Journal of Software Engineering and Applications, 2012, 5, 262-269
http://dx.doi.org/10.4236/jsea.2012.54031 Published Online April 2012 (http://www.SciRP.org/journal/jsea)

Rational Cost Estimation of Dedicated Software Systems

Beata Czarnacka-Chrobot

Department of Business Informatics, Warsaw School of Economics, Warsaw, Poland.
Email: bczarn@sgh.waw.pl

Received February 15th, 2012; revised March 17th, 2012; accepted April 15th, 2012

ABSTRACT

Dedicated software systems hold a character of individual solutions that entail particular problems with regard to their cost estimation. Thus for many years now objective and reliable approaches to the cost estimation of such systems have been sought out so that they could provide the possibility to make rational investment decisions concerning those systems. The purpose of this paper is to bring in the approach to cost estimation of dedicated software systems that has been recently growing in global popularity, using it as a background for presenting conclusions resulting from practical verification of author's own model of dedicated software system cost estimation based on that approach.

Keywords: Dedicated Software Systems; Cost Estimation; Functional Size Measurement; Functional Assessment

1. Introduction

In case of dedicated software systems, having character of individual solutions, particular difficulties with reliable and objective cost estimation come about. Given that this is costs for ready-made software systems (so-called commercial off-the-shelf—COTS) being planned then, similarly as with hardware spending, their estimation usually does not cause any serious difficulties since the client may base it on the market prices (often being negotiable). On the other hand, in case of dedicated software systems, investment costs and time needed for their execution depend most of all on the work effort having been spent on activities necessary to develop new system from scratch or to enhance the existing one. Meanwhile objective and reliable estimation of the effort of development activities continues to be a considerable challenge to the software engineering.

Accurate estimation of work effort is even of greater importance as projects aimed at the development of the considered systems often constitute serious investment undertakings whose costs are comparable even with those of building a 50-storey skyscraper, roofed football stadium, or cruising ship with a displacement of 70.000 tons [1]. It often happens that the client spends those sums without supporting his decision on getting engaged in such investment by proper analysis of the costs, based on rational, sufficiently objective and reliable basis. Thus for many years now objective and reliable approaches to the cost estimation of such systems have been sought out so that they could provide the possibility to make rational investment decisions concerning those systems.

Analysis of the usage of particular work effort estimation methods by Polish dedicated software systems providers, carried out by the author of this paper, has revealed that in most cases they employ the so-called "price-to-win" technique, which mostly is a result of the fact of this technique being commonly used when delivering such systems for the needs of public administration institutions—due to such clients preferring, as a result of legal regulations, the cheapest offers [2]. Client's decision based on such approach can hardly be regarded as rational: for "price-to-win" estimation boils down to deliberate underestimating of project's total costs, which as a rule leads to considerable overrun of the estimated costs or to the situation where product's functions and features are being adapted to the lowered costs thus being far from meeting client's requirements. This is because provider makes cost estimation of the system having in mind nothing but wining the contract and not basing it on objective and reliable approach, at the same time leaving aside product's attributes deciding on its effort.

In the practice of the execution of dedicated software systems one may also find other approaches to the product pricing, namely: fixed price contracts and time and material contracts. In the former case, price of the project product is calculated based on the fixed costs, which were previously agreed following the requirements specification, and as a rule were estimated on the basis of resources or activities. This means that client pays *de facto* not for the actual functionality delivered in the product but for activities/resources expended by the provider. In contracts of the latter type calculation of the price for a product is based on the fixed rate for work hours expended by product provider. This means that the work effort unit cost is measured with regard to the work time unit, and therefore this is work time that determines the total work costs. Thus in this case client too does not pay for the actual functionality delivered in the product but for the provider's work time instead. Project execution with *ex post* pricing of actually delivered product is still rare, at least in Polish conditions, where we deal with low (however growing) level of the so called "culture of measurement" in software engineering [2].

These approaches, however, too are promoting overrunning of budget designed for delivering of product that would meet client's requirements. In case of client-provider contracts based on hourly work rate of developers the provider can deliberately extend the time of product execution—the Standish Group studies indicate that currently an average overrun of the project execution time among projects that ended with the partial failure (so-called "challenged projects") is approx. 80% [3]. Also, there is no guarantee that provider would deliver product of required functions and features—according to the Standish Group, products delivered as a result of projects ended with partial failure lack on average approx. 35% of functions and features [3]. In case of fixed price contracts, apart from likely situation where the actually delivered software system may be of smaller functionality than the required one, there is also another problem that comes up: providers manifest strong resistance to any extension of requirements, being so characteristic of the discussed systems, often not including, in an objective and reliable way, the influence of changes in client's requirements on the project costs.

Thus one may formulate a hypothesis that these approaches to the dedicated software systems cost estimation, most often occurring in practice, promote overrunning of budget designed for delivering of product that would meet client's requirements—the Standish Group studies show that currently average cost overrun among projects that ended with partial failure is approx. 55% [3,]. Thus other solutions need to be sought out in this specific area.

In view of the above the goal of this paper is to bring in the approach to the cost estimation of the dedicated software systems that has been recently growing in global popularity, using it as a background for presenting conclusions resulting from practical verification of author's own model of dedicated software system cost estimation based on that approach. Thus the structure of the paper is as follows: in Section 2 we are pointing out factors promoting effective estimation of the dedicated software systems projects work effort. Section 3 is focused on explaining the concept and the methods of software system functional size measurement. In Section 4 we are presenting basic assumptions of the software projects functional scope management methodologies while Section 5 comprises conclusions brought about by the verification of the author's own model of functional assessment. Section 6, on the other hand, is dedicated to the brief analysis of unit costs measured with regard to the software system size unit. Then, in Section 7, conclusions from the considerations and solutions presented in the paper will be a crowning touch to it.

2. Factors of Effective Dedicated Software Systems Projects Work Effort Estimation

Analysis carried out by the author of this paper has revealed that the following three factors are of fundamental significance to the accuracy of estimates for the work effort being indispensable to the dedicated software systems execution (see [4-8]):

1) Using estimation method based on the project outcome, *i.e*. product size as this is what for the most part determines work effort—in practice, however, cost estimation is most often made using project inputs (activities/resources) as a base, which leads to even fifteen fold differences in the costs of very similar software systems [9];

2) Expressing project product size in appropriate units;

3) Accessibility and use of appropriate, *i.e*. resulting from the specificity of given development organization, benchmarking data which allow to adjust general estimations to provider's specificity.

3. Dedicated Software Systems Functional Size Measurement

Numerous studies indicate that in the context of cost estimation (and not only that one) the most adequate units of software product size are functionality units (usually function points). What is particularly worth mentioning here are studies commissioned by the State Government of Victoria in Australia which have revealed that the cost estimation of software systems realised for this very institution of public administration based on functionality units results in reducing an average budget overrun (let us remind here that according to the Standish Group analyses it currently amounts to approx. 55%) to less than 10% [10]. Analyses by International Software Benchmarking Standards Group (ISBSG) also confirm the correctness of this approach. They allowed to conclude that projects whose software product was estimated with the use of functionality unit were characterised by accurate estimates: in 90% of the cases cost estimates differed from the real cost by not more than 20% whereas in case of 70% of the projects the budget overrun did not exceed 10% of the actual costs [11]. Also analysis of the results of 25 studies concerning the reliability of the most important dedicated software system projects work effort estimation methods, made by the author on the basis of the subject literature in [6], revealed that currently the highest accuracy of effort estimations is delivered by the parametric extrapolation methods based on software product size expressed in functionality units.

This is just reliability and objectivity of the effort and cost estimates based on software system size expressed in functionality units, having been proven as a result of many years' verification, that in the recent years has led to the standardization of the concept itself as well as the methods of the so-called software Functional Size Measurement (FSM), based on that concept, by ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission). For the size of software system is measured first of all to be used as a basis for estimating project's key attributes: work effort, cost and time of execution.

Set of rules regarding software functional size measurement was included to the 6-part norm ISO/IEC 14143 [12]. For such measurement, this standard proposed definition of functional size of software product, which is understood as ”size of the software derived by quantifying the functional user requirements", while the "functional user requirements here represent the user's practices and procedures that the software must perform to fulfil the user's needs" [12].

The group of methods recognised by the ISO and IEC as conforming to the FSM rules laid down in the ISO/IEC 14143 norm currently includes the following:

- Function point method developed by the IFPUG (International Function Point Users Group) [13]—the most popular among techniques of software functional size measurement.
- MkII (Mark II) function point method developed by the UKSMA (United Kingdom Software Metrics Association) [14]—technique popular in the United Kingdom, offering higher level of measurement detailness comparing to the IFPUG method.
- NESMA (Netherlands Software Metrics Association) function point method [15]—developed in the Netherlands, simplified version of the IFPUG function point method.
- COSMIC (Common Software Measurement International Consortium) function point method [16]—offering different approach to the software functional size measurement in comparison with the above mentioned methods, resulting in higher universality with regard to various categories of software.
- FSM method in the version proposed by FiSMA (Finnish Software Measurement Association) [17]—similar to the COSMIC method, popular mostly in Finland.

The FSM methods accepted by the ISO/IEC differ in terms of software measurement capabilities with regard to different functional domains (software categories). Thus, prior to choosing given method one should first evaluate its adequacy to the type of software, whose functional size is going to be measured.

Popularity of FSM concepts and methods in recent times has been growing dynamically worldwide. It mainly results from their effectiveness, objectivity and reliability, having been proven in practice, especially in case of dedicated software systems projects work effort and costs estimation. Thus, in the sphere of public administration for instance, the IFPUG method is commonly employed in the USA, being a method recommended by the Government Accountability Office (GAO)—an audit institution of the United States Congress—in the execution of systems being under its control. Approach proposed by the COSMIC, on the other hand, has just recently become a national standard in Japan and Spain while at the turn of 2010/2011 it was added by the GAO to the list of methods it is recommending. This method originates in Canada where it is used in public administration sector, e.g. by the National Bank of Canada and Department of National Defence. It is also employed by the European Commission (e.g. Taxation and Customs Union Directorate General). In the United Kingdom, use of the UKSMA method is a formal requirement in case of execution of software system development projects for the needs of public administration above certain size while it is recommended for all projects of this type by the Central Computer and Telecommunications Agency. Similarly, other methods accepted by the ISO/IEC are used in Scandinavia, Italy, Australia and other countries. What's more, these methods are used not only by providers but more and more often they are employed by the clients [18].

As indicated by the author's own studies, also in Poland the interest in FSM methods is growing among providers of dedicated software systems, especially in the context of projects work effort and costs estimation [2]. Although the level of both awareness of those methods as well as using them does not appear impressive yet it is growing whereas overwhelming majority of those familiar with FSM methods are also using them – due to the conviction about their effectiveness, reliability and objectivity.

Among other basic reasons for the increased popularity of functional approach, both in Poland and worldwide, are the following: stronger care about financial means during recession and post-recession time, ever growing competition on the market and increasing market globalization level, ever growing awareness of clients therefore greater requirements concerning providing justification for the project costs and completion time offered by providers, and undoubtedly the acceptance of the FSM concept and a few of such methods by the recognised international organizations for standardization.

4. The SouthernSCOPE and NorthernSCOPE Methodologies

The FSM concept and methods form the foundation for two methodologies of the management of the so-called functional scope of dedicated software system projects, that is: southernSCOPE methodology and northernSCOPE methodology. The southernSCOPE methodology was developed in order to support clients when making estimation for software systems being developed for their needs while what drove its development was not only the enormous spending on products of this type but also the results of the Standish Group studies proving high scale of failure in the execution of projects aimed at developing such products [8]. What was considered the cause of this *status quo* was the fact that in the area of dedicated software system development projects the methods of the measurement of such projects' outcome (product) had not been used only until recently; instead, the product pricing used to be based on futile approach consisting in the measurement of inputs (resources/activities) to that process. According to the authors of the methodology, emergence of the FSM methods has diametrically changed this situation while development of repositories with benchmarking data based on those methods, e.g. the International Software Benchmarking Standards Group repository [19] or Software Productivity Research repository [20], allows for making more and more accurate estimation of such projects' costs on the basis of those methods. They also deliver the new knowledge of comparative character, the result of which is the possibility to ensure more and more effective management of projects scope.

Thus the southernSCOPE methodology was aimed at changing the approach to the project product pricing: what was suggested instead of payment based on fixed price or payment for the time of developers' work, possibly "price-to-win", was the pricing based on unit cost calculated with regard to the product size unit, that is functionality unit (Function Point—FP). After several years of employing such pricing, both in public as well as in private sector, independent study aimed at evaluating the effects of applying this methodology had been carried out [21, p. 26,27,30]. It turned out that each of the analysed projects, having been accomplished with the use of this methodology, had ended successfully, was controllable by a client while the products delivered met clients' goals and expectations. These studies also indicate that using pricing based on software product functional size makes the average budget overrun oscillate around 10% only. Such high effectiveness of this methodology triggered interest also in some European countries, in the USA, and in Japan. In 2007, the FiSMA published the rules for the northernSCOPE methodology, developed on the basis of the southernSCOPE approach and being very similar to that [22]. It is based on somewhat different processes and runs through it in a different order.

Fundamental assumptions of both methodologies include the following:

- price to be paid by client for a software system depends directly on the functional size of product;
- estimates are being derived throughout the entire duration of the project;
- structure of the changes management promotes proper management of introducing client's changes to the requirements;
- person responsible for the project scope management is the so-called scope manager.

The practice shows that the discussed methodologies prove useful in case of projects aimed at developing or enhancing dedicated software systems, regardless of whether or not they have internal or external character. However, conditions of the effective use of those approaches need to be met; these conditions, among others, include the following:

- accomplishment of project within the planned and controlled budget is of key significance, if not a priority, to a client;
- there is an acceptance for the software product FSM methods.

5. The SoftFAM—Conclusions from the Model Verification

Concurrently with the above methodologies the author of this paper proposed and then verified her own Software projects Functional Assessment Model (SoftFAM) designed for the functional assessment of dedicated software system development & enhancement projects. Functional Assessment (FA) of project is understood by the author as its *ex ante* and *ex post* evaluation carried out on the basis of FSM concept and methods. Key attributes (functional attributes) of such assessment include: product functional size, work effort which needs to be spent on product's development/enhancement, and project's functional productivity understood as the ratio of product functional size to the work effort spent on its development/enhancement, or being inversion of functional productivity—work effort necessary to achieve functionality unit, determining work measured with regard to the product functional size unit.

Assumptions for the SoftFAM model were presented in [23]. Due to the limited length of the paper we present here only conclusions coming from the verification of the model (for more details see [24]). They concern:

1) Reducing certain negative practices, common in case of execution of dedicated software system projects, thanks to the rules implemented in the model, what is shown in **Table 1**.

2) Advantages of the SoftFAM over the methodologies: southernSCOPE and northernSCOPE, result from the fact

Table 1. Negative practices in the execution of dedicated software system projects and SoftFAM rules promoting the reduction of such practices.

Item	Negative practices in the execution of dedicated software systems	SoftFAM rules reducing negative practices
1.	Client setting irrational requirements for functional attributes —client does not know what would be the cost of functionality required by him, what promotes making incorrect investment decisions.	Assuming adequate dependencies between functional attributes (on the basis of benchmarking data repositories) and allowed tolerance intervals for those attributes (according to the rules of FSM methods).
2.	Clients extending the required functionality during the project lifecycle (the so-called "scope creep") without effects of such a change being correspondingly reflected in the project execution costs	Monitoring of the effect of each change made to client's functional requirements on the product functional size and therefore on the project execution costs. It promotes introducing of such changes only that are truly necessary, and this being of great significance in the context of the Standish Group studies indicating nearly 50-percent redundancy of client's requirements with regard to product functionality (see below).
3.	Deliberate underpricing of product execution costs by offerors in order to win contract ("price-to-win" approach).	When choosing offers for project execution, preferring the highest allowed functional productivity (the lowest allowed effort/cost per functionality unit) instead of the cheapest offers.
4.	Provider delivering product of lower functionality than that required within fixed price contracts.	Cost estimation of project product based on the required (*ex ante* pricing) and actually delivered (*ex post* pricing) product functional size as well as work cost per functionality unit (basic criterion of choosing project provider), having been mutually and formally agreed at the stage of choosing provider—thanks to that, client is not obliged to pay for the required functionality which had not been delivered, neither he has to pay the price calculated on the basis of activities/resources, including developer's work time. This argument as well as the one above (No. 3) indicates the advantage of the SoftFAM model over commonly employed client-provider contracts.
5.	Provider delivering functionality (many a time also being lower than the required one) at total costs being higher than those expected in case of time and material contracts, which promotes extending of project time, often on purpose.	

Source: author's own analysis.

of SoftFAM using:

1) Allowed tolerance intervals for functional attributes, namely:

- Upper bound of the allowed tolerance interval for functional product size—the need to determine that bound results from redundancy, commonly occurring in practice, of client's requirements as to the product: as indicated by the Standish Group studies, only about 20% of functions and features specified by a client are always used in the implemented software system whereas 45% of them are never (sic!) used in those systems [25].
- Upper bound of the allowed tolerance interval for functional productivity—defining it comes in useful.

When making rational choice of provider's offer, *i.e.* from the point of view of reducing the risk of choosing the offer in which the offered productivity would be determined as not very achievable (overrated) value since such situation would mean that in reality the work effort per functionality unit will be probably exceeded which would entail the risk of delivering product of functional size being lower than the allowed one as provider would be probably aiming at not overrunning the offered effort.

- Lower bound of the allowed tolerance interval for work effort—minimum allowed work effort should not be lower than the work effort, resulting from benchmarking data, enabling for delivery of minimum required functional product size.

2) Two stages of making estimates—one made in order to enable client to make rational investment decision as early in the project lifecycle as possible, and second to make right choice of project provider.

What's more, the proposed model has a modular character—next to the full variant there are also 5 simplified variants while the two simplest variants are the closest to southernSCOPE and northernSCOPE methodologies.

3) Possibilities offered by functional assessment, not being met by the assessment of project economic efficiency—and conversely. Since the evaluation of economic efficiency does not allow for the evaluation of project effectiveness, neither it contributes to the limiting of negative practices, displayed in **Table 1**, which are commonly occurring in the execution of the discussed projects thus lowering the chance for achieving planned efficiency. Functional assessment, on the other hand, does not allow to state absolutely *whether* given execution variant is economically efficient but only *when* it is going to be economically efficient. Hence both assessments should complement each other thus ensuring that the two measurable criteria of rational investment decision, that is efficiency and effectiveness, are being met (for more details see [26]).

Summing up, verification of the SoftFAM has revealed that employing it reduces the risk of:

- execution of ineffective and inefficient project;
- choosing wrong project provider;

- redundancy of client's requirements as to the product;
- underestimated *ex ante* pricing of product;
- overestimated *ex post* pricing of product.

Thus functional assessment undoubtedly constitutes appropriate yet not necessarily sole basis for making rational investment decisions by clients commissioning the following categories of projects:

- consisting in constructing dedicated software systems from scratch;
- consisting in enhancement the existing dedicated software system;
- consisting in purchasing ready-made software system and tailoring it to the specific needs of given organization;
- aiming at keeping the functioning dedicated software system in readiness to perform necessary functions.

6. Unit Cost of the Dedicated Software Systems

Unit costs of the dedicated software systems are difficult to delineate if a system provider does not have at his disposal his own resources of benchmarking data on the basis of which he would be able to determine his own (organizational) unit costs with regard to the functional size unit, e.g. 1 IFPUG FP. This results from the fact that they depend on a number of specific factors—including, on a general level, most of all differentiated, with regard to the country, work costs but also type of project, type of software system, field where it is going to be applied and technological environment of the execution as well as many other factors having an effect on wide differentiation of development teams' work productivity.

However it should be pointed out that the adequate resources of own benchmarking data are owned by a relatively few development organizations since the condition to possess them is not only effective implementation of measurement programmes, which in its own right is not very common phenomenon, but collecting of such data for a relatively large number of similar projects that had been executed in the past and, additionally, relating them to the appropriate units of software system size. Such situation is even more typical of Poland where FSM methods, including IFPUG FP, have been employed for a relatively short time. This is when the usefulness of repositories with general benchmarking data, offered by organizations such as e.g. ISBSG, comes into view.

ISBSG is a *non-profit* organization established in the second half of the 1990s, with a goal to enhance processes of IT management both in business entities as well as in the institutions of public administration. This task is being accomplished by maintaining, developing and exploiting several repositories with benchmarking data. One of those repositories, the largest one (its current version containing data concerning over 5600 projects from 29 countries), comprises data for software development and enhancement projects [19]. It is normalized in accordance with the ISO/IEC 15939 standard [27], verified, and representative of current technology. Hence the ISBSG data are recognized in the IT industry while general conclusions coming from their analysis are consistent with the conclusions from the analysis of other organizations' benchmarking data repositories.

Based on the data it has gathered the ISBSG proposes to adopt the following general practical rules for the dedicated software systems projects' unit costs [28]:

1) cost for 1 IFPUG FP oscillates between 300 USD to 1000 USD, with the average of about 750 USD per 1 FP;

2) cost for 1 work hour oscillates between 60 USD to 105 USD, with the average being about 80 USD per 1 hour.

What's more, the ISBSG data indicate that the median of Project Delivery Rate (PDR), that is mean value of the number of person-hours necessary to deliver 1 IFPUG FP, ranges from about 8 to about 11 person-hours per 1 FP – this depending mostly on the type of project, type of software system, application field and technology. Besides, PDR is significantly higher in case of projects consisting in modification/enhancement of software system than in case of projects consisting in developing such systems from scratch [29].

7. Concluding Remarks

The goal of this paper was to bring in the approach to the cost estimation of the dedicated software systems that has been recently growing in global popularity, using it as a background for bringing up conclusions resulting from practical verification of the author's own model of functional assessment (SoftFAM), designed, among others, for the pricing of systems of this type based on that approach.

Considerations featured in the paper lead to the following conclusions:

1) Approaches to the dedicated software system cost estimation most often occurring in practice promote the overrun of costs designed for delivering of product meeting client's requirements. Thus new solutions need to be sought out in this very area.

2) Cost estimation of dedicated software systems should be based not on the lowest total costs offered by potential providers, often being purposely underestimated ("price-to-win" technique) or estimated in a non-objective and/or unreliable manner (on the basis of activities/resources) but on the lowest unit cost instead, which should be measured not with regard to developers' work time but with regard to the product size unit. Such approach to the pricing requires appropriate measure of software size. However not before that the product pricing acquires objective and reliable character—since cli-

ent gets possibility to plan the costs of project execution depending on the expected outcome and as a result of its execution he will pay for the actually delivered product size and not for his requirements not having been realized by a provider.

3) The studies indicate that the most appropriate unit of software system size, especially in the context of its cost estimation, is functionality unit. This has been proved, among others, by standardization—on the basis of thorough verification—of the concept and some methods of the so-called software functional size measurement, carried out by ISO and IEC. So far functionality unit has been the only measure of software product size formally accepted. Thus it is being more and more often used worldwide in the cost estimation of dedicated software systems—at least as a recommendation, if not formal requirement.

4) Functional size of software system constitutes basis of the southernSCOPE and northenSCOPE methodologies, designed for the management of the functional project scope including, among others, their products' pricing. However those methodologies do not take into account two significant assumptions that were adopted in the SoftAM proposed by the author: 1) the need to apply upper bounds of the allowed tolerance intervals for the required product functional size and functional productivity and lower bounds for work effort; 2) the need to employ at least two stages of estimation—first one for making rational investment decision as early in the project lifecycle as possible while second stage—in order to choose suitable product provider. Therefore, comparing to these methodologies, use of the full SoftFAM reduces the risk of choosing inappropriate provider and the risk of lowered product pricing and consequently, it reduces the chance of failing to deliver required functionality and/or to deliver product of inadequate quality. On the other hand, modular character of SoftFAM enables for choosing its version being most suitable to a given situation—it may be a version based on the simplest criteria, closest to the southernSCOPE and northernSCOPE methodologies.

5) Results of the proposed SoftFAM verification prove that it allows for rationalization of specific practical actions as well as business decisions being made on the basis of its criteria. What's more, they also indicate that formal software system pricing should base on the required size (*ex ante* pricing) and actually delivered size (*ex post* pricing) of this product expressed with the use of adequate units as well as on the work cost per unit measured with regard to the product size unit. Thanks to the mentioned options the SoftFAM promotes reducing of some negative phenomena commonly occurring in the Polish practice of such projects execution, having disadvantageous influence on the effectiveness.

The above presented data for unit cost calculated with regard to 1 IFPUG FP vary considerably—as this is not possible to derive accurate values in isolation from the specificity of a given development organization. There are many factors having influence on that very cost. However, lack of own (organizational) resources of adequate benchmarking data continues to be common situation, not only in Poland but worldwide as well. Hence the necessity to use general data.

REFERENCES

[1] C. Jones, "Software Project Management in the Twenty-first Century," Software Productivity Research, Burlington, 1999.

[2] B. Czarnacka-Chrobot, "Analysis of the Functional Size Measurement Methods Usage by Polish Business Software Systems Providers," *Lecture Notes in Computer Science*: *Software Process and Product Measurement*, Vol. 5891, 2009, pp. 17-34. doi:10.1007/978-3-642-05415-0_2

[3] Standish Group, "Chaos Summary 2009," West Yarmouth, Massachusetts, 2009.

[4] B. Czarnacka-Chrobot, "Factors of Effective Business Software Systems Development and Enhancement Projects Work Effort Estimation," *Proceedings of the* 8*th International Conference on Software and Data Engineering* (*ICSDE* 2012), Dubai, 29-31 January 2012, pp. 736-743.

[5] B. Czarnacka-Chrobot, "The Economic Importance of Business Software Systems Development and Enhancement Projects Functional Assessment," *International Journal on Advances in Systems and Measurements*, Vol. 4, No. 1&2, 2011, pp. 135-146.

[6] B. Czarnacka-Chrobot, "Reliability of the BSS Development and Enhancement Projects Effort Estimation Methods," In: J. Sobieska-Karpinska, Ed., *Business Informatics in Management*, Wroclaw University, Wroclaw, 2010, pp. 163-176.

[7] B. Czarnacka-Chrobot, "The Effectiveness of Business Software Systems Functional Size Measurement," *Proceedings of the* 6*th International Multi-Conference on Computing in the Global Information Technology* (*ICCGI* 2011), Luxemburg City, 19-24 June 2011, pp. 63-71.

[8] B. Czarnacka-Chrobot, "The Role of Benchmarking Data in the Software Development and Enhancement Projects Effort Planning," In: H. Fujita and V. Marik, Eds., *New Trends in Software Methodologies*, *Tools and Techniques*, IOS Press, Amsterdam, 2009, pp. 106-127.

[9] State Government of Victoria, "SouthernSCOPE, Reference Manual," Government of Victoria, Melbourne, 2000.

[10] T. Wright, "Report on the SCUD Methodology Review," Sage Technology, 2000.

[11] International Software Benchmarking Standards Group, "The ISBSG Report: Software Project Estimates—How Accurate Are they?" ISBSG, Hawthorn, 2005.

[12] ISO/IEC 14143, "Information Technology—Software Measurement—Functional Size Measurement—Parts 1-6," IS-

O, Geneva, 2011.

[13] ISO/IEC 20926, "Software and Systems Engineering—Software Measurement—IFPUG Functional Size Measurement Method 2009," ISO, Geneva, 2009.

[14] ISO/IEC 20968, "Software Engineering—Mk II Function Point Analysis—Counting Practices Manual," ISO, Geneva, 2002.

[15] ISO/IEC 24570, "Software Engineering—NESMA Functional Size Measurement Method Version 2.1—Definitions and Counting Guidelines for the Application of Function Point Analysis," ISO, Geneva, 2005.

[16] ISO/IEC 19761, "Software Engineering—COSMIC: A Functional Size Measurement Method," 2nd Edition, ISO, Geneva, 2011.

[17] ISO/IEC 29881, "Information Technology—Software and Systems Engineering—FiSMA 1.1 Functional Size Measurement Method," ISO, Geneva, 2010.

[18] G. Rule, "The Most Common Functional Size Measurement (FSM) Methods Compared," Software Measurement Services, Edenbrige, 2010.

[19] International Software Benchmarking Standards Group, "Data Demographics—Release 11," ISBSG, Hawthorn, 2009.

[20] C. Jones, "Software Estimating Rules of Thumb," 3rd Edition, Software Productivity Research, Cambridge, 2007.

[21] P. R. Hill, "Some Practical Uses of the ISBSG History Data to Improve Project Management," ISBSG, Hawthorn, 2007.

[22] Finnish Software Metrics Association, "NothernSCOPE—Customer-Driven Scope Control for ICT Projects," FiSMA, 2007.

[23] B. Czarnacka-Chrobot, "Methodologies Supporting the Management of Business Software Systems Development and Enhancement Projects Functional Scope," *Proceedings of the* 9*th International Conference on Software Engineering Research and Practice* (*SERP*'10), Las Vegas, 12-15 July 2010, pp. 566-572.

[24] B. Czarnacka-Chrobot, "Rational Pricing of Business Software Systems on the Basis of Functional Size Measurement: A Case Study from Poland," *Proceedings of the* 7*th Software Measurement European Forum* (*SMEF* 2010), Rome, 10-11 June 2010, pp. 43-58.

[25] Standish Group, "The Chaos Manifesto," West Yarmouth, Massachusetts, 2009.

[26] B. Czarnacka-Chrobot, "Evaluation of Business Software Systems Development and Enhancement Projects Effectiveness and Economic Efficiency on the Basis of Functional Size Measurement," *Proceedings of the* 10*th International Conference on Software Engineering Research and Practice* (*SERP*'11), Las Vegas, 18-21 July 2011, pp. 401-407.

[27] ISO/IEC 15939:2007, "Systems and Software Engineering—Measurement Process," ISO, Geneva, 2007.

[28] International Software Benchmarking Standards Group, "The ISBSG Special Analysis Report: Software Project Costs," ISBSG, Hawthorn, 2005.

[29] Ch. Symons, "The Performance of Real-Time, Business Application and Component Software Projects," COSMIC and ISBSG, 2009, pp. 1-45.