

Functional details visualization and classification in the COSMIC FSM framework

Luca Santillo (luca.santillo@gmail.com)

Abstract

Given the relevance of software functional size measurement to the industry and in practice, improved ways to represent and handle the measurement details can provide significant advantages and make possible to discover new interpretation and exploitation possibilities over measurement results. This work illustrates some enhancement proposal in the COSMIC framework, to improve the significance of the measures and to add such exploitation possibilities, by means of a so-called “data movement diagram” and a further extension, to include data manipulation classification for functional processes within the software, denoted “functional processing diagram”. Possible advantages of the proposal would be to allow process-to-process comparison and profiling, complexity and reuse evaluation, and visual impact analysis of software changes required by enhancement projects.

Introduction

For software developers, the ability to measure a size of software from its functional requirements or specifications early in the life of a project is a first key step for estimating development effort. Further, as the functional size measure is independent of the technology used, it provides a key component for software project performance measures such as productivity.

COSMIC (the COmmon Software Metrics Consortium) was founded in late 1998 as a voluntary association of software measurement experts from Australia, Canada, Finland, Germany, Ireland, Italy, Japan, the Netherlands and the UK. COSMIC’s aim is to ‘to develop, test, bring to market and facilitate acceptance of new software sizing methods to support estimation and performance measurement’. The current COSMIC FSM (Functional Size Measurement) method is the COSMIC-FFP (Full Function Point) method, version 2.2. This method of sizing the functional requirements of software has been approved as an International Standard in early 2003 (ISO/IEC 19761:2003) [2].

The idea of being able to measure the size of software requirements or specifications, independent of the technology used to build the software, was first proposed by Allan Abrecht of IBM in 1979. His method, known as ‘Function Point Analysis’ (or ‘FPA’) has evolved into the ‘IFPUG’ method, supported by the International Function Point User Group, and other ‘first generation’ variants, as the NESMA method from the Netherlands and the MkII FPA method from the UK. Those other methods have also been approved by ISO, but – since they were all designed years ago to work for business application software, or ‘MIS’ – they cannot handle adequately recent software domains and application types (their domain of applicability is limited).

The ‘second generation’ COSMIC-FFP method was specifically designed from the outset to measure the functional size of real-time, multi-layered software such as used in telecoms, process control, infrastructure software, and operating systems, as well as business application software, all on the same measurement scale. Having been developed in the last few years, the method is compatible with modern specification methods such as UML, and with OO techniques.

COSMIC FSM framework

The COSMIC-FFP measurement method involves applying a set of models, rules and procedures to a given piece of software as it is perceived from the perspective of its Functional User Requirements (FUR’s) [2]. The result of the application of these models, rules and procedures is a numerical “value of a quantity” representing the functional size of the software, expressed by CFSU (COSMIC Functional Size Units), or shortly ‘CU’.

Two phases are involved during the measurement process: the mapping phase and the nominal measurement phase. The mapping phase takes as input a FUR statement and produces a specific software model, by identifying:

- the Purpose, Scope and Measurement Viewpoint,
- the software Layer(s) under study,
- the software Boundaries,
- the Functional Processes (FP's),
- the Objects of Interests (OoI's, or, equivalently, the Data Groups),
- eventually, the Data Attributes.

The measurement phase takes as input the instance of software model and, using a defined set of rules and procedures, produces a value of a quantity the magnitude of which is directly proportional to the functional size of the model, based on the current COSMIC-FFP measurement principle: 'The functional size of software is directly proportional to the number of its Data Movements'. By convention, this numerical value of a quantity is then extended to represent the functional size of the software itself – each Data Movement (DM) is assigned 1 CU.

It is significant to identify and aggregate the Data Movements by FP, which is defined as 'an elementary component of a set of FUR's, comprising a unique cohesive and independently executable set of data movement types [...]'. Data Movements are sub-processes and comes in four classes: Entry (E), eXit (X), Read (R), and Write (W). Therefore, a suggested way to document measures is reported in the COSMIC-FFP Measurement Manual, here reproduced (Fig. 1).

		DATA GROUPS									
		Data Group 1					Data Group n	ENTRY (E)	EXIT (X)	READ (R)	WRITE (W)
LAYERS	FUNCTIONAL PROCESSES	:	:	:	:	:	:				
LAYER "A"		Functional Process a									
	Functional Process b										
	Functional Process c										
	Functional Process d										
	Functional Process e										
		TOTAL - Layer "A"									
LAYER "B"		Functional Process f									
	Functional Process g										
	Functional Process h										
		TOTAL - Layer "B"									

Figure 1. COSMIC-FFP Generic Software Model Matrix. Mapping Phase: each identified data group is registered in a column; each functional process is registered on a specific line, grouped by identified layer. Measurement Phase: for each identified functional process, the identified data movements are noted in the corresponding cell ('E', 'X', 'R', 'W'); for each identified functional process, the data movements are then summed up by type and each total is registered in the appropriate column at the far right of the matrix; the measurement summary can then be calculated and registered in the boxed cells of each layer, on the "TOTAL" line.

Remarks

- Data Groups (or equivalently, OoI's) are not assigned any value in the current measurement method version – still, they must be identified in order to count data movements correctly. The materialization of a data group may take different forms, from a data storage persistent file to a presentation cluster on screen or printed report, usually with transient persistence.
- In the current version of the method, data manipulation sub-processes are not recognized separately, but are considered to be associated with or part of specific data movement sub-processes. Given this approximation, the standard COSMIC-FFP method is suitable for sizing 'movement-rich' types of software, but it can be easily extended – as proposed next in this work – for sizing 'manipulation-rich' (or 'algorithm-rich') software (Fig. 2).

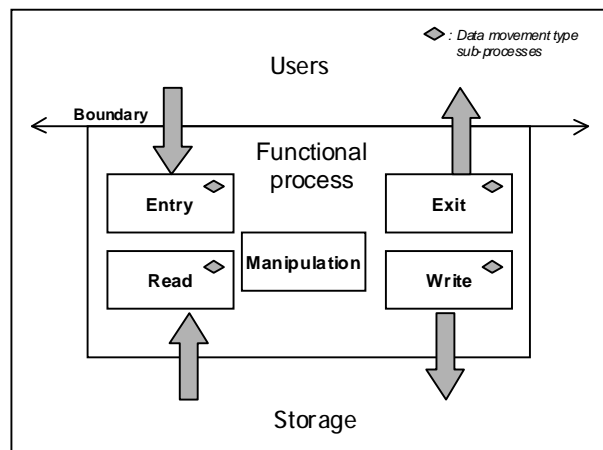


Figure 2. The sub-process types and some of their relationships.

Representing the Measurement Details

A new visualization diagram (“data movement diagram”, or DMD) is proposed, as an adaptation of the so-called Hinton diagram (weight matrix, usually used in the neural networks research field to represent the strength of the links between nodes of the network) [4]. The data movement diagram provides information about which functional process uses (entries, exits, read and/or writes) which object of interest, or data group, in the measured system (Fig. 3). In such diagram, the box filling design denotes ‘impact/use’, while the box size denotes “strength” (i.e. quantity of the same sub-process type over a single object of interest inside the given functional process scope). Note that in some case more of one instance of the same data movement type is permitted and measured within the same functional process over the same data group (e.g. FP “c” over OoI 1).

Specific features are provided in case of enhancement projects, where data movements may be added to, changed, or deleted from a functional process (Fig. 4).

Such DM diagram can be seen as a whole system (layer) profile, or can be examined by perspective:

- by functional process (by row), or
- by object of interest (by column).

Data Movement Diagrams can be seen as a sort of measurement profiles to facilitate the visualization of processing operations *density*, or of *impact of enhancement* over an existing system. Within the scope of a single measure, functional processes and objects of interest can be visually compared to identify *similarities* between them and used as a sort of check list during the first steps of software analysis (typically, during the requirements analysis).

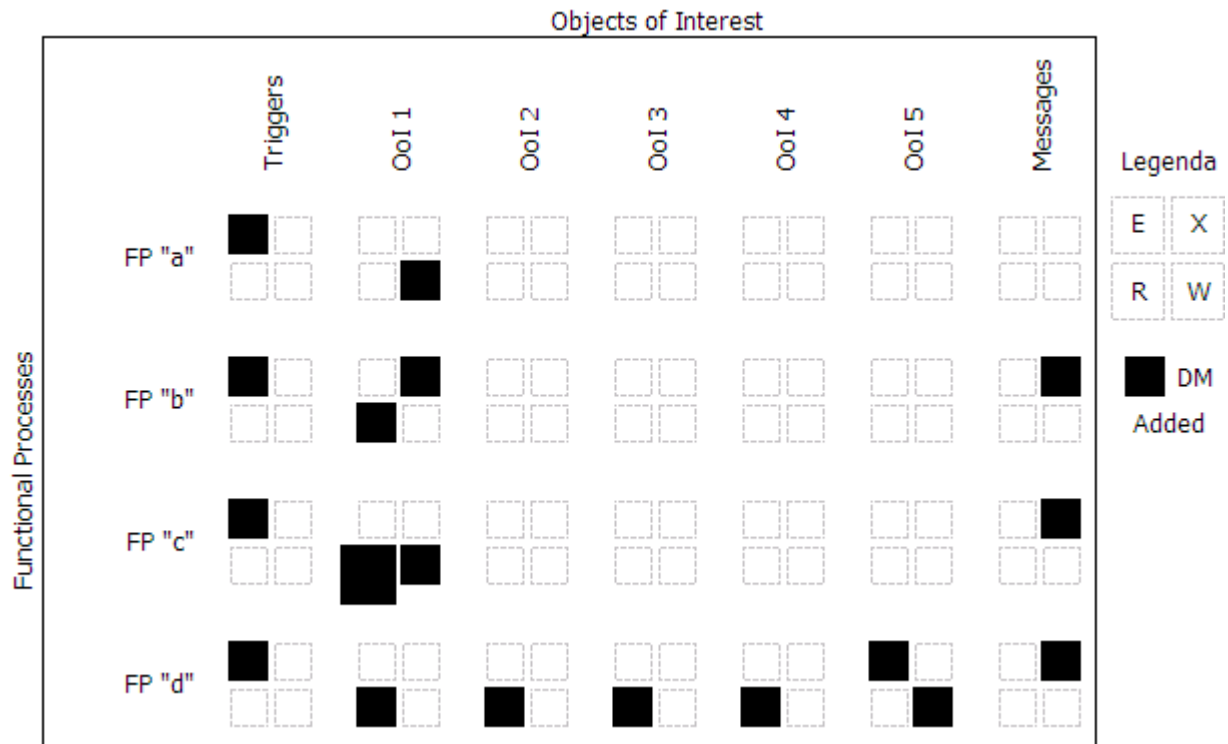


Figure 3. DMD example (baseline or development). FP "a" simply writes data on OoI, given a specific triggering event (the minimum size of any functional process is 2 CU). FP "b" represent an inquiry, where some data are read and shown, or an error message is provided if the required data are not found. The last row, FP "d", represents the case when all the data groups in the system can eventually require to be checked before one of them can be written. Reading the diagram by column, OoI 1 is a case when an object of interest (a data group) is eventually used by every and each functional process in the system, being critical for the system operation.

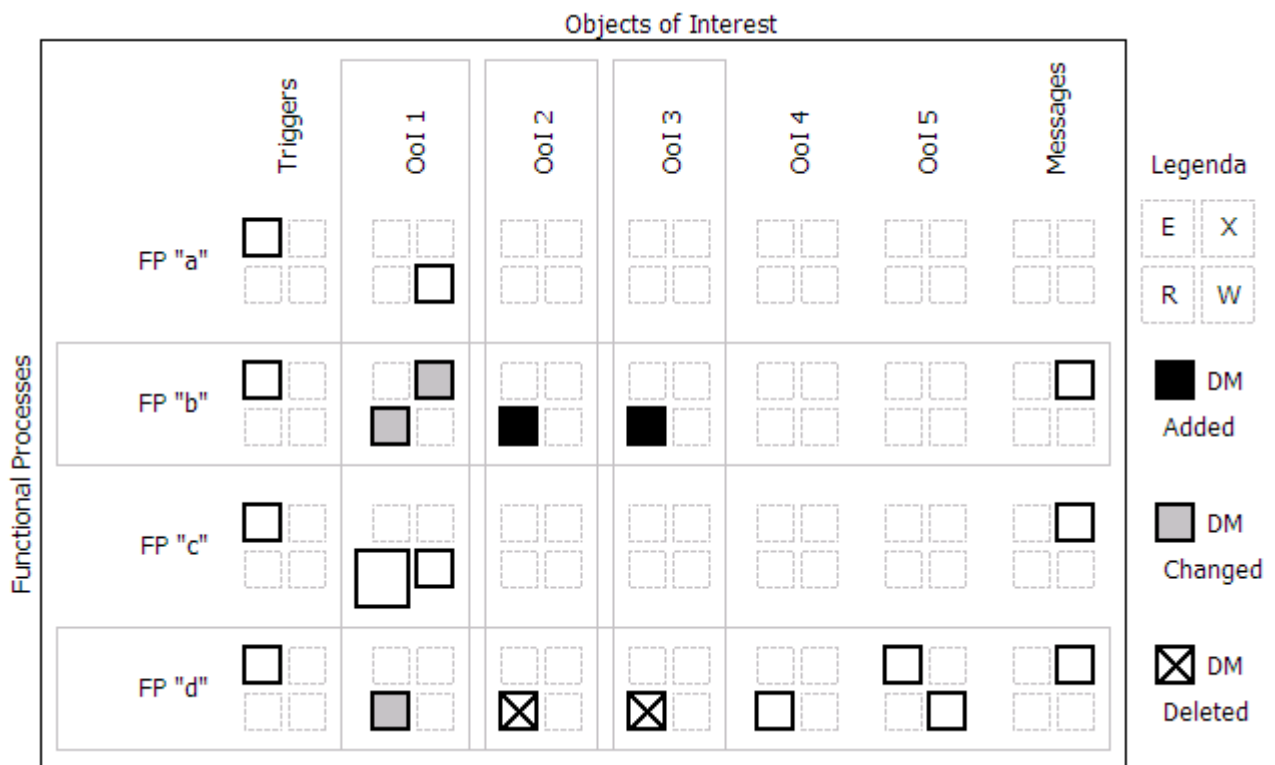


Figure 4. DMD example (enhancement with respect to previous example, Fig. 3). FP's "a" and "c" are unchanged; triggers, messages, and OoI's 4 and 5 are not impacted. Remaining processes are modified, due to added, changed or deleted data movements over one or more OoI.

Extending the Measurement Framework and Details

A possibly exhaustive list of 13 possible action types forming the processing logic of any functional process can be found in [3]; such list has already been proposed within a complexity evaluation framework in [5]. Here, we can classify any action with respect to its coverage in the COSMIC-FFP method as a data movement; if so, the action is excluded from further measurement (Tab. 1). The remaining actions form a subset of manipulation classes, which can be used to extend the standard COSMIC-FFP method, and, within the scope of the current work, to enrich the visualization diagram for classification and comparison purposes over measurement results.

	Action	Measured as Movement	Include as Manipulation
1	Validations are performed		Yes (Validation)
2	Mathematical formulas and calculations are performed		Yes (Creation)
3	Equivalent values are converted		Yes (Validation)
4	Data is filtered and selected by using specified criteria to compare multiple sets of data		Yes (Validation)
5	Conditions are analyzed to determine which are applicable		Yes (Validation)
6	One or more data groups are updated	Yes (W)	
7	One or more data groups are referenced	Yes (R)	
8	Data or control information is retrieved	Yes (R)	
9	Derived data is created by transforming existing data to create additional data		Yes (Creation)
10	Behaviour of the system is altered		Yes (Creation)
11	Prepare and present information outside the boundary	Yes (X)	
12	Capability exists to accept data or control information that enters the application boundary	Yes (E)	
13	Data is resorted or rearranged		Yes (Creation)

Table 1. Action list for data manipulation and movement classification.

However, since the action list should be further reviewed for lacks of coverage or functional manipulation overlapping, and for sake of simplicity and readability, only two main classes of data manipulation are considered in the further version of visualization: “validation” (“check”) versus “creation”. A simple criterion to differentiate those classes could be that whenever an action provides the user with a data content or structure, as requested by the user, which could not be simply retrieved and displayed starting from the data base, we identify some kind of data creation. The previously called “data movement diagram” (DMD) becomes a “functional processing diagram” (FPD), with additional representation of such data manipulation classes (Fig. 5).

Also for validation and creation data manipulation types, a convention can be stated so that the box size represents a sort of count of separate actions, if defined (see Fig. 5, case of FP “c” over OoI 1). Such case would require a further extension of the measurement method, to clarify how to distinguish separate manipulation sub-processes within a single functional process; such aim is beyond the scope of the current proposal. Further variations of the diagram conventions could also introduce a way to differentiate validation or creation that occurred associated with a specific data movement within the process (e.g. validation of the entered data versus validation of data read from a data group, data creation in order to update a data group versus data creation to allow a given exit to be performed, and so on).

Similarly, Fig. 6 shows an example of functional processing diagram in case of changes occurred because of a given enhancement project. Note that functional processing diagrams could be correctly express changes that occurred in the data manipulation sub-processing of a process, while the data movement sub-processes might be not affected at all by the required change.

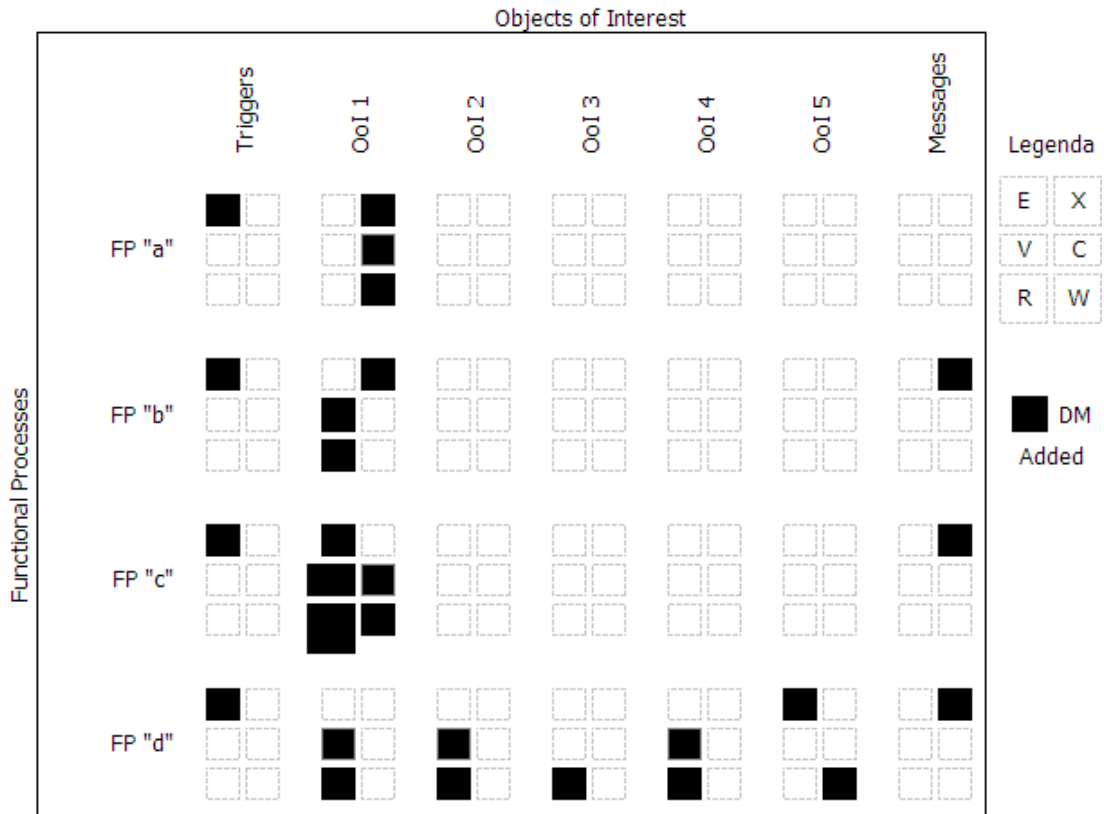


Figure 5. FPD example (baseline or new development). Here, “DM” stands for both Data Movement and/or Data Manipulation types.

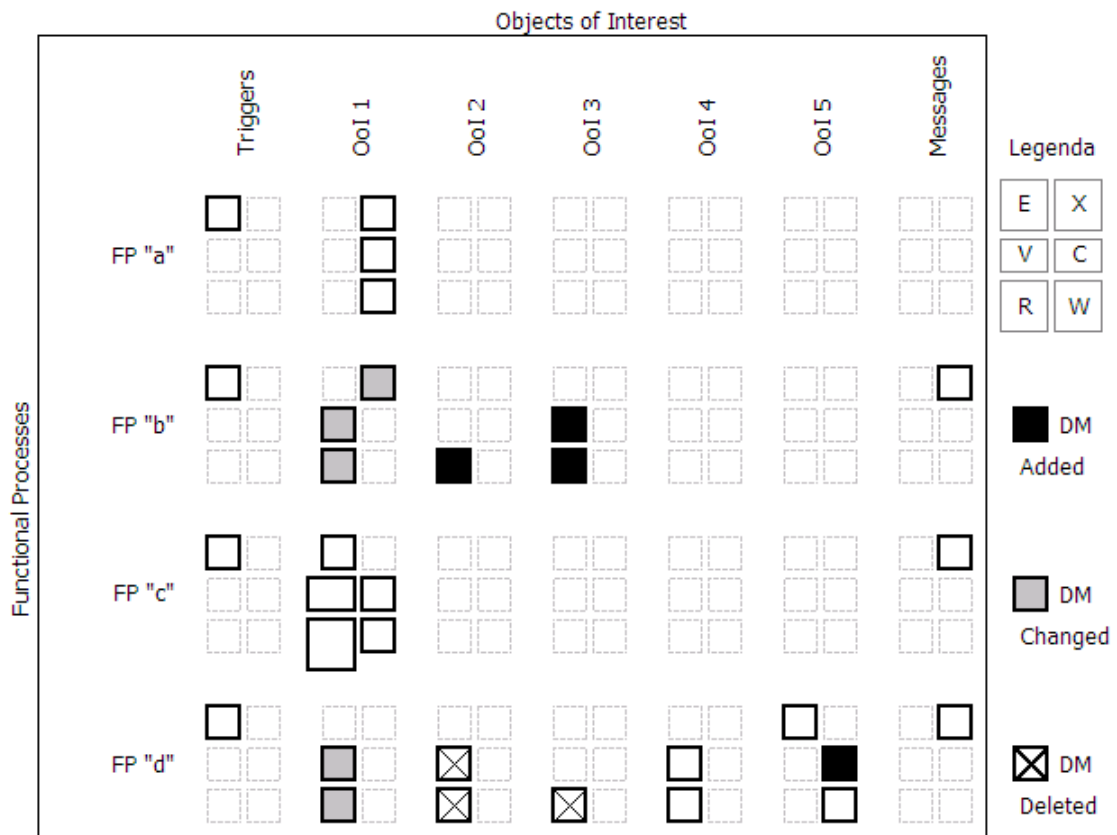


Figure 6. FPD example (enhancement with respect to previous example, Fig. 5). Once again, “DM” stands for both Data Movement and/or Data Manipulation types.

Application Example and Remarks

A first, trivial case study is offered by the simplest set of functional processes for a single data group management, the CRUDL (Create, Read, Update, Delete, and List). Regardless of the specific data group contents, we can easily argue that the base functional processing profile would be as shown in Fig. 7, case (a). Variations are possible, since in real world CRUDL's, the FUR's could require some external data update when inserting or deleting an occurrence in the data group being managed (Fig. 7, case (a1)), or more validation or creation sub-processing than in the standard case (Fig. 7, case (a2)).

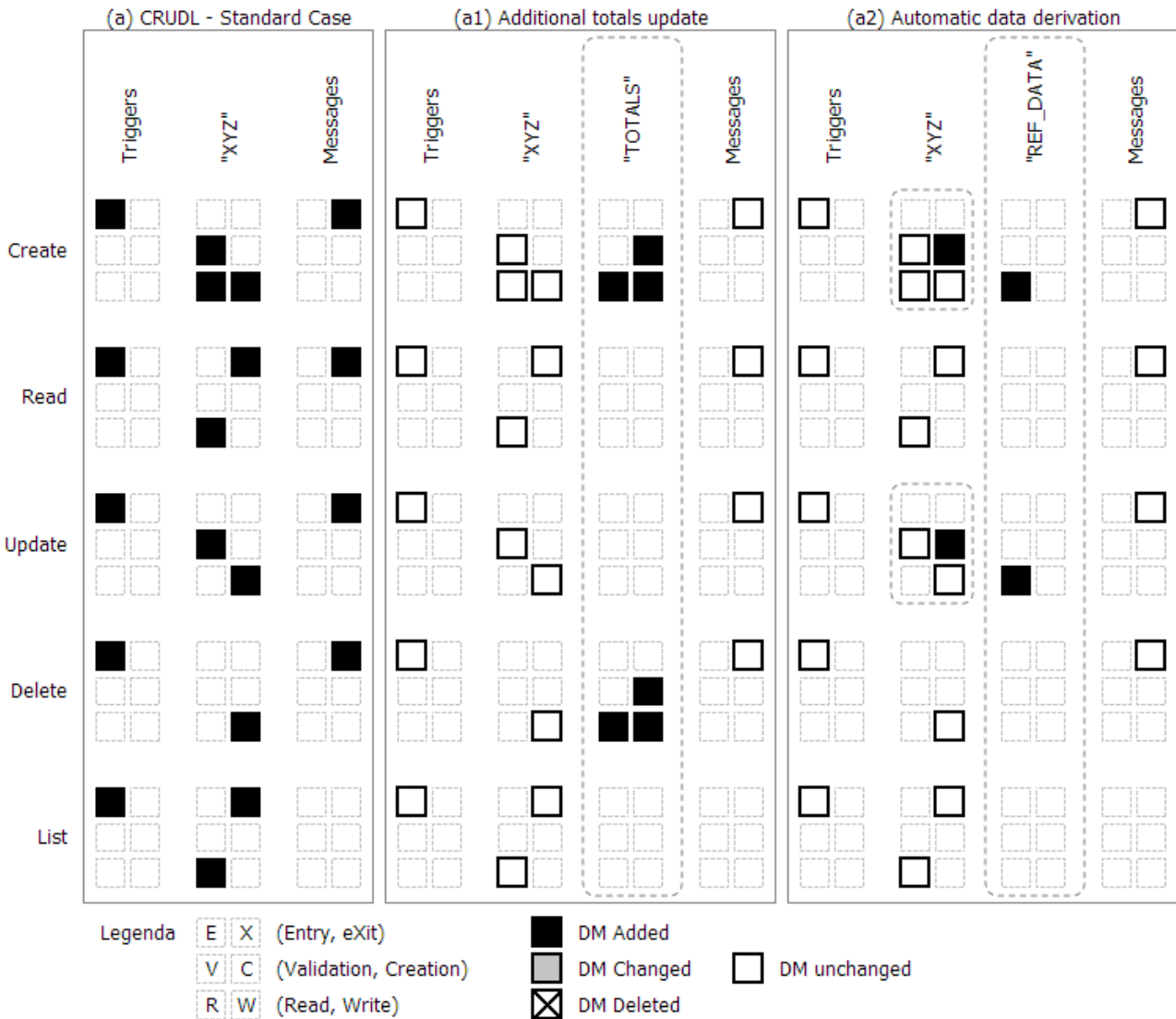


Figure 7. FPD's for three variations of the CRUDL case. "DM" stands for both Data Movement and/or Data Manipulation types. 2nd and 3rd variations are represented as if enhancements with respect to 1st case (a). Dotted frames highlight specific differences between the three cases. In interactive reporting, the three cases could be shown as overlapping one to each other, highlighting the differences:

(a) standard CRUDL case (in the Create FP it is assumed that "XYZ" is first read to check whether the user provided key for the new occurrence is not present in the data group);

(a1) whenever an occurrence is inserted or deleted from XYZ, the FUR's require that a generic "TOTALS" data group is maintained for further user reference;

(a2) when inserting or updating some attributes are automatically derived by the system, based on some corresponding attribute values read from a distinct OoI "REF_DATA".

We note that the proposed extension (and representation) for data manipulation sub-processing would then clearly distinguish variations in the given cases, which would be otherwise measured and reviewed as identical. Visually, such differences would clearly be expressed when comparing corresponding processes (e.g. the “Create” functional process in cases (a) and (a2) above, where the “REF_DATA” might be also not necessary for data to be created by the base process).

Remarks

Some remarks are reported, in order to plan potential future improvements of the proposed diagramming technique.

- Ÿ Some details of the proposed visualization diagrams should be fixed in further versions whenever accepted by the practitioners’ community and validated over a large set of real cases in any possible software domain. For instance, the examples shown above separately lists “fake” objects of interest for triggering events and messages. Since any functional process must have a triggering event, and such a triggering event is usually associated with the first (or only) part of data entry regarding a real object of interest (in online software types), it could be inferred that such Entry data movement, comprising the triggering event, should be assigned to and visualized over a specific object of interest, instead of the a generic one.
- Ÿ The COSMIC-FFP measurement principle has as a consequence the fact that “the smallest theoretical functional size for a piece of software is 2 CU [...], as the smallest functional process must have at least one Entry, and either one Exit or one Write”. This means that a DMD, or equivalently a FPD, for any functional process must show at least two “filled” boxes, for such data movements.
- Ÿ Again, according to the COSMIC-FFP measurement principle, “there is no upper limit to the functional size of a piece of software and, notably, there is no upper limit to the functional size of any of its functional processes”. This means that in theory there’s no limit to the size of a DMD, or equivalently of a FPD (although real cases would necessarily be limited in practice).
- Ÿ With respect to the previous remark, and for general reasons, automation of the measurement data collection, by means of tool support, and consequently of the visualization of the measurement details would be a great benefit for the practitioners.
- Ÿ Automation of measurement data collection and representation would also allow for general pattern recognition techniques to be applied to real world cases, with subsequent benefits in software assessment and estimation process.

Conclusions

This work introduced a new way to represent the measurement details and, collaterally, a proposal for extending the current framework to include data manipulation sub-processes type classification in the measurement process. Possible advantages of the proposal would be:

- Ÿ enhanced measurement data collection;
- Ÿ added *classification means* for software components;
- Ÿ added possibility of *type & structure comparisons* among software systems;
- Ÿ *visual traceability* for software development or enhancement requirements.

General reasons to take into account such enhanced visualization/classification tool would then be:

- to allow *complexity analysis* (from a functional perspective, by diagram comparison);
- to allow *reuse analysis* (from a functional perspective, by diagram overlapping);
- to improve cost estimation or value assessment based on effective impact of software size (by component) on project effort (by phase).

Regardless of specific details of the proposed visualization approach, which could be improved after ordinary “test & trial”, any visual representation tool could help in replacing (or in support to) hard-to-read reporting documents with an overall, instantaneous visual representation of sized systems and enhancement-impacted areas and components for such systems. Especially high level resources, who are not directly involved in measurement details, could appreciate the proposed visual perspective, for immediate perception of affected areas and components for decision making purposes or preliminary assessment of interesting software projects portions.

References

- [1] A. Abran, JM. Desharnais, S. Oigny, D. St-Pierre, C. Symons, “COSMIC-FFP Measurement Manual (COSMIC Implementation Guide for ISO/IEC 19761: 2003) - Version 2.2”, Common Software Measurement International Consortium, January 2003.
www.lrgl.uqam.ca/cosmic-ffp
- [2] COSMIC website: <http://www.cosmicon.com/>
- [3] IFPUG, “Function Point Counting Practices Manual, Release 4.2”, International Function Point Users Group, January 2004.
- [4] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., “Learning internal representations by error propagation”, in: Rumelhart, D. E. and McClelland, J. L., editors, “Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations”, MIT Press, Cambridge, MA, 1986.
- [5] L. Santillo, “Software Complexity Evaluation Based on Functional Size Components”, 14th Intl. Workshop on Software Measurement (IWSM), Berlin, 2004.