# Estimating the Required Test Volume and Effort for Software Verification and Validation

*A. Abran[1] and J. Garbajosa[2]*
*[1]Université du Québec, Canada; [2]Universidad Politécnica de Madrid, Spain*
*e-mail:* aabran@ele.etsmtl.ca, jgs@eui.upm.es

**Abstract**
*This paper introduces an approach to estimating the test volume and related effort required to perform verification and validation activities on software projects. This approach first uses size measures of functional requirements to estimate this volume, and then effort estimation models based on these test volumes. This estimation approach takes into account other types of non-functional requirements, as documented in ECSS-E-40, part 1-B.*

## 1. Introduction

The implementation of a verification and validation (V&V) process requires an estimation, early in the project life cycle, of the V&V effort for a specific project. This estimation can be produced based solely on expert experience. However, the use of former project data may help to obtain more accurate estimations and also to create a baseline for continuous improvement. This paper presents an initial study on the use of a functional size method to estimate V&V test volume and the effort required to perform the process. The advantages of this approach are that estimates are based on functional requirements, but taking into consideration other kind of requirements, and that it can be automated. An updated estimation can be produced if a new requirements baseline is defined.

This paper is structured as follows: section 2 briefly reviews the V&V process according to ECSSe40B. Section 3 introduces some software estimation concepts based on software functional size as the independent variable, and section 4, a V&V test volume estimation process. Next, section 5 presents a V&V effort estimation process based on functional and non-functional requirements. A discussion and some conclusions are presented in section 6.

## 2. A review of the V&V process

### 2.1 The process

In [3], the validation process (for software) is defined so as to ensure that the requirements' baseline functions and performances are correctly and completely implemented in the final product, as follows:

> The verification process (for software): to establish that adequate specifications and inputs exist for any activity, and that the outputs of the activities are correct and consistent with the specifications and input.

The software V&V process may start at any time following the System Requirements Review (SRR). This process is intended to confirm that the customer's requirements have been properly addressed, that all requirements have been met and that all design constraints are respected. Design constraints and V&V requirements are initially

defined at the system requirements analysis stage, and may end with the acceptance review.

The software V&V engineering processes consist of:

- verification process implementation;
- validation process implementation;
- verification activity;
- validation activity; and
- joint technical review process.

Verification activities are associated with all the intermediate products.

The software validation process is described in [3] as consisting of:

- validation process implementation;
- validation activities with respect to the Technical Specifications (TS); and
- validation activities with respect to the Requirements Baseline (RB).

The RB analysis and the TS analysis both constitute the input to the validation process, and to the estimation process as well. As the rest of the paper is focused on estimation based on functional requirements, it may be worth while to analyze the validation process in more detail here.

Taking [3] as the basic reference, supported by [4] and [5], the RB includes, in short, a set of system functions, performance requirements, overall safety and reliability requirements of the software to be produced, man machine interface, mock-up requirements and general requirements, system and software observability requirements, some configuration management requirements, and critical function identification and analysis. The RB must be ready at the System Requirements Review (SRR) stage, and, with it, the Interface Requirements Document (IRD). This IRD document provides useful information from an estimation point of view: interface requirements, software-hardware interface requirements, system-level interface requirements, system-level data interfaces, system-level integration support products and system-level integration preparation requirements. Validation with respect to the RB must be performed prior to the Qualification Review (QR).

The TS include the software requirements specifications, functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item executes, including budgets requirements, software product quality requirements, security specifications, human factors engineering (ergonomics) specifications, data definition, database requirements, the software logical model, the MMI specifications for software, and other non-functional and software quality requirements. The TS must receive initial approval at the Preliminary Design Review (PDR) stage, and then again at the Detailed Design Review (DDR) stage. The Interface Control Document (ICD) contains the specifications and design of the interfaces external to the software; this ICD must also be reviewed at the same time as the TS, and again, as in the case of the IRD, the information included within it can be useful as input to the estimation process. The validation with respect of the TS shall be performed prior to the Critical Design Review (CDR).

For each test, it is necessary to specify the testing specifications, which must include, whatever the testing approach, test items, inputs, outputs and test pass/fail criteria, as

well as test scripts, test procedures and environmental needs. Therefore, tests are interface-based. It must be noted that there may exist requirements that cannot be validated using tests, and where alternative techniques such as analysis will have to be used.

## 2.2 The types of requirements

In the ECSS-E-40 Part 1-B standard, a number of different types of requirements must be taken into account in the validation process – see Table 1.

**Table 1: Types of requirements in ECSS-E-40 Part 1-B**

| **Software Requirements Specification** |
|---|
| Requirements |
| 1 Functional requirements |
| 2 Performance requirements |
| 3 Interface requirements |
| 4 Operational requirements |
| 5 Resource requirements |
| 6 Design requirements and implementation constraints |
| 7 Security and privacy requirements |
| 8 Portability requirements |
| 9 Software quality requirements |
| 10 Software reliability requirements |
| 11 Software maintainability requirements |
| 12 Software safety requirements |
| 13 Software configuration and delivery requirements |
| 14 Data definition and database requirements |
| 15 Human factor-related requirements |
| 16 Adaptation and installation requirements |
| 17 Other requirements |

Interface requirements shall include software item external interfaces, interfaces between these software items and other software items, interfaces between the software items and hardware products, and interface requirements relating to the man-machine interaction (as applicable). Naming conventions applicable to the data comm and interface shall also be identified.

# 3. Software estimation based on functional requirements

### 3.1 Sizing functional requirements – Overview of ISO 19761: COSMIC-FFP

A key aspect of COSMIC-FFP [1] is the establishment of what is considered to be part of the software and what is considered to be part of the software's operating environment. Figure 1 below illustrates the generic flow of data from a functional perspective and from which the following observations can be made [7,14]:

- Software is bounded by hardware in the "front-end" and "back-end" directions. In the first, or front-end, direction, the software used by a human user is bounded by I/O hardware such as mouse, keyboard, printer or display, or by engineered devices such as sensors or relays. In the second, or back-end, direction, the software is bounded by storage hardware like a hard disk and RAM and ROM memory.
- Four distinct types of movement can characterize the functional flow of data attributes. In the front-end direction, two types of movement (ENTRIES and EXITS) allow the exchange of data with the users across a "boundary". In the

back-end direction, two types of movement (READS and WRITES) allow the exchange of data attributes with the storage hardware.

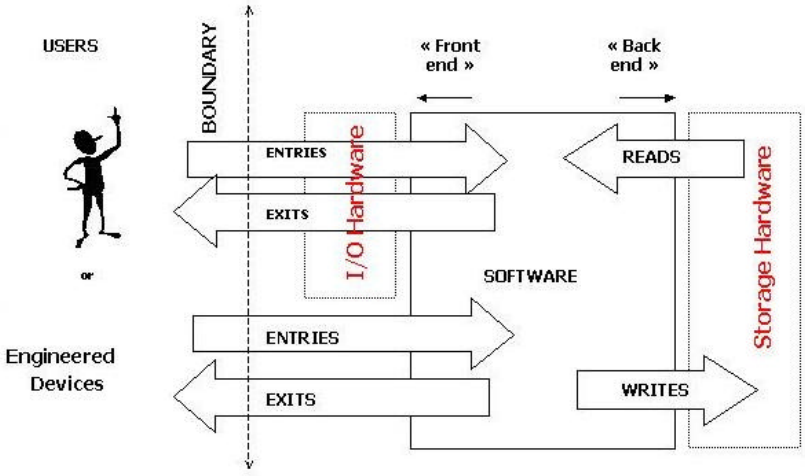


**Figure 1: Generic flow of Data Groups through software from a functional perspective [7]**

### 3.2. Estimation based on functional size

An estimation process is usually based on insights into the productivity of past projects. If this knowledge about past projects is quantitative and documented, then estimation models can be built based on statistics on past projects. If this knowledge about past projects is not documented, or quantified, then historical data cannot be accumulated, in which case the estimation process must rely entirely on people's subjective assessment of past projects; this is often referred to as experience-based estimation, or sometimes expert-based estimation. Such expertise cannot be either verified or validated.

In software engineering, software project productivity can vary considerably. In heterogeneous data sets with data coming from various organizations and from various countries, there may be significant variations, with a number of projects having much higher or much lower productivity than other projects of the same functional size.

The International Software Benchmarking Standards Group – ISBSG – is a not-for-profit organization established in the late 1990s [17] to improve software benchmarking and estimation by setting up and maintaining a publicly available repository of software projects. The ISBSG makes available to industry and to researchers, at a reasonable cost, an Excel data file containing 100 variables for each project in its repository, effort (in hours), functional size of the software (measured according to various standards of measurement available, e.g. Function Points, COSMIC-FFP – ISO 19761, MKII, etc.), programming language, etc. The ISBSG repository contains information about 3,024 projects, making it one of the largest software project databases publicly available at a reasonable cost. It is a multi-organizational, multi-application domain and multi-environment data repository; that is, its data content is fairly heterogeneous in terms of project characteristics.

In Figure 2, functional size is on the X-axis and effort on the y-axis. Figure 2 is typical of the data sets available in software engineering; that is, with an increasing dispersion of data, (referred to as heteroscedasticity in Kitchenham *et al.* [18]).

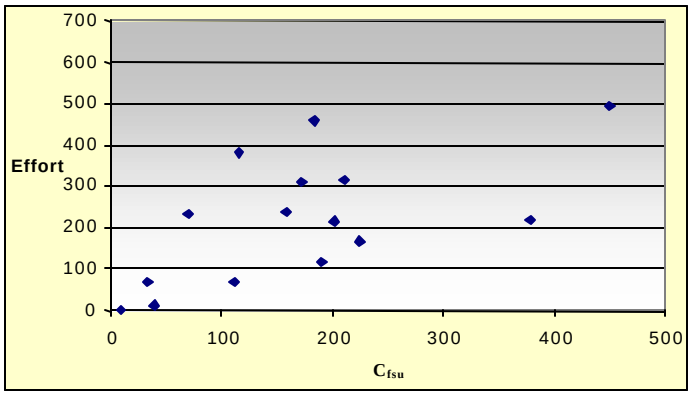


**Figure 2: A data set of 15 software projects (size units in Cfsu – ISO 19761)**

From such a data set, an estimation model can be obtained through a linear regression model which basically builds the line that best represents this set of projects in terms of effort with respect to corresponding project size (measured in ISO 19761 units: COSMIC functional size units – Cfsu in Figures 2 and 3).

In Figure 3, it can be easily observed that a number of projects have an effort cost lower than that predicted by the model, while there are also quite a few projects with an effort cost higher than that predicted by the model. This model is, of course, based on a single independent variable, functional size; it cannot be realistically expected that this variable would by itself be sufficient to produce a perfect estimate without taking into consideration the large number of other independent variables.

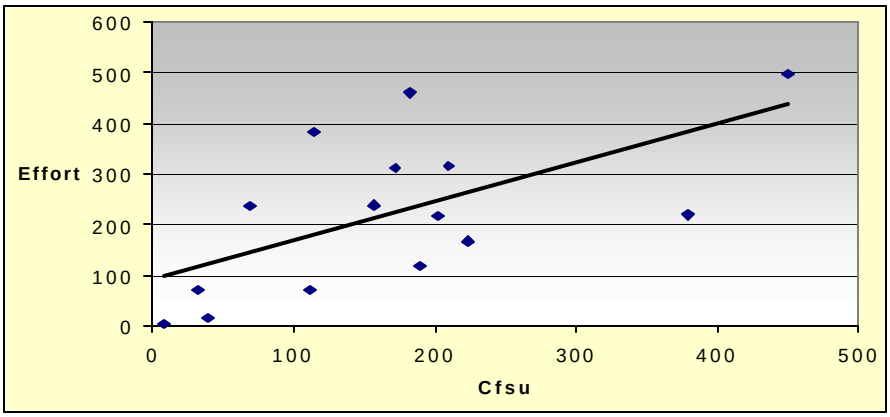


**Figure 3: Linear Regression Model of a15-project data set**

### 3.3. Upper and lower effort limits in data sets based on functional size

Of course, there are a number of other variables that can impact project effort, each having its own specific impact. Then, the combination of all these other variables will in turn have an overall impact on effort, the dependent variable. That is, the combination

of the impact of these other independent variables will lead to a particular effort for such a project: such effort may be lower or higher than the effort predicted by the regression line of a model with a single independent variable.

This is illustrated next with another data set [20]: in Figure 4 the circles point out the projects that have a large functional size with very little corresponding effort: these projects, within a functional size range of 500 to 2500 FP, did not cost more than many projects 10 to 20 times smaller (measured in ISO 20926 [15] units : Function Points Analysis – Unadjusted Function Points - UFP).

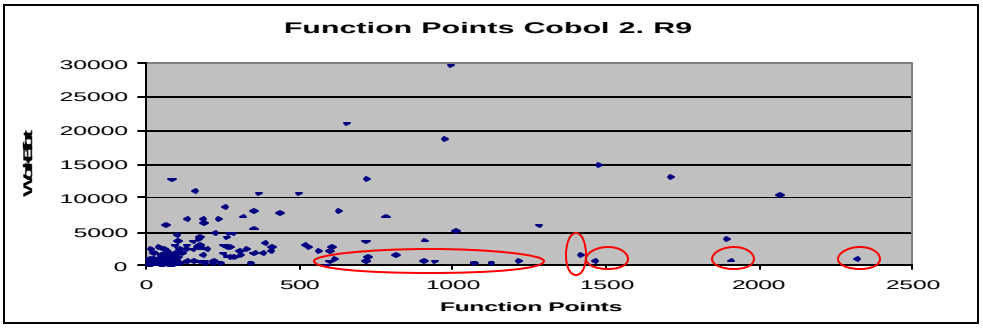


**Figure 4. Visual identification of projects with a much smaller unit cost [20]**

In Figure 5, the circles point out next, in the same data set, the projects with large effort with relatively small functional size. Again for illustrative purposes, three projects in the 400 to 1200 FP range cost at least 10 times more than other projects of similar size. This illustrates well that a number of other variables, in addition to size, must be taken into account to explain individual project effort.

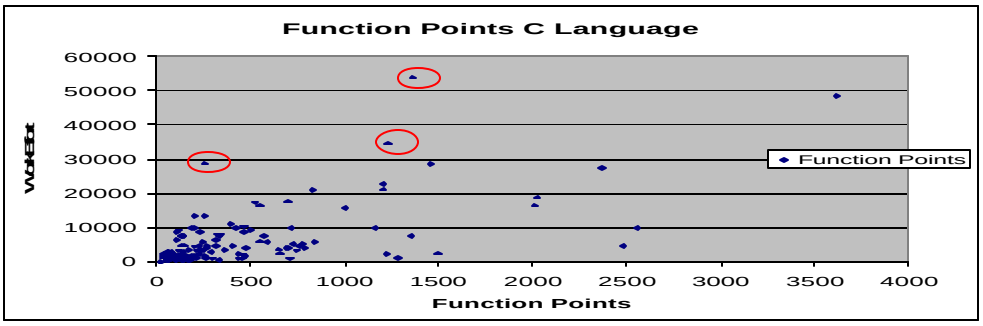


**Figure 5. Visual identification of projects with the highest unit cost [20]**

# 4. A software V&V Volume Estimation Process

This section proposes a way to use functional requirements to provide a V&V estimation of test volume, and then a way to assess the functional requirements for their usage as additional inputs to an effort estimation process.

## 4.1 Estimation of functional test volume

The first type of requirement in ECSS-E-40 (see Table 1) is the functional requirement. The high-level criteria for the measurement of functional requirements have been

adopted in ISO 14143 [12]. Four specific functional measurement methods have also been recognized by the ISO:

- ISO 19761: COSMIC-FFP [7,14].
- ISO 20926: Function Point Analysis (e.g. IFPUG 4.1, unadjusted function points – UFP only) [15];
- ISO 20968: Mk II [13]
- ISO 24570: NESMA [16][1]

The FPA, MarkII and NESMA methods were primarily designed to measure business application software, while COSMIC-FFP, the newest method, was designed to handle other types of software as well, such as real-time, telecommunications and infrastructure software (Figure 6).

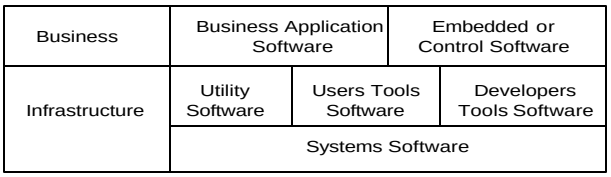


**Figure 6. Software types which can be measured with COSMIC-FFP [7]**

These functional size measurement methods must, as required by ISO 14143-1, measure the functional requirements of the software independently of the technical and quality requirements. This means that they must exclude all the other requirement types identified in ECSS-E-40: these will have to be dealt with in a subsequent estimation step.

The functional size measurement process requires that all software functions be identified, analyzed, measured and recorded for traceability purposes. This means that all software functions must be identified, measured and recorded. As a by-product, any such identified function must also be included in a functional test plan. For instance, the details of a functional sizing activity can be used as a detailed plan for functional testing.

The details of the functional requirements must exactly be mapped onto the set of functional tests required in V&V, the size of which can be referred to as the functional testing volume. These testing volumes can then be expressed using the same sized units: in Cfsu (COSMIC functional size units) for ISO 19761, and in Function Points (FP) for ISO 20926[2].

### 4.2 Assessment of other types of requirements to be validated

Each type of requirement from ECSS-e40 has to be taken into account in the V&V process and related effort, and, conversely, each of those requirements must have had an impact on the project productivity data of past completed projects in historical databases. The challenge is, of course, two-fold: how to 'size' these other types of requirements, and how to determine their respective impacts on V&V effort.

[1] NESMA [ISO 05] is a Dutch interpretation of FPA version 4.1, which produces similar results to FPA [NESM04]

[2] Convertibility of size units across both ISO standards is discussed in [21]

A COCOMO-like approach [9,10] is selected to take into account all 16 other ECSS-e40 types of requirement: for instance, for each type of requirement, a four-interval classification is defined, as illustrated in Table 2, using a similar set of four class intervals for all types of requirements. The first interval will represent basically the absence, or the minimum, of the corresponding variable to take into account in verification-validation, while the fourth interval would correspond to the maximum set of conditions for this independent variable.

Of course, standardized definitions of interval classification will be required to improve repeatability of the classification for each interval. This would ensure that the classification would be repeatable and reproducible across measurers and across projects.

An example of a classification is provided next, with similar class labels assigned to each ordered class, from "Low" assigned to the nil or minimum class, "Nominal" (as defined in a COCOMO-like model) to the middle class, "High" to the next one, and, finally, "Very High" to the highest-ordered class with the maximum value. It is to be noted that the middle, or nominal, class would correspond to the theoretical normal set of conditions, and is used as the reference base.

From this, a classification profile can be established for the analysis of the other requirements of a project: the project illustrated in Table 3 has 9 types of requirements with a classification of Very High (Table 3 – bottom row).

**Table 2. Example: A project assessment of 16 types of non-functional requirements**

| | **Types of requirements** | **Class 1** | **Class 2** | **Class 3** | **Class 4** |
|---|---|---|---|---|---|
| | | **Low** | **Nominal** | **High** | **Very High** |
| 1 | Performance requirements | Low | | | |
| 2 | Interface requirements | Low | | | |
| 3 | Operational requirements | Low | | | |
| 4 | Resource requirements | | | High | |
| 5 | Design req.& implementation constraints | | | | Max |
| 6 | Security and privacy requirements | | | High | |
| 7 | Portability requirements | | Nominal | | |
| 8 | Software quality requirements | Low | | | |
| 9 | Software reliability requirements | Low | | | |
| 10 | Software maintainability requirements | | Nominal | | |
| 11 | Software safety requirements | Low | | | |
| 12 | Software configuration and delivery req. | Low | | | |
| 13 | Data definition and database req. | | | | Max |
| 14 | Human factor-related requirements | Low | | | |
| 15 | Adaptation and installation req. | | Nominal | | |
| 16 | Other requirements | Low | | | |
| | Profile of the combined assessment of the 16 types of requirements for this simulated project | **9 Low** | **3 Nominal** | **2 High** | **2 Very High** |

## 5. A software V&V Effort Estimation Process

The estimation process consists of 5 steps:

1. Identification of a reference dataset;
2. Identification of the V&V functional test volume;
3. Building of the initial estimation model based on functional test volume;
4. Identification and classification of the set of non-functional requirements;
5. Adjustment of the initial estimation model (of step 3) to take into account the integrated set of non-functional requirements of step 4.

### 5.1 Identification of a reference dataset: ISBSG

In project management, the importance of planning, estimation and control is well known, and the problem of producing adequate estimation for software development projects has often been discussed in the literature [11]. The occurrence of this problem is not due to a lack of estimation process alternatives: expert-based, analogy-based, price-to-win, available resources-based and parametric models-based. Most estimation processes have defined steps, such as estimation of the size of the software (in lines of code, function points, etc.), which is then taken as input to estimation models based on the analysis of effort of past, completed projects.

For benchmarking, as well as for estimation models and tools requiring historical data, at least two prerequisites must be met: there must be enough historical data to ensure statistical validity, and the data must be homogeneous enough to provide meaningful interpretations.

The 2005 ISBSG repository, release 9 (R9), contains information on 100 data fields for 3,024 projects; however, these projects might not contain information about all 100 fields that can be collected in the repository, and only a subset of these are mandatory fields. The repository data originates from organizations across the world with projects from industries which have used various methodologies, phases and techniques. Of course, the ISBSG captures information about the process, technology, people, effort and product associated with the project, and has defined each of the fields to be collected in its repository with great care.

Two of the key fields in such repositories are, of course, size and effort. To adequately record the required background information on the total project effort reported in its repository, the ISBSG asks data collectors to map their own life cycle to a standardized ISBSG life cycle of six phases: Planning, Specification, Design, Build, Test and Implement.

In the ISBSG repository, total project effort is a mandatory field, while the effort per project phase is an optional one. This heterogeneity in project phases included in the effort data collected means, then, that the total project effort[3] recorded in the ISBSG repository has to be handled with care (e.g. the life cycle coverage is not identical across projects).

### 5.2 Identification of the V&V functional test volume

Before analyzing ISBSG data, it is important to understand how the repository's fields are defined, used and recorded, as recommended in [19]. In data set preparation, two verification steps must be carried out: data quality verification and data completeness verification. The verification of effort data quality analysis is easy: the ISBSG data

[3] The ISBSG refers to this total project work effort recorded as the 'summary work effort'

repository manager himself carries out such a data quality analysis right at collection time, and records his judgment in a project field. His rating will vary from very good (A) to unreliable (D). To reduce the risk of poor quality data and to improve the validity of the analysis reported here, projects with a data quality rating of C or D can be removed prior to the analysis.

In the ISBSG repository, not all projects have been sized according to the same functional sizing method. The IFPUG, COSMIC-FFP and Mark II methods are used, as well as a few others.

The 2005 version of the ISBSG data set, R9, contains 1,345 projects with information about the effort for the testing phase, which is of interest here.

Of these, 110 projects have an indicator of poor quality (C or D) data collection. This leaves 1,235 projects with good data quality collection (A or B).

Of this set, 84 projects use size measurement methods other than the IFPUG method, which leaves 1,151 projects measured with the same size method, that is, the IFPUG method.

Of this set of IFPUG-sized projects, 90 have a tag indicating doubts (tag = C or D) about the quality of the size measurement. This leaves 1,061 projects with reliable size measurement (size tag = A or B).

Furthermore, in terms of size measurement, there was a clear outlier at 16,148 FP, while the next biggest project in terms of IFPUG functional size measured 5,372 UFP. This outlier was deleted for further analysis. There remained, therefore, 1,060 projects.

These 1,060 projects could be further subdivided as a function of their project types:

- 556 were projects for the development of new software – Figure 7;
- 502 were projects for enhancements to existing software – Figure 8;
- 2 were redevelopment projects (these were dropped from further analysis).

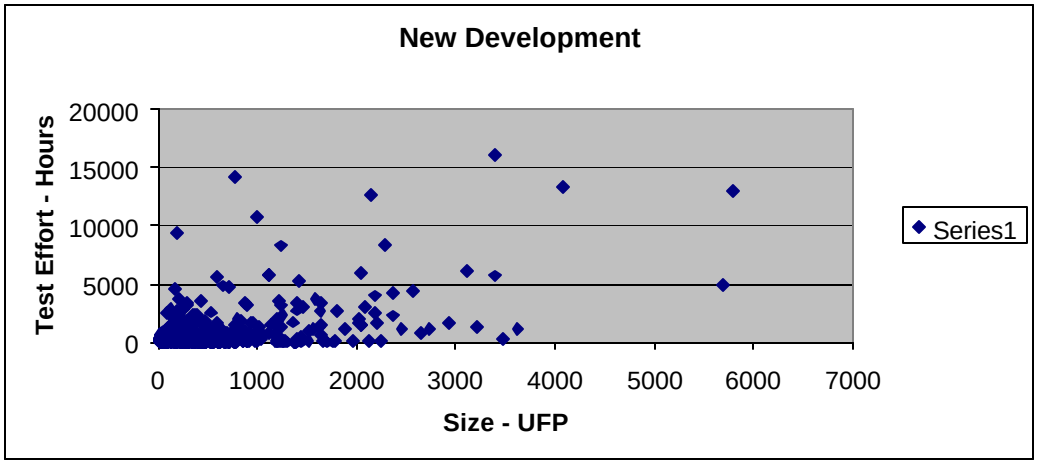


**Figure 7. ISBSG – Data set of 556 new software projects**

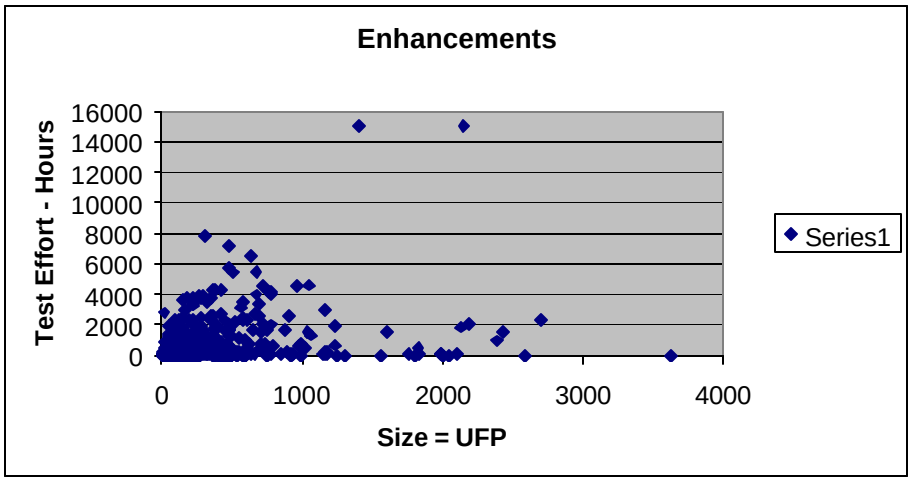


**Figure 8. ISBSG – Data set of 502 enhancement projects**

### 5.3 Building the initial effort estimation model based on functional test volume

The initial estimation model for the V&V process was initially based only on the functional test volume, and was built using the regression technique. For the 556 new development projects, the regression model based on functional test volume obtained is the following one:

V&V Effort = 1.3 *FP + 181 with an $R^2$ = .24

Of course, as can be seen in Figure 9, there is considerable dispersion from the single value predicted by the model.

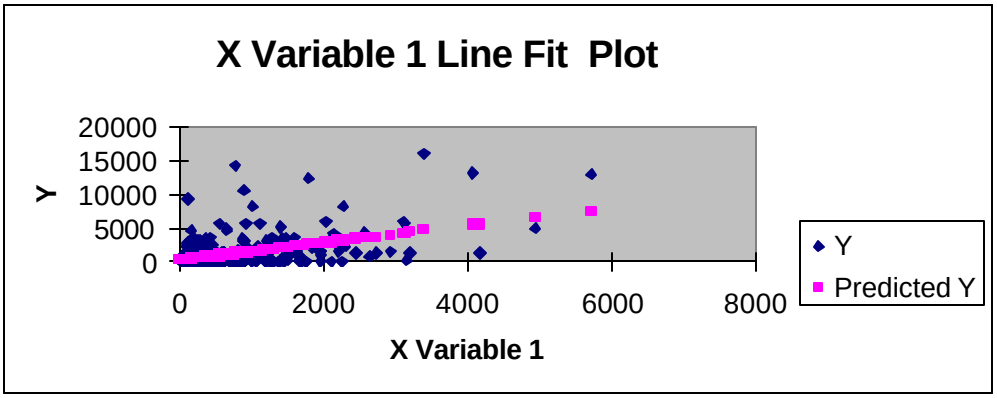


**Figure 9. Regression model for the 556 ISBSG's new software projects**

For instance, if we were to estimate the V&V effort for a project with a functional size of 1,000 FP, the regression model would predict:

Effort = 1.3 * 1,000 + 181 = 1,300 + 181 = 1,481 hours

With an R2 of only 0.24, a considerable dispersion across this value would be expected, as can be observed – see Figure 9.

Consider now that for this dataset the regression model represents the nominal values for all the other types of requirements. A graphical analysis can then be carried out to identify both the max and min values on the graph; from there, the High values could be considered between the Very High and regression model value.

That is, from the graph, the maximum is approximately 15,000 hours, while the minimum is approximately 10 hours. The low and high values would then be:
High = (Model value + Very High) / 2 = (1,481 + 15,000) / 2 = 15,481 / 2 = 7,741 hours

This, of course, represents considerable variation: a single type of requirement, that is, the functional requirements, explains only 24% of the effort variation, while all the other types of requirements contribute the other 76% of the variation in project effort. Therefore, the other non-functional requirements must also be taken into account as independent variables influencing project effort. This is discussed in the next section.

### 5.4 Identification and classification of the set of non-functional requirements

In the COCOMO I and II approaches, the 16+ independent variables (other than the size variable) have been described and classified in a four-interval range, each with its own definitions for classification within the four-interval set (these are, therefore, not really intervals, but rather ordered classifications without any specified numerical values).

The next step in the COCOMO approach is to assign a specific effort impact value to each of the ordered classes, for each of the 16 variables; this assignment was performed in COCOMO subjectively by a panel of experts in order to figure out, again intuitively, the perceived corresponding impact on effort. In this COCOMO approach, the commonly applied rule is that the impact on the size-based effort estimation model is nil[4] when an individual independent variable (referred to as a cost factor) is classified in the nominal ordered class; the impact is assigned a weight lower than 1 when in the lower-order classification, or higher than 1 in the higher-order classification. Then, in COCOMO, all the weights are combined, impacting the same size-based effort estimation equation, see [9,10].

In the approach presented in this paper, the first step of COCOMO is kept, but the second step of using combined subjectively assigned weights is bypassed entirely: there is, therefore, no attempt to subjectively figure out the impact of each individual variable, which in this case is the set of non-functional requirements.

### 5.5 Adjusting the initial estimation model to take into account the integrated set of non-functional requirements

The estimation approach selected does not attempt to model the individual effort relationship for each of the potential variables: it uses rather the information about the

[4] A null impact is represented by a weight = 1.0 in COCOMO I and II.

other non-functional requirements and the data from the ISBSG data set to position the project to be estimated, in terms of required effort, somewhere between the minimum and the maximum functional size as a function of the set of non-functional requirements.

The integration of all the other independent variables (that is, the non-functional requirements) is achieved through the graphical analysis of the data set on hand; that is, with the ISBSG data set for the specific V&V functional volume to be estimated: this is achieved by combining the information from the upper and lower effort limits in the available data sets based on functional size (see section 3.3) with the assessment of the non-functional requirements that impact effort (see section 4.2) in addition to functional size (see section 4.1). The integration of these concepts is illustrated next:

1- In the ISBSG dataset, all the projects precisely on the regression line of the initial estimation would correspond to projects with all non-functional requirements being in the nominal interval scale. Stated another way, the precise regression line is interpreted as corresponding to the expected nominal size-based effort relationship.

2- In the ISBSG dataset, all the projects with "very high" effort – that is, projects with the maximum effort above the regression line and along the functional size axis -, would correspond to projects with all non-functional requirements being the highest in the 4-interval scale.

Comment [MSOFFICE1]:

3- In the ISBSG dataset, all the projects with "low" effort – that is, projects with the minimum effort below the regression line, and along the functional size axis - would correspond to projects with all non-functional requirements being the lowest in the 4-interval scale

4- In the ISBSG dataset, all the projects with all 'high' non-functional requirements would somewhere in the mid-range between the regression model estimate and the 'very-high' effort estimate.

Of course, this approach does not presume that the range of data points above the regression line is not necessarily equal to the range below the regression line.

The set of 16[5] rated interval values assigned to each of the non-functional requirements can be used next to select a specific estimation value within the project effort data at the V&V functional volume that needs to be estimated.

### *5.6. Example*

The proposed technique is therefore quite simple, and an example is provided next using the ISBSG. For the data set illustrated in Figure 6, a project of 100 UFP with all its non-functional requirements rated as nominal would be estimated by the estimation model at:

$Y = 41 + 1.35 \times UFP$ $R^2 = 0.60$, for n=14 projects

$Y = 41 + 1.35 \times 100 UFP = 41 + 135 = 176$ days

[5] Corresponding to the 16 non-functional types of requirements in ECSSE-E-40 Part 1-B.33

At 100 UFP, the graphical analysis indicates an effort range between 75 days and 350 days.

It can be observed that the ranges have an asymmetry across the regression line:

- The upper range above the regression value at 100 UFP is (Very High: 350 days – Nominal:176 days) equal to 174 days;
- The lower range below the regression value at 100 UFP is (Nominal: 176 days – Low: 75 days) equal to 101 days.

Finally, the value for the High class is calculated as the mid-point between Nominal and Very High in the following way:
Nominal + (Very High – Nominal)/2 = 176 + (350 – 176)/2 = 176 +87 = 263 days.

The profile of non-functional requirements for the specific project illustrated in Table 2 (bottom row) is:

- 9 non-functional requirements rated as Low
- 3 non-functional requirements rated as Nominal
- 2 non-functional requirements rated as High
- 2 non-functional requirements rated as Very High

Taking this profile and the information from the ISBSG data set at 100 FP, the estimation process is illustrated next in Table 3, providing an estimation value of 166 days based on both the V&V functional volume and the corresponding set of 16 non-functional requirements listed in ECSS-E 40.

**Table 3: Estimation calculations for the project described in Table 2**

| Non-functional interval class | Number within a class<br>(1) | Effort on the data set for a class<br>(2) | Impact<br>(3) = (1) * (2) | Normalized value (= /16 classes)<br>(4) = (3) / 16 |
|---|---|---|---|---|
| Low | 9 | 75 days | 675 | 42 |
| Nominal | 3 | 176 days | 528 | 33 |
| High | 2 | 263 days | 526 | 33 |
| Very high | 2 | 350 days | 700 | 44 |
| Total | 16 | | 2429 | 152 days |

## 6. Discussion

This paper has presented an initial study on the use of the functional size approach to estimate V&V test volume and effort in the context of ECSS-e40 B. Next, a method for assessing and rating non-functional requirements in the context of ECSS-e40 was presented, followed by a proposal for its use in an effort estimation process which takes this set of non-functional requirements to make effort estimation adjustments within the lower and upper effort limits of a reference data set of completed projects. It is obvious that the applicability of this estimation approach to the ESA context would require data obtained from ESA projects.

The information required for the implementation of this estimation approach is available during the execution of projects developed in compliance with ECSS E-40. While this

estimation approach can be an entirely manual process, it would, of course, be extremely valuable to automate a large part of it, in terms of both data collection and data analysis. In particular, automated functional sizing with ISO 19761: COSMIC-FFP should be addressed first, since most of the required information is, like the measurement method for software functional size, available as part of the Requirements Baseline (RB) and the Interface Requirements Document (IRD) at the System Requirements Review (SRR) stage and the Software Requirements Specification (SRS) stage, and the Interface Control Document (ICD) at the Preliminary Design Review (PDR) and (Critical Design Review (CDR) stages. Standardization of the classification of non-functional requirements should be addressed next to ensure consistency of classification across software engineers and across organizations.

Finally, automation of graphical data analysis of available data sets should be investigated to take into account the adjustments derived from the classification of the non-functional requirements. In summary, a document based automated CASE support environment, such as the one described in [8], would be a good basis to facilitate the automation process. As long as most of the information is available, the cost would not be expected to be unreasonable and the advantages clear. It would also be expected that automation support could be provided without disturbing the software process.

## References


[1] ECSS E-10 Part 1B. System engineering – Part 1: Requirements and process. 18 November, 2004.

[2] ECSS E-10-02A. Verification. 17 November, 1998.

[3] ECSS E-40 Part 1B. Software – Part 1: Principles and requirements. November 28, 2003.

[4] ECSS E-40 Part 2B. Software – Part 2: Document requirements definitions (DRDs). March 31, 2005.

[5] ECSS Q-80B Software product assurance. October 10, 2003.

[6] Abran, A.; Maya, M.; Desharnais, J.-M.; St-Pierre, D., *Adapting Function Points to Real-Time Software*, in American Programmer, vol. 10 , 1997, pp. 32-43.

[7] Abran, A. ; Desharnais, JM.; Oligny, S.; St-Pierre, D.; Symons, C.; *Measurement Manual COSMIG-FFP 2.2*, The COSMIC Implementation Guide for ISO/IEC 19761, École de technologie supérieure, Université du Québec, Montréal, Canada, 2003.

[8] Alarcon , P.P.; Garbajosa, J.; Crespo, A.; Magro, B., *Automated integrated support for requirements-area and validation processes related to system development*, INDIN '04. 2004 2nd IEEE International Conference on Industrial Informatics, 2004.

[9] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, pp. 487, 1981.

[10] Boehm, B.W. et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.

[11] Ferens, D. V. (1999). The conundrum of software estimation models. *Aerospace and Electronic Systems Magazine, IEEE, 14*(3), 23-29.

[12] ISO/IEC 14143-1:1998 Information technology -- Software measurement – Functional size measurement -- Part 1: Definition of concepts, International Organization for Standardization, Geneva, 1998.

[13] ISO/IEC 20968: 2002, Software Engineering -- Mk II Function Point Analysis -- Counting Practices Manual, International Organization for Standardization -- ISO, Geneva, 2002.

[14] ISO/IEC19761:2003, Software Engineering -- COSMIC-FFP -- A Functional Size Measurement Method, International Standardization Organization - ISO, Geneva, 2003.

[15] ISO/IEC 20926: 2003, Software Engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting Practices Manual, International Organization for Standardization -- ISO, Geneva, 2003.

[16] ISO/IEC 24750: 2005, Software Engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Point Analysis, International Organization for Standardization – ISO, Geneva, 2005.

[17] ISBSG. (2005). *Worldwide Software Development - the Benchmark*. from www.isbsg.org

[18] Kitchenham, B.A.; Taylor, N.R., "Software Cost Models", *ICL technical journal*, Vol. 4, no 1, May 1984, pp. 73-102.

[19] Maxwell, K. D. (2002). *Applied statistics for software managers*. Upper Saddle River, N.J.: Prentice Hall PTR.

[20] Paré, D.; Abran, A., *Obvious Outliers in the ISBSG Repository of Software Projects: Exploratory Research*, Metrics News, Otto Von Gueriske Universitat, Magdeburg (Germany), Vol. 10, no. 1, August, 2005, pp. 28-36.

[21] Abran, A. ; Desharnais, JM.; Azziz; *Measurement Convertibility: From Function Points to COSMIC-FFP*. 15th International Workshop on Software Measurement. Montreal, Canada, 2005, pp 227-240.

**Acknowledgement.** This research work has been partially sponsored by the AGMOD project, Ref. TIC2003-08503, funded by the Ministry of Education of Spain.