

Software Projects Estimation & Control: the Versatility and Contributions of COSMIC Function Points

Alain Abran
ETS – University of
Québec, Canada
alain.abran@etsmtl.ca

Charles Symons
COSMIC
United Kingdom
cr.symons@btinternet.com

Christof Ebert
Vector Consulting
Services
Germany
christof.ebert@vector.com

Frank Vogezang
Ordina
The Netherlands
frank.vogezang@ordina.nl

Hassan Soubra
ESTACA
France
hassan.soubra@estaca.fr

Abstract – *This talk presents an outline of the design of the ‘2nd generation’ COSMIC Function Point method and industry evidence of its versatility and value in software project estimation and control, via both traditional and agile methods. COSMIC Function Points can also be used for early and rapid sizing at estimation time and can be automated with very high accuracy.*

Keywords – *COSMIC, functional size measurement, early software sizing, software project estimating, ISO 19761, real-time embedded software, project control, requirements measurement*

I INTRODUCTION

This talk has four aims:

1. To outline the design of the COSMIC method for measuring a functional size of software. This includes a discussion of the weaknesses and limitations of the 1st generation of functional size measurement (FSM) methods as well as the advances made by the Common Software Measurement International Consortium (COSMIC) to address them.
2. To present evidence that COSMIC Function Points are of practical value in software project performance measurement and estimating.
3. To present rapid COSMIC sizing techniques at estimation time.
4. To present industry evidence of its automation in a variety of contexts, including with very high accuracy for real-time embedded software.

The paper is structured as follows. Section II describes the background to software measurement and to the first generation of Function Points measurement methods. Section III presents an overview of the principles of the COSMIC Function Points, the second generation of Function Points. Section IV gives several examples of how COSMIC Function Points have been used in various organizations and in different domains that demonstrate how it delivers value in practice, including in Agile contexts. Section V presents COSMIC Guidelines for early or rapid sizing at estimation time. Section VI presents the work done for the automation of COSMIC size measurement,

including the work at Renault from software specifications documented in Matlab-Simulink. Section VII presents a summary.

II BACKGROUND TO SOFTWARE SIZE AND FUNCTION POINTS MEASUREMENT

A Measurement Objectives

There are two principal objectives that motivate an interest in software size measurement:

1. Enable managers to estimate and control software projects (for both new software and enhancements to existing software).
2. Compare performance across these projects and activities.

Broadly speaking there are four main categories of performance indicators that should be measured for development projects:

- project delivery to time and budget
- project productivity (product size/project effort)
- project speed (product size/project duration)
- product size (to control ‘scope creep’) and product quality, both from a functional and a technical point of view

All these indicators are of interest, since they are tradeable. For example: it is generally accepted that, within limits, a project may be speeded up by adding resource at the expense of lower productivity, though views vary on the degree of the trade-off of effort and duration [1]. Similarly, a project that was underestimated may meet its budget but deliver less product functionality or lower quality than was expected.

In order to compare performance across software projects, the product size (taken as a measure of ‘work-output’) must be measured in a standard way that is independent of the development methods and technologies used to implement the software product.

Past measurements of productivity and speed can be used to establish performance benchmarks and to develop estimation models for projects sharing common characteristics. To be able to use past performance data to estimate the effort and duration for future projects, it is important to be able to estimate first the size of the software to be developed from an early statement of requirements, as size is usually the

largest driver of software-related project effort. Combining the estimate of the size of the product to be developed by a new project with benchmark performance data from similar past projects enables estimation of the effort and duration for a new project.

The ability to estimate and/or measure the size of an item of software at various stages in its life-cycle is therefore critical to managing most software development activities.

B. FUNCTION POINTS Background

Allan Albrecht of IBM had the original idea of measuring a size of software from its Functional User Requirements - FUR [2]. His proposal evolved in the 1980's into the IFPUG [3] and NESMA [4] Function Points. Methods of measuring a size of the FUR of software have the advantage that the resulting size should be totally independent of the technology used for the software and any other aspects of the implementation.

Even from the early days, however, this initial Function Points model of software was criticised in various ways. Hence within a few years of Albrecht's proposal, a variety of variants of Function Points methods had been proposed. Of the numerous variants, only four of the 1st generation methods have been recognized by ISO.

This '1st generation' of FSM methods (e.g., methods defined before 2000) share a number of weaknesses.

- Most were designed to measure software only from the domain of business applications. Nowadays even much business software is built from components, and exploits infrastructure software and real-time functions.
- Most are step-functions, with very limited ranges (various min, max), and without a well-defined measurement unit of 1 Function Point - see Figure 1.

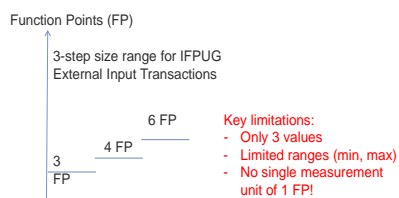


Figure 1. Step function in the 1st generation of Function Points

Concurrently over the years, in response to the weaknesses of 1st generation FSM methods, several

other types of sizing variants have been introduced. These are usually associated with a specific method for modelling requirements or a specific software development process, such as: Use Case Points [5], Object Points [6], and Story Points [7]. Most of these variants share a number of common weaknesses, including improper mathematical operations on distinct scale types and the lack of a well-defined measurement unit [8]. Furthermore, they lack both detailed rules to eliminate subjectivity in measurement, and to ensure general applicability.

The Story Points method in particular is, by design, interpreted differently across project teams, so no benchmarking is feasible. Story Points therefore does not support accountability in project management and does not provide a sound foundation for estimation purposes [9].

III. OVERVIEW OF THE COSMIC FUNCTION POINTS

A. Background

During the late 1990's, an ISO Working Group established some principles for functional size measurement [10]. Subsequently, a sub-set of the WG members from around the world decided that the market needed a new Function Points method based on these principles and designed to overcome the weaknesses of the 1st generation methods. The COSMIC organization was formed in 1998.

The COSMIC group took as its initial input the designs of the Full-Function Points extension for real-time and embedded software published [11] and of the MkII FPA method [12]. The first official version of the COSMIC FSM method was made publicly available in 2001, after extensive field trials. This 2nd generation COSMIC method was approved by ISO following its full ISO development and review process¹ [13].

B. COSMIC Method Design Goals

The COSMIC FSM method was designed with the following goals.

- To be based on fundamental software engineering principles.
- To be applicable for sizing the software functional requirements of business application, real-time and infrastructure software and hybrids of these, in any layer of a software architecture. Sizes should be independent of technical requirements, and of project methods and effort used.
- To measure on a ratio scale so that all operations on measured sizes are mathematically valid. (Sizes measured by

¹ The four 1st generation FSM methods were approved through the ISO fast-track process without a detailed review to address their weaknesses.

most 1st generation FSM and by other sizing methods do not meet this criterion [8].)

- Measurement should be possible at any time in the software life-cycle, and at any level of decomposition of the software.
- To be usable for performance measurement and for estimating software activities.
- To be an 'open' method, available for free usage.

C. Overview of the COSMIC Method

ISO 14143-1 [37] specifies that a functional size measurement (FSM) method must measure software functional user requirements. The COSMIC Function Points [36] standard proposes a generic model of software that can capture the functionality of any type of software in a measurable way. Figure 2 illustrates the generic flow of data that can be observed in software from a functional perspective. From this generic model of software depicted in Figure 2 it can be observed that:

- Software is bounded by hardware. In the left-hand side of Figure 2, software used by a human is bounded by I/O hardware (e.g., a mouse, a keyboard, a printer, or a display), or by engineered devices (e.g., sensors or relays). In the right-hand side of Figure 2, software is bounded by persistent storage hardware (like a hard disk, or RAM or ROM memory).
- Software functionality is embedded within the functional flows of data groups. These data flows can be characterized by four distinct types of data movements:
 - two types of movements (ENTRIES and EXITS) allow the exchange of data with users across a boundary.
 - two types of movements (READS and WRITES) allow the exchange of data with the persistent storage hardware.
- Different abstractions are typically used for different measurement purposes.
 - In real-time software, the users are typically the engineered devices that interact directly with the software, that is, the users are considered the I/O hardware.
 - In business application software, the abstraction commonly assumes that the user is one or more humans who interact directly with the business application software across the boundary; the I/O hardware is ignored.

COSMIC Function Points measure the size of software based on identifiable functional user requirements. Once identified, those requirements are

allocated to hardware and software from the unifying perspective of a system integrating these two "components". Since COSMIC is aimed at sizing software, only those requirements allocated to the software are currently considered in its measurement procedure.

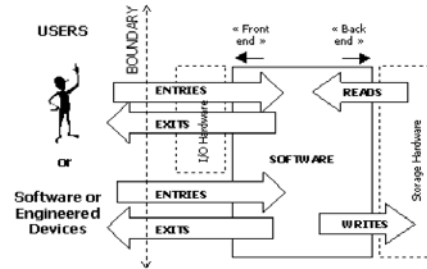


Figure 1. Generic flow of data groups through software in COSMIC Function Points

Figure 3 presents next a model of the functional user requirements in terms of functional processes.

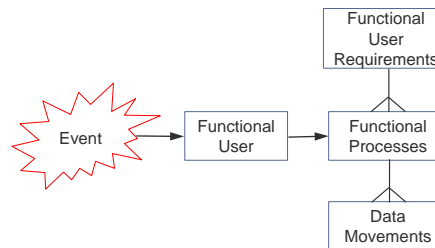


Figure 3. Functional User Requirements and Functional Processes [15]

The COSMIC model for software size measurement rests on the following principles²:

- The functional user requirements of any software can be decomposed into separate functional processes.
- When a functional user detects or causes an event ('something that happens in the real world') the user sends a data group to the software, triggering a functional process.
- A functional process is complete when it has executed all that must be done according to the functional user requirements in response to the triggering data group it receives.
- Functional processes comprise sub-processes that move data and that manipulate data. As

² A simplified description.

there is no agreement on how to measure data manipulation, the method assumes that each data movement sub-process accounts for any associated data manipulation

- There are four types of data movements:
 - Entries and Exits each move a data group in and out of the software, from/to functional users.
 - Reads and Writes each move a data group from/to persistent storage.
- A data group is a set of attributes that describes a single 'object of interest', i.e. a 'thing' in the real world of the functional users about which the software must enter, store, or output data.
- The unit of measurement of the COSMIC FSM method is one (1) data movement of one (1) data group, referred to as one (1) 'COSMIC Function Point' (CFP) – see fig. 4.
- The size of a functional process is the addition of its data movements. The size of an item of software is the sum of the sizes of its functional processes.
- There is no upper limit to the size of a functional process. (In practice, single processes of size over 100 CFP have been measured.) – see Figure 4, which should be contrasted with Fig. 1.
- The size of a change to a functional process is the sum of its data movements that have been added, changed and deleted. 'Changed' can mean either the data group moved has changed in some way, or the associated data manipulation has changed. A change may vary in size from 1 CFP, with no upper limit – see Fig. 4.

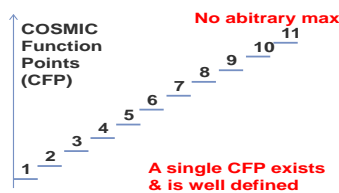


Figure 4. COSMIC – 2nd generation of Function Points

For a more comprehensive overview of the method, see [14] and for the full details of the method, including its principles, rules and definitions, see [15]. To be noted that all of the COSMIC documents, including the case studies, are available free on the web.

The current 2017 method is version 4.0.1.

Another COSMIC Guideline [21] describes approaches to assuring the accuracy of functional size

measurements, dependent on the available detail of the functional user requirements.

D. Versatility of COSMIC in Various Software Domains

While the COSMIC Measurement Manual [15] gives all the principles, rules and definitions of the method for software from any domain, with many examples, a number of domain-specific Guidelines have been written in order to provide more support for Measurers, with many more examples of how to measure specific types of FUR. Primary amongst these are the Guidelines for sizing:

- Business Application Software [23], and
- Real-Time Software [24].

In addition, other COSMIC Guidelines have been published on measuring:

- Data Warehouse software [25], and
- software built according to Service-Oriented Architecture (SOA) conventions [26].

Several papers and articles have also been published on sizing mobile applications [e.g. 27] and a related Guideline is under development.

These Guidelines covering Business and Mobile applications, Real-Time and SOA software should be sufficient to meet the need to measure software components of 'Internet of Things' systems.

Finally an increasing number of case studies of measurements of quite large systems have been published and more are in the pipeline at the time of writing.

IV. INDUSTRY EVIDENCE OF COSMIC CONTRIBUTIONS

All the theory and expert judgement in the world would be of no interest if the COSMIC Function Points measured functional sizes that do not correlate reasonably with project effort for software from its intended domain of applicability.

This section presents the results of studies from various organizations on business application and real-time software [9] [28] [29] [30] [31] [32]. Together, they form a convincing body of evidence that COSMIC Function Points meet both objectives: not only that, COSMIC meets them better than earlier FSM methods and is applicable to a wider range of software.

A. Applying COSMIC to Agile Projects

Both of the principal measurement objectives apply regardless of how the project will be managed, e.g. via either a waterfall or agile approach.

However, agile projects have their own particular estimating needs, namely to help plan the next activity

Commented [AA1]: Maybe a list of case studies can be added here

at every stage, e.g. at the level of user stories, sprints, iterations, releases, as well as for the whole system.

The COSMIC method fit well with the size estimation needs of agile projects, without any adaptation:

- Individual user stories can be measured in CFP according to the normal rules.
- The resulting sizes are objective, unlike the use of Story Points, being measured in a standard way, sizes may be compared across projects.
- Sizes can be added up at all the higher levels using the normal aggregation rules.

A COSMIC Guideline explains how to apply the COSMIC method to agile projects [22].

As noted previously, the use of Story Points to measure User Stories and to plan Sprints, etc., in agile projects is only practicable within one development environment, where all team members should have a common understanding of the unit of measure. Story Points also cannot help with estimating the total software size early in a project where usually a cost/benefit analysis is needed before committing to the project. Many organizations are now known to use COSMIC sizing for agile projects, for reasons of its objectivity and because it can be used at all levels of aggregation.

A Canadian software house supplies security and surveillance software systems to world-wide clients. In their case, a request for new or changed functionality is called a 'task': this organization uses the Scrum method, with iterations lasting 6 weeks. For an iteration, effort is estimated using a 'Planning Poker' process in units of Story Points, and this is then translated directly into estimated work-hours.

The supplier wished to understand if COSMIC sizing would help improve their effort estimates for tasks [9]. A sample of 24 tasks from nine iterations was measured in CFP for which SP-estimated and actual work-hours were available. Figure 5 shows the supplier's estimated versus actual effort figures for the 24 tasks. The SLR-fitted straight line would be poor for estimating accuracy ($R^2 = 0.33$).

When measured CFP sizes were first plotted against actual effort for the 24 tasks, the R^2 was much improved at 0.78. However, two tasks showed up as with very low actual effort figures: that is, these 2 tasks had very high productivity, as measured objectively with CFP. It was realised that these two tasks benefited from very significant re-use of some existing software. They were next excluded from the dataset for the purpose of building an estimation model in the usual context of limited functional reuse in this organization. Figure 6 shows the CFP sizes versus actual effort figures for the remaining 22 tasks.

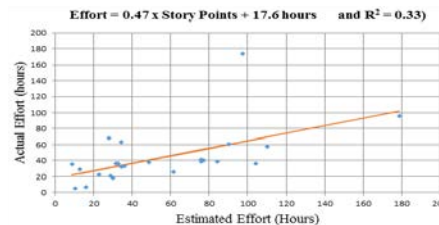


Figure 5. Actual versus Estimated (Story Points) for 24 Tasks [9]

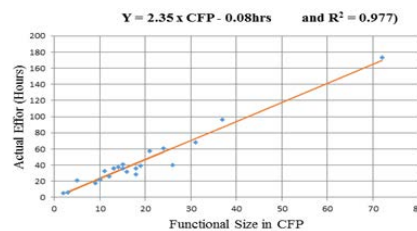


Figure 6. Actual Effort versus CFP Size for 22 Tasks [9]

The study concluded that 'although the Planning Poker/Story Points are widely used in the agile community, the COSMIC measurement method provides objective evidence of the team performance as well as better estimates'.

B. Cost Estimating from Automotive Electronic Control Unit (ECU) Designs

Renault, the European vehicle manufacturer, has published its progress in successful software development estimating, most recently in 2014 [28] [29] [30]. Similar results have been reported at another European automotive manufacturer – eg., Volvo and GM [31].

A modern average family car has roughly 50 Electronic Control Units (ECU's), small processors that form a distributed network to monitor and/or control almost every function, e.g. engine, lights, air-conditioning, tyre pressures, navigation, driver information, etc. The ECU's, their embedded software and their associated sensors are mostly bought from component suppliers, subject to specifications issued by Renault.

Renault has been collecting data on the costs and performance of its suppliers of ECU software for a few years. The process by which it contracts to procure ECU's is summarized next:

- Renault software departments, specialized by vehicle functional area (e.g. powertrain), develop specifications for new or enhanced

Commented [AA2]: To be verified and add references + some material from these papers

Commented [AA3]: This might need to be updated....

ECU software and store these in the Matlab Simulink tool.

- A Renault-developed tool automatically computes the COSMIC functional size of each specification (or the increase in size if an enhancement) – see also Section V on COSMIC automation.
- Past measurements and statistically-established relationships are used to predict the effort that the supplier will need to develop the software (see Fig. 7) and its memory size (Fig. 8).

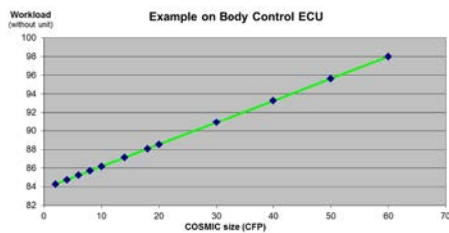


Figure 7. Effort vs COSMIC size for an ECU software [28]

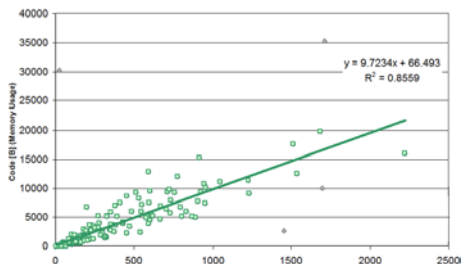


Figure 8. Memory usage vs COSMIC size [28]

- This information is used by the Purchasing Department to negotiate the price for each ECU. Further, the information available to Renault is now sufficiently well-established that it can be used to negotiate annual price changes in the same way that car manufacturers periodically negotiate prices of other materials such as steel, paints etc., and other components (Fig. 9).
- COSMIC functional sizes are also used to monitor the performance of the internal staff who develop the specifications, since Renault has established a specification-size/staff-level relationship for their work.

Renault states that at the end of a new ECU software development, the difference between the initially estimated effort from the established

correlation and the actual value 'has to be lower than 5%' (see Fig. 10) [28].

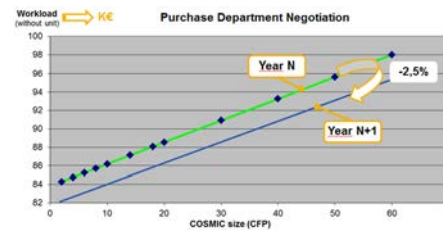


Figure 9. Purchase Department negotiation [28]

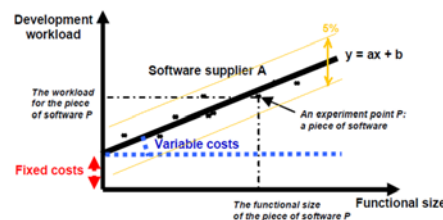


Figure 10. Control of precision of cost estimates [28]

C. Estimating Effort for ECU Maintenance Changes

A leading German global automotive supplier has applied COSMIC sizing for change requests to Electronic Control Unit (ECU) software. In this case, the sizing and effort estimation is required at a much earlier stage when only diagrams in a modelling tool and text specifications are available. (In the Renault case, the estimation take place only after a design is completely modelled in Simulink).

Prior to the introduction of COSMIC sizing, estimation was only by analogy or by informal methods. The organization recognized the following benefits of using COSMIC.

- Use of a repeatable process for analysing the change requirements and mapping to a design proposal in the modelling language. This significantly increased trust leading to more efficient collaboration between customer and supplier.
- However, each technical environment and modelling approach needed some different mapping rules to the COSMIC Generic Software Model. Variable input data quality also means that Measurers must be experienced to ensure accurate measurements.
- The measurements now provide a solid base for benchmarking.
- Effort estimation accuracy improved from up to 50% (and variable) uncertainty before

starting the measurement programme, to within 10 – 20% within one year.

- The measurement programme also aims for continuous feedback and improvement.

D. Effort Estimation for a Web Software Supplier

An Italian supplier of industrial web applications had used IFPUG Function Points to measure the size of the FUR, as input to its effort estimation method.

Wishing to understand whether COSMIC FP sizes would be more accurate for predicting development effort than IFPUG FP sizes, 25 applications were re-measured in units of CFP [32].

The 25 applications formed a rather heterogeneous dataset, including e-government, e-banking, Web portals, and Intranet applications. All the projects were developed with SUN J2EE or Microsoft .NET technologies. Oracle was the most commonly adopted DBMS, but also SQL Server, Access and MySQL were employed in some of these projects.

Application sizes ranged from roughly 100 to 900 FP (or from under 200 to over 1100 CFP). Effort ranged from roughly 1000 to 5000 work-hours.

In order to build effort estimation models, the size/effort relationship was analyzed in two ways for the sizes measured on both FSM methods. They were Simple Linear Regression (SLR) and Case-Based Reasoning (CBR), i.e. a Machine Learning-based solution. Both analyses led to similar results. The 'median of the absolute residuals' was 180 work-hours for the effort predicted from CFP sizes and 515 work-hours for the effort predicted from FP sizes.

The study concluded that for this dataset 'COSMIC was significantly more accurate than FPs in estimating the development effort' [32].

V. EARLY SIZE MEASUREMENT AT ESTIMATION TIME

For project estimation purposes, estimates of functional size are almost invariably needed quite early in a project, before the requirements have been worked out in full detail. Hence the need for versions of the detailed FSM method that can measure an approximate functional size from outline requirements.

The same need for approximate size measurement can arise if there is a need to measure a large number of software items for the purpose of controlling performance of maintenance and enhancement activities. In this case, measuring an approximate size may be sufficient, much faster and more cost/effective than making precise measurements.

Given these needs, several researchers and practitioners have carried out studies to develop

approximate variants of the COSMIC FSM method. These have been summarized with extensive examples from both the business application and real-time domains in a COSMIC Guideline for Early or Rapid COSMIC Functional Size Measurement [20-26].

In the early phases of a project, the documentation completeness cannot be expected upfront, and will evolve progressively. For sizing purposes, COSMIC recognizes various quality level of the documentation of the software functional processes – see Table 1.

Table 1. Quality levels of the documentation of Functional Processes

Functional Process Quality Level	Quality of the functional process definition
Completely defined	Functional process and its data movements are completely defined
Documented	Functional process is documented but not in sufficient detail to identify the data movements
Identified	Functional process is listed but no details are given of its data movements ³
Counted	A count of the functional processes is given, but there are no more details
Implied (A 'known unknown')	The functional process is implied in the actual requirements but is not explicitly mentioned
Not mentioned (An 'unknown unknown')	Existence of the functional processes is completely unknown at present

This related COSMIC Guidelines documents the applicability, the reported use as well as their strengths and weaknesses of each of the following approximation techniques to complement the lack of quality in the documentation of the functional user requirements:

- Average functional process approximation.
- Fixed size classification approximation – see Table 2.
- Equal size bands approximation – see Tables 3 and 4.
- Average use case approximation.
- Early and quick COSMIC approximation – see Table 5.
- Easy function points approximation – see Table 6
- Approximation from informally written textual requirements.
- Approximation using fuzzy logic – the EPCU model.

Table 2. Fixed size classification from [8]

Classification	Size (CFP)	#E	#X	#R	#W	Error messages
Small	5	1	1	1	1	1
Medium	10	2	2	3	2	1
Large	15	3	3	4	4	1
...						

Table 3. Equal size bands from 37 business applications

Band	Average size of a Functional Process	% of total Functional Size	% of total number of Functional Processes
Small	4.8	25%	40%
Medium	7.7	25%	26%
Large	10.7	25%	19%
Very Large	16.4	25%	15%

Table 4. Equal size bands from a major component of an avionics system

Band	Average size of a Functional Process	% of total Functional Size	% of total number of Functional Processes
Small	5.5	25%	49%
Medium	10.8	25%	26%
Large	18.1	25%	16%
Very Large	38.8	25%	7%

Table 5. Candidate values for the functional categories of the Early & Quick approach

Type	Level	Ranges COSMIC Equivalent	min CFP	most likely	max CFP
Functional Process	Small	1-5 Data movements	2.0	3.9	5.0
	Medium	5-8 Data movements	5.0	6.9	8.0
	Large	8-14 Data movements	8.0	10.5	14.0
	Very large	14+ Data movements	14.0	23.7	30.0
Typical process	Small	CRUD (Small/Medium processes) CRUD + List (Small processes)	15.6	20.4	27.6

Medium	CRUD (Medium/Large processes) CRUD + List (Medium processes) CRUD + List + Report (Small processes)	27.6	32.3	42.0	
Large	CRUD (Large processes) CRUD + List (Medium/Large processes) CRUD + List + Report (Medium processes)	42.0	48.5	63.0	
General process	Small	6-10 Generic FP's	20.0	60.0	110.0
	Medium	10-15 Generic FP's	40.0	95.0	160.0
	Large	15-20 Generic FP's	60.0	130.0	220.0
Macro process	Small	2-4 Generic GP's	120.0	285.0	520.0
	Medium	4-6 Generic GP's	240.0	475.0	780.0
	Large	6-10 Generic GP's	360.0	760.0	1,300.0

Table 6. Probability distributions of approximate values in the business domain

Classification of the FP	Specific action level	CFP min	CFP	CFP max	Approximate CFP	Probability
Small FP	Little unknown	2 (10%)	3 (75%)	5 (15%)	3.2	>80%
Small FP	Unknown (No FUR)	2 (15%)	4 (50%)	8 (35%)	5.1	<50%
Medium FP	Little unknown	5 (10%)	7 (75%)	10 (15%)	7.25	>80%
Medium FP	Unknown (No FUR)	5 (15%)	8 (50%)	12 (35%)	8.95	<50%
Large FP	Little unknown	8 (10%)	10 (75%)	12 (15%)	10.1	>80%
Large FP	Unknown (No FUR)	8 (15%)	10 (50%)	15 (35%)	11.45	<50%
Complex FP	Little unknown	10 (10%)	15 (75%)	20 (15%)	15.25	>80%
Complex FP	Unknown (No FUR)	10 (15%)	18 (50%)	30 (35%)	21	<50%

VI. COSMIC METHOD AUTOMATION

A. Automation in industry and in R&D

Researchers in industry and academia have already demonstrated several tools and some have reached commercial use, such as:

- The specifications of the automated process to measure a CFP size of Simulink blocks has been published by Renault for public use [33] [34]. Renault's motivation for automation was 'speed and accuracy of measurement'.
- Some tools are available to assist CFP measurement data capture and/or to enter data directly into an estimating tool [35].
- Various researchers have pointed out the strong links between the concepts of COSMIC and the Unified Modelling Language and how to automate CFP measurement of Use Cases. A Turkish telecommunications company has described its mapping of UML requirements ontology and COSMIC measurement ontology [36, which also includes a survey of similar approaches]. The paper describes the validation of the mapping by measuring requirements automatically for new and changed functionality of a Call Centre application.
- Another example is the Polish software house which has developed a tool to measure CFP sizes from requirements stored according to UML conventions (with certain restrictions) in the Enterprise Architect CASE tool. The tool is available under a Creative Common Licence [37]. It is used by seven software suppliers to three Polish public-sector bodies under consortia contracts and the contracts allow that COSMIC sizes are not measured exactly according to the method's rules.
- Semi-automated measurement of CFP sizes from executing programs has also been reported [38], [39]. The process in [38] involved inserting code into a 3-tier Java business application to capture the data movements. The process was subsequently applied for three other applications. The resulting CFP measurements were found to be 92% accurate; the cost of manual measurement was almost three times higher than that of automated measurement.
- A model-based and automated approach to size estimation of embedded software components is also discussed in [40].

B. Accuracy Verification Protocol of Automation Tools

A literature review in [42] has shown that very little work has been conducted on verifying measurement results produced by FSM automation, and hence to independently demonstrate the accuracy of a FSM automation tool, a verification protocol has been proposed [42]. The COSMIC automation at Renault

has been fully verified using this verification protocol that recommends the use of a sequence of input specifications, starting from specifications with only one FP and having the minimum combination of mandatory data movements, and then progressing in an orderly and systematic sequence to achieve the most extensive completeness, in terms of combinations of flows.

In this protocol, the samples input to the automation tool should cover all the types of input combinations that might be encountered. To verify the accuracy of individual automated measurements, each sample must also be measured manually, in parallel, using the FSM procedures being automated, and all the details of the measurement steps must be kept for traceability purposes. So, in addition to the total functional size obtained in CFP, each of the FPs and data group movements is identified and compared to each of the FPs identified in parallel in the manual measurement procedure.

Moreover, because an error could result from the manual measurement, or be caused by the prototype tool, the verification protocol must identify which party is responsible for such an error.

Verification of the measurement accuracy of a tool is executed in three phases, as shown in Fig. 11:

1. Phase 1: Numerical results comparison. The measurement results (in CFP) of the whole of the software measured, both those produced by the automation tool and those obtained manually, are compared. If the results match, then there is no difference between the automated and manual measurement processes. However, this does not mean that the processes used for identifying the individual COSMIC elements and to obtain the final results are similar. It is important to understand that, if the verification stops with this phase, only the final results will have been verified.

2. Phase 2: Detailed comparison. If the final numerical results at the end of Phase 1 do not match, and to find the reason for the difference, the results at the detailed level are compared, that is, the FPs obtained automatically and manually are verified. It is also necessary to verify whether or not there were any human errors in those results, using the detailed measurement results obtained by the automation tool:

- If there is no difference in the number of FPs, each FP obtained automatically is verified against its manually obtained 'peer' to determine whether or not there is a difference in their names (or their identifiers).
- If every FP obtained automatically matches its peer obtained manually, then their functional sizes are compared. A difference indicates that one or more data movements in the FP must be responsible. Then, at the end of this phase, any data movement

responsible for an error is isolated, in both the manual and automated measurement results.

3. Phase 3: Automation tool and input verification. This phase is triggered when the possibility of human error (in the manual measurement) is discarded at the end of Phase 2. A detected error can come from two sources: a measurement error or an error in the requirements input to the measurement process. Therefore, this phase consists of the following:

- Determining which module of the automation tool is responsible for the error. These modules can have sub modules; if they do, the sub modules causing the error are inspected as well.
- Determining, in parallel, the input requirements, in order to detect a possible defect that may be causing the error.

Once an error has been detected, the following steps are taken:

- If the error was caused by the automation tool, a correction is made to the tool and the appropriate specification is re-measured with the new version of the tool, and then re-verified.
- If the cause of the error was in the specification itself, the defect is recorded for possible future enhancements to the specification or to a specific functionality, or both, in order to bypass this defect in the tool.

Finally, the revised version of the input requirements – if there is one – is verified by the protocol, to ensure that the results of the manual and automated measurements are the same.

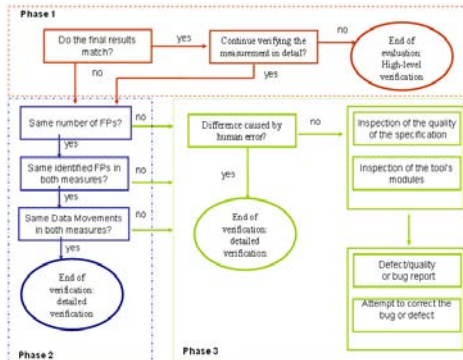


Figure 10. The 3-phase verification protocol for FSM tool automation accuracy [42]

C. COSMIC Automation Accuracy at Renault S.A.

The verification protocol was applied at Renault with a set of 77 distinct specification models (designed in Simulink): various sizes of specifications were chosen

among a number of software functions that represented different ECMs (Engine Control Modules) in the department where the automation prototype-tool was initially developed [42].

Overall, the difference in the total size of the 76 correct requirement models obtained both manually (i.e. 1,729 CFP) and using the automation prototype (i.e. 1,739 CFP) is less than 1% – see Table 7. The accuracy of the automation prototype after testing is greater than 99%: this means that it is 99% accurate not only at the total size, but also that 99% of the individual sizes of the data movements individually are accurate.

In addition to detecting one incomplete requirement specification, the proposed verification protocol helped identify the limitations of the prototype tool, which stem from the limitations inherent in the libraries that it uses. These limitations were next corrected in the automation tool developed by a Renault subsidiary based on the verified prototype: this automation tool is now more robust and has a greater level of accuracy, and is currently in use in a number of departments at Renault S.A. [28] [34].

Table 7. Difference between the total size obtained manually and using the prototype tool [42]

Total Number of Models	Total Size obtained manually (CFP)	Total Size obtained using the prototype tool (CFP)	Difference (%)	Accuracy
76 fault-free models	1,729	1,739	Less than 1%	>99%
All 77 models	1,758	1,791	1.8%	>98%

D. COSMIC Automation at ESTACA

Two COSMIC-based automation prototype tools were developed at ESTACA: they can be downloaded for free from the ESTACA website [44]. While the first prototype tool was developed to measure the size of aerospace real-time embedded software modeled using the SCADE commercial tool [45], the second one was developed to correctly measure the functional size of ECU application software designed following the AUTOSAR (AUTomotive Open System Architecture) standard [46]. AUTOSAR is the new generation of ECU software design architecture, methodology, and metamodel [47] [48]. It has become an important part of the production design criteria for many vehicle manufacturers, especially in the automotive

electronics industry [49]. The procedure automated by the prototype is based on the measurement guideline presented in [50] and has a set of mapping rules to be applied to the system modeled in order to obtain its functional size.

The automation algorithm developed at ESTACA is presented in Fig. 16 and follows the proposed FSM procedure step by step. It was implemented first in Python [42] and then in Java [44].

The protocol on the prototype tool used the Steer-by-Wire system described in [50]. The system provides two main functionalities: the feedback torque and the rack torque. The two functionalities were implemented using a set of 10 *Runnables* distributed in 6 software components (SWCs) - see Fig. 17.

The manual and automated measurement results showed that there was no difference in the final measurement results of the two measurement procedures (manual and automated): the functional processes identified were exactly the same. In addition, precisely the same data movements were identified in both measurements. Lastly, the total functional size measured in both the manual and automated application of the FSM procedure was the same.

VI. SUMMARY AND FUTURE CHALLENGES

As a summary, from this body of evidence:

- The COSMIC method design is simple and has been accepted as applicable to all the different types of software for which it was designed.
- The 'open' COSMIC community has made all the significant advances in functional size measurement in this century.
- Evidence from applying the COSMIC method in multiple organizations around the world demonstrates that it can be used as a key component to help achieve the two principle objectives of software measurement: namely to enable control of software projects and other activities, and to be used for estimating future activities.
- As new types of software emerge, new Guidelines will be needed on how to apply COSMIC sizing. For instance, it is possible that a new Guideline will be needed for measuring software assembled from pre-existing components (i.e. services). This is currently being studied.
- However, the advantage of the flexibility of a principles-based method (over the rules-based approach of '1st generation' methods) is that no new rules are needed to measure new types

of software. Fundamentally, the COSMIC method is stable and hence 'future-proof'.

The method is now being used world-wide. The Measurement Manual has been translated into 10 other languages besides English. It has been or is being adopted as a national standard for use by Governments in countries such as China, Mexico and Poland.

As the COSMIC organization has no membership, the real extent of use is not known precisely. Over 600 people have now passed the COSMIC Foundation-level certification examination. The largest numbers are from countries such as Brazil, China, India, Italy, Poland and Turkey.

In 2009, the General Accountability Office of the US Government recommended the IFPUG and COSMIC methods for use in software cost estimating [51]. In 2016, the National Institute of Standards and Technology of the USA in its investigation on a 'Rational Foundation for Software Metrology' [52] cited only the COSMIC method as having a well-defined unit of measurement.

The single most important technical challenge is to develop automated tools to measure or to support measurement of COSMIC sizes. Automation is of course dependent on the technology supporting the inputs to the automation. For automated measurement of the size of the functional requirements, partially or entirely documented using distinct requirements formats and a variety of requirements tools, automation is required for each requirements technology environment. Similarly, measurement automation of the requirements as implemented (e.g. lines of code): distinct automation tools are required for each of the large variety of programming languages.

Automation will help to progressively overcome a major barrier to acceptance of Function Points, namely the time and experience needed for accurate manual measurement and data recording.

Automatic size measurement from requirements is particularly difficult when the requirements are not expressed in sufficient detail, or are full of ambiguities, etc. So this is a very large challenge, though made relatively easier by the simplicity of the COSMIC Generic Software Model and its foundation on software engineering principles.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the enormous contribution over the years of the many members of the Measurement Practices Committee and other measurement experts who have given their time to continually improve the COSMIC method and to produce its various publications.

REFERENCES

References labelled with an asterisk are available for free download from:

<http://cosmic-sizing.org/cosmic-publications/overview/>

- [1] Charles Symons, 'Exploring Software Project Effort versus Duration Trade-offs,' IEEE Software, July-August 2012, pp 67-74.
- [2] Allan Albrecht, "Measuring application development productivity," IBM Applications Development Symposium, Monterey, California, pp. 83-92, 1979.
- [3] ISO, "ISO/IEC 20926:2009: Software and systems engineering -- Software measurement - IFPUG functional size measurement method," International Organization for Standardization, Geneva, 2009.
- [4] ISO, "ISO/IEC 24570:2005 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, Geneva, 2005.
- [5] Gustav Kerner, "Resource Estimation for Objectory Projects," Objective Systems SF AB, 1993.
- [6] Kauffman, Wright, Zweig, "Automating Output Size and Reuse Metrics in a Repository-Based Computer Aided Software Engineering (CASE) Environment," IEEE Trans. Software Engineering, 20(3), p. 169-186, 1994.
- [7] M. Cohn, "User Stories Applied for Agile Software Development," Addison-Wesley, 2004.
- [8] Alain Abran, 'Software Metrics and Software Metrology,' IEEE-CS Press & John Wiley & Sons – Hoboken, New Jersey, May 2010, pp. 328.
- [9] Christophe Commeysne, Alain Abran, Rachida Djouab, "Effort Estimation with Story Points and COSMIC Function Points - An Industry Case Study," Software Measurement News, Vol. 21, No. 1, 2016. *
- [10] ISO, "ISO/IEC 14143/1:2007 Software and systems engineering - software measurement - functional size measurement - definition of concepts," International Organization for Standardization, Geneva, 2007.
- [11] Alain Abran, Marcela Maya, JM Desharnais, Denis St-Pierre, "Adapting Function Points to Real-Time Software," American Programmer, Vol. 10, Issue 11, November 1997, pp. 32-43.
- [12] Charles Symons, 'Function Point Analysis: Difficulties and Improvements', IEEE Transactions on Software Engineering, Vol. 14, no. 1, January 1988.
- [13] ISO, "ISO/IEC 19761:2011 Software engineering -- COSMIC: a functional size measurement method," International Organization for Standardization, Geneva, 2011.
- [14] COSMIC, "Introduction to the COSMIC method of measuring software," version 1.1, January 2016. *
- [15] COSMIC, "Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2011)," version 4.0.1, April 2015. *
- [16] Alain Abran, Khalid Al-Sarayreh, J.J. Cuadrado-Gallego, "A Standards-based Reference Framework for System Portability Requirements," Computer Standards and Interfaces, Vol. 35, 2013, pp. 380-395.
- [17] Christof Ebert, 'Putting Requirements Management into Praxis: Dealing with Non-functional Requirements,' Information and Software Technology, Vol. 40, No. 3, pp. 175-185, 1998.
- [18] COSMIC and IFPUG, "Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating," v1.0, September 2015. *
- [19] COSMIC, "Guideline on Non-Functional & Project Requirements: How to consider non-functional and project requirements in software project performance measurement, benchmarking and estimating," version 1 November 2015. *
- [20] COSMIC, "Guideline for Early or Rapid COSMIC Functional Size Measurement by using approximation approaches," July 2015. *
- [21] COSMIC, "Guideline for assuring the accuracy of measurements," version 0.93, February 2011. *
- [22] COSMIC, "Guideline for the use of COSMIC FSM to manage Agile projects," version 1.0, September 2011. *
- [23] COSMIC, "Guideline for Sizing Business Application Software," version 1.2b, September 2015. *
- [24] COSMIC, "Guideline for Sizing Real-Time Software," version 1.1 April 2015. *
- [25] COSMIC, "Guideline for Sizing Data Warehouse Application Software," version 1.1, April 2015. *
- [26] COSMIC, "Guideline for Sizing Service-Oriented Architecture Software," version 1.1, March 2015. *
- [27] F. Vogelezang, J.K. Ramasubramani, S. Arvamudhan, "Estimation for Mobile and Cloud Environments," book chapter in Modern Software Engineering Methodologies for Mobile and Cloud Environments, (eds.) A.M. Rosado da Cruz and S. Paiva, DOI 10.4018/978-1-4666-9916-8.
- [28] Alexandre Oriou, E. Bronca, B. Bouzid, O. Guetta, and K. Guillard, "Manage the automotive embedded software development cost & productivity with the automation of a Functional Size Measurement Method (COSMIC)," Joint 24th International Workshop on Software Measurement (IWSM) and 9th MENSURA conference, Rotterdam (The Netherlands), IEEE CS Press, 2014.
- [29] Sophie Stern and Olivier Guetta, "Manage the automotive embedded software development cost by using a Functional Size Measurement Method (COSMIC)," in ERTS² 2010, 5th International Congress & Exhibition, Toulouse, France, 2010.
- [30] Cigdem Gencel and Sophie Stern, "Embedded software memory size estimation using COSMIC: a case study," IWSM/MetriKon/Mensura, Stuttgart, Germany, 2010.
- [31] Kenneth Lind, and Rogardt Heldal, "Estimation of Real-Time Software Code Size using COSMIC FSM," IEEE Intl. Symposium on Object/component/service-oriented Real-time distributed Computing - ISORC 2009, pp. 244-248.
- [32] S. Di Martino, F. Ferruci, C. Gravion, F. Sarro, "Web Effort Estimation: Function Point Analysis vs. COSMIC," Information and Software Technology 72, 2016, pp. 90-109.
- [33] Hassan Soubra, Alain Abran, Sophie Stern, and Amar Ramdane-Cherif, "Design of a Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed using the Simulink Model," IWSM-MENSURA, Nara, Japan, IEEE CS Press, 2011, pp. 76-85.
- [34] Renault S.A., "Design of a Functional Size Measurement Tool for Real-Time Embedded Software Requirements Expressed Using a Simulink Model," MathWorks Automotive Conference. Stuttgart, 2012.
- [35] See <http://cosmic-sizing.org/organization/commercial-support/vendor-list/>
- [36] S. Bagriyanik, A. Karahoca, "Automatic COSMIC function point measurement using requirements engineering ontology", submitted to "Information and Software Technology," 2016.
- [37] J. Swierczek, "Automatic COSMIC sizing of requirements held in UML," in the "COSMIC Masterclass," IWSM/Mensura Conference 2014 www.iceaonline.com, Tool available from <http://300dc.pl/oferta/standardy-modelowania/>
- [38] A. Akca, A. Tarhan, "Run-time measurement of COSMIC functional size for Java business applications: is it worth the cost?" IWSM/Mensura Conference, 2013.

- [39] H. Huigens, M. Bruntink, A. van Deursen, T. van der Storm, F. Vogelezang, "Exploratory Study on Functional Size Measurement based on Code," International Conference on Systems and Software Processes - ICSSP, 2016 DOI 10.1145/2904354.290436
- [40] Kenneth Lind and Rogardt Haldal, "A Model-Based and Automated Approach to Size Estimation of Embedded Software Components," in ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems, Wellington (New Zealand), 2011.
- [41] Olavo Mendes, "Développement d'un protocole d'évaluation pour les outils informatisés de comptage automatique de points de fonction," Master's thesis, Computer Science Department, Université du Québec à Montréal (Canada), 1996.
- [42] Hassan Soubra, Alain Abran, Amar Ramdane Cherif 'Verifying the Accuracy of Automation Tools for the Measurement of Software with COSMIC – ISO 19761 including an AUTOSAR-based Example and a Case Study,' Joint 24th International Workshop on Software Measurement & 9th MENSURA Conference, Rotterdam (The Netherlands), Oct. 6-8, 2014, IEEE CS Press, pp. 23-31.
- [43] Keith Paton and Alain Abran, "A Formal Notation for the Rules of Function Point Analysis," Research Report 247, Montreal: Computer Science Department, Université du Québec à Montréal, Canada, 1995.
- [44] <http://www.estaca.fr/hassan-soubra/>
- [45] H. Soubra, L. Jacot, S. Lemaire, "Manual and automated functional size measurement of an aerospace real-time embedded system : a case study based on SCADE and on COSMIC ISO 19761," International Journal of Engineering Research and Science Technology - IJERST, Vol. 4 n°2, pp. 79-100, 2015.
- [46] H. Soubra, A. Abran and M. Sehit, "Functional size measurement for processor load estimation in AUTOSAR," Joint 25th International Workshop on Software Measurement and 10th MENSURA Conference, Kraków, Poland, October 5-7, 2015, Lecture Notes in Business Information Processing - Springer, vol. 230, pp. 114-129, 2015.
- [47] <http://www.autosar.org>
- [48] Heinecke, Harald et al., "AUTOSAR – Current results and preparations for exploitation," Euroforum Conference, May 3, 2006.
- [49] H. Fennel et al., "Achievements and Exploitation of the AUTOSAR Development Partnership," SAE Convergence Congress, Detroit, MI, 2006.
- [50] H. Soubra, and K. Chaaban, "Functional Size Measurement of Electronic Control Units Software Designed Following the AUTOSAR Standard: A Measurement Guideline Based on the COSMIC ISO 19761 Standard," Joint 22nd International Workshop on Software Measurement and 7th MENSURA Conference, IEEE CS Press, 2012.
- [51] GAO, "Cost Estimating and Assessment Guide," <http://www.gao.gov/new.items/d093sp.pdf>, March 2009.
- [52] NIST, "A Rational Foundation for Software Metrology," National Institute for Standards & Technology, NIST IR 8101, January 2016.