Automated COSMIC Measurement and Requirement Quality Improvement Through ScopeMaster® Tool

Erdir Ungan[1], Colin Hammond[2] and Alain Abran [3]

[1] Université du Québec à Montréal – UQAM (Montréal, Canada)
[2]Albion Technology Ltd – (Marlow, Buckinghamshire, United Kingdom)
[3]École de Technologie Supérieure – ETS, University of Quebec (Montréal, Canada)
ungan.erdir@uqam.ca colin@albiontech.com alain.abran@etsmtl.ca

Abstract. This paper presents a new COSMIC functional measurement automation tool ScopeMaster®, which automatically generates a size estimation from textual requirements. The tool points out problems in requirements in clarity, completeness, consistency and concision. This facilitates both COSMIC measurement and quality inspection processes. Utilizing ScopeMaster guides users to create higher quality requirements. It improves measurement accuracy, increases the number of defects found in inspections and drastically reduces the effort for both activities. Enabling higher number of requirement defects to be found early in the SDLC has a huge effect on overall software quality and rework costs. ScopeMaster also provides detailed reports on project estimates and measurement details.

Keywords: COSMIC, Functional Size Measurement, Automation, Requirement Quality, Requirement Defects

1 Introduction

This paper presents the tool ScopeMaster® developed by Albion Technology Ltd. and discuss its benefits in terms of better COSMIC measurements, defect detection and improved requirement quality. ScopeMaster® is a COSMIC Functional Size Measurement tool that takes free form textual software requirements as input.
The tool, basically:

- detects the probable data movements and so measures them using the COSMIC FSM
- points out potential defects within the requirements
- generate project estimates based on COSMIC size

Automating functional size measurement with COSMIC is one of the top priorities in research in COSMIC community today. It is widely accepted that automating COSMIC measurements is crucial for its acceptance in industry.

There exist numerous proposals for tools that support COSMIC measurement (automated and non-automated) and new ones are emerging every day [1-8]. These tools can be grouped based on the main functionality they provide as [1,9]:

- Data collection and calculation: These tools make it easier to record and manage the measurement data. Measurement is performed manually, and the tools enable entering measurement data in an orderly fashion and keep the meta data about the measurements.
- Expert systems for measuring (Measurement Facilitation): These tools enable measurement to be performed within the tool. They enable entering and managing pre-set constructs for COSMIC measurements such as objects within a system, standard users, standard functionality. They may also provide basic checks for measurement rules to improve measurement quality. Measurement is still performed manually but with guidance and facilitation of expert systems for measuring.
- Automated Measurement:
  - Based on Structured Input: These tools perform COSMIC measurement autonomously and automatically utilizing structured inputs such as:
    - Structured/Formalized Functional Requirements
    - UML Diagrams
    - Software Design Models
    - User Interfaces
    - Source Code
  - Based on unstructured input: These tools perform measurement on free form textual requirements.

There exist a wide scope of both academic and industrial research potential on functional size measurement automation. Most of the existing research and tools utilize formalized input for measurement such as structured requirements, conceptual models, UML models, source code or similar constructs.

To the best of our knowledge there is only limited academic research on automating COSMIC measurement using textual requirements in natural language as input [10][11] and there exist no commercial tool that implements that.

The paper is structured as follows, Section 2 summarizes how applying COSMIC measurement to requirements inherently improves their quality. Section 3 presents ScopeMaster® including its features, generated reports and its current limitations. Section 4 describes how it helps improving requirement quality through pointing out defects and helping analysts fix those defects. Section 5 presents a summary and lists the road map for intended additional features for the tool.

2 Requirement Quality and COSMIC

Quality of software requirements hugely impact the quality of any software product as well as any measurement result based on them [12]. Between 16% and 20% of defects in software are requirement defects [13]. Moreover, any defect in requirements cost much higher to fix compared to defects injected in further phases of SDLC. The rework cost to fix a defect injected in requirements increase exponentially as it propagates through lifecycle phases. As illustrated in Fig. 1, rework cost for defects found later in the SDLC are significantly more expensive than those found in earlier phases.

Pressman [14] states the of the cost of fixing a defect in requirements within requirements phase is 1 unit, cost to fix it in coding, test and deployment phases will be multiples of 1, 6.5, 15 and 80.

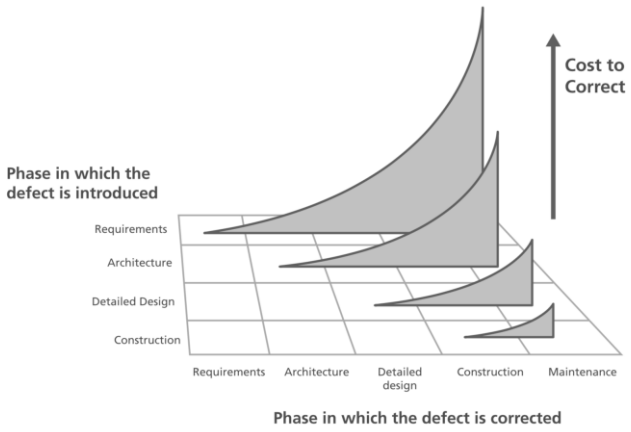

Fig. 1. Respective costs of correcting defects created in different phases of SDLC [14]

Based on these, it is imperative to detect and fix a defect in requirements as early as possible to minimize rework costs. For this, organizations typically perform reviews on their requirements documents. Reviews are typically performed to make sure that requirements have some generic and domain specific quality attributes. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications [15] defines a set of quality attributes for requirements such as being:

- Clear (Unambiguous)
- Complete
- Consistent
- Concise
- Correct
- Current

However, free form requirements expressed in natural language are prone to being vague, long and incomplete.

Like other recognized Functional Size Measurement (FSM) methods [16-19] COSMIC [20] requires the functional requirements to be defined in a certain level of detail and quality. In order for the measurement rules to be properly applicable, requirements should possess certain qualities.

Users, objects and functions should be clearly identifiable. Any functional process within a requirement should include clear definitions for triggering events, users, inputs, outputs and functional steps.

Studies showed that whenever a requirement cannot be properly measured, that also points out an inadequacy in terms of basic requirement qualities [12][21][22][23]. This renders COSMIC a valuable tool to verify requirement quality and detect defects extremely early in the software development life cycle (SDLC).

There are many studies that demonstrate how an FSM, and in particular COSMIC, can improve software specification quality through pointing out defects in requirements:

In [24], Talib et.al. illustrates COSMIC FSM can be used to asses the quality of specifications of a real-time software system. They state that COSMIC measurement especially helps in detecting ambiguous requirements and requirements whose hardware/software allocation is not clear.

In [22] Trudel and Abran demonstrate that using COSMIC measurement during inspections lead an increase of 16% to 32% in the number of identified critical functional defects. Moreover, the study also showed that inspectors spent only 54% of the planned effort for inspections when they utilized COSMIC measurement during inspections.

COSMIC Guideline for Measurement Accuracy [25] also proposes an approach to assess the quality of requirements in terms of COSMIC measurement principles. The assessment is performed in terms of:

- The presence or absence of a data model.
- The presence of absence of information to identify the data movements (entry, read, write, exit).
- The presence (or absence) of documentation enabling identification of each functional process.

3 ScopeMaster Tool

ScopeMaster® is a web-based tool which users login using their credentials and can create many measurement projects.

Once the user creates a measurement project in the tool, ScopeMaster® enables adding requirements one at a time or importing them in bulk via a CSV file as well as through integration with the JIRA[26] tool.

As the user adds user software requirements or user stories, ScopeMaster® performs several successive steps of analysis individually and collectively on the requirements in order to detect possible Objects of Interest, potential users, potential data movements and potential defects.

The details of how ScopeMaster® performs these techniques are proprietary and subject to a pending patent application, however the results are fully transparent. The underlying steps include natural language processing and several modules of pattern matching.

The tool focuses on detecting a “Subject verb object” structure to identify necessary measurement elements (functional users, objects of interest, data movements) from requirements.

A user story such as “As a user I want to display orders” has “user” as the subject, “display” as the verb and “orders” as the object. In this structure, the subject is a candidate for the Functional User, the object is a candidate for an Object of Interest and verb corresponds to one of Create, Read, Update, Delete, List (CRUDL) set of functions.

Below, we present features of the tool through the example of C-REG COSMIC Case study[27].

Each requirement consists of four fields: Title and Body, which are analyzed for possible interpretation, ID and Notes, which are searchable but not analyzed for functional interpretation. The Body field is the main focus of the requirement. The tool encourages this field to be a succinct but complete statement of the overall purpose of the requirement or user story. The scenarios, conditions and success criteria should all be put into the notes field.

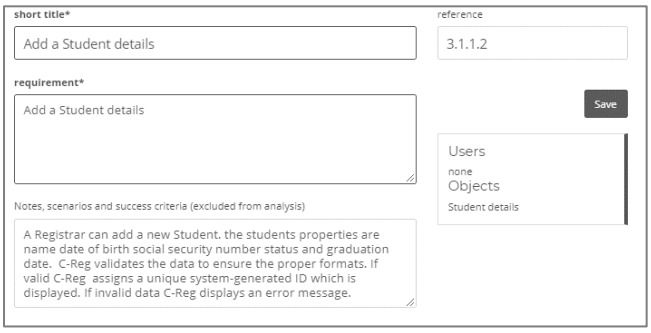

Fig. 2. Adding Requirements

ScopeMaster® believes that the most important characteristic to get right first, is what is the requirement's main purpose. The main purpose of a requirement can be different functionalities such as:

- "Update requirements"
- "Maintain calendar entries"
- "Display orders", "Delete invoice",
- "Search for companies"
- "Book a room"

Information such as the triggering event conditions, the scenarios in which the requirement applies and the outcomes to be tested are all deemed peripheral to the primary purpose of the requirement and should be put into the notes field.

As soon as the requirement is added, the tool measures the size the requirement and displays the Functional Process, Objects of Interests, CRUDL operation type and Data Movements. Tool also assesses the readability of the requirement and color codes the requirement to indicate any problems in its readability.

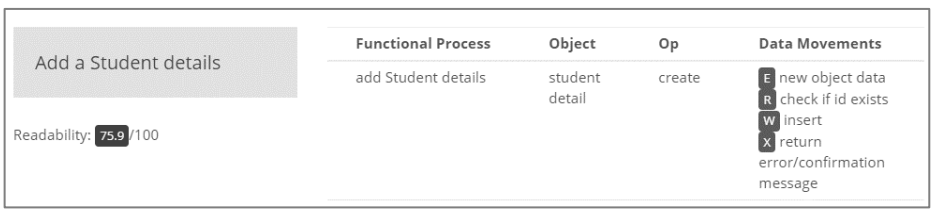

Fig. 3. Analysis Details for a Requirement

Requirements are displayed as a list with their corresponding estimated sizes next to them. The color of the size information indicates whether the requirement is potentially too big or involves different functional changes to multiple objects of interest.

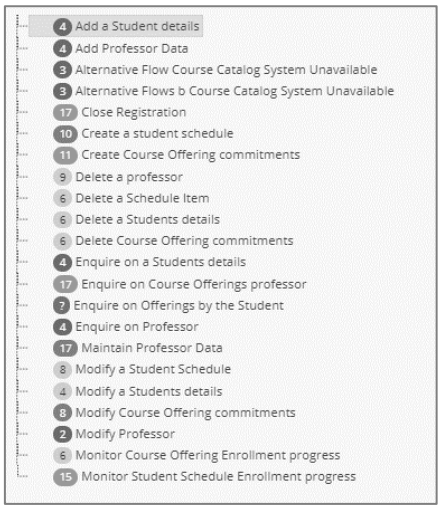

Fig. 4. List of Requirements and Corresponding Size Estimates

3.1 Reports

ScopeMaster® generates a number of reports for the project based on the analysis of requirements. Primary analysis outcomes include:

Reporting of ambiguous requirements. ScopeMaster reports possible ambiguities in the requirement texts which may lead to a reader interpreting the requirement differently from the author's intent. Tool highlights words causing ambiguity in a sentence and advise the author to revise them (see **Fig. 5.**) and also generates reports for requirements that are not concise and that require multiple verbs for a single operation (see **Fig. 6.** Ambiguous or verbose requirement warning

Fig. 5. Ambiguous wording warning

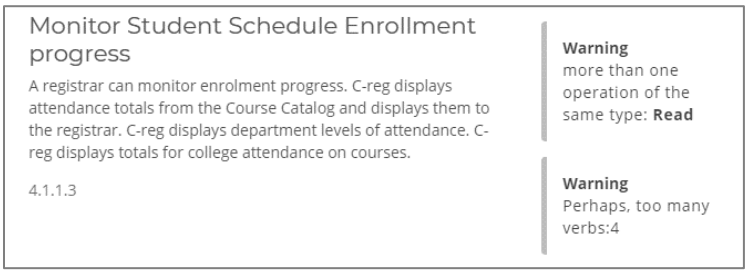

Fig. 6. Ambiguous or verbose requirement warning

Report of detected potential users and objects of interest. Tool lists all of the suggested users and suggested objects in the project with their number of occurrences in the requirements.

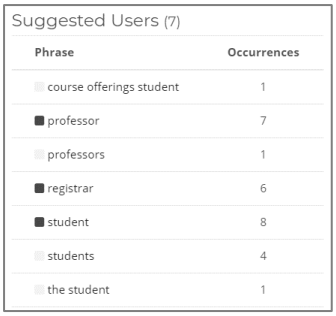

Fig. 7. List of users and number of their occurrences in the requirements

Report potential missing and duplicate requirements. The tool shows which data movements correspond to CRUDL operations on any given object of interest and reports on potential missing and duplicate requirements. It assumes whenever an object is maintained in the system through one or more of CRUDL operations, it indicates other activities should be defined in a requirement as well. It also detects if there are multiple requirements depicting one of these operations on an object. This may point out to inconsistencies among requirements and flags those requirements for further inspection.

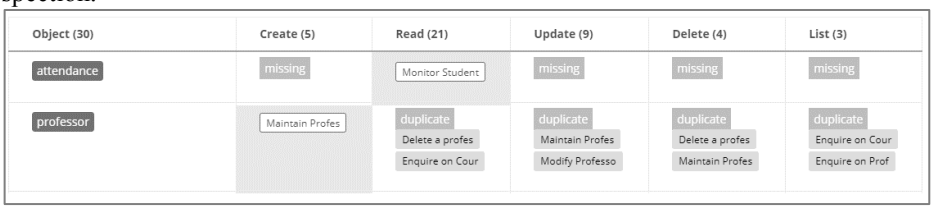

Fig. 8. Report on potential defects for missing and duplicate operations

Secondary analysis steps derive:

IFPUG sizing estimates. ScopeMaster® also generates size estimates and reports for IFPUG [16] measurement. However, the estimates and reports for IFPUG are less accurate than those for COSMIC due to the sensitivity of detecting actual ILFs. Any wrongfully identified ILF in IFPUG measurement has a much bigger impact on accuracy then any wrongfully identified Object of Interest in COSMIC measurement.

Project estimates for cost, duration, resources needed: The tool generates indicative estimates and gives a range for development cost, duration and resources. It also gives an estimate for scope creep and expected number of defects in all phases of SDLC. Most of these estimations are based on public benchmark data.

Classes and Methods. The tool generates a list for potential classes and method for implementation, based on the objects and operations identified from the requirements.

3.2 Limitations

ScopeMaster® interprets the English used to write the requirements or user stories.

Currently this interpretation is not 100% accurate, but algorithms are being continuously improved, through formal verification and through machine learning. It is expected to reach a consistent accuracy in excess of 85%. Currently results are in the 70-95% accuracy range. Dealing with the nuances of the natural language renders %100 accurate interpretation almost impossible. However, the clearer the requirements the more accurate the counts will be. Moreover, any measurement with the tool is repeatable, that is, given the same set of requirements it always produces the same measurement results.

Currently the tool can only analyze the main function within a requirement. It does no text analysis of the success criteria or triggering events mentioned in the requirement text. This sometimes result in mixed results if there are too much side information given within the requirement text.

Similarly, interpretation accuracy decreases when there are multiple sophisticated condition based requirements within a text. It is suggested that such requirements are broken down into smaller requirements.

Ontology used is not customizable at the moment. For example the list of recognized verbs is fixed and same for each project. There might a need for nuances or combinations of verbs for different development contexts. It is planned to make ontology customizable in the future.

4 Improvements in Measurement Process and Requirements Quality Introduced by ScopeMaster®

We believe COSMIC size estimation performed by ScopeMaster® provides a very effective starting point to fine tune the measurement manually. As ScopeMaster® displays the interpreted data movements of every FUR, the user has full transparent visibility as to the makeup of the count and can adjust the wording of the requirement appropriately.

In its current state, ScopeMaster® size estimates are observed to be typically within 20-30% of manual count equivalents. This was also verified with the case study explained above. Given that manual COSMIC estimates also tend to have a level of discrepancy among different measurers in manual measurement [28], an automated estimate in these ranges pose a fairly good basis especially when the automated results are reviewed and fixed manually.

ScopeMaster® also helps making requirements "measurable" by pointing out possible problems regarding the requirements in the measurement process. It allows requirement texts to be improved by giving constant feedback about their measurability and ambiguity. Requirements can be tweaked and updated until the desired structure is attained for the COSMIC measurement. This, in turn, results in improving the quality attributes of requirements mentioned in section 2:

- By pointing out too long or too short requirements, it enables requirements to be defined in appropriate length for measurement. It prevents many Functional Processes to be combined under a single Functional User Requirement or User Story. This enable the requirements to be more **concise**.
- By pointing out inconsistencies between the number of data movements detected in a requirement and the length of its text, it points out probable non-functional or unnecessary information cramped in a functional user requirement. This enables the requirements to be **clear** and **atomic**.
- By pointing out similar objects and users among requirements, it highlights naming inconsistencies so that they can be corrected. This enables the requirements to be **consistent** with each other.
- By pointing out potentially missing operations (eg. CRUDL) it helps requirements to be **complete**.
- By pointing out multiple operations of same type (eg. CRUDL) on a given object of interest, it enable detection of any inconsistencies among requirements and enables requirements to be **consistent** and **correct**.
- There is a limited list of verbs that are recognized by the tool. Authors are required to use verbs from that list for their requirements to be measurable. This forces requirements to be written in a more formalized fashion and in turn becoming more **concise** and **clear**.

ScopeMaster® also decreases the time and effort needed for both COSMIC measurement and finding and fixing requirement defects.

Typically a COSMIC certified measurer can measure 125-500 CFPs per day, however, using ScopeMaster® this productivity goes up to 500-2500 CFPs a day. The tool performs the measurement analysis in a matter of minutes, these daily CFP numbers are for cases when the measurement results that are analyzed, fixed and verified by the measurer.

Moreover, when the tool's measurement results are used as a basis for measurement, it standardizes measurement throughout an organization. This will also reduce discrepancy among measurement results from different measurers and increase repeatability of results.

Manual requirements quality inspections typically require half an hour in a 4 people meeting to groom and find 1 defect and requires half a day of work for 1 person to fix it. This totals 5-6 hours of effort for each defect found and fixed manually. The ScopeMaster® authors claim that a single person can find and fix a defect in as little as 15 minutes. The results of our case study described below also proved this improvement.

This improvement is due to the fact that the when the author of a requirement is going through and fixing the initial COSMIC estimations of ScopeMaster®, he is also fixing most of the defects in the requirements.

4.1 Case Study

In order to demonstrate how ScopeMaster® can benefit practitioners, we conducted a small case study. This study was conducted as a proof of concept rather than an academic study.

A senior software project assurance expert participated in the study and was asked to analyze a software specification for a real life game software using ScopeMaster® without any prior experience with the tool.

Specifications consisted of 80 user stories at start and increased to 90 user stories by the time the study finished.

When the requirements were run, ScopeMaster® counted 400 CFPs and discovered over 200 potential defects in less than 60 seconds.

The participant was able to go through all 90 requirements and fixed 150 requirements defects within 16 hours. This corresponded to finding and fixing defects at a rate of 1 every 10 minutes.

Participant commented that while he was working through requirements, it was possible to review measurement results, see defects, fix defects and add missing requirements all at the same time.

5 Summary and Future Work

In this paper we presented some of the features of the automatic COSMIC size measurement tool ScopeMaster®. To our knowledge this tool is the first commercial tool that performs COSMIC measurement on free form textual requirements.

We believe it has a great potential in not only automating COSMIC size measurement from requirements in natural language but also improving the quality of requirements. It allows users to immediately see if their requirements are measurable and enables them to revise and update them on the fly by proving constant feedback.

As shown by many studies, the process of making requirements measurable, on its own, improves quality of requirements. ScopeMaster® improves requirement quality through COSMIC measurement principles and on top of this, it explicitly points out defects in clarity, completeness, concision and consistency.

Using the tool to facilitate COSMIC measurement improves the accuracy of measurement results compared to purely manual measurement. It significantly reduces measurement effort. It also enables requirement defects to be detected and fixed with much less effort compared to manual inspections.

We believe, ScopeMaster® would particularly benefit big organizations, consisting of many teams that similar software with similar requirements, by streamlining their measurement and requirement verification activities.

On the other hand, the tool lets practitioners to utilize COSMIC principles and measurement results without any prior knowledge about the method. This means, whenever ScopeMaster® is deployed in an organization for quality improvement and defect detection purposes, it will be indirectly introducing COSMIC Functional Size Measurement and its benefits to the organization as well. This would be particularly beneficial for the COSMIC community to widen the methods adaption as industry always demands tools to mitigate rework cost and improve software quality however, it is not always easy to get organizations appreciate the value added by FSM methods.

The tool is under constant development and improvement. In the near future these features are expected to be added:

- Managing measurement elements
- Custom ontology: Making ontology used for requirements such as making the verb list customizable for organizations or defining a pre-determined set of objects to use for entire organization. This will enable organizations better tune the interpretation of their requirements and also will let them introduce and enforce usage of their own requirement standards and wording guidelines.
- Requirements suggestions engine (re-use, security): Certain frequent and typical requirements will be presented in the form of functional patterns. Certain generic functions such as security functions will be suggested based on type of identified objects. This will further prevent facilitate requirement development and prevent missing requirements.
- More Machine Learning and better language interpretation.
- Multilanguage support: It is expected to include other language than English for requirement interpretation.
- Local Benchmarking: Currently certain reports utilize generic ratios and productivity values from public benchmarking data sets. It is planned to enable organizations utilize their internal benchmarking data sets and standard values.
- Test suggestions: The tool will suggest a list of potential unit tests for any given method on an object. cases for requirements.

References

1. Mendes O., Abran A., Bourque P., (1996), Function Point Tool Market Survey Software Engineering Management Laboratory, Université du Québec à Montréal, Montréal, Canada.
2. Fraternali, Piero & Tisi, Massimo & Bongio, Aldo. (2006). Automating function point analysis with model driven development. 233-247. 10.1145/1188966.1188990.
3. Soubra, Hassan & Abran, Alain & Stern, Sophie & Ramdan-Cherif, Amar. (2011). A Refined Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed Using the Simulink Model. 76-85. 10.1109/IWSM-MENSURA.2011.52.
4. Lind, Kenneth & Heldal, Rogardt. (2011). CompSize: A Model-Based and Automated Approach to Size Estimation of Embedded Software Components. IEICE Transactions on Information and Systems. E95.D. 334-348. 10.1587/transinf.E95.D.2183.
5. Marín, Beatriz & Pastor, Oscar & Giachetti, Giovanni. (2008). Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment. 5089. 215-229. 10.1007/978-3-540-69566-0_19.
6. Lamma, Evelina & Mello, Paola & Riguzzi, Fabrizio. (2003). A System for Measuring Function Points from an ER-DFD Specification. 47. 10.1093/comjnl/47.3.358.
7. Uemura, Takuya & Kusumoto, Shinji & Inoue, Katsuro. (1999). Function point measurement tool for UML design specification. Journal of Software Maintenance and Evolution: Research and Practice. 13. 62 - 69. 10.1109/METRIC.1999.809727.
8. Bagriyanik, Selami & Karahoca, Adem. (2016). Automated COSMIC Function Point measurement using a requirements engineering ontology. Information and Software Technology. 72. 10.1016/j.infsof.2015.12.011.
9. R. Dumke and A. Abran, COSMIC Function Points: Theory and Advanced Practices. 2011
10. Hussain, Ishrar & Ormandjieva, Olga & Kosseim, Leila. (2009). Mining and Clustering Textual Requirements to Measure Functional Size of Software with COSMIC.. 599-605.
11. Linguistic Approaches for Early Measurement of Functional Size From Software Requirements. H M Ishrar Hussain, Doctoral Thesis, Concordia University, Montreal, Quebec, Canada, August 2014
12. The Effect of the Quality of Software Requirements Document on the Functional Size Measurement.. G. Yılmaz, E. Ungan, O. Demirörs. United Kingdom Software Metrics Association International Conference on Software Metrics and Estimating. London, UK. 2011
13. Jones C., Bonsignour O., Economics of Software Quality, Addison-Wesley, 2011.
14. Pressman, Roger S., Software Engineering, A Practitioner's Approach, 3rd Edition, McGraw Hill, New York, 1992. p.559.
15. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
16. ISO/IEC ISO/IEC 20926:2009. Software and systems engineering - Software measurement - IFPUG functional size measurement method 2009, 2009.
17. ISO/IEC ISO/IEC 20968:2002, Software engineering - Mk II Function Point Analysis — Counting Practices Manual, 2002.
18. ISO/IEC 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, 2005.
19. ISO/IEC ISO/IEC 29881:2010, Information technology - Systems and software engineering -FiSMA 1.1 functional size measurement method, 2010
20. COSMIC Group, 2015. The COSMIC Functional Size Measurement Method Version 4.0.1 Guideline on Non Functional & Project Requirements. http://www.cosmic-sizin.org.

21. Desharnais, Jean-Marc & Kocatürk, Bugra & Abran, Alain. (2011). Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories. 269-272. 10.1109/IWSM-MENSURA.2011.45.
22. Trudel S., Abran A. (2008) Improving Quality of Functional Requirements by Measuring Their Functional Size. In: Dumke R.R., Braungarten R., Büren G., Abran A., Cuadrado-Gallego J.J. (eds) Software Process and Product Measurement. Mensura 2008, MetriKon 2008, IWSM 2008. Lecture Notes in Computer Science, vol 5338. Springer, Berlin, Heidelberg
23. J.-M. Desharnais, A. Abran, Assessment of the Quality of Functional User Requirements documentation using criteria derived from a measurement with COSMIC- ISO 19761, I International Workshop on Software Measurement - IWSM 2010, Stuttgart, Germany, November 2010, pp. 481-496.
24. Abu Talib M., Khelifi A., Abran A., Ormandjieva O. (2008) Assessment of Real-Time Software Specifications Quality Using COSMIC-FFP. In: Cuadrado-Gallego J.J., Braungarten R., Dumke R.R., Abran A. (eds) Software Process and Product Measurement. Mensura 2007, IWSM 2007. Lecture Notes in Computer Science, vol 4895. Springer, Berlin, Heidelberg
25. COSMIC, Guideline for Assuring the Accuracy of Measurements, v 0.92, Common Software Measurement International Consortium, 2011. URL: www.cosmicon.com
26. JIRA Software https://www.atlassian.com/software/jir
27. Lesterhuis, Arlan, Alain Abran, & Charles Symons. 2015. Course Registration ('C-REG') System Case Study, Version 2.0. www.cosmic-sizing.org
28. Ungan,E., Demirörs O., Top O.O, Özkan B. An Experimental Study on the Reliability of COSMIC Measurement Results.. Lecture Notes in Computer Science, Volume 5891/2009, 321-336.Springer Berlin / Heidelberg. 2009.
29. Soubra, Hassan & Abran, Alain & Ramdane-Cherif, Amar. (2014). Verifying the accuracy of automation tools for the measurement of software with COSMIC - ISO 19761 including an AUTOSAR-based example and a case study. 23-31. 10.1109/IWSM.Mensura.2014.26.
30. Trudel S., Buglione L., Guideline for sizing Agile projects with COSMIC, International Workshop on Software Measurement - IWSM 2010, Stuttgart, Germany, November 2010.
31. Weinberg, Gerald M., Quality Software Management, Volume 1, Systems Thinking, Dorset House, New York. 1992.
32. ISO/IEC 19761 Software Engineering - COSMIC - A Functional Size Measurement Method, 2011