COSMIC Measurement Manual for ISO 19761

Combination of Case Studies

ACME Car Hire Case Study v1.0.1
Automatic Line Switching v1.1
Course Registration ('C-REG') System v2.0.1
Expert System June 2022
Sizing software in a Machine Learning context v1.0
RestoSys v1.2
Rice Cooker v2.0.1
Industrial Automation Robot v1.0
Valve Control System v1.0.1
Web Advice Module v1.1.1

(Issued Nov. 2022 with unchanged copies of the existing Case Studies)

The COSMIC Functional Size Measurement Method
Version 4.0.2

ACME Car Hire Case Study

VERSION 1.0.1

August 2018

Table of Contents

1 ABOUT THIS CASE STUDY

1.1 Introduction

This Case Study presents the results of applying the COSMIC Functional Size Measurement method, version 4.0.2, to measure some of the implemented functionality of the ACME Car Hire System, as described in Chapter 2.

General information about the software application to be measured:

- Application domain: Business Application
- This is an implemented on-line system for which only some screen shots are available. These must be measured. Apart from a small amount of background information there is no other information available, for example there is no statement of requirements.

1.2 How to use this ACME case study

This case study aims to be suitable for new users of the COSMIC method who have at least read the 'Introduction to the COSMIC method' [1] or have received some introductory training in the method. Ideally, readers should have read the Measurement Manual [2]. It is also assumed that the reader is generally familiar with on-line business application software.

The purpose of the ACME case study is to show how to apply the COSMIC method to measure an existing implemented on-line business application. This type of measurement is often required after a project is finished to determine the size of the software actually delivered.

Often when a system has been in existence for some years, there may be no up-to-date documentation to help the Measurer. Normally in practice the Measurer should find an expert on the system to explain how it works but sometimes, as in this case, the challenge is to do the measurement just by examining the input and the output.

The ACME case aims to support the following teaching points on the COSMIC method:

- Identifying functional processes from the physical display screens (they do not always map one-to-one).
- Identifying the objects of interest about which data must be entered or is displayed.
- Identifying the data movements of a functional process, including that alternative processing paths may or may not lead to identifying additional data movements.
- The need to recognise that an enquiry functional process may or may not be identical to an 'enquire-before-update process'. Only one enquiry functional process should be counted per application if the FUR are identical for both situations. But in this case the FUR are different for the two situations.
- The COSMIC rules for ignoring 'Control Commands' and Menu selections and for measuring 'Error/Confirmation messages'.
- The software to be measured may have functionality that cannot be properly identified just from the physical input and output screens. The Measurer should ask questions of a system expert to find out details of any 'hidden' functionality. The case illustrates the very important lessons to be learned on the importance of identifying and measuring any 'hidden' functionality of installed software.

2 THE MEASUREMENT TASK AND THE SCREEN SHOTS

2.1 The Measurement Purpose and Scope

The Purpose is defined to the Measurer as 'to measure the functionality of the ACME Car Hire system represented by the screen-shots shown in section 2.2 (which define the Scope of the measurement).

The precision of the measurement will depend on what can be learned from examining the physical screens and what might be learned from asking an expert about how the system works.

The Measurer accesses and uses the system by entering trial data, finding the set of seven screens shown that must be measured. Figure 2.1 shows the screen titles and the sequence in which they are described in section 2.2; this is also one of the ways they may be navigated.

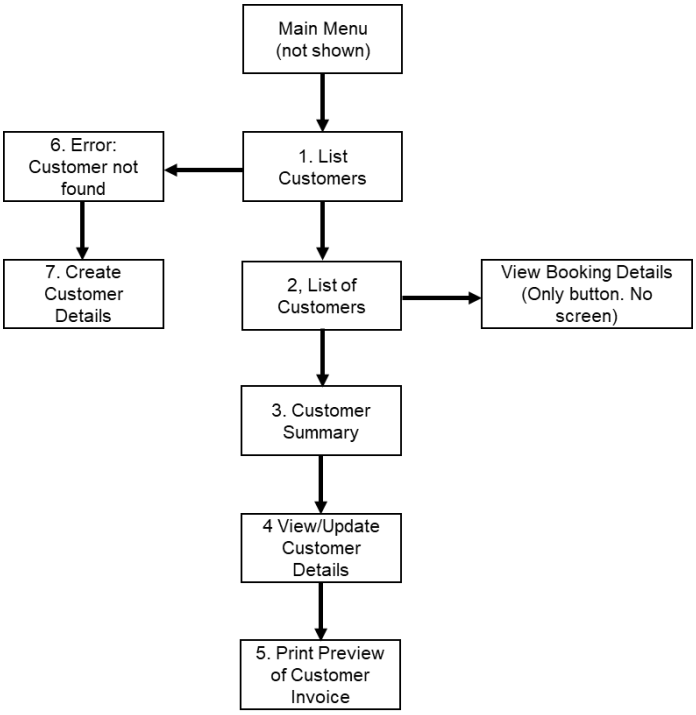


Figure 2.1 - ACME screens that must be measured

It is evident that the only functional user of the software is a human who enquires on and enters data about ACME Car Hire customers and their bookings.

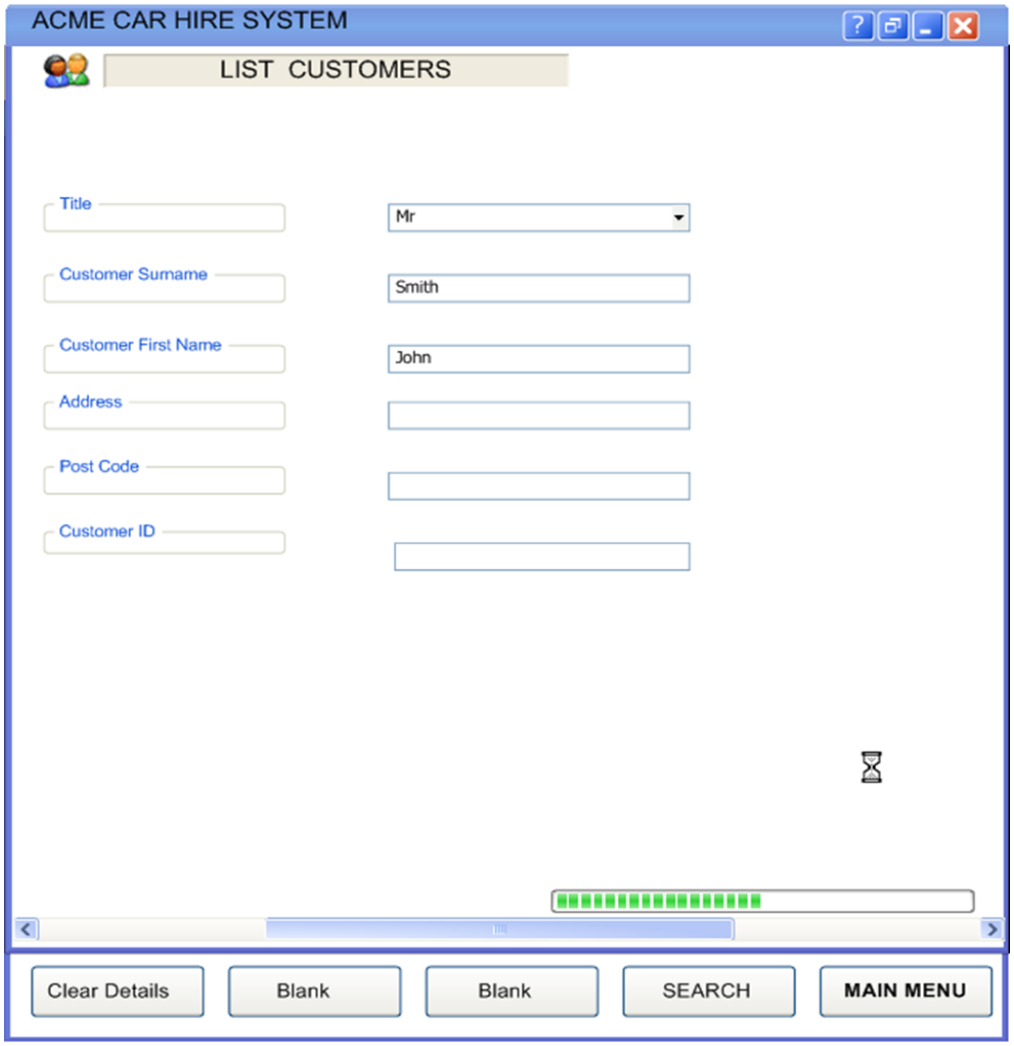
Figure 3.1 (see section 3.1) shows the names on the control buttons and all the ways in which the screens may be navigated.

2.2 The Screen Shots

The User selects 'View Customer' from the Main Menu (not shown) which results in displaying the data entry screen 1

2.2.1 Screen 1. List Customers

The User enters the title and name of a customer, a Mr John Smith in this case, and presses the ‘Search’ button to enquire if the customer exists in the system.

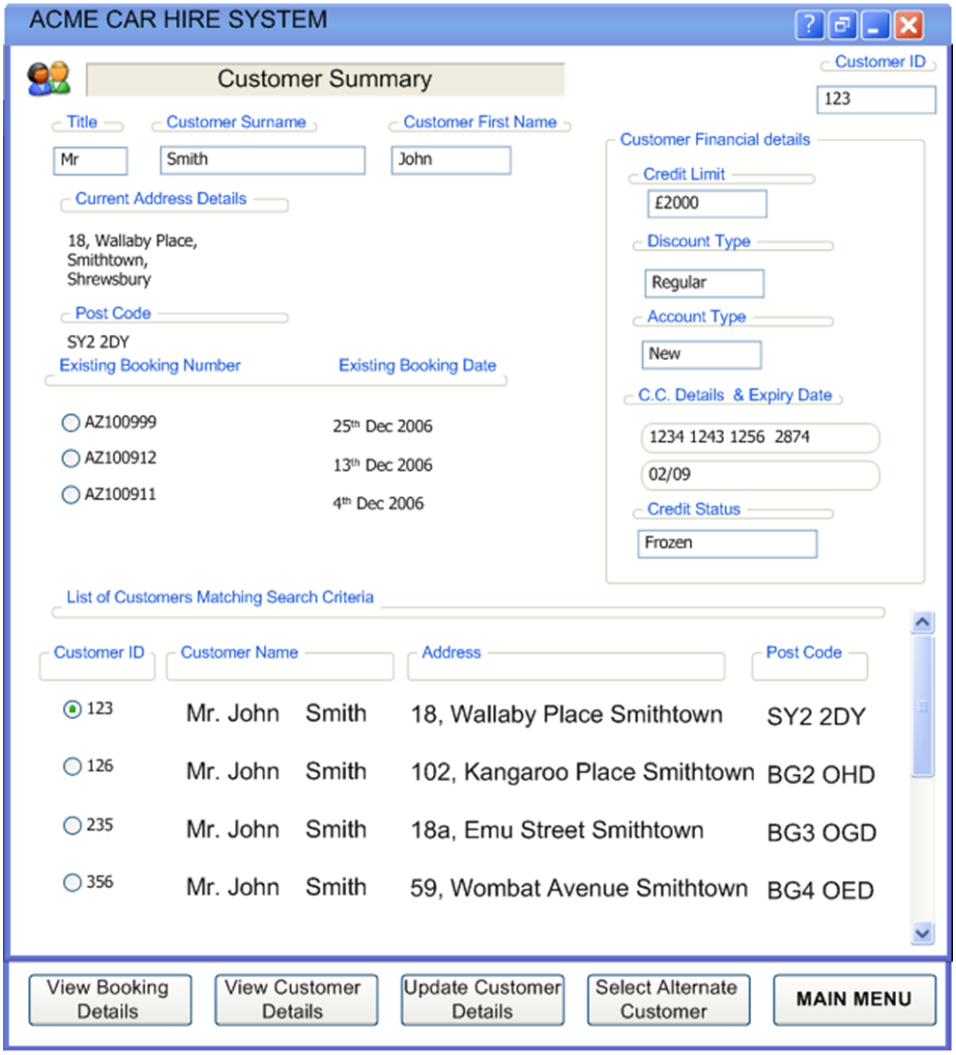


2.2.2 Screen 2. List of Customers

Screen 2 shows all four customers with their addresses that have the name ‘Mr John Smith’. If no customers of this name are found, Screen 6 is displayed.

2.2.3 Screen 3. Customer Summary

The User presses the radio button on Screen 2 corresponding to the customer he is really searching for (in this case the first of the four John Smiths). If the user next presses the ‘View Customer Summary’ button, then this Screen 3 is displayed.

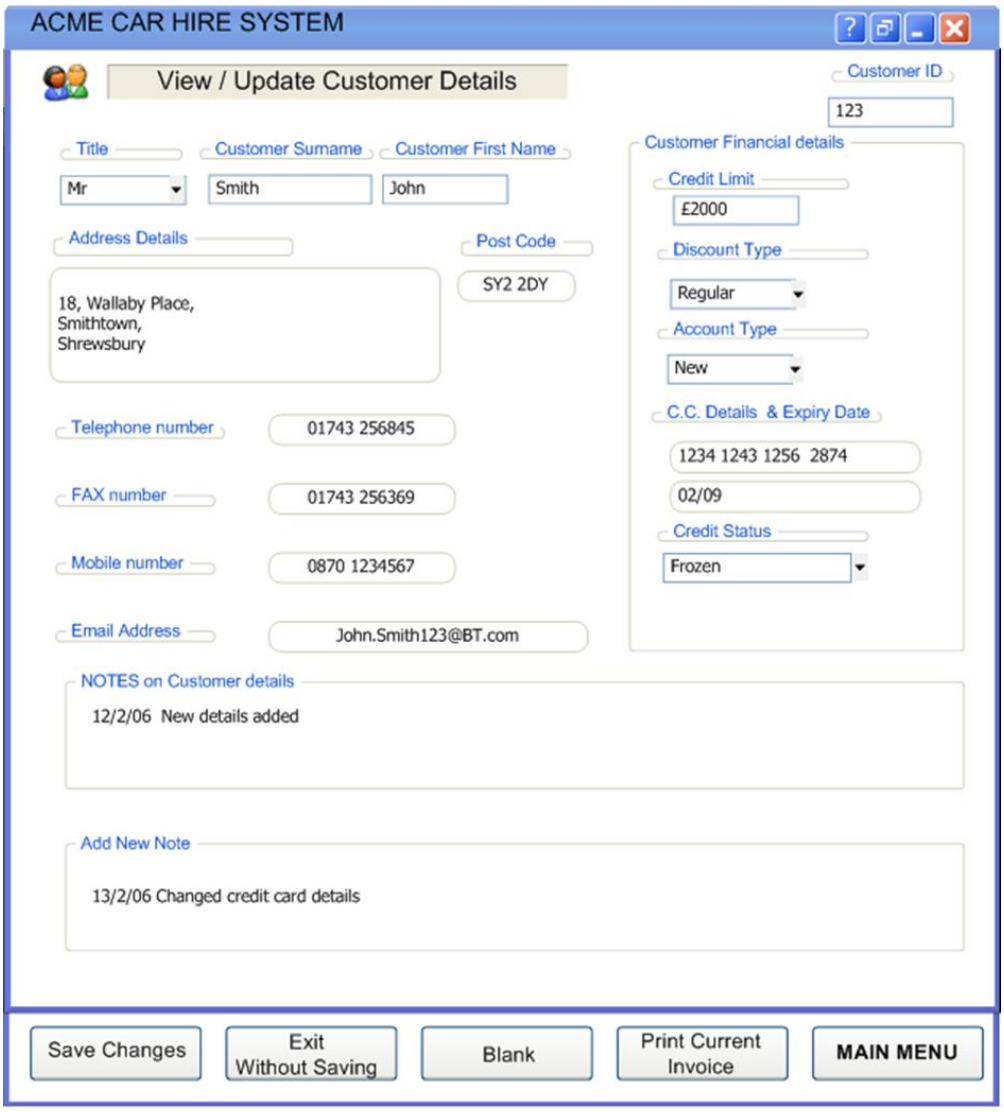


Screen 3 still displays the original four Mr John Smiths and their addresses, but also a summary of the data held about the selected Mr John Smith.

2.2.4 Screen 4. View/Update Customer Details

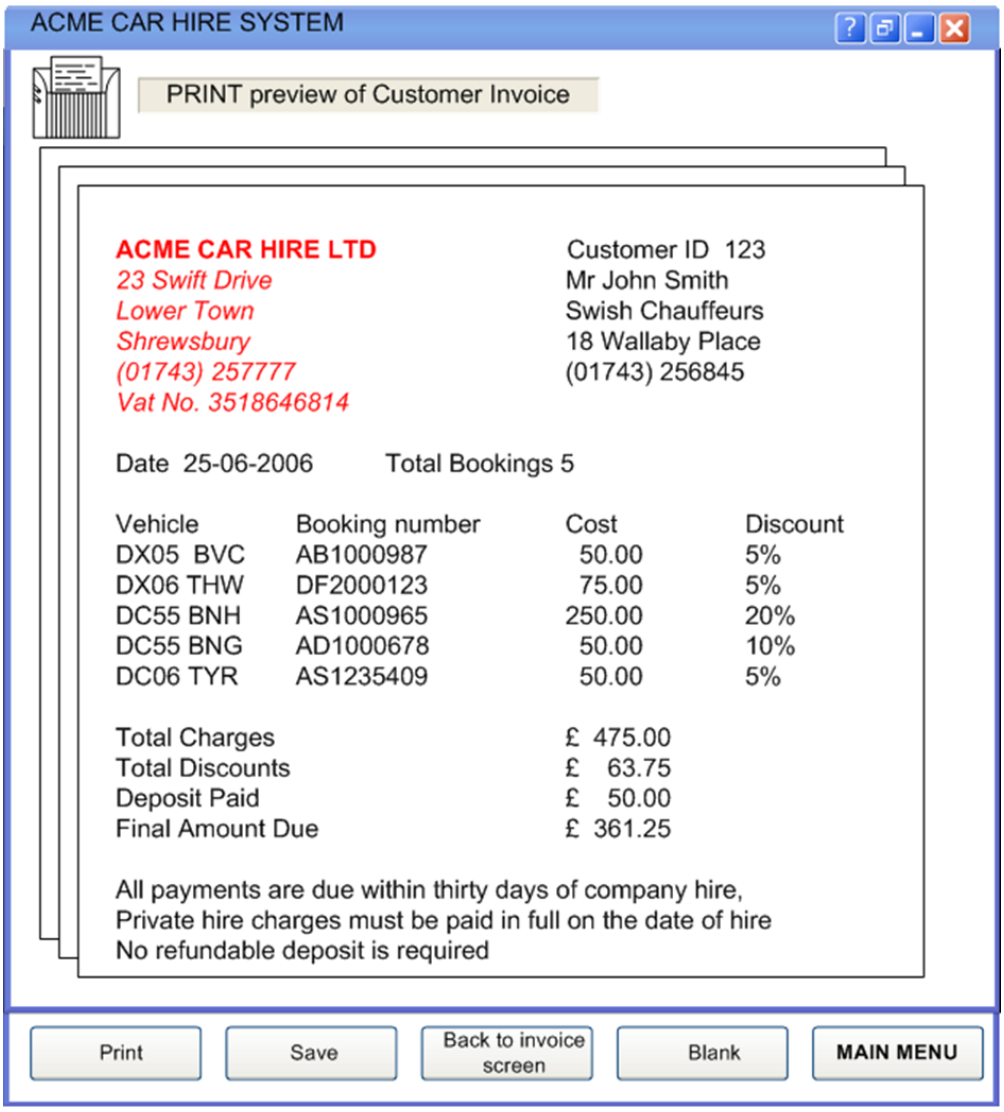
If the user presses either the ‘View Customer Details’ or ‘Update Customer Details’ buttons at the bottom of Screens 2 or 3, then this Screen 4 is displayed.

Note: the Measurer learns from using the system that if Screen 4 is displayed after pressing ‘View Customer Details’, then no fields may be updated. But if Screen 4 is displayed after pressing ‘Update Customer Details’, then many fields may be updated. Further, the Measurer learns that only certain authorized users may display Screen 4 in ‘Update’ mode and make changes to the displayed data. In contrast, any user may display Screen 4 in ‘View’ mode. The functionality that controls which users may view Screen 4 in ‘Update’ mode is outside the scope of the measurement.



2.2.5 Screen 5. Print Preview of Customer invoice

Following the ‘View’ or ‘Update’ of Customer Details from Screen 4, the user has the option to display a Print Preview of the current Invoice for this customer, as Screen 5.

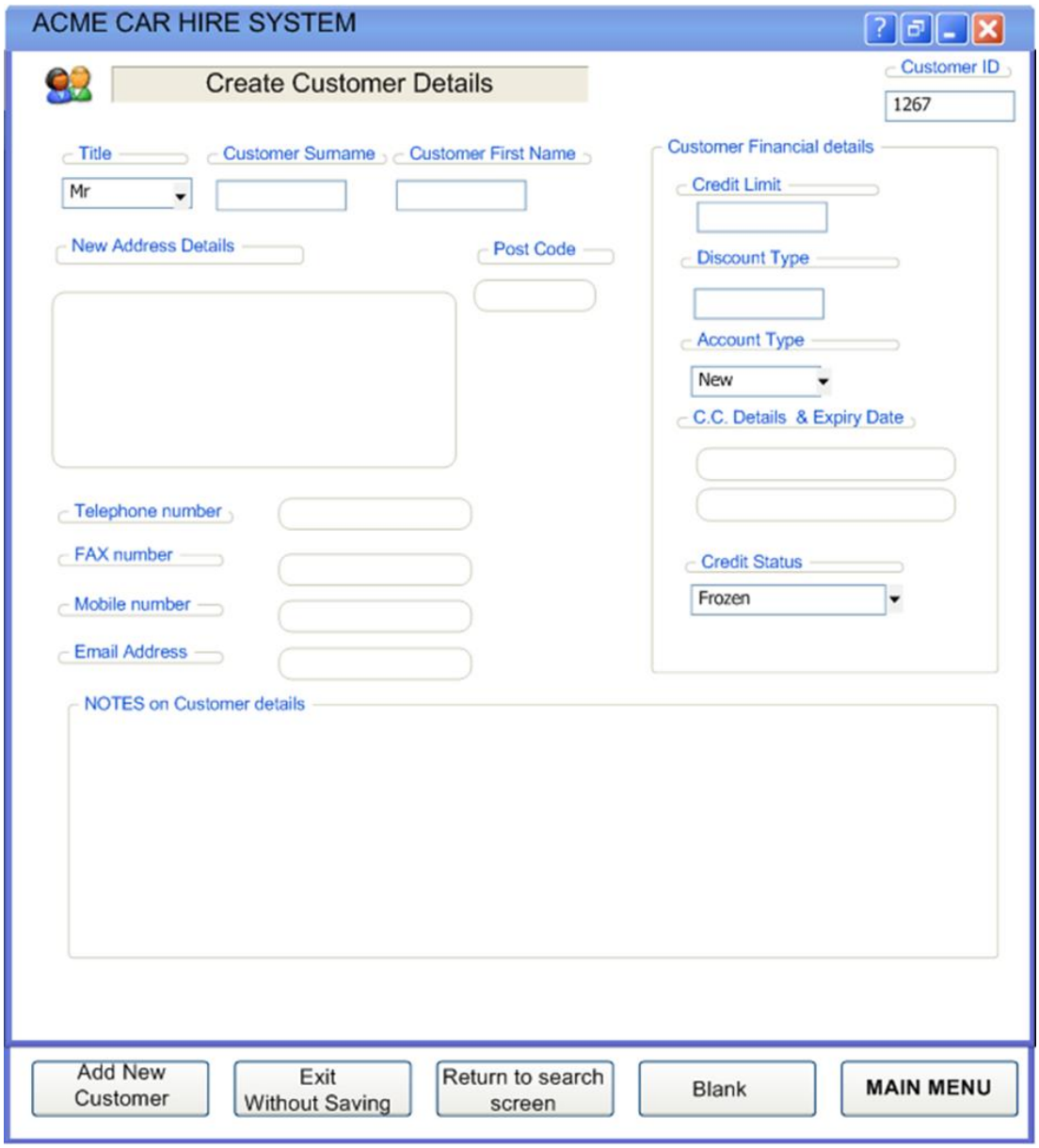


2.2.6 Screen 6. Error: Customer not found

If, when entering a customer name in Screen 1, no customer is found with that name, then Screen 6 is displayed. Another name, or a corrected name, may then be entered and searched.

2.2.7 Screen 7. Create Customer Details

As a result of Screen 6 showing the error message ‘Customer not found’, the user has an option at the bottom of Screen 6 to ‘Create New Customer’, i.e. to enter data about a new customer. This results in the display of the Customer data entry Screen 7, with a system-generated ID for the new customer. Pressing ‘Add New Customer’ at the bottom of this screen results in saving the new customer details.


ACME CAR HIRE SYSTEM
Create Customer Details
Customer ID
1267
Title
Customer Surname
Customer First Name
Mr
New Address Details
Post Code
Customer Financial details
Credit Limit
Discount Type
Account Type
New
C.C. Details & Expiry Date
Telephone number
FAX number
Mobile number
Email Address
Credit Status
Frozen
NOTES on Customer details
Add New Customer
Exit Without Saving
Return to search screen
Blank
MAIN MENU

3 MAPPING AND MEASUREMENT PHASES

As for practical reasons this chapter also contains the measurement result, the Mapping phase is combined with the Measurement phase.

N.B. The detail in the following analysis is provided for training purposes. In practice, it would not be necessary to document the analysis and the measurement in this detail. Section 3.7 shows a simple spreadsheet to document the measurement. (The context and the assumptions made in the measurement should also be described).

3.1 Navigation of the screens via the control buttons

The Measurer finds it helpful to draw a model of how the seven screens may be navigated, using names on the control buttons. Figure 3.1 shows these possible navigation paths relevant to the measurement scope, including the screen number(s) on which the control button appears.

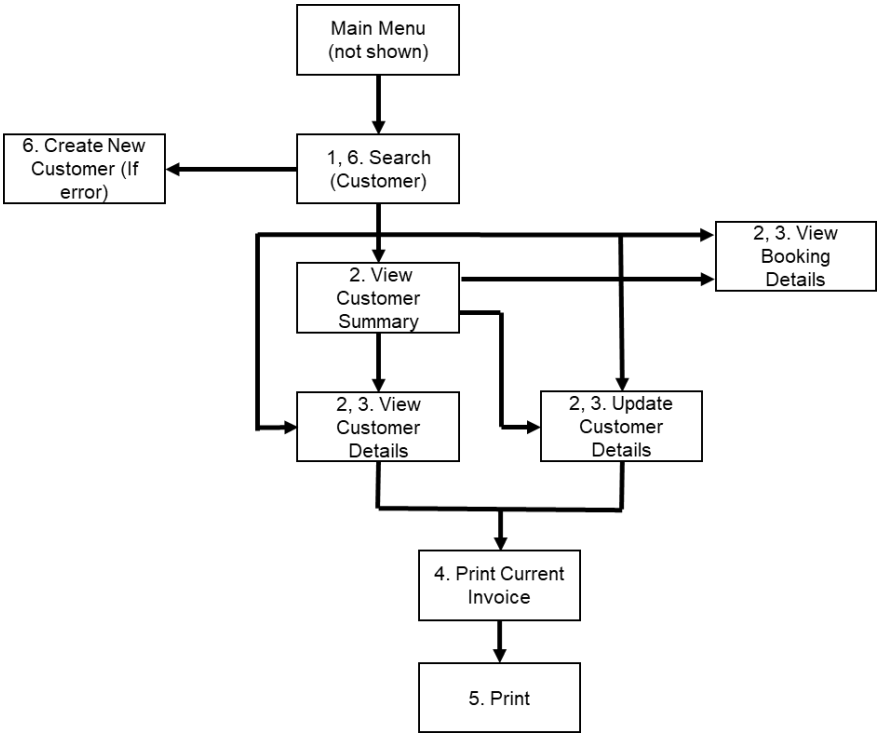


Figure 3.1- Navigation possibilities via the control buttons on the seven screens

Note: buttons such as ‘Save’ or ‘Main Menu’ or ‘Back to invoice screen’ that are not relevant to the navigation over the seven screens are not shown in Figure 3.1.

These navigational control buttons provide us with good evidence for identifying the functional processes, since each button selection requires a separate decision by the human functional user, which is a key criterion for distinguishing functional processes. However, to be sure, we should also check the mapping of the physical screen layouts to the (logical) functional processes.

3.2 Mapping the physical screen displays to the COSMIC concepts

We first note that the physical screens do not map one-to-one with functional processes.

- Screen 3 ('Customer Summary') shows the summary data about a selected customer, but also continues to display the output of Screen 2 ('List of Customers'). The design of the display is helpful to a user to check that he has really found the correct 'Mr John Smith'. But the two parts of the display show the output of two separate enquiry functional processes that serve different purposes:
  - *'Enquire if a customer of a given title and name exists in the system';*
  - *'Display summary data for all customers with the given title and name' (to enable the user to select the customer of interest).*

  When the user is certain he has identified the correct customer, this screen then provides the input to three other functional processes ('View Booking Details', 'View Customer Details' and 'Update Customer Details' as well as a button that enables the user to repeat the display of 'Customer Summary' for an 'Alternate' customer, i.e. to invoke another occurrence of the 'Display summary data' process.
- Screen 4 ('View/Update Customer Details') also serves two separate functional processes. It is the output screen resulting from pressing 'View Customer Details' on Screen 3 but also serves as the data input screen for the 'Update Customer Details' process. 'View' and 'Update' are clearly two separate functional processes.

Figure 2.1 which shows a possible sequence of using the seven physical screens does not therefore reveal the functional processes directly. In contrast Figure 3.1, which shows all the possible flows from pressing the navigation control buttons relevant to the scope of the measurement, is much more helpful for identifying the functional processes.

Note, however, that the control buttons at the bottom of each screen have various roles. Some examples:

- The 'Clear Details' button on Screen 1, and the 'Print' button on Screen 5 are pure 'Control Commands' in the COSMIC sense (See [2], section 3.5.10, or [3], section 4.4.1). They should be ignored.
- The 'View Customer Details' button on Screen 2 and the 'Back to Invoice Screen' on Screen 5 are sub-menu navigation commands (therefore also pure Control Commands). They should be ignored.
- The 'Print Current Invoice' button on Screen 4 is a Control Command that invokes the triggering Entry for a functional process that does not require any data to be physically entered. The Customer ID is needed in order to determine the Customer's Current Invoice. Pressing the print button initiates this Entry data movement that starts the functional process.
- The 'Print' button on Screen 5 is assumed to be a Control Command that prints whatever is on the screen at the moment, or what is selected on the screen.

3.3 Identifying the functional processes

The triggering event(s) and functional processes are identified and listed in Table 1. The 'Screen' numbers in this table show the screen numbers for input and the output data, respectively.

Table 1: List of candidate triggering events and candidate functional processes

Screens	Triggering event: A User wants to:	Functional Process
1 (in), 2 or 6 (out)	Enquire if a customer of a given name exists in the system	List customers
3 (in) 3 (out)	Display data to enable the user to select the customer of interest (where more than one customer with the same given name exists)	View customer summary
2 or 3 (in) 4 (out)	View the detailed data for the selected customer	View customer details (Enquiry)
2 or 3 (in) 4 (out)	Display the detailed data for the selected customer before updating the customer details	View customer details (pre-Update)
4 in and out	Update the data for the selected customer	Update customer details
4 (in) 5 (out)	Display a Print Preview of the current Invoice for the selected customer	Display invoice print preview
7 (in)	Enter details for a new customer	Create new customer
2 or 3 (in) (No out screen)	Display bookings for a given customer	View customer bookings details

Identifying the functional processes should be straightforward to an experienced Measurer. Only two points are worth noting.

- The buttons on Screens 2 and 3 ‘View Customer Details’ and ‘Update Customer Details’ both result in the display of Screen 4. The difference is that Screen 4 does not allow any updates if the ‘View Customer Details’ button was pressed, but does allow updates if the ‘Update Customer Details’ button was pressed. As the two processes result from different user decisions (‘I only want to view’ or ‘I want to update’), there must be two functional processes (see [3], the last paragraph of section 4.1.5.)
- When the ‘Create New Customer’ button on Screen 6 is pressed, it appears that the data entry Screen 7 includes the Customer ID already provided by the system. This might be interpreted as implying an Exit data movement from a functional process, but that would be a mistake. The Customer ID is a field on Screen 7 for entering data about a new customer and is therefore an attribute of the ‘Customer details’ data group moved by the triggering Entry for this ‘Create’ functional process. It should make no difference to the size measurement whether the user happens to enter the customer ID or whether the software creates the ID and pre-fills the data-entry screen with the ID.

3.4 Identifying the objects of interest

There are three objects of interest in the system that are in the scope of the measurement. We can tell that they are different objects of interest because they have different frequencies of occurrence (See Figure 3.2, where the ‘crows-foot’ sign indicates one-to-many.)

‘Vehicle’ is shown in dotted outline because it appears from the data on the screens and an assumption (see the Note below) that the functional processes within the scope do not need to access vehicle records. So with this assumption, ‘vehicle’ is not an object ‘of interest’ for this measurement.

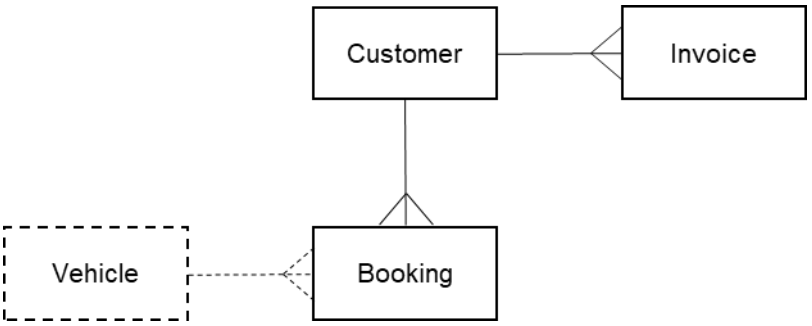


Figure 3.2 - The three objects of interest in the measurement scope

The objects of interest are listed below with their attributes that appear on the screens.

Customer: A person (or company?) registered in the ACME Car Hire system.

> **Key**: Customer ID. Other attributes: title, name, address, notes, credit limit, discount type, account type, credit card number, expiry date and status,

Invoice: A statement for a given customer at a given date of the amount to be paid for one or more bookings.

> **Keys**: (guess) Customer ID and Invoice date. Other attributes: customer name, address, total number of bookings, total charges, total discounts, deposit paid, final amount due, fixed text (payment terms).

Booking: A single Customer reservation for hire of a vehicle.

> **Key:** Booking number. Other attributes: booking date, vehicle ID, invoice amount, discount %.

Vehicle: A vehicle owned by ACME that may be rented.

> **Key:** Vehicle ID.

Note: The above interpretation of the data that appears on the screens makes an assumption about how a car hire system works, which might be obtained in practice via a system expert.

The assumption is that a ‘booking’ starts its life in the system with a status such as ‘reservation’. When the customer starts the hire period, a vehicle is assigned to the booking and the booking gets a status such as ‘rented’. At some time after the return of the vehicle, an invoice is issued for one or more bookings, at which point a booking gets the status ‘invoiced’ and data about the booking appears as an invoice-item. At this point in its life-cycle, therefore, booking’ is synonymous with ‘invoice-item’; they have a one-to-one relationship on the invoice. There is one underlying object of interest ‘booking’ for all statuses.

3.5 Identifying the data movements of the functional processes

The table below shows the functional processes identified in section 2.3.3, including all their movements of data groups (each of which describes an object of interest identified in section 2.3.4).

In the following, ‘Error messages’ means an error or confirmation message such as appears on Screen 6 for the ‘List customers’ functional process, or that it is reasonable to assume must exist for other functional processes.

Screens	Functional Process	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move-ment Type	CFP
1	List customers	User enters a customer name	Customer name	Customer	E	1
		System searches the customer file for the entered name	Customer details	Customer	R	1
2		System displays customer name(s) that match the entered name, and other customer data	Selected customer data	Customer	X	1
6		Display error message if no customer is found with that name	Error Messages	Errors	X	1
					Total size	**4**
2	View customer summary	User selects a customer from the displayed list by pressing a radio button	Customer ID	Customer	E	1
		System searches the customer file for the selected customer ID	Customer details	Customer	R	1
		System searches to see if the customer has any bookings	Booking details	Booking	R	1
3		System displays customer summary data	Customer summary data	Customer	X	1
3		System displays current booking(s) for the customer	Current booking	Booking	X	1
					Total size	**5**
2 or 3	View customer details (Enquiry)	User presses ‘View customer details’ button	Customer ID	Customer	E	1
		System retrieves details for the selected customer ID	Customer details	Customer	R	1
4		System displays details for the selected customer ID	Customer details	Customer	X	1
					Total size	**3**

2 or 3	View customer details (pre-Update)	User presses 'Update customer details' button	Customer ID	Customer	E	1
		System retrieves details for the selected customer ID	Customer details	Customer	R	1
4		System displays details for the selected customer ID	Customer details	Customer	X	1
					Total size	**3**
4	Update customer details	Having pressed 'Update Customer Details' on Screen 2 or 3, the user now enters changed customer data on Screen 4 and presses 'Save details'	Customer details	Customer	E	1
		System updates customer record	Customer details	Customer	W	1
4		(Assumption). There will be error messages from validation failures	Error Messages	Errors	X	1
					Total size	**3**
4	Display invoice print preview	User presses 'Print Current Invoice' for the customer whose details are displayed	Customer ID	Customer	E	1
		System searches for the selected customer in order to find any bookings that have status 'rented' (so not yet 'invoiced')	Customer details	Customer	R	1
		System retrieves bookings for the customer that must be invoiced	Booking details	Booking	R	1
5		System displays invoice header and account summary details	Customer Invoice data	Invoice	X	1
5		System displays data for each invoiced booking	Booking invoice data	Booking	X	1
5		(Assumption). An error message will be issued if no invoice is due	Error Messages	Errors	X	1
					Total size	**6**
7	Add new customer	System enters data for new customer and presses 'Add new customer' button	Customer details	Customer	E	1
		System creates new customer	Customer	Customer	W	1

		record	details			
		(Assumption). There will be error messages from validation failures	Error Messages	Errors	X	1
Total size						**3**
2 or 3	View customer booking details	User presses 'View Booking Details' for the customer whose data is displayed	Customer ID	Customer	E	1
		System searches for the selected customer in order to find any bookings	Customer details	Customer	R	1
		System retrieves bookings for the customer	Booking details	Booking	R	1
Not shown		(Assumption) some customer data must be displayed	Customer summary data	Customer	X	1
Not shown		System displays booking details	Booking details	Booking	X	1
		(Assumption). An error message will be issued if no bookings exist	Error Messages	Errors	X	1
					Total size	**6**

3.6 Exploring hidden functionality - Exercises

The above measurement results are based on the data shown on the physical screens and some observations and assumptions that have been documented.

But there is almost certainly more 'hidden functionality' that is not immediately obvious from the physical Screens. It would be good practice for the Measurer to ask a system expert about possible hidden functionality, such as the following. Possible questions are presented as exercises for the reader.

Question 1.

We have assumed that on Screen 3, if the user first presses a radio button for another of the 'Mr John Smiths' and then presses the 'View Alternate Customer' button, this means that the details of the other John Smith are displayed on the same screen. This sequence implies that the user is invoking another occurrence of the 'View customer summary' functional process. Instead, suppose that pressing this button takes the user back to Screen 2 to enter another name, would this change the size measurement?

Answer.

There would be no effect on the size. The button 'View Alternate Customer' is for navigation purposes. We do not need to know the outcome of pressing the 'View Alternate Customer' button for the measurement.

Question 2.

On Screen 4 in 'Update Customer Details' mode, there is an option to 'Add New Note'. We have assumed in the analysis above that there is one attribute of a customer called 'Notes'. But it is possible that a separate Note record is created (with the date as part of its key) every time a new Note is written for a given customer. In this case there would be a one-to-many relationship between the objects of interest 'Customer' and 'Customer Note' and the latter should also appear as an object of interest on Figure 3.1. If this latter assumption is correct, what would be the effect on the sizes measured?

Answer.

The two 'View Customer Details' functional processes would have one additional Read to retrieve, and an Exit to display, one or more 'Customer Notes'. Further, both the 'Add new customer' and the 'Update customer details' functional processes would have an additional Entry and a Write to enter and store a new 'Customer Note'. The total increase in size with this assumption would be eight CFP.

Question 3.

In the real-world, customer attributes such as his/her address, credit card number, etc., change over time.

Normal business practice suggests that it would be wise for the ACME Company to create 'history' records for the replaced data values of certain customer attributes (rather than just over-writing the out-of-date values). Customer attributes that may change and for which it is worth keeping history records include:

- Customer address
- Customer account (because 'credit limit' and 'account status' might change)
- Credit card

If each of these 'things' becomes the subject of a history record, with different frequencies of occurrence, it would now be an object of interest rather than a customer attribute. (Example: the object of interest 'Customer' would have a one-to-many relationship with the object of interest 'customer address'.)

What would be the effect on the measured sizes if history records must be maintained for these three 'things'?

Answer

For these three 'things' that have become objects of interest, we might expect the following

- Assuming the 'Customer details' data group contains the current values and that the 'View' functional processes are required to show only the current values (and not the history of the values), there would be no effect on the size of these processes.
- However, there would almost certainly be a need for the user to display the history of these three objects of interest. This implies three additional 'Display history' functional processes having one Entry, one Read and two Exits (one for the data value plus an error message for when there is no history record). The total increase in size is then 3 x 4 = 12 CFP.
- The 'Update Customer Details' functional process would have three additional Writes to store a history record for each of the three objects of interest.

Question 4.

We do not know how the Cost and the Discount shown on Screen 5, 'Print Preview of Customer Invoice', are calculated. 'Cost' presumably varies with 'Vehicle Class' and the duration of the hire. 'Discount' varies with the booking in rather complex ways that are not clear. We also see from the 'Customer details' on Screen 4 that 'Discount Type' is a variable

for a customer. However, the actual discount for an individual booking must vary with one or more other factors, since all invoice items are for the same one customer. (Does the discount also vary over time, with hire duration, with vehicle-class, or whatever? We don't know.)

Assume that, as well as varying with customer, the discount applicable to a booking varies with 'Vehicle-class' and that 'Vehicle-class' is an attribute of the object of interest 'Vehicle'. Assume also that the discount is applied at the time the invoice is calculated. With these assumptions, what would be the effect on the size of the 'Display invoice print preview' functional process?

Answer.

First note that the 'Vehicle ID' is shown as one of the booking attributes. The 'Display invoice print preview' functional process would therefore need an additional Read of the object of interest 'Vehicle' associated with the booking to obtain its 'Vehicle-class'. As the functional process already has the 'Discount type' from the 'Customer details' record, the applicable discount can be obtained. The size of this functional process would increase by one CFP.

Question 5

The 'Print Preview' screen shows that the Customer ID, name and address are output. Why is no Exit measured for this data describing the object of interest 'Customer? Also, why is no Exit measured for the output of the three lines of fixed text describing invoice payment terms?

Answer

The three attributes Customer ID, name and address are all attributes describing the object of interest 'invoice'. If we wish to be really precise in attribute naming, they should really be called 'ID of the customer invoiced', 'name of the customer invoiced' and 'address of the customer invoiced' respectively. The purpose of this process is to output data about an invoice, not about a customer. Therefore, the fixed text is also data about the invoice (see the attributes of the object of interest 'Invoice')..

If pieces of text such as this were stored by the Car Hire system as variables that could be edited by an administrator, then they still remain attributes of the invoice. In the case of an administrator as an editor there will be additional functional processes to edit the fixed text, with the administrator as functional user and 'fixed text' as its object of interest .

Question 6. (An exercise in measuring a Change Request)

Suppose there is a requirement to add the customer's 'Driving License No.' to the Customer details record. For all functional processes within the scope of the measurement, what would be the size of this change and the increase in functional size (if any) of the software?

Answer.

An extra attribute would need to be included in the Entry and Write of the 'Update customer details' and 'Create new customer' functional processes and on the Read and Exit of both 'View customer details' functional processes. In total the size of the change requirement is 8 CFP, because eight data movements must be changed.

In addition it is possible that the error messages for the Update and Create functional processes would need to be changed to handle validation failures of the customer's 'Driving License No'. This would add a further 2 CFP to the size of the required change.

The total size of the Change Request would therefore be 10 CFP.

As no new data movements are added, and none are deleted, the size of the ACME Car Hire software measured in CFP would not be changed as a result of this Change Request.

3.7 ACME Car Hire System. Summary COSMIC size measurement.

Table 3 below summarizes the results of the analysis given in section 3.5. It shows the functional processes and their sizes for which direct evidence is available from the screens, or where reasonable assumptions have been made about the existence of data movements. (We have assumed a total of 4 CFP for which there is no direct evidence, of which 3 CFP are for assumptions that error messages must be present).

Table 3: List of the functional processes and their sizes

	Data Group Names										Nos. of Data Movements				
Acme Car Hire Functional Procesess	Customer name	Customer details	Selected customer data	Customer ID	Customer summary data	Customer invoice data	Booking detailss	Current booking	Booking invoice data	Error/confirmation message	Entries	Exits	Reads	Writes	**Total**
List customers	E	R	X							X	1	2	1		**4**
View customer summary		R		E	X		R	X			1	2	2		**5**
View cust. details (Enquiry)		R, X		E							1	1	1		**3**
View cust. details (pre Update)		R, X		E							1	1	1		**3**
Update customer details		E, W								X	1	1		1	**3**
Display invoice print preview		R		E		X	R		X	X	1	3	2		**6**
Add new customer		E, W								X	1	1		1	**3**
View customer booking details		R		E	X		R, X			X	1	3	2		**6**
					Totals for Acme System:						**8**	**14**	**9**	**2**	**33**

Table 4 shows the possible sizes to be added to account for the 'hidden functionality' identified in Questions 2 – 4, as discussed in section 3.6.

Table 4: 'Hidden functionality' and the possible effect on sizes of Functional Processes

Hidden Function	**Functional processes affected**	**Additional Data Movements**				
		Entries	Exits	Reads	Writes	Total
Q2: Customer to have multiple Notes	View customer details (Enquiry)		1	1		2
	View customer details (pre-Update)		1	1		2
	Add new customer	1			1	2
	Update customer details	1			1	2
Q3: Customer to have multiple history	Update customer details				3	3
records for address, customer account	View customer address history	1	2	1		4
and credit card. The three new 'View'	View customer account history	1	2	1		4
FP's are new.	View customer credit card history	1	2	1		4
Q4: Calculation of invoice discount to depend on Vehicle-class	Display invoice print preview			1		1
	Totals	**5**	**8**	**6**	**5**	**24**

The very important lesson from the Measurer asking questions to a system expert about how the ACME Car Hire system works in practice is that it may lead to discovering much more functionality (and thus extra size) than was first apparent from examining only the data shown on the physical screens.

In this case, if the assumptions about how to deal with hidden functionality are correct, the total size at 57 CFP is 73% greater than was first apparent.

REFERENCES

All COSMIC documents are available for free download from www.cosmic-sizing.org.

[1] Introduction to the COSMIC method of measuring software.

[2] The COSMIC Functional Size Measurement Method Measurement Manual. (The COSMIC Implementation Guide for ISO/IEC 19761.

[3] Guideline for Sizing Business Application Software.

ACKNOWLEDGEMENTS

This case study was kindly provided by Capgemini UK. The measurement of the COSMIC Function Point sizes was made in collaboration with the Capgemini reviewers shown below.

Version 1.0.1 reviewers		
Mike Eagles Capgemini UK	Paul Hope Capgemini UK	Arlan Lesterhuis MPC COSMIC The Netherlands
Bruce Reynolds Telecote Research USA	Francisco Valdés Souto, Spingere, Mexico	

Version 1.0 reviewers		
Mike Downing Capgemini UK	Mike Eagles Capgemini UK	Paul Hope Capgemini UK
Phil James Capgemini UK	Arlan Lesterhuis MPC COSMIC The Netherlands	Bruce Reynolds Telecote Research USA
Charles Symons COSMIC UK		

VERSION CONTROL

The following table gives the history of the versions of this document.

DATE	**REVIEWER(S)**	**Modifications / Additions**
April 2017	COSMIC Measurement Practices Committee	First version 1.0 issued
August 2018	COSMIC Measurement Practices Committee	Version 1.0.1 issued with some clarifications.

CHANGES IN V1.0.1 FROM V1.0 OF THIS GUIDELINE

Note. The nature of a change is indicated by

- 'Method' when a definition or rule of the COSMIC method has been changed
- 'Editorial' when the description of the guidance was changed to improve ease of understanding.
- 'Correction' when an error in the previous version v1.0 of this Guideline has been corrected.

References in version 1.0.1	Nature of change	Comment
3.2	Editorial	In the sentence 'The 'Print Current Invoice' button on Screen 4…' the different roles of the Control Command and the triggering Entry clarified.
3.2	Editorial	Explained why the 'Print' button on Screen 5 doesn't lead to another functional process.
3.3	Editorial	The sentence 'Every functional process must start with an Entry' removed, in view of the Note to the GSM in MM 1.3.2. Also, the phrase 'The Customer ID...' is an explanation to the word 'mistake' in the previous sentence, the deleted sentence disturbs this.
3.5 Question 5	Editorial	Answer clarified

CHANGE REQUESTS, COMMENTS, QUESTIONS

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

The COSMIC Functional Size Measurement Method
Version 4.0.2

Automatic Line Switching Case Study

Version 1.1
November 2018

Table of Contents

1 AUTOMATIC LINE SWITCHING REQUIREMENTS

1.1 Background of the requirements

The Automatic Line Switching (ALS) system is documented in the ISO technical report: ISO/IEC TR 14143-4 (Version 2000). This ISO document provides various sets of functional requirements, described usually in a textual formal. The purpose of this ISO document is to provide researchers and practitioners with sets of requirements to be used as publicly available documents for measuring the functional size of software. The Automatic Line Switching system used in this real-time case study corresponds to set RUR B.8 of this ISO document.

1.2 Context

(N.B. in the text below reproduced from ISO/IEC TR 14143-4:2000, some words have been added to help the reader to understand the text quickly. All such added words are in italics.)

The functional requirements of the software of the ALS system below describe the control of two lines - a Work Line and a Backup Line - provided for a communication channel. If the Work Line degrades or fails the Backup Line is used instead. The decision to switch from one Line to another is made either automatically by the ALS software or by a technician at the receiving side. The switch to the Backup Line will remain in effect even after the Work Line becomes fully operational.

A standard redundancy method is used to continuously check the accuracy of the transmissions. Error correction, however, is not part of the software and is carried out externally. The error rate of a line signal will determine if the quality of a line is normal, degraded, or has failed. Since the lines are monitored continuously *by their Quality Level Change Monitors (QLCMs)* a complete loss of signal will initially be detected as a degraded quality. The expected response to a degraded or failed signal on the working line is to automatically switch to the backup line, if that line is in better condition.

1.3 External commands

Technicians are provided with a set of commands *for the Work Line and the Backup Line* to change the configuration of the channel *via the Channel Configuration Software on a PC*:

remove line:	the line is taken out of service,
restore line:	the line is placed in service,
forced switch:	the line is selected for communication if it is not out of service, and
conditional switch:	the line is selected for communication, if it is not out of service and at least at the same quality as the line currently selected.

1.4 Properties

a) The quality of a line has four levels:
   1. "normal" ($<10^{-9}$ error rate),
   2. "degraded" (10^{-5} to 10^{-9} error rate),
   3. "failed" ($>10^{-5}$ error rate, or no signal), and
   4. "out of service".

b) One and only one of the two lines is selected for communication at any given time.

The next four requirements describe what must happen to the quality of a line when a command is entered by a technician from the Channel Configuration Software on the PC.

c) When a “remove Work Line” event *(command)* occurs:
if the Work Line is not out of service then it goes out of service,
otherwise the Work Line remains out of service.
(The behaviour on a “remove Backup Line” event is analogous).

d) When a “restore Work Line” event *(command)* occurs:
if the Work Line is out of service then it becomes normal
otherwise the level of the Work Line does not change.
(The behaviour on a “restore Backup Line” event is analogous.)

e) When a “forced switch to Work Line” event *(command)* occurs:
if the Work Line is not out of service then it becomes the selected line,
otherwise the selection of the lines remains unchanged.
(The behaviour on a “forced switch to Backup Line” event is analogous.)

f) When a “conditional switch to Work Line” event *(command)* occurs:
if the Work Line is not out of service and is not of poorer quality than the Backup Line then the Work Line becomes the selected line,
otherwise, the selection of the lines remains unchanged.
(The behaviour on a “conditional switch to Backup Line” event is analogous.)

The next three requirements describe what must happen to the current quality level of a Line when its QLCM detects certain conditions that may result in the need to change the Line’s quality level.

g) When a “Work Line degraded” event occurs:
if the quality of the Work Line is “normal” then it will change to “degraded”,
otherwise, the quality of the Work Line remains unchanged.
(The behaviour on a “Backup Line degraded” event is analogous.)

h) When a “Work Line failed” event occurs:
if the quality of the Work Line is “degraded” then it will change to “failed”,
otherwise, the quality of the Work Line remains unchanged.
(The behaviour on a “Backup Line failed” event is analogous.)

i) When a “Work Line cleared” *(i.e. quality is ‘normal’)* event occurs:
if the quality of the Work Line is “degraded” or “failed” then it is set to “normal”,
otherwise, the quality of the Work Line remains unchanged.
(The behaviour on a “Backup Line cleared” event is analogous.)

j) If a “remove line”, “restore line”, "line degraded”, “line failed”, or “line cleared” event occurs *(i.e. one of the events of requirements c), d), g), h) and i) occurs)*, and the currently un-selected line becomes of a higher quality than the selected line, then the selection will be switched, triggered by the ALS software.

k) Removing, restoring, deterioration, or clearing of a line does not affect the quality of the other line.

l) Switching the selected line does not affect the quality of either line.

m) It is forbidden to switch to a line that is out of service, except when both lines are out of service.

n) The selected line will only change as a result of one of the following:
   1. the selection is changed with a switch command *(provided, in the case of a conditional switch, that the quality of the other line is not out-of-service or of poorer quality than that of the currently selected line)*,
   2. the currently selected line deteriorates to a quality inferior to the other line,
   3. the currently selected line goes out of service, or
   4. the currently un-selected line clears (or is restored) to a quality better than the selected line.

Note. There is overlap between requirements j) and n).

On first reading, it may appear from the requirements that each line needs two status indicators – its current 'quality' (from a)) and whether or not it is currently 'selected' to be used for communication (from b)). But with one exception, any decision by the system to switch depends only on the relative quality of the two lines – the current selection status does not matter in the system's decision to switch. The exception is a 'forced switch' which is determined wholly by the Technician, ignoring the relative quality of the two lines.

1.5 Assumptions

The following assumptions are made for the measurement solution proposed in this case study:

1 The analysis of the error rate (by the standard redundancy method used to continuously check the accuracy of the transmission) is performed outside of the ALS software to be measured, by the two (hardware-software) functions, the QLCMs, which monitor the error rate as input. When a change of error rate is detected the QLCM issues one of the three following signals for each type of line: line degraded, line failed, line cleared.

2 Similarly, the physical switching between lines is assumed to be made by another (hardware/software) function, the Switch Device, triggered by a signal from the ALS software.

3 The line quality of each line (see 1.4 a) must be stored every time it is updated (changed) because the ALS software needs to know the existing status when a new status is signaled, in order to decide what to do.

4 The quality of both lines must be displayed on the Channel Configuration Panel (the display of the Channel Configuration Software on the PC), in order to enable the Technician to decide what to do.

2 MEASUREMENT STRATEGY

2.1 Measurement purpose

The measurement purpose is to measure all of the Functional User Requirements (FUR) of the ALS software documented in the set of Reference User Requirements selected for this case study, using the COSMIC functional sizing method. FUR are derived from the functional requirements of the software which, in turn, are a subset of the Reference User Requirements (RUR).

2.2 Measurement scope

The measurement scope is all of the FUR. The measurement scope is therefore a subset of the ALS system requirements documented in this ISO case study, that is, only those related to software, and not those related to the hardware.

There is a single software layer for this set of requirements

2.3 Identification of functional users

The functional users that interact directly with this ALS software are the following devices:

a) Hardware-software devices sending information to the software:
- *Channel Configuration Panel software on the PC: issues the commands entered by the Technician*
- *Quality Level Change Monitors (for the Work/Backup Lines)*

b) Hardware-software devices receiving information from the software:
- *Switch Device, to switch to the other (currently non-selected) Line*
- *Channel Configuration Panel software on the PC, displays status of each Line.*

Note that there are two *occurrences* of the Quality Level Change Monitor, one for the Work Line and one for the Backup Line. As both are subject to the same FUR (namely: 'if a change in the quality level of the line is detected, inform the Automatic Line Switching software'), identify only one functional user *type* 'Quality Level Change Monitor', as in Figure 1.

From the requirements, as written, there are no human users interacting directly with the software being measured (the human interaction is carried out through the Channel Configuration Panel software), nor is there any other software interacting with this software being measured.

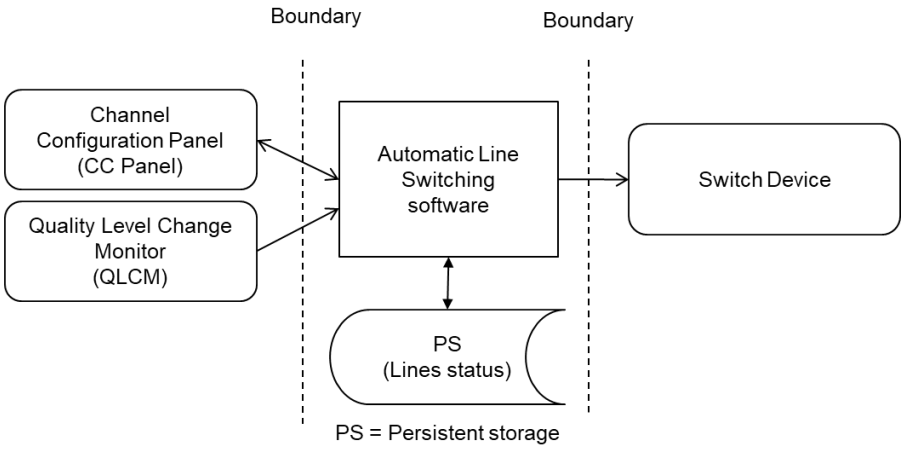


Figure 1 – Context diagram of the Automatic Line Switching software

2.4 Other measurement strategy parameters

Level of granularity. The software requirements of the Automatic Line Switching system are at the 'level of granularity' of a functional process because the functional users are individual hardware devices or software (not groups of these) and a single event occurs that the software must respond to (not groups of events).

Level of Decomposition: Not applicable

Persistent storage. As there is no requirement stating that the storage be accessible via another piece of software, from the COSMIC measurement perspective the data (the status of the lines) resides on persistent storage within the software boundary.

3 THE MAPPING AND MEASUREMENT PHASES

3.1 Identification of the triggering events

From the documented requirements the following triggering events are identified:

1 Remove Work Line
2 Restore Work Line
3 Forced switch to Work Line
4 Conditional switch to Work Line
5 Signal of Work Line failed
6 Signal of Work Line degraded
7 Signal of Work Line cleared

+ all similar triggering events for the Backup Line, hence 14 triggering event *occurrences*.
However,

- The triggering events for the Work Line and the Backup Line of the Remove Command are analogous, as they share the same FUR of the service to be performed. This service *type* could be called 'Remove Line', where 'Line' stands for 'Work Line' or 'Backup Line'). The same for the other three Commands.
- The three 'Signal of Line Quality' triggering events 5, 6 and 7 require different services as described by points g), h) and i) in section 1.4), i.e. lead to services with different FUR from those triggered by a technician. But again they are the same for the Work Line and for the Back-up line. Identify three triggering event types.

Total = 7 triggering event types.

3.2 Identification of functional processes

As the requirements for the Work line and the Backup line share the same FUR according to the properties of Chapter 2 for commands from technicians, one functional process type must be identified that accounts for handling both lines. The first four triggering events require different handling (FUR), hence identify four different functional processes.
With the line quality data from the QLCM the software produces the three 'Line Quality Commands', each requiring different handling, leading to three different functional processes. In total there are 7 functional processes:

1 Remove Line: the Line is taken out of service
2 Restore Line: the status of the Line is set to 'Normal'.
3 Forced Line switch: the Line is selected for communication as long as it is in service
4 Conditional Line switch: the Line is selected for communication, as long as it is in service and at least at the same quality as the line currently selected
5 Handle QLCM Line failed signal,
6 Handle QLCM Line degraded signal, and
7 Handle QLCM Line cleared signal.

Each ‘Handle...’ functional process (5, 6 and 7) transforms its QLCM signal to a command to the Switch Device, which makes the physical switching between the lines.

3.3 Identification of objects of interest

All the data moved into and out of the ALS software and to/from persistent storage describes the same one object of interest ‘Line’. The data groups moved are

Data group name	Attributes	From/to
Line Command	LineID, Command (Remove, etc.)	From CC Panel to ALS
Line Status	LineID, Quality	From QLCM to ALS, From ALS to CC Panel and to persistent storage From persistent storage to ALS
Line Switch	LineID, Selected Y/N	From ALS to Switch Device

3.4 The functional processes and their data movements

This section describes each functional process with its 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: Remove line **Triggering Event: Remove line needed**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
CC Panel	Enter Remove line command	Line Command	E	1
	Read Work/Back-up Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Remove line command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: Restore line **Triggering Event: Restore line needed**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
CC Panel	Enter Restore line command	Line Command	E	1
	Read Work/Back-up Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Restore line command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: **Forced Line switch** **Triggering Event:** **Forced switch needed**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
CC Panel	Enter Forced switch command	Line Command	E	1
	Read Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Forced switch command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: **Conditional Line switch** **Triggering Event:** **Conditional switch needed**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
CC Panel	Enter Conditional switch command	Line Command	E	1
	Read Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Conditional switch command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: **Handle QLCM Line failed signal** **Triggering Event:** **QLCM Line failed signal**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
QLCM	Enter QLMC Line failed signal	Line Status	E	1
	Read Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Line switch command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: **Handle QLCM Line degraded signal** **Triggering Event:** **QLCM Line degraded signal**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
QLCM	Enter QLMC Line degraded signal	Line Status	E	1
	Read Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Line switch command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

Functional Process: **Handle QLCM Line cleared signal** **Triggering Event:** **QLCM signal**				
Functional user	**Data movement Description**	**Data Group moved**	**Data Mvt. Type**	**CFP**
QLCM	Enter QLMC Line cleared signal	Line Status	E	1
	Read Line status	Line Status	R	1
	Write Line status	Line Status	W	1
Switch device	Exit Line switch command	Line Switch	X	1
CC Panel	Display Line status	Line Status	X	1
				Total size: 5 CFP

The total functional size of the Automatic Line Switching software is 7 * 5 CFP = 35 CFP.

A good insight in the movement of data within a functional process gives a Message Sequence Diagram. Figure 2 shows the message 'traffic' within the 'Remove line' functional process:

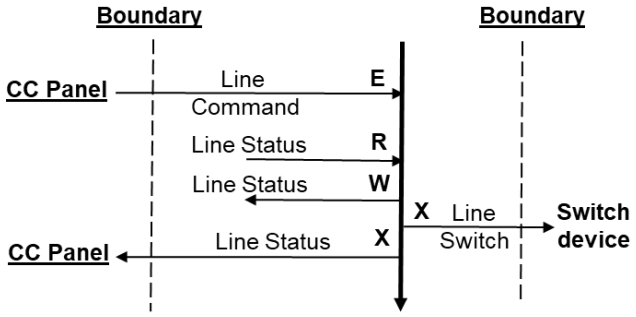


Figure 2 – Message Sequence Diagram of Remove line

3.5 Discussion of the identification of the functional processes

There are reasons for arguing that there are only two functional processes, namely one that fulfills all Technician commands and one that fulfills all changes of line status reported by a QLCM. Supporting this argument are the facts that the size, functional users and data

movements of the first four processes are identical; they differ only in the *values* of their triggering Entries (the Technician commands). Similarly the size, functional users and data movements of the last three processes are identical; they also differ only in the values of the line quality detected by a QLCM. With this interpretation, the different processing paths simply result from the different input values.

However, our reason for concluding that there are seven distinct functional processes is that they result from seven different triggering events occurring in the external world. The technician makes separate decisions for the four cases dependent on his/her knowledge of what's happening in the real world. Similarly, each QLCM responds to different events depending on its detecting different physical changes in the quality of the line it is monitoring.

3.6 Observations on the clarity of the documented requirements

Even though the documented requirements used for this case study are coming from an ISO technical report, there is no mention in this ISO report about the quality of these requirements. IEEE Std 830-1998 recommends that requirements meet the following quality criteria:
- Correct;
- Unambiguous;
- Complete;
- Consistent;
- Ranked for importance and/or stability;
- Verifiable;
- Modifiable;
- Traceable.

In the ISO technical report, there is no claim that their sets of documented requirements meet the quality criteria specified in IEEE 830. The following uncertainties or ambiguities have therefore been noted in the Requirements:

1) It is not clear if the status and the quality of the lines are recorded. Neither is it specified where or how they should be saved.
2) There is no indication of how messages are sent or received between processes or between users and processes.
3) There is no information about the exit from the process when it is finished.
4) The Technician doesn't know the currently selected line. This will be confusing to him when he wants to return back to the beginning of the session.
5) The Requirements do not ensure the security of having acceptable quality if one line is out of service.
6) The Requirements do not cover the case where a line of 'normal' quality receives a signal that it has 'failed', without first passing through the 'degraded' quality level.

3.6 Observations

During the measurement process, uncertainties and ambiguities about the documented requirements have been noted and it has been necessary to make assumptions about the functionality of the system that is allocated to software. It was also observed that these requirements do not meet all of the quality criteria listed in IEEE 830.

The reader is alerted to the fact that different interpretations of the system requirements and different assumptions to correct these requirements may result in different measurements of the software functional size.

APPENDIX - GENERAL INFORMATION

1 Acknowledgements

Version 1.1 Reviewers		
Alain Abran École de Technologie Supérieure, Université du Québec Canada	Arlan Lesterhuis* MPC The Netherlands	Bruce Reynolds Tecolote Research USA
Charles Symons United Kingdom	Francisco Valdés Souto SPINGERE Mexico	

* Editor of version 1.1 of this Case.

2 Version control

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

Date	Reviewers (s)	Modifications / Additions
2004-08-26	Adel Khelifi	First Draft
2004-2005	Alain Abran, Adel Khelifi, Charles Symons	Several revisions
2018	Alain Abran, Arlan Lesterhuis, Bruce Reynolds, Charles Symons, Francisco Valdes Souto	Complete revision

3 Main changes in v1.1 from v1.0 of this Guideline

Note. The nature of a change is indicated by

- 'Method' when a definition or rule of the COSMIC method has been changed
- 'Editorial' when the description of the guidance was changed to improve ease of understanding.
- 'Correction' when an error in the previous version v1.0 of this Guideline has been corrected.

References in version 1.1	Nature of change	Comment
General	-	The general style of the case changed to match the usual structure of the COSMIC documents. All general information (on acknowledgements, version control etc.) moved to the end of this case.
General	Editorial	The Automatic Line Switching (ALS) *system* is more clearly distinguished from the ALS *software*.
General	Correction	In v1.0, fourteen *occurrences* rather than *types* of functional processes were identified, leading to twice as many functional processes as are actually needed. This corrected and explanation added.
General	Editorial	The section 'Questions & Answers' omitted as it doesn't add much value
1.2	Editorial	In this section the paragraph of the RUR 'A standard redundancy method…' added, for ease of understanding.
1.3	Editorial	For clarity in the first paragraph 'Work Line' and 'Backup Line' added
2.1	Editorial	The text of v1.0 greatly shortened; irrelevant information omitted.
2.3	Correction	In points a) and b) '-software' added, as in 1.5 it reads 'hardware-software functions'
2.3	Editorial	The outdated term 'electro-mechanical devices' replaced by 'hardware-software devices'
2.3	Editorial	Figure 1 added, the context diagram of the Automatic Line Switching software
3.3	Correction	The object of interest and the data groups moved explicitly described
3.4	Editorial	The use case diagram omitted as it didn't add value. Two of the three Message Sequence Diagrams omitted: for illustration one suffices.
3.4	Correction	The data movements adapted to the data model of this case. Data movements added to account for the CC Panel display on the PC

References in version 1.1	Nature of change	Comment
3.5	Editorial	New section added: 'Discussion of the identification of the functional processes, which explains that 7 (rather than 2) functional processes must be identified.
3.6	Editorial	Both first paragraphs removed as they repeated other information

4 Change requests, comments, questions

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

The COSMIC Functional Size Measurement Method
Version 4.0.2

Course Registration ('C-REG') System Case Study

VERSION 2.0.1

August 2018

Table of Contents

1 ABOUT THIS CASE STUDY

1.1 Introduction

This Case Study presents the results of applying the COSMIC v4.0.2 Functional Size Measurement method (ISO/IEC 19761: 2017) to the Course Registration ('C-Reg') software system requirements as described in Chapters 2 - 4.

General information about the software application to be measured:

- Application domain: Business Application
- Application type: PC client interacts with server over an organization's network and the world-wide web.

1.2 How to use this C-Reg case study

This case study aims to be suitable for new users of the COSMIC method who have at least read the 'Introduction to the COSMIC method' document or have received some introductory training in the method, and have ideally read the Measurement Manual [1].

We also assume that the reader is generally familiar with statements of requirements for business application software. The requirements are written as plain text, so no knowledge is needed of any specific system or data analysis method to understand the requirements. To help the reader understand the case, we describe not just C-Reg requirements but also information that would normally be available later when the software design had started (such as menus and screen layouts).

After describing the C-Reg general requirements and context in section 1.3, the requirements and their respective analysis and measurements are dealt with in three Chapters, of concern to:

2 Registrars and Professors

3 Registrars and Students

4 Registrars

We recommend that readers of this case study should try to understand the requirements, analysis and measurement results from Chapter 2 fully before moving on to the requirements of Chapters 3 and 4.

The C-Reg case aims to support the following teaching points on the COSMIC method:

- Determining the various parameters of the Measurement Strategy phase, including establishing a context diagram showing the functional users of the software to be measured.
- Identifying functional processes within the requirements.
- Identifying the objects of interest in requirements.
- How the names of data attributes in requirements may be misleading. They must be analysed carefully to ensure the correct identification of data groups.
- Identifying the data movements of a functional process, including that alternative processing paths may or may not lead to identifying additional data movements.
- The need to recognise that an enquiry functional process may be identical to an 'enquire-before-update', or 'enquire-before-delete' process; such a process should be counted only once per application.
- Identifying the data movements involved in communications between software systems

- The COSMIC rules for ignoring ‘Control Commands’ and Menu selections and for measuring ‘Error/Confirmation messages’.
- Dealing with practical software requirements issues such as the need to check for any data dependencies before deleting an entity from a database.
- How to measure some aspects of Graphical User Interface features.

Note: Comments on the requirements intended to help the reader to understand the case study easily are shown in italics between square brackets.

NOTE

These requirements aim to be easy-to-understand, self-contained (but not complete) and defect-free so that they can be used to teach COSMIC functional size measurement.

(Almost all real-world requirements are incomplete and have ambiguities, etc., so need interpretation and assumptions in order to measure a functional size. These must be documented to trace the application of the measurement rules to the incomplete and ambiguous requirements. Requirements developed iteratively, e.g. using Agile methods, are never ‘perfect’ until the software is signed off by the user.)

If you are new to functional size measurement, be assured that experienced Measurers use the COSMIC method regularly to measure real-world requirements, even very early in a software project when requirements are evolving, assumptions have to be made about some details and approximate sizing may have to be used. But to get to this stage you need to understand the basic measurement principles and rules. Studying this C-Reg case will help you along this path.

1.3 System Requirements and Context

The system requirements below describe the functionality of the software to be developed by a project that will replace the existing Course Registration System (CRS) with an on-line system (“C-Reg”) that allows students and professors access through PC clients.

The current CRS system has been in use since a number of years and lacks the capacity to handle the student and course load projected for the future. In addition, the current system uses outdated mainframe technology, which only supports access through the clerks in the Registration Office. C-Reg will enable all professors and students to access the system, in addition to the clerks in the Registration Office, through PCs connected to the Wylie College computer network and through any personal computer connected through the Internet.

C-Reg will continue to interact with the Course Catalog System which maintains the list of courses and details of the courses that will be offered for the upcoming semester (known as ‘Course Offerings’). C-Reg will continue to interact with the student Billing System and the E-mail system – see Figure 1. How the data is loaded and stored in the Course Catalog system is outside the scope of this case study.

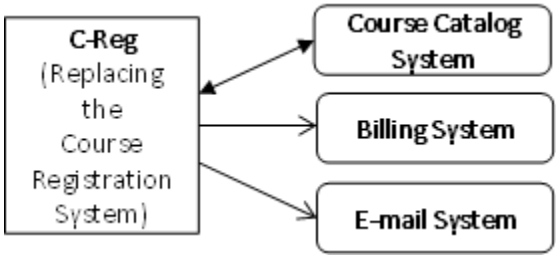


Figure 1 - C-Reg and interacting systems

All human users of C-Reg (i.e. Registrars, Professors and Students) access its functionality via a 'Main Form' menu (see Figure 2a). How the system controls security of access and the functionality available to each type of user is beyond the scope of these requirements.

[Note: in all of the following requirements, where 'he' is written, this can mean 'he' or 'she'. Similarly, 'his' can mean 'his' or 'her'.]

2 REGISTRARS AND PROFESSORS

2.1 Requirements for Registrars and for Professors

This section has two groups of requirements.

2.1.1 Maintain Professor data (by Registrars)

2.1.2 Maintain Course Offering commitments (by any Professor)

2.1.1 Maintain Professor Data

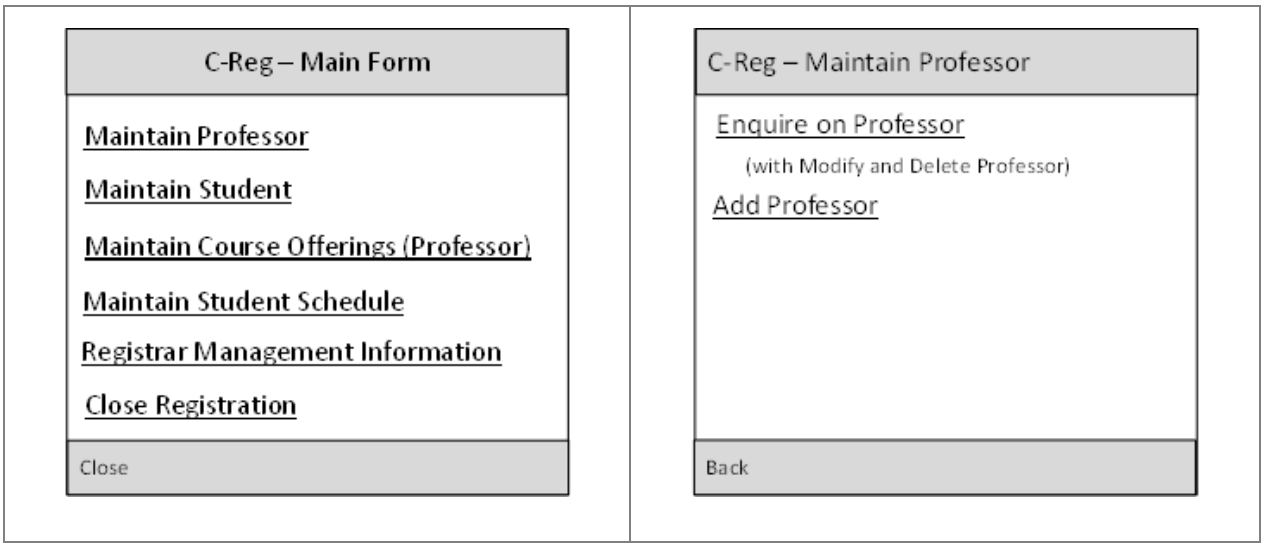
2.1.1.1 Brief Description

This group of requirements allows Registrars to maintain data about any Professor in C-Reg.

Each Professor is identified by a unique identification (or 'ID') in the form [surname, serial number], Example: 'Smith3'. ***[For simplicity in the requirements, we assume that when making any enquiries, a Registrar knows the ID of all Professors, and that each Professor knows his own unique ID.]***

C-Reg must enable a Registrar to perform any of the tasks 'Add Professor, 'Modify Professor', 'Delete Professor' or 'Enquire on Professor'.

[We assume C-Reg will be implemented with a menu system, as illustrated in Figure 2. Hence for this group of requirements, a Registrar must first select 'Maintain Professor' from the Main Form as in 2a) and then choose the appropriate sub-option as in 2b).]



(a) (b)

Figure 2 – a) Main Form and b) Example Sub-options

2.1.1.2 Add a Professor‘s details

a) When a Registrar wishes to enter data about a new Professor, he selects the sub-option "Add Professor."

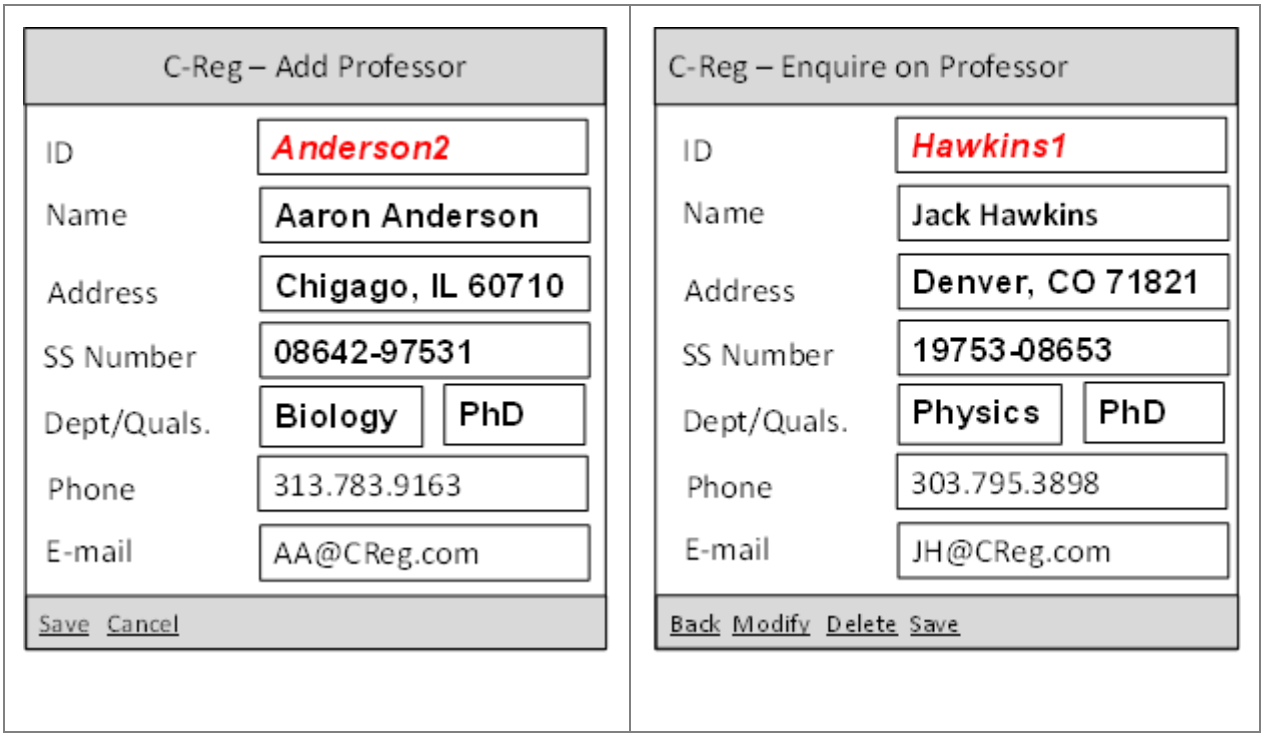
b) C-Reg displays a blank formatted screen for entry of Professor data.

c) The Registrar enters the following details for the Professor: ID, name and address, social security number, Department, qualifications and contact details and presses ‘Save’. (See Figure 3a) for an example screen showing the entered data.)

d) C-Reg validates the data to ensure the proper formats and checks whether a Professor of that ID already exists. If the entered data are valid, C-Reg creates a record for the new Professor and .

e) Alternatively, if data entered is not valid, C-Reg displays one or more error messages, for example: ‘Professor ID unknown’, Professor name already exists’, "Professor Data Invalid". The Registrar can then change or correct the data, or cancel the operation.

Steps a) to e) may be repeated for each Professor that the Registrar wishes to add to C-Reg.



(a) (b)

Figure 3 – (a) Add and (b) Enquiry screens

2.1.1.3 Enquire on a Professor’s details

a) When a Registrar wishes to enquire on the details of a Professor, he must first select the sub-option ‘Enquire on a Professor’ as in 2.1.1.2 and enter the Professor ID

b) C-Reg searches for a Professor with the specified ID and displays the Professor’s name and address and other details, as in Figure 3(b).

c) Alternatively, if a Professor with the specified ID is not found, C-Reg displays an error message, "Professor Not Found". The Registrar can then type in a different ID or cancel the operation.

2.1.1.4 Modify a Professor's details

a) If a Registrar wishes to modify the details of a Professor, he must first retrieve the Professor details as in 2.1.1.3 'Enquire on a Professor'
b) The Registrar presses 'Modify'.
c) The Registrar may then change one or more of the displayed Professor data items (except the Professor ID). When changes are complete, the Registrar presses "Save" and C-Reg updates the professor data.
d) Alternatively, if data entered is not valid, C-Reg displays an error message, "Professor Data Invalid". The Registrar can then correct the data or cancel the operation.

2.1.1.5 Delete a Professor's details

a) If a Registrar wishes to remove a Professor from C-Reg, he must first retrieve the Professor data as in 2.1.1.3 'Enquire on a Professor'
b) The Registrar presses 'Delete'.
c) C-Reg enquires on the Course Catalog whether the Professor has any Course Offerings that he has committed to teach. The Course Catalog replies to C-Reg with a 'yes/no' indication.
d) If the Professor has no Course Offering teaching commitments, C-Reg displays a message asking the Registrar to confirm the deletion.
e) If the Registrar selects 'yes', the Professor data is deleted from C-Reg.
f) Alternatively, if the Registrar selects 'no', the operation is cancelled.
g) If the Professor is committed to teach any Course Offerings, deletion is not allowed, C-Reg displays an error message and the Registrar must abandon the operation. *[Note: how the Registrar deals with this conflict is outside the scope of this case study.]*

2.1.2 Maintain a Professor's Course Offering commitments

2.1.2.1 Brief Description

The Course Catalog holds the dates, times and locations of all Courses that Wylie College offers to students in the upcoming semester (known as 'Course Offerings').

This group of requirements for the C-Reg system enables a Professor to enquire on Course Offerings that he may wish to teach, and to commit (i.e. add his ID) to teach a Course Offering, or to modify or to delete existing teaching commitments. The Course Catalog holds data on the qualifications needed to teach each Course.

The Course Catalog also holds the 'availability indicator' for each Course Offering. This indicator may have values:

- 'unavailable' (meaning so far no Professor has committed to teach the Course Offering),
- 'available' (meaning a Professor has committed to teach the Course Offering and Students may enrol)
- 'full' (meaning that a Course Offering is 'available' but Students may not enrol as the Course Offering is fully subscribed),
- 'cancelled' (for a Course Offering that was 'available' but has now been cancelled),
- 'closed' (meaning that Professors may no longer change their commitment to, and Students may no longer enrol in this Course Offering (see section 4.1.2 'Close Registration')

2.1.2.2 Enquire on Course Offerings

a) When a Professor wishes to enquire on the courses he may teach, he must first select ‘Maintain Course Offerings’ from the Main Form and then ’Enquire on Course Offerings (Professor)’ from the Sub-Menu.

b) The Professor enters his ID.

c) C-Reg obtains the Professor’s qualifications and Department, and sends these to the Course Catalog.

d) C-Reg obtains from the Course Catalog and displays (as shown in Figure 4) the list of Course Offerings for the Professor’s Department that he is qualified to teach and that are ‘unavailable’, i.e. no other Professor has committed to teach the Offering in the upcoming semester. The scheduled month and room for each Course Offering are also shown so that a Professor can select commitments that will not clash on date or location.

e) Alternatively, if there are no Course Offerings that the Professor may commit to teach in the upcoming semester. C-Reg will display an error message. The Professor acknowledges the message and abandons the operation.

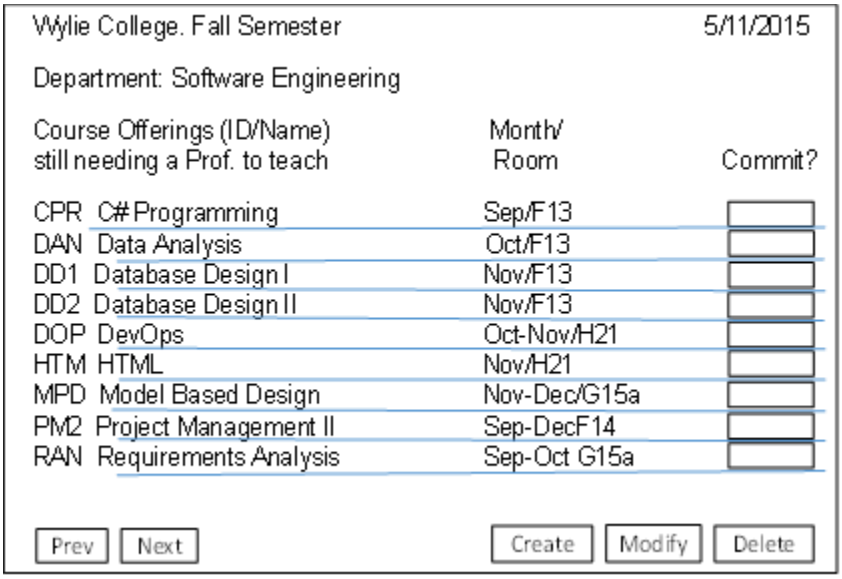


Figure 4 – Display resulting from Enquiry requirements 2.1.2.2

2.1.2.3 Create Course Offering commitments

a) When a Professor wishes to enter his first set of teaching commitments, he must first enquire on and display the Course Offerings as in 2.1.2.2, and then select a sub-option "Create Course Offering".

b) The Professor selects the Course Offerings from those displayed in step a) that he will commit to teach for the upcoming semester by adding his ID to the selections.

c) C-Reg returns each selected Course Offering the Professor has committed to teach to the Course Catalog.

d) The Course Catalog checks if the selected Course Offerings conflict on date or location (in case the Professor made a mistake) and returns a message to C-Reg containing:

- *the count of conflicting Course Offerings,*
- *a pair of ID's for each conflict, if any.*
- *(If there are no conflicts, the Course Catalog can then change the status of selected Course Offerings from 'unavailable' to 'available' so that Students may now enroll).). [The reason the count of conflicting Course Offerings must be returned is in case the count is zero. C-Reg interprets this as a confirmation that the committed Course Offerings have been accepted by the Course Catalog.]*

e) If any Course Offerings do conflict on date or location, C-Reg indicates the ID's of the conflicting pairs on the display of Course Offerings, with an error message. The Professor may then resolve the conflict by de-selecting one or more Course Offerings and selecting new ones, or cancelling the operation, in which case any selections will be lost.

f) C-Reg sends the details of any changed Course Offerings back to the Course Catalog as per step c).

Steps c) to f) may be repeated until the Professor is satisfied with the selection or cancels.

2.1.2.4 Modify Course Offering commitments

a) If a Professor wishes to modify any of his teaching commitments, he must first enquire on and display the available Course Offerings for his Department that he may teach and any which he has already committed to teach as in 2.1.2.2 and then select a sub-option "Modify Course to Teach".

b) The Professor then modifies the Course Offerings from those displayed in step a) that he commits to teach for the upcoming semester by adding or removing his ID from the Course Offerings, as necessary.

c) C-Reg returns each modified Course Offering to the Course Catalog. Steps d), e) and f) from 2.1.2.3 are repeated until the Professor is satisfied with this selection by pressing 'Save', or cancels. (The Course Catalog can change the status of selected Course Offerings from 'unavailable' to 'available', or vice versa, according to the Professor's decisions.)

d) C-Reg sends data for each modified Course Offering to all Students via a broadcast e-mail. *(Note: how the e-mail system broadcasts this information is outside the scope of the case study. Students must take action if a Course Offering that was 'available' and that they had enrolled for has now become 'unavailable' – see 3.1.2 below.]*

2.1.2.5 Delete Course Offering commitments

a) If a Professor wishes to delete all of his teaching commitments, he must first enquire on and display the Course Offerings that he has already committed to teach as in 2.1.2.2, and then select a sub-option "Delete Course Offerings".

b) The Professor re-enters his ID.

c) C-Reg displays a message asking the Professor to confirm that he wants to delete all his commitments.

d) If the Professor selects 'yes', C-Reg sends the Course Offering data to the Course Catalog, which changes their status to 'unavailable' (for Students to enroll).

e) Alternatively if the Professor selects 'no', the operation is cancelled.

f) C-Reg sends the ID of each deleted Course Offering to all Students via a broadcast e-mail (as in 2.1.2.4 d)).

2.1.2.6 Alternative Flow

Course Catalog System Unavailable

If C-Reg is unable to communicate with the Course Catalog after three tries, C-Reg will display an error message to the Professor. The Professor acknowledges the error message and the Professor must abandon the operation.

2.2 The Measurement Strategy

N.B. The Measurement Strategy parameters are the same for the requirements of Chapters 2, 3 and 4, hence this section is not repeated in the next Chapters 3 and 4.

2.2.1 Measurement purpose

The purpose is to demonstrate to new users of the COSMIC method how to measure the functional size of the requirements, as stated, of the C-Reg software, which is a typical business application. The requirements are not intended to be complete or suitable for building a real system.

2.2.2 Measurement scope and level of decomposition

The measurement scope is all of the Functional User Requirements (FUR) of the C-Reg system. There is a single software layer for this set of requirements, the application layer.

The requirements do not mention exchanges between the PC client and the web server, i.e. the C-Reg system must be measured 'as a whole', ignoring that physically it has two components.

2.2.3 Functional users

The human functional users of the C-Reg system are clerks in the Registrar's office (referred to as 'Registrars') and Professors (see section 2.1) and the Students (see section 3.1.2).

There are also two other functional users: the Course Catalog system and the Billing System (see section 4.1.2). Figure 4 is a context diagram for the C-Reg system showing all its functional users.

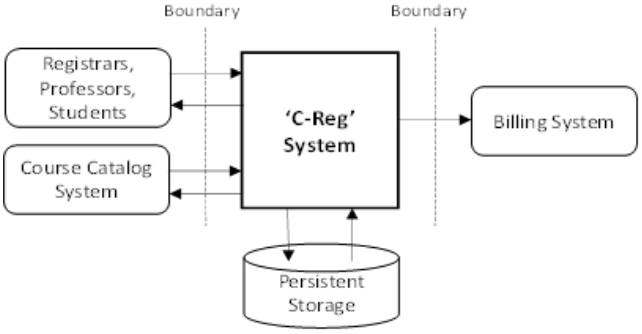


Figure 4 – C-Reg Application Context Diagram

(The E-mail System might also appear to be a functional user, but it is not. See 4.3.)

2.2.4 Level of granularity of the requirements

The functional users of the C-Reg System identified in the requirements are individual humans and software applications.

The requirements in Chapters 2 and 3 sometimes describe clusters of events, such as 'Maintain Professor' Data' but they are all broken down to a level of granularity where single events occur that C-Reg must respond to, such as a Registrar wishing to enquire, add, modify or delete data about a

Professor or to enquire on Professor data. At this level of granularity functional processes and their data movements can be identified, i.e. an exact COSMIC functional size measurement is possible.

2.3 Mapping and Measurement Phases

As for practical reasons this section also contains the measurement result, the Mapping phase is combined with the Measurement phase.

2.3.1 Identifying the functional processes

From the textual descriptions of the requirements, the following triggering event(s) and functional processes are identified as listed in Table 1.

Section	Triggering event	Functional Process
	A Registrar needs to:	
2.1.1.2	add a Professor's details	Add a Professor's details
2.1.1.3	enquire on a Professor's details	Enquire on a Professor's details
2.1.1.4	modify a Professor's details	Modify a Professor's details
2.1.1.5	delete a Professor's details	Delete a Professor's details
	A Professor needs to:	
2.1.2.2	enquire on Course Offerings	Enquire on Course Offerings (Professor)
2.1.2.3	create Course Offering commitments	Create Course Offering commitments
2.1.2.4	modify Course Offering commitments	Modify Course Offering commitments
2.1.2.5	delete Course Offering commitments	Delete Course Offering commitments

Table 1: List of candidate triggering events and candidate functional processes

2.3.2 Identifying the objects of interest

From the requirements in sections 1.3 and 2.1, we can identify the following objects of interest, and the system that holds data about the object of interest.

Source	Object of interest	System storing data about the Object of interest
2.0	Course	Course Catalog
2.0	Course Offering	Course Catalog
2.1.2.3, 2.1.2.4	All conflicting Course Offering selections	(No data is stored about this object of interest but one of its attributes, the count of all conflicting Course Offering selections, is sent in a message from the Course Catalog to C-Reg.)
2.1.1	Professor	C-Reg
2.1.1.2	Errors	(Not stored. We use 'errors' as the name of the object of interest for all error messages output by C-Reg.)

Note: a Registrar is also potentially an object of interest, but the stated requirements do not specify any data describing any Registrar, so an object of interest is not listed for a Registrar.

Below is a list of the objects of interest and their stored data attributes that we know about so far from the requirements.

Course: A standard series of lectures, etc. on a specific subject from the Course Catalog

Key: (Course ID). Other attributes (assumed): Course name, description, Department.

Course offering: A Course that is offered to students during the upcoming Semester

Key: (Course ID, Semester Name). Other attributes: month, room code of the lectures, etc., availability indicator (unavailable, available, full, cancelled, closed), assigned Professor ID, number of students enrolled, maximum number of students that may be enrolled.

Professor: A person who is registered at Wylie College who may deliver a Course Offering for one of his Department's Courses.

Key: (Professor ID). Other attributes: name, address, date of birth, Social Security Number, qualifications, Department, phone, e-mail

2.3.3 Identifying the data movements of the functional processes

The table below shows the functional processes identified in section 2.3.1, including all their movements of data groups (each of which describes an object of interest identified in section 2.3.2). The requirement numbers in the leftmost column refer to the paragraph numbers in the requirements section 2.1.

(N.B. this layout of the analysis of the functional processes is designed for COSMIC training purposes. It would not be an efficient way of recording the results of real measurements.)

In the following 'Errors' means any of the error messages mentioned in the requirements for the functional process concerned.

Require-ment no.	Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move-ment Type	CFP
Maintain Professor Data							
2.1.1.2	Add a Professor	Registrar	Registrar enters details for the Professor	Professor details	Professor	E	1
			C-Reg validates the entered data and checks if the data describe a Professor who already exists	Professor details	Professor	R	1
		Registrar	Registrar selects 'Save' (This step will be omitted from now on in all other processes, except where needed)	Control command		-	-
			C-Reg creates a new Professor	Professor details	Professor	W	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**4**
2.1.1.3	Enquire on a Professor's details	Registrar	Registrar enters Professor ID	Professor ID	Professor	E	1
			C-Reg retrieves the Professor details	Professor details		R	1
		Registrar	C-Reg displays the Professor details	Professor details	Professor	X	1

		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**4**
2.1.1.4	Modify a Professor's details	Registrar	Registrar modifies the displayed details for the Professor	Professor details	Professor	E	1
			C-Reg validates the details and updates the Professor record	Professor details	Professor	W	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**3**
2.1.1.5	Delete a Professor's details	Registrar	Registrar presses the delete command for the Professor whose details are displayed	Professor ID	Professor	E	1
		Course Catalog	C-Reg asks Course Catalog if Professor has any Course Offering commitments	Professor ID	Professor	X	1
		Course Catalog	Course Catalog replies 'yes' or 'no'	Professor's Course Offering commitment status	Course Offering	E	1
			C-Reg prompts the Registrar to confirm the deletion	Prompt Control Command		-	-
			The Registrar confirms or cancels the deletion	Confirmation Control Command		-	-
			C-Reg deletes the Professor	Professor details	Professor	W	1
		Registrar	Display error messages	Error Messages	Errors	X	1
						Total	5

Require-ment no.	**Process descriptions**	**Functional user**	**Sub-process Description**	**Name of Data Group moved**	**Object of interest of Data Goup moved**	**Data Move-ment Type**	**CFP**
Maintain Professor's Course Offering commitments							
2.1.2.2	Enquire on Course Offerings (Professor)	Professor	The Professor enters his ID	Professor ID	Professor	E	1
			C-Reg retrieves the Professor's qualifications and Department	Professor qualifications	Professor	R	1
		Course Catalog	C-Reg sends Professor's qualifications and Department to Course Catalog to retrieve eligible and committed Course Offerings	Professor qualifications	Professor	X	1

		Course Catalog	Course Catalog returns the relevant Course Offerings	Course Offering data	Course Offering	E	1
			C-Reg displays the heading showing Wylie College, upcoming Semester and Date	'Application General data. Not counted. See [2] 4.4.1	-	-	-
			C-Reg displays the Professor's Department	Department name	Department	X	1
		Professor	C-Reg displays the returned Course Offerings	Course Offering data	Course Offering	X	1
		Professor	Display error message	Error Messages	Errors	X	1
						Total	**7**
2.1.2.3	Create Course Offering commitments	Professor	The Professor selects the Course Offerings by entering his ID in each Offering that he wishes to teach	Professor's Course Offering selections	Course Offering	E	1
		Course Catalog	C-Reg sends the Professor's selected Course Offerings to the Course Catalog	Professor's Course Offering selections	Course Offering	X	1
		Course Catalog	C-Reg receives the count of Course Offerings that conflict from the Course Catalog	Count of conflicting courses	All conflicting Course Offering selections	E	1
		Course Catalog	C-Reg receives the ID's of any conflicting pairs of Course Offerings from the Course Catalog	Conflicting Course Offering selections	Course Offering	E	1
		Professor	C-Reg updates the display of Course Offerings with conflicts (if any), or 'Commitments accepted'	Conflicting Course Offering selections	Course Offering	X	1
			Professor repeats the above steps until satisfied with commitments, or cancels			-	-
		Professor	Display error/confirmation message	Error Messages	Errors	X	1
						Total	**6**
2.1.2.4	Modify Course Offering commitments	Professor	Professor modifies his Course Offering selections by entering or removing his ID	Professor's Course Offering selections	Course Offering	E	1
		Course Catalog	C-Reg sends the Professor's selected Course Offerings to the Course Catalog	Professor's Course Offering selections	Course Offering	X	1
		Course Catalog	C-Reg receives the count of Course Offerings that conflict from the Course Catalog	Count of conflicting courses	All conflicting Course Offering selections	E	1
		Course Catalog	C-Reg receives the ID's of any conflicting pairs of	Rejected Course	Course Offering	E	1

			Course Offerings from Course Catalog	Offering selections			
		Professor	C-Reg updates the display Course Offerings with conflicts (if any), or 'Commitments accepted'	Rejected Course Offering selections	Course Offering	X	1
			Professor repeats the above steps until satisfied with commitments or cancels			-	-
		Student	C- Reg sends an e-mail for broadcast to Students	Modified Course Offering e-mail	Course Offering	X	1
		Professor	Display error message	Error Messages	Errors	X	1
						Total	**7**

Require-ment no.	**Process descriptions**	**Functional user**	**Sub-process Description**	**Name of Data Group moved**	**Object of interest of Data Goup moved**	**Data Move-ment Type**	**CFP**
2.1.2.5	Delete Course Offering commitments	Professor	Professor enters his ID and presses delete command	Professor ID	Professor	E	1
			C-Reg prompts the Professor to confirm the deletion	Prompt Control Command		-	-
			The Professor confirms or cancels the deletion	Confirmation Control Command		-	-
		Course Catalog	C-Reg sends ID's of deleted Course Offerings to Course Catalog	Course Offering ID	Course Offering	X	1
		Student	C- Reg sends an e-mail for broadcast to Students	Deleted Course Offering e-mail	Course Offering	X	1
		Professor	Display error message	Error Messages	Errors	X	1
						Total	**4**

2.4 Questions and Answers

Question re 2.1.1.2 'Add a Professor's details'

Suppose that C-Reg generates a Professor ID rather than, as currently specified, a Registrar must enter a Professor ID in this process. What would be the effect on the analysis of this functional process?

Answer

According to the requirements as stated, there would be the following consequences:

- The Registrar would not need to enter a Professor ID, but an Entry for the other Professor details would still be needed.

- the Create functional process would still need a Read to check whether a Professor of the same name already exists (given the convention that a Professor ID is a combination of his surname and a serial number). If that condition is found, there would be another error message, e.g. 'Professor of this name already exists' but we would still identify only one error/confirmation Exit to account for all such messages.
- Although not stated in the current requirements, logically C-Reg would display the newly-created Professor ID, which would imply one additional Exit.
- As Registrars would not now be maintaining a list of Professor ID's outside C-Reg, they would almost certainly need an additional enquiry functional process to enter a Professor surname and display all ID's that match that name (if any), e.g. 'Smith1', 'Smith2', 'Smith3', etc. This functional process would add 4 CFP to the C-Reg size.

Question 1 re 2.1.1.2 'Add a Professor's details and 2.1.1.4 'Modify' a Professor's details

The requirements do not state how entered Department names are represented and hence how an entered Department can be validated. Suppose the C-Reg system holds a table of valid Department names, and that when entering data for a 'Add' or 'Modify' Professor details there is a requirement to display a drop-down list of valid Department names from which the Registrar must select a valid name. What would be the effect on the analysis of these functional processes?

Answer

There would be no effect on the size of these two functional processes because a Department is not an object of interest for these processes. (No data describing a Department other than its name would be required to be stored by C-Reg.) See [2], section 4.1.7.

An alternative solution for validating entered Department names (rather than holding a table of valid Department names in C-Reg) would be for C-Reg to send an entered name to the Course Catalog for validation. This would require an extra Exit/Entry pair in each of the 'Add' or 'Modify' Professor's details functional processes

Question 2 re 2.1.1.2

Does the answer given above to 'Question 1 re 2.1.1.2' depend on whether the table of names in C-Reg is hard-coded in the software or is stored in a table of names that can be maintained by a System Administrator?

Answer

It makes no difference to the size of the Add and Modify Professor's details functional processes how Department names are stored. Department is not an object of interest for these functional processes, so no Read is needed to access the names.

However, if the Department names are stored in a table that may be maintained by a System Administrator, then 'Department' is an object of interest to the Administrator. Clearly he/she will need functional processes to maintain Department names and these would add to the size of C-Reg.

Question re. 2.4.1

Can one triggering event (Registrar selects the "Maintain Professor" activity from Main Form) trigger more than one functional process (Add, Modify, Delete and Enquire on a Professor)?

Answer

There is no such *triggering event* as 'Maintain Professor' in this case study. In the real world of a Registrar there are events like:

- A new Professor starts work, so his details must be added to C-Reg;
- Something changes for a Professor, e.g. his address, so the data stored about him must be changed;

- A Professor leaves or dies or whatever, so he must be removed from the database;
- A Registrar wants to enquire on a Professor's data.

These events cause functional users to enter data groups into the software, i.e. they are triggering events, leading to the corresponding functional processes (i.e. add, modify, delete and enquire) respectively. 'Maintain Professor' is a group of functional process types. Selection of this group from a menu by a Registrar is a 'Control Command', not measured in COSMIC because it does not move data describing an object of interest– see section 3.5.10 of the Measurement Manual, v4.0.2 [1].

Question re 2.1.2.3 e)

Why doesn't the solution given for requirement 2.1.2.3 e) show all the data movements that allow a professor to change the selection(s) to resolve any conflict that the Course Catalog has found?

Answer

The actions by the Professor following notification of a conflict to de-select one of the conflicting Course Offerings and send his new choices back to the Course Catalog involve the identical data groups that have been moved earlier in the process. Repeating the cycle of requirement 2.1.2.3 e) involves repeated *occurrences* of the same data movements. According to the 'Data movement uniqueness' rule (see [1], section 3.5.7), these data movements should be identified only once.

Cancelling the whole operation is a Control Command and should not be identified.

Question re 2.1.1.5

In the analysis of the requirements 2.1.1.5, 'Delete a Professor', a Registrar enters a delete command, which is identified as an Entry. Why is the delete command identified as an Entry and not as a Control Command? The delete command just says 'delete the Professor whose details are displayed'; it does not convey data about a Professor. This same comment applies to several functional processes.

Answer

The delete command DOES convey data about a specific object of interest, the Professor ID of the already-identified and displayed Professor, so this must be identified as an Entry. It may be that no attribute such as a specific Professor ID must be physically re-entered by the Registrar, but the displayed ID is certainly implied in this command. This is quite different from a 'real' Control Command (as defined in section 3.5.10 of the Measurement Manual v4.0.2 [1]), such as 'page-up' or 'page down'. This same phenomenon occurs in many of the functional processes.

3 REGISTRARS AND STUDENTS

3.1 Requirements for Registrars and for Students

This section has two groups of requirements.

3,1.1 Maintain Student data (by a Registrar)

3.1.2 Maintain Student Schedule items (by any Student)

3.1.1 Maintain Student Data

3.1.1.1 Brief Description

This group of requirements enables a Registrar to maintain data about Students, by first selecting the "Maintain Student" activity from the Main Form.

Each Student is identified by a unique identification (or 'ID'). *[For simplicity in the requirements, we assume that when making any enquiries, a Registrar knows the ID of all Students, and that each Student knows his own ID.]*

3.1.1.2 Add a Student's details

a) When a Registrar wishes to enter data about a new Student, he selects the sub-option "Add Student".
b) C-Reg displays a blank formatted screen for entry of student data.
c) The Registrar enters the following details for the student: name, date of birth, social security number, status, and graduation date.
d) C-Reg validates the data to ensure the proper formats. If the data are valid C-Reg creates a new student record and assigns a unique system-generated ID, which is displayed.
e) Alternatively, if invalid data are entered, C-Reg displays an error message, "Student Data Invalid". The Registrar can then correct the data or cancel the operation.

3.1.1.3 Enquire on a Student's details

a) When a Registrar wishes to enquire on the details of a particular Student he must first select the sub-option "Enquire on a Student" and then enter a Student ID.
b) C-Reg searches for a Student with the specified ID. If the ID is valid C-Reg shows the Student's name, date of birth, status, graduation date.
c) Alternatively, if the Student ID is not found, C-Reg displays an error message, "Student Not Found". The Registrar can then type in a different ID or cancel the operation.

3.1.1.4 Modify a Student's details

a) If a Registrar wishes to modify the details of a Student, he must first retrieve the Student's details as in 3.1.1.3 and then select the sub-option "Modify Student."
b) The Registrar may then modify one or more of the displayed student data items (but not the Student ID). When changes are complete, the Registrar presses "Save."
c) C-Reg validates the data to ensure the proper formats and if correct updates the Student details.
d) Alternatively, if invalid data are entered, C-Reg displays an error message, "Student Data Invalid". The Registrar can then correct the data or cancel the operation.

3.1.1.5 Delete a Student's details

a) If a Registrar wishes to delete the details of a Student, he must first retrieve the Student's details as in 3.1.1.3 and then select the sub-option "Delete Student."
b) C-Reg checks whether the Student has a Student Schedule (see 3.1.2 below).
c) If the Student does not have a Student Schedule, C-Reg displays a message asking the Registrar to confirm the deletion.
d) The Registrar selects "yes" and the student is deleted from C-Reg.
e) Alternatively, if the Student has a Student Schedule, deletion isn't allowed, C-Reg displays an error message and the Registrar must abandon the operation.

3.1.2 Maintain Student Schedule items

3.1.2.1 Brief Description

This group of requirements enables a Student to create his personal 'Student Schedule' for the upcoming semester by enrolling in up to six Course Offerings. Each item in a Student Schedule is uniquely identified by the ID of the Student and the ID of the Course Offering.

The Student can also enquire on, modify or delete Course Offerings in his Schedule, provided the changes are made before course registration is closed (see 4.1.2 Close Registration).

As already stated, the Course Catalog holds the Course Offerings for the upcoming semester. It also holds any 'Prerequisite' for each Course Offering. For any Course 'X', a Pre-requisite is any one other Course 'Y' for which a Student must have achieved satisfactory examination results before he may enroll for Course 'X'.

The C-Reg system holds for each current Student:

- his 'Schedule' of up to six Course Offerings (known as 'Student Schedule items') that he wishes to attend for the upcoming semester;
- his results for successfully-completed examinations for each Course in previous semesters.

3.1.2.2 Enquire on Course Offerings (Student)

a) When a Student wishes to enquire on Course Offerings, he must first select 'Maintain Student Schedule' from the Main Form and then 'Enquire on Course Offerings (Student)' from the Sub-menu.
b) The Student enters his ID.
c) C-Reg retrieves the details of all Course Offerings (date/time and location, and their availability status) from the Course Catalog as well as the ID of a pre-requisite Course (if any) for each Course Offering.
d) C-Reg displays these Course Offering details and the entered Student ID.

3.1.2.3 Create a Student Schedule

a) When a Student wishes to create his Schedule by selecting Course Offerings for the first time, he must first retrieve the Course Offerings as in 3.1.2.2 and then select the sub-option "Create Student Schedule".
b) The Student selects at most six available Course Offerings from the displayed list for his Schedule. Once the selections are complete, the Student selects "Submit", which results in his ID and the selections being entered *[The Student must manually check there are no conflicts on date/time of his selected Course Offerings. C-Reg will only accept selection of a Course Offering that has the status 'available'.]*

c) C-Reg checks whether the Student has achieved the necessary pre-requisite for each Course Offering that the Student has selected for enrollment. If the check is OK, C-Reg accepts each selected Course Offering as an item in the Student's Schedule, with the status "enrolled".

d) Alternatively, if C-Reg determines that the Student has not satisfied the necessary pre-requisite for a particular selected Course, an error message is displayed.

e) The Student can then either select an alternative Course Offering and submit his selection again, or cancel the whole operation (in which case no Schedule is saved).

f) C-Reg stores the Student Schedule items if all prerequisites have been satisfied.

g) C-Reg sends the ID's of the added Course Offerings to the Course Catalog so that the latter can update the number of Students enrolled for each Course Offering.

3.1.2.4 Modify a Student Schedule

a) If a Student wishes to modify the Course Offerings in his Schedule, he must first retrieve the Course Offerings as in 3.1.2.2 and then select "Modify Student Schedule".

b) C-Reg retrieves the Student's current Schedule items, both 'enrolled' and 'selected' (see 3.1.2.6 below) and adds this status to each of the displayed Course Offerings that are 'available'.

c) The Student adds and/or deletes Course Offerings to/from the selections in his current Schedule. When his new selections are complete, the Student selects "Submit" *[Subject to the same limitations as in 3.1.2.3 b).]*.

d) C-Reg checks whether the Student has achieved the necessary prerequisite for each Course Offering that the Student has now selected for enrollment. If the check is OK, C-Reg then updates the selected Course Offering in the Schedule, with the status "enrolled".

e) Alternatively, if C-Reg determines that the Student has not satisfied the necessary prerequisite for a particular selected course, an error message is displayed.

f) The Student can either select an alternative Course Offering and submit his selection again, or cancel the whole operation (in which case the modified Schedule is not saved, i.e. the original Schedule is unchanged).

g) C-Reg stores the Student Schedule items.

h) C-Reg sends the ID's of the modified Course Offerings, if any, to the Course Catalog so that the latter can update the number of Students enrolled for each Course Offering.

3.1.2.5 Delete a Student Schedule

a) If a Student wishes to delete his Schedule, he must first select 'Maintain Student Schedule' from the Main Form and then select the sub-option "Delete Schedule".

b) The Student enters his ID.

c) C-Reg retrieves and displays his Student Schedule items (his list of enrolled and/or selected Course Offerings).

d) C-Reg prompts the Student to verify the deletion; the Student verifies the deletion.

e) C-Reg deletes the Schedule items and issues a confirmation message.

f) C-Reg sends the ID's of the deleted Course Offerings to the Course Catalog so that the latter can update the number of students enrolled for each course.

g) C-Reg issues an error message in case the Student's Schedule is not found.

3.1.2.6 Alternative Flows

a) Save the Student Schedule items

At any point during the 'Add' or 'Modify' processes, the Student may choose to save his Schedule items by pressing "Save". Selected Course Offerings that were not previously marked as 'enrolled' are marked as 'selected' Schedule items and are saved on C-Reg and the operation ceases, In this case the ID of any Course Offerings of status 'selected' are not sent to the Course Catalog.

b) Course Catalog System Unavailable

If C-Reg is unable to communicate with the Course Catalog after three tries, C-Reg will display an error message to the Student. The Student acknowledges the error message and the Student must abandon the operation.

3.2 Mapping and Measurement Phases

For the Measurement Strategy parameters, see section 2.2.

3.2.1 Identifying the functional processes

From the textual descriptions of the requirements, the following triggering event(s) and functional processes are identified as listed in Table 2.

Section	Triggering event	Functional Process
	A Registrar needs to:	
3.1.1.2	add a Student's details	Add a Student's details
3.1.1.3	enquire on a Student's details	Enquire on a Student's details
3.1.1.4	modify a Student's details	Modify a Student's details
3.1.1.5	delete a Student's details	Delete a Student's details
	A Student needs to:	
3.1.2.2	enquire on Course Offerings	Enquire on Course Offerings (Student)
3.1.2.3	create a Student Schedule	Create a Student Schedule
3.1.2.4	modify a Student Schedule	Modify a Schedule
3.1.2.5	delete a Student Schedule	Delete a Student Schedule

Table 2: List of candidate triggering events and candidate functional processes

3.2.2 Identifying the objects of interest

This list of objects of interest in 3.2.2 must now be extended to include the fact that we are told in 3.1 2 1 that C-Reg holds past examination results for each Student for each Course. *[The requirements do not specify how these past examination results are loaded for each student.]*

Source	Object of interest	System storing data about the Object of interest
2.0	Course	Course Catalog
2.0	Course Offering	Course Catalog
2.1.1	Professor	C-Reg
2.1.2.4 d)	Student	C-Reg
2.1.2.4 d)	Student Schedule item	C-Reg
3.1.2.1	Student/Course	C-Reg

3.1.1.2	Errors	(Not stored. We use ‘errors’ as the name of the object of interest for all error messages output by C-Reg.)

The list of the objects of interest and their stored data attributes that we know about so far from the requirements is also now extended as below.

Course: A standard series of lectures, etc. on a specific subject from the Course Catalog

Key: (Course ID). Other attributes (assumed): Course name, description, Department.

Course offering: A Course that is offered to students during the upcoming Semester

Key: (Course ID, Semester ID). Other attributes: dates, locations of the lectures, etc., availability indicator (unavailable, available, full, cancelled, closed), pre-requisite Course ID, assigned Professor ID, number of students enrolled, maximum number of students that may be enrolled.

Professor: A person who is registered at Wylie College who may deliver a Course Offering for one of his Department’s Courses.

Key: (Professor ID). Other attributes: name, address, date of birth, Social Security Number, qualifications, Department, phone, e-mail address

Student: A person who is registered at Wylie College who may enrol in a Course Offering.

Key: (Student ID). Other attributes: name, date of birth, social security number, graduation date, e-mail address.

Student Schedule item: A record for each Student for each of his selected or enrolled Course-Offerings

Key: (Student ID, Course Offering ID). Oher attributes: status (enrolled or selected)

Student/Course: A record for each Student of his examination results for previously-attended Courses

Key: (Student ID, Course ID, Semester ID). Other attributes (assumed): Examination grade.

3.2.3 Identifying the data movements of the functional processes

The table below shows the functional processes identified in section 3.2.1, including all their movements of data groups (each of which describes an object of interest identified in section 3.2.2). The requirement numbers in the leftmost column refer to the paragraph numbers in the requirements section 3.1.

In the following ‘Errors’ means any of the error messages mentioned in the requirements for the functional process concerned.

Require-ment no.	Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move-ment Type	CFP
Maintain Student Data							
3.1.1.2	Add a Student's details	Registrar	Registrar enters Student details	Student details	Student	E	1
			C-Reg creates a new Student record	Student details	Student	W	1
		Registrar	C-Reg displays the generated Student ID	Student ID	Student	X	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**4**
3.1.1.3	Enquire on a Student's details	Registrar	Registrar enters Student ID	Student ID	Student	E	1
			C-Reg obtains the Student details	Student details	Student	R	1
		Registrar	C-Reg displays the Student details	Student details	Student	X	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**4**
3.1.1.4	Modify a Student's details	Registrar	Registrar modifies the displayed Student details	Student details	Student	E	1
			C-Reg stores the modified details	Student details	Student	W	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**3**
3.1.1.5	Delete a Student's details	Registrar	Registrar enters the delete command for the selected Student	Student ID	Student	E	1
			C-Reg checks if the Student has a Student Schedule	Student Schedule item	Student Schedule item	R	1
			C-Reg prompts the Registrar to confirm the deletion	Prompt Control Command		-	-
			The Registrar confirms the deletion	Confirmation Control Command		-	-
			C-Reg deletes the Student details	Student details	Student	W	1
		Registrar	Display error message	Error Messages	Errors	X	1
						Total	**4**

Require-ment no.	Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move-ment Type	CFP
Maintain Student Schedule items							
3.1.2.2	Enquire on Course Offerings (Student)	Student	Student enters his ID and 'Enquire on courses' command	Student ID	Student	E	1
		Course Catalog	C-Reg asks Course Catalog for Course Offerings	Course Offering request	Course Offering	X	1
		Course Catalog	C-Reg receives Course Offering details, incl. pre-requisite Course ID	Requested Course Offerings	Course Offering	E	1
		Student	C-Reg displays Course Offering details, incl. any pre-requisite Course ID	Requested Course Offerings	Course Offering	X	1
		Student	C-Reg displays Student ID	Student ID	Student	X	1
		Student	Display error message	Error Messages	Errors	X	1
							6
3.1.2.3	Create a Student Schedule	Student	Student selects 'Create Schedule' and his ID is entered	Student ID	Student	E	1
		Student	Student selects up to 6 Course Offerings	Student's selected Course Offerings	Student Schedule item	E	1
			C-Reg checks if the Student satisfies pre-requisites of selected Course Offerings	Student previous exam results	Student/ Course	R	1
		Course Catalog	Validated Course Offering ID's are sent to the Course Catalog so that it can maintain the count of Students for each Course Offering	Course offering ID	Course Offering	X	1
	'Submit'		Student's Schedule items are marked 'enrolled' and made persistent on C-Reg	Student Schedule item details	Student Schedule item	W	1
	"Save"		Student's Schedule items are marked 'selected' and made persistent on C-Reg	Student Schedule item details		(Same as for Submit)	-
		Student	Display error message	Error Messages	Errors	X	1
							6
3.1.2.4	Modify a Student Schedule	Student	Student selects 'Modify Course Offerings' and his ID is entered	Student ID	Student	E	1

			C-Reg retrieves Student's Schedule items	Student Schedule item details	Student Schedule item	R	1
		Student	C-Reg adds the item statuses (enrolled or selected) to the displayed Course Offerings	Student Schedule item details	Student Schedule item	X	1
		Student	Student modifies his displayed Course Offering selections	Modified Student Schedule item details	Student Schedule item	E	1
			C-Reg checks if the Student satisfies pre-requisites of modified Course Offerings	Student previous exam results	Student/ Course	R	1
	"Submit"		C-Reg marks Student Schedule items as 'enrolled' and makes them persistent	Student Schedule item	Student Schedule item	W	1
3.1.2.6 a)	"Save"		C-Reg marks Student Schedule items as 'selected' and makes them persistent	Student Schedule item		(Same as for Submit)	-
		Course Catalog	C-Reg sends modified Course Offering ID's to the Course Catalog	Course Offering changes	Course Offering	X	1
		Student	Display error message	Error Messages	Errors	X	1
							8
3.1.2.5	Delete a Student Schedule	Student	The Student selects the 'Delete Schedule' command and his ID is entered	Student	Student	E	1
			C-Reg retrieves the Student's Schedule items	Student Schedule Item details	Student Schedule item	R	1
		Student	C-Reg displays the Student's Schedule items	Student Schedule Item details	Student Schedule item	X	1
			C-Reg prompts the Student to verify the deletion	Control Command		-	-
			The Student confirms the deletion	Control Command		-	-
			C-Reg deletes the Student Schedule items	Student Schedule item details	Student Schedule item	W	1
		Course Catalog	C-Reg sends the deleted Course Offering ID's to the Course Catalog	Course Offering deletions	Course Offering	X	1
		Student	Display error message	Error Messages	Errors	X	1
							6

3.3 Questions and Answers

Question re. 3.3.1

The functional process “Delete a Student’s details” has a requirement: “C-Reg prompts a Registrar to verify the deletion”. Confirmations like this appear in other functional processes like Delete a Professor. Why are these ‘confirmation prompts’ not identified as data movements?

Answer

In the Business Application Guideline [2], the rule is given in 4.4.1 that clicking ‘OK’ to confirm some entered data should be ignored as it is a Control Command. The command does not move data describing an object of interest.

Question re 3.1.2.3 and 3.1.2.4

The requirements for these two functional processes (‘Create’ and ‘Modify’ a Student Schedule) do not specify that the Student must enter his ID at the start of the process, but the analysis shows an Entry is identified for the Student ID. Why?

Answer

The requirements state that these two functional processes can only be started after executing an ‘Enquire on Course Offerings (Student)’ process. The output of this latter process will show the Student ID which he entered, as well as the retrieved Course Offerings. This display will become the data entry screen for the ‘Create’ or ‘Modify’ a student Schedule functional processes if they are selected.

When the Student has made his Course Offering selections, he presses ‘Submit’ (or ‘Save’) and the data on the screen crosses the boundary from the Student functional user to enter the ‘Create’ or ‘Modify’ functional process. Therefore we must identify an Entry for the input of the Student ID. The Student may not need to physically re-enter his ID, but the ‘Create’ and ‘Modify’ functional processes need this Student ID from their input screen.

Question re. 3.1.2.3

The requirement 3.1.2.3 “Create a Student Schedule” states that the Student ‘submits’ his Schedule Items and later that C-Reg ‘saves’ the Schedule. The alternative flow requirement 3.1.2.6 states that the student may ‘save’ a Schedule without submitting it. This appears to indicate that there are as many as three Writes. How does the ‘data movement uniqueness’ rule apply here?

Answer

The ‘Submit’ is the final action of entering data that is measured as one Entry. After the Student presses ‘Submit’, his Student Schedule is made persistent (i.e. stored) by C-Reg with the status ‘enrolled’ for each Course Offering. The action by a Student to ‘save’ his Student Schedule, is the same data movement, but with the status ‘selected’. The data group moved - ‘Student Schedule item’ - is the same type, regardless of the value of the ‘status’ data attribute. There is only one Write data movement needed to satisfy all these requirements.

Question re. 3.1.2.4

In section 3.1.2.4, ‘Modify Course Offerings in a Student Schedule’, after the current Course Offerings have been displayed and the Student presses ‘Modify Course Offerings’, C-Reg retrieves the current Student Schedule items and adds their status (enrolled or selected) to the displayed Course Offerings. The Student can now modify his selection. Why is an Exit identified for this update of the displayed Course Offerings?

Answer

At the start of the process ‘Modify Course Offerings in a Student Schedule’, the screen displayed for data input shows the available Course Offerings. When C-Reg retrieves a Student Schedule item and adds its status (enrolled or selected) to the displayed Course Offering, C-Reg is now displaying a Student Schedule item in place of a Course Offering, i.e. it is displaying a different data group. (The display is now showing the list of Course Offerings that are not of interest to the Student interspersed with those that are of interest, i.e. with Student Schedule items. So the display now has two Exits.)

Question re. 3.1.2

Why is ‘Student Schedule’ not listed as an object of interest?

Answer

‘Student Schedule’ is a name we give for convenience to a collection of up to six Student Schedule items. There is no data stored or moved that describes the ‘Student Schedule’ collection; data are stored only about the individual Student Schedule items. ‘Student Schedule’ is therefore not an object ‘of interest’.

Question re 3.2.2

‘Department’ is shown as an attribute of both the objects of interest ‘Professor’ and ‘Course Offering’. How can this be true?

Answer

The meaning of this attribute is different for each object of interest.

The meaning of the attribute of Professor that is named ‘Department’ is ‘the Department to which the Professor belongs’

The meaning of the attribute of Course Offering that is named ‘Department’ is ‘the Department which provides this Course Offering’.

These two attributes should therefore really be named differently. A good convention for naming attributes is to form the name from the object of interest that the attribute describes, followed by a name that distinguishes the attribute; these two are joined by an underscore ‘_’. For these two attributes, the names could be:

Professor_Department

Course Offering_Department

In practice as in this case study, attributes are usually listed in statements of requirements, and on displays and in report headings, with short, simple names whose meanings are clear in the context.

Measurers must be aware of these practices if objects of interest and data groups are to be distinguished correctly. See [2] section 4.2.1 ‘Warning on misleading attribute names’.

4 REGISTRARS

4.1 Requirements for the Registrars

This section has two groups of requirements:

4.1.1 Registrar Management Information

4.1.2 Close Registration

4.1.1 Registrar Management Information

4.1.1.1 Brief Description

This group of requirements enables a Registrar to monitor progress in course registrations by a report and an Enquiry, and to close registration, i.e. prevent further registrations.

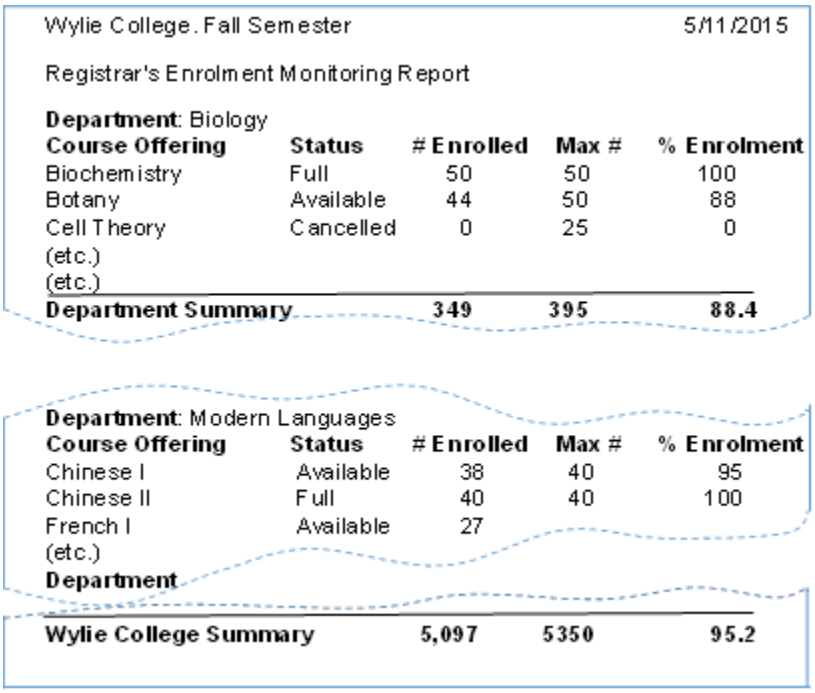
4.1.1.2 Monitor Course Offering Enrollment progress

a) When a Registrar wishes to check progress on Course Offering enrollment, he selects the menu sub-option ‘Monitor Course Offering Enrollment’.

b) C-Reg requests from the Course Catalog and receives all Course Offerings including their current status, the name of the Department that ‘owns’ the Course, the number of students enrolled, and the maximum number of students that may be enrolled.

c) C-Reg sorts the received Course Offerings by Department name and produces a report (see Figure 5) showing, at the displayed date:

- For each Department (by name), a list of Course Offerings, each with its status; the actual number of Students enrolled; the maximum number that may be enrolled; and the ‘enrollment percentage’ (= actual/maximum)
- For each Department, the actual number of Students enrolled; the maximum number that may be enrolled; and the average enrollment percentage for all the Department’s Course Offerings
- For the whole of Wylie College, the actual number of Students enrolled; the maximum number that may be enrolled; and the average enrollment percentage for all Course Offerings.



Wylie College. Fall Semester 5/11/2015

Registrar's Enrolment Monitoring Report

Department: Biology

Course Offering	Status	# Enrolled	Max #	% Enrolment
Biochemistry	Full	50	50	100
Botany	Available	44	50	88
Cell Theory	Cancelled	0	25	0
(etc.)				
(etc.)				
Department Summary		**349**	**395**	**88.4**

Department: Modern Languages

Course Offering	Status	# Enrolled	Max #	% Enrolment
Chinese I	Available	38	40	95
Chinese II	Full	40	40	100
French I	Available	27		
(etc.)				
Department				
Wylie College Summary		**5,097**	**5350**	**95.2**

Figure 5. – Three sections from an example report resulting from requirement 4.1.1.2

4.1.1.3 Monitor Student Schedule Enrollment progress

a) When a Registrar wishes to check progress on Student Schedule enrollment, he selects the menu sub-option 'Monitor Student Schedules'.

b) C-Reg retrieves the Student Schedule items for each Student and the names of all Students that have not enrolled for or selected any Course Offerings.

c) C-Reg displays, at the report's date:

- the name of every Student that has not enrolled for or selected any Course Offerings i.e. has no Student Schedule items
- the overall percentage of enrollments in Student Schedules (= total actual enrollments in Schedules as a percentage of the maximum possible number of enrollments in a Schedule, where the latter is six per Student) for Wylie College.

4.1.2 Close Registration

4.1.2.1 Brief Description

This requirement allows a Registrar to close the registration processes, by changing the status of each Course Offering (i.e. the value of its availability indicator field) from 'available' or 'full', to 'cancelled' or 'closed'. As close registration processing cannot be performed if a registration is in progress, the Registrar announces in advance the closing date to people concerned. *[This announcement is not part of the scope of the measurement.]* Course Offerings that do not have enough students are cancelled. C-Reg deals with the consequences of course closure or cancellation for billing Students and for their Schedules.

4.1.2.2 Close Registration

a) When a Registrar wishes to close all registration processes, he selects the menu sub-option "Close Registration".
b) C-Reg obtains the Course Offerings from the Course Catalog that are ‘available’ or ‘full’.
c) C-Reg performs the following processing:
   - For each Course Offering that has less than 3 students enrolled, C-Reg sets the Course Offering status to ‘cancelled’.
   - For all other Course Offerings, C-Reg sets the status to ‘closed’.
d) C-Reg returns the Course Offering data to the Course Catalog so that it can update Course Offering statuses in its files.
e) For each combination of enrolled Student and closed Course Offering, C-Reg sends the Student ID and the Course Offering ID to the Billing System.
f) C-Reg prepares and issues e-mails announcing cancelled Course Offering(s) to be sent to each affected Student via the E-mail System. The E-mail system requires a message to have three fields: To (e-mail address), Subject (header), Message (body). The e-mail address must conform to normal internet standards; the header and body may be unstructured text.

   An example message could be:
   To: John.Doe@wyliemail.com, Subject: 'Notice of course cancellations',
   Message:
   'Dear John Doe, I regret to inform you that due to insufficient numbers the following courses in which you enrolled have been cancelled: Asian History III, Korean for Beginners.
   Please make an appointment to visit my office to discuss your options.
   (Signed) Dr Mary Shelley, Registrar.'

g) C-Reg updates Student Schedule items where necessary.

4.2 Mapping and Measurement Phases

For the Measurement Strategy parameters, see section 2.2.

4.2.1 Identifying the functional processes

There are two functional processes triggered by a Registrar’s need to monitor progress in enrollments in Course Offerings and per Student in their Schedules.

There is one functional process triggered by the need for a Registrar to close registration.

4.2.2 Identifying the objects of interest

The two ‘Management Information’ processes produce data describing the objects of interest shown in the table below.

Note: no data is stored *about* a ‘Department’ or *about* ‘Wylie College’. But they are “*things in the world of the functional user that are identified in the Functional User Requirements, about which the software is required to process .... data*” (as per the definition of an object of interest). ‘Department’ and ‘Wylie

College' are therefore objects of interest for the C-Reg output which includes some derived, transient data describing them.

Source	Object of interest	System storing data about the Object of interest
4.1.1.2 c)	Course Offering	Course Catalog
4.1.1.2 c)	Department	No data is stored about this object of interest in C-Reg. Transient data about Department is output by C-Reg.
4.1.1.2 c) and 4.1.1.3 c)	Wylie College	No data is stored about this object of interest in C-Reg. Transient data about Department is output by C-Reg.
4.1.1.3 b	Student Schedule item	C-Reg
4.1.1.3 b)	Student	C-Reg

The list of the objects of interest and their data attributes from the requirements of section 4.1 is as below.

Course Offering: A Course that is offered to students during the upcoming Semester

Key: (Course ID, Semester Name). Other attributes: month, room code of the lectures, etc., availability indicator (unavailable, available, full, cancelled, closed), assigned Professor ID, number of students enrolled, maximum number of students that may be enrolled.

Department: A part of Wylie College with specific research and teaching responsibilities.

Key: (Department name). Other (derived) attributes: the actual number of Students enrolled; the maximum number that may be enrolled; and the average enrollment percentage for all the Department's Course Offerings, all at the date shown.

Student: A person who is registered at Wylie College who may enrol in a Course Offering.

Key: (Student ID). Other attributes: name, date of birth, social security number, status, graduation date and e-mail address.

Student Schedule item: A record for each Student for each of his selected or enrolled Course-Offerings

Key: (Student ID, Course Offering ID). Oher attributes: status (enrolled or selected)

Wylie College: A teaching college that sponsors the C-Reg system.

Key: (Wylie College name). Other (derived) attributes: the actual number of Students enrolled; the maximum number that may be enrolled and the average enrollment percentage for all Course Offerings; the overall percentage of enrollments in Student Schedules and the overall percentage of selections in Student Schedules, all at the date shown.

The 'Close Registration' process does not introduce any new objects of interest. The list of objects of interest relevant to this process is therefore the same as in section 4.2.2.

4.2.3 Identifying the data movements of the functional processes

The table below shows the functional processes identified in section 4.2.1, including all their movements of data groups (each of which describes an object of interest identified in section 4.2.2). The requirement numbers in the leftmost column refer to the paragraph numbers in the requirements section 4.1.

In the following 'Messages' means any of the 'Error/confirmation messages' mentioned in the requirements for the functional process concerned.

Require-ment no.	Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move-ment Type	CFP
Registrar Management Information							
4.1.1.2	Monitor Course Offering Enrolment progress	Registrar	Registrar selects 'Monitor Course Offering Enrolment progress'	Course Offering Enrolment Data request	Course Offering	E	1
		Course Catalog	C-Reg requests Course Catalog to send Course Offering data	Course Offering Data request	Course Offering	X	1
		Course Catalog	C-Reg receives Course Offering data	Course Offering Data	Course Offering	E	1
		Registrar	C-Reg outputs Department name, the number of Students enrolled, etc., and the average percentage enrolment for all Course Offerings that it 'owns'	Department data	Department	X	1
		Registrar	C-Reg outputs number of Students enrolled, etc for each Course Offering the Department 'owns'	Departmental Course Offering Enrolment Data	Course Offering	X	1
		Registrar	C-Reg outputs , the number of Students enrolled, etc., and the av. % enrolment in Course Offerings for Wylie College	Wylie College Course Offering Enrolment data	Wylie College	X	1
						Total	**6**
4.1.1.3	Monitor Student Schedule Enrolment progress	Registrar	Registrar selects 'Monitor Student Schedule Enrolment progress'	Student Schedule Enrolment progress request	Student Schedule item	E	1
			C-Reg retrieves Student details	Student details	Student	R	1
			C-Reg retrieves Student Schedule items	Student Schedule item data	Student Schedule item	R	1

		Registrar	C-Reg displays names of Students with no enrolled or selected Course Offerings	Student name	Student	X	1
		Registrar	C-Reg displays the overall % of enrolments in Student Schedules for Wylie College	Wylie College Student Schedule enrolment data	Wylie College	X	1
						Total	**5**
Close Registration							
4.1.2.2	Close Registration	Registrar	The Registrar selects sub-option "Close registration"	Date closed	Course Offering	E	1
		Course Catalog	C-Reg requests Course Offering data (with number of students enrolled, etc.) from the Course Catalog	Course Offering Data request	Course Offering	X	1
		Course Catalog	Course Offering data received	Course Offering Data	Course Offering	E	1
			C-Reg checks that at least three students enrolled. If <3, it sets Course Offering status to 'cancelled'	(Data manipulation)		-	-
		Course Catalog	C-Reg sends updated statuses of Course Offerings to the Course Catalog	Course Offering statuses	Course Offering	X	1
			C-Reg retrieves the Student Schedule items for the Students enrolled for each Course Offering	Student Schedule item data	Student Schedule item	R	1
		Billing System	C-Reg sends data to the Billing System for each Student enrolled for each Course Offering that has not been cancelled	Student Schedule item data	Student Schedule item	X	1
			C-Reg retrieves Student name and e-mail address	Student details	Student	R	1
		Student	C-Reg sends info on each cancelled Schedule item to each Student in the form of an e-mail	Student e-mail address.	Student	X	1
		Student		Cancelled Student Schedule item message	Student Schedule item	X	1
			C-Reg updates Student Schedule items for cancelled Course Offerings	Student Schedule item data	Student Schedule item	W	1
						Total	**10**

4.3 Questions and Answers

Question 1 re. 4.1.2

In the 'Close Registration' functional process, the requirement 4.1.2.2 e) states: 'For each closed Course Offering and enrolled Student combination, C-Reg sends the Student ID and the Course Offering ID to the Billing System.

The size of this requirement has been given as one Exit in the analysis of 4.1.2. The object of interest of the data group moved is 'a Student Schedule item' and its unique identifier (or key) is [Student ID, Course Offering ID]. The requirements do not specify any other data attributes. The data group will occur once for every combination of Course Offering ID and enrolled Student ID, so this is one data group moved in one Exit.

In practice, if the Billing System is to produce an invoice for the Student, it will need more than just a Student ID; it will also need his name and address. The Billing System could, of course, access the C-Reg system to get that data.

However, suppose as an alternative to the requirement 4.1.2.2 e), that C-Reg is required to send the Student name and address, as well as the Student ID and Course Offering ID to the Billing System, how would that affect the size of this function process?

Answer

The output would now be two Exits. Why is the size greater?

The [Student ID, Course Offering ID] data group that is output resulting from requirement 4.1.2 occurs once for every combination of Course Offering ID and enrolled Student ID.

A Student's name and address are two data attributes of the 'Student' object of interest (This is fixed quite independently of whether the Student is enrolled or not in any Course Offerings). The Student ID, name and address attributes form a data group that occurs once for each 'Student'.

This difference in the frequency of occurrence tells us that C-Reg must output two different data groups to meet this requirement, i.e. there would be two Exits to the Billing System. (See Chapter 2 of the 'Guideline for sizing Business Application Software', v1.2, [2].

Question 2 re. 4.1.2

Why is the message sent by C-Reg about cancelled Course Offerings to the E-mail System for onward transmission to affected Students measured as two Exits when the E-mail system requires only three fields? These three fields could be interpreted as attributes of one object of interest 'E-mail'

Answer

C-Reg assembles an e-mail message to meet the formatting standards of the E-mail system. These specify that any message must have three fields, namely 'To' (a standard e-mail address), 'Subject' (header text), 'Message' (body text). From the viewpoint of the E-mail system, these are clearly three attributes of the one object of interest 'E-mail' (so it would process the message as one Entry).

However, the E-mail System is NOT the functional user of the C-Reg system for this requirement. The functional user (the 'intended recipient' of the data) is the Student identified in the 'To' field of the e-mail. This should be clear from the requirements which state that the e-mail message must include the Student's e-mail address and name ('John Doe' in the example).

This Functional User Requirement has two components: a) the information required by the Student, and b) the format of the transmitted information must comply with the standard of the E-mail System.

C-Reg, which assembles the message, outputs the Student e-mail address and name (attributes of the Student object of interest) and the list of one or more cancelled Course Offerings, as per the example in Question 1 above. From C-Reg's point of view it is outputting data describing two objects of interest (Student and Course Offering), hence it is outputting two Exits.

If the information output by C-Reg in this process were in the form of a printed report rather than a series of e-mails, then following COSMIC rules we would expect to count two Exits. The format of the output (report or e-mail) should not influence our analysis of the data movements of the software being measured which is based on the number of data group types that are output.

Question 3 re 4.1.2

In the 'Close Registration' functional process, C-Reg outputs a 'Enrolled Student/Course Offering' data group to the Billing System and a 'Cancelled Student Schedule item message' data group to the Student in the e-mail. These two data groups each describe the same object of interest – a Student Schedule item - i.e. a Student/Course Offering combination. The only difference is that one data group is for enrolled Students, the other is for Students whose Course offering has been cancelled; this is a difference of the value of the status attribute.

As the output to the Billing System and to the E-mail System describe the same object of interest, why identify two Exits?

Answer

The data groups required to be output to the Billing System and to the Student in the e-mail are not identical. The data group output to the Student contains information in text form (that C-Reg generates) that does not occur in the output to the Billing System. Further, the Billing System and the Student are two different functional users.

These two differences – of data groups and of receiving functional users – satisfy the conditions of the 'Data Uniqueness and possible exceptions' rule b), section 3.5.7 of the Measurement Manual [1], so that we must identify two different Exits.

5 C-REG SUMMARY FUNCTIONAL SIZE MEASUREMENT

No	Functional Process	CFP
1	Add a Professor	4
2	Enquire on a Professor	4
3	Modify a Professor	3
4	Delete a Professor	5
5	Enquire on Course Offerings (Professor)	7
6	Create Course Offering commitments	6
7	Modify Course Offering commitments	7
8	Delete Course Offering commitments	4
9	Add a Student's details	4
10	Enquire on a Student's details	4
11	Modify a Student's details	3
12	Delete a Student's details	4
13	Enquire on Course Offerings (Student)	6
14	Create a Student Schedule	6
15	Modify a Student Schedule	8
16	Delete a Student Schedule	6
17	Monitor Course Offering Enrolment progress	6
18	Monitor Student Schedule Enrolment progress	5
19	Close Registration	10
	Total size	**102**

Table 3: List of the functional processes and their sizes

Important

The above measurement results are based on the information available at the Requirements level. It is feasible that more information is added at the Specifications level which would add further data groups; this would most certainly impact the functional size. It is also plausible that the detailed specifications lead to additional functional processes and/or the addition of sub-processes. Again, this could lead to changes to the functional size in order to take into account these additions.

6 REFERENCES

All COSMIC documents are available for free download from www.cosmic-sizing.org.

[1] 'The COSMIC Functional Size Measurement Method v4.0.2: Measurement Manual. (The COSMIC Implementation Guide for ISO/IEC 19761:2017)' December 2017.

[2] 'Guideline for Sizing Business Application Software', v1.3, May 2017

APPENDIX

A.1 Version Control

Date	Reviewers (s)	Modifications / Additions
2003	IBM	The original version of this case study was documented in the Rational Unified Process (RUP Version 2003.06.00.65) document as an example of a Web site Project. The case study was reproduced with permission from IBM.
Most recent update of version 1: 2008-02-23	Many reviewers	The first version of this case study was published on 26th August 2004, copyright of the École de Technologie Supérieure, Montréal, Canada. This version went through several successive updates, the final one dated 23rd February 2008, all of which were published on www.cosmicon.com.
Version 2.0 December 2015	Arlan Lesterhuis, Alain Abran, Charles Symons Other members of the COSMIC Measurement Practices Committee Metrics specialists from Capgemini UK	Updated to comply with v4.0.1 of the COSMIC method, and partly re-written to simplify the requirements to make the whole document easier to understand by eliminating ambiguities. Some requirements have been removed from earlier versions and some new requirements added (in Chapter 4) to introduce new teaching points. The document has been re-structured so that each of the main Chapters 2, 3 and 4 contain the requirements followed by the analysis. These can be handed out separately for teaching purpose Copyright of the case has been transferred from ETS to COSMIC.
Version 2.0.1 April 2018	Arlan Lesterhuis, Alain Abran, Charles Symons Bruce Reynolds, Francisco Valdes Souto, Metrics specialists from Capgemini UK	Updated to comply with v4.0.2 of the COSMIC method. No substantive changes. Font changed from 10 to 11 pt.

A.2 Change requests, Comments, Questions

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org. You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

COSMIC Measurement Manual for ISO 19761

Expert System Measurement Case Study

Version 5.0

June 2022

FOREWORD.

Purpose of this document.

The COSMIC method measures a ‘functional size’ of software based on functional user requirements (FUR). The purpose of this document is to show that the functional size of the FUR of an expert system can be measured with the COSMIC Function Points method.

Editors:

Alain Abran, Ecole de technologie supérieure – University of Quebec (Canada),
Arlan Lestherhuis (The Netherlands).

Other members of COSMIC Measurement Practices Committee:

Jean-Marc Desharnais, Ecole de technologie supérieure – University of Quebec (Canada),
Peter Fagg, Pentad (UK),
Dylan Ren, Measures Technology LLC (China),
Bruce Reynolds, Tecolote Research (USA),
Hassan Soubra, German University in Cairo (Egypt),
Sylvie Trudel, Université du Québec à Montréal - UQAM (Canada),
Frank Vogelezang, IDC Metri (The Netherlands).

Table of Contents

1 FUNCTIONAL USER REQUIREMENTS.

An expert system is a computer system that emulates the decision-making ability of a human expert. It does so by:

- storing the rules used by an expert in decision-making, and
- providing logic to allow a client to use the same experts rules for making decision himself (using expert knowledge embedded into the expert system).

The following is an initial statement of requirements for a simple expert system.

For ease of understanding, maintenance and explanation, the expert system shall be rule-based.

Note. To ensure traceability and correspondence between the functional requirements and the measurement, the functional requirements are indicated by numbers in brackets that correspond to the numbers in the 'Req#' column of the measurement table.

1.1 Functional requirements for the client users.

(1) An expert system must be developed for retrieving possible holidays for clients:

The client user must answer a number of questions, such as:

- *the type of the holiday (beach, cruise, city travel, etc.),*
- *destination, and*
- *price.*

'Any of the above' is allowed as an answer.

When all questions have been answered the expert system will:

- *(2) display the questions and answers, and*
- *(3) list and provide details of the holidays that satisfy the entered answers, if any are found.*

(1,2,3) The client may subsequently repeat the functional process to see the effect of changing the input requirement parameters.

(4) The user may store any set of questions and the given answers for re-use, under a name to be entered.

(5) The expert system shall explain why it has posed a question when the user enters 'Why?' rather than an answer.

- *The expert system shall then show the rules in which the answer to this question is a factor.*

(6) When the expert system shows the holidays that satisfy the entered answers it shall explain how it derived these results when the user enters 'How?'

- *The expert system shall then show the rules that were used in deriving the results.*

(7) An error message shall be displayed in case no holiday satisfies the requirements. A confirmation message shall be displayed as the client needs assurance that the set of questions and answers has been saved.

1.2 Functional processes for the client user.

The functional processes to be expected are, amongst others:

- Specify the client's wishes by entering answers to the questions, store the questions and answers if desired, and show the holidays, if any, that satisfy the entered answers;
- Show the list of names of the previously stored question and answer sets;
- Show the questions and answers of a specific set;
- Allow the user to modify the answers to one or more questions and to show the resulting holidays that satisfy the modified answers (the 'what-if' scenario);

1.3 Functional requirements for the system administrator.

- Maintenance of holiday data (add, change, delete holiday data).
- List all holidays stored.
- Show data of a specific holiday.
- Maintenance of the rules (add, list, change, delete rules).
  - *List all rule names.*
  - *Show data of a specific rule.*

Note. Depending on the functionality of the expert system, there may be one or more functional processes for the drop-down lists for answering the questions.

2 SIZING OF THE FUNCTIONAL PROCESSES WITH COSMIC.

The first functional process for the client user is 'Show holidays'. We assume there will be one functional process to enable the client to enter his/her 'wishes' in answer to all questions, which has the following data movements.

2.1 Functional process 'Show holidays'.

Note. This functional process obviously has rule-processing logic which we assume accesses a set of persistently-stored rules to generate the list of recommended holidays from the entered set of holiday requirements and the stored holiday knowledge. This rule-processing logic, which is pure data manipulation, is associated with the 'Selected holiday details' Exit.

Functional size is 13 CFP:

DM	***Key Attribute(s)***	***Data Group***	***Req#***
Entry	*Question ID*	*Answer*	*1*
Read	*Rule ID*	*Holiday rule*	*1*
Read	*Holiday ID*	*Holiday details*	*1*
Exit	*Question ID*	*Question and answer*	*2*
Exit	*Holiday ID*	*Selected holiday details*	*3*
Entry	*Set name for client's questions and answers*	*Set name (if desired to store the questions and answers)*	*4*
Write	*Set name for client's questions and answers*	*Set name*	*4*
Write	*Set name, Question ID*	*Question and answer*	*4*
Entry	*Why*	*Selection criteria for rules involved in the Why request (user answered 'Why')*	*5*
Exit	*Rule ID*	*Rule involved in the Why request*	*5*
Entry	*How*	*Selection criteria for rules involved in the How request (user entered 'How' when seeing the holidays)*	*6*
Exit	*Rule ID*	*Rule involved in the How request*	*6*
Exit	*Error*	*Error/confirmation message*	*7*

The COSMIC Functional Size Measurement Method
Version 4.0.2

Sizing software in a Machine Learning context: A COSMIC Case study

Version 1.0

October 2019

Table of Contents

1

SIZING GENERIC SOFTWARE SUPPORTING MACHINE LEARNING

1.1 Introduction

COSMIC Function Points [1] quantify ('measure') the functionality of the requirements of software from many domains. This Case study presents an example of its applicability to measure the requirements of generic software that is a component of the Image Classifier software using a Machine Learning (ML) algorithm, as shown in Figure 2.1. It is based on [2].

This document segregates the functionality of the generic ('classical') software from the functionality specific to Machine Learning (ML) software [3]. This segregation will facilitate delegating tasks to staff with programming expertise, thereby freeing up time of ML data analysts. It also enables to collect data in a standardized fashion to develop estimation models for planning purposes and for on-going monitoring of the software tasks within a ML development project.

In ML, a neural network is a software application that can 'learn' to classify input data with the help of 'training examples' of that input data.

A. For example, a neural network can 'learn' to assign a handwritten digit the desired digit with – depending on the network – an accuracy of over 97%.
B. A training example is an example of an input together with its desired output.

There are many neural network architectures (and variants of these), and this Case study presents an example with a feedforward architecture (see Appendix A).

1.2 Objective of the software in this Case study

The generic software component of a feedforward neural network must be developed, the latter assigns and records the correct digit of images of separate individual handwritten digits, with a classification accuracy equal or above 98%. This software corresponds to an image classifier.

1.3 Context and facts

Starting points for the Case study are:

A. Images are available for training, each including a hand-written digit and its correct digit.
    - An image of a hand-written digit consists of 28x28 pixels.
    - The pixels are greyscale, with a value of 0.0 representing white, a value of 1.0 representing black, and in between values representing shades of grey.
    - All images are stored in a file.
B. Functional reuse of a pre-programmed feedforward network algorithm, including its Cost function. This feedforward algorithm software:
    1. assigns random values to the weights and biases on the basis of a mean and standard deviation;
    2. 'forward-propagates' the training images of a 'mini-batch' (a fixed number of training examples) and determines the cost (average deviation or 'error') of the actual and the desired output values of the training images in the mini-batch;

3. ‘backpropagates the changes to all weights and biases backwards through the layers in the network and stores the updated values; the algorithm determines the changes on the basis of the ‘cost’ in the output layer of the mini-batch just processed -;
4. processes all mini-batches of training examples, finishing an ‘epoch’ of training;
5. Repeats steps 2) to 4) for a specified number epochs of training.

C. The feedforward algorithm software stores the training parameters, so that it is possible to train anew with one or more changed parameters, the other parameters remaining the same.

D. The feedforward algorithm provides the storage of data it processed to meet the requirements of the generic software component of the Image Classifier software.

1.4 System requirements

This set of requirements is at the system level, it includes the data analyst requirements. The three graphs below are used to determine appropriate values of the hyper-parameters learning rate η, number of epochs and mini-batch size.

System Requirement 0: Images must be pre-processed.

Note. Since pre-processing is specific to each context, pre-processing is not documented here, its description and related measurement are not included within this case study. If such requirements were to be specified, the software functionality involved could be measured separately.

System Requirement 1: Initialize the feedforward network architecture

To initialize the feedforward network architecture, the data analyst must provide to the software application the following inputs: the number of layers, the number of units (‘neurons’) of each layer and, in the hidden layers, one weight per input and one bias.

The data analyst must be able to tune the network by varying these parameters of the network architecture.

System Requirement 2: Expand the number of images

The data analyst may ask the software to expand the number of images. The data analyst must then provide to the software ‘expansion instructions’. On the basis of an expansion instruction, the software must expand the number of images available for training by adding one distorted copy of each image.

Note. Expand instructions are not documented here, assume that the instructions will consist of a single data group.

System Requirement 3: Divide the images into three sub-sets

The software must divide the images into the three sub-sets of training, test and validation images, the members of which are randomly chosen. The data analyst inputs the sub-set names and the number of images of each sub-set. All images must be stored with their sub-set name.

System Requirement 4: Training step

After receiving from the data analyst the training parameters the software must start a training.

The data analyst provides to the software (a sub-set of) the training parameters: the mean and standard deviation for the weights and biases, sub-set name and the hyper parameters; in this case study the hyper-parameters are the learning rate η, the number of epochs and the mini-batch size.

During training the software calculates the classification accuracy of each epoch for each elapsed training time, and print these next to monitor the performance of learning.

System Requirement 5: Produce the graph 'Cost per epoch'

With help of this graph the data analyst can identify a suitable value of the learning rate η. After receiving the training parameters, among which the validation set, the neural network is executed and the software must display the graph 'Cost per epoch'.

The execution of the training is repeated with different learning rates provided by the data analyst.

The data analysts stops the training repetition when he identifies a suitable value of the learning rate η with the help of these graphs by comparing the rates of decrease of cost (i.e. average error between the desired and the actual output of training examples) – Figure 1.1.

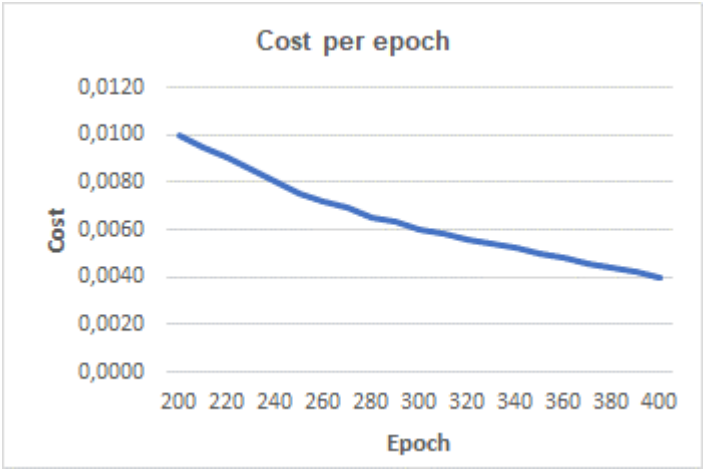


Figure 1.1 - Cost per epoch

System Requirement 6: Produce the graph 'Accuracy per epoch'

After a training step with the validation images, the software must produce the graph 'Accuracy per epoch'. The data analyst determines a suitable number of epochs for the training step with the help of this graph.

The data analyst will select the smallest number of epochs with which the required accuracy can be reached - Figure 1.2.

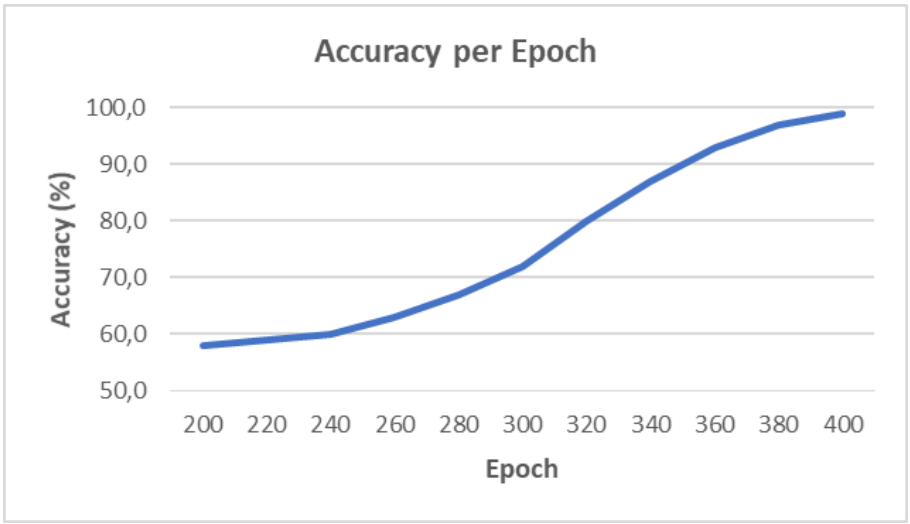


Figure 1.2 - Accuracy per epoch of training

System Requirement 7: Produce the graph 'Speed per mini-batch size'

The data analyst determines a suitable mini-batch size using this graph. The data analyst specifies the number of epochs and the number of intended mini-batch sizes, then the software must print the mini-batch size, the time and accuracy data on the graph in Figure 3:

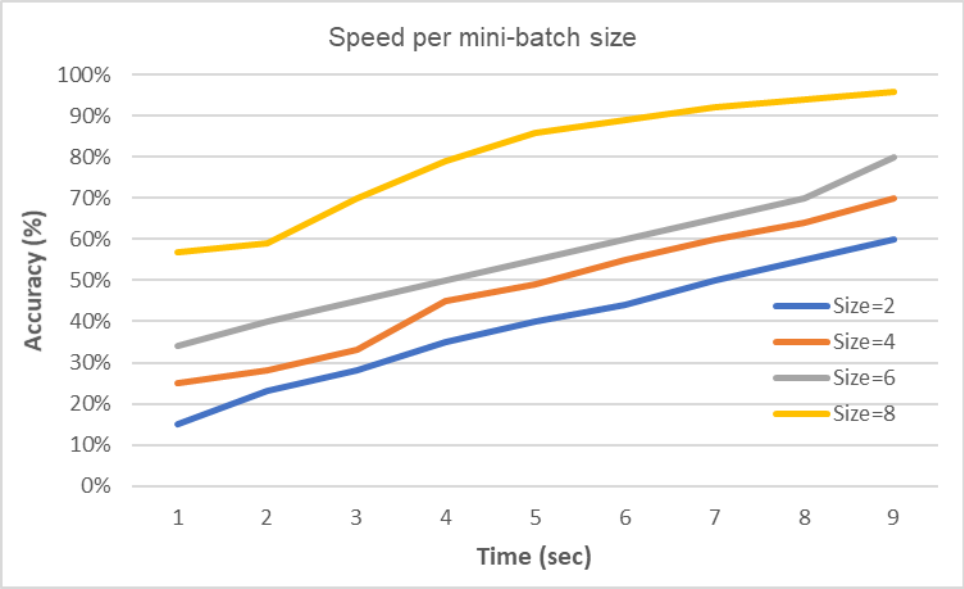


Figure 1.3 - Speed per mini-batch size

System Requirement 8: Error and confirmation messages

Each output must inform that an error has or has not occurred (but does not provide details - , in this case study the requirements for finding the errors are not specified).

System Requirement 9: Display of the x-axis and y-axis

The software must display a number of multiples on the x-axis and y-axis of graphs.

1.5 Software Requirements

The list of requirements of the generic software that follows includes the main data group names and corresponding attributes.

Software Requirement 1: Initialize the feedforward network architecture

The software receives a sequence of numbers to initialize the feedforward neural network architecture. The *length* of the sequence indicates the number of layers, each *number* in this sequence indicates the number of units of its layer, and each unit in the hidden layers has one weight per input and one bias.

Input. Data group for initializing the feedforward architecture: Number of units.

Result: Initialized feedforward architecture.

Software Requirement 2: Expand the number of images

The software receives the expansion instruction, then

1. copies each image,
2. changes it, and
3. adds the result to the images.

Input. Data group Expansion instruction (data attributes: not specified in this case study)

Output: Changed images, added as additional new images

Note. In the absence of details on 'expansion instruction' an assumption is made that this will consist of a single data group. If in other cases there is more in the 'expansion instructions', including more than one data group, this could then lead to additional data movements since there would be more data groups,

Software Requirement 3: Divide the images into three sub-sets

The images must be divided into the three sub-sets of 'training images', 'test images' and 'validation image'. The software receives the numbers of their members. The members of the sets must be randomly chosen. All images must be stored with their sub-set name.

Input. Data group Sub-set of images (attributes: sub-set name, number of images).

Output: The images with the sub-set names.

Software Requirement 4: Training step

The software receives the training parameters to start a training. The training parameters are mean and standard deviation, the name of the sub-set to be used and the hyper parameters (learning rate η, the number of epochs and the mini-batch size), or a sub-set of these training parameters. During training the classification accuracy per epoch and elapsed training time must be printed.

Input: Data group of training parameters

Results: A trained network and the data needed for the graphs stored. Data group: epoch ID, number of correctly classified images, training time.

Software Requirement 5: Produce the graph 'Cost per epoch' – Figure 1.1

The software receives the training parameters, on basis of which the graph 'Cost per epoch' must be produced.

Input. Data group of desired training parameters

Output: Data groups Epoch range (x-axis), Cost range (y-axis) and Cost per epoch

Software Requirement 6: Produce the graph 'Accuracy per epoch' – Figure 1.2

The software receives from the data analyst the first and the last epoch ID (epoch number) to be shown after a training step (during which the required data has been stored) and produces the graph.

Input: Data group epoch ID

Output: Data groups Epoch range (x-axis), Accuracy range (y-axis) and Accuracy per epoch

Software Requirement 7: Produce the graph 'Speed per mini-batch size' – Figure 1.3

The software receives from the data analyst a number of epochs and a number of intended mini-batch sizes and produces the graph 'Speed per mini-batch size'.

Input: Data groups the number of epochs and the number of intended mini-batch sizes

Output: Data groups Time (x-axis), Accuracy range (y-axis), Accuracy of mini-batch at point of time and mini-batch sizes legend for the graph.

The system requirements 8 and 9 of the error and confirmation messages and on the display of the x-axis and y-axis are taken into account in the measurement section 3.2.

1.6 The data model of the Image Classifier software

The entities and their attributes are as follows:

Feedforward architecture. Feedforward architecture ID, sequence of numbers (the length of which indicates the number of layers and each number of the sequence indicates the number of units of its layer)

Training step. Training step ID, mean, standard deviation, sub-set name, number of epochs, number of images per mini-batch, learning rate (η), values of the weights and biases.

Epoch. Epoch ID (= epoch number), cost, accuracy, elapsed time

Figure 1.5

2

MEASUREMENT STRATEGY

2.1 Measurement purpose

The purpose of the measurement *for this Case study* is to determine the functional size of the generic software, on the basis of its software functional specifications (i.e. excluding the specifications to develop the algorithms themselves that are being reused). In practice, the purpose would usually be to estimate the effort of implementing the network's generic software.

2.2 Measurement scope

The scope of the measurement consists of the software requirements of section 1.4.

2.3 Functional users of the generic software

The functional users are

- the data analysts of the neural network, i.e. those who tune the network with help of the generic software so as to meet the accuracy requirement.
- The reused feedforward neural network algorithm.

In this example, the reused software doesn't have to be measured: therefore, it is considered a functional user of the generic software component.

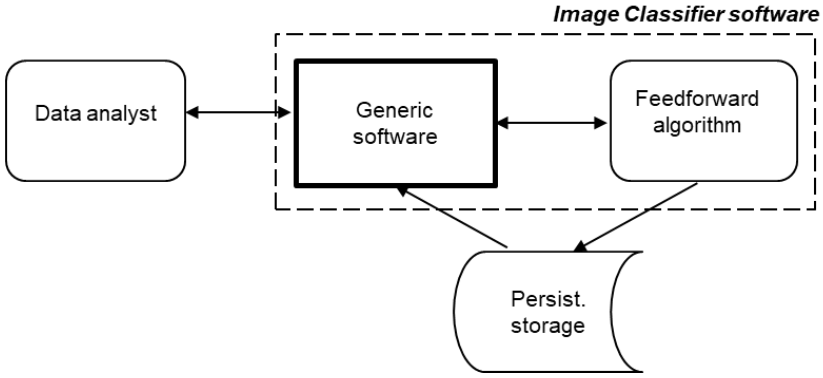


Figure 2.1 - Context diagram of the generic software

2.4 Other measurement strategy parameters

The functional users that initiate services of the generic software are individual data analysts of the neural network and hence the functional processes are individual functional processes.

3

THE MAPPING AND MEASUREMENT PHASES

3.1 The functional processes

Functional processes are the unique elementary parts of the requirements initiated by a functional user. In the requirements the following functional processes can be identified.

FP 1: Initialize of the feedforward network architecture

The software receives from the data analyst a sequence of numbers to initialize the desired neural network architecture.

FP 2: Expand the number of images

The software receives from the data analyst an expansion instruction to initiate expanding the images.

FP 3: Divide the images into three sub-sets

The software received from the data analyst the sub-set names and their number of images to initiate the division of the images into the sub-sets.

FP 4: Training step

The software receives from the data analyst the training parameters to initiate a training step.

FP 5: Produce the graph 'Cost per epoch'

The software receives from the data analyst a set of training parameters and produces the graph 'Cost per epoch' of the training step,

FP 6: Produce the graph 'Accuracy per epoch'

The software receives from the data analyst the first and the last epoch ID to produce the graph 'Accuracy per epoch' of the training step.

FP 7: Produce the graph 'Speed per mini-batch size'

The software receives from the data analyst the number of epochs and the number of mini-batch sizes to produce the graph 'Accuracy per epoch' of the training step.

3.2 Measurement of the feedforward neural network

The objects of interest of the data movements are in italics between [] after the corresponding data groups. Note that fixed text does not require movement of data.

FP 1: Initialize a feedforward network architecture

DM	Data Group/ Data attributes
Entry	Number of units [*Feedforward architecture*]
Exit	Number of units [*Feedforward architecture*], to feedforward algorithm
Entry	Error/Confirmation data [*Feedforward architecture*] from feedforward algorithm
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 4 CFP

FP 2: Expand the images

DM	Data Group/ Data attributes
Entry	Expansion instruction [*Expansion*]
Read	Image data [*Image*]
Write	Add distorted image data [*Image*]
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 4 CFP

FP 3: Divide the images into three sub-sets

DM	Data Group/ Data attributes
Entry	Sub-set of images (sub-set name, number of images [*Sub-set of images*]
Read	Image data [*Image*]
Write	Image data with sub-set name [*Image*]
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 4 CFP

FP 4: Train the network

DM	Data Group/ Data attributes
Entry	Training parameters (mean, standard deviation, number of epochs, number of images per mini-batch, learning rate (η), sub-set name) [*Training step*]
Exit	Training parameters [*Training step*] to feedforward algorithm
Entry	Epoch ID, number of correctly classified images, total number of images, elapsed time [*Epoch*] from feedforward algorithm
Exit	Print epoch ID, accuracy, elapsed time [*Epoch*]
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 5 CFP

FP 5: Display Cost per epoch (graph – Figure 1.1)

DM	Data Group/ Data attributes
Entry	Epoch ID [*Epoch range*] Input lowest and highest ID values
Read	Epoch ID, Cost [*Epoch*] stored by the feedforward algorithm
Exit	Epoch ID (x-axis, appropriate multiples) [*Epoch range*]
Exit	Cost (y-axis, appropriate multiples) [*Epoch Cost range*]
Exit	Epoch Cost [*Epoch*]
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 6 CFP

FP 6: Display classification accuracy per epoch (graph – Figure 1.2)

DM	Data Group/ Data attributes
Entry	Epoch ID [*Epoch range*] Input lowest and highest ID values
Read	Epoch ID, Epoch accuracy [*Epoch range*] stored by the feedforward algorithm
Exit	Epoch ID (x-axis, appropriate multiples) [*Epoch range*]
Exit	Epoch accuracy (y-axis, appropriate multiples) [*Epoch accuracy range*]
Exit	Epoch ID, Epoch accuracy [*Epoch*]
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 6 CFP

FP 7: Determine mini-batch size (graph – Figure 1.3)

DM	Data Group/ Data attributes
Entry	Training parameters (without number of images per mini-batch) [*Training step*]
Exit	Training parameters (without number of images per mini-batch) [*Training step*] to feedforward algorithm
Entry	Mini-batch size [*Mini-batch sizing*] Input the sizes to be compared
Exit	Mini-batch size [*Mini-batch sizing*] Sizes to be compared to feedforward algorithm
Read	Elapsed time, Epoch accuracy of mini-batch size [*Epoch*] stored by the feedforward algorithm
Exit	Elapsed time (x-axis, in seconds) [*Epoch time range*]
Exit	Epoch accuracy (y-axis: appropriate multiples) [*Epoch accuracy range*]
Exit	Mini-badge legend [*Mini-badge*] from Entry above
Exit	Epoch accuracy of last epoch at each second per mini-batch size [*Epoch accuracy per size*
Exit	Error/Confirmation message [*Error/Confirmation*]

Size: 10 CFP

The total functional size of the application is the sum of the sizes of its functional processes FP1 to FP7, that is:

4 CFP + 4 CFP + 4 CFP + 5 CFP + 6 CFP + 6 CFP + 10 CFP = **39 CFP**

REFERENCES

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

[1] Measurement Manual

[2] Neural networks and deep learning, M. Nielsen, Determination Press, 2015 (available at neuralnetworksanddeeplearning.com)

[3] Arlan Lesterhuis, Alain Abran, 'COSMIC Sizing of Machine Learning Image Classifier Software Using Neural Networks', 29th IWSM-MENSURA conference, Oct. 7-9, 2019, Haarlem, The Netherlands. Proceedings are online at http://ceur-ws.org/Vol-2476.

[4] Deep learning neural networks, D. Graupe, World Scientific, 2016

APPENDIX A – ARCHITECTURE, MACHINE LEARNING

A.1 The feedforward neural network architecture

The feedforward neural network consists of units (‘neurons’) organized in a number of layers:

- input layer: consists of units that encode the values of the input
- output layer: consists of units that indicate the outcome of the classification
- hidden layers in between

A unit in a layer receives input from all the units in the previous layer, multiplies each input with a ‘weight’ and add a ‘bias’ to the result.

Next an ‘activation’ (‘transfer’) function is applied to this result and the outcome is sent to each unit in the next layer.

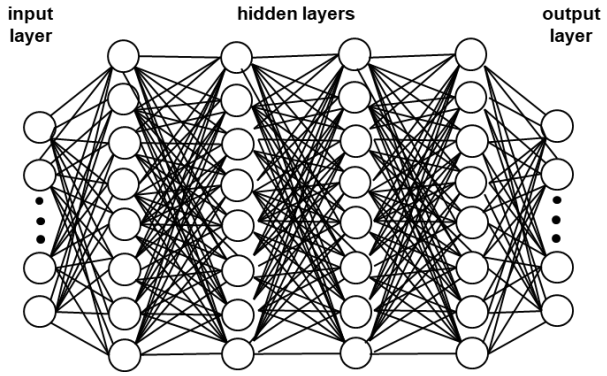


Figure A.1 – Architecture of a feedforward neural network

A.2 Machine learning

During training, a so-called cost function C of the neural network quantifies the average over the error of all individual training examples in a mini-batch. An example of a cost function is $C(w,b) = (1/2n).\sum_x(y(x)-a(x))^2$, where:

- n indicates the number of training examples in a mini-batch,
- x is a training example,
- y(x) its desired output,
- a(x) its actual output
- w indicates the weights in the hidden layers , and
- b the biases in the hidden layers.

During training each training example is input together with its desired value, the latter being used to determine the error between desired and actual output.

With the so-called backpropagation algorithm it appears possible to reduce the ‘average error’ C(w,b) by systematically and repeatedly adapting the weights and biases so that the output a(x) from the network approximates y(x) for all training inputs x: the neural network ‘learns’. The backpropagation algorithm is ‘the workhorse of learning in neural networks’ [2].

The purpose of the backpropagating algorithm is to find a global minimum of the cost function C, or at least a minimum of which the error is acceptable for the purpose of the application.

In practice it is useful to experiment with the sizes of the changes of the weights and biases supplied by the backpropagation algorithm. The sizes of the changes will then be multiplied with a positive factor η (eta), the 'learning rate'.

'Overfitting' is ineffective training, e.g., when the properties of the training set have been learned, rather than the software really has learned to classify. This parameter causes weights to be small so that incidental properties in the input will not easily be learned.

The classification accuracy can be improved by using more training examples. One way of expanding the training data is to add artificial training examples by applying an operation on the present training examples that reflect real-world variation and add the result as new training examples.

A.3 The training, the testing and the validation sets

Learning can take place on basis of 3 sub-sets of the training images, called the training set, the test set and the validation set. The training set is used for training the network, the test set for testing the result of training. The validation set is to be used to determine the values of the 'hyper parameters' learning rate (indicated by η), the size of the mini-batches to be used, and the number of epochs of training.

APPENDIX B - GLOSSARY OF TERMS

This Glossary contains terms that are specific to this Guideline. The Measurement Manual [1] contains the main Glossary of terms of the COSMIC method, In the definitions given below:

- terms are shown in **bold**.
- terms that are defined elsewhere in this Glossary are under-lined, for ease of cross-reference.

Bias. A measure of how easy it is to get a unit to output.

Classification accuracy. Percentage of correctly classified input items.

Cost. Average error between the desired and the actual output of training examples.

Deep neural network. A neural network consisting of more than one hidden layer

Epoch. A single pass through the full training set.

Hidden layer. A layer that is not an input layer or output layer.

Hyper-parameter. Special training parameter, e.g. the learning rate η, the number of epochs, the mini-batch size.

Input layer. The layer that encodes the values of the input.

Learning rate. A positive number (indicated by η) to be multiplied with the sizes of all the changes of weights and biases during learning.

Mini-batch. A fixed number of training examples used to update learning on the basis of the average cost (error) of the training examples in the mini-batch.

Neural network. A set of connected units, grouped in an input layer, an output layer and one or more hidden layers.

Neuron. A part of a neural network containing one weight per input and one bias.

Output layer. The layer that indicates the outcome of the classification.

Overfitting. Ineffective training (e.g. training that learns the properties of input data, rather than really improving classification).

Overtraining. Synonym of overfitting.

Test example. An example used to verify the accuracy of learning.

Training example. An example to learn from.

Training parameter. A parameter that determines the training; in this Case these are the mean and standard deviation of the weights and biases, the sub-set name and the hyper parameters.

Unit. Synonym of neuron.

Validation example. An example used to determine hyper-parameters of the neural network

Weight. A real number expressing the importance of the respective input to the output.

APPENDIX C - ESTIMATING WITH SOFTWARE SIZE

The main use of functional size is to provide an estimate of the effort of implementing an application on basis of its specification.

For estimating on the basis of functional size, both the size and the effort (i.e. the related number of work hours) of a number of applications must be captured. With help of a simple regression analysis ('Ordinary Least Squares'), the relationship between sizes and work hours can now be obtained and made visible in Excel. The relationship between sizes and work hours can be expressed by e.g. a linear function (see Figure B.1). This function can be used for

- estimation of effort of future implementations
- to serve as a second opinion to an estimate given by a project manager or a contractor.

With a new application size and the work hours of implementations the company data repository can be updated for estimations of future implementations.

As an example, suppose that the specifications of a number of applications below have been measured and realized, with the indicated implementation effort:

Application	Size (CFP)	Effort (hours)
Application1	50	190
Application2	90	355
Application3	45	165
Application4	60	315
Application5	85	370
Application6	120	365
Application7	30	195
Application8	65	295

These data result in the graph of Figure B.1, with the relationship between size and effort indicated by the trend line. The Figure also displays the formula of the trend line and the 'determination coefficient' R^2. The determination coefficient indicates the preciseness of the model, of which the maximum is 1. The closer to 1, the better the line fits the points.

As the formula is a 'best fit', it makes no sense to use all decimals of the indicated formula, it produces 'order of magnitude' estimates of effort needed to implement an application, given its sizes in CFP. Therefore the formula can be rounded, meaning that efforts can be estimated with help of the simple formula.

Effort (hours) = 2,5 * Size (CFP) + 109

As an example, with a measured size of a specification of 100 CFP, the estimated effort would become 2,5 * 100 + 109 = 359 hours.

Note that it is vital to realize that:

- The hours of all applications need to be collected for the same set of implementation activities. Each estimate is the estimate of the hours to perform *that* set of implementation activities.
- The formula is calibrated for the organization where these numbers were collected. An organization with a different development environment may get different efforts.
- The formula is calibrated for the available size range, i.e. don't use it for sizes far outside this range.
- The formula will change when new data are added.

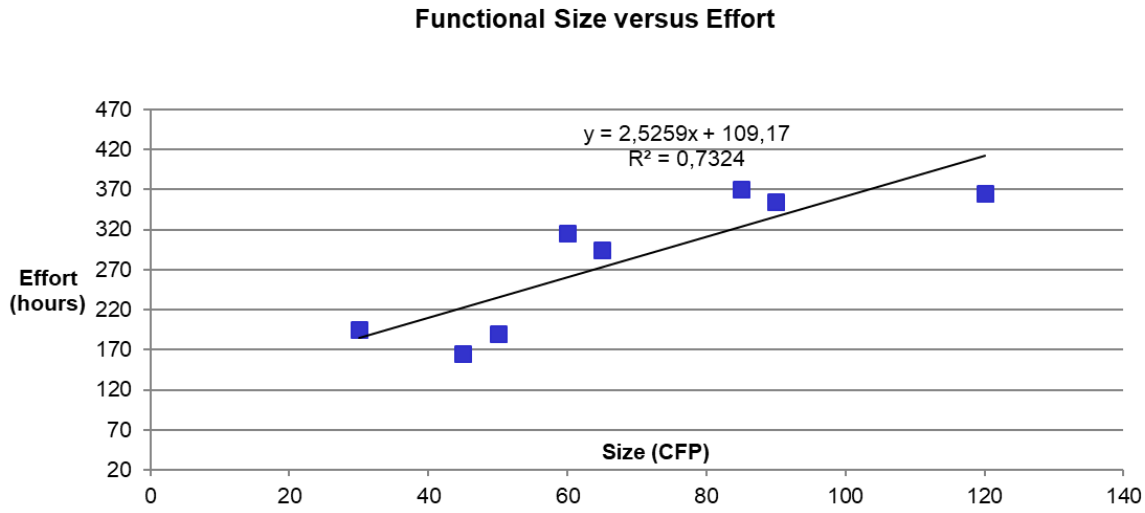


Figure C.1 – The estimation model

APPENDIX D – GENERAL INFORMATION

D.1 Acknowledgements

Version 1.0 authors and reviewers 2018 (alphabetical order)		
Alain Abran* École de Technologie Supérieure, Université du Québec, Canada	Arlan Lesterhuis* MPC, The Netherlands	Bruce Reynolds, Tecolote Research, USA

* Author(s) of this Guideline

D.2 Version control

The following table gives the history of the versions of this document.

DATE	REVIEWER(S)	Modifications / Additions
15-10-2019	COSMIC Measurement Practices Committee	First version 1.0 issued

D.3 Change requests, comments, questions

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

The COSMIC Functional Size Measurement Method

Version 4.0.2

Case Study

Sizing Natural Language/ User Stories/ UML Use Cases for Web and Mobile Applications using COSMIC FSM

Version 1.2

May 2019

Authors and reviewers of Version 1.1		
Asma Sellami * Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Arlan Lesterhuis* MPC Netherlands
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Hanêne Ben-Abdallah* Higher Colleges of Technology Dubai, UAE	Francisco Valdes Souto National Autonomous University of Mexico, Science Faculty, CDMX Mexico City, Mexico
Bruce Reynolds Tecolote Research United States		
Authors and reviewers of Version 1.0.3		
Asma Sellami* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Arlan Lesterhuis* MPC Netherlands
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Colin Hammond United Kingdom	Hanêne Ben-Abdallah* Higher Colleges of Technology Dubai, UAE
Francisco Valdes Souto National Autonomous University of Mexico, Science Faculty, CDMX Mexico City, Mexico	Bruce Reynolds Tecolote Research United States	
Authors and reviewers of Version 1.0.2		
Asma Sellami* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Sylvie Trudel * Université du Québec à Montréal Université du Québec Canada
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Arlan Lesterhuis* The Netherlands	Luigi Lavazza Universita degli Studi dell'Insubria Italy
Hanêne Ben-Abdallah* King Abdulaziz University Jeddah Kingdom of Saudi Arabia	Charles Symons* United Kingdom	

* Authors of this case study

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

Date	Reviewer (s)	Modifications / Additions
2016-05-20	Sellami, Haoues, Ben-Abdallah, Abran	This case study was entitled “Sizing the functional requirements in COSMIC FP units as documented in natural language/ UML use cases: A case study for Web and Mobile Apps” It was tested with graduate students at ETS
2016-06-01	Sellami, Haoues, Ben-Abdallah, Abran, Symons, Lesterhuis	Updates to: - propose a new title “Case study: Sizing natural language/ UML Requirements for Web and Mobile Applications using COSMIC FSM” - update the purpose of measurement - clarify the used terminolgy and avoid confusion - correct the measurement of the error/confirmation messages
2016-06-25	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis	Updates to: - identify Functional Processes - correct the use case diagrams and its related textual descriptions - identify object of interests and data groups
2016-07-03	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Trudel	Updates to: - identify object of interests and data groups - adjust layout of the document - replace “data validate” action type in UC textual description by “data persist”
2016-09-05	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Trudel, Lavazza	Significant revision. Updates to: - propose a new title “Case Study Sizing Natural Language/UML Use Cases for Web and Mobile Applications using COSMIC FSM” - adjust UC diagrams - explain the UC textual descriptions - clarify data groups and data attributes, and the description of some functional processes - delete some redundant tables - delete some functional processes
2017-11-10	Sellami, Haoues, Ben-Abdallah, Abran	Significant revision. Updates to: - Clarify the UC description and their associated FP in both mobile and web apps. - Follow the COSMIC method v4.0.2 (COSMIC, 2017) - Take into account some points raised by (Haoues *et al.*, 2017b) - Clarify which use cases are represented each with one functional process and which use cases are represented with two functional processes
2018-01-18	Sellami, Haoues, Ben-Abdallah, Abran, Colin Hammond	Minor revision. Updates to: - Breackdown REQ 9 into REQ 9 and REQ 10 in order to visualize separately each UC (Delete an order) and (View the list of orders), and that correspond respectively to (FP33) and (FP32) - Clarify the use cases description and their associated functional processes in web app s to avoid redundant actions. In particular, UC involving Modify or Delete object. - Adjust the total Functional Size of RestoSys according to the main changes.
2019-03-16	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Valdes Souto, Reynolds	Significant revision. Updates to: - Breackdown FP2 into FP2 –FP6 and FP3 into FP 7 –FP9 to correct the FP identification - Delete REQ 10, consider cheap meals as one of the item families and delete all its FP - Clarify the data exchanges between mobile and web app components for REQ1 and REQ3 - Update the data persistent storages - Rectify the objects of interests, data groups, and data attributes - Adjust the total functional size of RestoSys according to the main changes
2019-05-27	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis	Minor revision. Updates to: - Quantification of REQ using User Stories Formats - Identification of functional processes

Table of Contents

1 RESTAURANT MANAGEMENT SYSTEM REQUIREMENTS

1. 1 Context

This Case Study presents the measurement results of applying the COSMIC FSM method (Version 4.0.2) to the “Restaurant Management System”. This system is composed of hardware and software components. The software component named RestoSys includes two parts: a mobile app and a web app.

The “Restaurant Management System” requirements are documented in a technical report of the final project of study for the Professional Master's Degree at the University of Sfax-Tunisia (Mhadhbi 2013).

This case study is structured as follows:

- Chapter 1 presents the background for the “Restaurant Management System” case study. It provides different representations of Hardware/Software requirements. The focus of this chapter is especially on the description of the Functional Requirements as documented in natural language, user stories, UML use case diagrams, and the textual descriptions of use cases using action-type (see Appendix A).
- Chapter 2 presents the measurement strategy phase, including: measurement purpose and scope, functional users, level of granularity and decomposition.
- Chapter 3 presents the mapping phase including: triggering events, functional processes, data groups, data attributes, and objects of interest.
- Chapter 4 presents the measurement phase. The measurement results are provided in two ways:
  (i) a direct application of COSMIC FSM method on functional requirements described in natural language in section 4.1; and
  (ii) an application of COSMIC FSM method on the action-type of use case actions in section 4.2.

1. 2 Hardware Components

As shown in Figure 1, the hardware configuration of “Restaurant Management System” is composed of:

- A database server allowing a high storage capacity.
- A Web server that hosts the web app part of RestoSys.
- The admin interacts with RestoSys via a PC with Web browser, while the waiter interacts with the same system via a Smartphone.

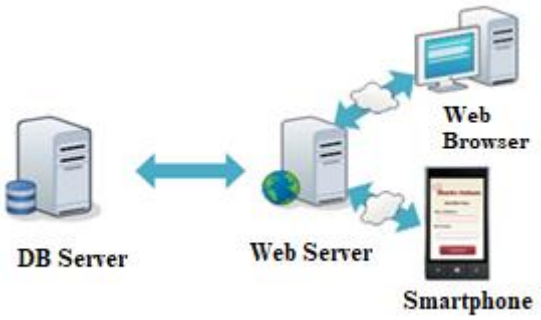


Figure 1 Hardware Configuration

1. 3 Software-Hardware Interactions

The RestoSys is composed of two parts: a mobile app and a web app, collaborating according to the client-server model:

- The RestoSys ensures communication between the mobile app (the waiter) and the web app (the admin).
- The admin physically maintains the database (that is included in the DB server) directly via a Web browser.

The RestoSys application includes the following functionality:

- The waiter logs in with help of the mobile app.
- The Web app retrieves data from the DB and provides the required data to the mobile app.
- The waiter maintains a customer's order (by adding or modifying an order).
- The admin logs in with help of the Web browser.
- The Web app retrieves/provides data from/to the DB.
- The admin maintains the required data via the web app, see the requirements in section 1.4 A.

1. 4 Software Requirements

According to (COSMIC 2017), software requirements are classified into three categories: Functional User Requirements (FUR), Non-Functional Requirements (NFR), and Project Requirements and Constraints (PRC). FUR express “what the software shall do in terms of tasks and services”. NFR include “any requirement for a hardware/software system or for a software product, including how it should be developed and maintained and how it should perform in operation, except any functional user requirement for the software”. PRC describe “how a software system project should be managed and resourced or constraints that affect its performance”. In this case study we will mainly use “Requirements” for convenience.

A. Requirements

In this section, the Requirements (REQ) for the RestoSys are identified. First, the REQ are described in natural language. Then, those same REQ are modeled in different formats including user stories and UML Use Case diagrams. Each use case in a UML Use Case diagram is next detailed in textual descriptions of use cases using action-type as presented in Appendix A.

a. REQ expressed in natural language

The RestoSys includes the following tasks:

- Order management: This task allows the waiter to add, and-or modify an order via a Smartphone. It also allows the admin to delete an order. During working hours, the waiters (mobile app) and the admin (web app) are continuously connected.
- Account management: This task involves employees’ management, and it enables access to the application with a username and a password.
- Restaurant Menu management: This task allows the management of item[1]families and the classification of items into item families.

Note that the users of RestoSys (waiter and-or admin) must be logged in before executing one of the previous tasks (Order management, Account management, and Restaurant Menu management).

The RestoSys must establish the following functionality:

[1]Item is used to describe a dish and beverage

- **REQ1 - Login "Mobile App":** The employees (waiters) must be logged in to RestoSys using their Smartphones. Each waiter has a username and a password.
- **REQ2 - Maintain Order:** The waiter can add and-or modify an order. The waiter takes an order by selecting the customer's table and the chosen items. Each customer is installed at a table and can request a set of items such as: fruit salad, orange juice, etc.
- **REQ3 - Login "Web app":** The employee (administrator) must be logged in to access the web app using a username and a password.
- **REQ4 - Maintain an Employee:** The admin can add an employee (waiter). The admin can view and-or modify a waiter data. The admin can delete an existing waiter and-or view the employees list.
- **REQ5 - Maintain Item:** The admin can add a new item by entering the necessary data for an item. The admin can also view, modify an item. The admin can delete an existing item and view the items list. Each item is classified into a single item family.
- **REQ6 - Maintain Item family:** The admin can add a new item family by entering the required data. The admin may also view and-or modify an item family data. The admin can delete an existing item family, and view the list of existing item families. Examples of item family: juice, salad, soda, etc.
- **REQ7 - Maintain Table:** The admin can add a new table. The admin may also view and-or modify a specific table data. The admin can delete an existing table and view the table list to know the table state (unoccupied, occupied or reserved).
- **REQ8 - View the List of Orders:** The admin can view the list of orders.
- **REQ9 - Delete an Order:** The admin can delete a customer's order.

These requirements will be detailed in the following sections using UML use case diagrams.

b. REQ expressed in UML Use case diagram

Identification of Actors. The employees of the RestoSys are the admin and the waiters.

- Admin: is the manager of the application. The admin can manage the entire RestoSys. The admin can access to the web app via his username and his password (REQ3).
- Waiter: is responsible for customers' orders. The waiter can access to the mobile app via his Smartphone and using his username and his password (REQ1).

Identification of User Stories and Use Cases. Based on the REQ listed in section A) a, the description of each of these REQ is presented below in the form of User Story (see Section B.) and Use case descriptions (see Section C).

B. User Stories Description

To make requirements visible not only to users but also to developers, the notion of splitting-up large sets of user stories into smaller stories is applied. Large US are identified with a high level of detail. Some of them are split into 'smaller' user stories. For example, "US2: Maintain Order" can be split into "US 2.1: Add an Order" and "US 2.2: Modify an Order". The user stories with their corresponding description are presented in Table 1.

Table 1 User stories Identification

Actor	REQ	User Story (US)	User Story Description
Waiter	**REQ1**	**US1 : "Login" Mobile app**	**As a** waiter **I want to** connect to the RestoSys **so that** I can use its services/ functionality
	REQ2	**US2: Maintain Order**	**As a** waiter **I want to** maintain orders **so that** I can add or modify a customer's order
		US2.1 : Add an Order	**As a** waiter **I want to** add an order **so that** I can add a new order as requested by the customer
		US2.2 : Modify an Order	**As a** waiter **I want to** modify an order **so that** I can modify an order as requested by the customer

	REQ3	**US3 : "Login" Web app**	**As an** admin **I want to** connect to the RestoSys **so that** I can use its services/ functionality
	REQ4	**US4: Maintain an Employee**	**As an** admin **I want to** maintain employees **so that** I can update the employees data/list
Admin		US4.1 : Add an employee	**As a**n admin I **want** to add an employee **so that** I can add a new employee to the employee list
		US4.2 : View the Employee List	**As an** admin **I want to** view the list of employees **so that** I can view the users list
		US4.3 : View an Employee data	**As an** admin **I want to** view an employee data **so that** I can view an employee data
		US4.4 : Modify an Employee Data	**As an** admin **I want to** modify an employee data **so that** I can modify an employee data
		US4.5 : Delete an employee	**As an** admin **I want to** delete an employee **so that** I can delete an employee
	REQ5	**US5 : Maintain Item**	**As an** admin **I want to** maintain an item **so that** I can update the items data/list
		US5.1 : Add an Item	**As an** admin **I want to** add an item **so that** I can add a new item
		US5.2 : View the Items List	**As an** admin **I want to** view the list of items **so that** I can view the items list
		US5.3 : View Item data	**As an** admin **I want to** view an item data **so that** I can view an item data
		US5.4 : Modify an Item	**As an** admin **I want to** modify an item data **so that** I can modify an item
		US5.5 : Delete an Item	**As an** admin **I want to** delete an item **so that** I can delete an item
	REQ6	**US6 : Maintain Item Family**	**As an** admin **I want to** maintain an item family **so that** I can update the items family data/list
		US6.1 : Add an Item family	**As an** admin **I want to** add an item family **so that** I can add a new item family
		US6.2 : View the Item families list	**As an** admin **I want to** view the item families list **so that** I can view the items families list
		US6.3 : View Item family data	**As an** admin **I want to** view an item family data **so that** I can view an item family data
		US6.4 : Modify an Item family	**As an** admin **I want to** modify an item family **so that** I can modify an item family
		US6.5 : Delete an Item family	**As an** admin **I want to** delete an item family **so that** I can delete an item family
	REQ7	**US7 : Maintain Table**	**As an** admin **I want to** maintain a table **so that** I can update the table data/list
		US7.1 : Add a table	**As an** admin **I want to** add a table **so that** I can add a new table
		US7.2 : View the tables List	**As an** admin **I want to** view the tables list **so that** I can view the tables list
		US7.3 : View table data	**As an** admin **I want to** view the table data **so that** I can view table data
		US7.4 : Modify table data	**As a** admin **I want to** modify table data **so that** I can modify table data
		US7.5 : Delete a table	**As an** admin **I want to** delete a table **so that** I can delete a table

	REQ8	**US8: View the list of orders**	**As an** admin **I want to** view the list of orders **so that** I can view the orders list
	REQ9	**US9: Delete an order**	**As an** admin **I want to** delete an order **so that** I can delete an order

C. Use Cases Description

The global use case diagram, as presented in **Figure 2**, identifies the main functionality provided by RestoSys. Thus, the employees (admin and waiters) must be logged in before maintaining data and orders. Each use case identified in the global use case diagram can include one or more use cases. As an example, the use case *"Maintain Data"* in the global use case diagram is detailed using four use cases: "Maintain Employee", "Maintain Item", "Maintain Item family", and "Maintain Table". The use case *"Maintain Order"* for the "Admin" is also detailed using "View the List of Orders" and "delete an order" (**Figure 3**).

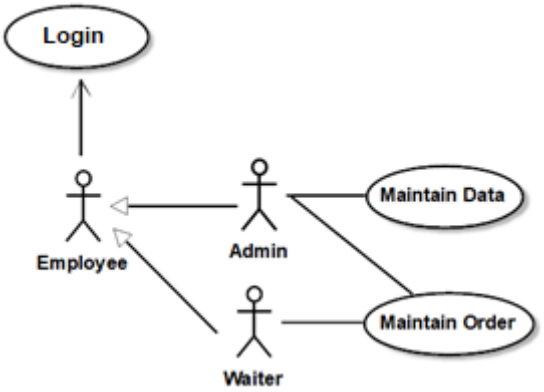


Figure 2 Global Use Case Diagram

Use Cases Textual Descriptions. Each detailed use case identified in **Figure 2** and **Figure 3** is represented using a use case textual description (see Appendix A). Note that a use case textual description represents a discrete unit of interaction between the system RestoSys (mobile app and web app) and its users (Waiter and Admin).

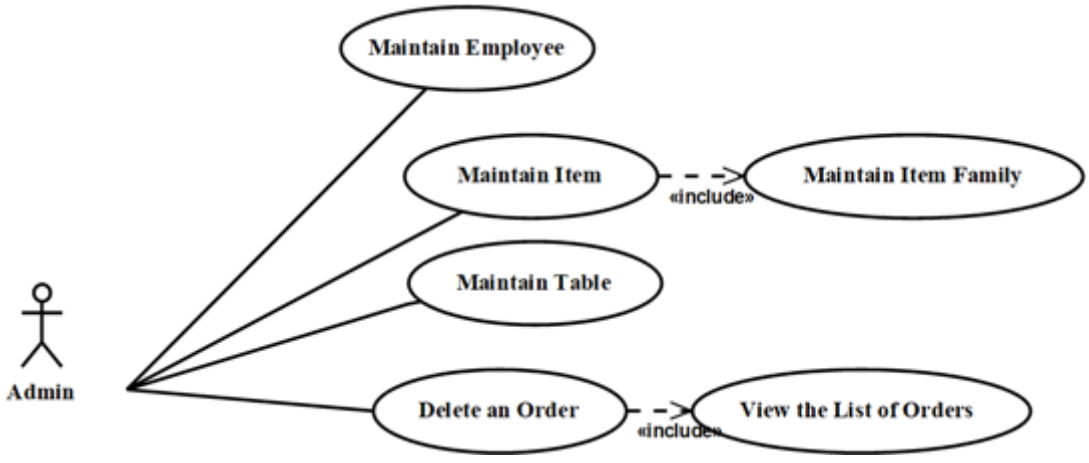


Figure 3 Detailed Use Case Diagram for "Maintain Data" and "Maintain Order" use cases

REQ1 Use case "Login" Mobile app

Name: <Login>
Description: <This use case describes how the employee (the waiter) login to the RestoSys>
Actors: <Primary actor: waiter>

Begin

MS /* Main scenario */

1. <Waiter><Expletive><The waiter requests for a connection to the RestoSys>
2. <System><Expletive><The RestoSys asks the waiter to enter his username and password>
3. <Waiter><Request><The waiter enters his/her username and password> [<Waiter ID>]
4. <System><Response & Request> <The RestoSys checks the validity of username and password> [<Waiter ID>] [<UserState>]
5. <System><Response><The RestoSys displays the user state (Connected / Not Connected) to the waiter> [<UserState>]

/*If the username and password are valid, the waiter is connected (user state = Connected)*/

/*Point 4 of login is detailed by using two actions: Response and Request. These actions lead to an exchange of data. For Response action, out-Parameters are required (username and password). For Request action, one Int-Parameter is required (user state). */

End

ES /* Error scenario */

Begin <Username and password are incorrect, begin at Num '3'>

6. <System><Request & Response> <The RestoSys displays the message "the username and/or the password are invalid">

End
End use case

REQ2: Use case "Maintain Order"

Name:<Maintain Order>
Description: <This use case allows the waiter to add or modify a customer's order>
Actors: <Primary actor: Waiter>

Begin

MS /* Main Scenario */

Add an Order

Begin

1. <System><Expletive><when the waiter logged in, the list of restaurant options is displayed>.
2. <Waiter><Expletive><The waiter presses the option "Add a new order">.
3. <Waiter><Request & Response><The waiter enters the table where the customer wants to be installed. The system changes the order state from active to closed > [<Table ID>&<Table ID>].

/*If a new order for the same table is entered, the previous order (if any) should be closed. In this case, there is always at most one order 'active')*/

4. <System><Request & Response><The system displays the list of item families> [<Item family data>] & [<Item family data>].
5. <Waiter><Request & Response><The waiter chooses the item family based on the items requested by the customer> [<Item family ID>&<Item familyID>].
6. <System><Request & Response><The system displays the list of items according to the selected Item families> [<Item Data>&<Item Data>].
7. <Waiter><Request & Response><The waiter selects the items requested by the customer and specifies for each item the recommended quantity and customer's comment. The system creates a new order> [<Order Items>] & [<Order Items>].

End

AS /*Alternative Scenario*/

Modify an Order

Begin

1. <System><Expletive><When the waiter logged in, the list of restaurant options is displayed>.
2. <Waiter><Expletive><The waiter presses the option “Modify an order”>
3. <Waiter><Request & Response><The waiter enters the table where the customer is installed> [<Table ID>&<Table ID>].
4. <System><Request & Response><The system displays the list of items ordered by the customer with their quantities and comments> [<Order Items> & <Item data> & <Order Items>].
5. <Waiter><Request & Response><The waiter deletes an existing item, modifies items' quantities or modifies items' comments. The system updates the order data > [<Order Items> & <Order Items>].

End

ES /* Error Scenario */

Begin<No table available, begin at Num '2' in the main scenario>
2.1 <System>< Request & Response><The system displays the message “No table available”>
/* The scenario restarts at point 1. */
End
Begin<Orders list is empty, begin at Num '2' in the alternative scenario>
2.1 <System>< Request & Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
Begin<No table available, begin at Num '7' in the main scenario>
7.1 <System>< Request & Response><The system displays the message “New order not created”>
/* The scenario restarts at point 1. */
End
Begin<No table available, begin at Num '5' in the alternative scenario>
5.1 <System>< Request & Response><The system displays the message “The changes cannot be saved”>
/* The scenario restarts at point 1. */
End

End

End use case

REQ3 Use case "Login" Web app

Name: <Login>
Description: <This use case describes how the admin logs into the Web app>
Actors: <Primary actor: Admin>

Begin

MS /* Main scenario */

1. <Admin><Expletive><The Admin requests for a connection to the system>
2. <System><Expletive><The system asks the Admin to enter his username and password>
3. <Admin><Request><The Admin introduces his username and password> [<Admin Data>]
4. <System><DataRecovery><The system checks the validity of username and password> [<Admin Data>]
5. <System><Response><The RestoSys displays the user state (Connected / Not Connected) to the admin> [<UserState>]

/*If the username and password are valid, the Admin is connected*/

End

ES /* Error scenario */

Begin <Username and password are incorrect, begin at Num '3'>

6. <System><Response><The system displays the message "the username and the password are invalid.">

End
End use case

REQ4: Use case "Maintain an Employee"

Name: <Maintain Employee>
Description: <This use case allows employees update>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario*/

Add an Employee

1. <Admin><Expletive><The admin requests to add a new employee>.
2. <System><Expletive><The system displays the employee form and requires the admin to enter the necessary data for adding new employee>.
3. <Admin><Request><The admin enters the employee data> [<Employee Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if admin is trying to add an existing waiter> [<Employee Data>].
6. <System><DataPersist><The system adds the new employee> [<Employee Data>].

AS /* Alternative Scenario */

View the Employees List

1. <Admin><Request><The admin selects to view the list of employees> [<Employee Data>].
2. <System><DataRecovery><The system retrieves the list of employees> [<Employee Data>].
3. <System><Response><The system displays the list of employees> [<Employee Data>].

View an Employee Data

1. <Admin><Request><The admin selects the desired employee> [<Employee ID>].
2. <System><DataRecovery><The system retrieves employee data from the Database> [<Employee Data>].
3. <System><Response><The system displays the data of the selected employee to the admin> [<Employee Data>].

Modify an Employee Data

/*Modify an Employee data should inherit all the default actions from View an Employee Data [1,2,3] */

1. <Admin><Request><The admin modifies employee data that can be changed> [<Employee Data>].
2. <System><DataPersist><The system saves the change> [<Employee Data>].

Delete an Employee

/*Delete an employee should inherit the default actions from View an Employee Data [1,2,3] */

1. <Admin><Request><The admin chooses the user to be deleted> [<username &<password>].
2. <System><DataPersist><The system deletes the selected user> [<username>, <password>].

ES /* Error Scenario */

Begin <Employee already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Employee already exists”>
/* The scenario restarts at point 2. */
End

Begin <The employees list is empty, begin at Num '3' in the *alternative scenario* 'View an Employees List'>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End

Begin <The employees list is empty, begin at Num '3' in the *alternative scenario* “Modify an Employee Data”>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End
Begin<The employees list is empty, begin at Num '3' in the *alternative scenario* “Delete an Employee”>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ5: Use case “Maintain Item”

Name: <Maintain item>
Description: <This use case allows the admin to add, modify, view and delete an item>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add an Item

1. <Admin><Expletive><The admin asks for adding a new item>.
2. <System><Expletive><The system displays the item form and asks the admin to enter the necessary data for adding an item>.
3. <Admin><Request><The admin enters the data of the item and instructs the system to validate the addition> [<Item Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if admin is trying to add an existing item> [<Item Data>].
6. <System><DataPersist><The system adds the new item> [<Item Data>].

AS /* Alternative Scenario */

View the Items list

1. <Admin><Request><The admin selects to view the items' list> [<Item Data>].
2. <System><DataRecovery><The system retrieves the list of items> [<Item Data>].
3. <System><Response><The system displays the list of items> [<Item Data>].

View an Item Data

1. <Admin><Request><The admin selects the desired item> [<Item ID>].
2. <System><DataRecovery><The system retrieves the item data from the Database> [<Item Data>].
3. <System><Response><The system displays the selected item data> [<Item Data>].

Modify an Item

/*Modify an Item should inherit all the default actions from View Item Data [1,2,3] */

1. <Admin><Request><The admin modifies the desired fields (name, price, quantity, image, etc.)> [<Item Data>].
2. <System><DataPersist><The system saves the change> [<Item Data>].

Delete an Item

/*Delete an Item should inherit the default actions from View Item Data [1,2,3] */

1. <Admin><Request><The admin selects the item to be deleted> [<Item ID>].
2. <System><DataPersist><The system deletes the selected item> [<Item ID>].

ES /* Error Scenario */

Begin<Item already exists, begin at Num '5' in the *main scenario*>

5.1 <System><Response><The system displays the message "Item already exists">

/* The scenario restarts at point 2. */

End

Begin<The Item list is empty, begin at Num '3' in the *alternative scenario* "Modify an Item">

3.1<System><Response><The system displays the message "The item list is empty">

/* The scenario restarts at point 1. */

End

Begin<The Item list is empty, begin at Num '3' in the *alternative scenario* "Delete an Item">

3.1<System><Response><The system displays the message “The item list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 6: Use case “Maintain Item Family”

Name: <Maintain Item Family>
Description: <This use case allows the admin to add, modify, view and delete an item family>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add an Item Family

1. <Admin><Expletive><The admin requests for adding a new item family>.
2. <System><Expletive><The system displays the item family form and asks the admin to enter the necessary data for adding an item family>.
3. <Admin><Request><The admin enters the data of the item family and instructs the system to validate the addition> [<Item Family Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if the Item Family already exists> [<Item Family Data>].
6. <System><DataPersist><The system adds the new item family> [<Item Family Data>].

AS /* Alternative Scenario */

View Item Families List

1. <Admin><Request><The admin selects to view the item families list> [<Item Families Data>].
2. <System><DataRecovery><The system retrieves the item families list> [<Item Family Data>].
3. <System><Response><The system displays the item families list> [<Item Family Data>].

View Item Family Data

1. <Admin><Request><The admin selects the desired item family> [<Item family ID>].
2. <System><DataRecovery><The system retrieves the item family data from the Database> [<Item family Data>].
3. <System><Response><The system displays the selected Item family data to the admin> [<Item family Data>].

Modify an Item Family

/*Modify an Item Family should inherit all the default actions from View Item Family Data [1,2,3] */
1. <Admin><Request><The admin modifies the desired fields> [<Item family Data>].
2. <System><DataPersist><The system saves the change> [<Item family Data>].

Delete an Item Family

/*Delete an Item Family should inherit the default actions from View Item Family Data [1,2,3] */
1. <Admin><Request><The admin selects the item family to be deleted> [<Item family ID>].

2. <System><DataPersist><The system deletes the selected item family> [<Item family ID>].

ES /* Error Scenario */

Begin<Item Family already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Item family already exists”>
/* The scenario restarts at point 2. */
End
Begin<The Item Family list is empty, begin at Num '3' in the *alternative scenario* “Modify an Item Family”>
3.1<System><Response><The system displays the message “The item family list is empty”>
/* The scenario restarts at point 1. */
End

Begin<The Item Family list is empty, begin at Num '3' in the *alternative scenario* “Delete an Item Family”>
3.1<System><Response><The system displays the message “The item family list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 7: Use case “Maintain Table”

Name: <Maintain Table>
Description: <This use case allows the admin to add, view, modify and delete a table>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add a Table

1. <Admin><Expletive><The admin requests to add a new table>.
2. <System><Expletive><The system displays the table form and asks the admin to enter the necessary data for adding a table>.
3. <Admin><Request><The admin enters all the table data> [<Table Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if the table already exists> [<Table Data>].
6. <System><DataPersist><The system adds the new table> [<Table Data>].

AS /* Alternative Scenario */

View Tables List

1. <Admin><Request><The admin selects to view the tables list> [<Table Data>].
2. <System><DataRecovery><The system retrieves the tables list> [<Table Data>].
3. <System><Response><The system displays the tables list > [<Table Data>].

View a Table Data

1. <Admin><Request><The admin selects the desired table> [<Table ID>].

2. <System><DataRecovery><The system retrieves the Table Data from the Database> [<Table ID>].
3. <System><Response><The system displays the data of the selected table to the admin> [<Table Data>].

Modify Table Data

/*Modify Table Data should inherit all the default actions from View Table Data [1,2,3] */
1. <Admin><Request><The admin modifies the table data> [<Table Data>].
2. <System><DataPersist><The system saves the change> [<Table Data>].

Delete a Table

/*Delete a Table should inherit the default actions from View Table Data [1,2,3] */
1. <Admin><Request><The admin chooses the table to be deleted> [<Table ID>].
2. <System><DataPersist><The system deletes the selected table> [<Table ID>].

ES /* Error Scenario */

Begin<Table already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Table already exists”>
/* The scenario restarts at point 2. */
End
Begin<The Table list is empty, begin at Num '3' in the *alternative scenario* “Modify Table Data”>
3.1<System><Response><The system displays the message “Tables list is empty”>
/* The scenario restarts at point 1. */
End
Begin<Tables list is empty, begin at Num '3' in the *alternative scenario* 'Delete a Table>
3.1<System><Response><The system displays the message “Tables list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 9: Use case “View the List of Orders”

Name: <View the List of Orders>
Description: <This use case allows to view the list of orders>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

View the List of Orders

1. <Admin><Request><The admin selects to view the orders list> [<Order Data>].
2. <System><DataRecovery><The system retrieves the orders list> [<Order Data>].
3. <System><Response><The system displays the orders list> [<Order Data >].

ES /* Error Scenario */

Begin<Orders list is empty, begin at Num '2' in the *main scenario*>
2.1 <System><Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
End

REQ 10: Use Case “Delete an Order”

Name: <Delete an Order>
Description: <This use case allows to delete a customer’s order>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Delete an Order

1. <Admin><Request><The admin selects the order to be deleted from the orders list> [<Order ID>].
2. <System><DataPersist><The system deletes the order> [<Order ID>].

End

ES /* Error Scenario */

Begin<Orders list is empty, begin at Num '2' in the *main scenario*>
2.1 <System><Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
End

D. NFR - Non-Functional Requirements

Non-functional requirements define some quality requirements such as: performance, usability and security. The mobile app interface must be easy to use, simple and clear. Moreover, the web app must be compatible with any operating system, easy to use, understandable; with a “good response time”. It is also evident that RestoSys (including web and mobile applications) requires the presence of an internet connection.

E. PRC - Project Requirements and Constraints

The PRC address types of constraints surrounding the software development project, such as the competing demands of time, cost, and scope limitations on the project resources needed; dependencies on other projects, data storage space, etc.

2 THE MEASUREMENT STRATEGY PHASE

2. 1 Measurement Purpose

The purpose of the measurement is to estimate the development efforts for the mobile app and the web app separately, based on the size of the RestoSys software according to its Requirements, as specified in Chapter 1 of this case study.

2. 2 Measurement Scope

The scope is all functionality as specified in the functional requirements for the software as provided in section 1.4 A. The RestoSys software is in the application layer. The Requirements (REQ) Measurement Purpose requires a decomposition of its software in a mobile app component and a web app component. Hence, because of the Measurement Purpose, a measurement scope for each of both components is defined.

2. 3 Identification of Functional Users

The human functional users of the RestoSys are the employee (admin and waiter).

- The Admin: is the manager of the application. The Admin is entitled to manage the entire RestoSys and specify users and their individual rights.
- The Waiter: is responsible for adding or modifying the customers' orders.

Note that the DB Server is not an actor (functional user) of RestoSys, as the RestoSys requirements do not require data to be stored or retrieved with help of other software (i.e. another functional user). Rather it is merely a storage device, i.e. the physical implementation of the web app's persistent storage.

Figure 4 presents the contextual diagram for the RestoSys showing all its functional users.

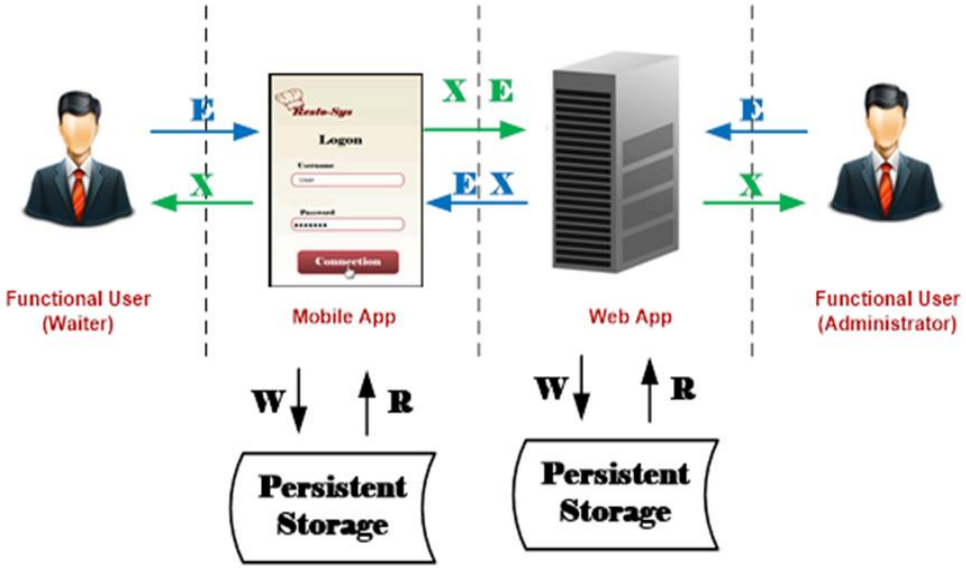


Figure 4 Contextual diagram

Figure 5 presents an example of data exchanges between mobile and web applications for REQ1 (Login for mobile app) and REQ3 (Login for web app). For example, login for mobile app is triggered by an Entry from the waiter which consists of the parameters (user name and password). The measurement details are provided in Section 4.1.

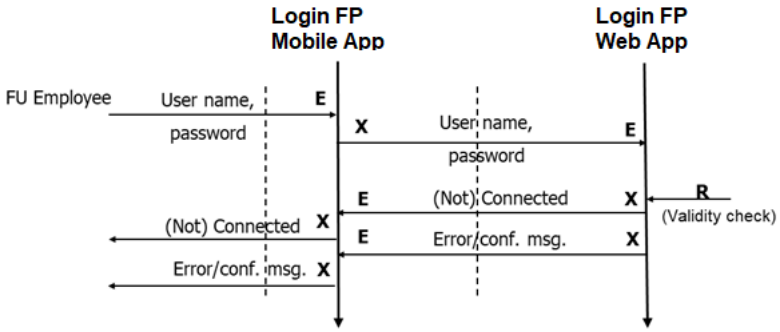


Figure 5 Data exchanges between Client (Mobile app) and Server (Web app) components For REQ1 and REQ3

2. 4 Other Measurement Strategy Parameters

A. Level of Granularity

The REQ of the RestoSys are at two levels of granularity. The first level of granularity is that of the Use Case global diagram (Figure 2). At this level, the data movements cannot be observed. For such cases, a COSMIC approximation method can be applied (COSMIC 2015b). However, since each use case identified in the first level can be detailed in the second level (such as Maintain Data which is detailed in Figure 3), the required level to apply COSMIC is where use cases are detailed using their textual description. More specifically, actions in a use case can be associated with the COSMIC data movement, as their level of granularity is the standard level, the 'functional process level of granularity'.

B. Decomposition

Figure 6 presents the decomposition for the RestoSys. The Mobile App interacts with the Web App via eXit/Entry data movements. The web app can require data to be stored or retrieved to/from persistent storage. Here, the architecture used to implement this system is 2-tier architecture. The two major components are: the Mobile app and the Web app.

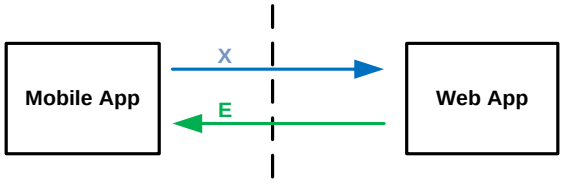


Figure 6 Decomposition

3 THE MAPPING PHASE

3. 1 Identification of the Functional Processes and the Software Triggering Events

From the REQ, the following triggering events are identified. For each functional process, there is only one triggering event (see Table 2). It is to be noted that when a functional process is required to obtain/save some data from/to another piece of software, it is the case of client/server' relationship. The REQ of the client component "Mobile App" can identify the server component "Web App" as one of its functional users, and vice versa. In addition, there is no standard mapping from a Use Case to a COSMIC Functional Process. Some Use Cases are mapped to several Functional Processes (e.g., the UC "Add an Order" is mapped to FP 2 and FP3). Other use cases are mapped to just one FP (e.g., the UC "Add an Employee" is mapped to FP8).

Table 2 Triggering Event Identification

REQ	Functional Processes	Triggering Events
REQ 1: Login "Mobile App"	FP 1: Login	The employee (waiter) needs to access to the login form
REQ 2: Maintain Order	FP 2: Add order's data	The waiter requires to add a new order
	FP 3: Create an order	The web app needs to create a new order
	FP 4: Modify order's data	The waiter requires to modify an existing order
	FP 5: Retrieve selected table data	The mobile app requires to retrieve the selected table data
	FP 6: Save modified data	The mobile app requires to save the modified data
REQ3: Login "Web app"	FP 7: Login	The employee (admin and waiter) needs to access to the login form (FP1 sends the login data (waiter data) to the web application for verification)
REQ 4: Maintain an Employee	FP 8: Add an employee	The admin requires to add an Employee (a waiter)
	FP 9: View the employees list	The admin needs to view the Employees list
	FP 10: View an employee data	The admin requires to view an Employee data
	FP 11: Modify an employee data	The admin requires to modify an Employee data
	FP 12: Delete an employee	The admin requires to delete an Employee
REQ 5: Maintain Item	FP 13: Add an Item	The admin requires to add a new item
	FP 14: View the items list	The admin requires to view the items list
	FP 15: View an Item data	The admin requires to view the item data
	FP 16: Modify an Item	The admin asks to modify an item
	FP 17: Delete an Item	The admin requires to delete an item
REQ 6: Maintain Item Family	FP 18: Add an Item Family	The admin requires to add a new item family
	FP 19: View Item Families List	The admin requires to view the item families list
	FP 20: View Item Family Data	The admin requires to view an item family data
	FP 21: Modify an Item Family	The admin asks to modify an item family
	FP 22: Delete an Item Family	The admin requires to delete an item family
REQ 7: Maintain Table	FP 23: Add a Table	The admin requires to add a table
	FP 24: View Tables List	The admin requires to view the tables list
	FP 25: View a Table Data	The admin requires to view a table Data

	FP 26: Modify Table Data	The admin asks to modify a table data
	FP 27: Delete a Table	The admin requires to delete a table
REQ 8: View the List of Orders	FP 28: View the List of Orders	The admin requires to view the list of orders
REQ 9: Delete an Order	FP 29: Delete an Order	The admin requires to delete an order

3. 2 Identification of Objects of Interest, Data Groups, and Data Attributes

From the REQ, seven objects of interest are identified. These are listed in Table 3 with their data groups and data attributes.

Table 3 List of Objects of Interest, Data Groups, and Data attributes

REQ	Objects of Interest	Data Groups	Data attributes	Example
REQ 1 REQ 3	Employee	Employee ID	username, password	username: alx84 password: 123
		UserState	userstate (Connected/Not Connected)	userstate: connected
		Employee Data	EmpNumber username, password, name, phone_number, address, job	EmpNumber: W2 username: alx84 password: 123 name: Alex phone_number: 22234567 address: Sfax, Tunisia job: waiter
REQ 2 REQ 7	Table	Table ID	table number	table number: 1
		Table Data	table number, state, capacity	table number: 1 state: occupied capacity: 2
	Set of Tables	Unoccupied Tables	table number, state, capacity	table number: 2 state: unoccupied capacity: 4
		All Table Data	table number, state, capacity	table number: 3 state: unoccupied capacity: 5
REQ 2 REQ 6	Item family	Item family Data	item family number, designation	item family number: 1 designation: Juice
				item family number: 2 designation: soda
				item family number: 3 designation: Cheap meals
REQ 2	Item	Item Data	item number,	item number: 1

REQ 5			item family number designation, image, price, quantity cheap price	item family number: 1 designation: orange juice image: orange.jpg price: 1 D quantity: 30 cheap price: 0.5D
				item number: 2 item family number: 2 designation: coca cola image: coca.jpg price: 2 D quantity: 100 cheap price: 1D
		Item ID	item number	item number: 1
REQ 2 REQ 7 REQ 8 REQ 9	Order	Order ID	order number	order number: 20
		Order Data	EmpNumber order number, table number, order state (active/closed), date, time	EmpNumber: W2 order number: 20 table number: 1 order state: active date: 05/07/2016 time: 8:00 pm
		Table ID	table number	table number: 1
REQ 2 REQ 6	Order Item	Order Items	order number, item number, quantity, comment	order number: 20 item number: 1 quantity: 3 comment: none item number: 2 quantity: 1 comment: with ice
		Item family ID	item family number	item family number: 1
From REQ 1 to REQ 9	Messages	E/C message	Message description	Message description: “Orders list is empty”

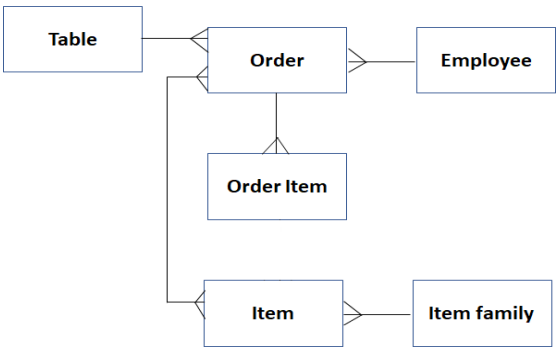


Figure 7 Data Model Diagram

Figure 7 presents the data model diagram for the RestoSys. Symbol **|** represents a one-and-only-one relationship. For instance, an item belongs to a single one item family. The crow’s foot symbol represents a one-or-many relationships. For instance, an employee manages multiple customers’ orders.

4 THE MEASUREMENT PHASE

4. 1 Functional Size Measured from REQ - Natural Language

A. For Mobile App

In this section, we give a detailed measurement of the functional size of the mobile app.

FP1: Login "Mobile App" **Triggering event: The employee (waiter) access to the login form**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	Enter username and password	Employee ID	Employee	E	1
Web app	The mobile app provides waiter data to the web app	Employee ID	Employee	X	1
Web app	The mobile app receives user state (Connected/Not connected)	UserState	Employee	E	1
Employee	The mobile app displays the userstate (Connected/Not connected)	UserState	Employee	X	1
Web app	The mobile app receives the Error/Confirmation message	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages from the web application	E/C Message	Messages	X	1
					Total size = 6 CFP

FP 2: Add Order's Data **Triggering event: The employee (waiter) adds a new order**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The waiter enters the table where the customer wants to be installed	Table ID	Order	E	1
Web app	The mobile app sends the selected table to the web app and then the web app creates a new order	Table ID	Order	X	1
Web app	The mobile app receives the item families list	Item family data	Item family	E	1
Employee	The mobile app displays the item families list to the waiter	Item family data	Item family	X	1

Employee	The waiter enters the item family based on the items requested by the customer	Item family ID	Order Item	E	1
Web app	The mobile app sends the selected item family to the web app	Item family ID	Order Item	X	1
Web app	The mobile app receives the item list of the selected family	Item Data	Item	E	1
Employee	The mobile app displays the list of items to the waiter	Item Data	Item	X	1
Employee	The waiter enters the items required by the customer and specifies for each item the recommended quantity and the customer comments	Order Items	Order item	E	1
Web app	The mobile app sends the selected items, recommended quantities and customer's comments to the web app	Order Items	Order item	X	1
Web app	The mobile app receives Error/Confirmation messages from the web app	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages to the waiter	E/C Message	Messages	X	1
					Total size = 12 CFP

FP 4: Modify Order's Data **Triggering event: The employee (waiter) modifies an existing order**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The waiter enters the table where the customer is installed	Table ID	Table	E	1
Web app	The mobile app sends the selected table to the web app	Table ID	Table	X	1
Web app	The mobile app receives the list of items previously selected by the customer with their quantities and comments	Order Items	Order item	E	1

FP 4: Modify Order's Data Triggering event: The employee (waiter) modifies an existing order					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The mobile app displays the items previously selected by the customer with their quantities and comments to the waiter	Order Items	Order item	X	1
Employee	The waiter deletes the existing item, modifies items' quantities and/or modifies items' comments	Order Items	Order item	E	1
Web app	The mobile app sends the modified items to the web app	Order Items	Order item	X	1
Web app	The mobile app receives Error/Confirmation messages from the web app	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages to the waiter	E/C Message	Messages	X	1
					Total size = 8 CFP

The total functional size of the mobile app is the sum of the sizes of its three functional processes, that is:

6 CFP + 12 CFP + 8 CFP = 26 CFP

B. For Web app

The web app includes 27 functional processes. In this section, we give a detailed measurement of the functional size of the web app.

FP3: Create an order Triggering event: The web app receives the selected items					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the selected items, quantities and customer comments	Order Items	Order Items	E	1
	The web app creates the new order	Order Data	Order	W	1
Mobile App	The web app sends Error/Confirmation messages to the mobile app	E/C Message	Message	X	1
					Total size = 3 CFP

FP5: Retrieve selected table data **Triggering event: The mobile app sends the selected table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the selected table	Table ID	Table	E	1
	The web app retrieves the list of item families selected by the customer	Order Items	Order Item	R	1
Mobile App	The web app sends the list of item families selected by the customer to the mobile app	Order Items	Order Item	X	1
	The web app retrieves the list of items selected by the customer	Item data	Item	R	1
	The web app sends the list of items selected by the customer	Item data	Item	X	1
					Total size = 5 CFP

FP 6: Save modified data **Triggering event: The mobile app sends the modified data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the modified items	Order Items	Order	E	1
	The web app updates the order data	Order Data	Order	W	1
Mobile App	The web app displays the Error/Confirmation message	E/C Messages	Message	X	1
					Total size = 3 CFP

FP7: Login **Triggering event: The employee (admin) access to the login form**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	Enter username and password	Employee ID	Employee	E	1
	The web app checks the validity of admin data	Employee ID	Employee	R	1
Employee	The web app displays the user state (Connected/ Not connected)	UserState	Employee	X	1

Employee	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP8: Add an Employee **Triggering event: The admin adds an employee**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the employee data	Employee Data	Employee	E	1
	The web app retrieves the employee's data to verify if the employee exists	Employee Data	Employee	R	1
	The web app adds the new employee	Employee Data	Employee	W	1
Admin	the web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP9: View the Employees list **Triggering event: The admin views the users list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects to view the employees list	Employee Data	Employee	E	1
	The web app retrieves the employees list	Employee Data	Employee	R	1
Admin	The web app displays the employees list	Employee Data	Waiter	X	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP10: View an Employee data **Triggering event: The admin asks for an employee data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the desired employee	Employee Data	Employee	E	1
	The web app retrieves user data from the Database	Employee Data	Employee	R	1

Admin	The web app displays the data of the selected waiter	Employee Data	Employee	X	1
					Total size = 3 CFP

FP11: Modify an Employee Data **Triggering event: The admin modifies employee data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin modifies employee data that can be changed	Employee Data	Employee	E	1
	The web app saves the change	Employee Data	Employee	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP12: Delete an Employee **Triggering event: The admin deletes an employee**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the employee to be deleted	Employee ID	Employee	E	1
	The web app deletes the selected employee	Employee ID	Employee	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP13: Add an Item **Triggering event: The admin adds a new item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the data of the item	Item Data	Item	E	1
	The web app retrieves the items data to verify if the admin adds an existing item	Item Data	Item	R	1
	The web app adds the new item	Item Data	Item	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP14: View the items list **Triggering event: The employees view the items list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The web app receives the waiter request	Item family ID	Order Item	E	1
Admin	The admin selects to view the items list	Item Data	Item	E	1
	The web app retrieves the items list	Item Data	Item	R	1
Admin/ Mobile app	The web app displays/sends the items list	Item Data	Item	X	1
Admin Mobile app	The web app displays/sends Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP15: View an Item data **Triggering event: The admin views an item data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired item	Item ID	Item	E	1
	The web app retrieves the item data from the Database	Item Data	Item	R	1
Admin	The web app displays the data of the selected item	Item Data	Item	X	1
					Total size = 3 CFP

FP16: Modify an item **Triggering event: The admin modifies an item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin modifies the desired fields (name, price, quantity, image, etc.) and asks the web app to update the data of the item	Item Data	Item	E	1
	The web app saves the change	Item Data	Item	W	1

Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP17: Delete an Item **Triggering event: The admin deletes an item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the item to be deleted	Item ID	Item	E	1
	The web app deletes the selected item	Item ID	Item	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP18: Add an item family **Triggering event: The Admin adds a new item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the data of the item family	Item Family Data	Item Family	E	1
	The web app retrieves the item families' data to verify if the admin adds an existing item family	Item Family Data	Item Family	R	1
	The web app adds the new item family	Item Family Data	Item Family	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP19: View Item families list **Triggering event: The employees view the item families list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The web app receives the waiter requests	Table ID	Order	E	1
Admin	The admin selects to view the item families list	Item Family Data	Item Family	E	1
	The web app retrieves the item families list	Item Family Data	Item Family	R	1

Admin/Mobile app	The web app displays/sends the item families list	Item Family Data	Item Family	X	1
Admin/Mobile app	The web app displays/sends Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP20: View Item family data **Triggering event: The admin views an item family data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired Item Family	Item Family ID	Item Family	E	1
	The web app retrieves the Item Family data from the Database	Item Family Data	Item Family	R	1
Admin	The web app displays the data of the selected Item Family to the admin	Item Family Data	Item Family	X	1
					Total size = 3 CFP

FP21: Modify an item family **Triggering event: The admin modifies an item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters modified fields (name, image, color)	Item family Data	Item Family	E	1
	The web app saves the change	Item family Data	Item Family	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP22: Delete an item family **Triggering event: The admin deletes an item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the family items to be deleted	Item family ID	Item Family	E	1
	The web app deletes the selected family item	Item family ID	Item Family	W	1

Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP23: Add a table **Triggering event: The admin adds a table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the table data	Table Data	Table	E	1
	The system checks if the table already exists	Table Data	Table	R	1
	The web app adds the new table	Table Data	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP24: View Tables list **Triggering event: The employee (admin or waiter) views the tables list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The mobile app selects to view the tables list	Unoccupied Tables	Set of Tables	E	1
Admin	The admin selects to view the tables list	All table Data	Set of Tables	E	1
	The web app retrieves the Tables list	Table Data	Table	R	1
Admin / Mobile app	The system displays / sends the Tables list	Table Data	Table	X	1
Mobile app / Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP25: View a table data **Triggering event: The admin views table data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired table	Table ID	Table	E	1
	The web app retrieves the Table data from the database	Table Data	Table	R	1

Admin	The web app displays the data of the selected table to the admin	Table Data	Table	X	1
					Total size = 3 CFP

FP26: Modify table data **Triggering event: The admin modifies table data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters modifies\d table data	Table Data	Table	E	1
	The web app saves the change	Table Data	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP27: Delete a table **Triggering event: The admin deletes a table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the table to be deleted	Table ID	Table	E	1
	The web app deletes the selected table	Table ID	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP28: View the List of Orders **Triggering event: The admin views the list of orders**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects to view the orders list	Order Data	Order	E	1
	The web app retrieves the orders data	Order Data	Order	R	1
Admin	The web app displays the orders list	Order Data	Order	X	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP29: Delete an order Triggering event: The admin deletes an order					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the order to be deleted	Order ID	Order	E	1
	The web app deletes the selected order	Order ID	Order	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

The total functional size of the web app is the sum of the sizes of its 26 functional processes, that is:

3 CFP + 5 CFP + 3 CFP + 4 CFP + 4 CFP + 4 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 3 CFP = 93 CFP

C. For the Whole System

The total size of RestoSys is then equal to the sum of functional sizes of mobile and web apps (26 CFP + 93 CFP) = 119 CFP.

4. 2 Functional Size Measured from REQ – US and UML UC Textual Description

A. Using User Stories Description

The functional size of RestoSys using User stories description are presented in Table 4. A large user story usually includes a number of Functional Processes. To quantify the size of RestoSys through user stories, Table 4 identifies the set of functional processes for each user story with their size measurement.

Table 4: Sizing RestoSys through US description with their corresponding functional processes

User Story (US)	Functional Processes	CFP	
US1 : "Login" Mobile app	FP 1: Login		**FS(US1) = 6 CFP**
US2: Maintain Order			**FS(US2) = 31 CFP**
US2.1 : Add an Order	FP 2: Add order's data	FS(FP2) = 12 CFP	**FS(US2.1) = 15CFP**
	FP 3: Create an order	FS(FP3) = 3 CFP	
US2.2 : Modify an Order	FP 4: Modify order's data	FS(FP4) = 8 CFP	**FS(US2.2) = 16CFP**
	FP 5: Retrieve selected table data	FS(FP5) = 5 CFP	
	FP 6: Save modified data	FS(FP6) = 3 CFP	
US3 : "Login" Web app	FP 7: Login		**FS(US3) = 4 CFP**

US4: Maintain an Employee		**FS(US4) = 17 CFP**
US4.1 : Add an employee	FP 8: Add an employee	FS(US4.1) = 4 CFP
US4.2 : View the Employee List	FP 9: View the employees list	FS(US4.2) = 4 CFP
US4.3 : View an Employee data	FP 10: View an employee data	FS(US4.3)= 3CFP
US4.4 : Modify an Employee Data	FP 11: Modify an employee data	FS(US4.4)= 3CFP
US4.5 : Delete an employee	FP 12: Delete an employee	FS(US4.5)= 3CFP
US5 : Maintain Item		**FS(US5)= 18 CFP**
US5.1 : Add an Item	FP 13: Add an Item	FS(US5.1)= 4CFP
US5.2 : View the Items List	FP 14: View the items list	FS(US5.2)= 5CFP
US5.3 : View Item data	FP 15: View an Item data	FS(US5.3)= 3CFP
US5.4 : Modify an Item	FP 16: Modify an Item	FS(US5.4)= 3CFP
US5.5 : Delete an Item	FP 17: Delete an Item	FS(US5.5)= 3CFP
US6 : Maintain Item Family		**FS(US6)= 18 CFP**
US6.1 : Add an Item family	FP 18: Add an Item Family	FS(US6.1)= 4CFP
US6.2 : View the Item families list	FP 19: View Item Families List	FS(US6.2)= 5CFP
US6.3 : View Item family data	FP 20: View Item Family Data	FS(US6.3)= 3 CFP
US6.4 : Modify an Item family	FP 21: Modify an Item Family	FS(US6.4)= 3CFP
US6.5 : Delete an Item family	FP 22: Delete an Item Family	FS(US6.5)= 3CFP
US7 : Maintain Table		**FS(US7)= 18 CFP**
US7.1 : Add a table	FP 23: Add a Table	FS(US7.1)= 4CFP
US7.2 : View the tables List	FP 24: View Tables List	FS(US7.2)= 5CFP
US7.3 : View table data	FP 25: View a Table Data	FS(US7.3)= 3CFP
US7.4 : Modify table data	FP 26: Modify Table Data	FS(US7.4)= 3CFP
US7.5 : Delete a table	FP 27: Delete a Table	FS(US7.5)= 3CFP
US8: View the list of orders	FP 28: View the List of Orders	**FS(US8)= 4CFP**
US9: Delete an order	FP 29: Delete an Order	**FS(US9)= 3CFP**
Σ size (US1, US2, US3, US4, US5, US6, US7, US8, and US9) = 119 CFP		

B. Using Action-Type

Table 4 presents the mapping of action-type in use case description with COSMIC data movements. Note that each use case can be associated with more than one functional process. As an example, the “Login” use case is represented by “FP1: Login”. Whereas, the use case “Add an order” is associated with FP2 and FP3.

Table 5 Equivalence between action-type in use case description and COSMIC concepts in terms of Data movements

Actions-types in Use case description	Actions-types description	COSMIC concepts	CFP
Action-type = Expletive	An action that does not lead to an exchange of data	Not applied	0 CFP
Action-type = Request	An action representing the act of asking for something, it is directed by an actor	An Entry data movement from the Functional user to the software to be measured	1 CFP
Action-type = Response	An answer or a reply sent after a Request, it is directed by the system	An eXit data movement from the software to be measured to the Functional user	1 CFP
Action-type = DataRecovery	An action that allows the retrieving of data	A Read data movement from the persistent storage to the software to be measured	1 CFP
Action-type = DataPersist	An action allowing the recording of data	A Write data movement from the software to be measured to the persistent storage	1 CFP

a. For mobile app

Table 6 presents the measurement results of the functional size of the mobile app based on the Action-Type.

Table 6 Measurement Results-Using Action-Type (mobile app)

Functional Requirements	Functional Processes	Sub-Process Description	Action Type	CFP
REQ 1: Login “Mobile App”	**FP 1: Login**	Enter username and password	Request	1
		The mobile app provides waiter data to the web app	Response	1
		The mobile app receives userstate (Connected/Not connected)	Request	1
		The mobile app displays the userstate (Connected/Not connected)	Response	1
		The mobile app receives the Error/Confirmation message	Request	1

		The mobile app displays Error/Confirmation messages from the web application	Response	1
				FS(FP 1: Login) = 6 CFP
	FP 2: Add order's data	The waiter enters the table where the customer wants to be installed	Request	1
		The mobile app sends the selected table to the web app and then the web app creates a new order	Response	1
		The mobile app receives the item families list	Request	1
		The mobile app displays the item families list to the waiter	Response	1
		The waiter enters the item family based on the items requested by the customer	Request	1
		The mobile app sends the selected item family to the web app	Response	1
		The mobile app receives the item list of the selected family	Request	1
		The mobile app displays the list of items to the waiter	Response	1
		The waiter enters the items required by the customer and specifies for each item the recommended quantity and the customer comments	Request	1
		The mobile app sends the selected items, recommended quantities and customer's comments to the web app	Response	1
		The mobile app receives Error/Confirmation messages from the web app	Request	1
		The mobile app displays Error/Confirmation messages to the waiter	Response	1
				FS(FP 2: Add order's data) = 12 CFP
	FP 4: Modify Order's Data	The waiter enters the table where the customer is installed	Request	1
		The mobile app sends the selected table to the web app	Response	1
		The mobile app receives the list of items previously selected by the customer with their quantities and comments to the mobile app	Request	1
		The mobile app displays the items previously selected by the customer with their quantities and comments to the waiter	Response	1

		The waiter deletes the existing item, modifies items' quantities and/or modifies items' comments	Request	1
		The mobile app sends the modified items to the web app	Response	1
		The mobile app receives Error/Confirmation messages from the web app	Request	1
		The mobile app displays Error/Confirmation messages to the waiter	Response	1
	FS(FP 4: Modify order's data) = 8 CFP			

The total functional size of the mobile app is the sum of the sizes of its three functional processes (FP1, FP2, and FP4), that is:

6 CFP + 12 CFP + 8 CFP = 26 CFP

b. For web app

Table 7 presents the measurement results of the functional size of the web app based on the Action-Type.

Table 7 Measurement Results - Using Action-Type (web app)

Functional Requirements	Functional Processes	Sub-Process Description	Action Type	CFP
	FP3: Create an order	The web app receives the selected items, quantities and customer comments	Request	**1**
		The web app creates the new order	DataPersist	1
		The web app sends Error/Confirmation messages to the mobile app	Response	1
	FS(FP3: Create an order) = 3 CFP			
	FP 5: Retrieve selected table data	The web app receives the selected table from the mobile app	Request	1
		The web app retrieves the list of item families selected by the customer	DataRecovery	1
		The web app sends the list of item families selected by the customer to the mobile app	Response	1
		The web app retrieves the list of items selected by the customer	DataRecovery	1
		The web app sends the list of items selected by the customer to the mobile app	Response	1
	FS(FP 5: Retrieve selected table data) = 5 CFP			
		The web app receives the modified items	Request	**1**

	FP 6: Save modified data	The web app updates the order data	DataPersist	1
		the web app displays Error/Confirmation messages	Response	1
			FS(FP 6: Save modified data) = 3 CFP	
REQ 3: Login "Web app"	**FP7: Login**	Enter username and password	Request	1
		The web app checks the validity of admin data	DataReovery	1
		The web app displays the user state (Connected/Not connected)	Response	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP7: Login) = 4 CFP	
REQ 4: Maintain an Employee	**FP8: Add an Employee**	The Admin enters the employee data	Request	1
		The web app retrieves the employee's data to verify if the employee exists	DataRecovery	1
		The web app adds the new employee	DataPersist	1
		The web app displays Error/Confirmation message	Response	1
			FS(FP8: Add an Employee) = 4 CFP	
	FP 9: View the Employees list	The admin selects to view the employees list	Request	1
		The web app retrieves the employees list	DataRecovery	1
		The web app displays the employees list	Response	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP 9: View the Employees list) = 4 CFP	
	FP 10: View an Employee Data	The admin selects the desired employee	Request	1
		The web app retrieves user data from the Database	Data-Recovery	1
		The web app displays the data of the selected waiter	Response	1
			FS(FP 10: View an Employee Data) = 3CFP	
	FP 11: Modify an Employee Data	The admin modifies employee data that can be changed	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP 11: Modify an Employee Data) = 3 CFP	
	FP 12: Delete an Employee	The admin enters the employee to be deleted	Request	1
		The web app deletes the selected employee	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1

	FS(FP 12: Delete an Employee) = 3 CFP			
REQ 5: Maintain an Item	**FP 13: Add an Item**	The admin enters the data of the item	Request	1
		The web app retrieves the items data to verify if the admin adds an existing item	Data-Recovery	1
		The web app adds the new item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP 13: Add an Item) = 4 CFP			
	FP 14: View the items list	The web app receives the waiter request	Request	1
		The admin selects to view the items list	Request	1
		The web app retrieves the items list	Data-Recovery	1
		The web app displays the items list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP14: View the items list) = 5 CFP			
	FP 15: View an Item data	The admin enters the desired item	Request	1
		The web app retrieves the item data from the Database	Data-Recovery	1
		The web app displays the data of the selected item	Response	1
	FS(FP 15: View an Item data) = 3 CFP			
	FP 16: Modify an item	The admin modifies the desired fields (name, price, quantity, image, etc.) and asks the web app to update the data of the item	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP16: Modify an item) = 3 CFP			
	FP 17: Delete an Item	The admin selects the item to be deleted	Request	1
		The web app deletes the selected item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP 17: Delete an Item) = 3 CFP			
REQ 6: Maintain Item family	**FP 18: Add an item family**	The admin enters the data of the item family	Request	1
		The web app retrieves the item families' data to verify if the admin adds an existing item family	Data-Recovery	1

		The web app adds the new item family	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP18: Add an item family) = 4 CFP			
	FP19: View Item families list	The web app receives the waiter requests	Request	1
		The admin selects to view the item families list	Request	1
		The web app retrieves the item families list	DataRecovery	1
		The web app displays the item families list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP19: View Item families list) = 5 CFP			
	FP20: View Item family data	The admin enters the desired Item Family	Request	1
		The web app retrieves the Item Family data from the Database	DataRecovery	1
		The web app displays the data of the selected Item Family to the admin	Response	1
	FS(FP20: View Item family data) = 3 CFP			
	FP21: Modify an item family	The admin enters modified fields (name, image, color)	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP21: Modify an item family) = 3 CFP			
	FP22: Delete an item family	The admin enters the family items to be deleted	Request	1
		The web app deletes the selected family item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP22: Delete an item family) = 3 CFP			
REQ 7: Maintain Table	**FP23: Add a table**	The admin enters the table data	Request	1
		The system checks if the table already exists	DataRecovery	1
		The web app adds the new table	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP23: Add a table) = 4 CFP			
	FP24: View Tables list	The mobile app selects to view the tables list	Request	1
		The admin selects to view the tables list	Request	1
		The web app retrieves the Tables list	DataRecovery	1

		The system displays the Tables list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP24: View Tables list) = 5 CFP			
	FP25: View a table data	The admin enters the desired table	Request	1
		The web app retrieves the Table data from the database	DataRecovery	1
		The web app displays the data of the selected table to the admin	Response	1
	FS(FP25: View a table data) = 3 CFP			
	FP26: Modify table data	The admin enters modified table data	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP26: Modify table data) = 3 CFP			
	FP27: Delete a table	The admin enters the table to be deleted	Request	1
		The web app deletes the selected table	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP27: Delete a table) = 3 CFP			
REQ 8: View the List of Orders	**FP28: View the List of Orders**	The admin selects to view the orders list	Request	1
		The web app retrieves the orders data	DataRecovery	1
		The web app displays the orders list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP28: View the List of Orders) = 4 CFP			
REQ 9: Delete an order	**FP29: Delete an order**	The admin selects the order to be deleted	Request	1
		The web app deletes the selected order	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP29: Delete an order) = 3 CFP			

The total functional size of the web app is the sum of the sizes of its 26 functional processes, that is:

3 CFP + 5 CFP + 3 CFP + 4 CFP + 4 CFP + 4 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 3 CFP = 93 CFP

The total size of RestoSys is then equal to the sum of functional sizes of mobile and web apps (26 CFP + 93 CFP) = 119 CFP, which is the same functional size as measured by referring to the documented REQ in natural language.

REFERENCE

COSMIC (2017) The Common Software Measurement International Consortium. 2017. The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual

COSMIC (2015a) Guideline on Non-Functional & Project Requirements : How to consider non-functional and project requirements in software project performance measurement, benchmarking and estimating.

COSMIC (2015b) Guideline for Early or Rapid COSMIC Functional Size Measurement by using approximation approaches.

(Haoues *et al.,* 2017a) M. Haoues, A. Sellami, and H. Ben-Abdallah, “Functional change impactanalysis in use cases: An approach based on COSMIC functional sizemeasurement,” Science of Computer Programming, Special Issue onAdvances in Software Measurement., 2017

(Haoues *et al.,* 2017b) M. Haoues, A. Sellami, and H. Ben-Abdallah. 2017. A Rapid Measurement Procedure for Sizing Web and Mobile Applications basedon COSMIC FSM Method. In Proceedings of 27th International Workshopon Software Measurement and 12th International Conference on SoftwareProcess and Product Measurement, Gothenburg, Sweden, October 25–27, 2017(IWSM/Mensura ’17), 9 pages.

Mhadhbi S (2013) Conception et Développement d’un Système de Gestion Restaurant Mobile.

APPENDIX A - STRUCTURED USE CASE DOCUMENTATION FORMAT USING ACTION-TYPE

Many individual proposals of textual descriptions are available to assist in documenting a Use Case (UC). We proposed an extension of the UC textual description with the “Type” of an action which can be: “Request”, “DataPersist”, “DataRecovery”, “Expletive”, or “Response” (Haoues *et al.*, 2017a). A “Request” corresponds to an action representing the act of asking for something, it is directed by an actor. A “Response” corresponds to an answer or a reply sent after a Request, it is directed by the system. A “DataPersist” action corresponds to an action allowing the recording of data. An “Expletive” action is used to get an action started but does not lead to an exchange of data, such as an action allowing data manipulation. “DataRecovery” action allows the retrieving of data.

The following documentation of a use case is provided in (Haoues *et al.*, 2017a).

Number:<unique ID assigned to a use case> **Name:**<unique name assigned to a use case> **Level:**<level of use case description> **Description:**<a summary of use case purpose> **Actors:**<Primary actor: actor that initiates the use case> <Secondary actor: actor that participate within the use case>
Pre-condition:<a list of conditions that must be true to initialize the use case> **Post-condition (success):**<state of the system if goal is achieved> **Post-condition (failure):**<state of the system if goal is abandoned> **Relationship** [**Include:**<use cases in relation with this use case by "include">, **Extend:**<use cases in relation with this use case by "extend">, **Super use case:**<list of subordinate use cases of this use case>, **Sub use case:** <list of all use cases that specialize this use case>] **Begin** **MS /* Main scenario */** <Steps of the scenario from trigger to goal> **Begin** <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] **End** **AS /* Alternative scenario */** **Begin**<Event, begin at Num "action number"> <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] The main scenario back to NUM **End** **ES /* Error scenario */** **Begin**<Event, begin at Num "action number"> <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] **End** **End Use Case**
Special requirements:<list of non-functional requirements>

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.

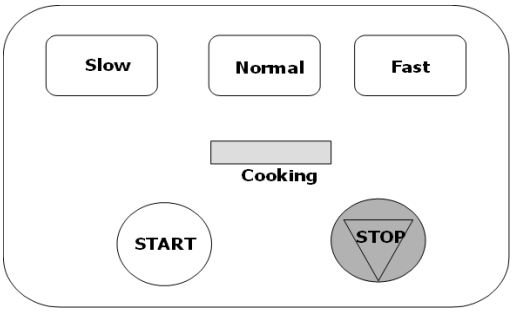


Figure 1 - The operator control panel

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.

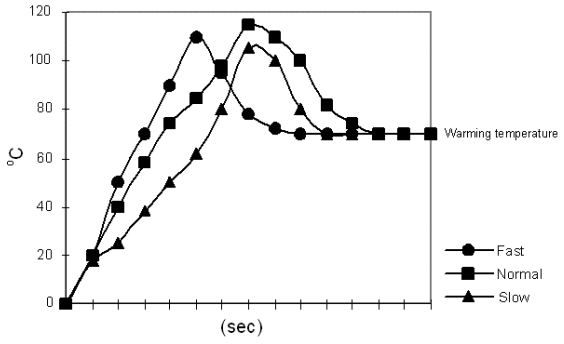


Figure 2 - Target Temperatures by Cooking Mode at 30 second Intervals

(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.

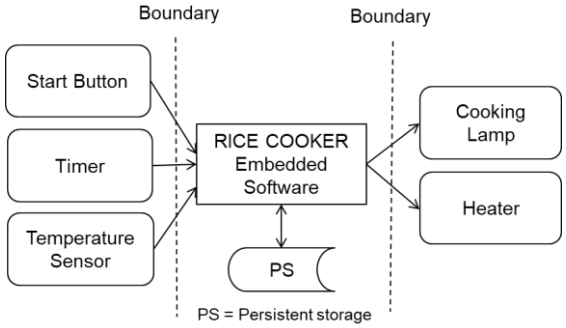


Figure 3 – Context diagram of the Rice Cooker software from its Functional User Requirements

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.

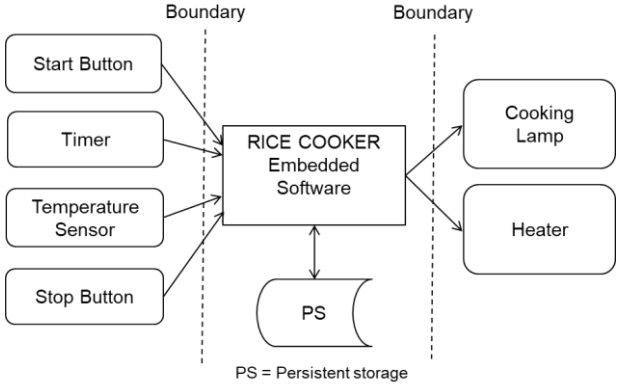


Figure 4 – Context diagram for the second prototype of the Rice Cooker software

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

The COSMIC Functional Size Measurement Method
Version 4.0.2

Industrial Automation (Robot) Case Study

Version 1.0

October 2018

Table of Contents

1

ROBOT FUNCTIONAL REQUIREMENTS

1.1 Introduction

1.1.1 The case

This case study illustrates the application of the COSMIC software measurement method (as defined in the Measurement Manual [1]) to measure the functional size of *user applications* for production tasks of industrial robots (short: robots) [2].

A main use of functional size is to estimate the effort of implementing a user application, as described in its specifications. For a concise overview of estimating based on functional size, see Appendix B, 'Estimating with software size'.

A production task (short: task) consists of connected activities to fulfil a specification (requirements) of an (industrial) process. If it is feasible to have a robot perform a task, software must be created for the robot to control execution of that task, a user application (short: application*)*. *Application software* is the umbrella name of the software that consists of all the robot's applications.

A robot performs one task at a time. When a task is finished, another task may be chosen to be performed by executing the application for that task, possibly at another location if the robot can be moved. Each application is separately entered into an existing robot by the customer or by a supplier of robot services, i.e. by a supplier that implements applications for customers.

Application software must be distinguished from the permanent *robot controller software,* which controls the sensors and the motors that are needed for the robot arm motions, as requested by the application. The robot controller (shortly: controller) consists of hardware and software and is a computer of its own. The controller software is supplied with the robot and is not part of this case (but can be measured if desired, if its specifications are available).

Robots exist in many shapes and sizes [3], the robot type here presented is one of the robot arm type as shown in Figure 1.1, but the approach discussed should be applicable to other robot types as well.

The requirements of a task that the robot has to perform can be described by a specification in the form of a *script*. A script describes the task of a robot as if the robot is a human actor performing the script. If useful, the script may further be explained and/or checked by adding e.g. a description of a finite state automaton.

1.1.2 Some terminology

A robot of the robot arm type comprises (see Figure 1.1, actuators and sensors not shown):

- An *end effector*, a device that may be used to grip objects or to attach tools or other devices that are to be manipulated, such as a welding torch, a cutter, a drill, a paint spray, etc. In Figure 1.1 the end-effector is a gripper.
- *Linkages* ('joints') enabling the end effector to follow a spatial path;
- *Actuators* like motors, effecting the motions of the joints and
- *Sensors*, for input to the application and the controller software and feedback of the control of the end effector and motors

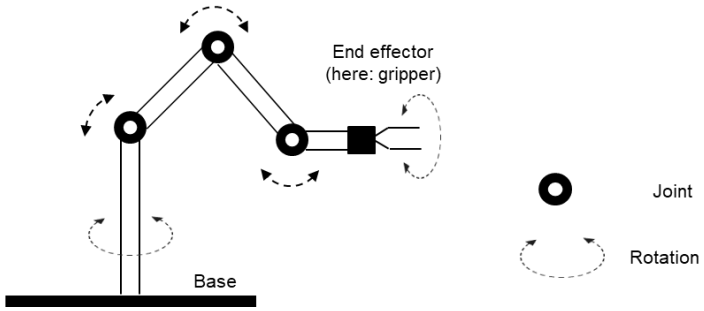


Figure 1.1 – An example robot arm

An application may be produced in a number of ways. One way is the 'teach-in' mode of programming. In this mode a number of positions and the required actions of the end effector are identified by moving the end effector and performing the actions manually (with the motors detached). These positions, actions and the manually added process parameters (in case of welding e.g. voltage, current, speed of movement) are stored by a 'motion recorder' as program instructions, which can be edited afterwards. Other modes are using a dedicated programming language or a simulation system.

1.2 Specifications

The specifications for a production task comprise the following descriptions [4]. For this case they are elaborated in the following sub-sections:

- functional specifications, a description of the task in the form of a script,
- a description of the robot type used, with its possibilities and limitations,
- a description of the environment in which the robot is to perform the task, often clarified by one or more pictures,
- performance and capacity specifications.

1.2.1 The script for this Case

The robot's task is to pick up nails from a bin, one at a time, and to put each one on a conveyor. The conveyor feeds a machine that fills nail magazines with 50 nails, which nail magazines are used in automatic nail guns for carpenters.

Before performing this task the robot and its environment must be initialized, i.e. it must be ensured that the robot and the devices in its environment are in working order. To do so, the robot arm moves to its 'start position' perpendicular to the belt, enabling to fill the bin from a nail stock (see Figure 1.2). The bin is filled and the sensors and motors checked. Each of them returns its state, which is either 'OK' or 'Error'. The application informs the operator on these outcomes via the operator PC. If all is OK, the processing can be started. If not, the application informs the operator on each malfunctioning device via the PC. After repair initialization must start anew.

A timer interval is manually set in the application to produce ticks with the time interval required by the task. After issuing the Start command the conveyor and the robot come to life. On the set timer tick, the robot picks one nail from the bin and put it on the conveyor. The nails on the belt must be put side by side and with the heads to one side.

The nails are thrown in the bin unorderly. Therefore, to pick up a nail the robot has to bring its gripper in a position that enables the gripper to pick up the selected nail, i.e. move the gripper to a nail and rotate the gripper transversal to the nail so that the gripper can seize it. For locating a nail and position the gripper, a video camera is mounted above the bin. Its image is processed by vision software, which selects one nail and conveys its position and orientation to the

application. The application than instructs the robot to bring the gripper in position to pick and place it.

The conveyor has a constant speed, which enables a fixed frequency of the timer ticks and prevents to implement control of timing of the interaction between the camera, the robot arm and the conveyor.

When the video camera software notes that the bin is empty and refill is needed, it informs the application. The latter stops the timer and orders the controller to move the robot arm to the start position, enabling the nail stock to fill the bin. When finished the nail stock notifies the application to resume the pick and place processing.

If the controller notifies that a nail rolled off, the application must stop processing, stop the conveyor and notify the operator (i.e. the PC) of an error. Also, if during processing a device or the controller finds that a situation needs attention (e.g. found by a sensor) the operator must be notified. The operator can then decide to stop processing immediately or stop in time. After repair, processing can be resumed by an operator command.

All communication (alarms, errors, confirmations, commands) between the application and its functional users vice versa takes place with help of messages. Messages contain a device ID, a code and date/time. The code depends on the device and is either a code for 'OK', or one of a number of codes for several sorts of alarms and errors, or one of a number of codes for commands.

For evaluation of the course of the task 'task info' must be stored and displayed on the PC. Task info consists of the relevant attributes of some of the messages:

- During Initialization all messages must be stored and displayed.
- During processing only the data of messages on alarms and errors and the Start and Stop commands must be stored and displayed.
- The number of nails processed must be stored on the PC.

1.2.2 The robot characteristics

Assumed is that the specifications fit within the possibilities and limitations of the robot.

1.2.3 The environment

Figure 1.2 schematically shows the robot in its environment, seen from above (the video camera above the bin not shown):

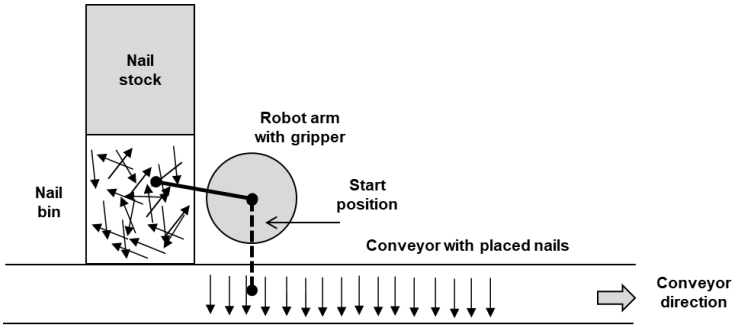


Figure 1.2 – The robot in its environment

1.2.4 Performance specifications

In practice, quantified performance specifications are important. These specifications relate to the required performance of both the devices and the execution of the task as a whole (cycle times of crucial parts of the task) and its capacity (its performance and error rates). They enable

to select critical hardware and software components in an early stage, i.e. before implementation. After implementation they enable verification whether or not the components really meet the specifications. Below a number of performance specifications have been added as examples.

Within the context of this case there is no relationship with the required functionality, therefore they may be skipped.

Nr.	Specifications	Min	Max	Typical	Unit
-	The video camera detects the nail tip with a positional accuracy of		3.0	1.0	mm
-	The video camera locates a new nail within	0.1	0.4		Sec
-	Cycle time: Pick and place a nail is finished within		1.0		Sec
-	Filling the bin is finished within		5.0		Sec
-	Nail rolled off error		0,1%	...	-
-	...	...	...	...	...

2

MEASUREMENT STRATEGY

This case is intended for study purposes, hence it is much more elaborate and detailed than measurements in practice are.

2.1 Measurement purpose

The purpose of the measurement *for this case* is to determine the size of the application on the basis of its functional specifications. *In practice*, the purpose would usually be to estimate the effort of implementing the robot task's application. This takes place on basis of its functional specifications and a registration of previous sizes and corresponding implementation efforts. For an explanation, see Appendix B, 'Estimating with software size'.

2.2 Measurement scope

The scope of the measurement consists of the functional specifications of the application of section 1.2.1. The PC's software, the controller software nor the vision software are part of the scope.

The functional specifications neither suggest nor necessitate decomposition of the application. The application therefore consists of one piece of software.

2.3 Identification of the functional users

The application does not directly interact with the sensors, motors and the robot arm and its gripper. Rather, it sends commands to the controller, which translates each command to the requested settings of the motors, resulting in the arm and gripper motions concerned. Conversely, the application receives the outcomes of the controller's processing. The robot controller is therefore a functional user of the application. The robot controller controls the supplied devices. If a device is controlled by the application itself it is a functional users of its own. In this case the video camera is assumed to be controlled by the application and is thus an example of the latter device.

In summary, the application has the following functional users, see Figure 2.1:

- PC, starts or stops the task and stores the required data;
- Timer, a tick of which is needed to start a new robot pick and place action;
- Conveyor, feeds a machine that processes the nails
- Video camera, its vision software translates an image of the bin to the location and position of a nail;
- Nail stock, from which the bin is filled;
- Controller, its software translates the application commands to robot motions and actions.

Figure 2.1 shows the context diagram of the application, its functional users and the boundary between them, based on the functional specifications of section 1.2.1.

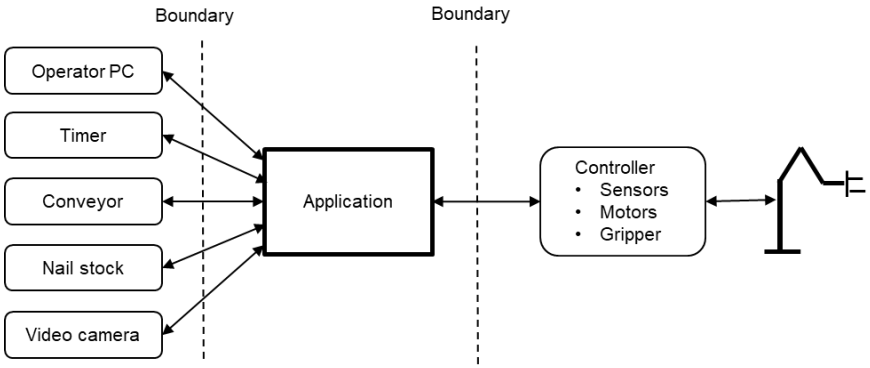


Figure 2.1 – Context diagram of the robot software

2.4 Other measurement strategy parameters

As the specifications show, the functional users are individual devices and software components, not groups of these, therefore the specifications for the application are at the 'functional process level of granularity'. The same is true of the events: single events occur that the application must respond to, not groups of events. In consequence, the application size is 'precise', i.e. not an approximation.

Data storage is delegated to the PC.

3

THE MAPPING AND MEASUREMENT PHASES

3.1 Identification of triggering events and their functional processes

In the script of section 1.2.1 the functional processes below have been identified, of which each name is in bold and its triggering event in italics.

Functional process 1: Initialization. *The operator issues the Start initialization command* via the PC, before the task can be started. On receipt, the application requests the states of the controller, the conveyor, video camera and nail stock device. If OK it orders the controller to move the robot arm to its start position, orders the nail stock to fill the bin, and sets the number of nails processed for a new series to zero. The devices respond with 'OK' or else with an error message. The application transfers the message to the PC and awaits intervention if one is not OK. The message data must be stored and displayed. If all is OK, the operator can start processing.

Functional process 2: Start Pick and place. *The operator issues the Start command.* On receipt the application starts the timer and the conveyor. It is issued to start after initialization or to resume after repair of an error.

Functional process 3: Pick and place. *On receipt of a timer tick* the application requests the video camera to return the position and orientation of a nail in the bin. On receipt the application orders the controller to pick and place this nail. It increases the number of processed nails with one.

Functional process 4: Fill bin. When *the video camera detects that the bin is empty*, it informs the application. The latter stops the pick and place actions by stopping the timer and orders the controller to move the robot arm to the start position. The application than orders the nail stock to fill the bin. The nail stock informs the application that the bin has been filled.

Functional process 5: Resume Pick and place. When *the application receives the message that the bin has been filled* the application restarts the timer and the conveyor, i.e. the robot resumes the pick and place actions.

Functional process 6: Nail rolled off. When *the gripper pressure sensor detects that a nail has rolled off the controller notifies this*, the application stops the timer and the conveyor, and notifies the PC of this error.

Functional process 7: Stop Pick and place. *The operator issues the Stop command.* The application stops the timer, orders the controller to move the robot arm to its 'start position' and stops the conveyor.

Functional processes 8 to 11: Attention. *If during processing a device or the controller notes a situation that needs attention* (e.g. measured by a controller sensor) this results in a message to the PC. The operator can than decide to stop processing immediately or stop in time. When an error has been repaired the operator issues the 'Start Pick and place' to continue processing.

3.2 Identification of objects of interest

As noted in the Measurement Manual [1], section 3.3.5, a functional user can be an object of interest as well, namely when the functional user provides or receives data about itself. In that

case data movements move data about the 'functional user object of interest'. Attributes of a functional user object of interest include its ID and its state. In consequence, a movement of data *to* such a functional user conveys the functional user*'s new* state or is a request to return its present state. A movement of data *from* the functional user conveys its *present* state.

All devices except the PC are 'functional user objects of interest' as they only provide or receive data about themselves. The PC issues commands and receives messages about the states of devices, not about itself.

The objects of interest (in bold) and their data attributes are as follows. The structure of the stored data is as displayed in Figure 3.1. Obvious states as 'Waiting', 'OK' or 'Error' have been left out.

Command. Command (values: Start Initialization, Start Pick and place, Stop Pick and place)

Timer. ID**,** State (value: Active)

Controller. ID, State (values: Moving arm to Start position, Checking sensors and motors, Picking and placing a nail, Nail rolled off)

Nail stock. ID, State (values: Filling bin, Filling finished)

Conveyor. ID, State (value: Moving)

Video camera. ID, State (value: Seeking nail, Nail selected, Fill bin needed)

Device state. Device ID, State (OK, Error MessageNr)

Task. Task ID, number of nails processed.

Task Info. Task ID, MessageNr, data from message

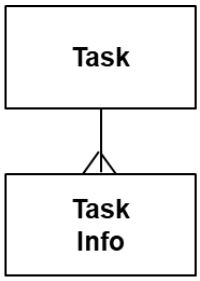


Figure 3.1 – Data model

3.3 The functional processes and their data movements

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

The following conventions have been used for describing the data being moved by the data movements:

- From PC to application Command
- From application to PC Device's present state
- From device to application Device's present state
- From application to device Device's next state or request to return present state

Functional process 1: Task initialization

DM	Object of interest	Data Group/ Data attributes
Entry	Command	Start Initialization
Exit	Controller	Request controller state
Entry	Controller	Controller state (of the sensors and motors, 'OK' or 'Error')
Exit	Nail stock	Request Nail stock state
Entry	Nail stock	Nail stock state ('OK' or 'Error')
Exit	Conveyor	Request conveyor state
Entry	Conveyor	Conveyor state ('OK' or 'Error')

Exit	Video camera	Request video camera state
Entry	Video camera	Video camera state (‘OK’ or ‘Error’)
Exit	Device state	State of controller, nail stock, convey or video camera to PC, for display and store
Exit	Controller	Move arm to Start position
Exit	Nail stock	Fill bin
Exit	Task	Number of nails processed to zero (to PC)

Size: 13 CFP

Functional process 2: Start Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Command	Start Pick and place
Exit	Timer	State(Active)
Exit	Conveyor	State(Move)
Exit	Command	Start Pick and place (to PC, for display and store)

Size: 4 CFP

Functional process 3: Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Timer	Timer tick
Exit	Video camera	Request nail location and orientation
Entry	Video camera	Receipt nail location and orientation
Exit	Controller	State(Pick and place)
Exit	Task	Add 1 to number of processed nails (to PC)

Size: 5 CFP

Functional process 4: Fill bin

DM	Object of interest	Data Group/ Data attributes
Entry	Video camera	Bin empty
Exit	Timer	State(Stop)
Exit	Conveyor	State(Stop)
Exit	Controller	State(Move arm to Start position)
Exit	Nail stock	State(Fill bin)

Size: 5 CFP

Functional process 5: Resume Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Nail stock	State(Bin filled)
Exit	Conveyor	State(Start)
Exit	Timer	State(Start)

Size: 3 CFP

Functional process 6: Nail rolled off

DM	Object of interest	Data Group/ Data attributes
Entry	Controller	State(Nail rolled off)
Exit	Timer	State(Stop)
Exit	Conveyor	State(Stop)
Exit	Controller	State(Nail rolled off) (to PC, for display and store)

Size: 4 CFP

Functional process 7: Stop Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Command	State(Stop Pick and place)
Exit	Timer	State(Stop)
Exit	Conveyor	State(Stop)
Exit	Controller	State(Move arm to Start position)
Exit	Command	Stop Pick and place (to PC, for display and store)

Size: 5 CFP

Functional processes 8, 9, 10 and 11: Attention

This are four almost identical functional processes (only their objects of interest differ), one for the conveyor, the video camera, the nail stock and the controller.

DM	Object of interest	Data Group/ Data attributes
Entry	Conveyor	State(Attention)
Exit	Conveyor	State(Attention) (to PC, for display and store)

Size:4 x 2 = 8 CFP

The total functional size of the application for this robot is the sum of the sizes of its functional processes, that is:

13 CFP + 4 CFP + 5 CFP + 5 CFP + 3 CFP + 4 CFP + 5 CFP + 8 CFP = **47 CFP.**

3.4 Proof of feasibility: the FUR as a finite state automaton

A measurement only makes sense if the FUR it is based on make sense. FUR make sense if they enable a feasible application. A finite state diagram of the FUR may give this insight.

Figure 3.1 shows the functional processes as states of a finite state automaton. Two states have been added (indicated by dashed lines). The first added state is 'Idle', the initial state of each production order. The other added state is 'Attention'. This is the state in which the application awaits the Start command of the operator after initialization or after repair of an error. It also is the state in which the application ends if a production order is finished. The arrows represent schematically the messages that cause the transition from one state to the next one (possibly the same one). For clarity, a few 'Error' arrows have been left out.

All sequences beginning in the 'Idle' state and ending in the 'Attention' state represent regular behaviour of the application, as specified by its FUR. The diagram thus shows that the FUR enable a feasible application.

Note. The states of the finite state automaton represent the states of the application as a whole. Do not confuse these states with the states in the analysis in the functional processes, which are states of devices.

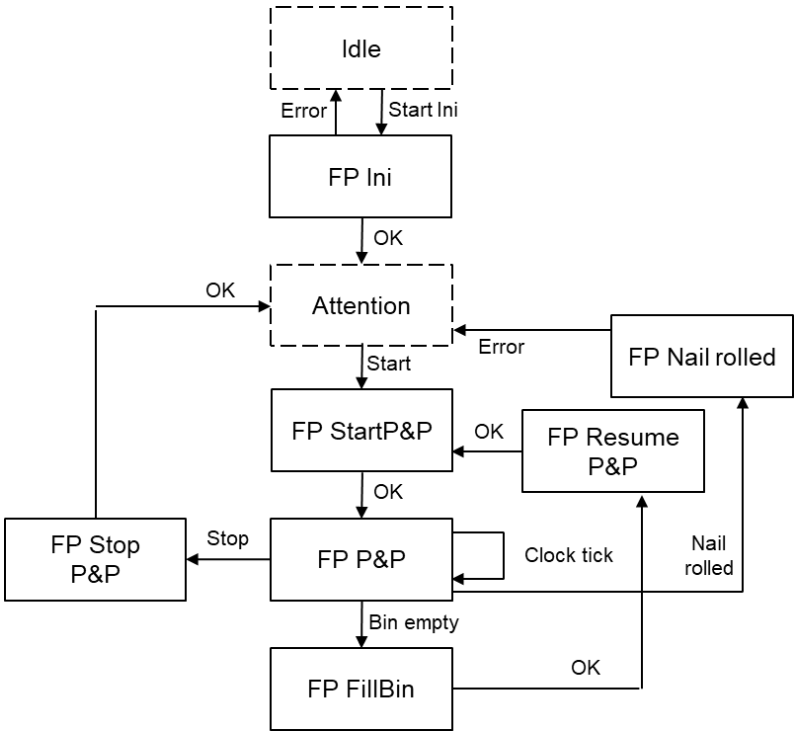


Figure 3.1 – The nail picking process as a finite state automaton

4

MEASUREMENT OF CHANGES

This chapter describes new requirements to be considered for changes because of two problems that came to light during tests of the robot actions. The size of each change of the application must be measured.

4.1 Changed requirements

During the acceptance test it appeared that the required maximum nail roll off rate of 0,1% by far wasn't reached. After a study of the handling of the nails it appeared that if a nail was put on a moving conveyor, it sometimes rolled off the conveyor, causing a 'nail rolled off' stop. These stops caused the excessive error rate, which in addition required undesirable human intervention.

It was decided to not implement a continuously running conveyor but rather to keep the conveyor stationary and move it over a short distance just after the nail has been put. The distance to be moved, 50 mm (hereafter called 'a step'), and the stop after that motion is set in the conveyor's controller. The application issues the (unchanged) move command to the conveyor, which than moves it one step.

4.2 Measurement of this change.

The functional processes affected by the change are listed below, with text to be removed struck out, text to be added or changed underlined.

Functional process 2: Start Pick and place. *The operator issues the Start Pick and place command.* On receipt the application starts the timer ~~and the conveyor~~. It is issued to start after initialization or to resume after repair of an error.

Functional process 3: Pick and place. *On receipt of a timer tick* the application requests the video camera to return the position and orientation of a nail in the bin. On receipt the application orders the controller to pick and place this nail and <u>moves the conveyor one step</u>. It increases the number of processed nails with one.

Functional process 6: Nail rolled off. When *the gripper pressure sensor detects that a nail has rolled off, the controller notifies this*, the application stops the timer ~~and the conveyor~~ and notifies the PC of this error.

Functional process 7: Stop Pick and place. *The operator issues the Stop Pick and place command.* The application stops the timer, orders the controller to move the robot arm to its 'start position' ~~and stops the conveyor~~.

Functional process 2: Start Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Command	Start Pick and place
Exit	Timer	State(Active)
~~Exit~~	~~Conveyor~~	~~State(Move)~~

Exit	Command	Start Pick and place (to PC, for display and store)

Starting the conveyor removed. Size of the change: 1 CFP

Functional process 3: Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Timer	Timer tick
Exit	Video camera	Request nail location and orientation
Entry	Video camera	Receipt nail location and orientation
Exit	Controller	State(Pick and place)
Exit	Conveyor	State(Move one step)
Exit	Task	Add 1 to number of processed nails (to PC)

Moving the conveyor one step added. Size of the change: 1 CFP

Functional process 6: Nail rolled off

DM	Object of interest	Data Group/ Data attributes
Entry	Controller	State(Nail rolled off)
Exit	Timer	State(Stop)
~~Exit~~	~~Conveyor~~	~~State(Stop)~~
Exit	Controller	State(Nail rolled off) (to PC, for display and store)

Stopping the conveyor removed. Size of the change: 1 CFP

Functional process 7: Stop Pick and place

DM	Object of interest	Data Group/ Data attributes
Entry	Command	State(Stop Pick and place)
Exit	Timer	State(Stop)
~~Exit~~	~~Conveyor~~	~~State(Stop)~~
Exit	Controller	State(Move arm to Start position)
Exit	Command	Stop Pick and place (to PC, for display and store)

Stopping the conveyor removed. Size of the change: 1 CFP

The total size of the change is 4 CFP.

5

COOPERATING ROBOTS

In many production situations multiple, cooperating robots are put on, in order to speed up performance or to perform a number of tasks simultaneously [6], [7]. This chapter describes the application involved and its measurement.

5.1 Multiple robots

If multiple robots have to perform simultaneously and in a coordinated way and if delay caused by a network is to be avoided, one controller is put on to manage the motions of the robots, a constellation called 'multi-arm robot' or 'cooperating industrial robots (CIR)'. This cooperation may have the form of a 'robot street', a sequence of robots next, above and/or opposite to each other along a conveyor, where each robot arm performs its task on the objects that pass on the conveyor.

EXAMPLE. A spot-welding task on car frames on a conveyor consists of four sub-tasks. Each sub-task is assigned to one of four robots arms, two at the left and two at the right side of the car frame, to perform the task in a synchronized way. A position sensor notifies the application that a car frame is in position. The application than sends four arrays of welding locations to the Master Robot controller (short: MR controller), each with its corresponding robot ID. A MR controller directly controls the robots, the robots don't need their own controllers). It enables the MR controller to instruct the robots where to weld the car frame. After the welding the MR controller returns the weld status of the four robots to the application. This status is either 'Done' or 'Attention', together with the corresponding robot ID. Only if the 'Done' status has been received from each of the four robot arms the application will start the conveyor and the next car frame task.

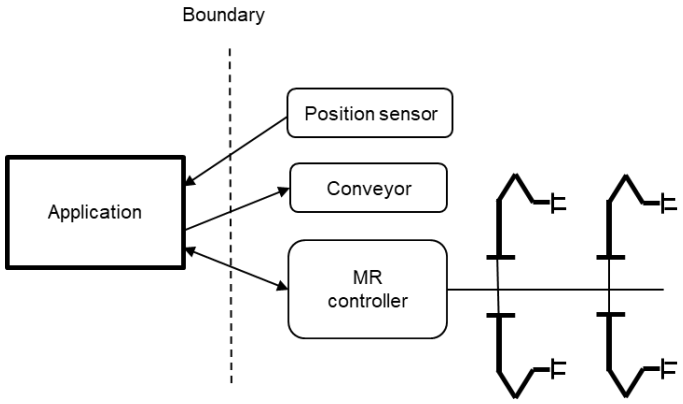


Figure 5.1 – Spot-weld car frame context diagram

Functional process: Spot-weld car

Triggering event:	Car frame in position
Functional users:	Conveyor. position sensor, MR controller
Objects of interest	Conveyor. position sensor, MR controller

DM	Object of interest	Data Group/ Data attributes
Entry	Position sensor	Car frame in position message
Exit	Conveyor	Stop conveyor
Exit	MR controller	Perform spot welding (message with robot ID and array of locations, one to each robot)
Entry	MR controller	Welding status (one from each robot)
Exit*	Conveyor	Start conveyor (if status is 4x ‘Done’, no action else)

Size: 4 CFP

*) According to the Data Movement Uniqueness Rule, identify one Exit to the conveyor for moving the ‘new status’ data group.

REFERENCES

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

Version numbers and publication dates of COSMIC publications given below are correct at the time of publication of this Guideline. However, these publications are updated from time to time as necessary. The 'cosmic-sizing' website will always have the latest version available.

[1] Measurement Manual v4.0.2

[2] ISO 8373 1996 - Manipulating industrial robots -- Vocabulary

[3] Introduction to Robotics, Wiley, S.B. Niku, 2nd edition, 2010

[4] Murphy, R.R., Introduction to AI Robotics, The MIT Press, 2000.

[5] Control Multiple Robots, Morel, M.K., The Fabricator, October 2003.

[6] Principles of Controls, Sensoric and Software Development of Automation and Robots, Mies, G., 2012.

[7] A Work-Piece Based Approach for Programming Cooperating Industrial Robots, Zaidan, S., 2015

[8] Example on youtube: https://www.youtube.com/watch?v=8E6aN4r_YIQ

APPENDIX A - ABBREVIATIONS AND GLOSSARY OF TERMS

This Glossary contains terms that are specific to this Guideline. The Measurement Manual [1] contains the main Glossary of terms of the COSMIC method, In the definitions given below:

- terms are shown in **bold**.
- terms that are defined elsewhere in this Glossary are under-lined, for ease of cross-reference.

Actuator. A device that enables the motion of a robot, such as an electric motor or an hydraulic or pneumatic cylinder.

Application software. An umbrella name for all user applications.

Computer vision. A combination of hardware and software that produces an image of the environment in the form of an array of pixels (picture elements), in order to recognize objects.

Controller software. The permanent software that controls the sensor signals and the motions of the robot, as requested by the user application.

End-effector. A device to grip objects that are to be manipulated or to attach various tools or devices such as for welding or spray painting.

Multi-arm robot. Multiple robots controlled by one controller.

Robot. A reprogrammable and multipurpose machine.

Script. Specification (requirements) of the task of a robot as if it is a human actor performing the script.

Sensor. A device that senses and/or measures an attribute of the environment.

Production task. Connected activities to perform a specification

User application. The software that is entered by the human user to enable the robot to perform a production task of an (industrial) process.

APPENDIX B - ESTIMATING WITH SOFTWARE SIZE

The main uses of functional size is to provide an estimate of the effort of implementing an application on basis of the specification.

For estimating on basis of functional size, both the size and the effort (i.e. the related number of work hours) of a number of applications must be captured. With help of a simple regression analysis ('Ordinary Least Squares') the relationship between sizes and work hours can now be obtained and made visible in Excel. The relationship between sizes and work hours can be expressed by e.g. a linear function (see Figure B.1). This function can be used for

- a tender and/or a Return of Investment estimate,
- to serve as a second opinion to an estimate given by a project manager or a contractor.

With a new application size and the work hours of implementations the company data registration can be updated for estimations of future implementations.

As an example, suppose that the specifications of a number of applications below have been measured and realized, with the indicated implementation effort:

Application	Size (CFP)	Effort (hours)
Application1	50	190
Application2	90	355
Application3	45	165
Application4	60	315
Application5	85	370
Application6	120	365
Application7	30	195
Application8	65	295

These data result in the graph of Figure B.1, with the relationship between size and effort indicated by the trend line. The Figure also displays the formula of the trend line and the 'determination coefficient' R^2. The determination coefficient indicates the preciseness of the model, of which the maximum is 1. The closer to 1, the better the line fits the points.

As the formula is a 'best fit', it makes no sense to use all decimals of the indicated formula, it produces 'order of magnitude' estimates of effort needed to implement an application, given its sizes in CFP. I.e. the formula can be rounded, meaning that efforts can be estimated with help of the simple formula.

Effort (hours) = 2,5 * Size (CFP) + 109

As an example, with a measured size of a specification of 100 CFP, the estimated effort would become 2,5 * 100 + 109 = 359 hours.

Note that it is vital to realize that

- the hours of all applications be collected for the same set of implementation activities. Each estimate is the estimate of the hours to perform *that* set of implementation activities;
- the formula is calibrated for the organization where these numbers were collected, an organization with a different development environment may get different efforts;
- the formula is calibrated for the available size range, i.e. don't use it for sizes far outside this range;
- the formula will change when new data are added.

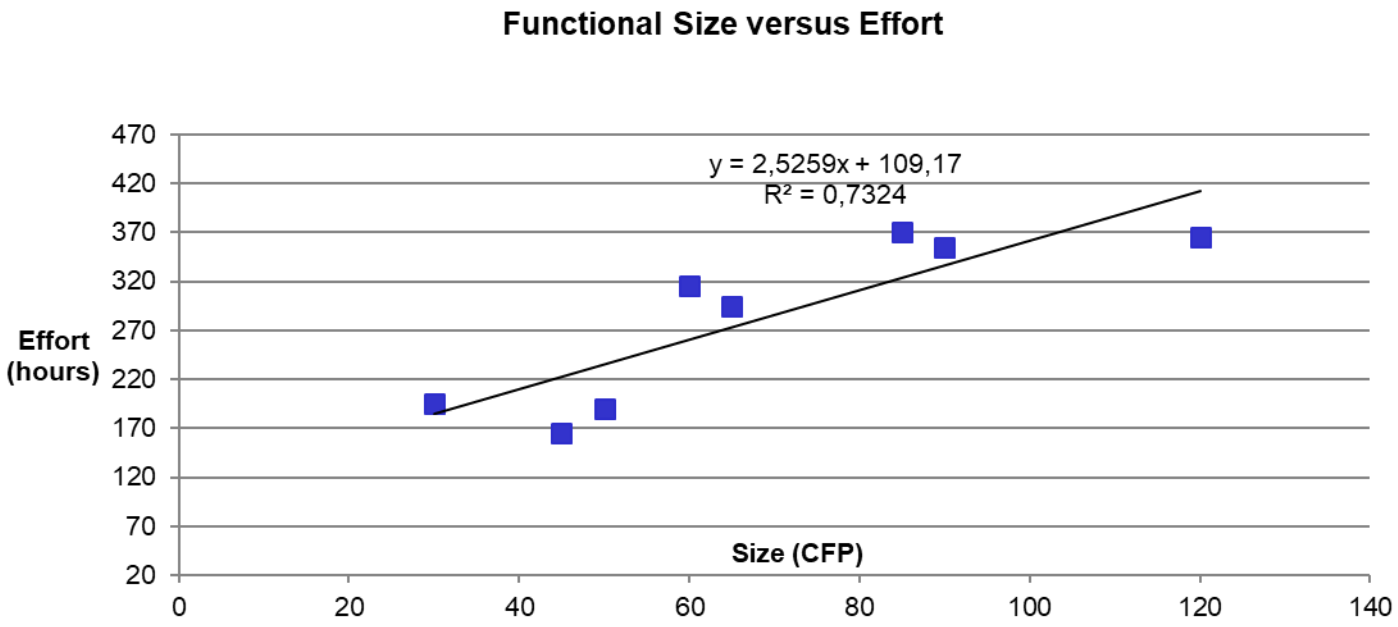


Figure B.1 – The estimation model

APPENDIX C – GENERAL INFORMATION

C.1 Acknowledgements

Version 1.0 authors and reviewers 2018 (alphabetical order)		
Alain Abran*, École de Technologie Supérieure, Université du Québec, Canada	Cigdem Gencel Free University of Bozen-Bolzano Italy	Arlan Lesterhuis*, MPC, The Netherlands
Bruce Reynolds, Tecolote Research, USA	Hassan Soubra, ESTACA, France	Francisco Valdés Souto, Spingere, Mexico

* Authors of this Guideline

C.2 Version control

The following table gives the history of the versions of this document.

DATE	REVIEWER(S)	Modifications / Additions
1-10-2018	COSMIC Measurement Practices Committee	First version 1.0 issued

C.3 Change requests, comments, questions

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

The COSMIC Functional Size Measurement Method
Version 4.0.2

Valve Control System
Case Study

VERSION 1.0.1

August 2018

Table of Contents

1 VALVE CONTROL REQUIREMENTS

1.1 Background of the requirements

The Valve Control system is documented in the ISO technical report: ISO/IEC TR 14143-4 (Version 2000). This ISO document provides various sets of Reference User Requirements (RUR), described usually in a textual formal. The purpose of this ISO document is to provide publicly available sets of requirements to researchers and practitioners to be used as input for measuring functional sizes of software.

The Valve Control system used in this case study corresponds to set RUR B.9 of this ISO document.

1.2 Context

The requirements below describe the behaviour of the solenoid control valve on a hydraulic circuit valve controlling a mechanical device for changing gear on an automatic transmission installed in a land vehicle.

- The valve can be open or closed: it is open by default and closed to engage the gear change mechanism.
- The process controls the amount of time the valve is closed during an operating cycle of several thousand microseconds.
- A clock supplying the operating cycle reference triggers the process.

1.3 Input

The valve control process uses as inputs:

1. A sensor signal (Gc) indicating gear change is in progress (value 1) or not (value 0),
2. A sensor signal (Su) indicating, during gear change, if shifting to upper gear (value 1) or lower gear (value 0),
3. A sensor signal (Idl) indicating whether the transmission is under stress (value 0) or idling (value 1),
4. A binary flag “A” whose value is stored in the processor ROM memory or
5. A binary flag “B” whose value is stored in the processor ROM memory,

Binary flags “A” and “B” describe some general configuration characteristics of the automatic transmission.

1.4 Output

The valve control process produces as output:

- Time (T), during one operating cycle, during which the control valve must be closed.

1.5 Requirements

1.5.1 Part A – Determine the general operating condition

Determine whether operating slowly or quickly from the closed state of the hydraulic valve.

IF (Gc = 1

AND Idl = 1
AND A = 0
AND B = 0)

THEN, operating under normal condition, perform PART B

IF (Gc = 1
AND Idl = 0
AND Su = 1
AND A = 0
AND B = 0)

THEN, operating during gear change, perform PART C

1.5.2 Part B – Control to open hydraulic valve slowly from its closed state

Reset T to the smaller value of either INIT or the value of T during the last process cycle, where INIT is a constant stored in the computer ROM memory,

Compute the new value of T: **T = T - (Cst_X * ET)**

where Cst_X is a constant stored in the processor ROM memory and ET is the elapsed time since an action that opens the hydraulic valve slowly from its closed state has been activated.

Condition for completion:

if the following conditions are met then valve control is passed to another process, not part of the measurement scope:

T is smaller or equal to LT OR Slp is greater or equal to Uslp

where LT is a lower threshold of time and Uslp is an upper threshold of the amount of slip stored in the processor ROM memory. Slp is the current amount of slip, which denotes the difference of number of revolutions between the engine output shaft and the power train shaft. The value is computed and updated according to the following formula and stored in the processor RAM memory.

$Slp = |E_{rev} - PS_{rev}|$

where E_{rev} is the engine's output shaft revolutions and PS_{rev} is the power train shaft revolutions. Both variables' values are supplied by concurrent processes using input from separate sensors and placing the calculated result in the processor RAM memory.

1.5.3 Part C – Control to open the hydraulic valve quickly from its closed state

- Reset T to the smaller value of either INITS(V_s) or the value of T during the last processing cycle, where INITS is a table of initial values stored in the processor ROM memory and V_s is the vehicle speed which is computed and updated by another process and stored in the computer RAM memory.
- Compute the new value of T: T = T - (INCR(V_s) * ET) where INCR is a table of increments which depend on the speed of the vehicle stored in the processor ROM memory and ET is the elapsed time since an action to close the hydraulic valve quickly from its closed state has been activated,
- Condition for completion: if the following conditions are met then valve control is passed to another process:

  T is smaller or equal to LT. Where LT is a lower time threshold stored in the processor ROM memory.

2 MEASUREMENT STRATEGY

2.1 Measurement purpose

The measurement purpose is to measure all of the Functional User Requirements (FUR) of the software requirements documented in the set of Reference User Requirements selected for this case study using the COSMIC functional sizing method. The Measurer must derive the FUR from the RUR.

2.2 Measurement scope

The measurement scope is all of the software Functional User Requirements derived from the set RUR B.9, and only these. The measurement scope is therefore based on a subset of the system requirements documented in this ISO case study, that is, only those related to software, and not those related to the hardware.

There is a single software layer for this set of requirements.

2.3 Identification of functional users

The functional users that interact with this software are the following mechanical devices:

a) Send information to the software:

- Clock
- Sensors: GC, Su and IDL

b) Receives information from the software:

- The control valve

From the requirements, as written, there are no human users, nor are there other software applications interacting with this software.

Based on the requirements, we have the following context diagram:

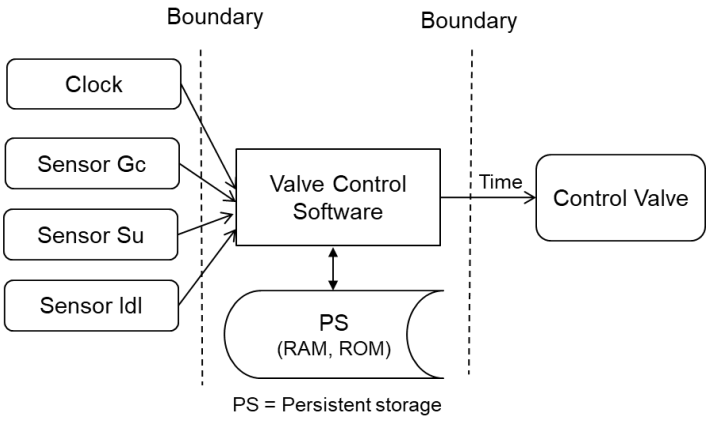


Figure 1 – Context diagram of the Valve Control software from its FUR

2.4 Other measurement strategy parameters

Level of granularity. The software requirements of the Valve Control are at the 'functional process level of granularity' as the functional users are individual hardware devices (not groups of these) and a single event occurs that the software must respond to (not groups of events).

Level of Decomposition: No decomposition.

Persistent storage. The COSMIC model does not recognize specific types of hardware storage as persistent storage, the ROM and RAM correspond to the persistent storage.

3 THE MAPPING AND MEASUREMENT PHASES

3.1 Identification of the software triggering event

From the documented requirements a single triggering event is identified:

- A Clock supplying the operating cycle reference which triggers the process.

3.2 Identification of functional processes

From the documented requirements with a single triggering event, there is one functional process:

- The Control of time during the operating cycle of the control valve.

3.3 Identification of objects of interest

As noted in the Measurement Manual [1], section 3.3.4, 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.

Clock. Operating cycle reference
Sensor Gc. Current value of the Gc sensor
Sensor Su. Current value of the Su sensor
Sensor Idl. Current value of the Idl sensor
Control Valve. New value of T

3.4 The functional process and its data movements

This section describes the functional process with its 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: **The Control of time during the operating cycle of the control valve** **Triggering Event:** **The operating cycle reference**					
Functional user	**Data movement description**	**Name of Data Group moved**	**Object of interest of Data Group moved**	**Data Mvt. Type**	**CFP**
Clock	Supply operating cycle reference	Operating cycle reference	Clock	E	1
Sensor Gc	Receive signal of sensor Gc	Gc value	Sensor Gc	E	1
Sensor Su	Receive signal of sensor Su	Su value	Sensor Su	E	1
Sensor Idl	Receive signal of sensor Idl	Idl value	Sensor Idl	E	1
-	Read Valve fixed	Valve fixed	Valve fixed	R	1

	parameters from ROM	parameters*)	parameters		
-	Read of T from RAM	T	Period valve-to-be-closed	R	1
-	Read of ET from RAM	ET	Period since last action	R	1
-	Read of E_{rev} from RAM	E_{rev}	Engine	R	1
-	Read of PS_{rev} from RAM	PS_{rev}	Power train	R	1
-	Read V_s from RAM	V_s	Vehicle	R	1
-	Send T to the Control Valve	T	Control Valve	X	1
-	Write T to RAM	T	Period valve-to-be-closed	W	1
					Total size: 12 CFP

*) The 'Valve fixed parameters' consist of Flag A, Flag B, INIT, Cst_X, LT, Uslp, INITS 1, 2, 3 etc and INCR 1, 2, 3, etc, all other data groups consist of one data attribute.

Note 1: The three parameters E_{rev}, PS_{rev} and V_s are, explicitly according to the requirements, obtained from RAM and provided by other processes. The requirements do not state if these other processes are hardware or software but that does not concern us. These three parameters have to be Read from RAM by the Valve Control process.

Note 2: The requirements, as documented, do not specify whether the Elapsed times (which are defined differently for Parts B and C) are given by the hardware, or whether they are calculated by the software, nor do the requirements state where the ET is obtained from. For the purposes of this case study (that is to measure what is explicitly allocated to software and FUR) this is then not FUR to be measured: the ET is provided by another process and the Valve Control process obtains it from the RAM. Should the documentation of the requirements be modified to explicitly assign this to a software function, another functional measurement would have to consider this added function to be developed and measured.

Note 3: Slp is calculated on each cycle and is not made persistent between cycles. It is therefore the result of data manipulation and is not involved in any data movement according to the COSMIC method.

4 QUESTIONS AND ANSWERS

Question 1

The following is written in the specifications: 'Reset T to the smaller value of either INIT or the value of T during the last process cycle'. Why must a Read data movement of T be identified?

Answer to Question 1

Because T is always updated according to the hydraulic valve operating state.

Question 2

It is written in the documented requirements that the Flags A and B are 'Inputs' to the software. When the software accesses the information in Flags A and B, why in COSMIC are these data movements considered as Reads and not considered as Entries?

Answer to Question 2

The vocabulary in the documented requirements is not standardized. Even though Flags A and B are mentioned as 'inputs', it is also written that they are coming from a ROM memory. In COSMIC, if there is no question of the ROM being accessible via another piece of software, information in a ROM is considered as persistent data within the software boundary. Any access to these data is then identified as a Read.

Question 3

What is the impact on the measurement result if the system requirements are changed and the elapsed time (ET) is to be implemented as a hardware function?

Answer to Question 3

If ET comes from hardware, then a 'functional user' (For ET as a hardware source) should be shown on the diagram and ET should NOT be shown in the RAM.

Then: ET is an Entry data movement from hardware, instead of one Read data movement from RAM. The overall size is unchanged.

Question 4

What is the impact on the measurement results if the software functional requirements are changed as follows:

Old requirement: ET is provided by hardware and stored in RAM

New requirement: this Valve Control process calculates and keeps track from one cycle to another, that is, Valve Control process stores a time-stamp to RAM between cycles and uses it to calculate ET.

Answer to Question 4

To do this the Valve Control process needs one Read and needs to Write a time-stamp to RAM between cycles. The overall size would increase by one CFP from the solution given in 3.4 above.

5 QUALITY CRITERIA FOR REQUIREMENTS

5.1 Observations on the clarity of the documented requirements

Even though the documented requirements used for this case study are coming from an ISO technical report, there is no mention in this report about the quality of these requirements.

IEEE Std 830-1998 recommends that requirements meet the following quality criteria:

- Correct;
- Unambiguous;
- Complete;
- Consistent;
- Ranked for importance and/or stability;
- Verifiable;
- Modifiable;
- Traceable.

In the ISO technical report, there is no claim that their sets of documented requirements meet the quality criteria specified in IEEE 830. The following ambiguities have been noted:

1 From the documented requirements, it is not clear neither where the control of the valve is performed nor where to send the time T after the calculation. For this measurement, the following clarification to the written requirements as been specified as follows:
   - *T is sent to the Control Valve*

2 It is not documented when the application interacts with the ROM and the RAM. The specifications do not make clear the sequence in which any of the data movements occur. This observation does not modify the result of measurement of the functional size.

3 In the parts B & C of the requirements, there is no indication for actions to take if the conditions for completion are not met.

4 It is not documented where the variable ET (Elapsed time) comes from (that is, the elapsed time since an action that opened the hydraulic valve slowly or closed it quickly). For the measurement of this case study, this requirement have been specified as follows in the documentation used for measurement:
   - *ET is read from the RAM.*

5 The parameters E_{rev}, PS_{rev} and V_s are all stated to be supplied into RAM by 'other concurrent functional processes'. Since these are seemingly asynchronous with the Valve Control functional process being measured it seems correct to assume that they are all attributes of separate Objects of interest, thus requiring separate Reads.

6 Note that the parameter T must be made persistent in RAM between each cycle of the Valve Control functional process. Therefore, each cycle of this process must Read the value from the last cycle and Write the latest value for use by the next cycle.

7 In section 1.5.1 it reads ‘THEN, operating under normal condition, perform PART B’. It is not explained what ‘normal condition’ means.

5.2 Conclusions and observations

This case study has provided an illustration of the functional size measurement with ISO 19761: 2017 COSMIC. The measurement is based on the requirements of RUR B.9 of ISO TR 14143-4, as documented in a textual format.

During the measurement process, uncertainties and ambiguities about the documented requirements have been noted. It was also observed that these requirements do not meet all of the quality criteria listed in IEEE 830.

The measurement solution presented takes into account some clarifications that have been required to document the allocation for instance of the Elapsed time calculation function to hardware. If the modifications to the documentation of the requirements are changed by something else, then the measured functional size could change.

Appendix A

A MESSAGE SEQUENCE DIAGRAM

For the single functional process, all data movements of a data group must be identified. In this case study, the Message Sequence Diagram (Figure 2) has been prepared to facilitate the identification of the data movements, and to ensure that all data movements have been identified.

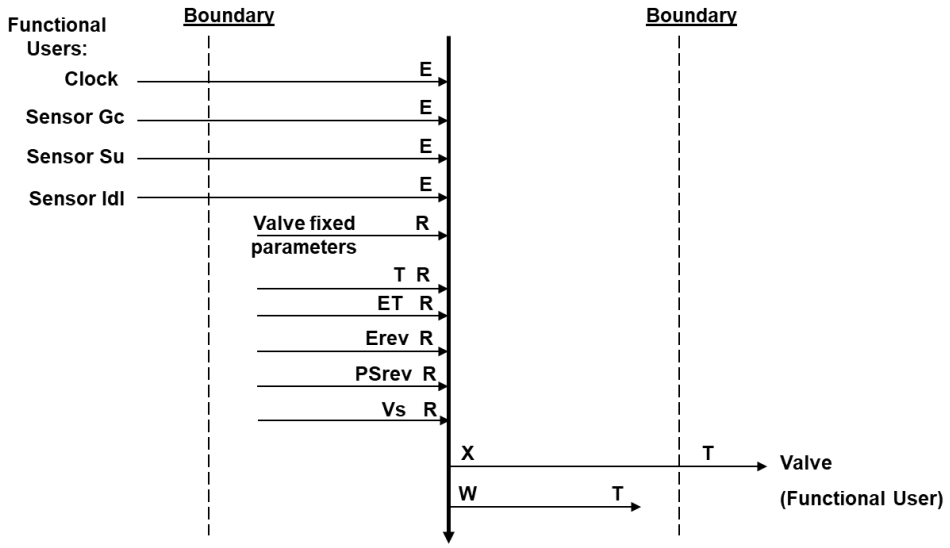


Figure 2 - Valve Control Application – Message Sequence Diagram

Appendix B

VERSION CONTROL

The following is a partial account of the evolution of this case study. For details see the previous version of this case.

Date	Reviewers (s)	Modifications / Additions
Jan 2006	Adel Khelifi, Alain Abran, Chantal Roy, Charles Symons, Marie O'Neill	COSMIC-FFP version 1.0 published.
August 2018	Arlan Lesterhuis, Bruce Reynolds, Charles Symons	Version 1.0.1 updated to align with Measurement Manual version 4.0.2: document style adjusted, context diagram redrawn, 'Cfsu' replaced by 'CFP'. No changes in content.

CHANGE REQUESTS, COMMENTS, QUESTIONS

Where the reader believes there is a defect in the text, a need for clarification, or that some text needs enhancing, please send an email to: mpc-chair@cosmic-sizing.org

You can use the forum on cosmic-sizing.org/forums to post your questions and receive answers from our world-wide community. The quality of any answers will depend on the knowledge and experience of the community member that writes the answer; the MPC cannot guarantee the correctness. Commercial organizations exist that can provide training and consultancy or tool support for the method. Please consult the www.cosmic-sizing.org web-site for further detail.

nesma

Web Advice Module

COSMIC Case Study

www.nesma.org

A publication of Nesma

Table of Content

Revision history

VERSION	DATE	DESCRIPTION	AUTHORS
1.0	January 2014	Final version	Frank Vogelezang Charles Symons Arlan Lesterhuis
1.1	February 2016	Updated to version 4.0.1 of the COSMIC method and updated versions of used Guidelines	Frank Vogelezang Charles Symons Arlan Lesterhuis
1.1.1	November 2018	Updated to version 4.0.2 of the COSMIC method and versions of updated Guidelines used.	Frank Vogelezang Charles Symons Arlan Lesterhuis

Reviewers of version 1.0

NAME	ORGANIZATION	COUNTRY
Alain Abran	École de technologie supérieure	Canada
Chris Woodward	DCG-SMS	United Kingdom
Cigdem Gencel	Deiser	Spain
Edwin van Gorp	Sogeti Nederland	the Netherlands
Eric van der Vliet*	CGI	the Netherlands
Fred de Wilde*	ING Domestic Banking	the Netherlands
Harold van Heeringen*	Metri group	the Netherlands
Jean-Marc Desharnais	École de technologie supérieure	Canada
Jolijn Onvlee*	Onvlee Opleidingen & Advies	the Netherlands
Luca Santillo	Agile Metrics	Italy
Peter Bellen*	Independent consultant	the Netherlands
Peter Fagg	Pentad-SE Ltd.	United Kingdom
Theo Prins	Sogeti Nederland	the Netherlands

** member of the COSMIC working group of Nesma*

Reviewers of version 1.1

NAME	ORGANIZATION	COUNTRY
Arlan Lesterhuis	COSMIC	the Netherlands
Charles Symons	COSMIC	United Kingdom

Reviewers of version 1.1.1

NAME	ORGANIZATION	COUNTRY
Arlan Lesterhuis	MPC	the Netherlands
Bruce Reynolds	Tecolote Research	USA
Francisco Valdés Souto	SPINGERE	México
Frank Vogelezang	METRI	the Netherlands

1 Foreword

This case study is an example of how the COSMIC method should be applied to measure a small web application. The case study has been assembled by members of the COSMIC working group of Nesma, based on real specifications, for the purpose of sharing our experience with the COSMIC community.

This application, although very small, contains a lot of discussion points about how the COSMIC method should be applied correctly. It is a real application, which could be encountered by any measurement professional. Because of its compact, yet complicated nature we decided to make a case study out of this application. We hope that it helps the COSMIC community to understand a number of aspects of the method better and assists in ensuring consistent interpretation of the COSMIC principles.

1.1 Authors

This document has been initiated by the COSMIC Working Group of Nesma, who served as the main reviewers, together with other volunteers from the COSMIC community.

1.2 Conformity to the COSMIC measurement principles

The conformity to the current version of the Measurement Manual, version 4.0.2 (MM), the Business Application Guideline, version 1.3 (BAG), the Guideline for Sizing Service Oriented Architecture Software, version 1.1 (GSOA) and the Guideline for Sizing Real-Time Software, version 1.1.1 (RTAG) has been verified by experienced COSMIC practitioners. Wherever one of these documents is referenced, the reference is made to the section title rather than to the section number.

1.3 Observations on the clarity of the documented requirements

Although based on real requirements, the original requirements have been edited for the purposes of this case study to make the requirements as unambiguous as possible to be able to focus on the measurement principles. From the length of the discussion section in chapter 4 one might get the impression that the COSMIC method is difficult to apply. However, the reader should bear in mind that a large portion of the discussion in chapter 4 deals with the interpretation of the requirements, rather than with the application of the COSMIC method. This is a deliberate choice of the authors, since in practice the interpretation of the requirements is an important part of the measurement process. The COSMIC method has included the Mapping Phase to extract the Functional User Requirements (FUR) in a form to which the COSMIC Generic Software Model can be applied (MM Applying the Generic Software Model).

1.4 Disclaimer

By publication of this user guide the Nesma wants to contribute to further understanding of applying the COSMIC method. Nesma is not responsible for the use of this publication, nor for the results obtained by using the described approach.

1.5 Contact information

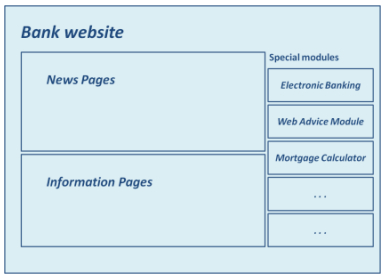
Any suggestions, remarks or questions regarding this case study can be sent to office@nesma.org.

1.6 More information on the COSMIC method

More information on the COSMIC method, including the latest versions of the mentioned manuals, and other case studies can be obtained freely from the COSMIC website at www.cosmic-sizing.org.

2 Requirements for the Web Advice Module

2.1 Context



The Web Advice Module is a special module on the website of a commercial bank to assist (young) customers with the choice whether they are going to rent a house or buy one with a mortgage. The customer fills in his or her opinion to a number of propositions and is presented rough advice from the Web Advice Module. Based on this advice the customer can request a rent or buy advice session via the Module if he or she wishes, or contact the local branch for advice.

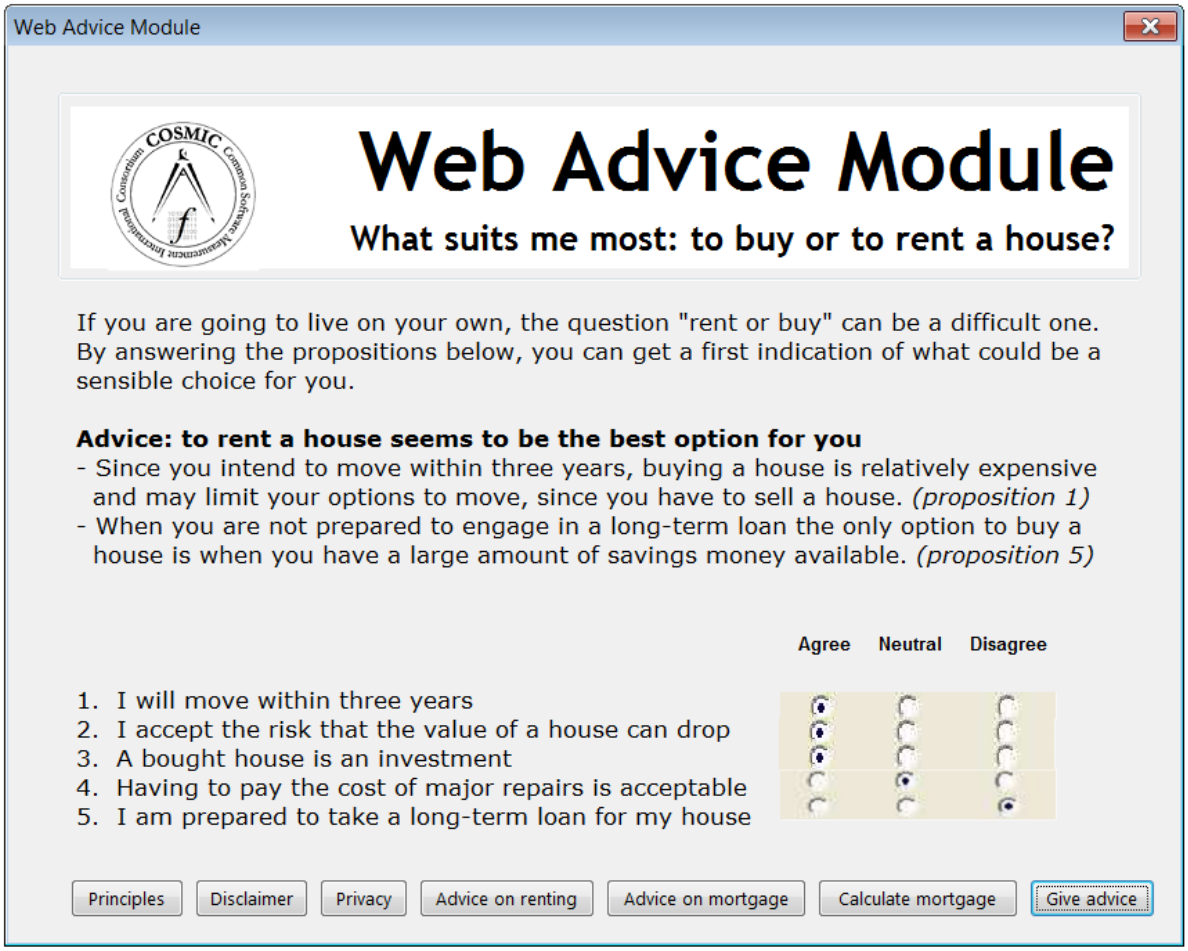
2.2 Start page

On the start page the customer is presented with a short introduction text and five propositions with which he or she can agree, disagree or choose neutral. By default the neutral choice is activated. When the customer is satisfied with the choices, he or she can push the Give advice button to see the advice, based on the set of choices.

The other buttons are all active and start functionality that is described in later sections.

2.3 Advice page

Based on the business rules, the Start page is re-displayed with a general advice on the best option and the relevant advice text for each choice that has an answer in the category of the general advice. The advice texts shown below are example texts that need to be finalized on implementation. The lower part of the page shows the propositions again with the current choices. These can be changed to generate a new advice.



2.4 Business rules

Each agree or disagree corresponds to a score for renting (R) or buying (B) a house and for each choice an advice text is available to accompany the general advice:

1. agree (R, rent text 1) disagree (B, buy text 1)
2. agree (B, buy text 2) disagree (R, rent text 2)
3. agree (B, buy text 3) disagree (R, rent text 3)
4. agree (B, buy text 4) disagree (R, rent text 4)
5. agree (B, buy text 5) disagree (R, rent text 5)

If the customer chooses neutral for a certain proposition, no advice text is shown in the advice for that proposition.
If the number of R-scores is greater than the number of B-scores, the customer is shown the general advice to rent a house together with the advice texts for the propositions that resulted in an R-score.

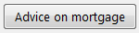
If the number of B-scores is greater than the number of R-scores, the customer is shown the general advice to buy a house together with the advice texts for the propositions that resulted in a B-score.

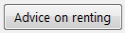
If the numbers are equal, then a mixed general advice is shown together with the advice texts of the questions that resulted in an R-score and the advice texts for the propositions that resulted in a B-score.

Three choices can change this logic:
- If the choice at proposition 2 is **disagree**, then the general advice is always to rent.
- If the choice at proposition 5 is **disagree**, then the general advice is always to rent.
- If the choice at proposition 1 is **agree**, then the general advice is to rent if the number of R-scores is equal to or higher than the number of B-scores. If the number of B-scores is higher than the number of R-scores, then the mixed general advice is given.

2.5 Request form for an Advice Session

The customer may require advice on renting or on a mortgage to buy a house. To help the customer, he or she may send a request for an advice session from the Web Advice Module. The customer enters personal and contact data in a request form.

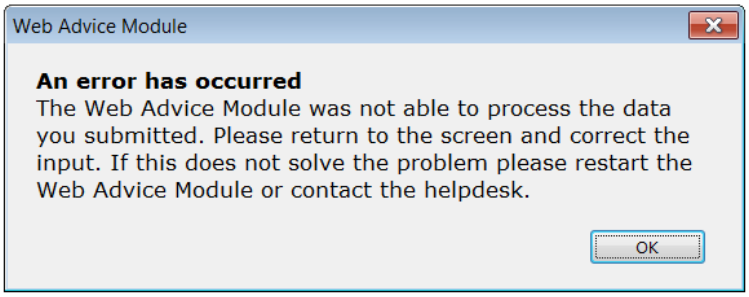
When the customer presses the 

or 

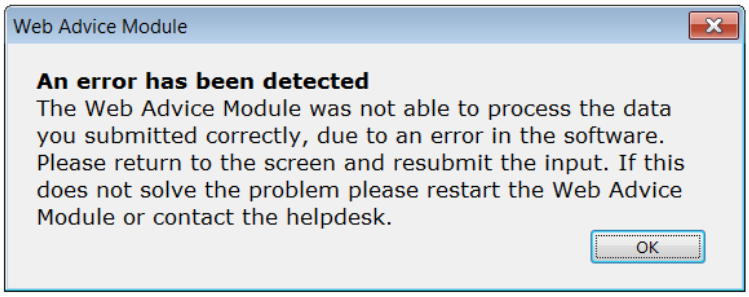
button the entered data, together with the answers from the start page and the general advice and advice texts from the advice page, are sent to a back office service of the bank to arrange the session. See also section 4.1.2.

2.6 Error messages

When the Web Advice Module detects erroneous processing, this must be communicated to the user as an error message in a separate window. Erroneous processing can be the result of either some (technical) error from a software component or an error of a human user. If a human user causes an error the following error message must be shown:



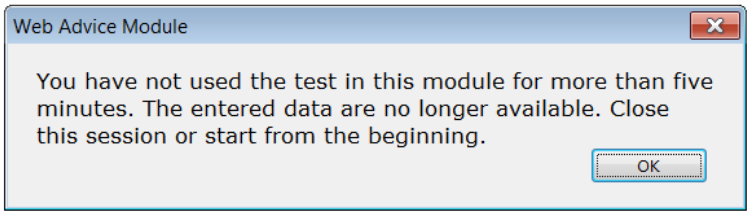
If a software component causes a (technical) error, the following error message must be shown:



To this message additional (technical) information may be added to assist users of the Web Advice Module to solve the cause of the erroneous processing.

2.7 Inactivity message

If the customer has not used any functionality of the pages of the Web Advice Module for more than five minutes, a pop-up message will appear with a message that the customer has been inactive for too long and that the entered data and answers are not available anymore:



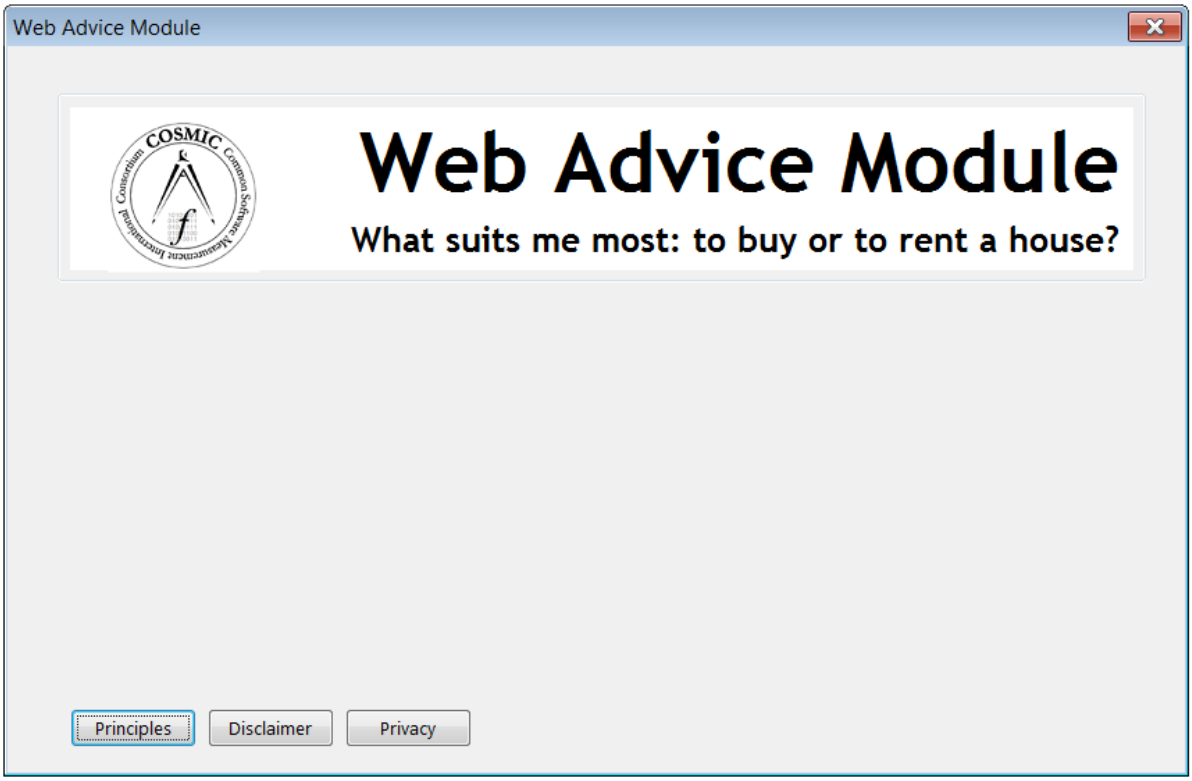
2.8 Mortgage assessment

If the customer presses the Calculate mortgage button the customer is directed away from the Web Advice Module to an existing mortgage assessment module. This existing module gives an indication of the maximum mortgage the bank is willing to provide, based on some financial data the customer has to provide in the mortgage assessment module. No data is sent to the mortgage assessment module from the Web Advice Module.

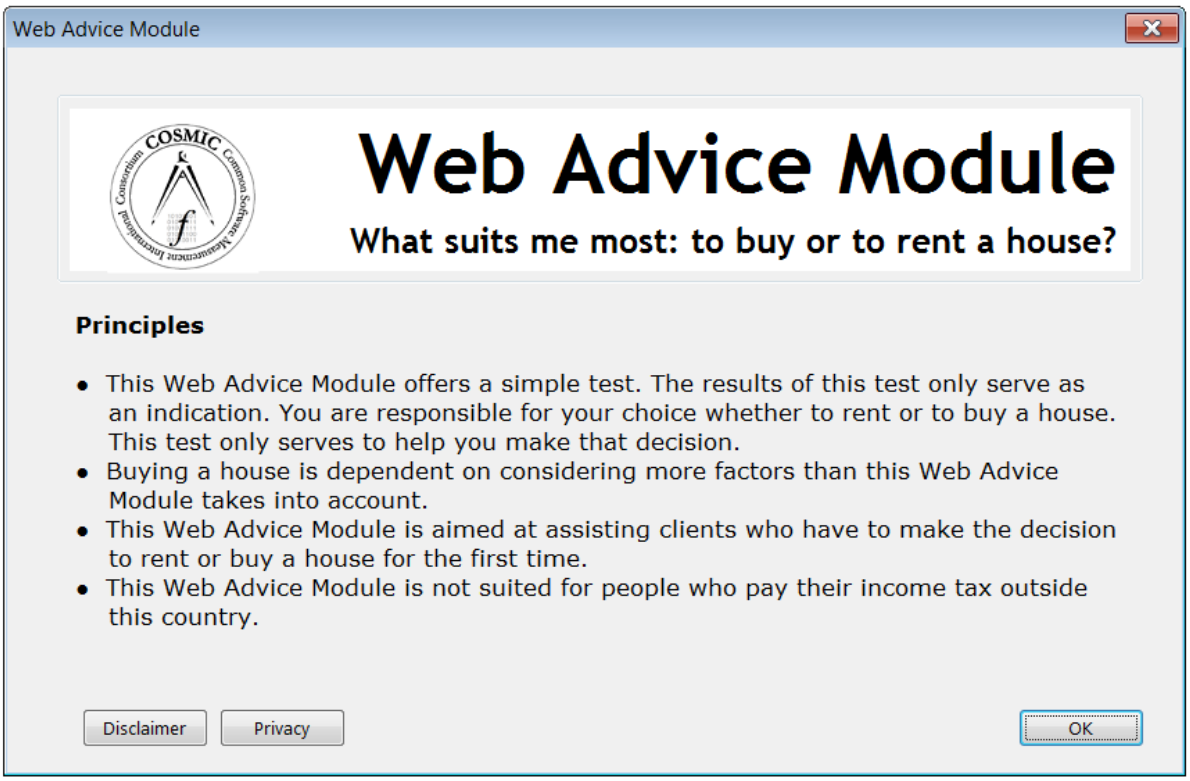
2.9 Requirements for all pages

Below each page are links to the Web Advice Module's principles, the bank's privacy statement and the bank's disclaimer for web pages. Both the privacy statement and the disclaimer are existing services that can be invoked from the Web Advice Module and are shown directly in the browser of the customer, without further interaction with the Web Advice Module. These existing services are available to all web applications from the bank. All entered data are subjected to basic validations for formatting and valid range. The customer is notified when invalid data is entered by an 'error has occurred' message. All entered data remain available within the Web Advice Module during the web session. When a customer requests an advice session these entered data are sent to a back office service. When the web session is closed by clicking on a 'close window' button, all entered data is no longer available.

The title of this module is "Web Advice Module" which appears in the title bar of the internet browser. In the top of the page the bank's logo is shown, together with the title "Web Advice Module". Below the title in a blue bar the following proposition is shown: "What suits me most: to buy or to rent a house?" The figure below this section is an example of what the general page lay-out should look like:



At the bottom of each page there is a "Principles" button. When this button is pressed, the Web Advice Module opens a new page with the following information:



2.10 Maintenance of the Web Advice Module

The following items must be editable without the assistance of a programmer:

- The introduction text on the Start page (see section 2.2)
- The proposition texts (see section 2.2)
- The general advice texts (see section 2.3)
- The advice texts (see section 2.3)

3 Measurement strategy phase

In this chapter sections in *italics* are explanatory text from the authors that are not a part of the actual measurement process.

3.1 Determine the PURPOSE

The purpose of this measurement is to measure a functional size of the Web Advice Module that can be used as a basis to estimate the required effort to build the software.

3.2 Determine the SCOPE

The scope of the measurement is all of the FUR that are related to the Web Advice Module, i.e. which are derived from the requirements in chapter 2 of this document.

This implies that the functionality of existing services is out of scope for this measurement. However, the functionality within the Web Advice Module to start the existing services is within the measurement scope.

3.3 Determine the LEVEL OF DECOMPOSITION

The level of decomposition of this scope is that of a whole application. All the functionality described in the FUR that is in scope for this measurement resides in the application layer.

3.3.1 One or more layers

The privacy statement and the disclaimer are utility services that provide common functionality (business or non-business) independently of, but available to, other services (GSOA Utility Services). The Web Advice Module and all other services that it invokes are all in the one application layer.

3.4 Determine the LEVEL OF GRANULARITY

The level of granularity of the requirements is at the standard level of granularity, the functional process level.

3.4.1 Determining the level of granularity

The requirements are at the standard level of granularity, meaning that the functional users are individual humans (Customer, Application manager) or individual pieces of software (Back Office service, System Clock (as part of the Operating System), Privacy statement service, Disclaimer service) and not groups of these. The functional users that provide input data detect single occurrences of events that the Web Advice Module must respond to (MM The standard level of granularity).

By measuring at the standard level of granularity it is possible to use this measurement not only for the purpose of this measurement (see §3.1) but also for benchmarking purposes, since most benchmark data is available at the standard level of granularity.

3.5 Identify the FUNCTIONAL USERS

3.5.1 Identifying functional users

The functional users are the senders and/or intended recipients of data in the FUR of the Web Advice Module.
In this case study the functional users are:

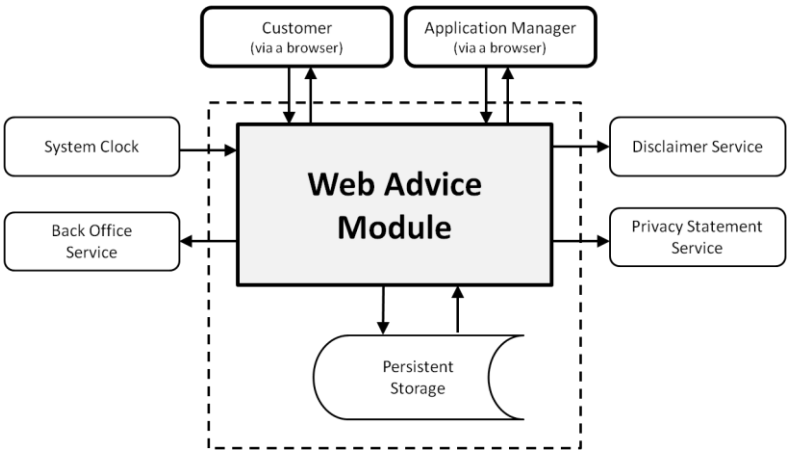
- The human customers that use the Web Advice Module to get advice.
- The application manager that maintains the editable texts within the application.
- The system clock that controls the inactivity control process.

Also all identified pieces of software in this case study must be considered functional users of the Web Advice Module (GSOA The functional users of services):

- Privacy statement service
- Disclaimer service
- Back Office service

There is no exchange of data between the Web Advice Module and the Mortgage calculation service. By pressing the Calculate mortgage button the customer is directed to the Mortgage calculation service. From that point the customer can start the functionality of that service. The direction from the Web Advice Module to the Mortgage calculation is mere navigation that does not start a functional process. Consequently, this should be ignored in the measurement of Web Advice Module (BAG Menus and the triggering Entry).

The boundary of the Web Advice Module is a conceptual interface between this piece of software and its functional users. The boundary allows the measurer to distinguish, without ambiguity, what is included inside the measured software from what is part of the measured software's operating environment. (MM Functional Users). It is indicated by the dashed line:



The arrows represent the exchange of data between functional users and the Web Advice Module. The arrows to the Disclaimer and the Privacy statement are only outward bound, since the data these services present are directly presented to the human user (by means of the web browser) and not to the Web Advice Module and no feed-back is required.

3.5.2 Persistent storage

Persistent storage is storage which enables a functional process to store a data group beyond the life of the functional process and/or from which a functional process can retrieve a data group stored by another functional process, or stored by an earlier occurrence of the same functional process, or stored by some other process. In the COSMIC model, persistent storage has a specific meaning, namely that it exists only within the boundary of the software being measured, i.e. that it isn't a functional user of the software being measured. In other words, it is storage where the FUR are not concerned with how those data accesses are handled by any other software.

From the requirements with respect to retrieving and storing data it appears that no storage requires intervention of other software, therefore all storage mentioned is persistent storage. It implies that for all retrieve and store activities, Read and Write data movements must be identified, rather than Exit / Entry pairs.

4 Mapping phase

The Mapping Phase is intended to express the Functional User Requirements in a form to which the COSMIC Generic Software Model can be applied (MM Applying the Generic Software Model). Applying the COSMIC Generic Software Model means identifying the set of triggering events sensed by each of the functional user (types) identified in the FUR, and then identifying the corresponding functional processes, objects of interest, data groups, and data movements that must be provided to respond to those events.

In this chapter sections in *italics* are explanatory text from the authors that are not a part of the actual measurement process.

4.1 Identify FUNCTIONAL PROCESSES

The first step in the mapping phase is to determine the unique event types and the corresponding functional processes. The most important general advice is that it is almost always useful to try to identify first the separate events that the software must respond to ('triggering events'), since each such event gives rise to usually one (but sometimes more than one) functional process.
A functional process is a set of data movements, representing an elementary part of the Functional User Requirements for the software being measured, that is unique within these FUR and that can be defined independently of any other functional process in these FUR. It starts processing on receipt of a data group moved by the triggering Entry data movement of the functional process. Its set of data movements is needed to meet the FUR for all the possible responses to its triggering Entry. (MM Identifying functional processes).

A triggering event is an event that causes one or more functional users of this software to generate one or more data groups, each of which will subsequently be moved by a triggering Entry (MM Identifying functional processes)

In this case study the following functional processes are identified (recipients of the result of the functional process are shown in grey):

Triggering event	Functional user	The data moved by the Triggering Entry	Functional process
Request for rent or buy advice	Customer Customer	Proposition choices	Display rent or buy advice
Request for an advice session (on renting)	Customer Back Office Service	Customer data	Advice session request form
Request for an advice session (on mortgage)	Customer Back Office Service	Customer data	Advice session request form
Need to select a text to be changed	Application Manager Application Manager	Request to display all editable texts	Select text
Need to edit selected application text	Application Manager Application Manager	Edited text for selected application text item	Edit text

Triggering event	Functional user	The data moved by the Triggering Entry	Functional process
Interest in principles	Customer Customer	Principles button pressed	Display principles
Interest in disclaimer	Customer Disclaimer Service	Disclaimer button pressed	Invoke disclaimer service
Interest in privacy statement	Customer Privacy statement Service	Privacy statement button pressed	Invoke privacy statement service
Time interval elapsed	System Clock	Threshold exceeded message	Display time-out message
User decides to end Module session	Customer Customer	Close Module session message	Close Module session

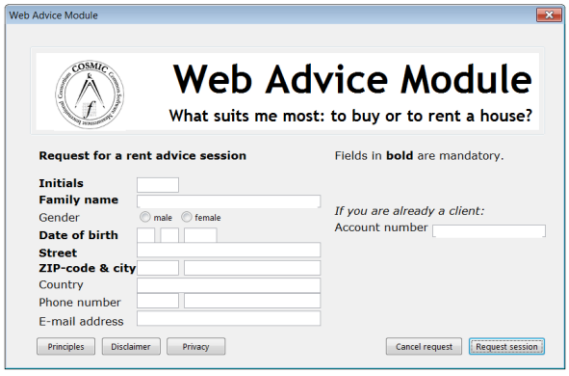
4.1.1 One or more functional processes to obtain an advice?

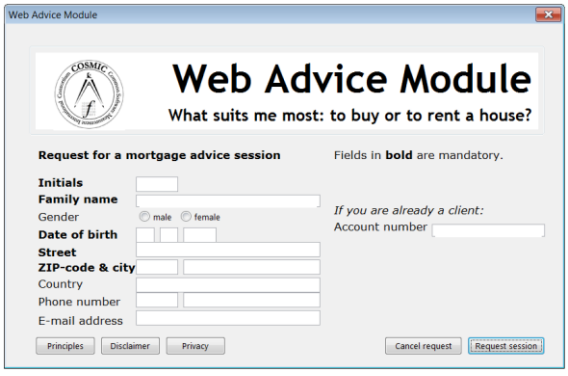
When an advice has been obtained based on the first time the advice module has been used, the advice module offers the possibility to modify the choices given the first time and to obtain a modified advice. Is this a second functional process or not?

When used for the first time this page presents default choices that come from the application itself. In any round of use of the advice module the user can change the choices to one or more questions to generate an advice. The only difference between the first use of the module and any following use is that no advice is shown when the module is used the first time. With any following use the previous advice, corresponding to the previously entered choices, is still displayed. As these uses are repetitions of the same functionality, the uses are occurrences of one functional process (type). In consequence there is only one triggering event type that represents its many triggering event occurrences. (See MM Types versus Occurences)

4.1.2 One or two advice session request form functional processes?

There is a requirement for a request form for an advice session on renting or mortgage, which leads to one functional process. Although the request has been implemented in two (slightly different) ways, the requirement is still for one functional process, i.e. we count the requirements, not their implementation.
In the delivered software the FUR have been implemented in the following two web pages:





4.1.3 Maintenance functional processes

In §2.10 the requirements state that a number of texts used by the Web Advice Module must be maintainable without the assistance of a programmer. This requirement is not very precise. We therefore assume the following:

- *The maintenance functional processes have a human functional user that is different from a regular bank customer user, who we call the Application Manager.*
- *Although the functionality of the Web Advice Module differentiates the application texts into 1 introductory text, 3 general advice texts, 5 proposition texts and 10 advice texts this differentiation is irrelevant to the Application Manager functional user. Therefore no distinctions are made between the maintenance of different types of application texts. Whether or not something is an object of interest can be different for different functional users and must be determined anew for each functional process.*
- *The requirements do not mention the need to create or delete new application texts. This implies that the number of application texts is fixed. This means that all application texts exist at all times, so there is no need to identify functionality to create or delete application texts, only to select and display the right text and to edit that text.*
- *It is assumed that the Application Manager maintains the data via a web browser and that maintenance via a web browser or via application software gives no difference in functional size of the Web Advice Module.*

Elaborating from these assumptions the requirements from section §2.10 lead to two (maintenance) functional processes: Show text (see §0) and Edit text (see §5.4).

4.1.4 Display principles functional process

The question is whether this functional process violates rule c) of a functional process (MM The approach to identifying functional processes) that requires a functional process to comprise at least two data movements. As the requirements do not mention any need to maintain the Principles statement, we assume that the information text of the principles is hard-coded as part of the web page. This only means that no Read is necessary to access it in a functional process. But when the Principles are displayed it is a data group that is moved across a boundary. Moving a data group across a boundary means processing data about an object of interest.

According to rule a) of the Exit (MM Identifying Exits) output of fixed text (the display of the Principles) shall be modeled as one Exit Together with the request to display the Principles, the functional process of displaying the Principles of the Web Advice Module satisfies the requirements of rule c) that a functional process shall comprise at least two data movements.

The display of the Principles of the Web Advice Module can be considered identical to a Utility service (GSOA Utility services), like the Privacy statement service and the Disclaimer service. However, the Privacy statement service and the Disclaimer service are existing services, so only the functionality to invoke the existing services is part of the scope of this measurement (see also the next section). The display of the Principles of the Web Advice Module is new

functionality and should be measured as part of the estimate of the required effort to build the software.

4.1.5 Functional processes that invoke existing services

The Web Advice Module uses two existing services that only need to be invoked without any data entered by the functional user of the Web Advice Module:

- *Disclaimer service*
- *Privacy statement service*

The functionality of these services is not in scope for the measurement, but the Web Advice Module must have functionality to invoke (or call upon) the functionality of these services. That is why we need to identify two functional processes to invoke existing services.

These functional processes appear to the functional user of the Web Advice Module in a similar way as the Display principles functional process. The measurement result is different, however, since the Display principles functional process is in scope for the measurement. The Invoke functional processes only invoke existing SOA services that are out of scope for this measurement.

Communication with other services uses the standard COSMIC model for the exchange of data between two peer pieces of software: the Entry/Exit pair (GSOA Services). For these Invoke functional processes we assume that no feed-back is required (see the diagram in §3.5.1) so these Invoke functional processes of the Web Advice Module therefore only consist of a triggering Entry and an Exit to send a request to invoke a SOA service.

4.1.6 Invoking existing services

The functionality of the Web Advice Module consists of both a business application (governed by the BAG) and SOA-services (governed by the GSOA). Invoking of services via buttons has some resemblance to menu functionality (BAG Menus and the triggering Entry). According to this section of the BAG, menu functionality that "enables the user to move around the software, but which does not launch any functional process" is to be ignored. Pressing the button that enables the user to move towards the Mortgage calculation service is therefore ignored in this measurement. However, pressing either the Disclaimer and Privacy buttons invokes, i.e. launches, a functional process. Counting these two invoke functional processes is therefore in line with both the BAG and the GSOA, even though most of the functionality of these functional processes is outside the scope of the software to be measured.

4.1.7 Display time-out message

Measuring timer functionality requires clear specifications on what functionality is allocated to the hardware or the operating system, and what is specifically allocated to the software part (RTAG Timer functionality). We do not have this information, so we have to make some assumptions.

- *We assume that all functional processes take place via the web server and that the inactivity time relates to the interactions of human users with these functional processes.*
- *We also assume that the inactivity timing is existing functionality of the web server. When the inactivity limit of a customer has been reached, the system clock notifies the web server, which triggers the Web Advice Module to show the inactivity message and delete the customer's data. This is the logical cause of events, since the display of the time-out message and the delete of data is Web Advice Module functionality. Physically, this may be solved in a different way.*
- *The system clock is assumed to be the functional user that triggers the time-out functional process.*

4.1.8 Close Module Session

The user can at any time end the Web Advice Module session by clicking on the 'close window' button of any screen that is headed 'Web Advice Module'. We assume this action causes a functional process of the Web Advice Module to close all windows and to delete any user data that has been entered.

4.2 Identify OBJECTS OF INTEREST and DATA GROUPS

The second step in the mapping phase is to identify the object(s) of interest for each functional process and the corresponding data groups.

Objects of interest are defined as:
Any 'thing' in the world of the functional user that is identified in the Functional User Requirements, about which the software is required to move a data group in or out of the software, or to or from persistent storage It may be any physical thing, as well as any conceptual object or part of a conceptual object. (MM Identifying objects of interest and data groups).

Note that an object of interest is not necessarily the same for all functional users throughout the functionality to be measured. In this Case Study this becomes most visible in the texts that are used for the advice and advice details. The same attributes are part of multiple data groups and multiple objects of interest, depending on the functional user. (See also MM Parameter (code) tables and objects of interest, which deals with this subject)

Data groups are defined as:
A data group is a distinct, non empty and non ordered set of data attributes where each included data attribute describes a complementary aspect of the same object of interest (MM Identifying objects of interest and data groups).The following objects of interest and corresponding data groups can be identified. Note that according to the Data movement uniqueness Rule a) all data describing any one object of interest that is required to be entered into one functional process shall be identified as one data group moved by one Entry (in exceptional cases multiple Entries must be identified). The same equivalent rule applies to any Read, Write or Exit data movement in any one functional process (MM Data

movement uniqueness and possible exceptions). In the measurement practice it suffices therefore to only identify the data group consisting of all data attributes of each object of interest, or only their objects of interest.

Functional process	Object of interest	Data group(s)
Display rent or buy advice	Customer advice	Customer proposition choices General advice Customer advice details
Advice session request form	Customer	Customer data
	Customer advice	Customer proposition choices General advice Customer advice details
Select text	Module application texts	Application texts
Edit text	Module application texts	Application texts
Display principles	Principles	Principles text
Invoke disclaimer service	Disclaimer	Invoke disclaimer request
Invoke privacy statement service	Privacy statement	Invoke privacy statement request
Display time-out message	Inactivity of a specific user	Threshold exceeded message
Close WAM session	WAM session	Close WAM session message

4.2.1 OOI Customer advice

The central object of interest in the Web Advice Module is the Customer advice whether to buy or rent a house. This is the central concept and there are three data groups that describe complementary aspects of the advice:

- *The Customer proposition choices the advice is based on. By default the choice is neutral for each proposition, but can be changed by the customer to reflect his or her personal situation. For each customer five proposition choices are always stored.*
- *The General advice whether to buy or rent a house. Based upon the proposition choices the business rules described in §2.4 lead to an advice to the customer to buy or to rent. For each customer there is only one occurrence (out of the three possible values) at a time.*
- *For each proposition choice the Web Advice Module contains Customer advice details that can be shown to the customer, if relevant, for each proposition choice that led to the general advice. The example in the Advice page in §2.3 shows two advice details that have led to the general advice to rent a house. For each customer up to five occurrences (out of the ten possible values) apply, based on the selected proposition choices.*

These three aspects correspond to different data groups because the aspects have different frequencies of occurrence (BAG Identification of objects of interest, data groups and data movements). The software needs to store all the data groups in a 'web session' to be able to send them to the functional user Back Office Service if the customer wants to have an advice session.

4.2.2 OOI Customer

In order to be able to process a request for an advice session the functional user Back Office Service needs to have information about the Customer. This object of interest only has one data group, containing the Customer data that is to be sent with the request for an advice session. This includes the type of session the Customer requires.

4.2.3 OOI Module application texts

For this object of interest there is one data group that contains all attributes of the dynamic application texts of the Web Advice Module from the perspective of the Application Manager. For the Application Manager, it is irrelevant to distinguish the different types of text as four different data groups describing the OOI Generic advice texts.

4.2.4 OOI Principles

For this object of interest there is only one data group, describing the principles of the Web Advice Module.

4.2.5 OOI Disclaimer

This object of interest only contains one data group, describing the general disclaimer of the bank.

4.2.6 OOI Privacy statement

This object of interest only contains one data group, describing the privacy statement of the bank.

4.2.7 OOI Inactivity of specific user

In order to fulfil the time-out requirement the Web Advice Module receives a signal for every user of which the inactivity time has passed the threshold value.

4.2.8 OOI WAM session

When a user closes a WAM session, the WAM must delete any data stored for that user. The command to close a session is data that changes the state of the OOI WAM session.

4.3 Identify DATA ATTRIBUTES

The third step in the mapping phase is optional, since it has no effect on the measurement of new software. However, it is very useful as a basis for functional size

measurements of future maintenance of the software. When the data attributes are known in detail, it is easier to determine whether a data group has been changed or not.

It is also very useful though to understand the functionality in detail. That is why this optional step is included in the case study.

The data groups identified in the previous section contain the following data attributes:

Data group	Data attributes
Customer proposition choices	Choice #1, choice #2, choice #3, choice #4, choice #5
Customer data	Initials, family name, gender, date of birth, street, number, ZIP-code, city, country, phone number, e-mail address, account number, requested type of session
Module application texts	Introductory text, rent advice, buy advice, mixed advice, proposition #1, proposition #2, proposition #3, proposition #4, proposition #5, buy detail #1, buy detail #2, buy detail #3, buy detail #4, buy detail #5, rent detail #1, rent detail #2, rent detail #3, rent detail #4, rent detail #5
Introductory text *(subset of application texts)*	Introductory text
General advice *(subset of application texts)*	Rent advice, Buy advice, Mixed advice
Generic proposition texts *(subset of application texts)*	Proposition #1, proposition #2, proposition #3, proposition #4, proposition #5
Generic advice details *(subset of application texts)*	Buy detail #1, buy detail #2, buy detail #3, buy detail #4, buy detail #5, rent detail #1, rent detail #2, rent detail #3, rent detail #4, rent detail #5
Customer advice details *(subset of generic details)*	Buy detail #1 *or* rent detail #1 *or* empty Buy detail #2 *or* rent detail #2 *or* empty Buy detail #3 *or* rent detail #3 *or* empty Buy detail #4 *or* rent detail #4 *or* empty Buy detail #5 *or* rent detail #5 *or* empty
Principles text	Principles of the Web Advice Module
Invoke disclaimer request	Request to show the disclaimer
Invoke privacy statement req.	Request to show the privacy statement
Threshold exceeded message	Message that a user exceeded the inactivity threshold
Close WAM session	Close WAM session message

4.4 Discussion on the mapping phase

From the length of the discussion sections in this chapter one might get the impression that the COSMIC method is difficult to apply. The reader should bear in mind that a large portion of this discussion deals with the correct mapping of the requirements to the COSMIC model. This is an important step in the COSMIC measurement process. It is a deliberate choice of the authors to discuss this part in depth, since in practice the interpretation of the requirements is the most important part of the measurement process.

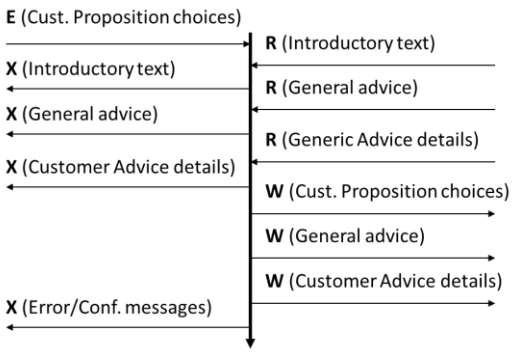
5 Measurement phase

With the functional processes, objects of interest and data groups identified in the mapping phase, the actual measurement can take place by identifying all data movements in each functional process. In the Business Application domain, when a piece of software to be measured can generate messages without user data, by convention one single Exit is identified to represent all of those software messages (MM "Identifying Exits", BAG "Error and confirmation messages").

In this chapter sections in *italics* are explanatory text from the authors that are not a part of the actual measurement process.

Each functional process identified in §4.1 will be described in detail by a message sequence diagram, describing the data movements that make up the functional process.

5.1 Display rent or buy advice



The functional process starts when the customer wants an advice from the Web Advice Module (triggering event) and enters the proposition choices and the trigger to generate an advice, based on the given choices (triggering Entry).

Based on the given choices an advice is generated with advice details.

This functional process has a size of **11 CFP**.

See also the discussion in §4.1.1.

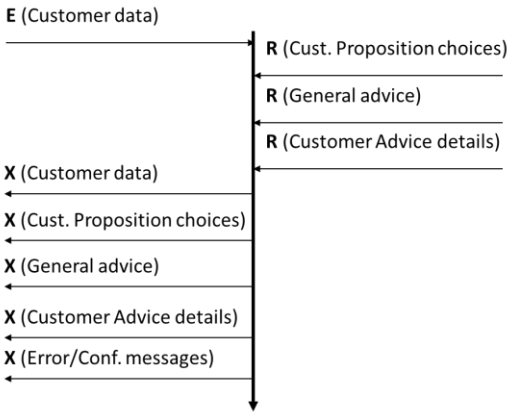
5.1.1 Measuring display rent or buy advice

The advice details are read from persistent storage, based on the entered proposition choices. Since the occurrences of the general advice and the advice details describe different objects of interest (BAG Identification of objects of interest, data groups and data movements) and have different frequencies of occurrence, they are counted as separate data movements.

If a data group survives the functional process using it, it needs to be stored. The fact that the proposition choices from the user are known when a customer fills in an advice form indicates that this data group survives the functional process.

5.2 Request form for an advice session (back office service)

From the application two kinds of advice sessions can be requested (see §2.5). Both requests are handled by the back office and are functionally identical. So only one functional process is identified (MM Identifying functional processes).



The functional process starts when the customers decides to request an advice session (triggering event) and activates a back office form, either to request a rent advice session or a mortgage advice session in which the customer enters his or her data.

This functional process has a size of **9 CFP**.

See also the discussion in §4.1.2.

5.2.1 Measuring advice session forms

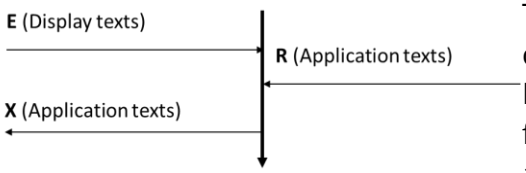
The functional process reads the (customer) proposition choices, the general advice and the advice details that have been stored to be available for this functional process (see §5.1).

The functional process Exits the customer data, the (customer) proposition choices, the general advice and the advice details to the back office. These represent four different objects of interest, so four separate Exits are identified. The requirements of 2.5 deliberately specify that not only the proposition choices, but also the general advice and the advice details have to be sent to the back office. Due to the flexibility to maintain the texts which the customer receives from the Web Advice Module, given a certain set of proposition choices, the requirements want to ensure that the human advisor receives exactly the same input values and advice texts the customer has received.

5.2.2 Measuring the cancel request

Pressing the button to cancel the request on the advice session form screen is not a separate functional process. It is a way to terminate the functional process of requesting an advice session without sending out the request. In the COSMIC method this is called a control command (MM Control commands) which is to be ignored in the measurement. A control command is specific to the business application domain.

5.3 Select text (maintenance functional process)



The Application manager can decide to change one or more editable texts within the Web Advice Module (triggering event) and requests the functional process to read and display all application texts that can be edited.

This functional process has a size of **3 CFP**.

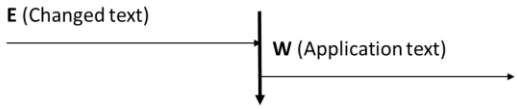
See also the discussion in §4.1.3.

Note that there is no requirement for error or confirmation messages, so they are not counted.

5.3.1 Measuring Select text

The functional process reads all the texts from persistent storage and displays them to the Application manager so he or she can select an application text for editing via the browser. This functional process is the first so-called 'enquire-before-update' process which precedes the 'edit text' step. Both steps are identified as separate functional processes (BAG Separate functional processes).

5.4 Edit text (maintenance functional process)



The decision to change one of the editable application texts (triggering event) and the subsequent update of an application text is a separate functional process (BAG "Separate functional processes"). The changed text is moved by the triggering Entry for this functional process. The changed text is written to persistent storage.

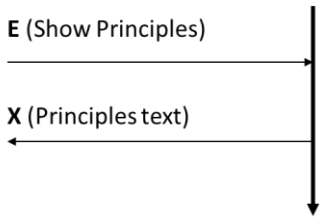
This functional process has a size of **2 CFP**.

See also the discussion in §4.1.3.
Note that there is no requirement for error or confirmation messages, so they are not counted.

5.4.1 Measuring Edit text

This functional process is the second step following the so-called 'enquire-before-update' process in which the text is edited and updated. Both steps are identified as separate functional processes (BAG Separate functional processes).

5.5 Display principles of this module



The functional process starts when the customer wants information about the principles of this website (triggering event) and requests to show information about the principles of the Web Advice Module.

In response a page is shown that shows the principles. The principles are not maintainable. They are contained in the software and therefore do not have to be read.

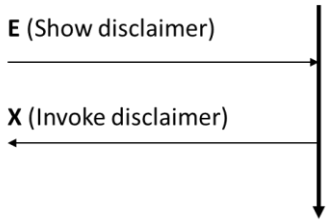
This functional process has a size of **2 CFP**.

See also the discussion in §4.1.4.

5.5.1 Measuring display principles of this module

The facts that the triggering Entry is the result of pressing a button and the output is fixed, non-maintainable text do not affect the result that this is a simple enquiry.

5.6 Invoke Disclaimer service



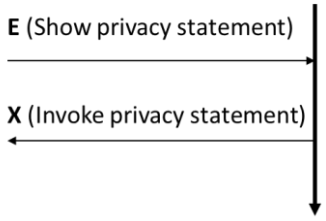
The functional process starts when the customer wants information about the disclaimer (triggering event) and activates the service with the disclaimer by pressing the button.

In response the service is invoked that shows the customer the disclaimer in his or her browser outside the Web Advice Module.

This functional process has a size of **2 CFP**.

See also the discussion in §4.1.5.

5.7 Invoke Privacy statement service



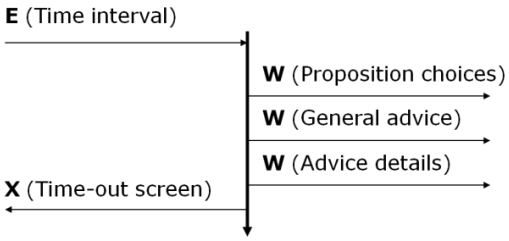
The functional process starts when the customer wants information about the privacy statement (triggering event) and by pressing the button activates the service with the privacy statement.

In response the service is invoked that shows the customer the privacy statement in his or her browser outside the Web Advice Module.

This functional process has a size of **2 CFP**.

See also the discussion in §4.1.5.

5.8 Display time-out message



As stated in §4.1.7 the Web Advice Module uses existing functionality of the web server, which uses the system clock of the operating system. If the inactivity time exceeds the threshold the system clock notifies the web server, which triggers the Web Advice Module to show the inactivity message and deletes the customer's data.

A requirement to delete data is represented by a Write data movement.

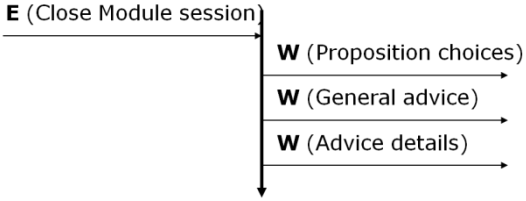
This functional process has a size of **5 CFP**.

See also the discussion in §4.1.7.

5.8.1 Measuring display time-out message

In 4.1.7 we have assumed that the Web Advice Module must delete any data entered by a user if the user's session closes as a result of the Web Advice Module receiving a timeout message ('threshold exceeded') from the system clock. If our assumption is wrong and the deletion of user data is performed by the webserver or some other function of the operating environment, then this functional process would not need the three Write data movements for the deletion of user data.

5.9 Close Module session



The user can close the session of the Web Advice Module by closing any window headed 'Web Advice Module' or by closing his or her browser. As stated in §2.9 the entered data is no longer available. This is represented by Write data movements to delete these data from permanent storage.

This functional process has a size of **4 CFP**.

5.9.1 Measuring close Module session

We have assumed that the Web Advice Model must delete any data entered by a user if the user's session closes as a result of a user clicking on a 'close window' button. If our assumption is wrong and the deletion of user data is performed by the webserver or some other function of the operating environment, then this functional process would not be needed at all.

5.10 List of data movements

Functional process	Data movement	Type	CFP
Display rent or buy advice	Customer proposition choices	E	1
	Introductory text	R	1
	General advice	R	1
	Generic advice details	R	1
	Introductory text	X	1
	General advice	X	1
	Customer advice details	X	1
	Customer proposition choices	W	1
	General advice	W	1
	Customer advice details	W	1
	Error/Confirmation messages	X	1
Advice session request forms	Customer data	E	1
	Customer proposition choices	R	1
	General advice	R	1
	Customer advice details	R	1
	Customer data	X	1
	Customer proposition choices	X	1
	General advice	X	1
	Customer advice details	X	1
	Error/Confirmation messages	X	1
Select text	Display texts	E	1
	Module application texts	R	1
	Module application texts	X	1
Edit text	Selected text	E	1
	Module application text	W	1
Display principles of this module	Principles button	E	1
	Principles	X	1
Invoke Disclaimer service	Disclaimer button	E	1
	Invoke disclaimer	X	1
Invoke Privacy statement service	Privacy statement button	E	1
	Invoke privacy statement	X	1
Display time-out message	Time interval	E	1
	Customer proposition choices	W	1
	General advice	W	1
	Customer advice details	W	1
	Time-out screen	X	1
Close WAM session	Close window	E	1
	Customer proposition choices	W	1
	General advice	W	1
	Customer advice details	W	1
Total functional size			**40**