COSMIC Method Update Bulletin #11

Proposal to improve the COSMIC Method v3.0.1 definition of a Functional Process and related topics

1. The problems that lead to the proposed changes

The origin of this Method Update Bulletin ('MUB') was the discovery that when using the 'MkII FPA' method (another functional size measurement method) a weakness in its definition of a 'logical transaction' (the MkII method's equivalent of a functional process) had resulted in some measurement errors. It was realized that the same weakness existed in the COSMIC method's definition of a functional process.

Further, some providers of training and consulting services have reported occasional difficulties in understanding the definition of a 'functional process', resulting in mis-interpretation. The consequence was that the COSMIC Measurement Practices Committee started a very thorough review of the definition of a functional process and several closely-related related concepts and measurement rules. The review showed that in addition to the need to improve the definition and rules for a functional process, several other areas of improvement are needed:

- making clearer what we mean by 'Functional User Requirements'
- explaining the possible degrees of the relationships (i.e. 'cardinalities') between events, functional users, triggering Entries and functional processes and improving the definitions of these concepts
- an explanation of how to measure functional processes that share some functionality in common or that are very similar
- an explanation of the need to distinguish between the triggering event that starts a software system, and the triggering events that start the functional processes that the software system must execute.

The functional process is one of the most basic concepts of the COSMIC method, so its definition should be changed only if there is a real risk of mis-interpretation. It is unilkely that the proposed changes will have any effect on existing measurements unless the current definition has already been mis-interpreted. But we acknowledge that possibility and apologize for any problems this may cause to our method users.

The purpose of this MUB is to describe the changes needed and also why they are needed. Until this review, the MPC believed that the current definition of the method in version 3.0.1 was 100% logically consistent, even if it might not be perfectly clearly explained. The review has shown, however, that there are logical inconsistencies between certain existing definitions which we must correct. In the review we have found nothing wrong with the intent of the principles, rules and definitions that were originally published in the year 2000. Our challenge in this review has been to eliminate the inconsistencies and to improve the explanations and examples so that the method is easier to understand and less likely to be mis-interpreted.

(To help understand the impact of the proposed changes, we investigated the effect of the proposed definition changes on the current COSMIC Foundation-level certification exam. Very few of the 214 questions available for use in the exam, concerned with the wording of the definitions, actually need to be changed to align with the revised definitions.)

We assume the reader of this Method Update Bulletin has a full understanding of the COSMIC method.

2. The current definition of a ‘Functional Process’

The current definition of a Functional Process in section 3.2.1 of the Measurement Manual (written out here as bullet points for ease of reference) is as follows:

“A functional process:

a) is an elementary component of a set of Functional User Requirements,
b) comprising a unique, cohesive and independently executable set of data movements.
c) It is triggered by a data movement (an Entry) from a functional user that informs the piece of software that the functional user has identified a triggering event.
d) It is complete when it has executed all that is required to be done in response to the triggering event.

NOTE: In addition to informing the piece of software that the event has occurred, the ‘triggering Entry’ may include data about the object of interest associated with the event.”

3. Clarification of ‘Functional User Requirements’

Understanding clause a) of the definition of a functional process depends on understanding the meaning of ‘Functional User Requirements’ (or ‘FUR’). However, it has become evident that we have used the term ‘FUR’ in the Measurement Manual in two subtly different ways.

In the Measurement Manual, version 3.0.1, section 1.2, we discuss how to derive the FUR from the available software artifacts. It is the FUR that must be measured and all the definitions, principles and rules for the method assume a set of FUR that is perfectly structured for COSMIC measurement. In practice, however, very often the artifacts from which FUR are derived are incomplete or unclear in some way. Additionally, there may not be a clear one-to-one mapping from individually stated requirements or physical artifacts to functional processes. The Measurer must therefore usually make some assumptions to produce the FUR that will actually be measured.

Elsewhere in the COSMIC documentation, there are many places where we give examples starting with e.g. ‘Suppose the FUR ....’, or ‘It is possible that FUR exist ....’, or ‘If the FUR state explicitly....’. This usage may give the impression that ‘FUR’ means ‘the functional sub-set of the actual statement of requirements’. As noted above, this sub-set almost invariably has some omissions, ambiguities, etc.

To remove any uncertainty, we propose to add the following after the definition of ‘FUR’:

“In this Measurement Manual, we will restrict the term FUR to mean the functional user requirements that:

- are derived from the available software artifacts (requirements, designs, physical artifacts, etc),
- are adjusted, if necessary, by assumptions to overcome uncertainties in the available artifacts,
- contain all the information needed for a COSMIC FSM.”

For the phrases quoted above that are used to introduce examples, we will use ‘actual requirements’ or ‘physical artifacts’, etc as appropriate to the example.

4. Reasons for proposed changes to the current definition of a ‘Functional Process’

The following are reasons to propose improvements to the current definition and related rules and text.

- A functional process is not, strictly speaking, a ‘component’ of the FUR, but is a *representation* of a part of the FUR of the software to be measured.
- In clause a), ‘component’ should be avoided because component tends to be used in relation to software; FUR are <u>requirements</u> for software.
- In clause a) it should be made explicit that the FUR in this definition are only those for the software being measured, which has already been defined by its scope, etc.
- In clause b), ‘cohesive’ should be deleted because it does not add any usable meaning to the definition.
- In clause b), the meaning of ‘independently executable’ is unclear. Two functional processes that have some common functionality may, when implemented and executing, use some common code. Also, it may be a system requirement that one functional process (-type) cannot be executed unless preceded by another functional process (-type). These two examples suggest there are circumstances where functional processes are not actually ‘independently executable’. In fact, the definition should convey that each functional process can be *defined* independently of any other functional process in the software being measured.
- Clause c) depends on a functional user identifying (or ‘sensing’) a triggering event. But a functional user may also be the *source of* a triggering event. (For example, a human functional user is the source of the triggering event when the user decides to make an enquiry. And when a clock sends a signal to start a real-time functional process, the clock is both the source of the event and the functional user that initiates the triggering Entry.). Actually, what matters for the definition is not *how* a functional user relates to the triggering event but that an event *causes* a functional user to initiate a triggering Entry.
- In Clause d), the phrase ‘all that is required to be done in response to the triggering event’ is too open-ended. It could be mis-interpreted to cover all the functional processes in different software systems that may arise from a single event. The definition should refer to all that is required to be done in response to the ‘triggering Entry’, not to the ‘triggering event’. (This problem was the original main reason for our deciding to re-examine the functional process definition.) The term ‘triggering Entry’ should be defined in the Glossary.
- The COSMIC method has not been clear on whether a single functional process may be partly processed on-line and partly processed in batch mode. (It may not.)

5. Proposed changes to section 3.2.1 of the Measurement Manual, version 3.0.1

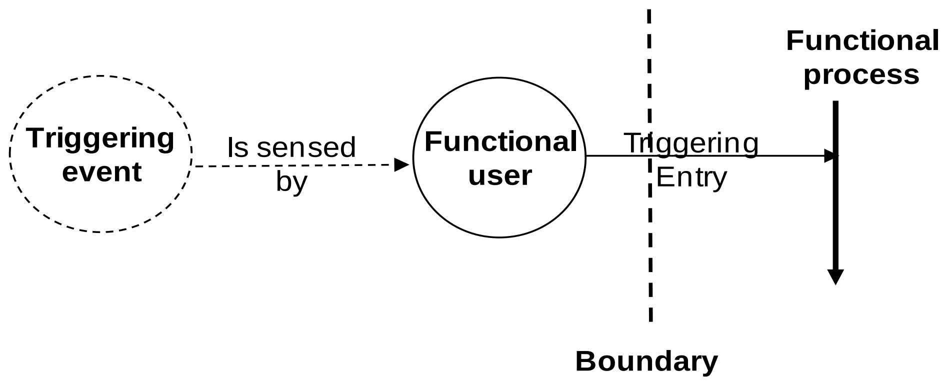
Change 5.1: The proposed revised definition of a functional process is as follows.

a) A functional process is a set of data movements representing an elementary part of the Functional User Requirements (FUR) for the software being measured, that is unique within that FUR and that can be defined independently of any other functional process in that FUR.
b) Each functional process starts processing on receipt of a data group moved by the triggering Entry data movement of the functional process.
c) The set of all data movements of a functional process is the set that is needed to meet its FUR for all the possible responses to its triggering Entry.

NOTE 1: When implemented, it is an *occurrence* of a functional process that starts *executing* on receipt of an *occurrence* of a data group moved by an *occurrence* of a triggering Entry.

NOTE 2: The FUR for a functional process may require one or more other Entries in addition to the triggering Entry.

Change 5.2: The existing Figure 3.2.1 below shows that an event is 'sensed by' a functional user which then triggers a functional process.



Two changes are needed for this Figure 3.2.1. First, as noted above, it is more generally useful to replace 'is sensed by' with 'causes'. Second, the diagram could be misleading if it is interpreted as meaning that a functional user generates the triggering Entry that triggers a functional process. This interpretation would be incorrect; any Entry is part of a functional process so cannot be 'generated' by a functional user. To avoid this potential mis-interpretation, Figure 3.2.1 and the related text will be modified as follows.

"The relationship between a triggering event, the functional user and the Entry data movement that triggers a functional process being measured is depicted in Figure 3.2.1 below. The interpretation of this diagram is: *a triggering event causes a functional user to generate a data group that is moved by the triggering Entry of a functional process to start the functional process.*

Note: for ease of reading, we often omit the reference to the data group and state that a functional user *initiates* a triggering Entry that starts a functional process or even more simply that a functional user *initiates* a functional process.

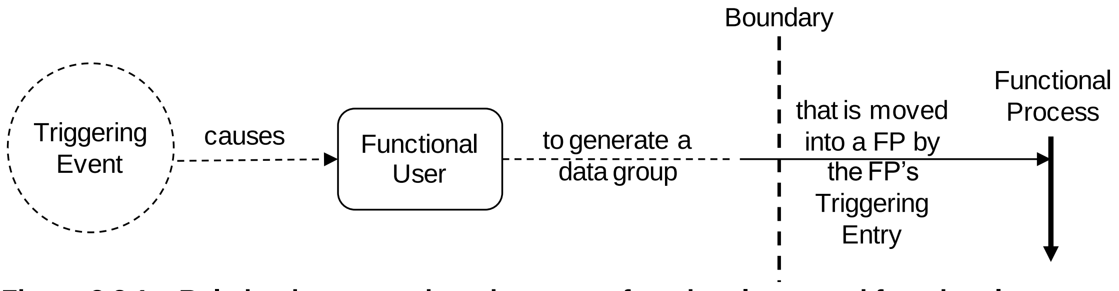


Figure 3.2.1 – Relation between triggering event, functional user and functional process

As a result of an in-depth study by the Measurement Practices Committee of the possible relationships that may occur along this chain, the following text will also be added.

"All the relationships between the concepts in Fig. 3.2.1 (the triggering event / functional user / triggering Entry / functional process) may be one-to-many, many-to one, or many-to-many, with one exception. (The exception is that the data group moved by any one triggering Entry may initiate only one functional process – see Rule b) for a functional process in section 6 of this Method Update Bulletin). Some examples of possible cardinalities:

- A triggering event may be sensed by many functional users, e.g. an earthquake is detected by many sensors
- Some types of functional user can detect many types of events, e.g. human operators of a business application who initiate a triggering Entry for each event
- A hardware functional user may initiate more than one triggering Entry on sensing one triggering event in certain types of safety-critical software
- Some functional processes may be initiated by different functional users, e.g. a software component that can be called by many of its software functional users

In practice, the cardinalities along all the chains for the software being measured (i.e. that describe the specific events that cause specific functional users to initiate specific functional processes) will be constrained by the software FUR. For a fuller discussion of the cardinalities along the chain of Fig. 3.2.1 and for more examples, see the Appendix to this MUB." (The Appendix to this MUB will become an Appendix to the Measurement Manual.)

Change 5.3: The existing definition of a triggering event needs to be modified in light of reconsidering the possible cardinalities along the chain of Fig. 3.2.1. Also this definition includes the definition of an 'event' embedded in its text. It is preferable to separate these two definitions, as follows.

Event

Something that happens.

Triggering Event

An event, recognised in the Functional User Requirements of the software being measured, that causes one or more functional users of the software to generate one or more data groups, each of which will subsequently be moved by a triggering Entry. A triggering event cannot be sub-divided and has either happened or not happened.

Change 5.4: The changes to the definition of a functional process and to the important Figure 3.2.1 mean we should now add a definition for a 'triggering Entry'.

The text following the existing Figure 3.2.1 in the Measurement Manual v3.0.1 states:
"The triggering Entry is normally a positive, unambiguous message that informs the software that the functional user has identified a triggering event. The triggering Entry often also includes data about an object of interest associated with the event."

This statement is unsatisfactory in two ways:

- Re the first sentence of this text: we now realise that there are cases (see the examples below) where the software is not required to be informed that the functional user has 'identified an event'. The FUR may require that the functional user provides the software with other data associated with the event, but not the event *per se.*
- Re the second sentence of this text: an event can be regarded as an object of interest. Therefore, the second sentence cannot be true, since a triggering Entry can move only one data group describing one object of interest.

We therefore now define a triggering Entry as follows:

"The Entry data movement of a functional process that moves a data group, generated by a functional user, that the functional process needs to start processing.

NOTE The FUR for a functional process may require one or more other Entries in addition to the triggering Entry."

To illustrate this definition, we will add the following:

"The data group(s) that a functional user will generate as a result of a triggering event depend on the FUR of the functional process that will process the data, as illustrated by the following examples:

a) *A functional process of a real-time software system may be started by its triggering Entry informing the functional process that a clock (functional user) has ticked. The data group moved conveys data (the tick, perhaps via a single bit) that informs only that an event has occurred.*
b) *A functional process of an industrial real-time fire detection software system may be started by a triggering Entry initiated by a specific smoke detector (functional user). The data group generated by the detector conveys the information 'smoke detected' (an event has occurred) and includes the detector ID (i.e. data about where the event occurred)'*
c) *A functional process of a personnel software system may be started by a triggering Entry that moves a data group describing a new employee. The data group is generated by a human functional user of the personnel software who enters the data. The object of interest of this data group is 'employee'. Normally, the event of a new employee starting work is not of interest per se. The entered data informs about the employee; it is not about the event of starting work. (However, the date of starting work will almost certainly be recorded as an attribute in the employee data group.)*
d) *A bar code reader (a functional user) at a supermarket checkout starts a scan when a bar code appears in its window (the triggering event). The reader generates a data group, comprising an image of the bar code. This data group is moved by a triggering Entry into its functional process that adds the product cost to the customer's bill if the code is valid. The object of interest of this group is the product whose bar code is scanned.*"

Change 5.5
A final consequence of the above is that Principle e) of the Generic Software Model must be changed. This Principle, currently states:

"Each functional process is triggered by an **Entry** data movement from a functional user which informs the functional process that the functional user has identified an **event.**

This statement has the same problem as already noted in 5.2 above (a triggering Entry cannot come 'from' a functional user; it is part of a functional process). The revised wording of this Principle will be based on the revised interpretation of Fig. 3.2.1:

"Each functional process is started by its **triggering Entry** data movement. The data group moved by the triggering Entry is generated by a functional user in response to a **triggering event**."

6. Proposed changes to section 3.2.2 of the Measurement Manual, version 3.0.1

Change 6.1: Improvements to the Rules for a functional process are proposed. The Rules have also been re-sequenced into a more logical order.

- The existing Rules a) and b) will be removed since they say nothing in addition to what the definition of a functional process states.
- The existing Rule c) is correct but does not cover all the possible relationships between a triggering event, the functional user, a triggering Entry and a functional process as shown in Figure 3.2.1 of the Measurement Manual. This topic will now be dealt with in section 3.2.1 (see Change 5.2 above). The new Rule b) now defines only the important relationship between a triggering Entry and a functional process: it replaces the existing Rule c)

- We add to the existing Rule d), now c), that there is no upper limit to the size of a functional process
- The existing Rule e) is moved to become Rule a), unchanged
- The existing Rule f), now d), is now considered misleading since there are circumstances where a functional process, when executing, can enter a wait state *during* its execution. (The existing rule d) refers to entering a wait-state *after* execution.) The rule has therefore been re-phrased; also it applies equally to business application as well as real-time software
- The existing Rule g) has a mistake, as also does a related Example 3 in section 3.2.5. This Rule must apply only to different *values* of data attributes in d*ifferen*t *occurrences* of the Entries that comprise the input; the Rule should not apply to different *numbers* of data attributes in an Entry. Furthermore, the content of this Rule now follows directly from the revised clause c) of the definition of a functional process (see Change 5.3 above). This Rule is therefore redundant and has been deleted. The topic has been moved to section 3.2.5 (see Change 8.1 below)
- The existing Rule h) is now better considered as describing examples and will therefore be removed from the Rules.

Note that in all these Rules, all mentions of triggering event, functional user, functional process, etc are to 'types' of these unless otherwise stated. All changed text is shown in bold, dark blue text, or with a strikethrough.

RULES – Functional process	
a)	A functional process shall belong entirely to the measurement scope of one piece of software in one, and only one, layer.
b)	**Any one triggering Entry to a piece of software being measured may initiate only one functional process in that software.**
c)	A functional process shall comprise at least two data movements, an Entry plus either an Exit or a Write. **There is no upper limit to the number of data movements in a functional process**
d)	~~In the context of real-time software,~~ **An executing functional process shall be considered terminated when it has satisfied its FUR for the response to its triggering Entry. A pause during the processing for technical reasons shall not be considered as termination of the functional process.**

Change 6.2: The text of section 3.2.2 will be enhanced to emphasize the *process* of identifying functional processes in light of the improvements to rule a) above. The process, after the functional users have been identified and given the FUR for the software being measured, should be:

- Identify the separate events in the world of the functional users that the software being measured must respond to – the 'triggering events'
- Identify which functional user(s) of the software may respond to each triggering event
- Identify the triggering Entry (or Entries) that each functional user may initiate in response to the event
- Identify the functional process started by each triggering Entry

Change 6.3 As noted above, the previous rule h) will now be presented as two examples:

BUSINESS EXAMPLES: *Separate triggering event (-types) and therefore separate functional process (-types) should be distinguished in the following cases:*

- *When a human functional user makes decisions outside the software on 'what to do next' that are independent in time and that require separate responses from the software, each separate decision is a separate triggering event for which the software must provide a separate functional process. Example: a supplier functional user enters a customer order for an item of complex industrial equipment and later confirms acceptance of the order to the customer. Between entering the order and accepting it, the user may make enquiries about whether the new order can be delivered by the requested delivery date, and about the customer's credit-worthiness, etc. Although acceptance of an order must follow entry of an order, in this case the user must make a separate decision to accept the order. This indicates separate functional processes for order entry and for order acceptance.*
- *When the responsibilities for activities are separated (e.g. in a personnel system where the responsibility for maintaining basic personal data is separated from the responsibility for maintaining payroll data). Separate functional users each with their own separate responsibilities indicates separate functional processes.*

7. Proposed changes to section 3.2.3 c) of the Measurement Manual

Change 7.1: Section 3.2.3 *('Triggering events and functional processes in the business applications domain')*, paragraph c), must be corrected in light of the above. Paragraph c) is reproduced below showing the proposed new text in bold blue and the deletions as strikethroughs.

"There is no difference in principle to the analysis of a functional process whether it is required to be processed on-line or in batch mode. **By definition, all data that has been entered as *input* for batch processing must be temporarily stored somewhere before the process can start. (N.B. we distinguish input data – all the Entries = from other persistent data that might also need to be read or written by the batch process.) When measuring a batch-processed application, any temporarily-stored input data should be analysed in the same way as if it were being entered directly to the application. This input data should not be regarded as 'persistent data'.**

NOTE. A requirement that some input data be batch-processed is a non-functional requirement (NFR). The effect of this NFR is that the input data must be available (as a 'batch') for input to the batch process. How that happens in practice does not concern the analysis of the FUR of the batch process The NFR does not affect the FUR for the batch process.

~~The overnight processing is a technical implementation decision, and 'all that is required to be done' hasn't happened until the overnight process is completed. Given a requirement for one or more~~ **Note also that each** functional process (-type) to be processed in a batch 'job' ~~in batch mode, each functional process~~ should be analyzed **in its entirety**, independently of any other functional processes in the same job. Each functional process **in the job must have** ~~has~~ its own triggering Entry." ~~which must be measured~~.

The Business Example 3 is also not correctly explained in the Measurement Manual version 3.0.1.

.

*"BUSINESS EXAMPLE 3: Suppose the orders in the Example 1 above are entered by an 'off-line' process in some way, **e.g. by optical scanning of paper documents,** and are stored **temporarily** for automatic ~~overnight~~ batch processing. **How to analyze the batch functional process?** The functional user is the human who causes the order **data to be entered off-line and** to be processed in batch mode~~; on-line~~; the triggering Entry **of the functional process that will process the orders in batch mode** is the **data movement that moves** the order data **group from temporary storage into the process. (The off-line process, if it must be measured, involves another, separate functional process. Effectively, the functional user***

***has initiated two triggering Entries, one starts the off-line process to load the orders to temporary storage and the other to start the batch processing of the orders.)*”**

8. Proposed revisions to section 3.2.5 of the Measurement Manual

Change 8.1: Following the deletion of Rule g) for a functional process (see Change 6.1 above), the Examples 2 and 3 of section 3.2.5 of the Measurement Manual will be replaced by the following text and simpler Examples.

“According to clause c) of its definition, a functional process must ‘meet its FUR for all the possible responses to its triggering Entry’. This means that the same one functional process *type* must be able to deal with all possible *occurrences* of *values* of the data attributes of the data group moved by its triggering Entry, including both valid and invalid data values, and even in some cases missing data values. All these variations in values of the data moved by the triggering Entry will usually result in different processing paths being followed when the functional process executes. But in spite of all these variations, we must still identify only the one functional process type started by its one triggering Entry. (And the Measurer only needs to identify the unique data movements of this functional process; the various processing paths in which they may occur are irrelevant to the measurement.)

BUSINESS EXAMPLE 2: *A business application functional process will normally execute different processing paths depending on the values (valid or not) of the input data.*
BUSINESS EXAMPLE 3: *A functional process that provides a general search capability against a database may be required to function with up to four search parameters (attributes of its triggering Entry). But the same functional process must still be able to function if the values of only one, two or three search parameters are entered.*
BUSINESS EXAMPLE 4: *For a functional process to register a new customer for a car rental company, it is mandatory to enter data for most data attributes, but some (e.g. some contact details) are optional and may be left blank. Regardless of whether all or a sub-set of these attributes are entered there is only one functional process for registering a new customer. Similarly, for the functional process to make a car rental reservation in the same company, there are several options which may or may not be taken up, e.g. for extra insurance, additional drivers, requests for child seats, etc. These different options lead to different processing paths within the car rental reservation functional process, but there is still only one functional process for reserving a car rental.*
REAL-TIME EXAMPLE*: One triggering Entry (aircraft altitude information sent by the Geographical Positioning System) to a functional process of an avionics system will trigger one of two quite different processing paths within the functional process depending on the value of the Entry, i.e. whether the altitude is above or below a given height. The different paths will display different data groups on the pilot’s map and, if the altitude is too low, additional warnings will be issued.”*

9. Proposed new section 3.2.7 of the Measurement Manual

Change 9.1: The ‘independence’ of functional processes as in clause a) of the definition of a functional process will be further supported by adding the following new section 3.2.7.

3.2.7: Independence of functional processes sharing some common or similar functionality

Any two or more functional processes in the same software being measured may have some functionality that is identical or very similar in each process. This phenomenon is referred to as ‘functional commonality’, or functional ‘similarity’.

However, in the COSMIC method (as in all other FSM Methods) each functional process is defined, modelled and measured independently of, i.e. without reference to, any other functional process in the same software being measured (see clause a) of the definition of a functional process). Any required functionality that is common to or similar across any two or more functional processes in the same software being measured must be accounted for in each of these functional processes. The following are examples of functional commonality or similarity that may be found in practice.

BUSINESS EXAMPLE: Several functional processes in the same software being measured may need the same validation functionality for e.g. 'date of order', or may need to access the same data, or may need to carry out the same interest calculation.

REAL-TIME EXAMPLE: Several functional processes in the same software being measured may need to obtain data from the same sensor (common movement of the same data group) or may need to carry out the same scale conversion calculation, e.g. from Fahrenheit to Centigrade (common data manipulation).

When a statement of FUR is implemented in software, any 'functional commonality' may or may not be developed as re-usable software. The extent of actual or potential software re-use arising from functional commonality or similarity may therefore need to be taken into account when using functional size measurements for performance measurement or project effort estimating purposes.

10. Proposed new section 3.2.8. 'Events that trigger a software system to start executing'

Change 10.1: In teaching the COSMIC method and in practical measurement work, we have observed that Measurers sometimes have difficulty deciding what are the triggering events and the resulting triggering Entries when measuring batch-processed business applications. To understand the correct way to measure any applications, particularly if batch-processed, the Measurer must distinguish:

- the events that give rise to the functional processes of the application to be measured, and
- the event that triggers the start (or 'launch') of the (batch) application to be measured.

The former are of interest. The latter should be ignored.

The confusion can lead to incorrectly identifying the triggering Entries for functional processes that are executed in batch mode. This topic is discussed in the 'Guideline for sizing Business Application Software' (section 4.4.3). But the topic is more general, so a new section is proposed for the Measurement Manual.

"Section 3.2.8: Events that trigger a software system to start executing

When measuring the size of a piece of software, identify only the events and corresponding triggering Entries that trigger the functional processes that the software must respond to as defined in its FUR. Functionality needed to start-up (or 'launch') the software itself is not part of these functional processes and should be ignored (or measured separately, if required).

The following examples describe how software is started in three domains.

BUSINESS EXAMPLE: *For a business application, the functional user that starts the application may be a scheduler component of the operating system, a computer operator, or any other human user (e.g. when a PC user launches a browser or word-processing software).*

REAL-TIME EXAMPLE: *For a real-time application, the functional user that starts the application may be the operating system or network management generating a clock signal, or a human operator (e.g. to start a process control system from an operator workstation).*

INFRASTRUCTURE EXAMPLE: *For a computer operating system, the functional user that starts the operating system is a bootstrap program that is started when the computer power is switched on.*

The following are examples from two domains of the relationships that may exist, if any, between the event/process that starts the software to be measured and the events/processes that the software must execute as described in its FUR.

BUSINESS EXAMPLE 1: *An application to process the input data for a variety of functional processes in batch mode may be started by a scheduler of the operating system. If the purpose is to measure the FUR of the batch application, the 'start-the-system' functionality should be ignored*. *The triggering Entries for the functional processes of the batch-processed application and any other Entries that may be required will form the input data for the batch application.*

BUSINESS EXAMPLE 2: *Exceptionally, a batch-processed application to produce summary reports at the end of a time-period may be started without needing any input data provided directly from the functional user. In spite of the lack of input data, each functional process of the application must have a triggering Entry that commands: 'produce this end-of-time-period summary report'.*

REAL-TIME EXAMPLE: *A modern vehicle has a distributed system of Electronic Control Units (ECU's) to control many functions, e.g. engine management, brakes, air-conditioning, etc. In a distributed system, the "Network Management" (NM) module, which is always running, is responsible for activating the ECUs that are connected together via a network (bus). This module also handles the coordinated switching between the ECU operating states: Normal Operation, Low Power and Sleep. Therefore it is the NM that wakes up or puts to sleep ECUs. When measuring any ECU application software, this NM functionality should be ignored.*

11. Futher impact of the proposed changes:

The proposed changes will result in the need for changes in a few other places in the Measurement Manual, and in the Guideline for measuring Business Applications.

All the above changes will be incorporated in the next release of the Measurement Manual.

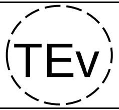
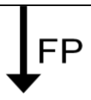
The Measurement Practices Committee

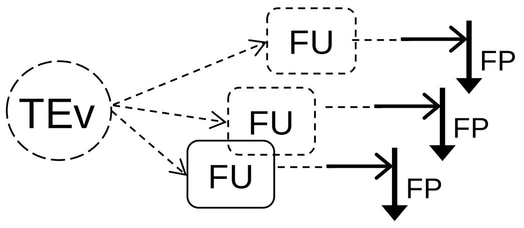
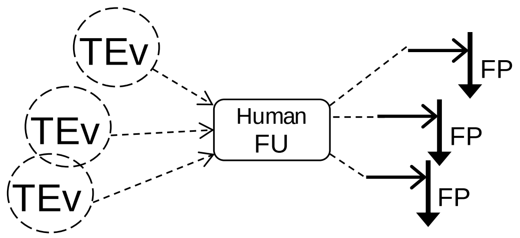
December, 2013

Appendix: Examples of the cardinality of relationships between triggering events, functional users, triggering Entries and functional processes

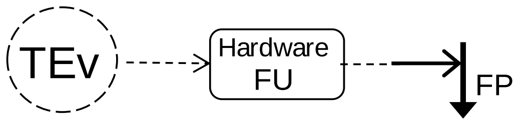
All the relationships along the triggering event / functional user / triggering Entry / functional process chain (as shown in Fig. 3.2.1) may be many-to-many in principle, with one exception. (The exception is that any one triggering Entry may initiate only one functional process – see Rule b) for a functional process in section 3.2.2.)

The table below shows examples of possible relationships. Note that the cases may not be exhaustive. The table uses the following abbreviations and symbols.

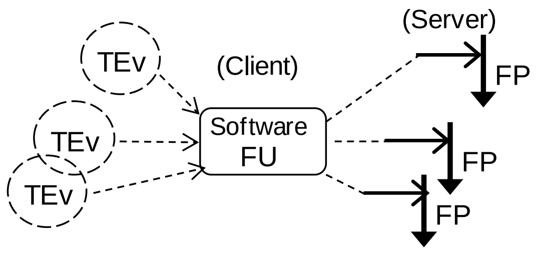
	Triggering event		Functional User
	Data group (dotted part) moved by a triggering Entry (solid arrow)		Functional Process

1. A single triggering event may cause multiple FU's to each initiate a triggering Entry in the <u>same</u> or in <u>different</u> software systems. Each triggering Entry starts its own FP
 *REAL TIME EXAMPLE: The triggering event of an earthquake may be detected by multiple independent FU sensors. Each FU initiates a triggering Entry that starts its FP in the same or in different systems.* *BUSINESS EXAMPLE: The triggering event of a new employee starting work causes one human FU to enter basic employee data to a Personnel system and another human FU to enter salary data to a Payroll system.*
2. A human functional user can sense or generate (e.g. when taking a decision) many types of triggering events. Each of these causes the FU to initiate a different triggering Entry. Each of these starts its FP in the <u>same</u> or in <u>different</u> software systems.
 *BUSINESS EXAMPLE: In a police emergency telephone call-handling system, many types of triggering events may be reported causing a human FU to decide to initiate different triggering Entries. Each of these starts its FP to record the event. Additionally, the human user may initiate different enquiry triggering Entries. Each of these starts its FP in the same call-handling system or in other systems.*

3. A hardware or software FU may be designed to sense (or 'generate') one or more specific type(s) of events. Each event causes the FU to initiate a triggering Entry. Each triggering Entry starts its FP in the <u>same</u> software system.

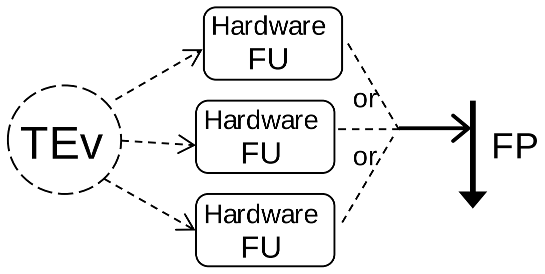


REAL-TIME EXAMPLE: When the temperature of a liquid reaches a pre-set level (the triggering event), a thermocouple FU initiates a triggering Entry to start its FP in one specific software system.



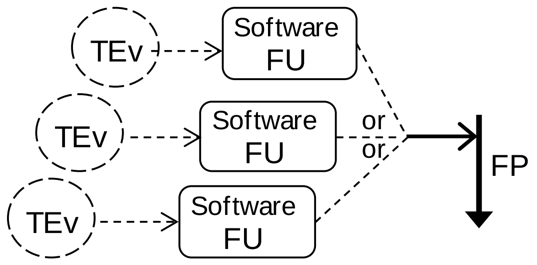
BUSINESS EXAMPLE: In a distributed software application, the client component is a FU of the server component. Different needs for information (the triggering events) of the client component cause it to initiate different triggering Entries, each to start its FP in the server component, for each different type of service it needs.

4. Two or more hardware FU's of the same software may sense the same triggering event. Each FU may initiate the triggering Entry that starts the <u>same one</u> FP.



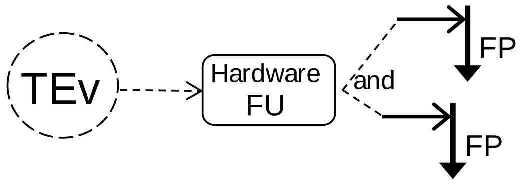
REAL-TIME EXAMPLE: The triggering event of an abnormal situation in a real-time process control system, may be sensed by one or more hardware FU's. Each FU may initiate the one emergency shut-down FP.
(N.B. any one occurrence of this FP will be initiated by the first FU to sense the triggering event.

5. **Two or more software FU's may each initiate a triggering Entry that starts the <u>same one</u> FP.**



INFRASTRUCTURE EXAMPLE: Several software FU's may each 'call', i.e. initiate, the same FP in the same re-usable software component. (In this case the software FU 'generates' the event when it calls the component.) (N.B. any one occurrence of this FP can be initiated by only one of its possible software FU's at any one time.)

6. On sensing one triggering event, a FU may initiate two or more triggering Entries. Each triggering Entry starts its FP in the <u>same</u> software,



REAL-TIME EXAMPLE: In a duplex safety-critical control system, one triggering event may cause a FU (usually hardware) to initiate two triggering Entries, each starting its FP. The two FP's could, for example, have the same FUR but be developed by separate groups as a result of a diversity strategy.