Estimating the Test Volume and Effort for Testing and Verification & Validation

Alain Abran[1], Juan Garbajosa[2], Laila Cheikhi[1]

[1] Ecole de technologie supérieure, Universtité du Québec, Canada;
[2] Universidad Politécnica de Madrid, Spain
alain.abran@etsmtl.ca, jgs@eui.upm.es, laila.cheikhi.1@ens.etsmtl.ca

Abstract. This paper discusses an approach to estimating the test volume and related effort required to perform both testing and verification and validation activities on software projects. The approach uses measures of functional requirements as a basis for quantitatively estimating this volume, followed by effort estimation models based on functional size to prepare a first-degree order, and, finally, an adjustment based on the assessment of non-functional requirements. This estimation approach is illustrated using data from the ISBSG 2006 repository and the set non-functional requirements as categorized in the European standard for the aerospace industry: ECSS-E-40, Part 1B.

Keywords: Estimation, Verification & Validation, ISBSG, Test Volume, Testing effort, Non-functional Requirements

1 Introduction

In the Guide to the Software Engineering Body of Knowledge – SWEBOK [1] – testing is documented as one of the ten knowledge areas (KAs) of this discipline and one of its life cycle phases, while verification and validation (V&V) is described throughout the life cycle phases.

Both testing and V&V activities are time-consuming, and to obtain an adequate level of resources for both, their related effort must have already been included in the initial project estimates.

1.1 The Testing Process

According to the SWEBOK Guide, software testing "consists of the *dynamic* verification of the behaviour of a program on a *finite* set of test cases, suitably *selected* from the usually infinite executions domain, against the *expected* behaviour" [1]. The software testing KA is broken down into five sub-areas, which include guidelines and bests practices to use when testing a software product in order to achieve good quality.

Again according to the Guide, the V&V process "determines whether or not products of a given development or maintenance activity conform to the requirement of that activity, and whether or not the final software product fulfills its intended purpose and meets user requirements" [1].

Moreover, what differentiates Verification from Validation is that verification is "an attempt to ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities" [1], while validation is "an attempt to ensure that the right product is built, that is, the product fulfills its specific intended purpose" [1].

1.2 The Verification and Validation Processes

Aligned with [2] and following [3] "The software verification process establishes 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 validation process is defined so as to "confirm that the requirements baseline functions and performances are correctly and completely implemented in the final product". Document [2] provide software engineers with a rigorous, well defined and lifecycle process specification and, therefore, it can be advantageously used to get a better understanding of the inner relationship between the measurement process and the rest of the software lifecycle processes.

Standard ECSS-E-40 Part 1B, reference [2], is one of the series of European Consortium for Space Standardization (ECSS) Standards intended to be applied together for the management, engineering and product assurance in space projects and applications, and defines the principles and requirements applicable to space software engineering. It is aligned with [3] and [4].

According to [2], the software V&V process may start at any time following the System Requirements Review (SRR).This V&V process is intended to confirm that the customer's requirements have been properly addressed, that all requirements have been met and all design constraints 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
- a joint technical review process.

Verification activities are associated with all the intermediate products.
The software validation process is described in [2] 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 TS contains the software supplier's response to the requirements baseline. The RB expresses the customer's requirements. It is generated by the requirements engineering processes, and it is the primary input to the SRR review process. The

Requirements Baseline (RB) expresses the customer's requirements. It is generated by the requirements engineering processes, and it is the primary input to the SRR review process.

In reference [2] the term testing is always used to express activities of the development, verification or validation processes such as in "coding and [unit] testing", "operational testing", "acceptance testing" or "validation testing".

1.3 The Inputs to the Estimation Process

Taking [2] as the basic reference, supported by [5] and [6], the Requirements Baseline (RB) and the Technical Specifications (TS) together constitute the inputs to the estimation process, as well as to all life cycle phases, including testing and V&V. As the rest of the paper is focused on estimation based on functional and non-functional requirements, these inputs are discussed in more detail here.

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 Technical Specification includes the software requirements specifications, functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item executes, including budgetary requirements, software product quality requirements, security specifications, human factors engineering (ergonomics) specifications, data definition, database requirements, the software logical model, the Man Machine Interface (MMI) specifications for the software, and other non-functional and software quality requirements. The Interface Control Document (ICD), a document part of the TS, is very relevant from the estimation perspective; the ICD is the supplier's response to the system Interface Requirement Document IRD. The IRD expresses the customer's interface requirements for the software to be produced by the supplier. This document is part of the RB.

In [2] a number of different types of requirements are categorized, including 16 non-functional requirement types – see Table 1.

1.4 The Estimation Process

Testing effort is usually estimated on the basis of the total project effort estimated for the whole life cycle, and next by dividing it by a ratio of this total estimate to be allocated to the testing phase. When an organization has information by life cycle phases on past projects, such a ratio is based on the analysis of testing effort in previous projects; however, when an organization neither collects nor keeps historical

data on past projects, this ratio is determined by expert opinion and cannot be verified. An alternative is to derive such a testing/whole life cycle ratio from a publicly available repository, such as that of the International Software Benchmarking Standards Group – ISBSG[1].

Table 1. Types of requirements in ECSS-E-40 Part 1B [2]

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

While this testing effort is introduced into the work breakdown structure early on in the life cycle, it is not actually used until much later, when the coding is well advanced and the specific detailed testing activities must be scheduled and staffed. These testing estimates are then usually revised and adjusted based on the project performance over the previous life cycle phases. It should be noted that agile process models start to make use of testing effort much earlier than conventional process models; the approach presented within this paper can be equally applied to both conventional and agile models.

By contrast, the V&V activities that span all life cycle phases are to be staffed and initiated much earlier on. V&V requires, therefore, both earlier and better estimates; however, the current publicly available repositories do not include effort data for activities that span multiple project phases, such as V&V activities.

In [7], an initial model was proposed to estimate such V&V effort based on functional size and non-functional requirements. This paper presents a follow-up study on the use of functional size to estimate both the V&V test volume and the related effort required to perform the process. The advantages of this approach are that estimates are based on functional requirements and then adjusted, but taking into consideration non-functional requirements. This estimation process includes first the documentation of both the functional and non-functional requirements as the estimation inputs for the production of the initial estimate baseline. An updated estimation can be produced if new requirements, either functional or non-functional, are added to (or modified from) the baseline used for estimation purposes.

The estimation process proposed consists of 4 steps:

[1] www.isbsg.org

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

This paper is structured as follows: section 2 introduces some software estimation concepts based on software functional size as the independent variable, and section 3, an adjustment process based on non-functional requirements. Section 4 presents a case study for illustrative purposes, and section 5, a discussion.

2 Estimation Based on Functional Requirements

2.1 Functional Requirements and Sizing Methods

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 [8]. Four specific functional measurement methods have also been recognized by the ISO as meeting the requirements set out in ISO 14143-1:

- ISO 19761: COSMIC-FFP [9,10].
- ISO 20926: Function Point Analysis - FPA (e.g. IFPUG 4.1, Unadjusted Function Points – UFP – only) [11];
- ISO 20968: MkII [12]
- ISO 24570: NESMA [13]

The FPA, Mark II 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 1).

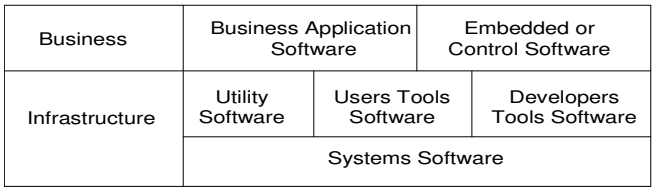

Figure 1. Software types which can be measured with COSMIC-FFP [9]

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 be exactly mapped onto the set of functional tests required, the size of which can be referred to as the functional test volume. These test volumes can then be expressed using the same size units: Cfsu (COSMIC functional size units - Cfsu) for ISO 19761, and in Function Points (FP) for ISO 20926. Convertibility of size units across both ISO standards is discussed in [14]

2.2 Estimation Based on Data from a Single Organization

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 either documented or quantified, then historical data cannot be accumulated, in which case the estimation process must rely entirely on a 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 datasets 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.

In Figure 2, functional size is on the X-axis and effort on the Y-axis. Figure 2 is typical of the datasets available in software engineering; that is, with an increasing dispersion of data, (referred to as heteroscedasticity in Kitchenham *et al.* [15]).

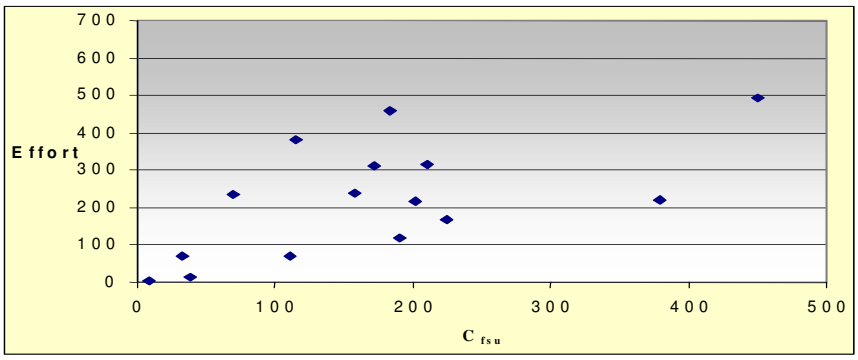

Figure 2. A dataset of 15 software projects (size units in Cfsu – ISO 19761)

From such a dataset, 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, and it cannot be realistically expected that this variable by itself would be sufficient to produce a perfect estimate without taking into consideration the large number of other independent variables.

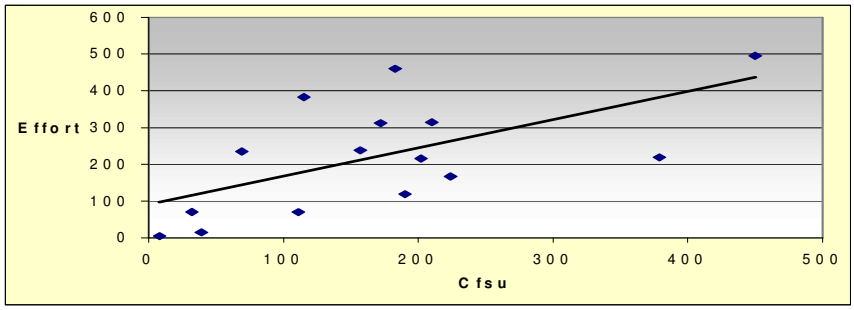

Figure 3. Linear Regression Model of a 15-project dataset

Of course, there are a number of other variables that can impact project effort, each having its own specific one. 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 an effort may be lower or higher than the effort predicted by the regression line of a model with a single independent variable.

2.3 Estimation Based on Functional Size Data from Many Organizations

The ISBSG is a not-for-profit organization established in the late 1990s [16] 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), the 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 February 2006 version of the ISBSG repository contains information about 3,855 projects, making it one of the largest affordable software project databases publicly available. It is a multi-organizational, multi-application and multi-environment data repository.

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 originate 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[2] recorded in the ISBSG repository has to be handled with care (e.g. the life cycle coverage is not identical across projects).

3 Estimation Adjustments Based on Non-Functional Requirements

This section proposes a way to use non-functional requirements to make adjustments to estimates initially based only on functional requirements and corresponding test volumes.

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

3.1 Assessment of Other Types of Requirements

Each type of requirement from ECSS-E-40 has an impact on the project productivity data of past completed projects in historical databases. The challenge is, of course, twofold: how to 'size' these other types of requirements, and how to determine their respective impacts on effort.

A COCOMO-like approach [17,18] was selected in [7] to take into account all 16 of the other ECSS-E-40 types of requirement: for instance, for each type of requirement, a four-interval classification has been defined, as illustrated in Table 2, using a similar set of four class intervals for all types of requirements. The first interval is represented basically by the absence, or the minimum, of the corresponding variable to take into account in V&V, while the fourth interval would correspond to the maximum set of conditions for this independent variable.

There is work in progress to standardize definitions of interval classification 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.

For illustrative purposes only, 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 2 has two types of requirements with a classification of Very High (bottom row).

3.2 Assessment of the Impact of Non-Functional Requirements

In the COCOMO I and II approaches, the some 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 not really intervals, therefore, 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 subjectively in COCOMO by a panel of experts in order to figure out, again intuitively, the perceived corresponding impact on effort. In this approach, the commonly applied rule is that the impact on the size-based effort estimation model is nil[3] 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 it is in the lower-order classification, and higher than 1 when it is in the higher-order

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

classification. In COCOMO, all the weights are combined, impacting the same size-based effort estimation equation (see [17,18]).

In the approach presented in this paper, the first step of COCOMO is kept, but the second step, in which combined subjectively assigned weights are used, is bypassed entirely. In other words, no attempt has been made to subjectively figure out the impact of each individual variable, which in this case is the set of non-functional requirements.

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 requirements & 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 requirements	Low			
13	Data definition and database requirements				Max
14	Human factor-related requirements	Low			
15	Adaptation and installation requirements		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**

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

the other non-functional requirements and the data from a historical dataset (here, the ISBSG repository) to graphically position the project to be estimated, in terms of required effort, somewhere between the minimum and the maximum effort for a specific functional size in a dataset 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 dataset at hand; that is, with the ISBSG dataset for the specific functional size volume to be estimated. This is achieved by combining the information from the upper and lower effort limits in the available datasets based on functional size with the assessment of the non-functional requirements that impact effort in addition to functional size. 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.
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 fall 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[4] 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.

This approach is illustrated in Figure 4.

[4] Corresponding to the 16 non-functional types of requirements in ECSS-E-40 Part 1B.

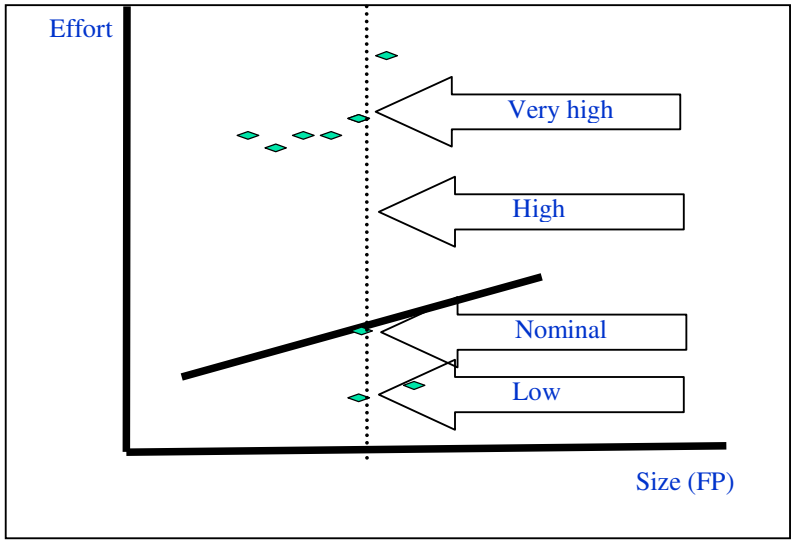

Figure 4. Estimation approach based on functional and non-functional requirements

4 A Case Study of a Testing Effort Estimation Process

The case study presented here takes for granted that the first of the following four steps has already been carried out to determine the initial functional size input to the estimation process:

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

4.1 Data Preparation

Before analyzing the ISBSG data, it is important to understand how the repository's fields are defined, used and recorded, as recommended in [19]. In dataset 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 repository manager himself does this 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 February 2006 version of the ISBSG repository contains 1,069 projects with information about the effort for the testing phase, which is of interest here.

Of these, 83 projects have an indicator of poor quality data collection (C or D). This leaves 986 projects with good data quality collection (A or B).

Of this set, 230 projects use size measurement methods other than the IFPUG method, which leaves 756 projects measured with the same size method, that is, the IFPUG method.

Of this set of IFPUG-sized projects, 95 have a tag indicating doubts (tag = C or D) about the quality of the size measurement. This leaves 661 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 5,732 UFP. This outlier was deleted for further analysis. There remained, therefore, 660 projects.

These 660 projects could be further subdivided as a function of their project types:

- 292 were projects for the development of new software;
- 366 were projects for enhancements to existing software;
- 2 were redevelopment projects (these were dropped from further analysis).

4.2 Initial Estimates Based on Functional Test Volume

The initial estimation model for the V&V process was initially based only on the functional test volume, and is built using the regression technique. For the 292 new development projects (see Figure 5), the regression model based on the functional test volume obtained is the following:

V&V Effort (New) = 1.57 hours/FP x No. of FP + 248 hours with an $R^2 = .31$

The regression model for the V&V Effort for the 366 Enhancement projects is:

V&V Effort (Enhancement)= 1.63 hours/FP x No. of FP + 236 hours $R^2 = .20$

Of course, as can be seen in Figures 5 and 6, there is considerable dispersion from the single value predicted by the model, and this is expected for such a large diversity of projects from many organizations, many functional domains and many countries.

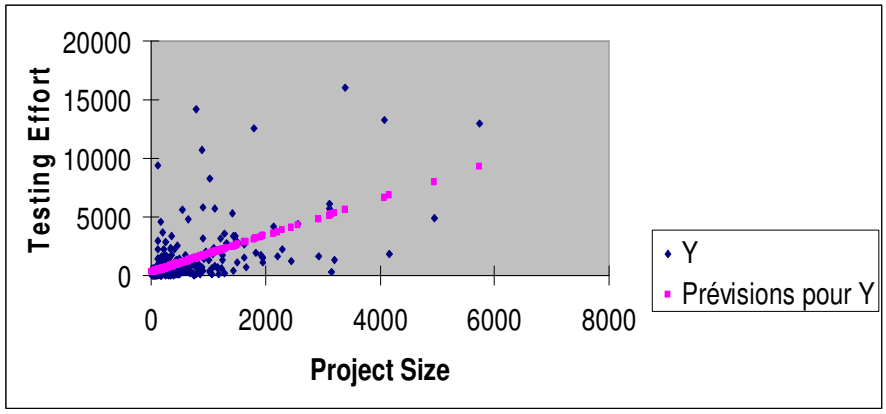

Figure 5. Regression model – ISBSG NEW software projects – N = 292 projects

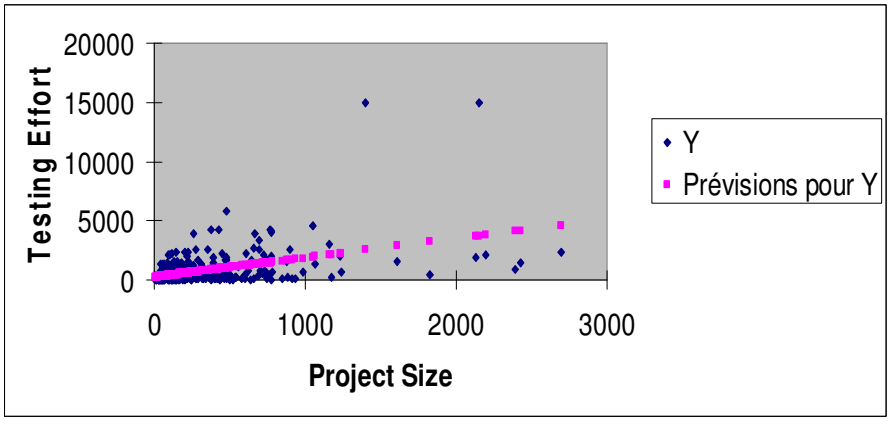

Figure 6. Regression model – ISBSG Enhancement projects – N = 366 projects

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

Effort = 1.57 hours/FP x 1,000 FP + 248 hours
Effort = 1,570 hours + 248 hours = 1,818 hours

With an R^2 of only 0,31, a considerable dispersion across this value would be expected, as can be observed (see Figure7).

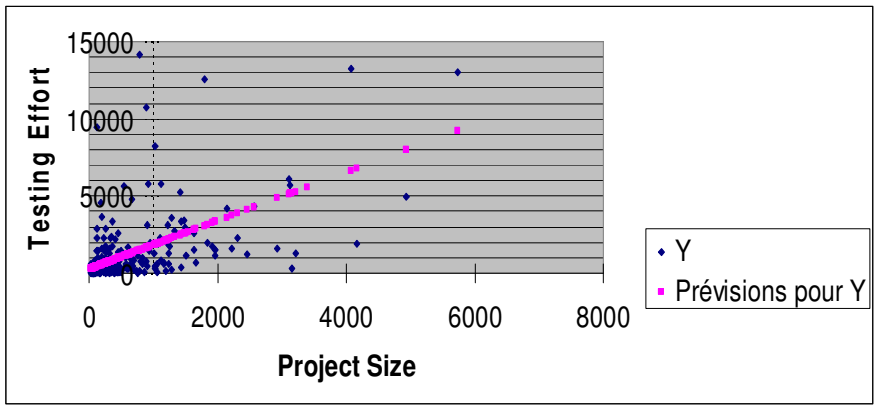

Figure 7. Regression model – ISBSG New Software projects – N = 292 projects

4.3 Estimation Adjustments Based on Non-Functional Requirements

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 values.

That is, from the graph, the maximum at 1,000 FP is approximately 15,000 hours (e.g. the Very High value), while the minimum is approximately 10 hours (e.g. the Very Low value).
The high value can then be calculated as the mid-value within the range of the model and very high values:

High Value = (Model value + Very High value) / 2
High Value = (1,818 hours + 15,000 hours) / 2
High Value = 16,818 / 2 = 8,409 hours

This, of course, represents considerable variation: a single type of requirement, that is, the functional requirements, explains only 31% of the effort variation for the New Software projects with respect to variations in project size, while all the other types of requirements contribute the other 69% 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.

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

- The upper range above the regression value at 1,000 UFP:
  (Very High: 15,000 hours) – (Nominal: 1,818 hours) = 13,182 hours.
- The lower range below the regression value at 1,000 UFP:
  (Nominal: 1,818 hours) – (Low: 10 hours) = 1,808 hours

This set of ranges of effort at the 1,000 FP size is illustrated in Figure 8.

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 dataset at 1,000 FP, the estimation process is illustrated in Table 3, which provides an estimation value of 3,273 hours for testing activities based on both the 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 (1)**	**Effort on the dataset for a class (2)**	**Impact (3) = (1) * (2)**	**Normalized value (= /16 classes) (4) = (3) / 16**
Low	9	10 hours	90	6
Nominal	3	1,818 hours	5,454	341
High	2	8,409 hours	16,818	1,051
Very high	2	15,000 hours	30,000	1,875
Total	**16**		**52,362**	**3,273 hours**

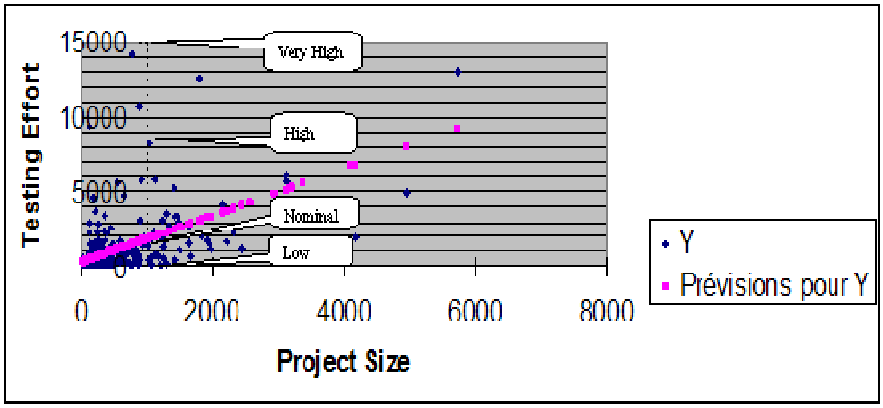

Figure 8. Ranges at 1,000 FP for ISBSG New Software projects - N = 292 projects

5 Discussion

This paper has presented an update to an initial study on the use of the functional size approach to estimating V&V test volume and effort in the context of ECSS-E-40 B. Next, a method for assessing and rating non-functional requirements in the context of ECSS-E-40 was presented, followed by a proposal for its use in an effort estimation process which takes this set of non-functional requirements into account to make effort estimation adjustments within the lower and upper effort limits of a reference dataset of completed 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, including 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. That is to say, input data for the estimation process is available at well identified project milestones. Therefore estimations can be refined across the project, what can be of great usefulness for the project success. Standardization of the classification of non-functional requirements should be addressed next to ensure consistency of classification among software engineers and across organizations.

Finally, automation of the graphical data analysis of available datasets 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 [20], and [21] would be a good basis on which 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.

Finally, a key limitation of this approach is that it is based entirely on the characteristics (functional and non-functional) of the software product to be developed and does not take into account the process costs drivers of the traditional estimation models. These traditional models are fairly weak, however, generally not into taking into account the non-functional requirements.

The approach described in this model could be referred to as a product-based approach to estimation and be used, for instance, for pricing purposes based on both the functional size of the software product to be developed and on its set of non-functional requirements (each classified on a five-point likert scale). Improvements to

the classification within such a likert scale should be investigated to ensure greater repeatability and consistency.

Of course, it would be preferable over the long term for the estimation model to integrate both the product-based approach and the cost-driver approach. However, this will require considerable additional research.

6 References

1. Abran, A., Moore, J., Bourque, P., Dupuis, R., Tripp, L.: Guide to the Software Engineering Body of Knowledge -- SWEBOK. Los Alamitos: IEEE-Computer Society. (2004),http://www.swebok.org/
2. ECSS-E-40 Part 1B. Software – Part 1: Principles and requirements. November 28 (2003)
3. ISO 9000 Quality management systems — Fundamentals and vocabulary. International Organization for Standardization -- ISO, Geneva (2000)
4. ISO/IEC 12207: Information technology — Software Lifecycle Proceses. International Organization for Standardization -- ISO, Geneva (1995)
5. ECSS-E-40 Part 2B. Software – Part 2: Document requirements definitions (DRDs). March 31 (2005).
6. ECSS-Q-80B : Software product assurance. October 10 (2003)
7. Abran, A., Garbajosa, J.: Estimating Software Validation Required Test Volume and Effort. SDSS 2005 First Annual ESA Workshop on Spacecraft Data Systems and Software, ESTEC 2005, Noordwijk, The Netherlands, Oct. 17-20 (2005)
8. ISO/IEC 14143-1:Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts. International Organization for Standardization, Geneva (1998)
9. Abran, A., Desharnais, J. M., Oligny, S., St-Pierre, D., Symons, C.: Measurement Manual COSMIC-FFP 2.2, The COSMIC Implementation Guide for ISO/IEC 19761. École de technologie supérieure, Université du Québec, Montréal, Canada (2003)
10. ISO/IEC19761: Software Engineering -- COSMIC-FFP -- A Functional Size Measurement Method. International Organization for Standardization -- ISO, Geneva (2003)
11. ISO/IEC 20926: Software Engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting Practices Manual. International Organization for Standardization -- ISO, Geneva (2003)
12. ISO/IEC 20968: Software Engineering -- Mk II Function Point Analysis -- Counting Practices Manual. International Organization for Standardization -- ISO, Geneva (2002)
13. ISO/IEC 24750: 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)
14. Abran, A., Desharnais, JM., Azziz, F.: Measurement Convertibility: From Function Points to COSMIC-FFP. 15th International Workshop on Software Measurement, pp. 227-240. Montreal, Canada (2005)
15. Kitchenham, B.A., Taylor, N.R.: Software Cost Models. ICL Technical Journal, Vol. 4, no. 1, 73-102 (1984)
16. ISBSG : Worldwide Software Development -- the Benchmark. (2005), www.isbsg.org
17. Boehm, B.W.: Software Engineering Economics. Prentice Hall (1981)
18. Boehm, B.W. et al.: Software Cost Estimation with COCOMO II. Prentice Hall (2000)
19. Maxwell, K. D: Applied statistics for software managers. Upper Saddle River, N.J.: Prentice Hall PTR (2002).

20. Alarcon , P.P., Garbajosa, J., Crespo, A., Magro, B.: Automated integrated support for requirements-area and validation processes related to system development. INDIN '04, 2nd IEEE International Conference on Industrial Informatics (2004)
21. Bollaín, M. Garbajosa, J.: Defining and Using a Metamodel for Document-Centric Development Methodologies. ENASE 2007, International Working Conference on Evaluation of Novel Approaches to Software Engineering, Barcelona, Spain. July 2007

Acknowledgment

This research has been partially sponsored by the OVAL/PM project, Ref. TIN2006-14840, funded by the Ministry of Education and Science of Spain, and the FLEXI ITEA2 project 06022, funded by the Spanish Ministry of Industry as FIT-340005-2007-37.

This research project has also been funded partially by the European Community's Sixth Framework Programme – Marie Curie International Incoming Fellowship under contract MIF1-CT-2006-039212.