Similar Tasks, Different Effort: Why the Same Amount of Functionality Requires Different Development Effort?

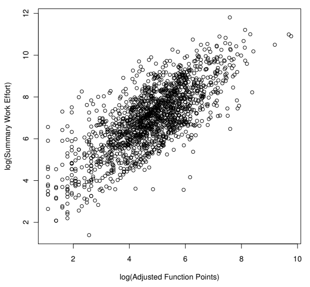
Alexander Serebrenik, Bogdan Vasilescu, Mark van den Brand
Technische Universiteit Eindhoven,
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
{a.serebrenik, b.n.vasilescu, m.g.j.v.d.brand}@tue.nl

***Abstract*—Since the appearance of Albrechts pioneering work, function points have attracted significant attention from the industry. In their work, project managers can benchmark function point counts obtained for their projects against large publicly available datasets such as the ISBSG development & enhancement repository release 11, containing function point counts for more than 5000 projects. Unfortunately, larger amount of functionality as reflected in the function points count does not necessarily correspond to a more significant development effort. In this paper we focus on a collection of ISBSG projects with a similar amount of functionality and study the impact of different project attributes on the development effort.**

In our study we consider the ISBSG development & enhancement repository release 11, the largest publicly available dataset with function point counts, containing data about more than 5000 projects developed in a variety of different countries using a variety of different design and development techniques [1]. The ISBSG repository contains information about 118 different project attributes, including its functional size as well as organizational (e.g., scheduling, development team size and its productivity), technical (e.g., architecture and the main programming language), and problem-specific attributes (e.g., business or application area). Data is provided by the project owners themselves. Functional size is for most of the projects measured by applying such methods as IFPUG function points (3799 out of 5052 projects or 75.2%) [2], hence we solely focus on the IFPUG projects.

For the IFPUG projects, the ISBSG repository contains data on the development effort. First, we exclude projects that report the effort figures only for some project phases. Second, while some projects report on the *recorded* effort, some other projects report on the *estimated* effort based, e.g., on the amount of functionality. To ensure validity of our study we solely consider projects that report on the actual recorded effort. Furthermore, we focus only projects that record the effort in staff hours (as opposed, e.g., "productive time") and that only record time spent on software development, including project management and project administration, but excluding non-project specific supporting activities such as control and audit or hardware support. Overall, excluding projects that report the effort figures only for some project phases, that report estimated rather than recorded effort, that report recorded effort expressed in some other unit than staff hours, or that include the effort dedicated to non-project specific supporting activities, reduces the number of considered projects to 1661.

Finally, as the data in the ISBSG repository is provided by the project owners themselves, it might become polluted by imprecise or unreliable values. Therefore, ISBSG quality reviewers assess the soundness of the data submitted. The result of the assessment ranges from "A" to "D", where "A" indicates that the data "was assessed as being sound with nothing being identified that might affect its integrity", while for "B" the data "appears sound but some aspects might have affected the data or count integrity". Assessment "C" indicates impossibility of assessment due to incompleteness of the data provided, while little credibility should be given to data assessed "D". Restricting our attention to the "A"-projects reduces the number of the eligible projects to 84, while 1609 projects remain if both "A"- and "B"-projects are considered. Hereafter we consider both "A"- and "B"-projects.

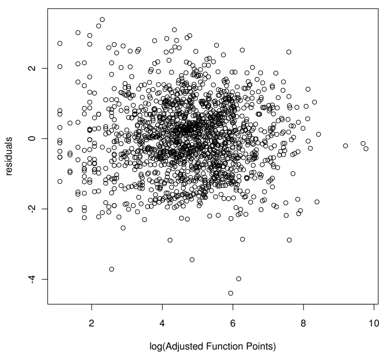


As mentioned, for each project the ISBSG repository contains information about 118 different project attributes including the summary work effort, functional size (measured using the so called unadjusted and adjusted function points counts), as well as organizational, technical and problem-specific attributes. We prefer adjusted function points count to unadjusted function points count (cf. [3]), and consider the following project attributes: primary programming language, language type, organization type, intended market, year of project, development type, platform, and architecture.

Plotting the summary work effort against the adjusted function points count reveals presence of four outliers, projects containing more than 5000 function points. Moreover smaller projects up to 500 function points cover the entire range of work effort. The log-scale plot recommended in [4] shows a clear linear relation between the summary work effort (SWE) and the adjusted function points count (AFP). Using linear regression we obtain the following model:

$$\log(\text{SWE}) = 2.92717 + 0.84617 * \log(\text{AFP})$$

The fitted linear model is adequate: F-statistic equals 2003 on 1 and 1607 degrees of freedom with the corresponding p-value not exceeding 2.2×10^{-16}, and p-values both for the intercept and for the coefficient do not exceed 2.2×10^{-16} as well.



We would like to get a better insight into the distribution of the residuals, i.e., we would like to explain their diversity by investigating to what extent they can be explained using one of the remaining project attributes such as the primary programming language or the intended market. We carry out the explanation step by means of econometric inequality indices, recently applied in the context of software engineering [5]. Due to the nature of residuals, the chosen inequality index should be applicable to negative as well as positive values. Moreover, to calculate the explanation percentage [6], the index should be *decomposable*, i.e., representable as $I^{between}(G) + I^{within}(G) = I$ for any partition G into mutually exclusive and completely exhaustive groups. From all the inequality indices studied in [7], there is only one that satisfies the requirements of decomposability and applicability to the negative numbers, namely the Kolm index [8]. Explanation percentages are shown in the column "With NA" below.

Project attribute	Explanation %	
	With NA N = 1609	Without NA N = 151
Primary programming language	16.11	25.37
Organization type	18.36	17.59
Year of project	5.41	10.88
Development platform	5.05	5.43
Architecture	3.35	8.68
Intended market	1.57	4.61
Language type	1.28	2.45
Development type	0.07	0.05

The table data clearly indicates that such attributes as the primary programming language, the organization type, and the year of the project explain a higher share of the inequality in the residual values than language type, intended market or development type. Language type is a type of the programming language, e.g., 3GL, 4GL or an Application Generator. Since many primary programming languages belong to one language type, and one programming language can belong solely to one language type, the language type induces a more coarse grained partition of the projects considered. Therefore, the explanation value of the language type is lower than that of the primary programming language [5]. High explanation values related to the organization type are caused by association of different organization types to the same project, e.g., "Wholesale & Retail Trade" *and* "Financial, Property & Business Services". Since explanation provided by the inequality indices is applicable solely to mutually exclusive decompositions, we had to introduce a very fine-grained partitioning, including a group containing only projects associated with both "Wholesale & Retail Trade" *and* "Financial, Property & Business Services". Extending inequality indices to non-mutually exclusive groups is considered as future work. The high explanation percentage obtained for the year of the project corroborates the earlier findings of [3] that stress the importance of the project age in effort estimation.

One of the main issues arising when analyzing the ISBSG data, recognized already in [4], is related to presence of missing values. Indeed, since the ISBSG data is based on self-reporting, many project aspects are not being reported. In particular, this would mean that that all projects with unreported value for, e.g., development type, would be put together in the same group. To evaluate the impact of missing values on the explanation percentages we have eliminated all the projects having a missing value in at least one of the project attributes considered, and recalculated the Kolm indices based on the remaining 151 projects. These values are present in the "Without NA" column. Overall, the explanation percentages are higher, which may be explained by the decrease in the number of projects, and, therefore, by a more fine grained partition induced by the same project attributes. We see that the primary programming language, the organization type, and the year of the project still provide high explanation values.

To conclude, in this paper we have applied econometric inequality indices to study how different project attributes can explain diversity of the residuals of the logarithm of the summary work effort with respect to the logarithm of the adjusted function points, i.e., how different project attributes can explain why projects with similar amount of functionality require different development effort.

References

[1] *ISBSG Development & Enhancement Repository, release 11*, International Software Benchmarking Standards Group, 2009.

[2] *Function Point Counting Practices Manual. Release 4.2*, International Function Point Users Group, 2004.

[3] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan, "An empirical study of maintenance and development estimation accuracy," *Journal of Systems and Software*, vol. 64, no. 1, pp. 57–77, 2002.

[4] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in *PROMISE'09*. ACM, 2009, pp. 4:1–4:5.

[5] A. Serebrenik and M. G. J. van den Brand, "Theil index for aggregation of software metrics values," in *ICSM'10*, 2010, pp. 1–9.

[6] F. A. Cowell and S. P. Jenkins, "How much inequality can we explain? a methodology and an application to the United States," *Economic Journal*, vol. 105, no. 429, pp. 421–430, March 1995.

[7] B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics," in *ICSM'11*, 2011, pp. 313–322.

[8] S.-C. Kolm, "Unequal inequalities I," *Journal of Economic Theory*, vol. 12, no. 3, pp. 416–442, 1976.