APEM journal
Advances in Production Engineering & Management 7 (2012) 2, 81-92
ISSN 1854-6250 Scientific paper
http://dx.doi.org/10.14743/apem2012.2.132

EVOLUTIONARY MEASUREMENT-ESTIMATION METHOD FOR MICRO, SMALL AND MEDIUM-SIZED ENTERPRISES BASED ON ESTIMATION OBJECTS

Celar, S.*; Vickovic, L.* & Mudnic, E.*
*University of Split, FESB, R. Boskovica 32, 21000 Split, Croatia
E-mail: stipe.celar@fesb.hr

Abstract:
In the early phase of software project (in the analyses or user requirements specification phase) there are little measurable software artefacts as well as little time for measurements. Reliable software size estimation in such early project phase significantly influence estimation reliability of other project parameters (like effort, duration or cost) and overall project success rate. This paper presents evolutionary method for estimation objects development based on historical measurement data. Estimation objects, recognized from the measurement data in three steps, are classified according their attributes into hierarchy dimensions. Estimation objects are enterprise native and easy for use in early software size estimation. Also, in the paper case study results from the SME applying Function Points for Functional Size Measurement (FSM) are described but the method is independent of FSM or functional size unit.

Key Words: Functional Size Measurement (FSM), Early Software Size Estimation, Function Point, Estimation Objects, Small and Medium-sized Enterprise (SME)

1. INTRODUCTION

We all witness big changes and improvements that software and the whole information and communication technology (ICT) undergone in the last few decades. However, the successful rate of software projects does not follow that rate [1], [2], [3], [4]. Results in Table I show that software projects for almost ten years do not show clear improvements despite the progress of management practices, software development methods, process improvement models and quality standards [5], [6], [7], [8], [9], [10].

Table I: Project success rate (CHAOS report)) [2].

	'94 %	*'96 %*	*'98 %*	*'00 %*	*'02 %*	*'04 %*	*'06 %*	*'09 %*
Successful	16	27	26	28	34	29	35	32
Challenged	53	33	46	49	51	53	46	44
Failed	31	40	28	23	15	18	19	24

Inaccurate estimates of software development effort or project development cost are frequently reported causes of ICT-project failures [11], [12], [13], [14]. Project effort and cost estimation are perhaps the two most crucial issues that a project manager has to make. Majority of the researchers in this field use terms “cost” and “effort” interchangeably [14].

Project managers have to estimate costs and allocate resources to the system development before it is built. Software cost estimation involves the estimation of one or more of the following variables:

- effort (usually in person-months)
- project duration (in calendar time)
- cost (in $ or €).

The above variables are related through the productivity of Software Production Unit (SPU) that depends on broad number of characteristics of development process and technology, SPU, business process. Some authors mention from several dozen to more than one hundred of different characteristics [7], [10], [15], [16].

The software size is the most important factor that affects the software cost. In the last three decades the software size, e.g. system functionality become more complex and therefore hard to estimate and measure. During that period methods for software functional size measurement were also developed. In fact there are well defined, explored and standardized methods for functional size measuring (like line of code – LOC, function point – FP, use case point – UCP). Further, there are wide spectrum of well described estimation methods – Direct Estimation Methods and Derived Estimation Methods [16], [17].

However, in practice there are some big obstacles for wider and more efficient use of Functionality Size Measurement (FSM) and estimation methods. They could be summarized in the following way:

- It is hard to measure something that is still in development and do not exist fully [17]. It can only be measured based on development documentation or software product itself and such measurement could be time-consuming and expensive [16].
- For efficient application of measurement and estimation techniques the micro and small software development enterprises should improve their overall maturity grade or use adequate software quality models [3].
- In the time of constant market pressure an agile movement become a dominant development paradigm and SPUs are forced to measure and estimate early in the software life cycle and with fewer requirements.

Today, with ISO, CMM, CMMI and other software quality assurance (SQA) models most of micro and small-medium enterprises (or their SPUs) struggle with software quality assurance tasks [18]. They use methods like Program and Evaluation Review Technique (PERT) and earned value to track progress. The problem is with the imprecision and inaccuracy of most software project plans [11].

This paper proposes a small contribution for adoption measurement and estimation methods in micro, small and medium-sized enterprises (MEs and SMEs) environment based on following premises:

- enterprise native SW-objects – estimation objects are produced in enterprise's ordinary development process and all people are familiar with
- independency of Functional Size Unit (FSU) – functionality size of estimation objects can be measured/estimated in any unit.

The remainder of the paper is organized as follows: Section 2 describes measurement and estimation challenges that micro, small and medium-sized software enterprises are confronting with. Brief overview of software size measurement, estimation metrics and methods is given in Section 3. Section 4 introduces proposed estimation method together with the case study in medium-sized software development enterprise. Section 5 concludes the paper.

2. MEASUREMENT AND ESTIMATION PROBLEMS IN MEs AND SMEs

Micro, small and medium-sized enterprises play a central role in the European economy. They are a major source of entrepreneurial skills, innovation and employment and account for a large proportion of Europe's economic and professional activity. In practice, 99.8 % of businesses in the European Union are SMEs. Between 2002 and 2008, the number of SMEs increased by 24 million (or 13%) whereas the number of large enterprise increased by only 2000 (or 5%). The growth was also reflected in employment figures. In absolute numbers 9.4 million jobs were created in the SME-sector between 2002 and 2008 [19].

Table II: Enterprises in Croatia, December 2011 (SW, ICT and general) [20], [21].

	Nr. of enterprises	Number of employees					
	Total	0	1 – 9	10 – 49	50 – 249	250 – 499	500 and more
ALL activities	**126.264**	**55.682**	**55.955**	**10.948**	**3.092**	**338**	**249**
%	100%	44%	44%	9%	2%	0,3%	0,2%
J - Information and communication	**4.867**	**2.145**	**2.242**	**408**	**59**	**8**	**5**
%	100%	44%	46%	8%	1%	0,01%	0,004%
SW enterprises (J 62 – Comp. programming)	**2.755**	**1.333**	**1.231**	**172**	**18**	**1**	**0**
%	100%	48%	45%	6%	1%	0,001%	0%
SW enterprises in ICT sector (%)	57%	27%	25%	4%	0,37%	0%	0%
SW enterprises in general economy (%)	2%	1%	1%	0,14%	0,01%	0,001%	0%

Data about local Croatian and broader USA market (Table II and Table III confirm the above statements. SW enterprises are mainly Micro Enterprises (ME), with 10-20 employees. Survey conducted on 150 software SMEs shows that majority of them very rarely recognised and used basic quality assurance concepts from the internationally recognised quality standards, including ISO and Capability Maturity Model Integration (CMMI) [18].

As an answer on constant market pressure an agile movement becomes a dominant development paradigm during last two decades. In their agility many SPUs reduce significantly time for planning, analysis and design [8]. Further, reducing documentation makes difficulties for applying FSM techniques based on FP that presumed user software requirements specification (SRS) and design documentation. Therefore it is hard to expect that SPUs, especially SMEs, devote enough time to adopt and use relative complex estimation and measurement techniques. In such environment it is hard to make reliable estimation, especially in early project phase, i.e. before design specification or some prototype are done [22], [23]. In fact, estimation failure can be huge [9], [10], [22], [24]. Because of nonexistent measurable artefacts it is not even possible to apply measurement methods as a mean to get reliable estimation. An estimation uncertainty became smaller as project progresses but in the same time its importance decreases significantly (Figure 1).

From those reasons we proposed the method for creating the estimation objects from the enterprise's historical data. From another side, reliable software size estimation or measurement is only the first step in reliable project cost and effort estimation [3], [4], [14], [25]. It is reason we propose evolutionary three-phased approach for creating native estimating objects and Functional Size Units (FSU) for MEs and SMEs.

Table III: ICT enterprises in USA, December 2011 (Number of Employees) [26].

Nr. of enterprises/employees	1 – 4	5 – 9	10 – 19	20 – 49	50 – 99	100 –249	250 - 499	500 and more
91.747	**60.999**	**15.907**	**6.345**	**3.890**	**1.686**	**992**	**1.395**	**533**
100 %	66	17	7	4	2	1	2	1

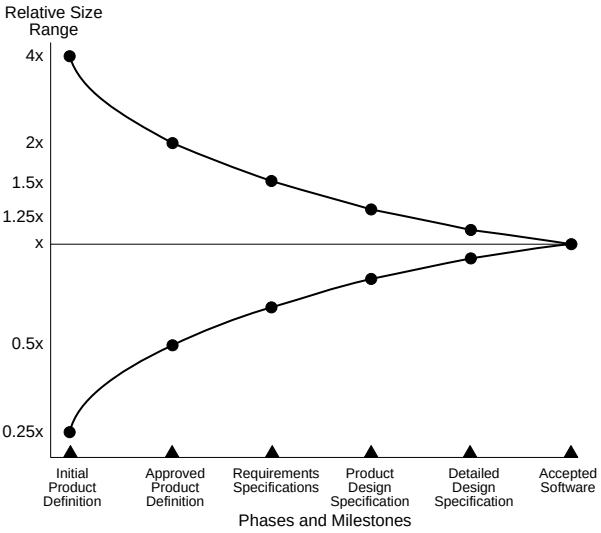

Figure 1: Uncertainty conus [9], [10], [24].

3. SOFTWARE SIZE MEASUREMENT AND ESTIMATION METHODS

Software metrics are measures that are used to quantify software artefacts, software development resources, software development process or software related projects. In software engineering more than 300 metrics have been defined [27]. Some of them are only used for scientific purposes and they don't have practical implementations. Especially for SME estimation methods, or combination of measurements and estimations are tried to be used instead of only measurements. This section gives a short overview of software attributes and methods for their measurement and estimation.

Object-oriented programming languages introduced new measurement objects and measurement attributes. Software complexity described by objects and their attributes can be measured with specialized object-oriented metrics based on use case (Figure 2).

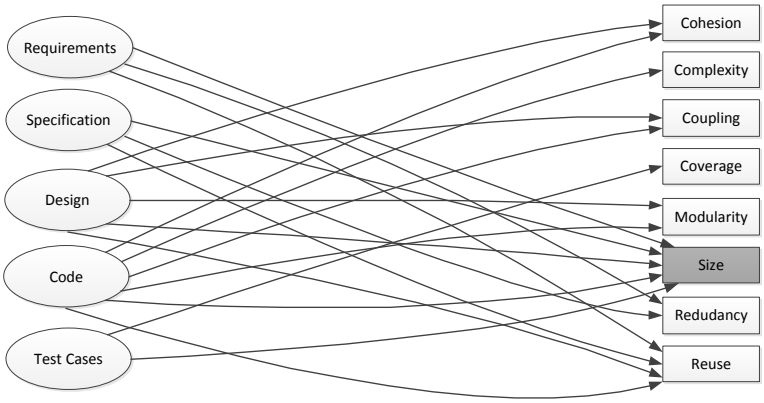

Figure 2: Object-oriented measurement objects and attributes [27].

The most measured and estimated attribute is software size. Software size is often difficult to measure and estimate, especially early in a project (prior to the completion of the requirements specification). Later in a project, certainty of an estimate becomes bigger and less important [9], [10], [28] (Figure 1). Certain methods and metrics used in FSM can impact on the measurement's potential accuracy.

3.1 LOC as functional size unit

Methods for measuring the complexity of software try to identify the measurable software properties and connections between them. Some of the measurable properties are the length, volume, space occupied in memory, the number of lines of code, function points.

The Line of Code (LOC, SLOC, KLOC, KSLOC) measure – a count of the number of machine instructions developed, was the first measure applied to software. Its first documented use, the attempt to formally measure software development productivity was in the 70's [29]. LOC method has some big drawbacks. LOC is dependent on the programming language and code size does not represent the functionality of the program. Accurate LOC is to know only after the project is completed, and therefore it is hard to use this method for estimation (except for expert judgments in analogy methods) [7], [16], [30].

Nevertheless, LOC is still popular functional size unit (FSU) and many enterprises express functionality of their products in Source Lines of Code (SLOC) and convert 'newly' measures into this one. On this way a measurement process can be automated [16] and SLOC counting for completed projects can be done with minimal effort and without subjectivity [29].

3.2 Function Points as functional size unit

In 1979 Allan Albrecht published Function Points metrics for quantifying the amount of business functionality an information system delivers to its users [15], [16]. It is a synthetic metric derived from enumerating five visible, well-defined external characteristics of software based on 5 Base Functional Components (BFCs) [16], [24], [31]:

1. transactions:
   - external inputs (EI) such as logical transaction inputs or system feeds;
   - external outputs (EO) or external inquiries (EQ) such as online displays, reports or feeds to other systems;
2. data entities:
   - internal logical files (ILF) such as logical groups of user defined data and
   - external interface files (EIF) such as interfaces to other systems.

Function Points related very closely to the types of business applications that IBM was developing at the time. For these types of systems, they are a far superior measure of business value than SLOC and can be much better for an organization that develops these types of systems to use for productivity comparison studies.
Function Points metric has its limitations that make it difficult to start using in software enterprises, especially in small and medium-sized ones. The most important limitations are described below.

FP is relatively complex metric, extensive for learning and requires lots of time to master it. Due to lower limits in calculating function point complexity, the lower limit for normal function point calculations is about 15 FPs [16], [32]. Therefore micro function point approach for smaller changes measurement is new challenging approach. Furthermore, FP cannot be extracted automatically from design documents [33].

FP metric is extensible in counting as well. Due to average price of functionality measurement (4 6 $/FP) the FP are not applicable in practice for SW projects with large

functionality (with more than 15000 FP) [16], [32]. Consequently, the measurement automation for projects with large functionality is challenge as well.

As of 2012 there are more than 20 variants of FP metrics with few conversion rules from one method to other [16], [32] and at least five of them (IFPUG, Mk II, COSMIC, NESMA and FiSMA) were confirmed by the International Organisation for Standardisation (ISO). This diversity of variants additionally complicates practical use of FP.

In literature it is possible to find additional function points' drawbacks (e.g. [34]), depending of author's specific appliance area or point of view. However, function points are widely accepted and applied concept, and as such they are a basis for many other measurement and estimation methods.

From the estimation point of view there is an important drawback of FP based methods. That is a necessity to analyze system/software under study or more precisely to decompose it (to recognize *logical transactions*) and make entity and relation model (to recognize *data entities*) in order to identify the Base Functional Components. It is slow and complicated.

3.3 Object-oriented FSMs

Use cases were first introduced by Ivar Jacobson in the mid 80's. They provide a language for describing the requirements of a software system in a way that facilitates communication between developers and the eventual users of the system. Each use case describes a typical interaction that may occur between a user (human operator or other software system) and the software. The focus is on the functions that a user may want to perform or have performed rather than on how the software will actually perform those functions.

Useful and practical FSUs that are oft used for the most measured software attribute are **Use Case Point** and **Story Point** [9], [16], [28]. An Use Case Points method (UCP) were introduced in 1993 by Gustav Karner [29]. Use Case Points count and classify two elements: 1) the actors in the use case and 2) the transactions that are required to make the use case happen. Use case points describe the functionality being delivered rather than the way this functionality is implemented. Unlike Function Points, the Use Case Points can cover a wider spectrum of application types. The problem with using use cases is their lack of standardization across the industry. An organization which has a well defined process for defining use cases could successfully use them for productivity tracking and effort estimation. [16], [29]

A Story Point is usually somewhat larger than a function point perhaps roughly equivalent to two IFPUG function points, or perhaps even more [16]. They are defined within the SPU and are used within that unit to estimate effort and measure productivity and quality.

3.4 Estimation methods

Software size estimation methods can be generally considered as Direct Estimation Methods (or non-algorithmic, Expert Opinion Methods) and Derived Estimation Methods [16], [17].

Direct Estimation Methods (or non-algorithmic) imply the cooperation of one or more experts which directly estimate required elements of the estimation of function points, basing their estimation on experience and intuition.

The basic difference between direct and derived estimation methods is that in the latter, estimation isn't performed directly on function point values, but rather on different project parameters which are somehow related to function point values.

According to the abovementioned categorization of direct and derived estimation methods, one of the most known direct estimation method is the Delphi according to which the predictions of a number of experts are combined. [17] Decomposition Method is also often used method. By decomposing the project into smaller subprojects, it is possible to conduct the evaluation part by part, in detail. Estimations based on analogy, Parkinson's Method and 'Price-to-win' also fall into the category of direct estimation methods.

Derived Estimation Methods or Algorithmic Model Methods provide with the estimation of complexity as a function of more variables which relate to certain attributes of software development project.

During the late 1970s and during the 1980s, some sophisticated algorithmic software estimating models were developed such as Dr. Boehm's Constructive Cost Model (COCOMO), PRICE-S (now True S), SLIM, SEER-SEM, and SPQR/20 (now Knowledge Plan).

Each algorithmic model has the form:

$$Effort = f(x1, x2, ..., xn), \quad (1)$$

where (x1, x2, ..., xn) are the cost factors.

The existing algorithmic methods differ in two aspects:

- in the selection of cost factors and
- in the form of function *f*.

Another important variable relating estimation is the productivity. The relationship between the size of software and the effort required to produce it is named productivity. There are plenty factors influencing the overall software project productivity of SPU – Albrecht mentioned 14 factors [15] while other researchers mention 45 [16] or even more than 100 different productivity factors [7], [10]. These factors have to be taken into account in effort estimation and project planning.

4. PROPOSED METHOD

Software enterprises (or Software Production Unit in software enterprises) can not apply measurement and estimation methods without tailoring them. Learning, tailoring and adopting those methods is not easy process for the ME&SMEs from few reasons:

- FSM and estimation methods itself are quite complex
- lack of trained internal estimators and measurement experts
- internal SME's processes have to be enough mature to implement FSM and estimation concepts
- lack of a database of historical measurement data.

4.1 Evolutionary approach

CMMI maturity level needed for efficient adopting FSM and estimation methods, of at least core processes, is 3-4 and learning time depends on more factors. Majority of software enterprises are in ME&SME category and struggle with software assurance activities. Measurement process implementation is big challenge for them also. That implementation is not simple and can take several years. As a result in selection of metrics and methods it is necessary to align enterprise's or SPU's capabilities with existing metrics and methods. An evolutionary approach, step by step could be useful.

In the first phase called measurement phase as a strategic goal is established application of adequate measurement method. In this phase it is important to:

- create politics of measurements
- identify a person or team to be carrier of measurement politics
- select adequate metrics and methods
- provide internal education and application on real projects
- establish repositories for measurement data

In this phase (Phase A, Figure 3) it is important to recognize objects/entities that could be used for future estimations. It is desirable that objects can be easily recognized in development process and easy connected with measurement data. In addition, metrics closer to large number of developers (not only scientists) have more chances to be widely accepted. For example, UCP is more understandable and simpler for appliance than Unadjusted Function Point (although less researched scientifically).

The objects' granularity can vary from single modules to whole projects, but it is the most relevant to identify those objects that are repeatable in the system (i.e. they can be grouped) but also differ from similar ones by some attribute. Those objects could be for example some class, forms, queries, reports or even some kind of modules.

The next phase is the most intensive phase (Phase B, Figure 3). Measurements and estimations are conduct in parallel. Estimation is done based on measurement data and recognized objects from the Phase A. Strategic goal of this phase is to select and implement adequate estimation metric and method, as well as to establish estimation data repository. Objects recognized from measurement data are classified according their attributes displayed in ontology model (Figure 4) and its **dimension** (n) described in the Table IV. Each dimension consists of some elements (**m**). After classification the objects can be used for software attributes estimation and/or project estimation and compared with measurement data. After reaching these goals, in the Phase B, the SPU can from own classified history data recognise typical produced objects and identify effort/time/cost spent for those objects.

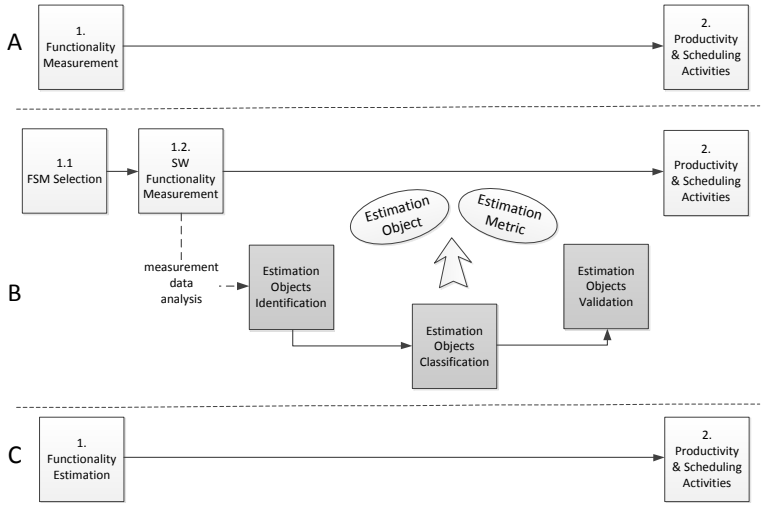

Figure 3: Three-phase measurement-estimation approach.

Some external collection of measured data like ISBSG [35], SPQR and others [16], [35] can be used for benchmarking, but their constraints regarding accurateness must be taken into account [16]. Advantage of internal repositories regarding benchmarking repositories is obvious – internal data and internal objects are native for process participants. As a result process participants can easily accept and use such repositories for estimation.

After estimation objects are classified the estimation itself should be validated on same real projects parallel with measurement process. After validation the measurement activities could be replaced with much simpler and cheaper estimation process. It means, the Phase B is completed and begins the final phase, Phase C (Figure 3).

A maximal number of object instances is calculated according to the following formula:

$$N_{max} = m^n \tag{2}$$

where is **m** maximum among all dimensions.

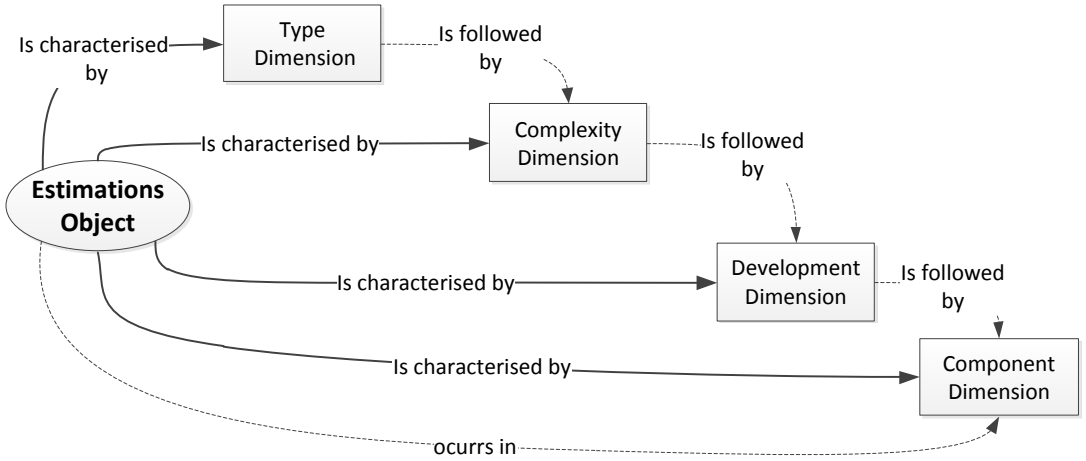

Figure 4: Ontology of m^n object point model.

For a model with four dimension and two elements in each dimension it comprises 16 estimation elements, what could be quite enough if validated.

4.2 Case study

Software Medium-seized Enterprise applied systematically proposed measurement & estimation method. Measurement data are acquired during two years long measurement in development projects. Objects are then classified, validated and used for small and big development projects (Table IV and Figure 5).

An estimation model consists of 4 dimensions with 2 elements at the first three dimensions (Type, Complexity and Development, Figure 5). In the 4^{th} dimensions the model has 4 objects in some branches (Figure 5). The number of estimation objects is 18 (theoretical maximal number would be 4^4). This number of estimation objects has proved as a sufficient for reliable estimates.

Table IV: Characteristics of m^n object point model.

Dimension	**Dimension's instances recommendations**
Object Type	It is advisable to distinguish at least two or three different levels according to the objects SPU produces. Business area specificity could be expressed on this level by adding separate vertical.
Object Complexity	It is advisable to distinguish between two or three different levels relating complexity
Development perspective	It is advisable to distinguish between two or three different levels according to the life cycle. Advisable is to distinguish between: - new development - enhancement - re-development of existing software
Component perspective	This is level for the objects. Business area specificity could be expressed on this level by adding separate objects.

Table V: Characteristics of 18 final software objects – case of medium-sized enterprise.

Object	Description	Measured avg. UFP
MCN-1	NEW COMPLEX MODULE – 1, it is a new module that links more entities and it does not possess any data processing (e.g. item entry or generating)	44.49
MCN-2	NEW COMPLEX MODULE – 2, it is a new module that links more entities and it possesses data processing (e.g. item entry or generating)	116.9
MCN-3	NEW COMPLEX MODULE – 3, it is a new module that possesses complex data processing (e.g. item entry or generating) or it contains more logical units (e.g. write-off record)	318.72
MCO-1	OLD COMPLEX MODULE – 1, it is just a field adding or removal modification on one tab in complex module (e.g. header modification)	12.66
MCO-2	OLD COMPLEX MODULE – 2, it is just a field adding or removal modification on several tabs in complex module (e.g. header and items modification)	31.77
MCO-3	OLD COMPLEX MODULE – 3, it is a modification in one batch in complex module	74.99
MCO-4	OLD COMPLEX MODULE – 4, it is a modification in several batch in complex module	295.86
MSN-1	NEW SIMPLE MODULE – 1, it is a new module in a common process category like preview or simple control	4.02
MSN-2	NEW SIMPLE MODULE – 2, it is a new module that references just one entity. This category also includes complex controls that appear on several places	16.65
MSO	OLD SIMPLE MODULE, is a modification on simple module (type 1 or 2) that keeps module still simple	3.56
RCN-1	NEW COMPLEX REPORT – 1, is a new report that receives data from more entities and does not include any data processing	18.94
RCN-2	NEW COMPLEX REPORT – 2, is a new report that receives data from more entities and includes data processing	32.01
RCO-1	OLD COMPLEX REPORT – 1, is a modification of the complex report that references less than four entities, or adding or deleting of five or less than five fields is performed without any changes on data processing	6.04
RCO-2	OLD COMPLEX REPORT – 2, is a modification of the complex report that references more than four entities, or adding or deleting from five to 15 fields is performed without any changes on data processing	12.75
RCO-3	OLD COMPLEX REPORT – 3, is a modification of the complex report that performs adding or deleting more than 15 fields or just one segment of data processing is changed	30.11
RCO-4	OLD COMPLEX REPORT – 4, is a modification of the complex report that changes more segments of data processing	89.60
RSN	NEW SIMPLE REPORT, is a new report that receives data from simple entity	5.09
RSO	OLD SIMPLE REPORT, it is a modification of simple report that keeps it simple	5.38

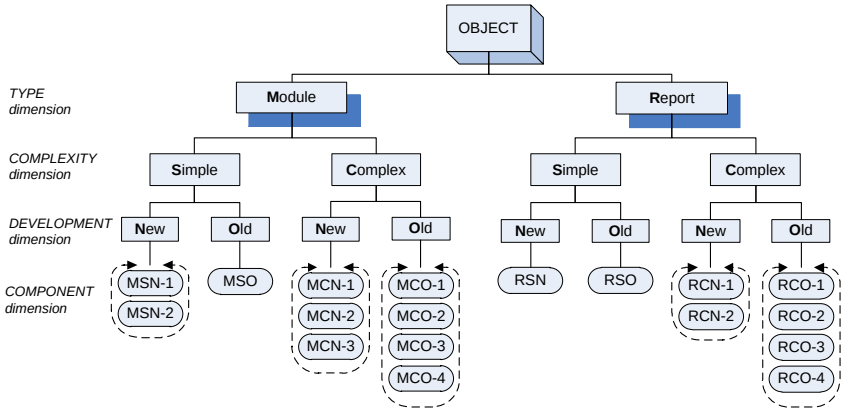

Figure 5: 18 final software objects for estimation – case of medium-sized enterprise.

Additional objects' validation followed in large development project lasting for 13 months and size of delivered software is measured by method Mk II FP Index (V1.3) [36] and it was 7.560

Unadjusted Function Points (UFPs). Software size was estimated in the early development phase using the initial and additional user requirements – estimation was 6.773. Overall estimation error was only 11,6% [37].

According to [10] typical software organizations are struggling to avoid estimates that are incorrect by 100% or more while only sophisticated software organizations can achieve results within ±5% of estimated results instead of within ±10%. Similar also states [32] and [35].

5. CONCLUSION

Adoption of measurement and estimation processes belongs under strategic goals for software enterprise. With reliable early software size estimation method an enterprise can improve overall project success rate significantly. Advantage of the proposed evolutionary measurement&estimation method is its independence of measurement and estimation methods, metrics and functional size units. The characteristics of the method could be briefly summarised:

- measurement data from own historical projects are used for estimation objects creating
- created estimation objects are easy to recognise and use for early software size estimation.

This method is appropriate especially for smaller SPU (micro, small and medium-sized enterprises), but it can also be used by large enterprises.

Drawback of this method is its complexity in establishing of estimation objects from the company's own history measurement data.

Future research will focus on reducing effort in data collection through implementing probabilistic methods like Bayesian network.

REFERENCES

[1] The Standish Group. The Chaos Report (1994). from http://www.standishgroup.com/sample_research/chaos_1994_1.php, accessed on 16-08-2011

[2] Jorge Dominguez (2009). The Curious Case of the CHAOS Report 2009, from http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html, accessed: 2011-11-13.

[3] Jørgensen, M., Gruschke, T.M. (2009). The Impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments, IEEE Transactions on Software Engineering, Vol. 35, no. 3, (2009), pp 368-383, doi: 10.1109/TSE.2009.2

[4] Boehm, B.W., Valerdi, R. (2011). Impact of Software Resource Estimation Research on Practice: A Preliminary Report on Achievements, Synergies, and Challenges, Proceedings International Conference on Software Engineering, May 21-28, 2011, Honolulu, Hawaii, USA, ISBN: 978-1-4503-0744-4, pp. 1057-1065

[5] Abran, A., Moore, J., Bourque, P., Dupuis, R. (2004). SWEBOK: Guide to the Software Engineering Body of Knowledge 2004 Version. IEEE Computer Society, Los Alamitos, California

[6] ANSI/PMI (2008). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fourth Edition. PMI Inc., Newtown Square, Pennsylvania.

[7] Jones, C. (1986). Programming Productivity (Mcgraw-Hill Series in Software Engineering and Technology, Mcgraw-Hill College, 1986, ISBN: 978-0070328112

[8] Boehm, B., Turner, R. (2009). Balancing Agility and Discipline. A Gide for the Perplexed, 7th ed. Boston, USA, Addison-Wesley, 2009, ISBN 0-321-18612-5

[9] Cohn, M. (2005). Agile estimating and planning. New York, USA: Prentice Hall, 2005.

[10] McConnell, S. (2006). Software Estimation: Demystifying the Black Art, WA, USA: Microsoft Press, 2006

[11] Humphrey, W.S. (2005). Why Big Software Projects Fail: The 12 Key Questions, CrossTallk – The Journal of Defense Software Engineering. 18, 3 (2005), pp. 25-29.
[12] Boehm, B.W., Papaccio, P.N. (1988). Understanding and Controlling Software Costs, IEEE Transactions On Software Englneerlng, Vol.14 (10), October 1988
[13] ISBSG (2011). Glossary of Terms for Software Project Development and Enhancement, (version 5.15), International Software Benchmarking Standards Group, Australia
[14] Jørgensen. M., Shepperd M. (2007). A systematic review of software development cost estimation studies (2007), IEEE Transactions on Software Engineering, 33 (1), pp. 33-53.
[15] Albrecht. A.J., Gaffney. J., Jr. (1983). Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Trans. Softw. Eng. Volume SE-9, No. 6, IEEE Press (Nov. 1983), pp. 639-648
[16] Jones, C. (2008). Applied Software Measurement – Global Analysis Of Productivity And Quality, 3rd ed. New York, USA: McGraw-Hill, 2008
[17] Meli, R.; Santillo, L. (1999). Function point estimation methods: A comparative overview. FESMA 9, 1999
[18] Pusatli, O.T., Misra, A. (2011). A discussion on assuring software quality in small and medium software enterprises: An empirical investigation, Technical Gazette, Vol 18 (3), (2011), pp. 447-452, ISSN 1330-3651
[19] European Commission (2010). European SMEs under pressure, Annual report on EU SMEs 2009
[20] Croatian Bureau of Statistics (2011). Number and Structure of Business Entities, September 2011, Y. XLVIII. (11.1.1/3.)
[21] Poslovna Hrvatska (2011). from http://www.poslovna.hr, accessed on 2011-12-27
[22] Santillo, L., Conte, M., Meli, R. (2005) Early & Quick Function Point: Sizing More with Less metrics. 11th IEEE International Software Metrics Symposium (METRICS'05), 2005, p. 41-46.
[23] Malik, A.; Boehm, B. (2011). Quantifying requirements elaboration to improve early software cost estimation, Information Science, Volume 181, Issue 13, 1 July 2011, pp. 2747-2760, doi:10.1016/j.ins.2009.12.002
[24] Boehm, B. et al. (1999). COCOMO II model definition manual. Center for Software Engineering, 1999
[25] Boehm, B.W. (1981). Software Engineering Economics. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981
[26] Manta Inc. (2011), ICT enterprises in USA. from http://www.manta.com/mb_33_G4_000/information_technology?refine_company_emp=E01, accessed on 31-12-2011
[27] Far, B.H. (2011). Software Metrics, from http://www.enel.ucalgary.ca/People/far/Lectures/SENG421/index.html, accessed on 30-11-2011
[28] Kan, S. H. (2008). Metrics and Models in Software Quality Engineering, 2nd ed. Boston, USA: Addison-Wesley Longman, 2008
[29] Minkiewicz, A. (2008). The Evolution of Software Size: A Search for Value, Software TechNews, Volume 11 (3), Data & Analysis Center for Software, 2008, pp. 18-22
[30] Sommerville, I. (2007). Software Engineering, 8th ed. Boston, USA: Addison-Wesley Longman, 2007
[31] ISO/IEC 14143-1:1998 (1998). Information technology – Software measurement – Functional Measurement – Part 1: Definition of Concepts. JTC1/SC 7, ISO/IEC, 1998.
[32] Jones, C. (2008). A new business model for function point metrics. Capers Jones & Associates Llc, 2008
[33] Moser, S., Henderson-Sellers, B., Mišić, V. B. (1999). Cost estimation based on business models. The Journal of Systems and Software. 49,1 (1999), str. 33-42.
[34] Stensrud, E. (1998). Estimating with Enhanced Object Points vs. Function Points. Proceedings COCOMO 13, 1998
[35] ISBSG (2009). International Software Benchmarking Standards Group, Functional Sizing Methods, from: http://www.isbsg.org/ISBSGnew.nsf/WebPages, accessed: 2011-10-24
[36] UKSMA MK II Function Point Analysis. Counting practices manual, V1.3.1. United Kingdom, 1998
[37] Celar, S.; Mudnic, E.; Kalajdzic, E. (2009). Software Size Estimating Method Based on Mk II FPA 1.3 Unadjusted. Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium / ed. Branko Katalinic, 2009, pp. 1939-1941