Two Novel Effort Estimation Models Based on Quality Metrics in Web Projects

Ljubomir Lazić
Department for Mathematics and Informatics
State University of Novi Pazar
SERBIA
llazic@np.ac.rs, http://www.np.ac.rs

Nikos E Mastorakis
Technical University of Sofia,
English Language Faculty of Engineering
Industrial Engineering, Sofia, BULGARIA
http://www.wseas.org/mastorakis

Abstract: - Web development projects are certainly different from traditional software development projects and, hence, require differently tailored measures for accurate effort estimation. Effort estimation accuracy will affect the availability of resource allocation and task scheduling. In this paper, we investigate the suitability of a newly proposed quality metrics and models to estimate the effort and duration for small or medium-size Web development projects. It then describes a new size metrics, presents two novel methods (WEBMO+ and VPM+), based on WEB model (WEBMO) using Web objects instead of SLOC and Vector Prediction Model (VPM), to fast estimate the development effort of Web-based information systems. We also empirically validate the approach with a four projects study. The results indicate that the approach provides a mechanism to measure the size of software systems, classify software systems, and estimate development effort early in the software life cycle to within +/-20 percent across a range of application types. In contrast with other existing methods, WEBMO+ and VPM+ uses raw historical information about development capability and high granularity information about the system to be developed, in order to carry out such estimations. This method is simple and specially suited for small or medium-size Web based information systems.

Key-Words: - Web Quality, Effort Estimation Method, Sizing Metric, Web-based Metrics, Web Engineering.

1 Introduction

Web development projects are certainly different from traditional software development projects and, hence, require differently tailored measures for accurate effort estimation.

Measurements have been widely used in software engineering to estimate the effort, time and cost, to predict error proneness of modules and assess the quality of software design. Our research [7][1] concluded that developing software is for most organizations no longer an independent software project, but is part of a business case which includes all disciplines involved. This means that the cost of building the software must be balanced by a profit somewhere else in the organization. So organizations want to have a good estimate of the effort of developing and/or delivering the software as early as possible.

Most estimators would like to use the more traditional processes, metrics, models, and tools for estimating Web projects. They are mature, and many of us in the field have confidence in their ability to accurately predict project costs and schedules. We also have a great deal of experience using these metrics, models, and tools and feel comfortable with them and their outputs.

It is widely accepted that among a variety of factors that potentially affect the cost of software projects, software size is one of the key cost drivers [1,2,8,10]. Unfortunately, there is no common way of measuring software size. Software development is an evolving process with often-changing technologies, development methods and tools, as well as often changing requirements.

A new type of software, so-called web applications, has been established in the business world in recent years. With the growing importance of web applications in general business matters, many practitioners see the measurement of web applications as a particularly valuable area of ongoing commercially relevant research. Thus, web development has become the focus of research interest as well. When comparing web development with traditional software development we can identify differences [1,3-7,9].

These differences comprise the software development methods and technologies, as well as

[1] This work was supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Grant No. TR-13018.

the development team and the project conditions. Usually high time pressure and very volatile software requirements play a central role in a web development project. Because of the identified differences, it is necessary to revise and adapt the commonly used software engineering methods and measures before applying them to the application domain of web development [1,3,6]. New size metrics are needed to cope with Web objects like shopping carts, Java scripts, and building blocks like Cookies, ActiveX controls, and Component Object Model components. In this paper, we investigate the suitability of a newly proposed quality metrics and parametric models to estimate the effort and duration for small or medium-size Web development projects.

Parametric web application cost estimation is referred to the usage of mathematical model to derive the estimated effort and duration of web application development. Typically, majority of web application developers are applying expert judgment and estimation by analogy in web application development.

New duration-estimating equations are needed to address the fact that the cube root laws used by most estimating models just do not seem to work for the Web projects. Because Web cost can be treated as a function of size, a meaningful size predictor is needed for Web projects. Those working such projects agree that the popular size metrics, function points (FP) and source lines of code (SLOC), are not suitable for Web estimation because they do not take all of the Web objects (buttons, templates, etc.) into account. In this paper, we investigate the suitability of two novel methods (WEBMO+ and VPM+), based on WEB model (WEBMO) using Web objects instead of SLOC [1,3,4,6] and Vector Prediction Model (VPM) [5], to fast estimate the development effort of Web-based information systems. The "Vector-Based Approach" introduced by Hastings and Sajeev [5], is based on two concepts: "Vector Size Measure (VSM)" to size the system and "Vector Prediction Model (VPM)" to estimate the corresponding effort.

We also empirically validate the approach with a four projects study. The results indicate that the approach provides a mechanism to measure the size of software systems, classify software systems, and estimate development effort early in the software life cycle to within +/-20 percent across a range of application types. In contrast with other existing methods, WEBMO+ and VPM+ uses raw historical information about development capability and high granularity information about the system to be developed, in order to carry out such estimations. This method is simple and specially suited for small or medium-size Web based information systems.

2 Literature Survey

Companies developing Web-based systems face the challenge of estimating the required development effort in a very short time frame. This problem does not have a standard solution yet. On the other hand, effort estimation models that have been used for many years in traditional software development are not very accurate for Web based software development effort estimation [1]. Web-based projects are naturally short and intensive [2,4,7], so not having an appropriate effort estimation model pushes developers to make highly risky estimations.

Moreover, the rapid evolution and growth of Web related technology, tools and methodologies makes historical information quickly obsolete. Although, the software effort estimation process is a completely necessary and critical task, it still looks more like a craft than a science [2,3,6]. The process is mainly dependent of the project type and the features of the development scenario. Without an appropriate model, cost estimation is done with a high uncertainty and the development effort estimation relies only on the experience of an expert, whose estimations are generally not formally documented. From the beginning of software engineering as a research area more than three decades ago, several development effort estimation methods have been proposed. We can classify these methods for our research as those for traditional software and those for Web oriented software. The traditional effort estimation methods are those used to estimate the development effort of software that consists of programs in a programming language, which eventually interact with data files or databases. Generally, this software has an active execution thread that provides system services. On the other hand, the Web-oriented methods use different metrics and they are focused on estimating the development effort of products that are event-oriented. These products generally involve code in a programming language, imagery, look-and-feel, information structure, navigation and multimedia objects.

The research is exploratory in nature. It was designed to be conducted one after another based on the result from the earlier survey. There are suggestions for a general approach to designing case studies, and also recommendations for exploratory, explanatory, and descriptive case studies [1,3,6,11]. It can be either single or multiple-case studies,

where multiple case studies are replication, not sampled cases [3,5].
When no other cases are available for replication, the researcher is limited to single-case designs. In this research, we applied research method as depicted in Figure 1. This figure represents the software cost estimation in research domain of knowledge since the major constraint is to get the data from local companies that are focusing on web application development, replication is hardly possible for multiple-cases studies. Thus, the case study design for this research is single-case study design. There is no duplication for the case studies as each respondent has different web application project specifications although they have similar application types.

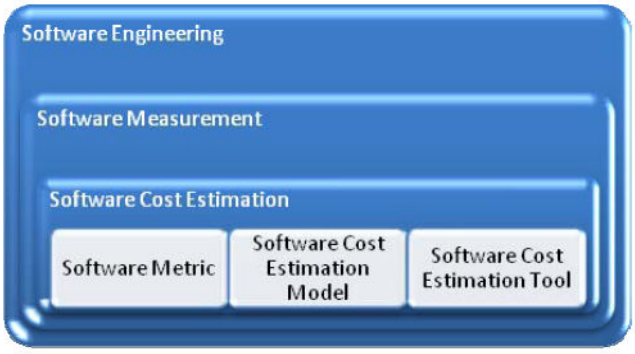

Fig. 1 Research Domain of Knowledge [2]

Traditional effort estimation methods like COCOMO [2,8] are mainly based on metrics like Lines Of Code (LOC) or Function Points (FP) . The estimation strategies supported by LOCs have shown several problems. Most working Web projects agree that LOCs are not suitable for early estimation because they are based on design [3]. Other reported problem is that the work involved in the development of multimedia objects and look-and-feel cannot be measured in LOCs.

Also, an important amount of reliable historical information is needed to fast estimate effort using this metric, and this information is hard to get in small or medium-size Web-based projects (S&MWEBPROJ) scenario.

Finally, to carry out estimations using LOCs requires a long analysis of the historical information, which reduces the capability to get reliable fast estimations. Speed is an important requisite of Web-based projects developed in our case.

Similarly, traditional estimation methods based on FPs are not appropriate because applications do more than transform inputs to outputs, i.e. the effort necessary for developing a Web-based application is much more than the effort required for implementing its functionality. FPs does not consider the imagery, navigation design, look-and feel, and multimedia objects, among others. In other words, the traditional categories of FPs should be redefined. This kind of estimation also requires an important amount of reliable historical information, which supports the used values of each FPs.

Although there are several software effort estimation methods, like: Price-S, Slim and Seer [8], COCOMO is the most well known and used by the software industry. It had shown to be appropriate in many development scenarios. The first version of such method used LOCs as the fundamental metric to support the estimations. Then, Boehm proposed COCOMO II, which could use alternatively LOCs, FPs or Object Points. Although COCOMO II was not defined to support the development effort estimation of Web applications, many people found the way to adapt the object point concept in order to get a sizing estimation [8]. Object points are an indirect metric, similar to FPs, which considers three categories: user interfaces, reports and components, which are probably needed to develop the final product. Every element in the system is categorized and classified in three complexity levels: basic, intermediate and advanced. Then, based on these classified elements, and taking into account the historical information, it is possible to generate a good estimation. Object Points and COCOMO II seem to be acceptable for traditional or multimedia software projects, but they are not good enough to get accurate effort estimations for Web-based information systems developed in S&MWEBPROJ scenario. The complexity of the estimation process and the need for detailed historical information make them difficult to apply in this scenario.

Several size metrics have been proposed for Web applications, like Object Points, Application Points and Multimedia Points [9]. However, the most appropriate seems to be Web Objects (WO) from Reifer [3]. WOs are an indirect metric that is based on a predefined vocabulary that allows defining Web systems components in term of operands and operators. To estimate the amount of WOs that are part of a Web-based application it is necessary to identify all the operators and operands present in the system. Then, they are categorized using a predefined table of Web Objects predictors and also they are classified in three levels of complexity: low, average and high. The final amount of WO in a Web-based application is computed using the Halstead equation [2], and it is known as the volume or size of the system. The effort estimation and the duration of the development are computed using WEBMO (Web Model), which is an extension of COCOMO II [3]. This model uses two constants, two power laws, several cost drivers, and the

product size expressed in WO (see Figure 2). Constants A and B, and power laws P1 and P2 are defined by a parameter table in the model. This table contains the values obtained from a database of former projects (historical information). The cost drivers are parameters used to adjust the effort and duration in terms of the development scenario. For this model *nine* cost drivers were defined: product reliability and complexity (RCPX), platform difficulty (PDIF), personnel capability (PERS), personnel experience (PREX), facilities of tools and equipment (FCIL), scheduling (SCED), reuse (RUSE), teamwork (TEAM) and process efficiency (PEFF) [3]. Each cost driver has different values that may be: very low, low, normal, high, and very high. The combination of WEBMO and Web Objects is, in this moment, the most appropriate method to estimate the development effort of Web applications. However, this combination does not seem to be the best for broad range of web development scenarios because it needs an important amount of historical detailed information to carry out the estimation. Also, the WO identification and categorization process is difficult to carry out in a short time, and it requires an expert that also knows how to carry out the project. As shown in Figure 2, the WEBMO estimating equations for effort (in person-months) and duration (in calendar months) assume size is provided in Web objects. To predict duration, the model assumes a square root instead of a cube-root relationship between duration and effort for small projects.

$$\text{Effort} = A \prod_{i=1}^{9} cd_i \, (\text{Size})^{P1} \qquad \text{Duration} = B(\text{Effort})^{P2}$$

Where: A and B = constants; cd_i = cost drivers
P1 and P2 = power laws; Size = # KSLOC

Fig. 2 WEBMO Effort Estimation Model

The current version of the WEBMO estimation model differs from the original COCOMO II model by having *nine* instead of *seven* cost drivers and a *fixed* instead of *a variable* effort power law. The expert should be competent about critical technical decisions of the development process because in small projects the technical feasibility study is implicitly included in this estimation. This expert feature and the complexity to identify and classify WO make WEBMO unfeasible to use in our fast estimate the development effort of Web-based information systems scenarios. The main weakness in this study is that Reifer did not publish any detailed empirical results that are needed to prove his claim and *assumption that size is provided in Web objects instead in SLOC* as we see from Fig. 2. Provided that the effort estimation methods presented are no appropriate to estimate the development effort of Web-based information systems in our scenarios, in the next section we present the WEBMO+ method which exceeds mentioned WEBMO main weakness. It intends to be more appropriate to estimate the development effort of small or medium size projects, especially in scenarios that require fast estimation with little historical information.

As we mentioned above The "Vector-Based Approach" introduced by Hastings and Sajeev [5], is based on two concepts: "Vector Size Measure (VSM)" to size the system and "Vector Prediction Model (VPM)" to estimate the corresponding effort of wide range of software systems. The approach attempts to measure the system size from the algebraic specifications described in the Algebraic Specification Language (ASL). The algebraic specifications are based on abstract data types (ADTs) and ASL provides a mathematical description of the system. The software size measure they proposed has two principle attributes: *functionality* and *problem complexity*. Functionality represents the services provided by a software system to its clients. Problem complexity represents the underlying semantics (meaning) of a software system. Intuitively, both users and developers of software systems refer to the functionality a system provides and the underlying complexity of the problems a system solves.

The approach accepts Fenton's multidimensional definition of size [10]. To measure VSM, first system functionality and complexity is calculated as Fenton [10] stated: "there appear to be three such [orthogonal and fundamental] attributes of software: *length, functionality, and complexity* of the underlying problem which the software is solving." In the VSM measure, length is a derived attribute. We show this formally below. Then, the system length is derived from these values. Similar to Halstead's method, the ADT properties are defined in terms of operators and operands.

As the expert should be competent about critical technical decisions of the development process, because in small projects, the technical feasibility study is implicitly included in this estimation. This expert feature and the complexity to identify and classify software: *length, functionality, and complexity* of the Web-based information systems make Vector Prediction Model (VPM) unfeasible to use in our fast estimate the development effort of the S&MWEBPROJ scenarios. As in case of WEBMO

in the next section we present the VPM+ method which exceeds mentioned VPM main weakness.

2.1 Effort Estimation Scenario

The S&MWEBPROJ, our fast estimate the development effort estimation scenario is not completely different from other scenarios, as shown on Fig. 3, but due to some of its characteristics, well-known effort estimation methods have low applicability. The central characteristics are the following:

Expert centered. An expert is in charge of the effort estimation process, who knows the development capability of the organization. He/she is responsible for calculating the budget for clients and to establish development compromises. The expert's experience is very useful for the company but not necessarily good for other companies.

Little historical information. Development companies generally have little historical information about past projects. Usually, they have the products produced by the project and an approximation of the total amount of man-hour spent on it. Besides, such information is usually unorganized and it is perceived by the expert as not very reliable because of its incompleteness.

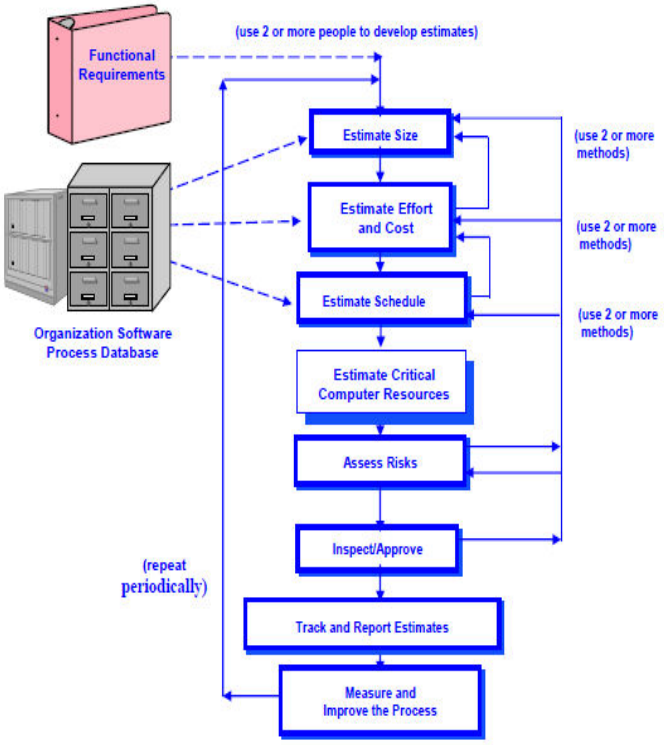

Fig. 3 Total Estimation Life Cycle

Short time to estimate. Development companies generally have between 24 and 72 hours to estimate the development effort of small or medium sized projects, from the moment the client provides the Web application requirements and information. In that time, the experts should analyze and clarify the information provided by the client, revise historical information, carry out the estimation, and build the budget.

Gross gained information. The information given by the client about the problem to be solved tends to be gross grained and could have unclear areas for the client. The client wants the estimation to include some flexibility to adjust such unclear issues. Therefore, a lot of experience is required from the estimator to dimension the system realistically only based on high level information.

Quite fixed development time. Generally, the development projects arise as response to immediate needs of the clients. Therefore, the development time for a project is quite fixed and the estimation process becomes a problem of feasibility and/or money.

Budget centric. The S&MWEBPROJ scenario is "budget centric". Typically, the clients request budgets to several development companies and usually the lowest development cost is chosen. This means that the developer should prepare budgets for many clients which could make the experts to become a bottleneck. On the other hand, work of such experts is an investment with low probability of success, because several budgets are requested and only one is chosen.

These features make the effort estimation scenario be highly demanding and little motivating. The well-known effort estimation models have shown low applicability to support this process; therefore, it is currently carried out in a handmade way. Next Section presents and discusses our Adaptive Cyclic Pair-wise Rejection Rules (ACPRR) of Estimation model candidates for fast estimate the development effort of the S&MWEBPROJ scenarios using established performance measures.

2.2 Performance Evaluation Criteria of various Effort Estimation Models

There are relatively few examples in the literature that address the assessment of cost estimation models, generally, and for web applications [7,11,12].

We consider COCOMO II, FP, UCP, WEBMO and VPM estimation model candidates using standard performance measures as many authors do like Menzies [11] and Nagappan [12]. The rejection rules are the core of our fast estimate the development effort of the S&MWEBPROJ scenarios workbench. In our own work, we revised the COSEEKMO effort estimation workbench for

assessing different estimation models based upon COCOMO data sets using data mining techniques despite large deviations rules from [11,12] until they satisfied certain sanity checks. Large deviations in effort estimation area require model performance comparison using some heuristic rejection rules that compare more than just mean performance data. The results for each treatment were compared using each treatment's of MMRE, SD, PRED(N) and correlation. MMRE comes from the magnitude of the relative error, or MRE, the absolute value of the relative error:

$$MRE = |predicted - actual| / actual \quad (1)$$

The mean magnitude of the relative error, or MRE, is the average percentage of the absolute values of the relative errors over an entire data set. Given T tests (estimators), the MMRE is:

$$MMRE = \frac{100}{T}\sum_{i}^{T}\frac{|predicted_i - actual_i|}{actual_i} \quad (2)$$

The standard deviation, or SD, is root mean square of predicted and actual effort deviation. Given a T tests (estimators), the SD is:

$$SD = \frac{100}{T}\sum_{i}^{T}(predicted_i - actual_i)^{1/2} \quad (3)$$

The PRED(N) reports the average percentage of estimates that were within N% of the actual effort candidates' estimated values. Given T test examples, then:

$$PRED(N) = \frac{100}{T}\sum_{i}^{T}\begin{cases}1 \cdot if \cdot MRE_i \le \frac{N}{100}\\ 0 \cdot otherwise\end{cases} \quad (4)$$

For example, a PRED(20)=50% means that half the estimates are within 20% of the actual effort data.

Another performance measure of a model predicting numeric values is the correlation between predicted and actual values. Correlation ranges from +1 to -1 and a correlation of +1 means that there is a perfect positive linear relationship between variables.

We choose estimating model candidates according to our reading of the literature (e.g. [1-10], only those which satisfy starting criteria (SC) defined by:

$$MMRE \le 20\% \text{ and } SD \le 20\%, \text{ and } ACPRR \quad (5)$$

This stratification of estimator candidates improve performance, then this estimator subsets of the estimators performance data [11,12] should usually generate better models with lower error rates than models learned from all the collected estimators performance data.

Adaptive Cyclic Pair-wise Rejection Rules for Estimation model performance evaluation (ACPRR)

The Estimation model treatments were examined in pairs and if either seemed to perform worse, that one was rejected. This process repeated until no treatment could be shown to be worse than any other. The remaining treatments were called the survivors and were printed.

In addition to these sanity checks, the following model evaluation rules were used in this study:

• *Rule 1* is a statistical test condoned by standard statistical textbooks. If a two-tailed t-test reports that the mean of two treatments is statistically different, then we can check if one mean is less than the other.

• *Rule 2* (which checks for correlation) is a common check used: the best effort model tracks well between predicted and actual values. Without rule 2, many parts of our data produce multiple survivors.

• *Rule 3* is added in case there is more than one model with the same average performance and R^2. The model with the smallest MMRE is selected using equation (1).

• *Rule 4* was added because PRED(20)=50% using equation (4) is our required performance measure for effort models.

• *Rule 5* This rule rejects treatments that have similar performance, but use more attributes. Different rules are applied because they measure different aspects of model performance. Figure 4 shows adaptive cyclic pair-wise rejection rules algorithm using standard statistical tests on MMRE, where an **error** is MMRE, **worse** function that apply **statistically different** criteria comparing two MMREs ***x*** and ***y,*** using, well known from statistics, a two-tailed t- test at the 95% confidence interval, i.e.

$$\frac{|mean(x) - mean(y)|}{\sqrt{\left(sd(x)^2/n(x)-1)\right) + \left(sd(y)^2/n(y)-1)\right)}} > 1.96$$

3 Adaptation of WEBMO and VPM Effort Estimation Models for Web Projects

We consider WEBMO and VPM, in this moment, the most appropriate estimation model candidates using proposed Adaptive Cyclic Pair-wise Rejection Rules (ACPRR) for fast estimate the development effort of the S&MWEBPROJ scenarios.

```
function worse(x,y)
  if   statisticallyDifferent(x,y)
  then
       if error(x) < error(y)                    then return y fi # rule1
       if error(y) < error(x)                    then return x fi # rule1
  else
       if correlation(x) < correlation(y)        then return x fi # rule2
       if correlation(y) < correlation(x)        then return y fi # rule2

       if sd(x)/mean(x)< sd(y)/mean(y)           then return y fi # rule3
       if sd(y)/mean(y)< sd(x)/mean(x)           then return x fi # rule3

       if pred(x)      < pred(y)                 then return x fi # rule4
       if pred(y)      < pred(x)                 then return y fi # rule4

       if |Subset(x)|  < |Subset(y)|             then return y fi # rule5
       if |Subset(y)|  < |Subset(x)|             then return x fi # rule5
  fi
  return 0  # if no reason to return true
```

Fig. 4 Adaptive Cyclic Pair-wise Rejection rules algorithm (ACPRR)

3.1 WEBMO+ is adapted WEBMO

We measured system size using both Function Points and Web Objects [3,7]. Web Objects are an extension of the traditional Function Points addressing the needs of web development as depicted on Fig. 5. So far, to our knowledge, no other alternative size measure for web applications is published or initially validated.

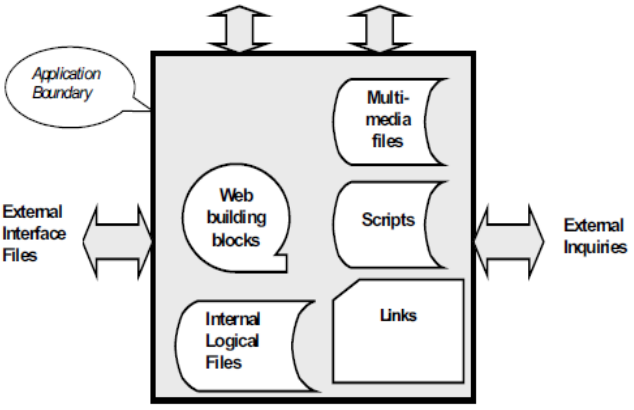

Fig. 5 Web Object Components

In general, the number of Function Points is lower than the number of Web Objects. But when comparing the number of Function Points and Web Objects project by project, the size difference between the two measures is up to 55%, i.e. the number of Web Objects is twice as high as the number of Function Points. This sizing difference may have a large effect on the effort estimation accuracy. Main reason is that Reifer [3] did not publish any detailed empirical results that are needed to prove his claim and *assumption that size is provided in Web objects instead in SLOC* as we see from Fig. 2.

The constants *A,B* and parameters *P1* and *P2*, in the effort and duration equations and power laws for each of the five application domains that we have studied are summarized in Fig. 6.

Application Domain	A	B	P1	P2
Web-based electronic commerce	2.3	2.0	1.03	0.5 or 0.32
Financial/trading applications	2.7	2.2	1.05	0.5 or 0.32
Business-to-business applications	2.0	1.5	1.00	0.5 or 0.32
Web-based portals	2.1	1.8	1.00	0.5 or 0.32
Web-based information utilities	2.1	2.0	1.00	0.5 or 0.32

Fig. 6 WEBMO Parametric Values (Table 5 in [3])

A brief explanation for each of the nine cost drivers (cd_i) used by the model is provided in Table 6 (see page 16) in [3]. The values for the driver ratings used in the model are also provided in Table 6 in [3].

SLOC is calculated using average and subjective the Language Expansion Factors (LEF) listed in Table 7 in Reifer's article. Please note that the values in this table differ from those currently endorsed by the International Function Point Users Group (IFPUG). Using next equation, we exceed this WEBMO weakness:

$$SLOC=272.11 * (\#External\ Use\ Cases)^{1.709} \quad (6)$$

Which is equation (1) in Y. Chen [5] paper. As shown in Table 5 in Chen's [5] paper, the SLOC vs. External Use Cases is the strongest correlation because it has the highest value of R^2= 0.538. Furthermore, the polynomial form of the size function provides the best fit in all of the cases but one. However, there is no strong behavioral explanation for a polynomial relationship. The equation form that best represents the relationship between SLOC and Number of External Use Cases (#EUC) based on of 14 small *eServices* projects for which they have data on number of requirements, number of UML artifacts, and numbers of SLOC in various languages.

3.2 VPM + is adapted VPM

We elaborated and validated a Vector Size Measure (VSM) and Vector Prediction Model (VPM) [5]. The purpose of VSM is to measure the size of software systems and to classify software systems. The purpose of VPM is to estimate development effort early in the SLC. The results in T.E. Hastings [5] work indicate that the approach provides a mechanism to measure the size of software systems, classify software systems, and estimate development effort early in the software life cycle to within +/-20 percent across a range of

application types, so VPM model satisfy our starting criteria (5) in Section 2.2.

The software size measure VSM propose has two principle attributes: *functionality* and *problem complexity*. Functionality represents the services provided by a software system to its clients. Problem complexity represents the underlying semantics (meaning) of a software system.

Intuitively, both users and developers of software systems refer to the functionality a system provides and the underlying complexity of the problems a system solves.

Shortly, we will see how these intuitive attributes are represented with numbers.

VSM have adapted Halstead's [2] concept of operators and operands as OPs, where: 1) Operators map to function references and rule connectors ("|" and "=") and 2) Operands map to generic parameters, parameters, and free variables. Halstead [2] used his approach to measure code in the solution domain. We have mapped his principle to a higher level abstraction of Web-based software specification in the problem domain. The *functionality* of an ADT is defined as the distinct number of syntactic properties measured by the sum of OPs in the syntactic section of an ADT. Thus, the functionality of an abstract data type, A, measured in OPs is:

$$f_A = \sum OP_{Fi} \text{ , [OPs]}$$

The problem *complexity* of an ADT is defined as the distinct number of no redundant semantic properties measured as the sum of OPs in the semantic section of the ADT. Thus, the problem complexity of an abstract data type, A, measured in OPs is:

$$c_A = \sum OP_{Ci} \text{ , [OPs]}$$

The *length* of an ADT is defined as the sum of atomic units specified in the syntactic and semantic sections of an ADT. Thus, the length of an abstract data type, A, measured in OPs is:

$$l_A = \sum OP_{Ai} = \sum OP_{Fi} + \sum OP_{Ci} = f_A + c_A \text{ , [OPs]}$$

Size Formulae

The size of an abstract data type, S_A, is defined by the tuple, (f_A; c_A), such that:

$$S_A = (f_A, c_A) \text{ , [OPs]}$$

The size of a software system (subsystem or component), S_S , is defined as the sum of the size of all ADTs, S_N , that specify the system (subsystem or component), such that:

$$S_s = \sum S_{Ni} \text{ , [OPs]}$$

Given that we have two fundamental software size attributes, i.e., functionality and problem complexity, we can represent software size as a two-dimensional vector which has both magnitude and direction. This representation allows us to understand and transform software size measurements using well-defined mathematical functions. Fig. 7 illustrates the concept.

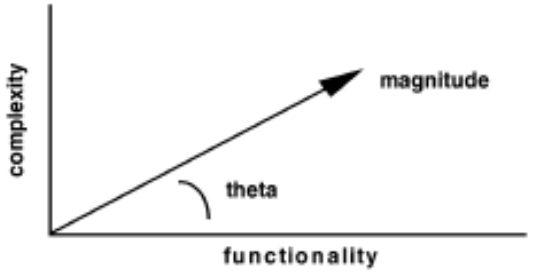

Fig. 7 Software size as a vector

Using plain vector algebra, magnitude, *m*, is defined as:

$$m = \sqrt{f^2 + c^2} \text{ , [OPs]} \quad (6)$$

The magnitude ***m*** is a measure that takes problem complexity and functionality into account in a balanced and orthogonal manner. There are many effort prediction techniques that could be adopted, including expert opinion, analogy, decomposition, and models [2]. VPM is based on a model. VPM is based on a cost model. The primary and secondary inputs to VPM are the **m** - the magnitude and gradient measures, respectively. There is no guess work to VPM, such as rating application characteristics. VPM uses a multivariate regression model to determine the relationships between effort, magnitude, and gradient. Equation (7) illustrate the linear regression (best fit) for Magnitude vs. Effort with the highest value of R^2= 0.729 correlation.

$$Effort = 3.5628 * m + 241.61 \quad (7)$$

Again, as in WEBMO case, VPM calculation process is to complex, require the competent expert of the Web-based information systems that make Vector Prediction Model (VPM) unfeasible to use in our fast estimate the development effort of the S&MWEBPROJ scenarios. Using next easy procedure, for ***m*** **-** Magnitude of software size calculation in S&MWEBPROJ scenarios, we exceed this VPM weakness.

Procedure for magnitude ***m***, calculation is:

$$m = AverageWO_{Weights} \quad (8)$$

Web objects (WOs) are identified in Web Project, the transactions and complexity with assigned weights for every WO is calculated in standard way

described in literature [2]. Similar procedure we use for #EUC calculation in WEBMO+ model, described in next section.

4 Empirical Validation

Within our study we elaborated VEBMO and VPM work by developing a counting manual to support the systematic and consistent measurement of Web Objects (WEBMO+ and VPM+), to satisfy our fast estimate the development effort of the S&MWEBPROJ scenarios. The manual was completed by company-specific counting examples to facilitate future applications.

Empirical validation confirms measures are useful and collaborates that measured values are consistent with predicted values [1-7]. Therefore, actual data is required which can then be compared to estimates.

4.1 Case Studies for WEBMO+ and VPM+ Web Effort Models Assessment

This study is based upon the past data collected from the twelve different Web projects, taken from the completed by the post graduate students course of "Software Engineering" (SE) at State University of Novi Pazar, during last three years.

Average collected data from these Web projects are summarized in Table 1 under Project ID No. 1 and Table 2. It is assumed that all the data of twelve different project requests studied is correct without any deviation. The model is then refined from the data collected from more than the 100 different Web projects from literature [3,4].

For Size calculation, we have opted for the Object Point Analysis. Object Points are a measure of the size of computer applications and the projects that build them. The size is measured from a components point of view. It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application.

Table 1 – *WebMetrics Data Collected from all Projects*

Project ID	Actual SLOC	#EUC	Estimated SLOC	Size magnitude m
1	9765	7.67	8848	1110
2	11392	9	11629	1068
3	5262	5.86	5586	526.2
4	30000	15	27841	3614.4

The data for Project 2 we used data from Reifer's work [3], shown on Table 3.

Table 2 – *WebMetrics Data Collected from students works on SE course, Project 1*

Function	Total Web objects
Outputs	0
Inquires	112
Inputs	2
Internal Files	31
External	0
#Multimedia	1
#Web building	7
#Scripts	33
#Links	184
Total Web points	**370**

For Project 3, we used average data in Y. Chan work [4] shown on Table 4. Finally, for Project 4, we used data presented in Table 5, from The CSCI 577 software engineering class slides at USC. The CSCI 577 is a 2 semester course which provides the opportunities for graduate students to experience the software life-cycle by working on real projects with real clients. Most projects are *eService* applications providing web-based services to campus users.

Table 3 - *Web Object Calculation Worksheet[3]*

Web Object Predictors	Low	Average	High	Notes
Traditional function point predictors				
• Internal logical files		2x10	1x15	From specification: 3 files
• External interface files		2x7		From specification: 2 interfaces
• External inputs		4x4	6x6	From specification: 10 inputs
• External outputs		3x5		From specification: 3 outputs
• External inquires				From specification: none
Number of XML, HTML, and query lines		16x4		From specification: 16 HTML lines
Number of multimedia files	1x4 3x4	13x5	1x7	Operands: audio file, 13 multimedia files, help file Operators: open, close, save
Number of scripts	3x2	1x3		Operands: animation script Operators: open, go (forward), close
Number of Web building blocks	3x3	10x4	5x6	Operands: 15 building blocks from library including 9 buttons, 1 cast, and 5 secure server icons Operators: find, add, and insert
TOTAL	31	237	88	

Each project is typically assigned to a 5-graduate-student team. The student developers are required to follow the Model Based (System) Architecting

Software Engineering (MBASE) Guideline [4] as a process guideline to complete their project.

Table 4 - *Typical eService Project [4]*

Feature	Description
Domain	eServices
Team Size	5 graduate students
Duration	24 weeks
Process Guideline	MBASE
Architecture Description Language	UML
Average Number of External Use Cases	5.86
Average # of Logical SLOC	5262

5 Effort Estimation WEB+ and VPM+ Results

Using described WEBMO+ and VPM+ Effort Estimation procedure, accuracy of these Web Project Estimation Models is given on Table 6.

The results indicate that the approach provides a mechanism to measure the size of software systems, classify software systems, and estimate development effort early in the software life cycle to within +/-20 percent across a range of application types. In contrast with other existing methods, WEBMO+ and VPM+ uses raw historical information about development capability and high granularity information about the system to be developed, in order to carry out such estimations. This method is simple and specially suited for small or medium-size Web based information systems.

6 Conclusion

Estimating the cost and duration of Web developments has a number of challenges associated with it. In short, there is no silver bullet in web application cost estimation in today's globalization edge. Every web application project would have different work breakdown structures with different activities. Adaptability and sensitive to customer requirements are vital to ensure time to market and quality of the web application delivery [2,13]. Each web application project in an organization would have different team of web developers. Even similar projects might have different duration based on different resources assigned to the activities with respective competency. Understand customer requirements and deliver the web application within the stipulated duration and cost are web developers' ultimate objective.

To cope with these challenges, we developed new size metrics, Web objects (WOs), and two novel estimating models WEBMO+ and VPM+. We have also validated and calibrated the metric and model in anticipation of building potential products based upon them for fast estimate the development effort of the S&MWEBPROJ scenarios.

The following significant results/findings were outputs of our initial research efforts:

• We validated that Web objects have better predictive accuracy (R^2) than traditional function points when counted using conventions developed for that purpose. These counting conventions allowed us to extend the excellent work done by the IFPUG so that we can better handle the sizing of Web applications.

Table 5 - *Typical Web Metrics data collected in eService Project [4]*

Use Cases	Transactions	Type	Complexity	UFP
End Users				
Logon	1	I	3	3
View Last Bill	1	Q	6	6
Create Account	1	I	6	6
View Current Services	1	Q	4	4
Establish Analog CATV Service	1	I	6	6
Add Data Service	1	I	6	6
Add/Delete a Premium Channel	1	I	4	4
Add/Delete a Digital Package	1	I	6	6
View Trouble Status	1	Q	4	4
View Order Status	1	Q	3	3
View Information	5	Q	3	15
BackEnd				
Get Account & Service Info	1	N	10	10
Get Last Bill	1	N	10	10
Create Account	1	N	10	10
Create Order	1	N	10	10
Account Validation	3	N	7	21
Order Validation	3	N	7	21
Get Trouble Status	1	N	7	7
Get Order Status	1	N	7	7
Management				
View Customer Use Statistics	5	Q	4	20
Troubleshoot Customer Scenario	5	Q	6	30
OA&M				
User Administration	2	F	7	14
Table Administration	15	F	7	105
Usage DB Administration	1	F	15	15
Temp DB Admin	1	F	15	15
Schedule Reports	1	I	4	15
Control Application	1	I	4	15
Create Reports	1	I	6	15
Application Alarms	1	O	7	15
Total Unadjusted Function Points				418

• We increased the statistical accuracy of our WEBMO+ estimating model from more than MRE=30% of effort estimation to MRE=5% and more than MRE=30% of the actual experience in project duration estimation to MRE=10% using a more than 100-project database of our and other published actual [1-5].

• We validated that a square root instead of a cube-root relationship exists between effort and schedule for Web application projects whose size was less

than 300 Web objects. These results are substantial because they indicate that the Web objects and the WEBMO+ estimating model can help address the gaps in the estimating technology.

• We increased the statistical accuracy of our VPM+ estimating model from more than MRE=30% of effort estimation to MRE=7.7% and more than MRE=30% of the actual experience in project duration estimation to MRE=16% using a more than 100-project database of our and othert published actual [1-5].

Table 6 - *Effort Estimation WEB+ and VPM+ Results*

Proj. ID	Actual Effort [m-h]	Model WEBMO+		Model VPM+		Actual Project Duration [months]	Model WEBMO+		Model VPM+	
		Estim. Effort [m-h]	MRE [%]	Estim. Effort [m-h]	MRE [%]		Actual Project Duration [months]	MRE [%]	Actual Project Duration [months]	MRE [%]
1	29.12	26.64	8.5	27.07	7	7.3	6.3	13.7	6.3	13.4
2	24	23.92	0.3	26.11	8.8	5	5.52	10.4	5.7	13.6
3	15	16.45	9.7	13.65	9	6	6.62	10.3	5.1	15.4
4	90	88.77	1.4	84.64	6	12	11.34	5.5	9.1	21.7
MMRE [%]			5.0		7.7			10.0		16.02

References

[1] E. Mendes, "Estimation techniques for web projects", ISBN 978-1-59904-135-3, by IGI Global, 2008.

[2] S. H. Kan,"Metrics and Models in Software Quality Engineering", Second Edition, Addison-Wesley, 2003.

[3] D. Reifer,"Estimating Web Development Costs:There Are Differences", *CrossTalk,* June 2002.

[4] Y. Chen et all, "An empirical study of eServices product UML sizing metrics", Empirical Software Engineering, 2004. ISESE '04, pp. 199-206.

[5] T.E. Hastings i ostali, "A Vector-Based Approach to Software Size Measurement and Effort Estimation", *IEEE Transactions On Software Engineering, Vol. 27, No. 4, April* 2001.

[6] M. Ruhe, R. Jeffery, and I. Wieczorek, "Cost Estimation for Web Applications", In Procs. Intl. Software metrics Symposium (METRICS'03), 2003, pp30-37.

[7] Lj. Lazić, N. Mastorakis." The COTECOMO: COnstractive Test Effort COst Model", Proceedings of the European Computing Conference, © Springer,Volume 2, Series: Lecture Notes in Electrical Engineering, Vol. 28, Mastorakis, Nikos; Mladenov, Valeri (Eds.), ISBN: 978-0-387-84818-1, May 2009, pp 89-110.

[8] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. Winsor Brown, S. Chulani and C. Abts, "Software Cost Estimation in COCOMO II", Prentice-Hall, 1stedition, January 2000.

[9] A. J. C. Cowderoy, "Size and Quality Measures for Multimedia and Web-Site Production". Proc. of the 14th Int. Forum on COCOMO Modeling, Los Angeles, CA, October 1999.

[10] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, second ed. London: PWS, 1997.

[11] T. Menzies, Z. Chen, J. Hihn, K. Lum „Selecting best practices for effort estimation", *IEEE Trans. on Soft. Eng. 32(11),* 2006.

[12] N. Nagappan, L. Williams, M. Vouk, J. Osborne, „Early estimation of software quality using in-process testing metrics: a controlled case study.", ICSE2005, pp 46-52, 2005.

[13] 1.Lj. Lazić, N. Mastorakis. " A Framework of Software Testing Metrics – Part 1and 2", in the WSEAS Book "ENGINEERING EDUCATION", Included in ISI/SCI Web of Science and Web of Knowledge, Agios Nikolaos, Crete Island, Greece,

July 24-26, 2007, ISBN: 978-960-8457-86-7, pp.137-143 and pp.144-153.