance of this improvement opportunity. A number of empirical studies were performed utilizing the projects data in different repositories to investigate the *product functional* size-effort relationship.

Among the *project* cost drivers[10] investigated (in terms of ISBSG terminology) the Functional Size, Team Size, Programming Language Type, Development Type, Organization Type, Business Area Type, Application Type and Development Platform have been found to affect the *product* size - *project* effort relationship at different levels of significance [1][11][19][20][21][22]. Can the results of those studies be generalized to other contexts?

The details of some of the attributes on which these studies derived some conclusions are provided in Appendix. ISBSG Dataset [12] has three different attribute definitions to provide context to the work effort variable[11] which is a basic attribute to measure the average performance of the projects in a repository:

- *Summary Work Effort:* "Provides the total effort in hours recorded against the project."
- *Normalized Level 1 of Work Effort*: "The development team full life-cycle effort. For projects covering less than a full development life-cycle, this value is an estimate of the full life-cycle effort for the development team only."
- *Normalized Work Effort:* "Full life-cycle effort for all teams reported. For projects covering less than a full development life-cycle, this value is an estimate of the full development lifecycle effort."

On the other hand, Laturi/FiSMA Experience Database [10] uses 'Cumulative Work Effort'. It is defined as "The effort measured from the planning up to the installation and/or user training in person hours". Can we really assume derived 'Normalized Work Effort' based on ISBSG's specific rules to correspond to 'Cumulative Work Effort' in the Experience Database?

Another significant categorical attribute is 'Application Type' of projects. Different repositories have different pre-defined categories; the repositories also leave the data providers the flexibility to enter a new type when none of the categories fit (see Appendix).

The non-standardization of these attributes and categories might be significant:

- Within one repository
- Across repositories

For instance, the Release 10 of the ISBSG Dataset [12] contains 4,106 projects. For 1,177 projects, the Application Type is not reported. For 134 projects, the Application Type is reported as 'Other' since these did not fall in any of the ISBSG categories: among those projects, for 111 projects there exists a definition for the 'Other' type Application Type while the others are qualified only as 'Other'.

For some of the projects, more than one Application Type is reported. For example:

- Customer billing/relationship management; Business;

---

[10] According COCOMOII model "a cost driver refers to a particular characteristic of the software development that has the effect of increasing or decreasing the amount of development effort, e.g. required product reliability, execution time constraints, project team application experience".

[11] In the ISBSG dataset, the second and third attributes are not collected directly but are transformed values, calculated on the basis of a number of variables.

- Customer billing/relationship management; Document management; Trading;
- Customer billing/relationship management; CRM;
- Customer billing/relationship management; Document management; Trading;
- Customer billing/relationship management; Financial transaction process/accounting; Online analysis and reporting; Trading; Workflow support & management; Process Control; Electronic Data Interchange;
- Customer billing/relationship management; Logistic or supply planning & control;
- Customer billing/relationship management; Other;
- Customer billing/relationship management; Other;
- Customer billing/relationship management; Stock control & order processing;

These project data must be handled with caution by the researchers for any analysis when attempting to identify the significance of this cost driver. Moreover, when this is done, this leads to a number of data points being filtered out for further analyses.

Another problem occurs when making analyses among different repositories. If one study based on the ISBSG dataset concludes that Application Type is significant in modeling the effort relationship with the categories defined, another study with a distinct dataset may not find the same pattern based on other categories. Then, which one should be considered by the practitioners as a basis for estimation?

Therefore, there is a need for the standard definitions and better categories for at least the cost drivers identified by different studies as significant such as Team Size, Programming Language Type, Development Type, Organization Type, Business Area Type, Application Type and Development Platform so that we can compare and generalize the results.

It is also needed to report Effort and Duration based on a standard way of expressing the Software Development Life Cycle (SDLC) phases. The information regarding the derived attributes should also be provided. For example, the relationship between effort recorded for different phases in the life cycle and the derived normalized effort which involves all the development phases should be well defined. When this is defined differently from one repository to another, then inconsistencies in analyses are unavoidable.

3 A Suggestion for Better Classification of Application Types

In this section, we propose a way of classifying the application types in Software Engineering based on the practices for classification in Civil Engineering and two Software Engineering standards.

3.1 Classification Practices in Civil Engineering

In Civil Engineering, the parametric estimating method employs databases in which key project parameters, which are priced from past projects using appropriate units, are recorded [7].

Various parametric systems exist for different types of civil engineering projects. An example can be as follows:

- Building
    - Apartment
        - Low rise
        - Mid rise
        - High rise
    - Airport
    - Hospital
    - …
- Motorway
    - Highway
    - …
- Bridge
- …

In this example, the type of the building is recorded in the historical databases as a categorical (nominal) parameter. The buildings are categorized into different sub-types such as Apartment, Bank, Hospital, etc. Each of these types is further categorized with respect to another categorical parameter; height and population density. For example, an Apartment can be categorized into low rise, mid rise, or high rise. Median cost per unit of measure for each parameter is recorded for each category. Other parameters important in estimating the cost (such as the location of the building) are also recorded.

For the Building type projects, one example of a parametric system is presented in the following table.

Table 1. Building parameters

Parameter
Site work
Foundations and columns
Floor system
Structural system
Roof system
Exterior walls
Interior walls
Electrical
Conveying systems
Plumbing
Finishes

As in Civil Engineering products, different types of software engineering products are being developed such as Management Information System (MIS), Process Control Software, Embedded Software, Real Time Software, etc. In the next section, different software application types are elaborated and an approach for better classification of the application types is proposed.

3.2 Application Type Classification for Software

In the software engineering standardization community, the ISO TR 12182 [14] defines software types and the ISO TR 14143-5 [15], the elements of Functional Domains[12] (in this paper 'Application Type' is consistently used) were identified for the specific needs of Functional Size Measurement (FSM) community. Here, the classification of ISO TR 12182 [14] is considered (see Table 2) and the defined software functionality types are explicitly mapped to respective software application types by using different methods.

One of the methods recommended in ISO 14143-5 [15] for determining the application types is the CHAR Method. In the CHAR Method, the characteristics (CHARs) of functional user requirements (FUR) are categorized into three groups; data-rich, control-and communication rich and manipulation-and algorithm-rich (see Table 2). This table provides some examples of well-known application types and is not intended to be an all-inclusive list. The CHAR Method uses a rating scale to determine the amount of functionality types in an application:

- *Negligible*: no CHAR present, or sum of CHARs < 3 % of requirements
- *Present*: at least one CHAR present and sum of CHARs < 50 % of requirements
- *Dominant*: one CHAR dominant or sum of CHARs > 50 % of requirements

One of the benefits of this method is that the application types of software are classified based on the requirements types and the audience is always aware of the detailed functionalities existing in a specific application. Therefore, it is possible to add one more application type to the existing list whenever new functionality types are added to the above.

The following examples show how this categorization might be more beneficial when using these projects' data for benchmarking purposes.

Let's suppose data on a number of projects are to be stored into a benchmarking repository (see Table 3). These example projects are real projects and we have detailed knowledge on these projects and organizations. For each project, Functional Size (in COSMIC Function Points - CFP), Development Effort and Productivity Delivery Rate (PDR) are given in Table 3.

Two of the projects' data are provided by one software organization; Organization A:

- Project-1 is a development project of one of the subsystems of an avionics managements system for small to medium size commercial aircrafts on a Flight Display System.
- Project-2 is a Collision Avoidance Subsystem of the Traffic Alert and Collision Avoidance System.

Two other projects' data are provided by another organization; Organization B:

- Project-3 involves the development of a multimedia sponsored call system.
- Project-4 involves the development of an equipment identification registrar which detects and warns the operator against potential fraud risks such as Subscriber Identity Module (SIM) card cloning and International Mobile Equipment Identity (IMEI) cloning.

---

[12] Functional Domain: "a class of software based on the characteristics of FUR which are pertinent to FSM" [13]. 'Software systems', 'Fields of application', 'Application type' are some of the terms used for Functional Domain in different resources.

Table 2. Application Types based on CHAR Method [15], ISO 12182 [14] and Analysis of software 'types' [15]

ISO 12182 software type	CHAR Method - Functional Domain Types	Control and Communic. Rich	Data-Rich	Manipulation and Algorithm-Rich
(no corresponding type)	Pure Data Handling System	Negligible	Dominant	Negligible
Management Information System (Business transaction processing), Decision Support	Information System	Negligible	Dominant	Present
Word Processing, Geographic Information System	Data Processing System	Negligible	Present	Present
(no corresponding type)	Controlling Information System	Present	Dominant	Negligible
Automated Teller Banking	Controlling Data System	Present	Present	Negligible
Business (Business Enterprise)	Complex Controlling Information System	Present	Dominant	Present
Military Command and Control	Non-Specific (Complex) System	Present	Present	Present
Real Time: Embedded, Device (Printer, Disc, etc.) Driver	Simple Control System	Dominant	Negligible	Negligible
(no corresponding type)	Control System	Present	Negligible	Present
Real Time: Embedded, Avionics, Message router	Complex Control System	Dominant	Negligible	Present
E-mail, Emergency dispatch call/receipt, Operating System	Data Driven Control System	Dominant	Present	Negligible
Process Control (Control System)	Complex Data Driven Control System	Dominant	Present	Present
Scientific, Standard math/Trig. Algorithms	Pure Calculation System	Negligible	Negligible	Dominant
Engineering	Controlling Calculation System	Present	Negligible	Dominant
Self-learning (Expert or Artificial Intelligence), Statistical, Spreadsheet, Secure Systems, Actuarial	Scientific Information System	Negligible	Present	Dominant
Safety Critical	Scientific Controlling Data Processing System	Present	Present	Dominant

Table 3. Functional domains of the case projects determined by CHAR Method

No	**Funct. Size (CFP)**	**Develop. Effort (work-hrs)**	**Productivity Delivery Rate (PDR) (work-hrs/ CFP)**	**(Application Type) Functional Domain Type**	**Contr- and Com rich FUR**	**Data-rich FURs**	**Manip. And Alg. rich FURs**
1	4036	18,003	4.46	Complex Data Driven Control System	Dominant	Present	Present
2	945	2,200	2.33	Complex Control System	Dominant	Negligible	Present
3	321	1,080	3.37	Complex Controlling Information System	Present	Dominant	Present
4	275	1,200	4.36	Information System	Negligible	Dominant	Present

Table 3 shows how to categorize these projects into application types by using the CHAR method.

This kind of categorization with an explicit meaning of an Application Type helps to understand partially the variations in PDR values. For example, the presence of data-rich FURs in the first application might be a significant factor which makes this application a hybrid and more complex system to develop.

For the other two projects (Project-3 and Project-4), even with this categorization it is not possible to explain why the PDR for Project-3 is less than Project-4 although Project-3 is a hybrid system as well. In fact, if the scale used by CHAR method defined a scale which distinguishes the amount of specific kinds of requirements; such as a rating between 1 to 5 instead of 1 to 3, then we would have seen that Project-4 involves much more manipulation and algorithm-rich FURs than Project-3, which also increases the complexity of Project-4 and decreases the PDR.

In practice, variations in PDR might be due to a lot of other variables, but a proper categorization of projects by their application types can at least help in making analyses on more homogeneous data subsets.

In its current form, the CHAR Method's categorization only distinguishes between 'negligible', 'present' and 'dominant'. This causes to lose data which is already available. However, this is not a weakness of the CHAR method since this method was developed with a specific purpose of assessing the applicability of different Functional Size Measurement methods to different software application types. And this level of differentiation is sufficient for that initial purpose.

We suggest extending this method's usage area in software engineering to a broader context: identification of the Application Type for a software product which is significant for many purposes including performance measurement and effort estimation using benchmarking repositories.

4 Conclusions and Prospects

Over the past years, interest in software benchmarking has been growing. Although considerable improvements have been accomplished to better serve the software engineering community, there is still some room for further improvements.

In this paper, the possible causes for inefficient usage of benchmarking repositories are identified for benchmarking and effort estimation purposes. One of the significant improvement opportunities for benchmarking is the development of a standard and unified vocabulary, definitions and categories for the benchmarking repository attributes. The significance of this was discussed by giving some examples from some of publicly available benchmarking repositories.

Another contribution of this paper is a proposal of an improved way of classifying the application types in software engineering derived from the practices for classification in civil engineering and two software engineering standards: ISO 12182 and 14143-5. Here, our aim was not to find the ultimate solution to categorization of software application types, but rather to provide an approach for refining it.

The future work involves implementing this method to identify application types of other projects and refine the method accordingly for one specific attribute; Application Type. There are other significant attributes such as Business Area Type, Development Type, etc. which require better categorization as well.

References

[1] Angelis, L., Stamelos, I., Morisio, M.: Building a Cost Estimation Model Based on Categorical Data. In: 7th IEEE Int. Software Metrics Symposium (METRICS 2001), London (April 2001)

[2] Basili, V.R., Bomarius, F., Feldmann, R.L.: Get Your Experience Factory Ready for the Next Decade – Ten Years after "How to Build and Run One". In: Companion to the Proceedings of the 29th international Conference on Software Engineering, May 20-26, pp. 167–168. IEEE Computer Society, Washington (2007)

[3] Basili, V., Caldiera, G., McGarry, F., Pajerski, R., Page, G., Waligora, S.: The software engineering laboratory: an operational software experience factory. In: Proceedings of the 14th intern. Conf. on Software Engineering, ICSE 1992, Melbourne, Australia, May 11-15, pp. 370–381. ACM, New York (1992)

[4] Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Bradford, K.C., Steece, B., Brown, A.W., Chulani, S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice Hall, New Jersey (2000)

[5] Buglione, L.: Strengthening CMMI Maturity Levels with a Quantitative Approach to Root-Cause Analysis. In: Proceedings of the 5th Software Measurement European Forum (SMEF 2008), Milan, Italy, May 28-30, pp. 67–82 (2008) ISBN 9-788870-909999

[6] Card, D., Zubrow, D.: Guest Editor's introduction, Benchmarking Software Organizations. IEEE Software, 16–17 (September/October 2001)

[7] CESMM. 1991: Civil Engineering Standard Method of Measurement, Thomas Telford Ltd., 3rd edn. (1991)

[8] ISBSG: SC 7 Proposed New Work Item on Software and systems engineering – IT Performance Benchmarking Framework (2008), `http://www.isbsg.org/ISBSGnew.nsf/WebPages/c7b3fcc5ce6308f7ca2574580013f206`

[9] Cukic, B.: The Promise of Public Software Engineering Data Repositories. IEEE Software, Guest Editor's introduction 22(6), 20–22 (2005)

[10] Experience Pro, `http://www.sttf.fi` (Last access: 2009/01/29)

[11] Forselius, P.: Benchmarking Software-Development Productivity. IEEE Software 17(1), 80–88 (2000)

[12] ISBSG Dataset 10 (2007), `http://www.isbsg.org`

[13] ISO/IEC 14143-1:2007: Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts (2007)

[14] ISO/IEC TR 12182:1998: Information technology – Categorization of software

[15] ISO/IEC TR 14143-5:2004 Information Technology – Software Measurement – Functional Size Measurement – Part 5: Determination of Functional Domains for Use with Functional Size Measurement

[16] Shirabad, J.S., Menzies, T.J.: PROMISE Software Engineering Repository, School of Information Technology and Eng., Univ. of Ottawa, Canada (2005), `http://promise.site.uottawa.ca/SERepository`

[17] Jørgensen, M., Shepperd, M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering 33(1), 33–53 (2007)

[18] Putnam, L.H.: A general empirical solution to the macro software sizing and estimating problem. IEEE Trans. Soft. Eng. 4(4), 345–361 (1978)

[19] Lokan, C., Wright, T., Hill, P.R., Stringer, M.: Organizational Benchmarking Using the ISBSG Data Repository. IEEE Software 18(5), 26–32 (2001)

[20] Maxwell, K.D.: Collecting Data for Comparability: Benchmarking Software Development Productivity. IEEE Software 18(5), 22–25 (2001)

[21] Morasca, S., Russo, G.: An Empirical Study of Software Productivity. In: Proc. of the 25th Annual International Computer Software and Applications Conference (COMPSAC 2001), Chicago, IL, USA, October 8-12, pp. 317–322 (2001)

[22] Premraj, R., Shepperd, M.J., Kitchenham, B., Forselius, P.: An Empirical Analysis of Software Productivity over Time. In: 11th IEEE International Symposium on Software Metrics (Metrics 2005), p. 37. IEEE Computer Society, Los Alamitos (2005)

[23] PROMISE Data Repositories, `http://promisedata.org` (last access: 2009/01/29)

[24] Park, R.E.: PRICE S: The calculation within and why. In: Proceedings of ISPA 10th Annual Conference, Brighton, England (July 1988)

[25] Park, R.: Software Size Measurement: A Framework for Counting Source Statements. Technical Report CMU/SEI-92-TR-020

[26] Tausworthe, R.: Deep Space Network Software Cost Estimation Model. Jet Propulsion Laboratory Publication 81-7 (1981)

Appendix A: Definitions and Classifications Some of the Attributes in ISBSG Dataset and Laturi/FiSMA Experience Dataset

ISBSG Dataset 10 [12]	**Laturi/FiSMA Experience Dataset [10]**
Normalized Level 1 Work Effort (The development team full life-cycle effort) **Normalized Work Effort** (Full life-cycle effort for all teams reported) **Summary Work Effort** (Provides the total effort in hours recorded against the project.)	**Cumulative work effort** (Measured from the planning up to the installation and/or user training in person hours)
Organization Type (This identifies the type of organization that submitted the project. (e.g.: Banking, Manufacturing, Retail)).	**Organization's sector** (banking, retail, insurance, management or manufacturing)
Application Type (3D modeling or automation, Artificial Intelligence, Catalogue/register of things or events, Customer billing/relationship management, Decision Support, Device or interface driver, Document management, Electronic Data Interchange, Executive Information System, Fault Tolerance, Financial transaction process/accounting, Geographic or spatial information system, Graphics & publishing tools or system, Image, video or sound processing, Embedded software for machine control, Job, case, incident, project management, Logistic or supply planning & control, Management Information Systems, Management or performance reporting, Mathematical modeling (finance or eng.), Network Management, Office Information System, Online analysis and reporting, Operating system or software utility, Personal productivity (e.g. spreadsheet) Process Control, Software development tool Stock control & order processing, Trading, Transaction/production system, Workflow support & management)	**Type of application** (Customer service, MIS, OIS, process control and automation, network management, transaction processing, production control and logistics, online and information services.)
Development Platform (Device Embedded, PC, Mid Range, Main Frame or Multi platform.)	**Development target platform** (Network, mainframe, PC, mini-computer, combination)