The COSMIC Functional Size Measurement Method
Version 4.0.2

Rice Cooker Case Study

Version 2.0.1
August 2018

Table of Contents

1

RICE COOKER FUNCTIONAL REQUIREMENTS

1.1 Context

This case study illustrates the application of the COSMIC measurement method (version 4.0.1) to measure the functional size of the software embedded within the prototype of a simple Rice Cooker.

This is the first Rice Cooker to be built by this manufacturing company: a decision was made to develop the product by using successive prototypes. The case study presented here corresponds to the first prototype to be developed plus some other possible functions specified for a second prototype. The lessons learned in manufacturing the prototypes will help make better informed decisions on whether or not some of the product functional requirements should be re-allocated to software rather than to the hardware.

For this first simple prototype of a Rice Cooker, the system engineer specified:

- which functions will be implemented through hardware devices (section 1.2);
- the software-hardware interactions (section 1.3); and
- which functions will be implemented through software (section 1.4).

Remember this is a prototype and not all functions of an operational rice cooker have yet been specified. Therefore Measurers must measure the functions described in the Functional Requirements, and **not** what else could have been specified.

1.2 Hardware functions

A power switch provides electricity to all hardware devices. This must be turned on by the Rice Cooker operator so that the cooker can work.

The Cooker is fitted with the following hardware devices that may send data to or receive data from the software directly. An exception is the Cooking Mode button which puts the cooking mode into a random access memory ('RAM') that is accessible by the software.

1. The Cooking Mode button is a set of three inter-connected buttons that allows an operator to set one of 3 cooking modes: slow, normal or fast.
   a) When one of the Cooking Mode buttons is pressed by the operator, the selected cooking mode is put into a RAM where it may be accessed by the software.
   b) Once the Start button has been pressed, the hardware will not allow the cooking mode to be changed.
2. The Start button.

   When the operator presses the Start button it:
   a) sends a 'start' signal to the software;
   b) starts a hardware Timer that provides the sole time reference for the software.

   If the operator presses the Start button without having set the cooking mode:
   c) the hardware sets the cooking mode to the default value of 'normal' in the RAM.
3. The Timer emits three types of signals to the software.
   a) At 1 second intervals, the timer emits a signal conveying the value of the elapsed time 't' (i.e. at t = 0, t = 1, t = 2, t = 3, ….; the elapsed time values are 0,1,2,3…. respectively);

b) Every 30 seconds the timer emits a signal for the software to update the target cooker temperature. (This signal is first emitted at t = 0 and after every subsequent interval of 30 seconds);

c) Every 5 seconds the timer emits a signal for the software to check the actual cooker temperature. (This signal is first emitted at t = 5 and after every subsequent interval of 5 seconds).

At time t = 0, the signals are emitted in a priority sequence b), a).

In the cases where three types of signals must be emitted at the same time, they are emitted in a priority sequence b), a), c).

4 The Cooking Lamp is turned ON by a command signal received from the software at start-up.

5 The Read-only Memory ('ROM') is used to store the 'target temperature / mode / elapsed time' data shown in Figure 2, which is accessible to the software. This table of values is used by the software to determine the target temperature depending on the elapsed time and the cooking mode.

6 The Temperature Sensor measures and makes the temperature available to the software when requested by the software.

7 The Heater is controlled (ON or OFF) by a signal received from the software.

8 The Stop button, when pressed at any time, will:

a) stop the Timer (if the timer is re-started the elapsed time signal will start at t =0);

b) cut off power to all devices.

Safety interactions between the buttons, the power supply and the Rice Cooker door are controlled by hardware in this prototype of the Rice Cooker and can be ignored.

Figure 1 shows the Rice Cooker control panel as seen by a human operator.

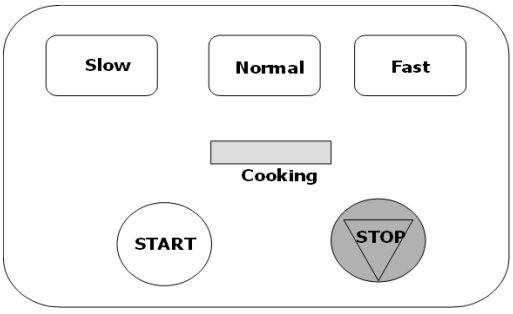

Figure 1 - The operator control panel

1.3 Software - hardware interactions

The software must:

1 accept a signal from the Start button;

2 accept 3 distinct types of signals from the Timer (the signals at 5 and 30 second intervals and the elapsed time after a 30 second signal);

3 get the cooking mode from the RAM;

4 get the 'target temperature / mode / elapsed time' data from the ROM;

5 get the current temperature from the Temperature Sensor;

6 send a 'Turn ON' command to the Cooking Lamp;

7 send a ‘Turn ON’ or ‘Turn OFF’ command to the Heater.

8 store the current target temperature in the RAM, and retrieve (or ‘get’) it from the RAM.

1.4 The software Functional User Requirements (FUR)

FUR 1 – on receipt of the Start signal

The software:

a) sends a ‘Turn ON’ signal to the Cooking Lamp’

b) sends a ‘Turn ON’ signal to the heater.

FUR 2 – on receipt of a 30-second signal

Starting at t = 0, and at each following 30-second interval, the software:

a) receives the 30-second signal;

b) waits for and receives the elapsed time signal;

c) gets the cooking mode from the RAM;

d) selects a new target temperature, for the cooking mode by getting it from the relationship in the ROM at time = [current elapsed time + 30 seconds]. See Figure 2;

e) stores this new target temperature into the RAM, which becomes the current target temperature until the next 30 second signal.

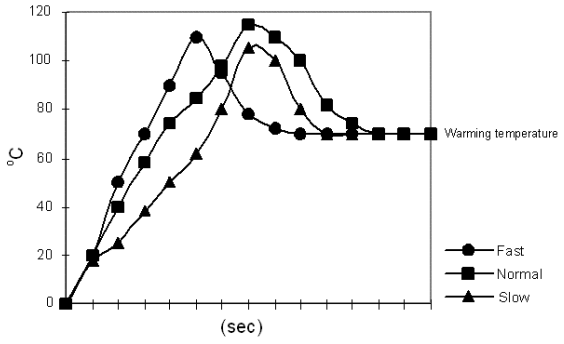

Figure 2 - Target Temperatures by Cooking Mode at 30 second Intervals

(Horizontal axis = elapsed time at 30 second Interval)

FUR 3 - on receipt of a 5-second signal

Starting at t = 5 seconds and at each following 5-second interval, the software:

a) receives the 5-second signal;

b) gets the current target temperature from the RAM;

c) gets the actual sensor temperature from the Temperature Sensor;

d) sends a Turn ON signal to the Heater if the ‘current target temperature’ is greater than the ‘actual sensor temperature’; otherwise, it sends a Turn OFF signal to the Heater.

For the specific purpose of this case study, only the functions explicitly assigned to the software are to be implemented through the software. All other functions (including those not yet explicitly documented or requested) are to be implemented through hardware in this first prototype, including functions that the engineers might find necessary in the course of building the prototype.

2

MEASUREMENT STRATEGY

2.1 Measurement purpose

The purpose of the measurement is to determine the size of the application software of the first prototype on the basis of its Functional User Requirements, **as specified**.

2.2 Measurement scope

The scope is all functionality allocated to software as specified in the functional user requirements (FUR) for the software. The FUR do not specify any decomposition of the software.

The Rice Cooker software is in the one application layer.

2.3 Identification of the functional users

For this first prototype of the Rice Cooker, there are 5 functional users for this software

- the Timer,
- the Start button,
- the Temperature Sensor,
- the Cooking Lamp,
- the Heater.

Note: The RAM and the ROM are not functional users. (A functional user is defined as 'a sender or intended recipient of data' in the FUR of the software being measured [1].) See further below.

The human operator of the Rice Cooker is not a functional user of the software. The operator is the user of the Rice Cooker as a product. The operator interacts with the software only via the hardware through which signals are sent to or received from the software (that is, the human operator interacts indirectly via the Cooking Mode, Start and Stop buttons, and is informed of what is happening by the Cooking Lamp).

Based on the written requirements, Figure 3 shows the context diagram of the software, its functional users and the boundary between them.

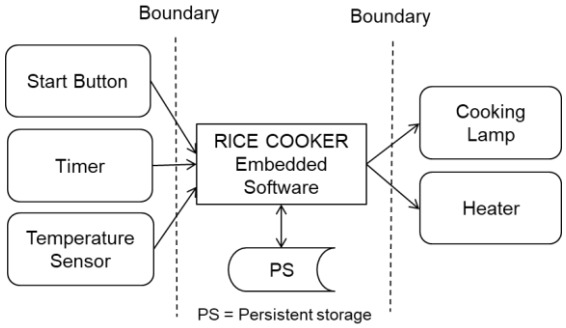

Figure 3 – Context diagram of the Rice Cooker software from its Functional User Requirements

Three of the hardware devices only send data to the software:

- The Start button,

- the Timer,
- the Temperature sensor. (Following the COSMIC method rules for a ‘dumb’ sensor [1], the ‘send me your data’ command from the software is considered as accounted for by a single Entry data movement.)

Two of the hardware devices only receive data from the software:

- the Cooking Lamp,
- the Heater.

2.4 Other measurement strategy parameters

Level of granularity. The FUR of the Rice Cooker are at the ‘functional process level of granularity’ as the functional users are individual hardware devices (not groups of these) and single events occur that the Rice Cooker software must respond to (not groups of events).

Persistent storage. The COSMIC model does not recognize specific types of hardware storage such as RAM and ROM. It recognizes only persistent storage which, in the model therefore holds:

- The current cooking mode;
- The target temperatures for each cooking mode at 30 second-intervals (as in Figure 2);
- The current target temperature.

3

THE MAPPING AND MEASUREMENT PHASES

3.1 Identification of software triggering events

From the functional user requirements, the following triggering events are identified:

1 Start button pressed. (FUR 1)

2 Timer signal event at 30 second intervals. (FUR 2)

The event is created by the Timer, starting at t = 0 and every following 30 seconds, when the software has to select a new target temperature.

3 Timer Signal event at 5 second intervals. (FUR 3)

The event is created by the Timer, starting at t = 5 seconds and every following 5 seconds, when the software has reset the heater as needed.

These events are Triggering Events because they cause a functional user (The Start Button or the Timer) to generate a data group (the respective signals) that triggers a functional process of the software.

The emission of the elapsed time signal is not a triggering event for the software. Although the elapsed time is a data group issued by the Timer, the FUR do not specify that the software must do anything with the emission of the elapsed time signal, except after receiving a 30-second time signal.

The physical mechanisms whereby the software takes notice of (i.e. accepts) the elapsed time after receiving the 30-second time signal, but ignores the elapsed time at all other times is of no concern to the measurement of the functional size.

3.2 Identification of functional processes

Given the triggering events of the previous section, the following candidate functional processes are identified from the requirements:

1 Start cooking (on receipt of the Start signal)

2 Update target temperature (30 second Timer signal).

3 Check cooker temperature, and switch the Heater ON or OFF, as needed (5 second Timer signal)

3.3 Identification of objects of interest

As noted in the Measurement Manual [1], section 3.3.4, in real-time software systems, a physical hardware device can be a functional user and also an object of interest. In effect the hardware device interacts with the software to provide or to receive data about itself. Consequently, the objects of interest and their respective data attributes are as follows.

Start button. Start signal

Cooking Lamp. Cooking lamp On/Off command

Heater. Heater On/Off command

Timer. Current elapsed time signal, 30-second signal, 5-second signal

Temperature sensor. Actual sensor temperature

Target settings. Cooking mode, elapsed time, target temperature (i.e. the data shown in Figure 2).

Current settings. Current cooking mode, current target temperature.

A consequence of the fact that a functional user can also be an object of interest is that the temperature provided by the Temperature Sensor is named as the 'actual sensor temperature'. This is odd in ordinary language. (We would not describe the reading of a domestic thermometer as the 'thermometer temperature', but as the temperature of its environment, e.g. 'room temperature'.) However, in real-time software, the state of a sensor is a property of the sensor, not of its environment.

3.4 The functional processes and their data movements

This section describes the three functional processes with their data movements. Sizes are indicated in the COSMIC unit: 1 COSMIC function point = 1 CFP.

Data movement types are abbreviated as E = Entry, X = Exit, R = Read, W = Write.

Functional Process 1: Start cooking
Triggering Event: Start button pressed

Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Mvt. Type	CFP
Start button	Receive Start signal	Start signal	Start button	E	1
Heater	Send a Turn ON command to the Heater	Heater On/Off command	Heater	X	1
Cooking Lamp	Send a Turn ON command to the Cooking lamp	Cooking On/Off lamp command	Cooking lamp	X	1
					Total size: 3 CFP

Functional Process 2: Update Target temperature
Triggering Event: 30-second signal

Functional user	**Sub-process Description**	**Name of Data Group moved**	**Object of interest of Data Group moved**	**Data Mvt. Type**	**CFP**
Timer	Receive 30-second signal	30-second signal	Timer	E	1
Timer	Receive Current elapsed time signal	Current elapsed time signal	Timer	E	1
	Get cooking mode	Current cooking mode	Current settings	R	1
	Get New target temperature for [Current elapsed time + 30	New target temperature	Target settings	R	1

	secs.] and Cooking mode. (This will replace the Current target temperature)				
	Store the (new) Current target temperature	Current target temperature	Current settings	W	1
					Total size: 5 CFP

Functional Process 3: Check cooker temperature **Triggering Event: 5-second signal**					
Functional user	**Sub-process Description**	**Name of Data Group moved**	**Object of interest of Data Group moved**	**Data Mvt. Type**	**CFP**
Timer	Receive 5-second signal	5-second signal	Timer	E	1
	Get Current target temperature	Current target temperature	Current settings	R	1
Temperature Sensor	Get Actual sensor temperature	Actual sensor temperature	Temperature sensor	E	1
	Decide if Heater must be turned ON or OFF	(None – this is data manipulation)	-	-	-
Heater	Send a Turn ON or OFF command (as needed) to the Heater	Heater On/Off command	Heater	X	1
					Total size: 4 CFP

The total functional size of the software for this first prototype of the Rice Cooker is the sum of the sizes of its three functional processes, that is:

3 CFP + 5 CFP + 4 CFP = **12 CFP.**

4

POSSIBLE REQUIREMENTS FOR A SECOND PROTOTYPE

This section describes three possible new requirements to be considered for a second prototype of the Rice Cooker. The effect of each possible requirement on the size of the software must be measured separately.

4.1 Stop function

Hardware requirements

In the first prototype, all the Stop functions were handled entirely by hardware. This first possible change to the Rice Cooker requirements is that when the Stop button is pressed, the hardware will:

a) Stop the timer;
b) Send a signal to the software to 'stop cooking'. The software must then perform the other 'stop' functions that the hardware performed in the first prototype.

The effect of this change on the other requirements are as follows.

Software – hardware interactions

The software must accept a 'stop' signal from the hardware.

Software FUR 4 – Allocate some Stop functions to software

Upon receiving the Stop signal from the Stop button:

a) the software sends an OFF command to the Heater;
b) the software sends an OFF command to the Cooking lamp;
c) the software stops executing, i.e. it enters a self-induced wait state.

Functional users

There is now a new functional user to the software: the Stop Button which sends a signal to the software. Figure 4 shows the updated context diagram for the software.

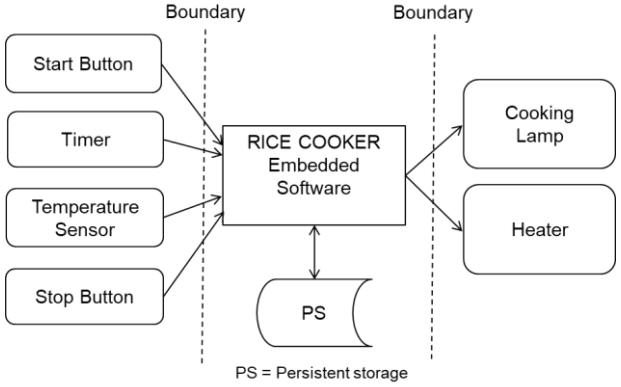

Figure 4 – Context diagram for the second prototype of the Rice Cooker software

Triggering event – Functional process

The triggering event of the operator pressing the Stop button triggers an additional 'Stop Cooking' functional process to begin executing.

Object of interest

There is now one more object of interest, the Stop button, with one data attribute, the Stop button signal.

The Functional Process 4 and its data movements

Functional Process 4: Stop Cooking
Triggering Event: Stop button pressed

Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Mvt. Type	CFP
Stop button	Receive Stop button signal	Stop signal	Stop button	E	1
Heater	Send a Turn OFF command to the Heater	Heater On/Off command	Heater	X	1
Cooking lamp	Send a Turn OFF command to the Cooking lamp	Cooking lamp On/Off command	Cooking lamp	X	1
	Stop executing and enter wait state	(None)	-	-	-
					Total size: 3 CFP

This possible change would therefore increase the software size by 3 CFP, bringing the total size of Prototype 2 to **15 CFP.**

4.2 Combine elapsed time and 30-second signals

The second possible change to the Rice Cooker requirements is that instead of emitting separate Elapsed time and 30-second time signals (as specified in section 1.2), the Timer should now emit a signal every 30 seconds, starting at t = 0, which also conveys the Elapsed time in the same one signal.

The Timer will no longer emit separate Elapsed time signals at 1-second intervals.

The following changes are needed to the Rice Cooker Requirements as specified in Chapter 1.

Hardware Function 3. This now becomes:

3. The Timer emits two signals to the software:

a) every 30 seconds a signal indicates it is time for the software to update the target temperature and informs the software of the current Elapsed time in seconds since t = 0. (This signal is first emitted at t = 0 and after every subsequent interval of 30 seconds).

b) every 5 seconds a signal indicates it is time for the software to check the cooker temperature. (This signal is first emitted at t = 5 and after every subsequent interval of 5 seconds).

In the cases where both types of signals must be emitted at the same time, they are emitted in a priority sequence a), b).

Software = hardware interaction 2. This now becomes:

2 The software must accept two distinct types of signals from the Timer, at 5 and 30 second intervals.

Functional Users. These are the same as in section 2.3.

Software triggering events. There are now only two triggering events generated by the Timer at 30 and 5-second intervals.

The objects of interest. These are the same as in section 3.3.

The software FUR. The requirement a) of FUR 2 now becomes:

Starting at t = 0, and at each following 30-second interval, the software:

a) receives the 30-second signal accompanied by the current elapsed time.

The effect of this possible change on Functional Process 2 and its data movements

Functional Process 2 will now receive only one Entry every 30-seconds. This also conveys the current elapsed time since t = 0 as a second attribute (instead of two separate Entries for the 30-second signal and the Elapsed time signal).

The size of Functional Process 2 decreases by 1 CFP to 4 CFP.

With this change, the total software size of Prototype 2 decreases from 12 CFP to **11 CFP**.

4.3 Use a simple timer that 'ticks' at 1-second intervals

The third possible change to the Rice Cooker requirements is that the Timer used in the first prototype (that emits three separate Elapsed time, 30-second time and 5-second signals, as specified in section 1.2), should be completely replaced by a simple Timer that emits a 'tick' every second to the software, starting at t = 0. The intention is that the software should now keep track of the Elapsed time, from t = 0.

The following changes are needed to the Rice Cooker Requirements as specified in Chapter 1.

Hardware Function 3. This now becomes:

3 The Timer. This emits a simple 'tick' signal (like a clock) to the software every second, starting at t = 0 seconds after the Start button has been pressed.

Software - hardware interaction 2. This now becomes:

2 The software must accept a 'tick' signal from the Timer at 1 second intervals, the first at t = 0 seconds after the Start button has been pressed.

Functional Users. These are the same as in section 2.3, Figure 3.

Software triggering events. The three triggering events of the old Timer are now replaced by the single triggering event of a tick emitted by the new, simpler Timer.

The objects of interest. These are the same as in section 3.3, with two exceptions:

- the new, simpler Timer has only a single attribute, the 'tick';
- the software must keep track of the elapsed time. There is therefore another object of interest, which we call '**Elapsed time'**, with one attribute 'current elapsed time'.

The software FUR. It is now required that the software must keep track of the elapsed time since t = 0. This means several changes must be specified to the existing FUR, as follows.

Functional Process 1. 'Start Cooking'

This existing process must be modified to store a value t = 0 for the current Elapsed time. (Note the software must generate this value; it has not received a value of 't' or any tick at this point in time.)

Functional Process 5. 'Control Cooking'

A new Functional Process 5 must replace the existing processes 2 and 3 with the following FUR.

FUR 5 – on receipt of a tick from the Timer

Starting at t = 0, and at each following 1-second interval, the software:

a) receives the timer 'tick' signal;
b) gets the current Elapsed time 't' from persistent storage;
c) adds 1 second to the current Elapsed time 't' and makes the updated value persistent, replacing the previous Elapsed time;
d) if t = 0, or t = an exact multiple of 30 seconds, the software:
   i. gets the cooking mode from persistent storage;
   ii. selects a new target temperature for the cooking mode by reading the corresponding target temperature in persistent storage (see Figure 1) at time = current Elapsed time + 30 seconds;
   iii. puts this new target temperature into persistent storage, which becomes the current target temperature until the next time a tick arrives at an exact multiple of 30 seconds.
e) If t > 0 and t = an exact multiple of 5 seconds, the software:
   i. gets the current target temperature from persistent storage (if not already known from step d ii);
   ii. gets the actual sensor temperature from the Temperature Sensor;
   iii. sends an ON signal to the Heater if the 'current target temperature' is greater than the 'actual sensor temperature'; otherwise, it sends an OFF signal to the Heater.

The functional processes and their data movements

The modified functional processes 1 and the new functional process 5 and their data movements are as follows.

Modified Functional Process 1: Start cooking
Triggering Event: Start signal

Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Mvt. Type	CFP
Start button	Receive Start signal	Start signal	Start button	E	1
Heater	Send a Turn ON command to the Heater	Heater On/Off command	Heater	X	1
Cooking Lamp	Send a Turn ON command to the Cooking lamp	Cooking On/Off lamp command	Cooking lamp	X	1

	Store a value t = 0 for the Current elapsed time	Current elapsed time	Elapsed time	W	1
					Total size: 4 CFP

Functional Process 5: Control Cooker
Triggering Event: 1-second tick

Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Mvt. Type	CFP
Timer	Receive 1-second tick	1-second signal	Timer	E	1
	Get current elapsed time from persistent storage	Current elapsed time	Elapsed time	R	1
	Add 1 second to the current elapsed time	(Data manipulation)			
	Store current elapsed time	Current elapsed time	Elapsed time	W	1
	IF t = 0 or t = an exact multiple of 30, get cooking mode. ELSE skip this step and the next two steps	Cooking mode	Current settings	R	1
	Get the New target temperature for the [Current elapsed time + 30 secs.] and the Cooking mode. (This will replace the Current target temperature)	New target temperature	Target settings	R	1
	Store the (new) Current target temperature	Current target temperature	Current settings	W	1
	IF t = an exact multiple of 5, get the current target temperature, if not already known. ELSE terminate.	Current target temperature	Current settings	R	1
Temperature Sensor	Get actual sensor temperature	Actual sensor temperature	Temperature sensor	E	1
	Decide if Heater must be turned ON or OFF	(Data manipulation)	-	-	-
Heater	Send a Turn ON or OFF command (as needed) to the Heater	Heater On/Off command	Heater	X	1
					Total size: 9 CFP

This possible change would result in Prototype 2 having a total software size of 4 + 9 = **13 CFP** (compared with 12 CFP for Prototype 1).

Note: This Functional Process 5 only needs to execute eight of its data movements (8 CFP) at t = 0 and every following 30 seconds.

- At t = 5 and every following 5 seconds, except at multiples of 30 seconds, it executes 6 CFP.
- Every other second, i.e. t = 1, 2, 3, 4, 6, 7 …., it executes only 3 CFP, to update the elapsed time.

Nevertheless, Functional Process 5 needs a total of 9 CFP to meet all its FUR.

REFERENCES

All COSMIC publications are available for free download from the Knowledge Base of www.cosmic-sizing.org.

[1] Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2017), version 4.0.2, December 2018

[2] Lavazza, L., Del Bianco, V., 'A case study in functional size measurement'; Rice Cooker case study revisited', IWSM/Mensura conference 2009.

ACKNOWLEDGEMENTS

Version 2.0 Reviewers		
Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Jean-Marc Desharnais École de Technologie Supérieure, Université du Québec Canada	Peter Fagg Pentad Ltd United Kingdom
Luigi Lavazza Universita degli Studi dell'Insubria Italy	Arlan Lesterhuis The Netherlands	Dylan Ren Measures Technology LLC China
Charles Symons* United Kingdom	Sylvie Trudel Université du Québec à Montréal Canada	Frank Vogelezang Ordina The Netherlands
Chris Woodward United Kingdom		

* Editors of this Case

VERSION CONTROL

The following is a partial account of the evolution of this case study.

Date	Reviewers (s)	Modifications / Additions
2000	Abran, Desharnais, Fagg, Oligny, St-Pierre, Symons, De Tran-Cao	COSMIC-FFP versions. Timing controlled by software
2003-01-02	Abran, O'Neill, Vinh Tuong Ho, et al …….	Updates to: - synchronize the vocabulary with ISO 19761:2003; - to clarify which functional requirements are allocated to the hardware, and which functional requirements are allocated to the software; - to segregate the requirements into 3 iterative prototypes
2008-05-22	Abran, Fagg, O'Neill, Khelifi, Symons	Aligned with COSMIC v3.0, ISO 19761:2008. For 'prototype 1'
2010-11-30	Abran, Symons	
2016-01-01	Abran. Lesterhuis, Symons, Trudel	Significant revision. Clearer separation of hardware and software requirements. Aligned with COSMIC method v4.0.1 [1] and takes account of some points raised by Lavazza [2].
2018-08-10	Lesterhuis	The case itself unchanged. Some formats and outdated version numbers updated