(Dis)Economies of Scale in Business Software Systems Development and Enhancement Projects

Beata Czarnacka-Chrobot
Department of Business Informatics, Warsaw School of Economics, Warsaw 02-554, Poland

Received: August 16, 2011 / Accepted: September 29, 2011 / Published: January 25, 2012.

Abstract: In the software engineering literature, it is commonly believed that economies of scale do not occur in case of software Development and Enhancement Projects (D&EP). Their per-unit cost does not decrease but increase with the growth of such projects product size. Thus this is diseconomies of scale that occur in them. The significance of this phenomenon results from the fact that it is commonly considered to be one of the fundamental objective causes of their low effectiveness. This is of particular significance with regard to Business Software Systems (BSS) D&EP characterized by exceptionally low effectiveness comparing to other software D&EP. Thus the paper aims at answering the following two questions: (1) Do economies of scale really not occur in BSS D&EP? (2) If economies of scale may occur in BSS D&EP, what factors are then promoting them? These issues classify into economics problems of software engineering research and practice.

Key words: (Dis)economies of scale, business software systems development and enhancement projects, software size metrics, functional size measurement, economies of scale factors.

1. Introduction

In the literature on software engineering, it is commonly believed that software Development and Enhancement Projects (D&EP) are characterised by the absence of economies of scale—contrary to the majority of other business undertakings, including, e.g., building engineering projects, where the per-unit cost decreases while productivity increases with the increase of product size, at least to a certain limits of size. Meanwhile in case of software D&EP, per-unit cost does not decrease yet increase (productivity decreases) with the growth of product size, therefore, this is diseconomies of scale that occur in them. This applying even to the smallest product sizes: “Per-unit cost of larger software system is more expensive than analogous cost of a smaller system” [1].

This phenomenon, considered as immanent to software D&EP, appears to be of significance as much as it is regarded as one of the fundamental objective, i.e., resulting from the specificity of such projects as compared with other engineering projects, causes of their low effectiveness. It is of particular importance in case of software D&EP whose products are Business Software Systems (BSS), as their effectiveness is exceptionally low comparing to D&EP delivering other types of software products.

As indicated by the results of the Standish Group analyses success rate for application D&EP has never gone beyond 35% [2]. It means that majority of them either end up with total failure, or they exceed costs and/or time estimated as well as they lack critical functions/features. The Standish Group estimates that now only 32% of application D&EP worldwide turn out successful while products delivered as a result of nearly 45% of them lack on average 32% of the required functions and features, the planned time of product delivery is exceeded by nearly 80% on average.
and the estimated budget by approximate 55% on average. On the other hand, analyses by T.C. Jones plainly indicate that those software D&EP, which are aimed at delivering BSS, have the lowest chance to succeed [3]. The Panorama Consulting Group, when investigating in their 2008 study the effectiveness of ERP (Enterprise Resource Planning) systems projects being accomplished worldwide, revealed that 93% of them were completed after the scheduled time while as many as 68% among them were considerably delayed comparing to the expected completion time [4]. Merely 7% of the surveyed ERP projects were accomplished as planned. Comparison of actual versus planned expenses has revealed that as many as 65% of such projects overran the planned budget. Only 13% of the respondents expressed high satisfaction with the functionality implemented in final product.

Meanwhile BSS are not only one of the fundamental application areas of software engineering, also their development/enhancement often constitutes serious investment undertaking: spending on BSS D&EP may considerably exceed the expense of building offices occupied by companies commissioning such systems, and in extreme cases, even 50-storey skyscraper, roofed football stadium, or cruising ship with a displacement of 70,000 tons [5]. Exceptionally low effectiveness of BSS D&EP as compared to other types of software projects, with their costs being considered, leads to the substantial financial losses, on a worldwide scale estimated to be hundreds of billions of dollars yearly.

Thus the paper aims at finding answers to the following two questions:

1. Is it true that economies of scale do not occur in BSS D&EP, as it is commonly assumed in the software engineering literature, and therefore, can this specific phenomenon be treated as objective cause justifying the low effectiveness in case of such projects?

2. If economies of scale may occur in BSS D&EP, what factors are then promoting them?

The paper is organized as follows: In section 2 the author presents and compares different BSS size measures. In section 3 the selected results of studies proving diseconomies of scale in software D&EP are pointed out and analysed. Section 4 is devoted to the presentation of the studies proving economies of scale in BSS D&EP, while Section 5—to the analysis of main factors promoting economies of scale in such projects. Finally, in section 6, the author draws conclusions and some open lines about further works.

2. Business Software Systems Size Metrics

Whenever investigating into (dis)economies of scale, it is important to weigh the costs of product development/enhancement against the product size expressed in appropriate size units. In economics or other (than software engineering) engineering disciplines the product size unit as a rule is evident whereas answer to the question about product size units in case of software products is not evident at all, and these units are of significance to the results of such investigations. The question then arises: in what units the size of software product, including BSS, is measured?

Basic approaches to the size measurement of every software product may be reduced to perceiving it from the perspective of (see also Ref. [6]):

- Length of programmes, measured by the number of the so-called programming (volume) units. These units most of all include source lines of code. However, these units measure neither size of the software products nor their complexity but only the attribute of “programme length” yet thus far these are them that in practice have been employed most often with regard to the software size [7];

- Software product construction complexity, measured in the so-called construction complexity units. Most of hundreds of such metrics having been proposed are limited to the programme code yet currently these units are used mainly in the form of object points [7]. These points are assigned to the construction elements of software (screens, reports, software modules) depending on the level of their complexity;
• Functionality of software product, expressed in the so-called functionality units. They most of all include Function Points (FP), which are assigned to the functional elements of software (functions and data needed to complete them) depending on the level of their complexity. Moreover, the measure of software functionality is not the so-called adjusted function points, present in some of the methods dedicated to its measurement, which are designed to correct functional user requirements with non-functional requirements.

Synthetic comparison of various software size metrics against a background of key requirements set for them was presented in Table 1.

The right metric of software product size has been sought out for several decades now. Many years’ verification of various approaches’ reliability and objectivity showed that what for now deserves standardization is just the concept of software size measurement based on its functionality—being an attribute of first priority to the client. Due to the empirically confirmed effectiveness of such approach, it was in the last years normalized by the ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission), and turned into the six-part international standard ISO/IEC 14143 for the so-called software Functional Size Measurement (FSM) [8].

Details displayed in Table 1 clearly indicate the reasons why functionality units were recognised as the most appropriate metric of software product size not only by the ISO/IEC but also, among others, by Gartner Group [9] as well as by International Software Benchmarking Standards Group (ISBSG) [10]. They show no limits being characteristic of programming units and construction complexity units—although one may have reservations as to their versatility and relatively high complexity of the methods based on them. However, it is hard to expect that the method of measurement of software products, by nature being complicated, would be effective yet simple.

The set of rules for software FSM enclosed in the ISO/IEC 14143 norm provides key definitions, characteristics and requirements for FSM, and defines Functional Size Measurement Method (FSMM) as a specific FSM implementation defined by a set of rules, which conforms to the mandatory features of such measurement. The first part of this standard defines also indispensable principles upon which the FSMM should be based—fundamental one is the definition of functional size, which is understood as “A size of the software derived by quantifying the functional user requirements”, while “The Functional User Requirements (FUR) represent the user practices and procedures, that the software must perform to fulfill the user’s needs. FUR excludes Quality Requirements and any Technical Requirements” [8].

<table>
<thead>
<tr>
<th>Requirement towards metrics</th>
<th>Programming units</th>
<th>Construction complexity units</th>
<th>Functionality units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unequivocality of definition</td>
<td>Freedom in formulating definitions (differences as big as even 5:1)</td>
<td>Depending on the method</td>
<td>In methods normalized by ISO/IEC</td>
</tr>
<tr>
<td>Possibility to make reliable prognosis on the size relatively early in the life cycle</td>
<td>Possibility to calculate program length only for the existing code</td>
<td>None—with regard to programming units and object points</td>
<td>As early as at the stage of requirements specification</td>
</tr>
<tr>
<td>Base for the reliable evaluation of the all phases work effort</td>
<td>Final programme length does not fully reflect the whole work done</td>
<td>Final software size does not fully reflect the whole work done</td>
<td>Relatively high reliability as early as at the stage of requirements specification</td>
</tr>
<tr>
<td>Software size being independent of the technology employed</td>
<td>Programme length determined by the language employed</td>
<td>Size being dependent on the technology employed</td>
<td>Size depends on functional user requirements</td>
</tr>
<tr>
<td>Measuring size in units being of significance to a client</td>
<td>No significance to a client</td>
<td>Secondary significance to a client</td>
<td>Measurement from the point of view of a client</td>
</tr>
<tr>
<td>Possibility to measure all software categories</td>
<td>Yes</td>
<td>Depending on the method</td>
<td>Depending on the method</td>
</tr>
<tr>
<td>Easiness of use</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Source: Author’s own analysis.
There are about 25 variants of the FSM techniques having been developed, however, only five of them have been now acknowledged by the ISO/IEC as conforming to the rules laid down in the ISO/IEC 14143 norm and certified as international standard, namely: (1) International Function Point Users Group (IFPUG) method approved in the ISO/IEC 20926 standard [11]; (2) Mark II (MkII) function point method proposed by the United Kingdom Software Metrics Association (UKSMA) and normalized in the ISO/IEC 20968 standard [12]; (3) Netherlands Software Metrics Association (NESMA) function point method approved in the ISO/IEC 24570 standard [13]; (4) Common Software Measurement International Consortium (COSMIC) method certified in the ISO/IEC 19761 standard [14], now being revised; and (5) FSM method developed by the Finnish Software Metrics Association (FiSMA) and normalized in the ISO/IEC 29881 standard [15].

The first three methods listed above are accepted by the ISO/IEC not in full versions, as proposed by the organizations developing them, but in part, however in the most important part with respect to the software functional size measurement [8]—that is why they are called the first-generation FSMM. In the approaches proposed by IFPUG, UKSMA and NESMA these methods involve also delineating of the so-called value adjustment factor, which is supposed to adjust functional size, being measured with the use of Unadjusted Function Points (UFP), to the environment of specified project by taking technical and quality requirements into consideration—thus final result is presented in the so-called Adjusted Function Points (AFP) [16]. Yet this part of these methods has not been approved by the ISO and IEC—as these organizations’ assumptions exclude the fact of FSM depending on requirements of this type. On the other hand, the COSMIC and FiSMA methods were recognized as international standard entirely as the phase of adjustment of functional size to non-functional requirements does not exist in them, which means that function points of the COSMIC method and FiSMA method always depict the functional size of software product [8, 15]. That is why these two methods are called the second-generation FSMM.

The FSMM standardized by the ISO/IEC differ in terms of software measurement capabilities with regard to different software classes (functional domains), but all of them are adequate for BSS. Also, these methods provide rational basis for methodologies supporting BSS D&EP scope management (for more details see Ref. [17]).

Summing up, the only metric of software product size being so far recognised as sufficiently objective and reliable are function points, which allow to measure product functional size by using one of the five FSM methods accepted by ISO/IEC—in case of first-generation methods this is unadjusted FP being used for that purpose.


Among exemplary studies indicating diseconomies of scale in software D&EP are those of the following authors: H.D. Knöll and J. Busse [18], L.H. Putnam and W. Meyers [19], M. Cusumano et al. [20], S. McConnell [21], T.C. Jones [22], and B. Boehm et al. [23]. All these analyses prove the occurrence of diseconomies of scale in software D&EP:

- With regard to software product size expressed in programming units [19-21, 23] or in adjusted function points [18, 22];
- Already at the smallest sizes of software product.

Based on data contained in Ref. [18] one may create graph presented in Fig. 1. It plainly indicates that in the case of exemplary categories of software systems the project effort grows exponentially with the increase of software product size expressed in IFPUG adjusted FP. On the basis of data for banking software system (being BSS), presented in Fig. 1, for which the fastest increase of work effort may be noticed, one may, on the other hand, derive dependencies pictured in Figs. 2-3. As it
may be seen, with the increase in the product size of such system being expressed in IFPUG adjusted FP, per-unit effort goes up while productivity decreases, as early as with the smallest values of product size.

The fact of the described dependencies being shaped in similar way was indicated also by the remaining studies mentioned above (as well as by many other studies), including those where programming units were chosen to express software product size.

What is considered to be the main reason for diseconomies of scale to occur in software D&EP is regularity according to which the number of communication channels between project participants \((n)\) increases with the growth of the software system being constructed so that the amount of work increases proportionally to \([n \times (n-1)]/2\) (actual number of communication channels) \([21]\). Other significant factors of diseconomies of scale (so-called scale factors), whose influence depends on software product size (the larger the product, the stronger their influence), include, according to McConnell, the following: immaturity of software processes, inappropriate risk management, project precedentness, poor collaboration between project participants, and inflexible user requirements.

4. Studies Proving Economies of Scale in BSS D&EP

In 2009, the International Software Benchmarking Standards Group in cooperation with COSMIC published results of the studies, which in case of BSS D&EP contradict the commonly adopted thesis on the absence of economies of scale in software D&EP \([24]\). These conclusions apply to BSS being measured in COSMIC function points.

The ISBSG is a non-profit organization that was established in the second half of the 1990s of the last century with the mission to improve processes of software projects execution in business companies and public administration institutions as well as to support development of software organizations \([25]\). This mission is being fulfilled by developing, maintaining and exploiting three kinds of repositories with benchmarking data regarding software product and process measurement. One of them, the largest one, comprises data for software D&EP, the second
one—data for software maintenance and support applications, while the third one—data for software package acquisition and implementation. These repositories, standardized according to the ISO/IEC 15939 norm [26], verified and representative of current technology, are meant to support decision-making process in the area of software engineering. Data in the repository for software D&EP have been obtained from over 5000 projects completed in more than 25 countries. Generally, they concern projects delivering business applications as their products. For all projects the software product size is expressed as functional size of one of the methods recognized by ISO/IEC.

Studies on (dis)economies of scale in BSS D&EP were based on the verified data from nearly 300 of such projects, of which more than half are projects being completed in the banking sector while the remaining ones took place in the sector of public administration, insurance, engineering, health care, and trade. Approximate 50% of the considered projects are BSS new development projects with product size ranging from 10 to 1670 COSMIC FP, with work effort ranging from 200 work-hours to 32 work-years (median equals 3 work-years) and duration of 1 to 65 months (median equals 9 months). The rest are BSS enhancement projects with product size ranging from 3 to 750 COSMIC FP, with work effort ranging from 24 work-hours to over 15 work-years. The results coming from the ISBSG and COSMIC studies are presented in Figs. 4-5.

Charts presented in Figs. 4-5 indicate that economies of scale occur both in BSS new development projects as well as in BSS enhancement projects:

- In BSS new development projects up to the size of 1000 COSMIC FP, then the productivity median decreases (however for product size over 1000 COSMIC FP there were data available for a little number of projects only);
- In BSS enhancement projects at least up to the size of 750 COSMIC FP (data were not available for larger sizes of such projects’ products).

![Fig. 4 Productivity versus software product functional size expressed in COSMIC FP for BSS new development projects. Source: Ref. [24].](image1)

![Fig. 5 Productivity versus software product functional size expressed in COSMIC FP for BSS enhancement projects. Source: Ref. [24].](image2)

What is also interesting is the comparison of dependencies between productivity and product functional size expressed in IFPUG unadjusted FP and COSMIC FP, which may be seen in Fig. 6. As it indicates, in case of both FSMM the productivity increases with the growth of product functional size: in the range of 0 to 1000 COSMIC FP and in the range of 50 to 2000 IFPUG UFP.

5. Factors Promoting Economies of Scale in BSS D&EP

According to the author of this paper, among fundamental factors of economies of scale in BSS D&EP should be first of all mentioned:

- Maturity of organization and BSS D&EP processes;
- Use of sufficiently objective and reliable method of BSS size measurement;
- Undertaking small BSS D&EP.
5.1 Maturity of Organization and BSS D&EP Processes

Criteria for data collection in ISBSG take into account only organizations employing the methods of software FSM. These organizations are considered more mature than the others within the meaning of CMMI (Capability Maturity Model Integration [27])—as they comprise programmes concerning implementation of software measures. The latest version of the CMMI for Development of 2006 is strongly oriented towards measurement of software processes and products. It regards measurement as being fundamental to achieving the next levels of organization maturity: the higher the level, the stronger its orientation towards quantitative approach. Measurement and analysis process area, distinguished explicitly, is ascribed to the second level of organization maturity (managed).

However, H. von Loon estimates that not many software organizations achieve high maturity levels, this being a problem applying to Europe in particular (in the ISBSG repository this is data from the USA, Japan and Australia that dominate [28]). Analyses presented in Ref. [29] prove that what for the companies at initial level of maturity appears to be one of the fundamental causes of difficulties in achieving maturity level 2 is the lack of software measurement procedures, with the metric of software product size being the one skipped the most often.

Meanwhile results of the studies concerning effectiveness of CMMI, carried out not only by the Software Engineering Institute [30], being this model’s author, but also by D. Rico [31], reveal that in organizations at the high level of maturity:

- Costs and time of project completion are estimated more accurately thanks to the appropriate collection of reliable benchmarking data: organizations at maturity level 5 (optimizing) practically do not exceed estimates whereas organizations at maturity level 1 (initial) exceed the estimated time by 150% on average, and the estimated cost by nearly 200% on average;

- Quality of final products increases due to lower number of errors and this being thanks to the control of the intermediate products quality initiated at the earliest phases of project lifecycle;

- Cost of improving bad quality of intermediate products (maintenance cost) decreases due to the lower number of errors: average cost of error correction in organizations at maturity level 5 amounts to approximate 4% of the total software development costs while in organizations at initial level it amounts to over 50% of such costs;

- Total cost of software development decreases as a result of the decrease in the cost of improving bad quality of products: average cost of developing one function is more than 3 times lower in organizations at the highest level comparing to that in organizations at the lowest level;

- Productivity grows as a result of the decrease in the above mentioned per-unit cost;

- Total time of products completion gets reduced due to lower number of errors and therefore reduction of the time designed for correcting them.

5.2 Use of Sufficiently Objective and Reliable Method of BSS Size Measurement

If BSS D&EP product size is not appropriately measured then the results for per-unit cost/per-unit work effort/productivity of such projects will be
incorrect. Analysis presented above indicates that the only methods of software product size measurement recognised as sufficiently objective and reliable are FSMM having been normalised by the ISO/IEC. They have to match with the proper functional domain (see Table 2—as indicated in there, all standardized FSMM are adequate for BSS) and be applied in accordance with the appropriate norms.

What also should be taken into account is that 1 IFPUG UFP does not equal 1 COSMIC FP. Dependencies between these units are only approximate and differ for different types of projects as well as for different kinds and different sizes of product. So there is no possibility of exact conversion of the results of both methods using mathematical formula. One of the approaches towards conversion is conversion based on statistical formula [32], but this issue requires further investigations.

5.3 Undertaking Small BSS D&EP

Fig. 6 indicates that the size of BSS (its fragment) being developed one-off should not exceed certain limit: for BSS new development projects being 1000 COSMIC FP or 2000 IFPUG UFP on average (exemplary estimates for functional sizes of many software products may be found in Ref. [33]). Such approach requires appropriate estimation of product size threshold point, with type of project and software product as well as iterative approach to development/enhancement being taken into account.

Small software products favor early involvement of a little users group in the project activities, lower the dynamics of requirements changes, facilitate precise defining of real, clear goals and expectations of a client as well as responsibility for the product, are being accomplished by small development teams which results in lower effort of interactions between project participants (see section 3) and therefore they are easier to manage.

There are numerous studies proving higher effectiveness of small BSS D&EP. For instance, according to T.C. Jones, the fact that BSS D&EP are characterised by significantly lower effectiveness comparing to D&EP delivering other types of software products, for the most part results from the large size of such projects while small software products D&EP decidedly more often lead to successful end [34]. Moreover, what Standish Group considered to be the major cause of the increase in the effectiveness of application D&EP execution during 1994-2008 by 100% (from the level of 16% to 32%) is minimisation of their size [35]. Eighty per cent of successful projects prove having work costs lower than USD 3 million while projects with work costs below USD 750.000 have 71% of the chance to succeed whereas for projects with work costs ranging from USD 750.000 to USD 3

<table>
<thead>
<tr>
<th>Table 2</th>
<th>The ISO/IEC standards for FSMM versus functional domains.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSMM</td>
<td>Functional domains specified in the norm</td>
</tr>
<tr>
<td>ISO/IEC 20926—IFPUG method (unadjusted FP)</td>
<td>All software classes</td>
</tr>
<tr>
<td>ISO/IEC 20968—UKSMA method</td>
<td>For any type of software provided that the so-called logical transactions may be identified in it (the rules were developed as intended for BSS).</td>
</tr>
<tr>
<td>ISO/IEC 24570—NESMA method</td>
<td>All software classes</td>
</tr>
<tr>
<td>ISO/IEC 19761—COSMIC-FFP method</td>
<td>Data-driven systems, e.g., business applications for banking, insurance, accounting Real-time systems (time-driven systems), e.g., telephone exchange systems, embedded software, operation systems Hybrid solutions combining both of the above, e.g., real-time systems of airline tickets booking.</td>
</tr>
<tr>
<td>ISO/IEC 29881-FiSMA method</td>
<td>All software classes</td>
</tr>
</tbody>
</table>

Source: Author’s own analysis based on Ref. [8].
million the chance is 38% and for those costing more than USD 10 million—0% [2]. This is a consequence of using iterative approach to development, agile approach in particular, on a more frequent basis over time. Thus particular attention should be given to the prioritisation of functions and selection of those being of significance to the fulfilment of project’s goal—since on average only about 20% of functions and features specified are used often and 50% of them are hardly ever or never used at all [36].

6. Concluding Remarks

Answering, based on the analysis carried out in the paper, the two questions posed in the introduction it should be stated that

(1) Economies of scale can occur in BSS D&EP—contrary to the view being common in the software engineering literature, which in case of BSS D&EP questions the importance of diseconomies of scale factor as an objective cause justifying their low effectiveness.

(2) Fundamental factors promoting economies of scale in BSS D&EP include:

- Maturity of organization and BSS D&EP processes;
- Use of sufficiently objective and reliable method of BSS size measurement;
- Undertaking small BSS D&EP.

In addition to the above, it should be mentioned that in the ISBSG and COSMIC studies that were quoted no economies of scale were revealed for real-time and component software D&EP [24].

It also should be pointed out that when analyzing the ISBSG repository with benchmarking data for software D&EP one should keep in mind that data gathered by the ISBSG are representative not for all of such projects but rather tend reflect “better than average” projects, which results from the following facts:

- Criteria for gathering data in the ISBSG repository refer only to these organizations, which are employing FSMM while apparently these organizations are more mature than others as they carry out programmes concerning implementation of software metrics;
- Data to be included to the ISBSG repository are chosen by the organizations themselves—they may choose projects that are typical of them as well as projects characterised by the best attributes;
- The ISBSG repository does not include a good deal of data about really large software D&EP.

Further works should be oriented first of all towards verification of conclusions coming from the analyses of ISBSG and COSMIC—concerning not only BSS D&EP, but also other types of software projects, the analysis of the impact of particular factors on economies of scale in BSS D&EP as well as continuous search for possibilities of BSS D&EP effectiveness growth.

Acknowledgment


References

[6] B. Czarnacka-Chrobot, The economic importance of business software systems size measurement, in:
(Dis)Economies of Scale in Business Software Systems Development and Enhancement Projects


