DOI 10.2298/CSIS110316068P

# A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle

Jovan Popović[1] and Dragan Bojić[1]

[1] Faculty of Electrical Engineering, University of Belgrade,
Bulevar Kralja Aleksandra 73,
11000 Belgrade, Serbia
{jovan.popovic, bojic}@etf.rs

**Abstract.** Even though there are a number of software size and effort measurement methods proposed in literature, they are not widely adopted in the practice. According to literature, only 30% of software companies use measurement, mostly as a method for additional validation. In order to determine whether the objective metric approach can give results of the same quality or better than the estimates relying on work breakdown and expert judgment, we have validated several standard functional measurement and analysis methods (IFPUG, NESMA, Mark II, COSMIC, and use case points), on the selected set of small and medium size real-world web based projects at CMMI level 2. Evaluation performed in this paper provides objective justification and guidance for the use of a measurement-based estimation in these kinds of projects.

**Keywords:** software measurement, effort estimation, comparative analysis, empirical evaluation.

## 1. Introduction

Estimating size and cost of a software system is one of the biggest challenges in software project management. It is one of the basic activities in the entire software development process, since a project budget, size of a development team, and schedule directly depend on the estimate of the project size and cost. Software teams use many different approaches for estimating effort required for implementing a given set of requirements. Approaches may rely on the experience (as in the expert judgment methods such as Wideband Delphi[1] or Planning Game [2]) or exploit requirement analysis by breaking requirements into elementary work items that can be easily estimated. This research focuses on algorithmic methods that try to quantify systems using certain measurement techniques, and apply algorithms and formulas in order to derive an estimate based on the measured size of the system.

Even though algorithmic methods provide more objective and accurate estimates, many software organizations are reluctant to apply them in practice. According to Hill [3], approximately 60% of the software organizations still use task-based estimates that rely on work breakdown structures (WBS) and expert judgment, 20% combines expert methods with the measurement methods as an additional validation, and only 10% rely solely on measurement methods. Goal of this research is to evaluate reliability of the methods that are most commonly used in practice, by applying them on preselected set of the real world projects.

Current studies [4] show that success rate of software project implementation is very low – only 30% to 35% of all software projects get finished within the planned time and within the budget. One of the most common reasons for such low success rate is an unsuccessful estimation (mostly based on the subjective judgment) and a lack of objectivity. As a result, there is an increasing pressure on software project teams to abandon the experience-based methods for estimating and planning, and to replace them with more objective approaches, such as measurement and analysis based methods. Measurement and analysis is a required process, even for the organizations with the lowest maturity level (level two) according to the CMMI standard [5], which supersedes the older SW-CMM.

Estimate accuracy varies with the phase of the project in which it was determined. Boehm [6, 7] presented this accuracy as so-called “Cone of uncertainty” shown on the Figure 1.

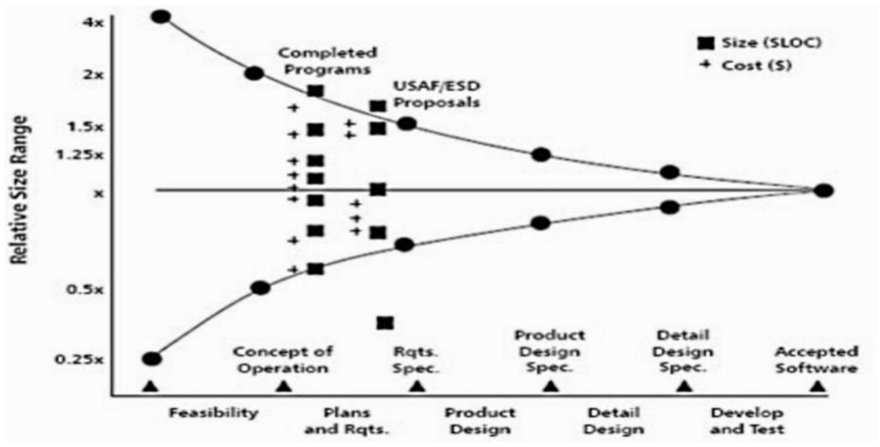

**Fig. 1.** Cone of uncertainty with effort estimate errors throughout project lifecycle (error varies from 400% at the start of the project and converges to the accurate effort at the end of the project)

Effort estimate determined in earlier phases of the project, such as definition of the initial concepts, might differ up to four times from the final one. This variation can be explained by the fact that initial concepts do not describe the final software system accurate enough. As more details get defined, the effort estimate converges to the actual value (that can be accurately determined when project is completed). Project Management Institute (PMI) [8] presents similar results about uncertainty, except that their results introduce an asymmetric cone where initial estimates vary between +75% and –25%, budgetary estimate between +25% and -10%, and the final estimate between +10% and -5%. We are using these boundary values as an evaluation criterion for the measurement methods used in the paper. Deviation between estimated and actual effort of the measurement methods used in this research needs to be within Boehm's and PMI boundaries; otherwise, methods should be rejected as inaccurate.

The rest of the paper, presenting the results of our analysis, is organized as follows:

1. Second section contains the problem statement and explains importance of evaluating applicability of the measurement methods in the practice;
2. Third section describes projects used for the analysis, classified measurement methods found in the literature, and methodology of measurement and analysis used in the research;
3. Fourth section describes how the measurements found in the literature were applied on the projects in our data set, how they can be applied in different phases of the project lifecycle, and what deviations exist between the effort estimated using the measurement methods and the real effort values;
4. The last section presents results of evaluation and shows that estimating effort using the measurement methods proposed in the literature has better accuracy than the one reported in practice.

## 2. Problem Statement

Avoiding the use of measurement and analysis in practice opens an important question – whether the measurement process is applicable in the software projects at all. In other engineering branches, scientific results and laws based on measurement are successfully applied in practice, giving engineering teams a valuable help in managing their own projects. Unfortunately, in software engineering, measurement is still not a preferred approach. According to Hill [3], only 30% of software companies use measurement, mostly as a method for additional validation. In order to determine whether the objective scientific approach can give better results than experience, we have evaluated several well-known measurement and analysis methods, applied them on a set of real-world projects and determined whether they are applicable or not. By obtaining a satisfactory evaluation results, we would prove that measurement techniques could be used for software project estimating, and it would allow us to define a methodology for applying measurement and analysis in practice.

Another important question is whether the software teams should use a single measurement method in the analysis, or they should combine multiple methods together. In practice, a software team is obliged to give various estimates throughout a project life cycle. At the beginning of the project, some ballpark estimates are needed with minimal engagement and analysis. In the later phases of the project, more accurate and more obliging estimates are needed. Various estimation methods are proposed in the open literature, of varying complexities. Hence, they provide different accuracies. We believe that the most efficient way for estimating a required effort is to use the proper combination of several different measurement methods. By combining the measures together, and applying them in the correct points of time in the project lifecycle, we get a continual estimating process, where the accuracy of an estimate converges to the right value as the project matures. The goal of this evaluation is to derive a methodology for applying the most appropriate measurement techniques during the software project lifecycle, including best practices in using measurement activities.

The evaluation performed in this paper is to provide objective justification and guidelines for using measurement-based estimates in software projects. Research results should prove that software companies could benefit from this methodology and increase the success rate of their projects over the value that can be expected from the current statistical results [4].

## 3. Measurement Approach

A measurement approach defines what kind of information, techniques, and tools will be used in the analysis. Our measurement approach is specified with the following elements:

1. A project data set gives details about a data collection used in the research. A data collection is a set of the software projects used for applying measurement methods and estimating the effort that is required for the projects to be completed;
2. Description and classification of the existing measurement methods found in the literature, with the detailed analysis of methods that might be applied on the project data set used in the research;
3. An analysis model describes the mathematical models and tools used in the research. A measurement model defines a structure for storing data, as well as techniques that are used in the analysis.

Once the measurement approach is defined, the measurement methods can be applied on the project data set, and the results can be evaluated and compared. The following sections describe concepts important for the measurement approach in more details.

### 3.1. Project Data Set

In this research, we have used a set of real-world software projects implemented from 2004 to 2010. All projects in the dataset are implemented by a single company for various clients located in the United Kingdom. Historical company database contains information about 94 projects, categorized by technology and application domain as shown in the figure 2.

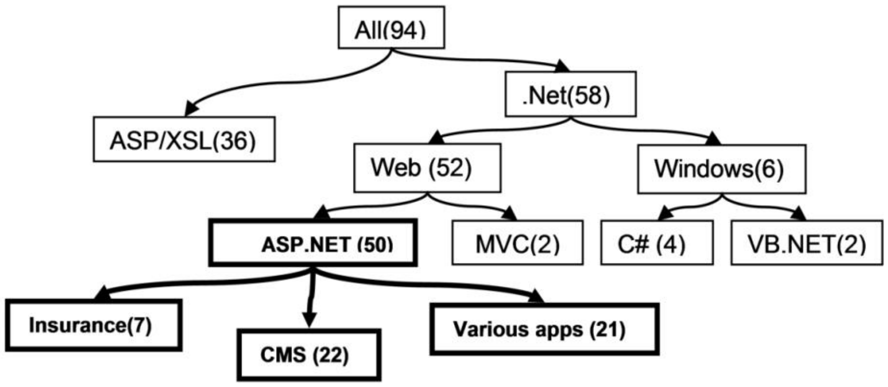

**Fig. 2.** Classification of the projects from the project dataset by used technology and by business domains. Highlighted projects are used in the analysis

In order to homogenize project dataset used in the analysis, we have excluded 36 ASP projects (OO approach is not applied in these projects and they are poorly documented), and 8 experimental Windows/MVC projects. The remaining 50 ASP.NET applications in the various domains represent set of the project used in the analysis.

Programming languages used in the ASP.NET projects are C#, T-SQL, HTML, and JavaScript. Consistent object-oriented analysis and design methodology was used in all projects in the dataset, and general software development [49] and project management practices [8] were applied. High-level domain models and use cases models were created in the earlier phases of the projects. These models were refined into the more detailed use case scenarios, sequence and entity-relationship diagrams. The core architectural elements were created according to the analysis models in the design phase. In addition, physical design of the system integration was described using the deployment diagrams. Most of the projects in the data set contain only partial technical design documentation. Detailed design documentation is created only for integration of third party components or legacy systems, where off-site teams have to review them before the project start. Only basic technical documents are created for detailed functionalities implemented in the projects, and in most of the cases, they are automatically generated using documentation generation tools. Hence, they do not contain enough information for applying the measurements based on detailed technical documentation.

A company that developed projects in observed data set implemented all practices at CMMI maturity level two. Hence, project management activities (e.g. planning, monitoring and control) were consistently performed across all projects in the data set providing enough information about the effort required for completing the project. Values of the effort spent on projects are recorded in the project plans, as well as in the invoices sent to customers; therefore, those values can be considered as valid. Various types of documents and specifications describe project parameters, functionalities, lifecycle, and team structure. Table 1 shows how many projects in the data set contain a particular type of documentation.

**Table 1.** Documentation and lifecycle models available in the projects

| Project items/characteristics | Number of projects |
|---|---|
| Project management documents | |
| Project plan | 46 |
| Estimate/Budget | 50 |
| Specification documents | |
| Vision scope and domain models | 24 |
| Requirement document | 9 |
| Use case models/scenarios | 27 |
| Database models | 30 |
| Functional specification | 23 |
| Analysis model | 14 |
| Software architecture document | 7 |
| Detailed UML design | 3 |
| Lifecycle model | |
| Phased(UP/MSF) | 23 |
| Agile(XP/Scrum) | 7 |
| Project characteristics/constraints | |
| Team structure description | 4 |
| Post mortem analysis | 5 |

Total amount of 30 projects, with enough technical documentation aligned with project plans where we can apply several different measurement methods, were selected for the final dataset. Statistical information about the effort, duration, team size, and experience of the project team members are shown in the table 2. The final set of the projects that is used in the analysis is homogenous by the team structure, with identical technical parameters (e.g. platform, programing languages, and complexity).

**Table 2.** Project statistical parameters

| | Effort (Person-month) | Duration (Months) | Team size (Person) | Experience (Years) |
|---|---|---|---|---|
| Minimum | 2 | 2 | 4 | 1 |
| First quartile | 4 | 4 | 6 | 4 |
| Median | 7 | 5 | 7 | 5 |
| Third quartile | 12 | 7 | 8 | 6 |
| Maximum | 15 | 12 | 10 | 8 |

## 3.2. Metrics Used in Research

One of the important tasks in the research was to select measures and metrics to be used for defining type of information that should be collected, as well as shaping the entire analysis process. There are many software size measurement approaches present in the literature. Metrics can be categorized as:

1. Metrics based on the source code such as a number of source lines of code [9], Halstead's [14] or McCabe's [15] complexity metrics;
2. Functional metrics such as IFPUG [10], NESMA [16], Mark II [17] or COSMIC [12] methods. These metrics are widely used in the software industry and they are standardized as ISO/IEC standards [13].
3. Object oriented metrics such as use case points [11], class points [18] or object oriented design function points [19].
4. Web metrics – a set of methods specialized for web application development. The first papers in this field were Reifer's web objects [21], followed by the work of Mendes[34, 35], and Ferrucci [36],
5. Other techniques and enhancements such as Galea’s 3D function points [20], Fetche's [22], Uemura's [23], or Buglione [40] approaches.

We examined project documentation shown in the table 1 in order to determine which measurement methods can be applied in the project data set. Results are shown in the table 3.

**Table 3.** Number of projects that can be used for the measurement methods

| Measurement methods | Number of projects |
|---|---|
| Source code based | 0 |
| Functional methods | |
| FPA/NESMA | 23 |
| NESMA Estimated/Indicative | 30 |
| Mark II | 30 |
| COSMIC | 21 |
| Object oriented methods | |
| Use case point | 27 |
| Class points/OOAD points | 6 |
| Web methods | N/A |
| Other methods | N/A |

Source code based metrics are not applied in the project data set. There are at least four different languages applied in each project, without the information what effort was required to implement pieces of code written in different language. Hence, it is questionable whether these metrics can be successfully applied in the project data set. In addition, as a significant (but unknown) amount of source code is automatically generated, it is impossible to isolate the code that is directly created by the development team and to measure just a size of that code.

All projects have enough functional documentation; therefore, it was suitable to apply measures in the functional group (IFPUG, NESMA, Mark II, and COSMIC). These measurements are widely used in the practice and certified as ISO/IEC standards [13] indicating that they should give satisfactory results.

The only object-oriented measure that is applied is a use case point metric. Functional specification has defined use case scenarios and models; hence, there is enough information for applying this metric successfully. The other object-oriented metrics are not applied in this research, as there is no sufficient information in the technical documentation. Therefore, applying these techniques on inadequate data will cause too many errors.

We have not applied web metrics in the research, although the projects in our dataset are web applications for several reasons. Application framework used in the projects (ASP.NET) encapsulates most of the web related code, and generates most of the HTML/JavaScript code automatically. Hence, there are many parameters used in the web methods that hidden from the development team in the framework, and therefore these parameters do not affects the effort. There are only few projects in the data set where web aspects of the application development are not completely encapsulated; however, there are not enough projects for the analysis.

The other nonstandard methods [20], [22], [23] were not applied, since each of them requires some specific information. In the existing documentation, we have not found enough information required for these methods; and our decision was to avoid any change or producing new documentation.

Most of these metric use both functional and nonfunctional parameters of the system when a final size is determined. Usually, two different sizes are created, the one that depends on the functional parameters, which we call unadjusted size; and another that is determined by adjusting an unadjusted size using nonfunctional parameters, which we call adjusted size.

As an example, there are 17 cost drivers and 5 scale factors used in COCOMO II [2] in order to adjust the functional size of the system. Each factor takes one of the rates (very low, low, nominal, high, very high, and extra high) and appropriate numeric weight. We have evaluated applicability of COCOMO II drivers in the projects from our data set, and we have found that most of them are too unreliable or undocumented. COCOMO II factors evaluated in our project data set are:

1. Product factors – required reliability (RELY), database size (DATA), and complexity (CPLX) might be considered as nominal according to the subjective judgment of team members. Documentation (DOCU) and reusability (RUSE) depends on the budget and project timelines; however, they are not documented in most of the projects.
2. Platform factors such as execution time (EXEC), storage (STOR), and platform volatility (PVOL) might be considered as nominal across all projects.

3. Personnel factors – there are no records about the team capability (PCAP and ACAP factors), experience (APEX, PLEX, and LTEX factors), nor about the personnel continuity (PCON factor).
4. Project factors – usage of software tools (TOOL), multisite development (SITE) and schedule (SCED) varies on the project but there are no organization-wide standard for assessing impact of these factors.
5. Scale factors process maturity (PMAT), team cohesion (TEAM) and development flexibility (FLEX) are nominal; however, there are no information about precedentedness (PREC) and architecture/risk resolution (RESL) factors.

Due to the fact that most of the non-functional parameters are either undocumented or depend on the subjective opinion of team members, only functional measurements were considered, without using any non-functional parameters of the system for adjustments. We are confident in the validity of the functional size metrics defined in this section because they are derived from the objective sources (e.g. documentation and prototypes). Combining these objective metrics with the factors that are either undocumented or subjective would affect objectivity of the results due to the high uncertainty of the non-functional parameters.

In this paper, the term “size” is equivalent to the term “unadjusted size” in the original measurement techniques.

### 3.3. Analysis Model

This section describes the model used for analysis of the projects in our dataset. The analysis model is represented through following components:
1. Data model that is used in the research where we have defined data structure used to store project data, and mathematical models used for analysis,
2. Prediction model where we have defined functional dependency between the size and effort,
3. Validation model where we have described methodology used to validate measurement/prediction methods used in the research.

#### Data Model

We have complete access to project characteristics and documentation for all projects in the dataset; however, these characteristics have to be organized so that we can easily use them in various measurement methods. Project characteristics required for all measurement techniques are collected from project documentation and grouped in tuples containing information about project name, effort, and all project characteristics required by selected measurement techniques. These characteristics are used to establish a uniform tuple space that represents a repository for all project measurements, and it is used as a base source of information for all selected measurement techniques in our research, as shown on the figure 3.

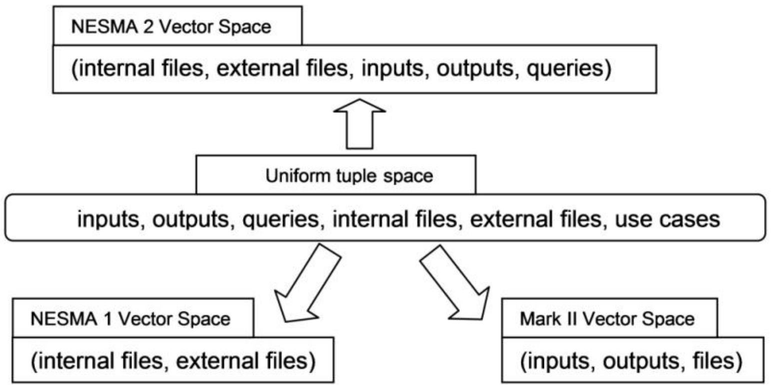
NESMA 2 Vector Space
(internal files, external files, inputs, outputs, queries)
Uniform tuple space
inputs, outputs, queries, internal files, external files, use cases
NESMA 1 Vector Space
(internal files, external files)
Mark II Vector Space
(inputs, outputs, files)

**Fig. 3.** Uniform tuple space with derived vector spaces specific for particular measurement techniques. Uniform tuple space contains information about all project characteristics. When vector spaces for the specific measurement techniques are needed, only attributes required for particular measurement are derived and projected into a new vector space that is used in measurement

In order to apply a particular measurement method to the given data set, specific information must be extracted from the uniform tuple space. Extraction of required project characteristics is done by restricting attributes in the tuple space, which creates a vector derived for a particular measurement technique. Example of deriving vector spaces for Mark II, NESMA indicative and NESMA estimated measurement techniques from the uniform tuple space is shown on the figure 3.

The uniform tuple space and derived vector spaces described in this section represent a flexible structure that can be used for applying a uniform measurement approach on various metrics. The model of the data structure is shown on the figure 4.

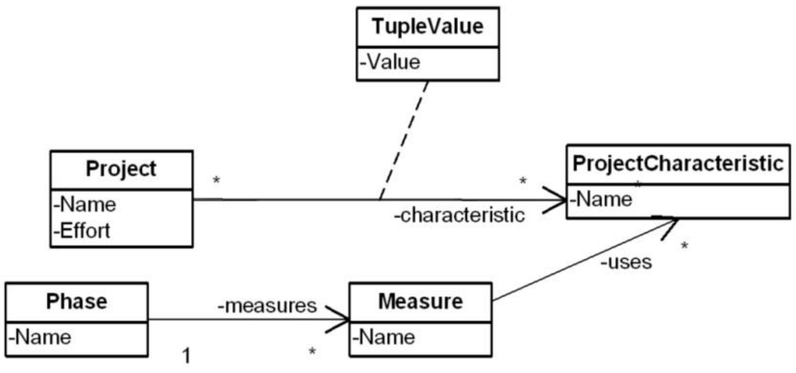

**Fig. 4.** Data model for the structure of the projects and their measured data used in the research

Each project in the data set has a name, effort, and set of project characteristics (e.g. number of use cases, inputs, or files). Each characteristic has an associated value in the tuple. Measures use some of the project characteristics to calculate the size of the system. Measures are grouped by phases of the project (this is described in more details in the section 4). This data model can be converted to any data structure for storing data e.g. in database relational model, or even in the flat MS Excel spreadsheets.

All measurement methods used in the research calculate size as a weighting sum of project characteristics, as shown in the formula 1.

$$Size = \sum_{i=0}^{N} w_i * x_i \tag{1}$$

In the formula 1, N is a number of characteristics that are used for some particular measurement, $w_i$ are constants, and $x_i$ represents tuple values of interest for some particular measurement method. For example, NESMA indicative method uses two characteristics (N=2) where $x_0$ is a number of internal files, and $x_1$ is a number of external files. Values of the weighting factors $w_i$ are specific for the particular measurement methods – reader can find more details about the specific rules and values of weighting factors in the section four, which describes ways of applying these measurement methods.

Formally, we are using different form of equation 1, represented as a scalar product shown in the formula 2.

$$||x|| = x * w . \tag{2}$$

In the formula 2, x is a vector that represents the project characteristics, and w is a weighting vector with the same number of dimensions as other vectors in the vector space. Vector w has constant value for each of the measurement technique. Scalar value ||x|| is the norm of the vector, and it represents measured size of the project from the formula 1. Applying formula 2 on the vectors derived from the tuple space, we are defining a normed vector space, where a norm represents the actual size of the project.

## Prediction Model

In order to evaluate a correlation between measured size and actual effort, it is necessary to hypothesize a functional model that describes the dependency between these two values. Such functional model would help us to predict an effort using the size, and compare predicted value with actual.

The most commonly used functional model is a linear dependency assuming that an effort needed to complete a project is directly proportional to the project size as it is shown in the formula 3.

$$Effort = PDR * Size . \tag{3}$$

Effort is a predicted value, Size is measured size for the project calculated using formula 1 or 2, and PDR is a Project Delivery Rate expressed as hours per unit of size [3]. Linear models are one of the earliest functional dependencies developed in the theory of effort estimation. They were used by Albrecht [24], then by Nelson [25], and finally Briand in COBRA [26] model.

The other group of models is a group of nonlinear models (an effort required for implementation is not directly proportional to the size). One of the first nonlinear models was created by Kemerer [27] who used polynomial dependency between the size and effort, and it has better accuracy than linear Albrecht's model [24]. Currently, the most common form of nonlinear models is a power function model defined in the formula 4.

$$\text{Effort} = \text{K Size}^{E} . \tag{4}$$

Linear coefficient K and exponent E are either constants or being calculated using the project constraints such as product complexity, delivery time, or team capability. Currently, the most common power function models are Boehm's COCOMO II and Putnam's SLIM model [28]. Other nonlinear models such as Walston-Felix [29] or Beily-Basily [30] models are not used in the practice. In the power models, a logarithm of effort (instead of effort) is directly proportional to the size of the system.

Different methods can be used to build a model for predicting the project effort based on the size. The most commonly used method is a regression model; however, there are lot of case studies where prediction models are significantly improved using the Fuzzy logic [37], Bayesian networks[38], Neural networks[39] etc. These methods are very effective when there are a large number of different parameters in the model that should be analyzed, or when models are too complex to be represented as a regular function.

In the projects in our dataset, we have only one independent variable – the size of the project that should be used to predict an effort. Organization that has developed projects in the dataset has CMMI maturity level two, where are implemented "Measurement and analysis" and "Project planning" practices. Therefore, we are convinced that size of the projects and efforts spent on the projects are valid parameters. Non-functional parameters, such as product complexity, analyst capability, or data complexity, are either undocumented, or we are not convinced that they are valid. Therefore, organization and process parameters are not used in the research.

The method we have chosen to build our prediction model is a linear regression technique. Due to the fact that we have only one independent value, and that functional dependencies are defined using the formulas 2 and 3 this is the most appropriate method in our data sets. In the linear regression model is the most common regression model, where estimated effort can be expressed as a linear dependency of the calibration constants and measured size, as shown in the formula 5.

$$Effort^{*}(Size) = k * Size + n . \tag{5}$$

In the formula, Effort* is a predicted value of the effort, parameter k and parameter n are constants. Ideally, k should be close to the PDR value given in formula 3.

As neither linear nor power model is favored in the analysis, the same regression formula is applied in both models. During the evaluation of linear model, effort was used as a predicted value, while in power models, logarithm of project effort is used instead of the effort.

The linear regression is just a basic model for a prediction. Many other techniques such as neural networks or case-based reasoning can be used instead of the regression [41]. However, most of these methods are more suitable in the cases where we have many different independent values that should be analyzed. As described above, we cannot be convinced that organizational parameters are valid in the observed project data set. Therefore, as we do not have a broad range of available independent variables, advanced methods are not applied in our data set. In the paper will be shown that companies on the maturity level two are limited to the basic prediction methods, and that if they want to use more accurate and advanced models, they will need to improve their maturity level to the CMMI level three at least.

**Validation Model**

In order to find metrics that are most suitable for the effort estimation, we have evaluated correlation between measured sizes and effort values. Pearson’s product-moment correlation factor is used as a criterion for evaluation of applicability of measures. If projects in the data set are represented as pairs of size and effort such as ($Size_i$, $Effort_i$), correlation between two sets of data is given in the formula 6.

$$R = \frac{\sum_{i=1}^{N}(Size_i - \overline{Size})(Effort_i - \overline{Effort})}{\sqrt{\sum_{i=1}^{N}(Size_i - \overline{Size})^2}\sqrt{\sum_{i=1}^{N}(Effort_i - \overline{Effort})^2}} \quad . \tag{6}$$

The value of the correlation coefficient R varies from 0 to 1, and values closer to 1 indicate that there is a better correlation between the sizes and efforts. Sizes that are highly correlated to the effort can be applied as valid measures of the system.

In addition, we have evaluated how well the predicted effort, calculated using the formula 5, is close to the actual effort. The actual effort is the sum of predicted value and prediction error as shown in formula 7.

$$Effort(Size) = Effort^*(Size) + \varepsilon \quad . \tag{7}$$

Estimation function Effort*(Size) (calculated using the formula 5) is determined so that the prediction error ε is minimized. In that case, estimated value of the effort is very close to the real values. Instead of the prediction error, we are using a mean relative error (MRE) [43] for evaluating the quality of the estimation model.

Mean relative error, or magnitude of relative error represents the ratio of the difference between the estimated and the real value ε, and the actual value itself, as shown in the formula 8.

$$MRE_i = \frac{\left|Effort_i^*(Size) - Effort_i(Size)\right|}{Effort_i(Size)} \ . \tag{8}$$

In order to evaluate the accuracy of the estimating model, mean and maximum values of magnitude of relative error are determined as well. Mean magnitude of relative error (MMRE) and maximum magnitude of relative error (MRE$_{max}$) are derived from the individual MRE values using formulas 9 and 10.

$$MMRE = \frac{1}{N}\sum_{i-1}^{N} MRE_i \ . \tag{9}$$

$$MRE_{\max} = \max_{i\in[1..N]}(MRE_i) \ . \tag{10}$$

MMRE indicates accuracy of the estimating model, while MRE$_{max}$ parameter reveals extreme cases. Values within the error range reported by Boehm and PMI prove that measurement technique can be successfully applied in the practice.

## 4. Applying Measurement During Project Lifecycle

Once the preparation for measurement is finished i.e. data set, measures, and model are defined, selected measurement techniques can be applied in order to evaluate their accuracy.

One of the goals of this research was to evaluate whether the measurement techniques selected from the literature can be used for estimating an effort during the project life cycle. We use the standard phases from Unified Process (UP) [31], as a lifecycle model of the software development process:

1. Inception – where a scope of the system is agreed upon, and requirements are described at the highest possible level;
2. Elaboration – where major analysis activities are done, and requirements, architecture and a general design are specified;
3. Construction – where detailed design is completed, code is created and tested according to the requirements and design;
4. Transition – where a solution is transferred to end users.

This phased model is applicable to all projects in the data set, even if some of them were not implemented according to the strict UP model (e.g.

waterfall or MFS [32]). UP phase model can be applied even on the iterative models such as Extreme Programming [2] or Scrum [33], where iterations can be divided into smaller phases mentioned above.

Documentation available in the projects within the project data set, can be categorized by the phase in which they were created. For example, a vision/scope document, statement of work, and project brief document, which are created at the start of a project, are categorized as inception phase documents. List of the available documents and creation phases is shown in the table 4.

**Table 4.** Documents and models available in various phases of the software development process

| Project Phase | Available Documents |
|---|---|
| Inception | Project Brief/Proposal |
| | Vision/Scope Document |
| | Conceptual Domain Model |
| Elaboration | Use Case Model and Scenarios |
| | User Interface Specification |
| | Logical Model |
| | Functional Specification |
| | Physical Model |
| Construction | UML Class Diagrams |
| | System Sequence Diagrams |
| Transition | Training Material |

### 4.1. Measurement During Inception Phase

In the earlier phases of projects such as Inception phase in the UP [31] model, or Envisioning phase in the MSF [32] model, only high-level requirements are defined. Table 4 shows that vision/scope document, statement of work, and project brief document are created during the inception phase. These documents contain information about scope of the project, users who will use the system, their needs, high-level features of the system, and domain models. From the set of available metrics, the following ones can be applied in the inception phase:

1. NESMA indicative metric – domain model which contains business concepts and their description is available, metric of the system based on the number of internal and external objects can be determined for each project;
2. Number of use cases – use cases can be identified from the vision/scope document, and their count can be used as a measure of the system size. Farahneh has used the same metric in his research [42].

NESMA indicative method approximates the standard NESMA functional point method, where size of the system is determined by using just basic functional information. The logical groups of pieces of information (files in the functional point measurement terminology) that represent data structures are identified in the system. Structure of files (relationships between files and information they contain) are not relevant. The only analysis that should be done is categorization of files as internal and external, depending on the fact whether they are maintained within the system or just referenced from the other systems. In this case, NESMA indicative metric uses vectors containing the number of internal/external files in the form (ILF, EIF). Size of the system is determined by deriving norm of the vector using a scalar product between weighting vector (35, 15) and the size vector.

Number of use cases approximates the standard use case point metric, since the use cases are just enumerated without any deeper analysis related to their complexity. The vision/scope document, which is created during inception phase, contains high-level features of the system without their detailed descriptions. Hence, use cases can be identified upon these features, although detailed scenarios are still not defined. The number of total use cases to be implemented in the system can be used as a measure of the system functionalities. This number is pure scalar metric; it has only one dimension that represents size of the system.

**Table 5.** Measurement applied during inception phase with their correlation with effort (R), Mean magnitude of relative error, and Maximal magnitude of relative error for both linear and nonlinear (power) models

| | Linear model | | | Nonlinear model | | |
|---|---|---|---|---|---|---|
| Metric | R | MMRE | MRE$_{max}$ | R | MMRE | MRE$_{max}$ |
| NESMA Indicative | 93% | 16% | 46% | 85% | 20% | 69% |
| Number of Use Cases | 58% | 46% | 126% | 59% | 40% | 94% |

These two metrics are applied on the projects in our data set to determine the size of each project expressed in NESMA function points and number of use cases. Correlation between the size and actual effort was analyzed and results of the analysis are shown in the table 5.

The NESMA indicative method shows significantly better results than a number of use cases as a size metric. Number of use cases does not show a true nature of the system, since beneath each use case frequently lie several hidden functionalities that might significantly increase the effort required for implementation of the use case. Hence, there is a significant variation between the effort and number of use cases. Farahneh [42] got similar results when he applied the number of use cases as a measurement. Accuracy in his case study was between 43% and 53%; however, this is still not even close to the accuracy of the NESMA indicative method.

The major issue with counting use cases is the absence of any standard that precisely defines what a use case is. Use cases in our dataset differ from project to project; they might be either merged or decomposed to several use cases. There is no strict rule defining when a set of user actions should be placed within one use case scenario, nor when they should be decomposed into set of different use cases (e.g. included or extended sub use cases). Granularity of use cases significantly affects a number of use cases, and therefore size of the system. We believe that inconsistency in the style causes large deviations in the number of use cases as a size metric.

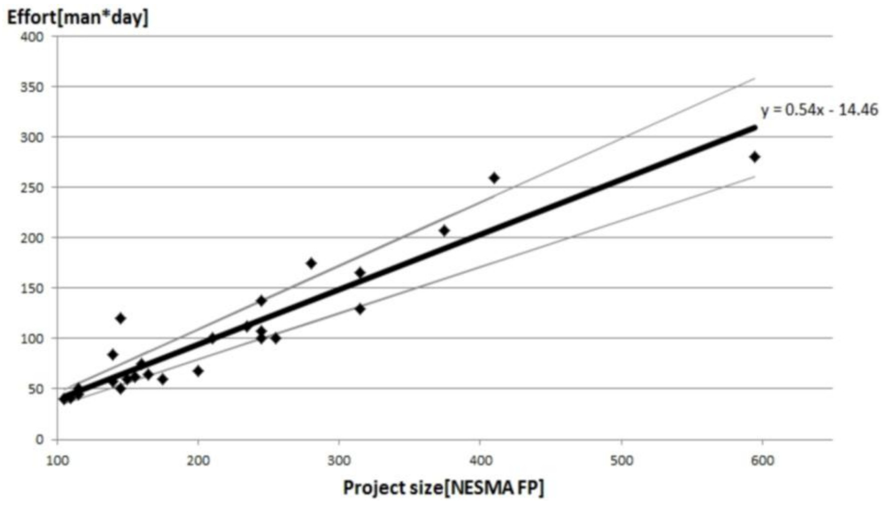

**Fig. 5.** Dependency between the effort and size measured in functional points using NESMA indicative method. There is a good linear dependency where most of the real effort values are near to the estimated values

Having in mind an absence of detailed information about the system functionalities, NESMA Indicative metric has a good accuracy that is acceptable in the practice. The inception phase milestone (Scope/Vision completed) is equivalent to the “Concept of Operation” milestone in Boehm’s report, or to “Project initiation phase” in the PMI terminology, because in these phases we have just a rough description of system features. Comparing these results with the results reported by Boehm’s and PMI it might be noticed that estimating software size using NESMA indicative method gives better results than the ones that can be found in practice. Using Boehm’s results, estimating error in Concept of Operation milestone can be up to 100%, and according to PMI accuracy varies from -25% to +75%. Therefore, the mean relative error of 16% is good, and even the maximal error of 46% is satisfactory.

Dependency between sizes of projects measured as NESMA Indicative function points and actual effort applied on the data set used in the research is shown on the figure 5.

Applying NESMA indicative functional point method in the inception phase of the project enables project team to estimate the effort with accuracy that is significantly better than the results found in the practice. Therefore, using measurement method in this phase can increase accuracy of project estimate.

## 4.2. Measurement During Elaboration Phase

During the elaboration phase of the project analysis and design are performed. Those activities are necessary for translating user requirements into the detailed specification. Elaboration phase can be divided into several separate iterations. In the earlier iterations, focus is still on the user needs, requirements and analysis, while in the later iterations focus is on the detailed design, technical solution and full functional specification.

In earlier iterations of the elaboration phase, detailed user requirements are collected; scenarios of usage are documented via use case scenarios; and user interface and logical model of data are created.

As elaboration phase continues, more documentation and design with details and technical requirements are created. Detailed design allows project team to apply more detailed and more accurate metrics. In later iterations of elaboration phase, detailed functional requirements with a description of all fields, data structures, rules, and processes that should be implemented in the system are created. These iterations may include creation of detailed database models with all necessary fields and relationships so that the project team can start with development. Several metrics can be used in elaboration phase:

1. Use case points – a method that is used to measure size of the system by analyzing a complexity of use case scenarios;
2. NESMA Estimated method – an approximation of the standard measurement technique based on functionalities of the system;
3. Mark II method – a method used to measure a functional size of the system, mostly applied in United Kingdom;
4. Functional point analysis (FPA) method – a method where detailed functionalities of the system are measured.

First three metrics can be applied in earlier iterations, while the fourth metric can be used when elaboration is near to end. In the rest of this section, we will separately discuss metrics that can be applied in earlier and later iterations of the elaboration phase, as well as the accuracy of these methods.

### Metrics Used in the Earlier Iterations of Elaboration Phase

Detailed use case scenarios are described in earlier iterations of elaboration; hence, there is enough data to determine the exact measure of use case complexity expressed in use case points. The use cases are categorized as low, average, and high, depending on the number of transactions within the use case scenarios. They are combined with three classes of user complexity (low, average, high) forming six-dimensional vector ($UC_H$, $UC_A$, $UC_L$, $A_H$, $A_A$, $A_L$) that is used to measure size of the system. The norm of the vector is determined using a scalar product between the use case point size vector and the weighting vector (15, 10, 5, 3, 2, 1).

Functionalities and basic concepts of the system are also defined. Hence, some estimated metrics based on functionalities such as NESMA Estimated and Mark II method can be applied.

NESMA Estimated method approximates the standard NESMA method for measuring the functional size. Size of the system is determined by identifying objects and processes in the documentation. Objects in the system are classified as internal or external files, depending on the fact whether they are maintained within the system or just referenced from other systems. Functional processes are classified as inputs, outputs, and queries. Total number of internal files, external files, inputs, outputs, and queries forms a five dimensional vector (ILF, EIF, EI, EO, EQ), which is used to determine size of the system. The norm of the vector is determined as a scalar product of the size vector and the weighting vector (7, 5, 4, 5, 4).

Mark II method is a modification of standard functional point method where information about the number of input processes, output processes and objects within the system are taken into consideration. In the Mark II method, a three-dimensional vector of inputs, outputs (exists), and objects ($N_i$, $N_e$, $N_o$) is used for determining size of the system. The norm of the vector is determined by using a scalar product between the weighting vector (0.58, 1.66, 0.29) and the vector itself.

### Metrics Used in the Later Iterations of Elaboration Phase

Detailed functional point metric (either standard IFPUG or very similar detailed NESMA method) can be applied in the later iterations of elaboration phase. Functional point analysis is done using detailed description of user interface such as UI design or process flows, and informational models such as Entity-Relationship diagrams or class diagrams.

In the functional point analysis five system parameters are analyzed – internal logical files (ILF), external interface files (EIF), inputs in the system (EI), outputs from the system (EO), and queries (EQ). However, these elements are not simply counted – they are also categorized as low, average, and high, depending on their complexity. Rules for classification depend on the number of data elements and files that are affected by the transaction for the inputs, outputs, and queries; and number of data and composite structures for files [10]. The number of elements in each class represents a separate dimension of the system, forming a vector with fifteen dimensions**.** The norm of the vector is computed using a scalar product between the vector that represents system and the weighting vector.

### Accuracy of Measurements Used in the Elaboration Phase

Results of applying the measurement methods described in the previous sections on the project data set are shown in the table 6.

Early iterations in the elaboration phase are equivalent to the "Requirement specification" phase in the Boehm's cone of uncertainty, or "Budgetary estimate" point in the PMI model, since in this phase there is enough information to create a valid budget of the project. Estimating error is up to 50% in the requirement specification phase by Boehm, and between -10% and +25% by PMI results. Therefore, mean error of 10-13%, achieved using NESMA estimated or Mark II methods is more than acceptable. Even the maximal error of 30-65% is within the acceptable range according to the Boehm's results.

**Table 6.** Accuracy of measurements applied during elaboration phase

| | Linear model | | | Nonlinear model | | |
|---|---|---|---|---|---|---|
| Metric | R | MMRE | $MRE_{max}$ | R | MMRE | $MRE_{max}$ |
| Use Case Points | 80% | 31% | 94% | 74% | 29% | 74% |
| NESMA estimated | 92% | 13% | 65% | 93% | 16% | 34% |
| Mark II | 92% | 10% | 30% | 93% | 15% | 35% |
| FPA | 92% | 10% | 22% | 96% | 12% | 29% |

Correlation between the project effort and a use case point size is significantly lower, and mean relative error of 31% is outside of the acceptable range of errors. The accuracy of the UCP method varies in the literature. In three case studies [45, 46, 47] Anda has reported that MMRE is between 17% and 30%, and Lavazza [49] got MMRE of 29.5%. Hence, this metric is not good enough, at least not in the data set used in the research. In the projects from our data set, we have found a variety of different styles of use cases, which might cause a high prediction error. Two major issues we have noticed in documentation (that affect use case point metric accuracy) are:

1. Decomposition of use cases – there is no strict rule that defines when a set of user actions should be placed within one use case scenario, and when they should be decomposed into set of different use cases (e.g. included or extended sub use cases). Granularity of use cases significantly affects a number of use cases, and therefore a measured size;
2. Level of details – some use cases contain just an essential functional description, while the other have many details. Number of details placed in the use case scenario affects a number of transactions and measured complexity of the particular use cases. Hence, size expressed in the use case points varies although functionality is the same.

We believe that these issues cause a low accuracy of the UCP method in our data set. Different data sets/studies might give better results if use cases styles are standardized. However, we have decided not to alter, standardize or "improve" quality of documentation in order to get real values of accuracy.

Later iterations of the elaboration phase in the Unified Process are equivalent to the Boehm's "Product design" phase where estimating error is around 25%, or final estimate in the PMI model where the error is from -5% to +10%. Complete functional point analysis metric with mean error of 10% and maximal error of 22% is within the Boehm's range, but not better than the error ranges reported by PMI, especially if a maximal error is considered. However, the FPA method cannot be held responsible for a lack of accuracy when it is compared with PMI results. This research analyzes dependency between the functional sizes and efforts without considering nontechnical factors. Therefore, it is reasonable to assume that adjusting the size examined in this research using the same nontechnical factors would increase the accuracy of estimate and align it with the results found in the practice. A result of applying FPA metric on the projects in the dataset is shown on the figure 6.

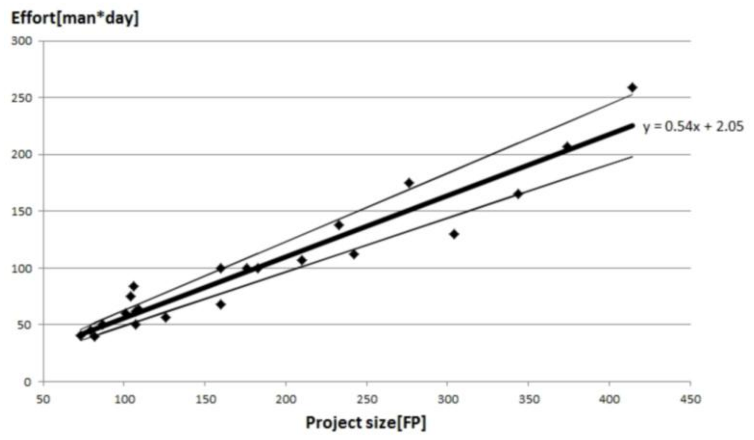

**Fig. 6.** Project sizes (expressed in FPA functional points) versus actual project effort. Most of the projects are placed around the regression line and within the MRE area bounded with the two thin lines

## 4.3. Measurement During Construction Phase

Focus of the measurement and analysis in the construction phase is not on the estimating size of the system. The main objective of the project management activities in the construction phase is project monitoring and control. Therefore, the focus of the measurement process is moved from the system as a whole to the individual pieces of the system (project tasks). Size of the entire system is measured in the previous phases; effort required to implement the software system is already determined; and budgets/plans for construction are developed according to the estimates. Development team needs to control a development process, compare a progress with the plan, and take the corrective actions if there are any significant deviations between the actual work and planned work. Even in the agile project approaches (where planning is done in the iterative manner) the most important task is to control performances of a software team. Therefore, a main goal of the measurement is estimating a size of the tasks that were planned in the iterations. In order to determine size of the particular functionalities we can apply two measurement methods in the construction phase:

1. Functional point analysis method – same technique used to determine size of the system method can be applied on the particular tasks as well;
2. COSMIC Full Functional Point method – latest method in functional metric family can be applied on all functionalities that have defined communication protocols.

The other methods presented in the paper are not suitable for measuring particular tasks. NESMA indicative and estimated methods are just approximations of the standard functional point method. Hence, standard NESMA or IFPUG methods have better results. Mark II measure is too robust because it only counts functional elements. Use case point method might be applied on the particular tasks under the assumption that use cases match project tasks. However, for projects in our data set, most of the project tasks represent parts of the use cases. Hence, use case point method cannot be applied on most of the projects in the dataset.

Standard functional point analysis can be applied on the particular tasks in the same way as on the entire system. The only difference is that functional sizes are not summed up in order to determine the size of the system. Measured sizes are associated to the particular tasks that are classified as inputs, outputs or enquiries and rated as low, average or high complexity tasks (based on the specification). The size is expressed as a scalar measure representing complexity of the particular functionalities.

Functional dependency between the effort required to complete tasks and corresponding sizes is shown in figure 7.

Size of transactions is between three and seven functional points (FP), depending on the type and complexity of the functionality. In our dataset, efforts of particular tasks are in the range from one hour to 2.5 days.

For medium sized tasks, one can establish some dependency between size and time. However, this method is not applicable on the tasks that are either too small or extremely long. Projects in our data set contain tasks in broad ranges of tasks from 1-2 hours, including minor queries or service calls, to the extremely complex reports that fell out of the medium range of tasks.

COSMIC method is the latest method from the functional point family, which should overcome drawbacks of the existing functional point methods. While using this method, we measured all communication messages that are exchanged between user of the system and system components.

The prerequisite for applying COSMIC method is an existence of detailed communication specification or sequence diagrams that define how messages between users, system components, and data modules are exchanged. In the COSMIC method, there are four types of messages that are identified – entry messages (E) where information is entered in the system, exit messages (X) where information is taken from system components, write messages (W) used to store information in the persistent storage, and read messages (R) used for taking information back from the persistent storage. Size of the functionality is calculated as a total amount of all messages exchanged between user and system components within that functionality. Maximum size of functionalities is not limited, since there is no upper boundary regarding the number of messages that can be exchanged within the functionality. Dependency between the task sizes and efforts is shown in the figure 8.

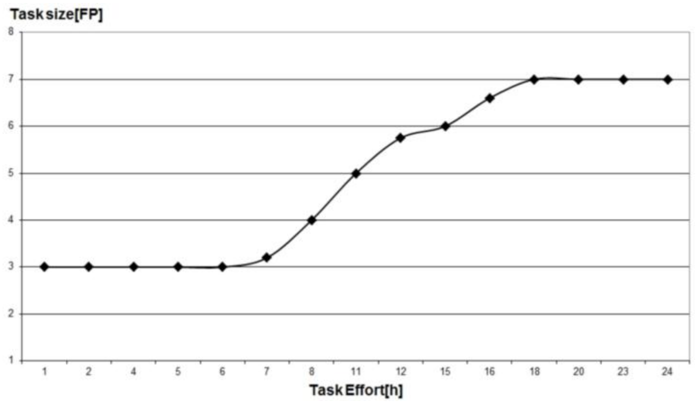

**Fig. 7.** Relationship between efforts required to implement particular tasks and task size expressed in functional points. For tasks that need between 7 and 17 hours to complete, a dependency is monotone; however, for tasks outside this range there are two saturation areas

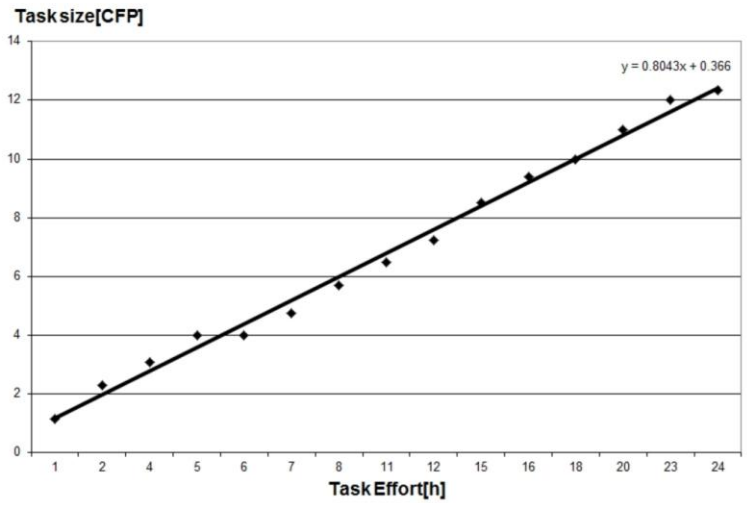

**Fig. 8.** Relationship between size of functionalities expressed in COSMIC points (CFP) and required effort recorded in hours. Applied linear regression returns good estimating function where actual values are close to the real ones

COSMIC metric does not have the same kind of constraints as FPA, since the number of messages that are exchanged between user and system components can vary from one to infinity. Hence, there is no saturation area in the figure 8. Deviation between the real task effort and the estimated one is less than a half of an hour, representing a very good accuracy with a mean error of 8%. Applying linear regression on the data set, it can be determined that there is a good functional dependency between the mean development time for tasks and size expressed in COSMIC functional points (CFP). Knowing that Boehm's results during the "detailed specification definition" milestone give error equal to 10%, this result is also acceptable.

Accuracies of the FPA and COSMIC methods are shown in the table 7. Correlation and error of the FPA method are determined only in the linear area.

**Table 7.** Compared accuracies of the FPA and COSMIC methods

| | R | MMRE | $MRE_{max}$ |
|---|---|---|---|
| FPA | 95% | 10% | 14% |
| COSMIC | 97% | 8% | 11% |

COSMIC method is highly correlated with the actual effort of the tasks with a very good accuracy. FPA method also has a good correlation if it is applied on medium sized tasks. However, this method cannot be successfully applied on the tasks outside the linear area.

## 5. Conclusion

Results presented in the paper are empirical evidence that measurement and analysis can be successfully applied in the practice. Our evaluation of measurement methods on the real projects shows that there is no objective reason for using a subjective judgment in the process of project effort estimation, because the measurement based methods are accurate enough. These results are explained in more details in the following section.

### 5.1. Results

We have evaluated the most common measurement techniques and applied them on the real projects to estimate the effort required for the project. Estimated values of efforts are determined using regression techniques based on the measured sizes. We have compared estimated effort with the real effort and determined the prediction error. Furthermore, correlation between the measured size and the actual effort are determined in order to evaluate whether the measures can be applied in the effort estimation.

We have determined a mean and maximum magnitude of relative prediction error for all measurement methods used in this research. These criteria are used to compare and evaluate accuracies of the estimating models. Evaluation methods that use the pure MMRE criterion were criticized in [44]. Therefore, we have determined distribution of the mean error for all methods including minimum value, median value, and the range where are placed 50% of the errors. Figure 9 shows compared distributions of the method accuracies applied in the UP project phases.

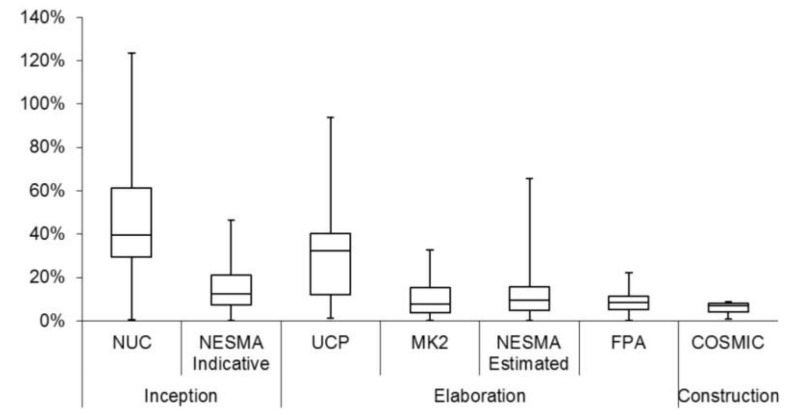

**Fig. 9.** Compared accuracies of the measurement methods applied in the research. Methods are grouped by UP project phases

The half of the errors displayed in the boxes surrounding median value of the error is in the acceptable ranges below 60%. The functional methods in most cases have good accuracy, although there are potential outliers with more than 60% for NESMA estimated method. Measurement methods based on the use cases are not accurate as the functional methods. We believe that cause of this inaccuracy is lack of the specification standards for definition of use cases.

Diagram shows the best practice methods that can be applied during the project lifecycle:

1. NESMA indicative method that has the best accuracy at the beginning of the project;
2. NESMA estimated and Mark II that have the best accuracy in the early iterations of the elaboration phase. Standard FPA method that has even better accuracy; however, it can be applied only in the later iterations of the elaboration phase;
3. COSMIC method has the best correlation with the effort at the beginning of the construction phase.

Accuracy of the best practice measurement techniques that are applied on the project in our data set is shown in figure 10. Results are divided per project phases and compared with Boehm's and PMI results.

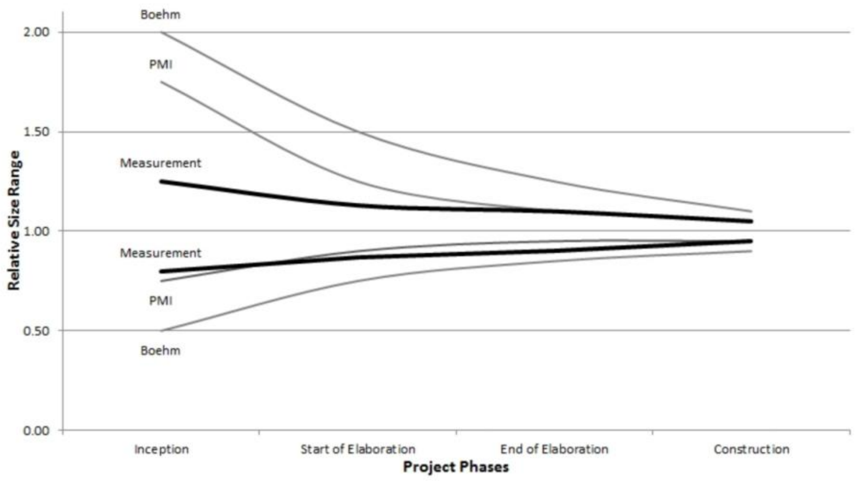

**Fig. 10.** Results of the estimate accuracy based on measurement compared with the Boehm's and PMI results

Results of the estimation using the measurement techniques and applying linear regression have better results than the results found in practice. Estimating errors applied on the data set are within Boehm's and PMI boundaries (except the lower boundary of the PMI where we have minor difference). In the earlier phases of the project, measurement methods have significantly better results than any results reported in the practice.

This is a concrete proof that there is no reason for assuming that the algorithmic models based on the measurement and analysis would not give good results. If applied correctly, estimating methods based on the metric can be used as quick and accurate methods for estimating the project effort.

Research shows that for small and medium projects, there is no significant difference between linear estimating models such as COBRA and nonlinear models such as COCOMO II or SLIM. Although it is reasonable to assume that many factors include nonlinear dependency between the size and the effort, it is shown that the effects of nonlinear behavior can be found only on the larger projects. A current trend in the software development is dividing large projects in smaller subprojects where a system is implemented in the iterative manner. Hence, this is a valuable conclusion. According to the results in the research, the linear models are applicable on the smaller projects; hence, project teams that practice an agile or iterative development can use simpler linear methods for determining functionalities between the size and the effort.

The fact that linear models can be applied is important because beside the simplicity, linear models enable project teams to track comprehensible performance indicators such as productivity. Measuring and tracking productivity is important in the software organizations, since one of the most important tasks is optimizing process performances.

A very important result in the paper is the fact that good accuracy of the estimating model is achieved although non-technical characteristics of the system are not used. Expert judgment methods consider both features that should be implemented and non-functional factors such as programmer/analyst capability or platform/technology complexity. Accuracy of estimates based on the bare functional size evaluated in this paper is close or even better than accuracies found in the practice, even if the non-functional parameters are not included in the research. This emphasis the great potential of the functional measurement presented in the paper. If the non-functional parameters that are used in the expert judgment methods are applied on metric presented in this paper, accuracy will be significantly improved and probably give far better results than those reported by Boehm and PMI.

## 5.2. Future Work

Our study shows that there is a good correlation between the pure functional size and the actual project effort; however, the influence of the nontechnical factors on the effort is not evaluated. Current estimation models, such as COCOMO II or SLIM, use several nontechnical parameters to adjust the functional size and improve accuracy of the predicted estimates. The inclusion of these parameters in the measurement model presented in this paper would significantly improve the accuracy of estimates. Note that our study shows that the accuracy of estimates in currently analyzed bare functional models is, in most of the cases, better than accuracy expected in the practice.

In the present data set, information about the nonfunctional parameters of the system is either incomplete, or it is arguable whether they are objective. The organization used as a source of information does not have implemented some crucial processes on the CMMI maturity level 3, such as organizational process definition (OPD) or decision analysis and resolution (DAR). Therefore, no organizational standards exist to assess the ability of workers who worked on the project. Lack of the objective organizational standards might cause significant variation in a team capability or product complexity assessment as it will rely on the subjective evaluation of the particular team members. As data about the nontechnical characteristics are too subjective, they have been discarded from the research. Next step in the research will be a creation of an organization-wide model for evaluation of nonfunctional parameters and including these values as parameters in the uniform tuple space. Once the nonfunctional parameters are collected, estimating models such as COCOMO II, SLIM, or COBRA can be directly applied, and we would be able to evaluate which of them corresponds to the project data.

In addition, current maturity level does not guarantee that processes are optimal; only that their cost can be consistently predicted using the measurement data. One of our future goals is to evaluate how measurement and analysis can be applied on the higher maturity levels where measurement is used both for prediction and process optimization.

**Acknowledgement.** This work has been partially funded by the Ministry of Education and Science of the Republic of Serbia, grant. no. TR32047.

## References

1. Stellman A., Greene J.: Applied Software Project Management, O'Reilly Media, Inc., 1 edition (November 2005), ISBN: 0-596-00948-8
2. Beck K., Extreme Programming Explained: Embrace The Change, Addison Wesley, 2004
3. Hill P.: Practical Software Project Estimation: A Toolkit for Estimating Software Development & Duration, 3rd Edition, McGraw - Hill, 2010
4. CHAOS Report: The Standish Group, 2008.
5. CMMI Product Team: CMMI® for Development, Version 1.3, Carnegie Mellon, Software Engineering Institute, November 2010
6. Boehm B.: Software Engineering Economics, Englewood Cliffs, N.J: Prentice-Hall, 1981.
7. Boehm B.: Software Cost Estimation with COCOMO II, Prentice-Hall, New Jersey, USA, 2000
8. A Guide to the project Management Body of Knowledge ( PMBOK Guide), 4th Edition, The Project Management Institute PMI. ISBN 978-1-933890-51-7
9. Park. R.: Software size measurement: a framework for counting source statements, CMU/SEI-92-TR-020.
10. Garmis D., Herron D.: Functional Point Analysis, Addison-Wesley, Boston USA, 2001
11. Clemmons R.: Project Estimation With Use Case Points, The Journal of Defense Software Engineering, 2006
12. Abran A., Meli R., Simons C.: COSMIC Full Function Point Method: 2004 – State of Art, SMEF04, Rome, Italy, Jan. 2004.
13. ISO/IEC 14143-1:2007 Information technology – Software measurement – Functional size measurement
14. Halstead M.: Elements of Software Science, Elsevier North-Holland, New York, USA, 1997
15. McCabe T.: A Complexity Measure, IEEE Transactions on Software Engineering: 308–320. December 1976
16. Netherlands Software Metrics Association (NESMA): Definition and Counting Guidelines for the Application of Function Point Analysis, Version 2.0, Amsterdam NESMA 1997.
17. Symons C.: Software Sizing and Estimating, Mk II Function Point Analysis, John Wiley & Sons, Chichester, England, 1991
18. Costagliola G., Ferrucci F., Tortora G., Vitiello G.: Class Point: An Approach for the Size Estimation of Object-Oriented Systems, IEEE Transactions on Software Engineering, Vol. 31, No.1 January 2005
19. Janaki Ram D., Raju S.V.G.K.: Object Oriented Design Function Points, apaqs, pp.121, The First Asia-Pacific Conference on Quality Software (APAQS'00), 2000

20. Galea S.: The Boeing Company: 3D function point extensions, v. 2.0, release 1.0, Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.
21. Reifer D.: Web Development: Estimating Quick-To-Market Software, IEEE Software, vol. 17, pp. 57-64, 2000.
22. Fetcke T., Abran A., Nguyen T.H.: Mapping the OO-Jacobson Approach into Function Point Analysis, Proc. TOOLS-23"97, IEEE Press, Piscataway, N.J., 1998.
23. Uemura T., Kusumoto S., Inoue K.: Function Point Measurement Tool for UML Design Specification, Proceedings of the 6th International Symposium on Software Metrics, p.62, November 04-06, 1999
24. Albrecht A., Gaffney J.: Software Function, Source lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Transactions on Software Engineering, SE-9 (6): 639-648, (1983).
25. Nelson R.: Management Hand Book for the Estimation of Computer Programming Costs, Systems Development Corp., 1966.
26. Briand L., El Emam K., Bomarius F.: COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment, International Software Engineering Research Network Technical Report ISERN-97-24, 1997
27. Kemerer C.: An empirical validation of software cost estimation models, Communications of the ACM, v.30 n.5, p.416-429, May 1987.
28. Putnam L.: A General Empirical Solution to the Macro Software Sizing and Estimating Problem, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO. 4, JULY 1978
29. Walston C., Felix C.: A method of programming measurement and estimation, IBM Systems Journal, vol. 16, no. 1, pp. 54-73, 1977.
30. Bailey J., Basili V.: A meta model for software development resource expenditure, in Proceedings of the International Conference on Software Engineering, pp. 107–115, 1981.
31. Kruchten R.: The Rational Unified Process – An Introduction, Addison-Wesley, Boston, USA, 2000
32. Microsoft Solutions Framework version 3.0 Overview, White paper, published 2003, Microsoft Press
33. Schwaber K., Beedly M.: Agile Software Development with Scrum, Prentice Hall, 2001.
34. Mendes E., Watson I., Triggs C., Mosley N., Counsell S.: A comparison of development effort estimation techniques for Web hypermedia applications, In: Proceedings of the Eighth IEEE Symposium on Software Metrics, 2002
35. Mendes E.: Cost Estimation Techniques for Web Projects, IGI Publishing Hershey, PA, USA 2007, ISBN:1599041359 9781599041353
36. Ferrucci F., Gravino C., Di Martino S.: Estimating Web Application Development Effort Using Web-COBRA and COSMIC: An Empirical Study, Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009
37. Attarzadeh I., and Ow S. H.: Improving the Accuracy of Software Cost Estimation Model Based on a New Fuzzy Logic Model. World Applied Sciences Journal, 8 (2): 177-184, January 2010. ISSN 1818-4952.
38. Mendes E., Mosley N.: Bayesian Network Models for Web Effort Prediction: A Comparative Study, IEEE Transactions on Software Engineering, v.34 n.6, p.723-737, November 2008

39. Ajitha, S. Kumar, T.V.S. Geetha, D.E. Kanth, K.R.: Neural Network model for software size estimation using Use Case Point approach,
40. Buglione L., Cuadrado - Gallego J., Gutiérrez de Mesa A.: Project Sizing and Estimating: A Case Study Using PSU, IFPUG and COSMIC, In: IWSM/Metrikon/Mensura '08 Proceedings of the International Conferences on Software Process and Product Measurement, 2008
41. Finnie G., Wittig G., Desharnais J.: A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case - Based Reasoning and Regression Models, Journal of Systems and Software, vol. 39, no. 3, pp. 281 - 289, 1997
42. Farahneh H., Issa A.: A Linear Use Case Based Software Cost Estimation Model, World Academy of Science Engineering and Technology, no. 32, 2011, pp.504 - 508
43. Conte S.D., Dunsmore H.E., Shen V. Y.: Software engineering metrics and models. The Benjamin/Cummings Publishing Company, Inc., 1986.
44. Foss T., Stensrud E., Kitchenham B., Myrtveit I.: A Simulation Study of the Model Evaluation Criterion MMRE, IEEE Transactions on Software Engineering, 29(11), 895 - 995, 2003
45. Anda, B.: Comparing effort estimates based on use cases with expert estimates. In Proceedings of Empirical Assessment in Software Engineering (EASE 2002) (Keele, UK, April 8-10, 2002)
46. Anda, B., Dreiem, D., Sjøberg, D.I.K., and Jørgensen, M.: Estimating Software Development Effort Based on Use Cases - Experiences from Industry. In M. Gogolla, C. Kobryn (Eds.): UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, 4th International Conference, Toronto, Canada, October 1-5, 2001, LNCS 2185, Springer-Verlag, pp. 487-502.
47. Anda B., et al.: Effort Estimation of Use Cases for Incremental Large Scale Software Development, 27th International Conference on Software Engineering, St Louis, MO, 15-21 May 2005: 303-311.
48. Lavazza L., Robiolo G.: The role of the measure of functional complexity in effort estimation, In: PROMISE '10 Proceedings of the 6th International Conference on Predictive Models in Software Engineering, 2010
49. Abran A., Moore J.: Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004. ISBN: 0769523307

**Jovan Popović** received a MsC degree in electrical engineering and computer science from the University of Belgrade in 2010. He is a software practicioner specialized in project management and web application development. The focus of his research is application of software metrics and business inteligence in the software development process.

**Dragan Bojić** received a PhD degree in electrical engineering and computer science from the University of Belgrade in 2001. He is an assistant professor at the Faculty of Electrical Engineering, University of Belgrade. His research interests focus on software engineering techniques and tools, and e-learning.

*Received: March 16, 2011; Accepted: December 16, 2011.*