COSMIC-FFP & Functional Complexity (FC) Measures: A Study of their Scales, Units and Scale Types

Manar Abu Talib**, Alain Abran*, Olga Ormandjieva**

*École de Technologie Supérieure - ETS

1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3

**Concordia University

1455 de Maisonneuve Blvd. W. Montreal, Quebec H3G 1M8, Canada

m_abutal@cse.concordial.ca , alain.abran@ele.etsmtl.ca , ormandj@cse.concordia.ca

Abstract:

This paper presents an overview of some measurement concepts across both COSMIC-FFP, an ISO standard (ISO/IEC 19761) for functional size measurement and Functional Complexity (FC), an entropy-based measure. It investigates in particular three metrological properties (scale, unit and scale type) in both of these measurement methods.

Keywords

COSMIC-FFP, ISO 19761, Entropy, Scale type, Measurement process model.

1 Introduction

The COSMIC-FFP [1] functional size measurement method was developed by the Common Software Measurement International Consortium (COSMIC) and it has been adopted as an international ISO standard: ISO 19761 [2]. COSMIC-FFP measures the software functional user requirements and is applicable throughout the development life cycle, right from the requirements phase to the implementation and maintenance phases [1].

This method has been designed to measure the functional size of management information systems, real-time software and multilayer systems. Since many of the software systems targeted by the COSMIC-FFP method are large-scale and inherently complex, feedback on this complexity would provide additional information to improve their effective management throughout the software life cycle: in [3] an entropy-based measure of functional complexity has been proposed.

This paper presents a study of the scales, units and scale types of both COSMIC-FFP and an entropy based functional complexity measure. Previous studies have analyzed the scale types of many software measures (such as: Zuse

[4], Fenton [5], Whitemire [6], but not the concept of 'scale' nor how it is used in the design of a measurement method.

Well designed and well defined measures in sciences and engineering should have most of the many characteristics as described in metrology [7], including 'scales', 'units' and 'etalons' to which should refer measuring instruments to ensure meaningfulness of the numbers obtained from measurement. However, some of these concepts, such as units, scale, and etalons, are not yet addressed and discussed by researchers on empirical validation approaches [5] of software measures: for instance, researchers on software measure have, to date, focused on scale types rather than on the scale embedded within the definitions of these measures. This could lead to less than optimally designed 'software measures'. Moreover, when these software measures are analyzed without taking into account these metrological concepts, it can lead to improperly stated conclusions about their strengths.

In this paper, section 2 presents the key elements of scale types, section 3 the key elements of COSMIC-FFP and section 4, the key elements of FC, an entropy-based measure of functional complexity measurement. In section 5 & 6, the scale, units and scale types of the both measures are investigated and, finally, a discussion and some future next steps are presented in section 7.

2 Scale Types

In measurement theory, the meaning of numbers is characterized by scale types [ ], but measurement theory does not address directly the concept of scale, as typically defined in metrology. A scale is defined as a set of ordered values, continues or discrete, or a set of categories to which the attribute is mapped [8], whereas scale type depends on the nature of the relationship between values on the scale [8].

In a mathematical representation, a scale is defined by <E, N, F>, where E is the empirical structure, N is the numerical structure and F is the mapping between them [ ]. On the other hand, a scale type is always defined by admissible transformations. Relationships between mappings are described in terms of transformations [6]. There are five major scale types: nominal, ordinal, interval, ratio and absolute, which can be seen describing certain empirical knowledge behind the numbers [6]. Knowing the characteristics of each type helps to interpret the measures [5]. In the following subsections, the scale types are summarized.

2.1 Nominal Scale Type

The nominal-scale measurement places elements in a classification scheme [5], [6]. The classification partitions the set of empirical entities into equivalence classes with respect to a certain attribute. Two entities are considered as

equivalent and therefore belonging to the same equivalence class if and only if they have the same amount of the attribute being measured. The empirical classes are jointly exhaustive and mutually exclusive. The classes are not ordered because of a lack of empirical knowledge about relationships among the classes. In nominal-scale measurement, each empirical class might be represented by a unique number or symbol, and the only mathematical operation allowed in the nominal scale type is "=". The admissible transformations are one-to-one mapping that preserve the partitioning.

2.2 Ordinal Scale Type

The ordinal scale type is the basis of software measurement. All other extended measurement structures are based on the ordinal scale [4]. Ordinal scale assigns numbers or symbols to the objects so they may be ranked and ordered with respect to an attribute [6]. The characteristic of ordinal scale is that the numbers represent ranking only, so addition, subtraction and other arithmetic operations have no meaning. Also, any mapping that preserves the ordering is acceptable as ordinal scale [5].

2.3 Interval Scale Type

Interval scale type is useful to augment the ordinal scale with information about the size of the intervals that separate the classes. That is, the difference in units between any two of the ordered classes in the range of the mapping is known, but computing the ratio of two classes in the range does not make sense. This scale type preserves order, as with an ordinal scale; however, in interval scales addition and subtraction are acceptable operations. Multiplication and division are not acceptable operations in this scale type [5].

2.4 Ratio Scale Type

A ratio scale type is an interval scale with ratio on which there exists an absolute zero. This zero element represents the smallest scale value, where an object has a null amount of the attribute. Therefore, the measurement mapping in ratio scale must start at zero and increase at equal intervals, known as units. All arithmetic in ratio scale can be meaningfully applied to the classes in the range of the mapping [5].

2.5 Absolute Scale Type

Absolute scale type represents counts of objects in a specific class. There is only one possible measurement mapping, namely the actual count, and a unique unit. As in ratio scale, all arithmetic analysis of the resulting count is meaningful [5]. More details in these scale types can be found in [4], [5] and [6].

3 COSMIC-FFP Measurement Method

3.1 COSMIC-FFP Overview

The COSMIC-FFP method has been designed to measure the functional size of management information systems, real-time software and multi-layer systems. Its design conforms to all ISO requirements (ISO 14143-1 [9]) for functional size measurement methods, and was developed to address some of the major weaknesses of the earlier method – that is Function Points Analysis - FPA [10], the design of which dates back almost 30 years when software was much smaller and much less varied. COSMIC-FFP focuses on the "user view" of software functional requirements and is applicable throughout the development life cycle, right from the requirements phase to the implementation and maintenance phases.

In the measurement of software functional size using the COSMIC-FFP method, the software functional processes and their triggering events must be identified [1], [2].

In COSMIC-FFP, the unit of measurement is a data movement, which is a base functional component which moves one or more data attributes belonging to a single data group. Data movements can be of four types: Entry, Exit, Read or Write. The functional process is an elementary component of a set of user requirements triggered by one or more triggering events either directly or indirectly via an actor. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The sub processes of each functional process are sequences of events, and comprise at least two data movement types: an Entry plus at least either an Exit or a Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. See Figure [2] for an illustration of the generic flow of data groups through software from a functional perspective.

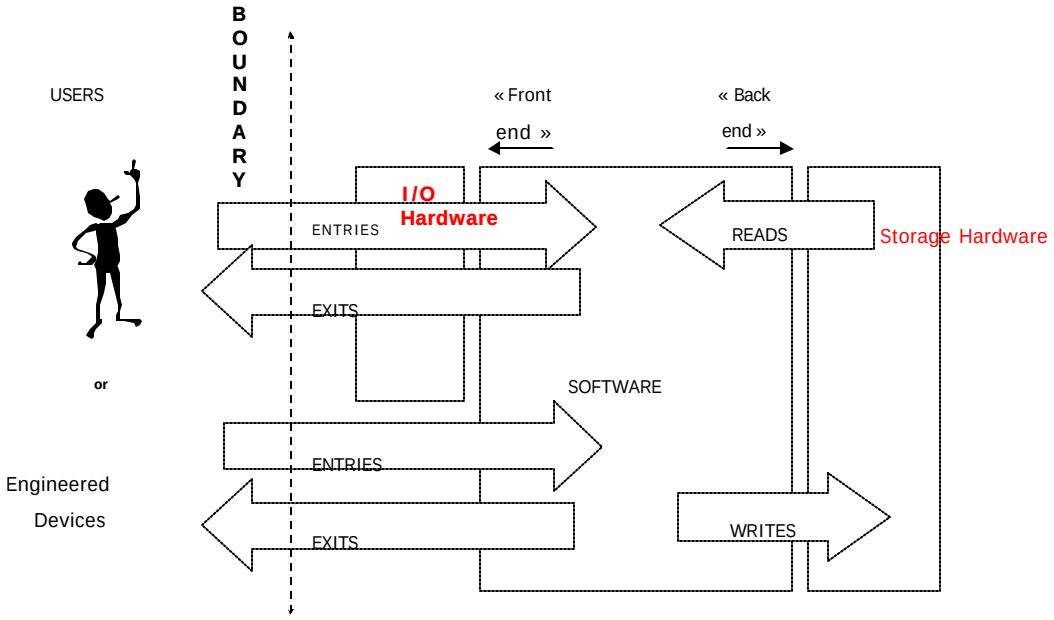

Figure 1: Generic flow of data groups through software from a functional perspective [1]

The COSMIC-FFP measurement method has two distinct phases: the mapping of the software to be measured to the COSMIC-FFP generic software model and the measurement of specific aspects of this generic software model. The following subsections describe in more details these two phases and they are summarized from COSMIC-FFP manual version 2.2.

3.2 COSMIC-FFP Mapping Phase

In all functional measurement methods, the functional size of software can not be measured directly from the Functional User Requirements (FUR): certain rules and procedures are to be applied to FUR of software to produce a specific software model that is suitable for measuring functional size. That technique is referred to as a "Mapping phase" in COSMIC-FFP. The general method for mapping software to COSMIC-FFP generic model is described in the Figure 2.

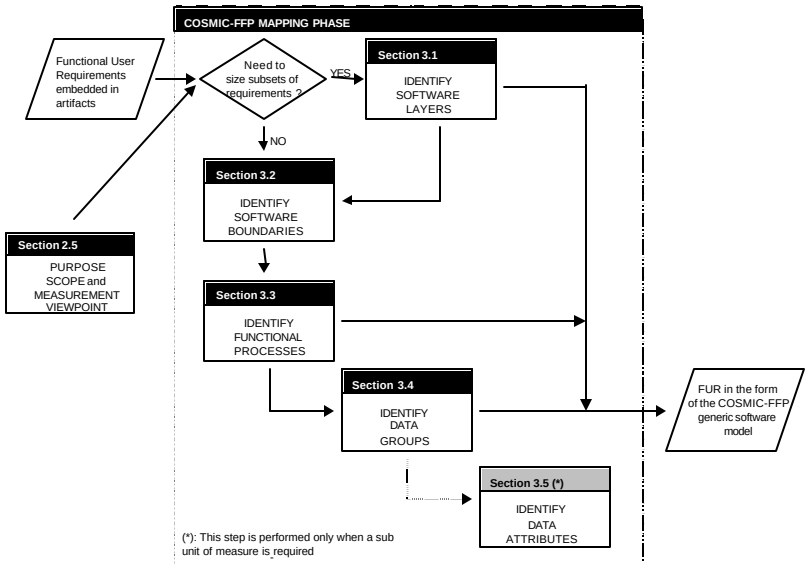

Figure 2: COSMIC-FFP Mapping Phase [1]

From Figure 2, before getting into mapping phase, the measurer must define why the measurement is being undertaken and/or what the measurement result will be used for. That is called "Purpose of a Measurement". The measurer also defines the scope of the measurement through the set of FUR to be included in a specific functional size measurement exercise. Finally, it is important for the measurer to identify the measurement view of the FUR of software. More definitions and principles regarding this context are provided in [1].

The COSMIC-FFP mapping phase takes the FUR of a piece of software as input to produce the COSMIC-FFP generic model of that software as output. The question: "Is there a need to size subsets of requirements?" will be raised as a first step in the mapping phase. That is because the FUR may apply to software in different layers or peer items. Therefore, the measurer needs to decide if the FUR or the software comprises one or more layers or peer items. A layer is a result of the functional partitioning of the software environment such that all included functional processes perform at the same level of abstraction [1]. In a multi-layer software environment, software in one layer exchanges data with software in another layer through their respective functional processes. It is to be noted that the layer identification is an iterative process. The exact layer will be refined as the mapping process progresses.

After identifying the software layers, the measurer must identify the boundary of each of each piece of software. According to COSMIC-FFP manual [1], the

boundary is defined as a conceptual interface between the software under measurement and its users (human beings, engineered devices or other software). The boundary allows the measurer to distinguish what is included inside the measured software from what is part of the measured software's operating environment.

The third step in the mapping phase is identifying the set of functional processes of the software to be measured from its FUR. A functional process is an elementary component of a set of FUR comprising a unique cohesive and independently executable set of data movements. It is triggered by one or more triggering events either directly or indirectly via an actor. It is complete when it has executed all that it is required to be done in response to the triggering event (-type) [1]. Once identified, each functional process can be registered on an individual line, under the appropriate layer, in the generic software model, under the corresponding label.

Identifying the data groups referenced by the software to be measured is the fourth step in mapping phase. A data group is a distinct, non empty, non ordered and non redundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest [1]. A data attribute is the smallest parcel of information, within an identified data group, carrying a meaning from the perspective of the software's FUR [1] and that is what will be identified in the last step of this phase. A data group must contain at least one attribute, and might contain one data attribute if this is sufficient, from the perspective of the functional requirements, to describe the object of interest.

3.3 COSMIC-FFP Measurement Phase

Measurement phase is the second phase considered in the COSMIC-FFP method. As described in figure 3, this phase takes the COSMIC-FFP generic model of software as input and produces a value of a quantity the magnitude of which is directly proportional to the functional size of the model [1].

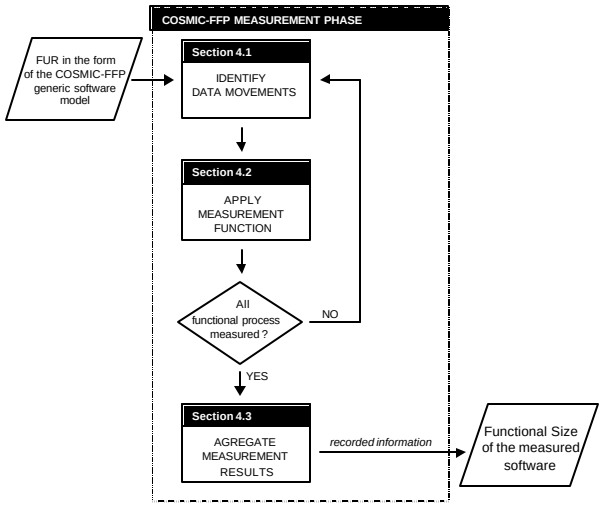

Figure 3: COSMIC-FFP Measurement Phase [1]

For each functional process, the measurer needs to identify the data movements' sub-process-types (entry, exit, read and write-types). That is the first step in this phase. Next the measurement method applies the COSMIC_FFP measurement function to each data movement identified in each functional process. According to this measurement function, each instance of a data movement (entry, exit, read or write) identified in step 1 receives a numerical size of 1 Cfsu. Finally, the measurer aggregates the results of the measurement function, as applied to all identified data movements, into a single functional size value arithmetically adding them together (formula 1).

$$Size_{Cfsu}\ (\text{functional process}_i) = \Sigma\ size(entries_i) + \Sigma\ size(exits_i) + \Sigma\ size(reads_i) + \Sigma\ size(writes_i) \ldots\ldots (1)$$

4 FC – A measure of functional complexity

Information theory-based software measurement was proposed in [11] to quantify functional complexity in terms of an amount of information based on some abstraction of the interactions between software components [12].

Entropy is one concept in information theory, and it was introduced by Shannon [13] as a quantitative measurement of the uncertainty associated with random phenomena. It is said that one phenomenon represents less uncertainty than a second one if we are more sure about the result of experimentation associated with the first phenomenon than we are about the result of experimentation associated with the second one.

Considering any set of n events and their probability distribution $\{ p_1, \ldots, p_n \}$, the quantification of this uncertainty quantity is calculated using the following entropy formula:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \quad \ldots\ldots (2)$$

In [14], a new method was proposed for quantifying functional complexity from a software behavior description. The method characterizes the functionality of the system as specified in the scenarios. Functional complexity is quantified in terms of the entropy of an amount of information based on an abstraction of the interactions among software components. Assuming that each message represents an event, therefore, entropy-based software measurement is used to quantify the complexity of interactions between the software and its environment and within the software (between software classes) in terms of the information content of the interactions, based on some abstraction of the interactions [15], [16], [17] and [18].

The probability p_i of the i-th event is equal to $p_i = f_i / \mathrm{NE}$, where f_i is the number of occurrences of the ith event and *NE* is the total number of events in the sequence. The classical entropy calculation quantifies the average amount of information contributed by each event. Therefore, the functional complexity in a time slice is defined in [3] as an average amount of information in the corresponding sequence of events and is computed as follows:

$$FC = -\sum_{i=1}^{n} (f_i / NE) \log_2 (f_i / NE) \quad \ldots\ldots (3)$$

where *n* is the number of different event types in the sequence.

We consider the COSMIC-FFP generic model of software as an abstraction of the interactions, thus conceptually justifying the applicability of the entropy to quantify the functional complexity in the COSMIC-FFP method. Functional complexity (FC) is the quantification for the amount of information interchanged in a given interaction (scenario). The functional complexity in a period of time with higher average information content should, on the whole, be more complex than another with lower average information content. That is, the FC measure is intended to order the usages of system in a time period in relation to their functional complexity.

The scale and scale types of both measurement methods are investigated next in sections 5 and 6.

5 Identification of COSMIC-FFP scale, unit and scale type

Measurement with COSMIC-FFP is more than counting and adding up the data movements. To identify the types of scales and analyze their uses in COSMIC-FFP measurement process, the procedure of the measurement process must be broken down into sub steps and each sub step is further analyzed in order to understand the transformation between the steps [10]. As mentioned previously, two phases (mapping and measurement) are required to measure the functional size of software in COSMIC-FFP. Basically, the mapping phase is the process of abstracting a set of FURs, described with whichever methodology, as a COSMIC-FFP generic model of the software. That is similar as if you want to map the distance on water into a meter scale or time into a dial of a mechanical clock. After that only, the measurer will be able to read the distance on the scale or read the specific position on the scale of the clock. Therefore, the mapping phase is an important step to map the FUR set into a measurement scale. This then gets the measurer into the next phase that is the measurement phase.

More specifically, the mapping phase is done by identifying software layers (**MAP 1**) and then for each layer the following steps are carried out:

MAP 1.1: Identifying software boundaries.

MAP 1.2: Identifying functional processes.

MAP 1.3: Identifying data groups.

MAP 1.3.1: Identifying data attributes.

For **MAP 1**, the concept of software layers in COSMIC-FFP is meant as a tool that identifies the separate components that have to be sized and their boundaries [1]. In a specific measurement exercise, layers can be distinguished and have an order where, for instance, software in any one layer can be a subordinate to a higher layer for which it provides services. Also, the measurement method defines "peer to peer" exchange, as two items of software in the same layer exchanging data [1]. From this point on, it can be said that a layer at level 2 is above a layer at level 1, which is used by the above layer or we can say that two softwares are at the same level or layer.

Next step, **MAP 1.1** is identifying the boundaries between each pair of layers where one layer is the user of another, and the latter is to be measured. In the same layer, there is also a boundary between any two distinct pieces of software if they exchange data in "peer to peer" communications [1].

MAP 1.2 is identifying the set of functional processes of software to be measured. In each layer, software delivers functionality to its own users. From at least one identifiable FUR, a functional process can be derived. A functional

process comprises at least two data movements, an entry and either an exit or a write [1].

Next, in (**MAP 1.3**) the data groups are identified. A data group is the object of interest that may or may not survive the functional process using it [1]. Each data group has a set of non empty and non ordered set of data attributes.

MAP 1.3.1 is the last step in the mapping phase. It considers identifying the data attributes for each data group.

After this analysis of the steps that are taken in the mapping phase, it is to be noted that the steps **MAP 1** to **MAP 1.3.1** by themselves are not taken into account in the measurement of COSMIC-FFP functional size: only 'data movements' are considered directly in the measurement with units of 1Cfsu as will be seen later in the measurement phase.

Figure 4 explains the contribution of the mapping phase to the measurement process. The measurand is basically the textual description of the text within which the Functional User Requirements are embedded in any kind of format. Then the 'measurement signal' would be basically the elements within the text that are related to the Functional User Requirements.

Next, the mapping from 'whichever format' into the 'generic COSMIC model of software' could be the 'transformed value'. It is to this 'transformed value' (e.g. the identification of all Functional Processes recognized by COSMIC-FFP) that the 'measurement function' would be applied with the corresponding measurement unit.

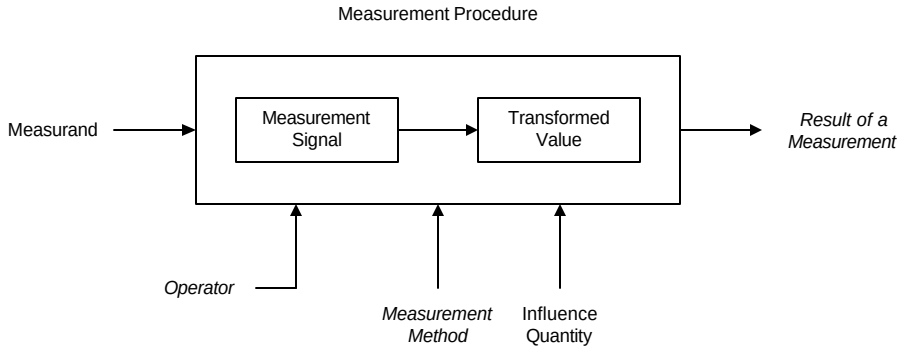

Figure 4: Measurement process - detailed topology of sub-concepts [19]

The second phase is the measurement phase where the measurer applies the measurement to the required elements of the model produced in the mapping phase. The Measurement phase is done for each functional process included within the software boundary identified in the mapping phase **(MAP 1.2)** and it is broken into three steps:

MSP1: Identifying data movements.

MSP2: Applying the measurement function.

MSP3: Aggregating the measurement results.

MSP1 identifies the data movements' types (Entry, Exit, Read and Write types) of each functional process [1]. It is to be noted that it not the sub-processes that are directly taken into account, but the data movements within a sub-process: in COSMIC a sub-process is defined as a combination of a 'data movement' & 'data manipulation'.

Then by convention, only a portion of the sub-process is taken into account in the use of a 'measurement scale', that is only the 'data movement' portion.

This could be similar to taking a sub-process, comparing it to a 'scale' of an 'etalon' and since the scale of the COSMIC etalon (defined by convention at a conceptual, rather that at a material level as for the 'meter' or the 'kilogram' etalons) considers only 'data movement', taking only this portion as input into the measurement function with its measurement unit, that is the 1Cfsu.

MSP2 is the next sub step in the measurement phase. It is applying the measurement function by assigning a numerical value of 1 Cfsu to each instance of a data movement (entry, exit, read or write). The results of this sub step in COSMIC-FFP are interpreted in the following way: once the data movements are identified, a 'measurement scale' is used and it is defined as '1 data movement of whichever type'. The measurer assigns to the data movement being measured a measurement unit of 1 with respect to that "etalon" and then assign to it a symbol of '1 Cfsu' (Cosmic Functional Size Unit). Therefore, those results are taken as numbers that are counted.

Finally, the last step **MSP3**, the results of assigning 1 to each data movement are added together using formula 1, taking for granted that the results of **MSP2** can be ratio numbers to be added.

This measurement of a functional process is closely similar to a classic measurement exercise: a measurement scale of '1 data movement' is used and this 'read' on the measurement scale is the equivalent of the marks (each mark being 1 data movement = 1 Cfsu). The size is then figured out in terms of the number of marks – or units read on the scale.

In conclusion, the COSMIC-FFP measure can be considered at least on the ratio scale. Moreover, the zero is meaningful, which means that when size = 0, a software does not have a size according to the measurement unit of COSMIC-FFP

6 Identification of FC scale, unit, and scale type

As introduced previously, formula 3 quantifies the amount of information interchanged in a given scenario. That is done through the following steps:

FC1: Calculating f_i for each event in the given scenario.

FC2: Calculating *NE* for the given scenario.

FC3: Calculating f_i/NE for each event in the scenario.

FC4: Calculating $\log_2$ of **FC3** for each event.

FC5: Multiplying **FC3** with **FC4.**

FC6: Adding up **FC5** for all events.

FC7: Multiplying **FC6** with -1.

FC1 is simply counting the frequency of the events' occurrences (that is identifying an event occurrence, and then adding all of those occurrences identified = frequency). Therefore we may suggest that it's at least on the ratio scale that depends on counting the frequency of events and as a result, the unit will be 'event's occurrence'.

FC2 is adding the total number of events' occurrences in a scenario, and it's also suggested to be defined at least on the ratio scale. The unit is 'event's occurrence'.

FC3 is a derived measure dividing **FC1** (ratio scale) by **FC2** (ratio scale), whose scale type will be the weakest one according to [5]. Therefore it can be on the ratio scale. Division is done through the unified unit "event's occurrence" and according to the analysis that has been proposed in [20], a ratio of quantities with the same dimensions is itself dimensionless. Therefore, the end result is therefore a dimensionless number that is a %. It is to be noted that **FC3** is the probability of the i-th event to be happened in the scenario.

FC4 is applying the logarithmic function to **FC3**. The ratio value of the logarithmic function $\log_2 n$ is exactly the number of binary digits (bits) required to represent the probability *n* of the event's occurrences. For instance, the combinations of a three-digit binary number can represent n=8. Thus, this step transforms the dimensionless probability value into the number of digits required to represent it in "bits".

In **FC5**, the representation size for probability in bits is multiplied by the probability of occurrences of the same event type. Each bit is a designator of the probability of one event's occurrence. The result is the total number of bits required for representing the probability of all occurrences of one event in the sequence. Therefore, such a multiplication would normally produce a number

with "bit" as a measurement unit. The scale type is suggested to be at least on the ratio scale.

In **FC6,** the representational size for the probability of all occurrences of all events is calculated.

The resulting number in FC6 is negative because of the logarithmic function's nature (it's negative on values less than one), but the amount of information shall be non-negative. In **FC7**, the multiplication by -1 is required to obtain the non-negative value for the amount of information. It is a simple transformation that doesn't change the scale type since -1 does not have a unit itself.

In conclusion, the FC measure is quantifying the representational size of the probabilities of all events' occurrences in bits, and can be considered at least on the ratio scale. Also, the absolute zero is meaningful since (theoretically) one scenario may have zero functionality thus requiring 0 amount of bits for its representation.

7 Discussion and Next Steps

In conclusion, it was seen how the "scale" concept is used in the COSMIC-FFP method to ensure meaningfulness of the numbers obtained from its measurement process. We also define the measurement unit for FC measure.

Whenever you do a 'measurement' in day-to-day life, you need a "scale". For instance, if you want to measure distance, you need a "measuring tape", then you map what you want to measure to the concept of "distance", then you carry out your measurement with a measurement procedure. The Mapping phase in COSMIC-FFP and **MSP1** (identifying the data movements) are our measurement tape in order to map the set of FUR into a measurement scale. That is exactly F, which maps the empirical structure E (FUR set in our case), into the numerical structure N (size in Cfsu unit). Therefore, the number we get as a result of applying **MSP2** and **MSP3** is the functional size for the corresponding set of FUR. We can then say that there is no loss of measurement information from **MSP2** to **MSP3** since both have a least a ratio scale type, as we have analyzed before in section 5. Further work is required to investigate whether or not it satisfies all the properties of an 'absolute scale'.

Entropy based functional complexity measure FC has no change of scale types through its steps. This could be interpreted as follows: FC transforms the measurement of the functional complexity of a scenario, based initially on the frequencies for each event, into quantification for the amount of information interchanged in a given interaction. By such study, we also conclude that the formula 3 has a measurement unit, which is "bit".

In this paper, even though some insights have been gained in the identification and analysis of the scale for the COSMIC-FFP measurement method, further analysis might be required to ensure that all metrology related issues in this measurement method have been adequately identified and analyzed.

Among some of the next steps is the investigation on how in some other software measures scales embedded within the definition of these measures are tackled and described. For example, how has the concept of "scale" been used in the McCabe Cyclometic complexity measure and other object oriented measures. The metrological concept of etalon presented in the introduction should also be investigated for both of the measurement methods discussed in this paper.

References

1. Abran, A., Desharnais, Jm., Oligny, S., St-Pierre, D. and Symons, C., *Measurement Manual COSMIC Full Function Points 2.2*, *The COSMIC Implementation Guide for ISO/IEC 19761*, École de technologie supérieure, Université du Québec, Montréal, Canada, 2003 - www.gelog.etsmtl.ca/cosmic-ffp.

2. ISO, *ISO/IEC 19761:2003 Software Engineering - COSMIC-FFP - A functional size measurement method*, in International Organization for Standardization - ISO, Geneva, 2003.

3. Alagar, V.S., Ormandjieva, O. and Zheng, M., *Managing Complexity in Real-Time Reactive Systems*, Sixth IEEE International Conference on Engineering of Complex Computer Systems, Tokyo, Japan, 2000.

4. Zuse, H. *Software Complexity Measures and Methods*, Walter de Gruyter, Berlin New York, 1991.

5. Fenton, N. and Pfleege, S.L. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing, 1998.

6. Whitmire, S.A. *Object Oriented Design Measurement*, John Wiley & Sons, 1997.

7. ISO, *International Vocabulary of Basic and General Terms in Metrology (VIM)*, International Organization for Standardization - ISO, Geneva, 1993.

8. ISO, *ISO/IEC IS 15939:2002 Software Engineering - Software Measurement Process*, International Organization for Standardization - ISO, Geneva, 2002.

9. ISO, *ISO/IEC 14143-1:1998 Functional size measurement - Definitions of concepts*, in, International Organization for Standardization - ISO, Geneva, 1998.

10. Abran, A. and Robillard, P.N., *Function Points: A study of Their Measurement Processes and Scale Transformations*, in Journal of Systems and Software, vol. 25, no. 2, 1994, pp. 171-184.

11. Khoshgoftaar, T. and Allen, E.B. *Applications of information theory to software engineering measurement*, Software Quality Journal, *3* (2). 79-103.

12. Ormandjieva, O., *Deriving New Measurement for Real Time Reactive Systems*, Department of Computer Science & Software Engineering, Concordia University,

Formatted: Font: Italic

Formatted: Left

Formatted: Font: Italic

Formatted: Font: Italic

Deleted: 8

Montreal, Canada, 2002.

13. Shannon, C.E., Weaver and Warren, *The Mathematical Theory of Communication*. the University of Illinois Press, Urbana, Chicago, 1969.

14. Abran, A., Ormandjieva, O. and Abu Talib, M., *A Functional Size and Information Theory-Based Functional Complexity Measures: Exploratory study of related concepts using COSMIC-FFP measurement method as a case study,* 14th International Workshop of Software Measurement (IWSM-MetriKon 2004), Konigs Wusterhausen, Germany, 2004, Shaker-Verlag, pp. 457-471.

15. Harrison, W., *An Entropy-Based Measure of Software Complexity*, IEEE Transactions on Software Engineering, *18* (11). November 1992

16. Davis, J.S. and Leblanc, R.J. *A Study of the Applicability of Complexity Measures*, IEEE Transactions on Software Engineering, *14* (9). September 1988 , pp. 1366-1372.

17. Alagar, V.S., Ormandjieva, O., and Liu, S.H., *Scenario-Based Performance Modelling and Validation in Real-Time Reactive Systems*. In Proceedings of Software Measurement European Forum 2004 (SMEF2004), Rome, Italy, 2004.

18. Shannon, Claude E., Weaver and Warren The Mathematical Theory of Communication. the University of Illinois Press, Urbana, Chicago, 1969.

19. A. Abran and A. Sellami, *Initial Modeling of the Measurement Concepts in the ISO Vocabulary of Terms in Metrology*, in Software Measurement and Estimation - Proceedings of the 12th International Workshop on Software Measurement - IWSM 2002, Magdeburg (Germany), Shaker-Verlag, Aachen, 2002, ISBN 3-8322-0765-1, pp. 315.

20. Al Qutaish, R. and Abran, A., *An Analysis of the Designs and the Definitions of the Halstead's Metrics*, 15th International Workshop of Software Measurement (IWSM-2005), Shaker-Verlag, Montreal, 2005.