International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 4, No. 2
May 2015

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 4, No. 2, May 2015

Research Paper

MANUAL AND AUTOMATED FUNCTIONAL SIZE MEASUREMENT OF AN AEROSPACE REALTIME EMBEDDED SYSTEM: A CASE STUDY BASED ON SCADE AND ON COSMIC ISO 19761

Hassan Soubra[1]*, Laury Jacot[1] and Steven Lemaire[1]

*Corresponding Author: **Hassan Soubra** ✉ hassan.soubra@estaca.fr

Software Functional size allows the application of engineering principles to software development, which provides an objective and quantitative base for management decisions. Automating Functional size measurement (FSM) is important for organizations needing to measure a large number of projects in a short time. In this paper, we present related work on FSM in the context of real-time embedded systems (RTES) with particular emphasis on works covering FSM automation and providing case studies. Next, we present a COSMIC based FSM procedure, for RTES designed with SCADE. Finally, we manually apply the FSM procedure to an aerospace system; and we compare the results obtained with those produced automatically by a prototype tool developed at ESTACA. Our experimentation concluded that no difference is detected between the automated and the manual measurements by comparing the results at the detailed level; however, applying FSM procedures manually is tedious, time-consuming and error-prone. The study, on the one hand, addresses the lack of fully detailed illustrations of FSM applied both automatically and manually, on RTES software via a full case study which tackles all the intermediate measurement steps. On the other hand, it demonstrates the time efficiency of FSM automation.

Keywords: Functional size measurement, Real-time embedded systems, SCADE, Automated measurement, COSMIC-ISO 19761, Function points

INTRODUCTION

Functional Size Measurement (FSM) has become an important task in software development projects for real-time embedded systems. Functional size can be used for a number of purposes: to estimate development effort, to manage project scope changes, to measure productivity, to benchmark, and to normalize quality and maintenance ratios. The COSMIC ISO 1976 standard (Lesterhuis and Symons, 2014) has been adopted in many companies, e.g. (Renault, 2012); and has been

[1] École Supérieure Des Techniques Aéronautiques et de Construction Automobile - ESTACA, Rue Georges Charpak, 53061 Laval, France.

the subject of many research papers, e.g. (Soubra and Ebert, 2013).

A measurement method is a generic description of a logical organization of operations used in a measurement (ISO and IEC, 2007). When it comes to applying the generic method of measurement, as defined in a standard, specific measurement procedures must be defined: what type of input is measured, with what steps, with what sizing rules, and with what level of accuracy and verification, etc. If the same measurement procedure is reapplied on the same inputs, the same results should be obtained. If no specific FSM procedures are defined and the generic method is used, but only as a reference, the results may be different. An FSM procedure, or a standard operating procedure, is, according to the International Vocabulary of Metrology (ISO and IEC, 2007), a detailed description of a measurement according to one or more measurement principles and to a given measurement method, based on a measurement model and including any calculation required to obtain a measurement result. A measurement procedure is usually documented in sufficient detail to enable an operator to perform a measurement.

In 2008 (Marín *et al*., 2008) published a survey on measurement procedures based on the COSMIC–ISO 19761 standard. This survey shows that most COSMIC-based FSM procedures target the Management Information Systems domain, while a few others cover the real-time embedded domain. Since then, an FSM procedure for real-time embedded software, based on the COSMIC method, version 3.0.1, and with requirements documented using the Simulink model, was proposed in (Soubra *et al*., 2011). A refinement of the proposed procedure was presented in (Soubra *et al.*, 2012). Another procedure (Lind and Heldal, 2011) has been proposed for real-time software size estimation using a UML profile. Both procedures are based on the COSMIC–ISO 19761 standard, and both are used in the car industry context.

Function point-based measurement methods, such as COSMIC ISO 19761, are applicable at both the beginning of the development process, in the requirements elicitation or specification phase, and at the end of the project, following implementation, for productivity and benchmarking studies.

Nevertheless, when performed manually, FSM still may be considered a complicated, tedious and time-consuming task. It is complicated because each modeling approach and environment need specific “translations”, and, on the other hand, insufficient data quality and environmental constraints require experienced counting to avoid errors. Manual application of FSM is tedious and time-consuming because often staff resources may not be available for measurement purposes within the required time frame. Automating FSM becomes feasible when the requirements are designed using a specific notation, and even more so when the requirements are explicit, safely-constructed and formal. FSM automation is especially attractive for organizations using formal modeling tools for documenting their software requirements.

The Safety-critical Application Development Environment (SCADE) (http://www.esterel-technologies.com/products/scade-system/) is a system design and modeling tool suite developed in 1995 and based on the synchronous language LUSTRE. SCADE allows the design of explicit safely-constructed formal models, where the

interpretation of a model designed using SCADE is unique and reader-independent. SCADE fully supports industrial systems engineering processes - such as ISO 26262 (ISO/DIS 26262, 2009) - and it has a C code generator developed in compliance with DO-178B level A (RTCA Special Committee 167, 1992).

To speed up the measurement process and to reduce the possibility of human error, several attempts have been made to automate FSM methods, initially for the first generation of these methods, such as the IFPUG Function points method (International Organization for Standardization, 2009), and subsequently for the COSMIC method, referred to as a second generation FSM for both the Management Information Systems domain and the real-time domain. However, none of the proposed procedures provides documented evidence of the performance achieved by automation via a fully detailed case study.

To independently illustrate the comparison between manual and automated Functional Size Measurement of real-time embedded aerospace system software designed in SCADE, this paper proposes a case study which tackle all the intermediate measurement steps to ensure that the whole measurement chain produced the right measurement results at a sufficiently detailed level.

The paper is organized as follows: Section 2 presents the related work on COSMIC based FSM procedures in the context of real-time embedded systems. Section 3 presents the COSMIC based FSM procedure proposed for real-time embedded software requirements expressed using SCADE. Section 4 presents the case study carried out on the Roll Control Aerospace Real-time Embedded System. Finally, section 5 presents our conclusions and a discussion of future work.

RELATED WORK

The COSMIC method is referred to as a 2nd generation FSM for both the Management Information Systems domain and the real-time. It has been reported (Lavazza and Morasca, 2013) that the COSMIC method is more suitable than FPA for measuring real-time embedded applications.

An FSM procedure is, according to the VIM (ISO and IEC, 2007), a detailed description of a measurement according to one or more measurement principles and to a given measurement method, based on a measurement model and including any calculation required to obtain a measurement result. A measurement procedure is usually documented in sufficient detail to enable an operator to perform a measurement.

This section presents work on COSMIC based FSM procedures in the context of Real-Time Embedded Systems (RTES) with particular emphasis on works covering FSM automation and providing case studies on RTES FSM.

In Lind and Heldal (2011) , a tool based on COSMIC is presented for measuring the functional size of embedded automotive software early on, using a UML profile that captures all the information needed for functional size measurement according to COSMIC and for estimating Code Size (Lind and Heldal, 2009). Nevertheless, a detailed evaluation of the FSM tool itself is not provided.

In Ergun and Gencel, a manual application of FSM to four projects is presented, three of which are from the real time domain. Both the COSMIC

FFP method and MK II FPA method are used. The case study presented in the paper is from the home appliances domain (with a fridge, a heating system and a shower regulator). The FSM results are summarized in tables without details. It also gives the needed efforts and a comparison with Lines Of Codes (LOC). However, the measurement procedure is not explained in details.

In Soubra *et al*. (2011), an FSM procedure for real time embedded software requirements documented using the Simulink modeling tool was proposed. This procedure is based on the COSMIC method. The rules of this procedure and a tool that automates the measurement procedure are also presented. This study was conducted using the prototype tool developed in collaboration with the École de Technologie Supérieure (Canada) and the University of Versailles at St-Quentin en Yvelines (France). In addition, an example of the measurement of a very simple system modeled in Simulink was described and then used in this work.

In Soubra and Chaaban (2012), the authors presented a guideline for measuring Functional size in accordance with the COSMIC ISO 19761 measurement method for ECU Application Software designed following the AUTOSAR architecture. A case study of a steer-by-wire system is also presented. The measurement results are summarized in a table, however, only the measurement of a part of the system is detailed in this paper as an example. The performed measurement is a manual one and no FSM procedures were provided in this study.

In Talib *et al*. (2007), the authors show that different alternatives for hardware/software allocation from their case study (steam boiler) requirements could lead to different results of functional size. To illustrate their results, they apply the COSMIC-FFP method to those alternatives proposed by the system. Each alternative is explained by text and by sequence diagram. Moreover, the manual measurement is detailed in tables, which contain the process description, data movement (description and type) and finally, the functional size of that process.

In Diab *et al*. (2002), a COSMIC based tool called µcROSE is proposed to automate FSM applied to RRRT models. It is based on a formal mapping between COSMIC-FFP concepts and RRRT model elements. The authors claim that the mapping has been validated with an expert of COSMIC-FFP to ensure its correctness and completeness. However, no detailed evaluation of the verification and the validation of the FSM tool is documented. In addition, neither a case study nor a comparison between the manual application and the automated one are provided in this paper.

In Gencel *et al*. (2005), the authors present the FSM results obtained by applying Mk II FPA and COSMIC FFP methods to a real-time system (a part of an avionics system) which have control as well as algorithmic components. These methods are detailed in a table and a diagram of process of measurement is presented. The manual measurement results are summarized in tables with productivity rates and comparison with LOC. Furthermore, it has been reported that both methods are not very suitable when algorithmic components and conditional statements exist.

In (Ho *et al*.,1999), a comparison between COSMIC FFP, Full Function Point and IFPUG measurement methods is proposed, using a comparative case study, about a Warehouse Software Portfolio. The case study details each method. Results are summarized in tables.

In (Azzouz and Abran, 2004), a tool for the automation of FSM is proposed for systems modelled in Rational Unified Process (RUP). A direct mapping between UML and COSMIC-FFP notations and concepts has been used. The COSMIC-FFP method is detailed. An overview of the tool is presented: its characteristics, main output screens and limitations. No case studies are reported in the study.

In Londeix (2012), an FSM using COSMIC method v3.0 is combined with MAN.6 of automotive Standardization into the automotive software process (SPICE) on a Speedometer component. The Process Man.6 and the case study are well explained. A short description of the measurement is provided. The measurement results are summarized in tables. This procedure was applied manually.

In Turetken *et al*. (2008), the paper discusses how the concept of entity generalization is considered in commonly used FSM methods (the IFPUG FPA method and the COSMIC method) and to investigate how different interpretations of this concept of entity generalization affect the functional size of the software obtained using these methods. The paper presents also a case study conducted on an innovative military software development project. The project was measured by different measurers. However, the authors deal only with the differences among methods in handling the entity generalization and how these affect the measurement results between the functional size figures obtained by different measurers.

In Soubra (2013), the author proposed a fast FSM procedure for safety-critical real-time systems expressed using LUSTRE. This procedure is based on the COSMIC -ISO 19761 measurement method. To clarify the procedure, an example of the measurement of a simple control system described in LUSTRE was introduced and then used in this paper. It was manually measured.

In conclusion, only studies (Soubra et al., 2011; Diab *et al., 2*002; Gencel *et al*., 2005; Azzouz and Abran, 2004; Soubra, 2013) provide detailed measurement procedures for FSM of software, however, no complete and detailed illustrations of the application of the measurements proposed are included in these studies. In addition, studies (Soubra *et al.*, 2011; Diab *et al*., 2002; Azzouz and Abran, 2004) present automations tools developed to implement the procedures proposed, however, they do not illustrate the advantage of FSM automation compared to the manual application of the FSM procedures proposed. Hence, the literature review clearly shows that there is a lack of full case studies covering illustration of RTES FSM and comparing automated and manual measurements at the detailed level.

For these reasons our approach addresses the lack of fully detailed illustrations of FSM applied, on a real-time embedded system software via a full case study and to ensure that the whole measurement chain produced the right measurement results at a sufficiently detailed level, both automatically and manually. The case study is accompanied with screenshots and a short video of the automation prototype tool.

A COSMIC BASED FUNCTIONAL SIZE MEASUREMENT PROCEDURE FOR REAL-TIME EMBEDDED SOFTWARE REQUIREMENTS EXPRESSED USING SCADE

A precise COSMIC size measurement is possible only when the description of the to-be-measured system is in sufficient detail to identify all the

COSMIC elements. SCADE allows the design of explicit safely constructed formal models, where the interpretation of a model designed using SCADE is unique and reader-independent. In order to correctly measure the Functional size of safety-critical real-time systems designed using SCADE, different elements must be identified and well defined: Functional Process, data groups movements, etc. This section presents a set of mapping rules, based on COSMIC -version 4.0, to be applied to the system designed using SCADE, in order to obtain its Functional size.

COSMIC Overview

The COSMIC method provides a standardized method for measuring the Functional size of software from both the business application domain (also referred to as Management Information Systems or MIS) and the real-time domain. The COSMIC method is considered a second-generation FSM.

This method has been accepted as an International Standard: ISO/IEC 19761 Software Engineering – COSMIC – A Functional size measurement method (hereafter referred to as ISO 19761). The latest version of the COSMIC manual available on the COSMIC website is version 4.0. While this release includes a number of refinements, the original principles of the COSMIC method have remained unchanged since they were first published in 1999.

The COSMIC method measures the Functional User Requirements (or FUR) of software. The result obtained is a numerical 'value of a quantity' (as defined by the ISO) representing the Functional size of the software.

Functional sizes measured by the COSMIC method are designed to be independent of any implementation decisions embedded in the operational artifacts of the software to be measured. This means that the FUR can be extracted not only from actual software already developed but also from the model of the software before it is implemented.

The measurement process of the COSMIC method, version 4.0, consists of three phases:

1. The COSMIC Software Context Model is applied to the software to be measured. This is the Measurement Strategy Phase.
2. The COSMIC Generic Software Model is applied to the software to be measured. This is the Mapping Phase.
3. The Measurement Phase, in which the actual size measurement results are obtained.

The measurement result corresponds to the Functional size of the FUR of the software measured, and is expressed in COSMIC Function Points (or CFP).

In COSMIC, a Functional Process is an elementary component of a set of FURs comprising a set of data movements that are unique, cohesive, and independently executable. A Functional Process is triggered by an 'Entry' data movement from a Functional User (FU) that informs the piece of software that the Functional user has identified a triggering event. It is complete when it has executed all that is required in response to the triggering event. According to the COSMIC method, software Functionality is embedded within the Functional flows of data groups. These data flows can be characterized by four distinct types of data movements. Two types of movements (E: Entries and X: Exits) between the FU and COSMIC FP, allow the exchange of data with FU across a boundary. Two types of movements (R: Reads and W: Writes) between a COSMIC FP and the persistent storage, allow the exchange of data with the persistent storage hardware.

Different abstractions are typically used for different measurement purposes. In real-time software, the users are typically the engineered devices that interact directly with the software, that is, the users are considered the I/O hardware.

As an FSM method, COSMIC is aimed at measuring the size of software based on identifiable FUR. Figure 1 shows an example of a generic representation with two Functional users, FU1 and FU2. Three Functional processes are identified in this example: FP1, FP2, and FP3. The data group movements are represented by directed arrows in Figure 1. There can only be one persistent storage logically (even though the persistent storage can, in fact, be physically distributed) for a given Functional process. The persistent storage is represented in blue and in conformity with the ISO 5807 stored data symbol.

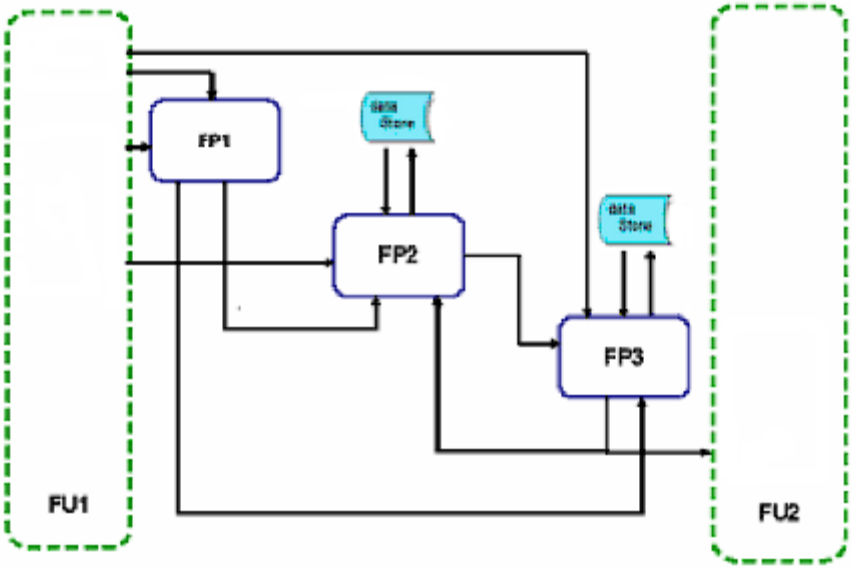
In FP1, 4 data group movements are identified: 2E from FU1; 1X to FP2; and 1X to FP3. In FP2, 6 data group movements are identified: 1W+1R (from/to persistent storage), 1E from FU1+1E from FP1, and 1E from FP3; in addition, 1X to FP3. In FP3, 6 data group movements are identified: 1W+1R (from/to persistent storage), 1E from FU1, 1E from FP1 and 1E from FP2; in addition, 1X to FP2/FU2 (Note: only 1X is identified here because the same Exit Data Movement is going to different elements: same source (FP3) with different destinations (FP2, FU2)).

It is impossible to apply the COSMIC's Measurement Phase directly, because COSMIC defines generic concepts and descriptions. An FSM procedure specific to an input type (for example, SCADE models) should be designed first. It could be defined as a set of operations, described explicitly and used in the performance of a particular measurement according to the generic concepts and descriptions of the method. And naturally, the purpose and scope(s) of the measurement should be defined as well.

SCADE Overview

SCADE (http://www.esterel-technologies.com/products/scade-system/) (Safety-Critical Application Development Environment) is an

Figure 1: Generic flow of data groups through software from a Functional perspective in COSMIC – ISO 19761



industrial development tool used in many major companies developing safety-critical embedded systems. It is based on the LUSTRE synchronous language. LUSTRE (Halbwachs *et al*., 1991) is a strongly typed declarative data-flow synchronous language having simple semantics.

Models in SCADE are built using hierarchical block diagrams. A system designed in SCADE is considered to be formal and safely-constructed. This property is particularly very interesting in the context of safety-critical reactive real-time systems because it guarantees a complete and unambiguous system design.

A system described in SCADE is structured in Nodes and/or Functions and has a cyclic behavior. Each Node/Function contains a set of blocks and takes at least one input and produces at least one output. SCADE Inputs and Outputs are considered as "Interface" variables; local variables (i.e., "Locals") can be defined in SCADE as well. Each variable denotes a flow -also called a stream- that can be seen as a sequence of values. For example, in Table 1, variable B represents a flow which is the sequence of values: (b*1*, b*2*, b*3*, ..., b*n-1*, b*n*).

All input variables have the same basic clock, which can be seen as the Boolean flow: basic = (true, true, …, true…). All other possible clocks are defined internally using this basic clock. SCADE possesses a clock calculus mechanism that ensures that any system described in SCADE has a well-defined meaning. The compiler of SCADE rejects a system if the constraints on clocks are not satisfied.

SCADE supplies a set of operator blocks, such as: arithmetic (e.g., the "Plus" block, the "Minus" block), Boolean (e.g., the "And" block, the "Or" block) and control operator blocks (e.g., the "If..Then..Else" block). Furthermore, LUSTRE provides sequential operator blocks such as: the "Previous" operator gives access to the cyclically previous value of its argument: for example, "pre(B)" is the flow (nil, b*1*, ..., b*n-1*, ...), where its first value is an undefined one ("nil" for "non-initialized") and its following values are equal to the cyclically previous values of the flow of B. On the other hand, the "Followed by" operator block combines two flows (e.g. A->B; flows A and B),

Table 1: Example of Variables/Equations and Their Flows

Variable	Cycle *1*	Cycle *2*	Cycle *3*	…	Cycle *n*
A	a1	a2	a3	…	an
B	b1	b2	b3	…	bn
C=A+B	c1= a1+b1	c2= a2+b2	c3= a3+b3	…	cn= an+bn
pre(B)	*nil*	b1	b2	…	bn-1
A->B	a1	b2	b3	…	bn
A-> *pre*(B)	a1	b1	b2	…	bn-1

where the first value of the resulting flow is defined using the first flow (e.g. a*1*), and its following values are always equal to the second flow (e.g.: (b*2*, b*3*, ..., b*n-1*, b*n*)).

Table 1 shows an example of the values of six different flows in each cycle: variables A, B and C -which is the sum of A and B; equations pre(B), A->B and A->pre(B). For instance, the value of the flow "A->B" in Cycle 2 is b*2*.

Functional Size of Safety-Critical Real-Time Systems Described in SCADE

FSM procedures must have a set of clear and consistent measurement rules to be applied in order to obtain accurate measurement results and to facilitate the measurement process. The number of these rules varies from one procedure to another. For instance in (Diab *et al.*, 2002), a set of 14 measurement rules is presented for measuring ROOM specifications using the COSMIC method and (Soubra *et al*., 2011) introduces a set of 19 measurement rules for real-time embedded software requirements documented using the Simulink modeling tool.

In this section, the main concepts defined in COSMIC are mapped to the main concepts of SCADE. An explanation is also given to underline the reasons that lead to the mapping proposed in this paper. The rules of the FSM procedure proposed cover all the measurement rules of the COSMIC FSM method. It has been verified and approved by a member of the COSMIC Measurement Practices Committee.

The Measurement Strategy Phase

Purpose: the purpose of this FSM procedure is to measure the COSMIC Functional size of the FUR of any real-time embedded system based on its SCADE model. In other words, this procedure allows obtaining COSMIC Functional size in CFP of a system directly using its FUR described using SCADE.

Scope of the measurement: a system in SCADE is structured using "Nodes" and/or "Functions". The behavior of a system described in SCADE, hence its Functionality in terms of tasks, is defined using these Nodes and Functions. The scope of this FSM procedure is the SCADE "Nodes" and "Functions".

Level of granularity: SCADE Nodes and Functions encapsulate the Functionality of a system. The level of granularity of this procedure is therefore at the Node/Functions description detail level of SCADE. It is at this level the Functional requirements of the system are expressed.

Functional User: According to SCADE, a system consists of one or more Node(s) having input and output flows. Hence, each 'entity' interacting with the system to be measured is a Functional user of the system to be measured. These 'entities' are usually other real-time systems or Hardware devices.

The Mapping Phase

Functional Process: a system in SCADE is defined using Nodes and Functions, which can be seen as bundles of equations. Nodes/ Functions receive, manipulate, and produce data flows. Hence, SCADE Nodes and Functions are COSMIC Functional Processes. They are identified using rule 1 in Table 2.

Boundaries: the boundary lies between the system to be measured, and its Functional User(s).

Data groups: In order to correctly identify the movements of data groups in each Functional

Table 2: The Mapping Rules of Cosmic Data Group Movements to Scade Elements

Rule number	SCADE Element	COSMIC Element	Rule description
1	Node/Function	Functional Process (FP)	Identify 1 Functional process for each Node/Function identified in the system
2	Input Flow	Entry (E)	Identify a 1E data movement for each Input flow identified in a Node/Function
3	Output Flow	Exit (X)	Identify a 1X data movement for each Output flow identified in a Node/Function

process, data groups should first be identified. An object of interest is any element (physical or conceptual) about which the piece of software being measured must process or store data. A data movement moves a single data group that consists of one or more data elements. All attributes in a data group describe the same one object of interest. In SCADE, flows provide transmission and reception of atomic data elements. Flows consist of finite or infinite sequences of values of some scalar type, for a variable or an equation. Each SCADE flow constitutes hence a COSMIC data group.

The Measurement Phase

The COSMIC triggering events in the context of SCADE are implicit: the basic clock of a Node/ Function is defined by its input flows. It is this basic clock that triggers the Node.

The COSMIC Entry (E) and Exit (X) data movements are identified using the input flows, which are sequences of data values received by Nodes, and the output flows, which are sequences of data values produced by Nodes.

Since SCADE allows the use, at each instant, of past values of flows to produce other flows, no data storage mechanisms are needed inside a Node. SCADE manages the availability of these values automatically. Hence, no SCADE elements can be mapped to COSMIC Read (R) and Write (W) data movements.

Conclusively, the only two possible types of COSMIC data group movements in the context of SCADE are the Entry (E) and Exit (X) data movements. They are identified in this phase using the mapping rules in Table 2.

Once all the data movements in a Functional process have been identified, the standard value of 1 CFP is assigned to each data movement. The final step consists of aggregating the results to obtain the Functional size of each Functional process. The Functional sizes of the Functional processes are next aggregated to obtain the Functional size of the system being measured. The rules for obtaining the Functional size of each Node (Functional Process) and the whole system are presented in Table 3.

The FSM Automation Prototype-Tool

Automating an FSM procedure should help in applying it and using it. An automation tool should implement an algorithm that must follow the FSM

Table 3: Rules For Obtaining The Size Of The Functional Processes And The Whole System

Rule number	Rule description
4	Aggregate the CFP related to the data movements identified in a specific Node (FP) to obtain the Functional size of that Node/Function.
5	Aggregate the CFP related to the data movements of (identified in) the Nodes/Functions of (identified in) the whole system to obtain the Functional size of that system.

procedure applied manually step by step. The number of steps to be followed varies from an algorithm to another. For example, the automation algorithm proposed in Soubra *et al*. (2011) consists of 22 steps and sub steps.

The FSM automation prototype tool presented in this section is freely available online on the ESTACA's website (http://www.estaca.fr/component/jdownloads/viewdownload/7/3). The automation algorithm of the FSM procedure proposed in this paper implements the 5 rules in Tables 2 and 3 for obtaining the Functional size. It begins with identifying the Functional processes. Next, the data movements in each Functional Process (FP) are identified and assigned a numerical value of 1 CFP each. Finally, the sizes of all the identified data movements of all the FP are aggregated into a single Functional size value.

The prototype identifies all the packages in a project's ETP file. Next, for each package identified, it opens the XSCADE file of the same name. SCADE Nodes and Functions could be located in different files.

The automation algorithm proposed in this paper consists of 7 steps and sub steps:

1. Search for all the *Nodes* and *Functions* in the XSCADE file using the "Operator tags". Each *Node* and each *Function* are FP.
2. For each identified FP, search all its flows:
3. For each flow, identify all elements called *Input* or *Hidden* using the "Variable Tag". Each *Input* and each *Hidden* are Entry data movements (E).
4. For each flow, identify all elements called *Output* using the "Variable Tag". Each Output is an Exit data movement (X).
5. Assign 1 CFP value for each Data Movement identified
6. Aggregate the size of all the Data Movements (i.e. the Entries, Exits identified in step 2.1 and step 2.2) inside that FP
7. Aggregate the size of all the FP identified in step 1.

Figure 2 shows our prototype-tool's interface: it has a main window with 3 tabs. In the FSM tab, users select the file to-be-measured and the dumping files (PDF and XML) if the "Save when measuring" checkbox is checked. The "Compare" is still under development and the "About Us" tab gives information about the tool. A "Measure" button appears so that the user triggers the measurement process. When

Figure 2: Interface of the prototype

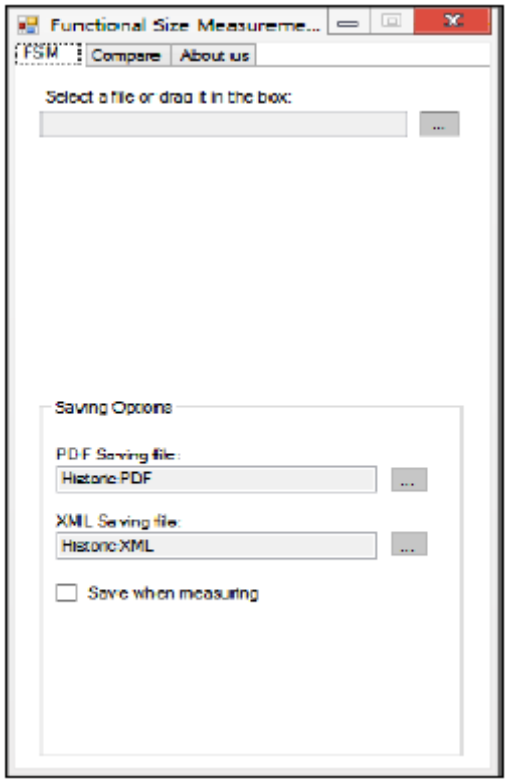
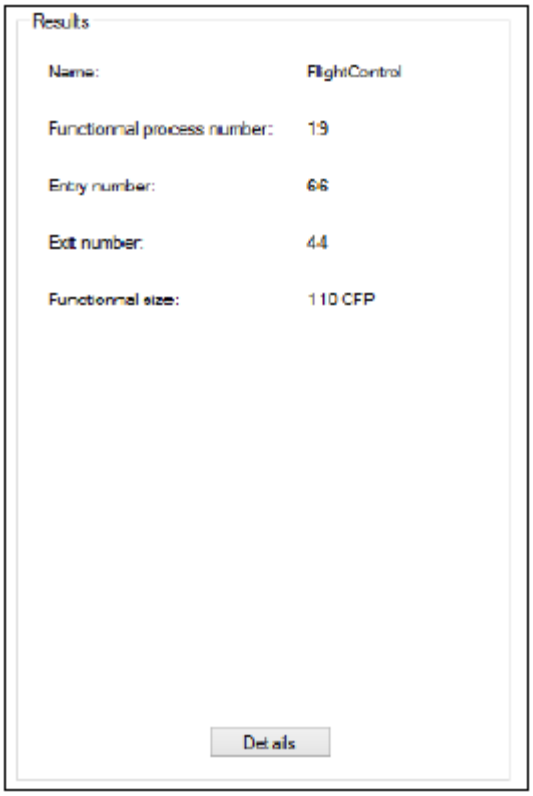


Figure 3: Measurement Results window



clicked, a pop up window (see Figure 3) shows the measurement results.

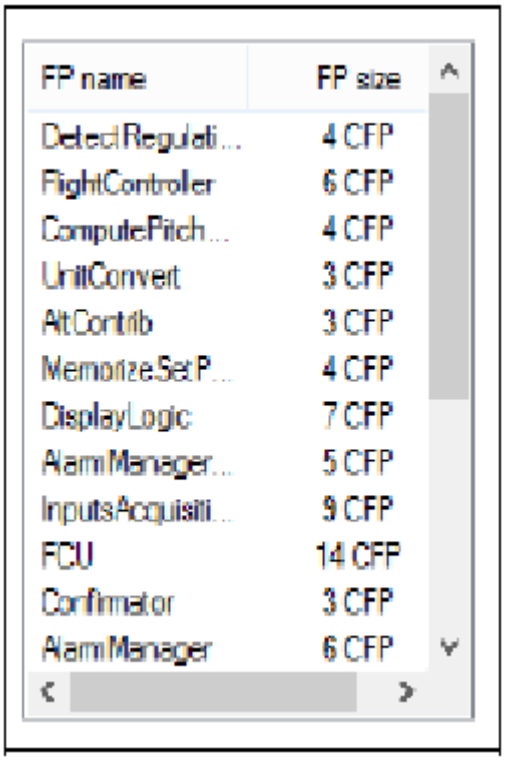
When clicked, the "Details" button shows the sizes in CFP of each FP identified (see Figure 4) and opens a pdf file with the details of the Data Movements identified in the FP identified (see Figure 5).

Figure 5 shows, as an example, a fragment of the pdf document produced when clicking the "Details" button. It gives the total FP number identified, in addition to the total number of Entries, Exits of the system. It also gives the size in CFP of the system. Next, it shows the measurement details per FP, Entries and the Exits Data Movements identified with the names of their corresponding Flows per package.

In addition to the *Roll Control* system presented in section 4, the automation prototype tool was applied on a set of 5 distinct systems designed using SCADE: *Pilot*; *Flight control*; *Digital Stop Watch*; *Cruise Control* and *ABC_N*.

For verification purposes (see Table 4), the systems have been measured manually, using the proposed measurement procedure (the procedure implemented by the prototype tool). Thus, in addition to the Functional size obtained, all the Functional processes and data groups' movements are identified by the manual measurement. We have also kept track of the time needed to manually apply the measurement procedure and to document the measurement results obtained using Excel sheets. The time needed varies from 22 min to 151 min, according

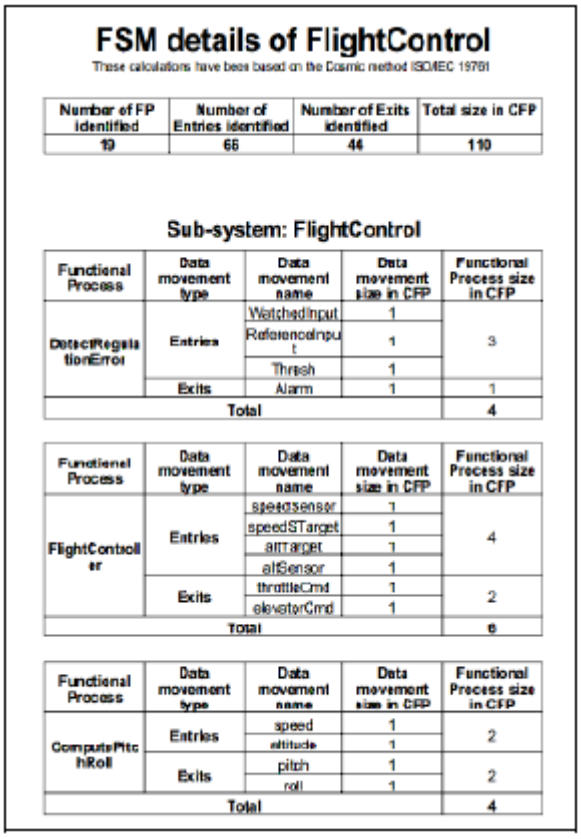
Figure 4: Functional size per FP identified



to the measured system's size. In contrast, the prototype tool produced the results almost instantly, including the detailed documentation, for all the systems measured.

In (http://www.youtube.com/watch?v=zsAZGHEzEOE), a short video presents how to use the prototype on the Cruise Control example.

Figure 5: Fragment of the pdf details file produced



FSM details of FlightControl

These calculations have been based on the Cosmic method ISO/IEC 19761

Number of FP identified	Number of Entries identified	Number of Exits identified	Total size in CFP
19	66	44	110

Sub-system: FlightControl

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
DetectRegulationError	Entries	WatchedInput	1	3
		ReferenceInput	1	
		Thresh	1	
	Exits	Alarm	1	1
Total				4

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
FlightController	Entries	speedSensor	1	4
		speedSTarget	1	
		attTarget	1	
		altSensor	1	
	Exits	throttleCmd	1	2
		elevatorCmd	1	
Total				6

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
ComputePitchRoll	Entries	speed	1	2
		attitude	1	
	Exits	pitch	1	2
		roll	1	
Total				4

CASE STUDY: FUNCTIONAL SIZE OF THE ROLL CONTROL REAL-TIME SYSTEM DESCRIBED IN SCADE

In this section, the "Roll Control" example

Table 4: Functional Size of The Systems Measured

System	Number of FP identified	Number of Entries identified	Number of Exits identified	Total size in CFP	Total Manual Measurement time in minutes
Pilot	24	45	32	77	105
Flight Control	19	66	44	110	151
Digital Stop Watch	6	15	11	26	39
Cruise Control	11	45	20	65	92
ABC_N	3	7	6	13	22
Roll Control	5	12	9	21	31

released into the public domain and is available in the SCADE Suite by Esterel Technologies.

Description of the Roll Control System

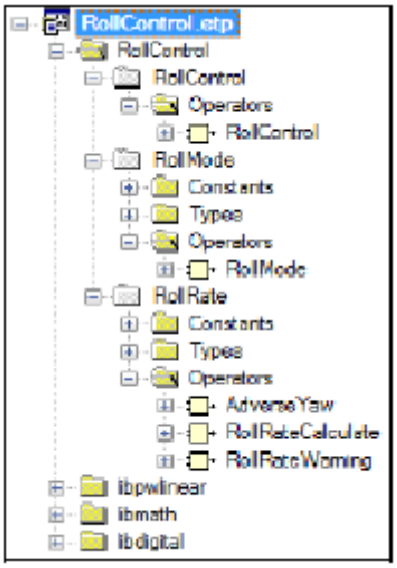
The Roll Control is a simple real-time embedded system that simulates the place roll rate control.

The system is composed of three packages named RollControl, RollMode and RollRate (see Figure 6). It also uses three libraries called "libpwliear", "libmath" and "libdigital". The system's root package is called Roll Control.

The Roll Control Package

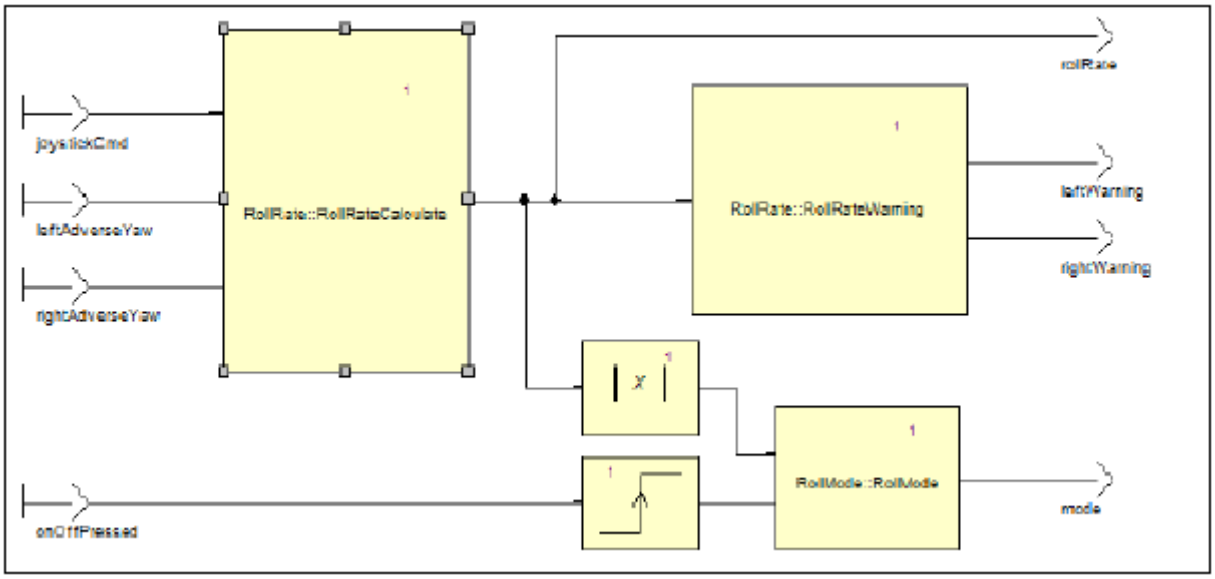
The RollControl package is composed of 1 Node: *RollControl*. Figure 7 shows the *RollControl* Node, which is composed of 2 Functions (*RollRateCalculate* and *RollRateWarning*) and 1 Node: *RollMode*. The RisingEdge (*libdigital* library) block detects the transition from a false to a true value; the output is brought to false after the transition clock cycle. The Abs block (*libmath* library) gives the absolute value of its input. RollControl has four inputs and four outputs:

Figure 6: Hierarchy of the Roll Control System



- LeftWarning: A Boolean data type output which inform, thanks to an alarm, that the aircraft roll rate is upper than -15Ú/s on the left wing.
- RightWarning: A Boolean data type output which inform, thanks to an alarm, that the aircraft roll rate is upper than +15Ú/s on the

Figure 7: The RollControl Node



the roll rate of the aircraft.

right wing.

- Mode: a TRollmode data type output that has 3 different values: off, Nominal and Failsoft.

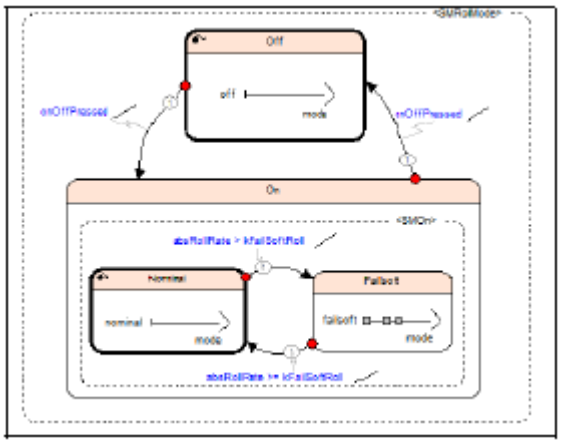
The RollMode Package

The RollMode package is composed of 1 Node: *RollMode*. Figure 8 shows the RollMode Node defined as state machine, which define the state of the output "mode" presented in the RollControl Node.

This Node has two inputs and one output:

- AbsRollRate: A Real data type input which inform if the roll rate is upper than 20Ú/s. It is a safety variable.
- OnOffPressed: A Boolean data type input which informs the system state (on or off)
- Mode: a TRollmode output type that has three different values:
- Off: if onOffPressed is False
- Nominal: if onOffPressed is true and absRollRate is lower than 20 °/s

Figure 8: The RollMode Node

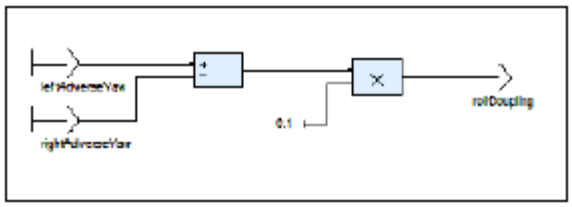


- Failsoft: if onOffPressed is true and absRollRate is upper of equal to 20 °/s

The RollRate package

The RollRate package contains three Functions:

Figure 9: The AdverseYaw Function



AdverseYaw (Figure 9), *RollRateCalculate* (Figure 10) and *RollRateWarning* (Figure 11).

The AdverseYaw Function contains 1 subtract block and 1 multiply block. It has 2 inputs and 1output as shown in Figure 5:

- LeftAdverseYaw: a Real data type input expressing the adverse due to the left wing.
- RightAdverseYaw: a Real data type input expressing the adverse due to the right wing.
- RollCoupling: a Real data type output which gives the value of the Coupling of both wings (it is obtained by subtract the rightAdverseYaw value to the leftAdverseYaw's, and multiply this result by 0.1).

RollRateCalculate uses the AdverseYaw Function. It contains three inputs and one output:

- JoystickCmd: a Real data type input expressing the joystick position.
- LeftAdverseYaw: a Real data type input expressing the adverse due to the left wing.
- RightAdverseYaw: a Real data type input expressing the adverse due to the right wing.
- RollRate: a Real data type output expressing the roll rate of the aircraft.

The block connected to the *rollRate* output flow is called limiter Symmetrical. It can be found in

Figure 10: The RollRateCalculate Function

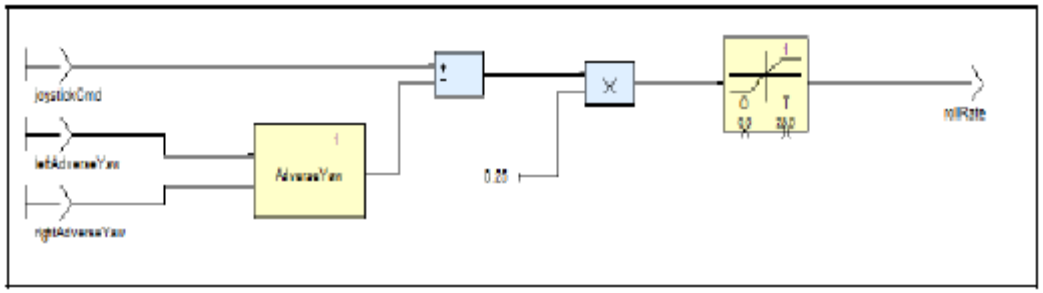
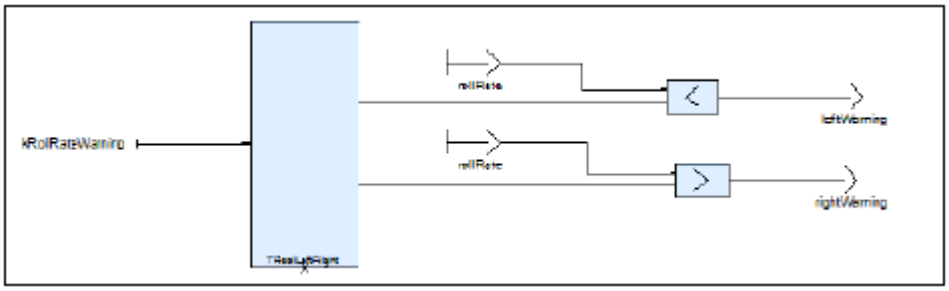


Figure 11: The RollRateWarning Function



llCoupling (value obtained in AdverseYaw and multiplied by 0.25) to the upper and lower SCADE's libpwlinear library. It compares an input (the difference between joystrickCmd input and rolimit. The output rollRate will be either the upper limit if the input is higher than it, either lower limit if the input is lower than it, or the input if it is contained between the upper and lower limits.

RollRateWarning contains one input and two exits:

- RollRate: A Real data type output expressing as it is called, the roll rate of the aircraft.
- LeftWarning: A Boolean data type output which inform, thanks to an alarm, that the aircraft roll rate is upper than —15 °/s on the left wing.
- RightWarning: A Boolean data type output which inform using an alarm that the aircraft roll rate is upper than +15 °/s on the right wing.

RollRateWarning computes the left and right warning alarms according the aircraft roll rate, by comparing it to the kRollRateWarning value. The TRealLeftRight block allows the segregation of kRollRateWarning into 2 values for the right wing and for the left one.

Functional Size of the Roll Control System

This section presents the measurement results obtained first when applying manually the FSM procedure proposed to the SCADE Roll Control system; and next, using the prototype tool presented in section 3.

Manual Application of the FSM to the Roll Control System

In this section, the procedure detailed in section 3 is applied to the RollControl system presented in section 4.

The RollControl is composed of two Nodes:

RollControl and *RollMode*; and three Functions: *AdverseYaw*, *RollRateCalculate* and *RollRate Warning*. Hence, according to rule 1 of Table 1, 5 Functional Processes are identified.

The *RollControl* Node has 4 inputs: joystickCmd, leftAdverseYaw, rightAdverseYaw and onOffPressed; hence 4 Entry data group movements are identified according to rule 2 of Table 1. The 4 output flows of the Node *RollControl* are: rollRate, leftWarning, rightWarning and mode; in consequence 4 Exit data group movements are identified using rule 3 of Table 1. According to rule 4 of Table 1, the total Functional size of the Node is *RollControl* is 8 CFP.

The *RollMode* Node has 2 inputs: absRollRate and onOffPressed; hence 2 Entry data group movements are identified according to rule 2 of Table 1. The Node *RollControl* has 1 output flow: *mode*; in consequence 1 Exit data group movement is identified using rule 3 of Table 1. According to rule 4 of Table 1, the total Functional size of the Node is *RollMode* is 3 CFP. 18

The AdverseYaw Function has 2 inputs: leftAdverseYaw and rightAdverseYaw; hence 2 Entry data group movements are identified according to rule 2 of Table 1. The Function AdverseYaw has 1 output flow: rollCoupling. 1 Exit data group movement is identified following the rule 3 of Table 1. According to rule 4 of Table I, the Functional size of this Function is 3CFP.

The Roll Rate Calculate Function has 3 inputs: joystick Cmd, left Adverse Yaw and right Adverse Yaw. Following the rule 2 of Table 1, 3 Entry data group movements are identified. The Function Roll Rate Calculate has 1 output, rollRate, in consequence 1 exit data group movement is identified using rule 3 of Table 1. According to rule 4 of Table 1, the total Functional size of the Function rollRateCalculate is 4 CFP.

Finally, the RollRateWarning Function has 1 input: rollRate, which allows to identify thanks to the rule 2 of Table 1, 1 Entry date group movement. In the same way, the RollRateWarning has 2 output: leftWarning and rightWarning. In this case, 2 Exit date movements are identified according to rule 3 of Table 1. The RollRateWarning Function Functional size is 3 CFP, following the rule 4 of Table 1.

To obtain the Functional size of the whole system, the size of all the Functional processes of the system are aggregated (rule 5): Size (Roll Control System) = Size (*RollControl*) + Size (*RollMode*) + Size (*AdverseYaw*) + Size (*RollRateCalculate*) + Size (*RollRateWarning*) = 21 CFP.

The Functional sizes of all the Functional processes identified in the Roll Control system are presented in Table 5.

Measuring the Roll Control System using the prototype tool

Figure 12 shows the measurement result of the Roll Control System obtained with the proposed prototype tool. The prototype tool for automating the FSM identified 5 Functional processes: *RollControl*, *RollMode*, *AdverseYaw*, *RollRate Calculate* and *RollRateWarning*, for a total Functional size of 21 CFP:

1. In the Functional process of *RollControl*, it identified 8 data movements: 4 Entry data movements and 4 Exit data movements (Sub-total = 8 CFP).
2. In the Functional process of *RollMode*, it identified 3 data movements: 2 Entry data movements and 1 Exit data movement (Sub-total = 3 CFP).

Table 5: Functional Size of The Roll Control System

Names of Nodes or Functions/FP identified	Flows/data group movements identified	Type of data group movements identified	Functional Process Size, in CFP
RollControl	joystickCmd	E	1
	leftAdverseYaw		1
	rightAdverseYaw		1
	onOffPressed		1
	rollRate	X	1
	leftWarning		1
	rightWarning		1
	Mode		1
Total			8 CFP
RollMode	absRollRate	E	1
	onOffPressed		1
	Mode	X	1
Total			3 CFP
AdverseYaw	leftAdverseYaw	E	1
	rightAdverseYaw		1
	rollCoupling	X	1
Total			3 CFP
RollRateCalculate	joystickCmd	E	1
	leftAdverseYaw		1
	rightAdverseYaw		1
	rollRate	X	1
Total			4 CFP
RollRateWarning	rollRate	E	1
	leftWarning	X	1
	rightWarning		1
Total			3 CFP
Total of the system			21 CFP

3. In the Functional process of *AdverseYaw*, it identified 3 data movements: 2 Entry data movements and 1 Exit data movement (Sub-total = 3 CFP).
4. In the Functional process of *RollRate Calculate*, it identified 4 data movements: 3 Entry data movements and 1 Exit data movement (Sub-total = 4 CFP).

Figure 12: Measurement results using the prototype tool

FSM details of RollControl

These calculations have been based on the Cosmic method ISO/IEC 19761

Number of FP identified	Number of Entries identified	Number of Exits identified	Total size in CFP
5	12	9	21

Sub-system: RollRate

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
AdverseYaw	Entries	leftAdverseYaw	1	2
		rightAdverseYaw	1	
	Exits	rollCoupling	1	1
Total				3

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
RollRateCalculate	Entries	joystickCmd	1	3
		leftAdverseYaw	1	
		rightAdverseYaw	1	
	Exits	rollRate	1	1
Total				4

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
RollRateWarning	Entries	rollRate	1	1
	Exits	leftWarning	1	2
		rightWarning	1	
Total				3

Sub-system: RollControl

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
RollControl	Entries	joystickCmd	1	4
		leftAdverseYaw	1	
		rightAdverseYaw	1	
		onOffPressed	1	
	Exits	rollRate	1	4
		leftWarning	1	
		rightWarning	1	
		mode	1	
Total				8

Sub-system: RollMode

Functional Process	Data movement type	Data movement name	Data movement size in CFP	Functional Process size in CFP
RollMode	Entries	absRollRate	1	2
		onOffPressed	1	
	Exits	mode	1	1
Total				3

5. In the Functional process of *RollRateWarning*, it identified 3 data movements: 1 Entry data movements and 2 Exit data movement (Sub-total = 3 CFP).

CONCLUSION

To correctly measure the functional size of real-time embedded systems requirements documented and modeled using commercials tools, FSM procedures must be designed and well defined. This paper has proposed a Functional Size Measurement (FSM) procedure for real time embedded software requirements documented using the SCADE modeling tool. This procedure is based on the newest version of the COSMIC measurement method, version 4.0, which is designed, unlike the other traditional FSM methods, to measure both real time and management information systems.

FSM may be considered a time-consuming task when performed manually and also because staff resources may not be available for measurement purposes within the required time frame. The rules of the procedure presented in this paper can be considered as a set of rules that allow mapping the SCADE conceptual elements to the COSMIC concepts. FSM procedures must have a set of clear and consistent measurement rules to be applied; in order to obtain accurate measurement results, to facilitate the measurement process, and to permit automation of the procedures. The number of these rules varies from one procedure to another. The measurement procedure presented introduces 5 measurement rules to be applied in order to obtain the functional size of safety-critical real-time systems described using SCADE. This is rather a small number of rules, compared to other the FSM procedure, because of the advantages that SCADE provide over other commercial modeling tools in terms of structure, unambiguity and completeness.

However, the manual application of our procedure to a set of large systems still takes time and requires specialized expertise especially when these requirements are also complex.

Measurement with automated tools eliminates possible measurement variances caused by different measurers, which may lead to different measurement results for the same set of

requirements. That is why a tool that automates the measurement procedure while ensuring the accuracy of the measurement results is useful, and can benefit organizations in terms of reducing the workload of measurement specialists, as well as eliminating measurement delays. This work also provides a basis for the development of other types of automated measurement tools.

We also presented overviews on the SCADE model and the COSMIC method, and explained how to map the COSMIC method to the SCADE model. To clarify the rules, an example of the measurement of the Roll Control real-time embedded system modeled in SCADE was described and then used in this work as a detailed case study. The Roll Control system has been measured both manually and using the automation prototype tool proposed in this paper. The results of our experimentations showed no difference exists between the automated and manual measurements by comparing the results at the detailed level; however, applying FSM procedures manually is tedious, time-consuming and error-prone.

As an open question, one can discuss the impact of this premise: if SCADE had not been used but another modeling tool had been used instead to express the FUR of a system, would the same FSM size in CFP come up? The answer to this open question is currently being under study along with translation mechanisms to ensure the accuracy and precision and equivalence between different measurement procedures applied to the same FUR but which are documented and modeled using different commercials tools.

REFERENCES

1. Azzouz S and Abran A (2004), "A proposed measurement role in the rational unified process and its implementation with ISO 19761: COSMIC-FFP", Software Measurement European Forum, Rome, Italy.
2. Diab H, Koukane F, Frappier M and St-Denis R (2002), "µcROSE: Functional Size Measurement for Rational Rose RealTime".
3. Ergun A N and Gencel C, "A Case Study on Functional Size Measurement Based Effort Estimation for Embedded Systems."
4. Esterel Technologies, http://www.esterel-technologies.com/products/scade-system/
5. FSM prototype tool, http://www.estaca.fr/component/jdownloads/viewdownload/7/3
6. Gencel C, Demirors O and Yuceer E (2005), "A case study on using functional size measurement methods for real time systems", Proc. of the 15th Intl Workshop on Software Measurement (IWSM), Montreal, Canada, pp. 159-178.
7. Halbwachs N, Caspi P, Raymond P and Pilaud D (1991), "The synchronous dataflow programming language Lustre", In Proceedings of the IEEE, Vol. 79, pp. 1305-1320.
8. Ho V T, Abran A and Fetcke T (1999), "A comparative study case of COSMIC-FFP, full function point and IFPUG methods", Department of Informatics, University of Quebec at Montreal, Canada.
9. International Organization for Standardization (2009), "ISO 20926:2009 IFPUG - ISO/IEC 20926:2009. Software Engineering - Software and systems engineering - Software measurement - IFPUG functional size measurement method." Geneva.
10. ISO/DIS 26262 (2009), Road Vehicles - Functional Safety - Part 1-10, Standard Under Development.

11. ISO and IEC (2007), International Vocabulary of Metrology - Basic and GENERAL Concepts and Associated Terms (VIM), ISO/IEC Guide 99.

12. Lavazza L. and Morasca S. (2013), "Measuring the Functional Size of Real-Time and Embedded Software: a Comparison of Function Point Analysis and COSMIC." ICSEA 2013, The Eighth International Conference on Software Engineering Advances, Venice, Italy, pp. 465-470.

13. Lesterhuis A and Symons C (2014), "The COSMIC Measurement Manual, version 4.0", http://www.cosmicon.com/portal/

14. Lind K and Heldal R (2011), "A Model-Based and Automated Approach to Size Estimation of Embedded Software Components", ACM/IEEE The 14th International Conference on Model Driven Engineering Languages and Systems, Wellington, pp. 334-348.

15. Lind K and Heldal R (2009), "Estimation of Real-Time Software Code Size using COSMIC FSM", Proc. of the IEEE Intl. Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009), Tokyo, Japan, pp. 244-248.

16. Londeix B (2012), "A COSMIC Analysis of a Speedometer", An Automative Case Study, Telmaco, Alliare.

17. Marín B, Giachetti G. and Pastor O (2008), "Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures Based on COSMIC", IWSM/Metrikon/Mensura 2008: International Conferences on Software Process and Product Measurement, Springer-Verlag, Munich, Germany, pp. 170-183.

18. Prototype tool demonstration, http://www.youtube.com/watch?v=zsAZGHEzEOE

19. Renault S A (2012), "Design of a Functional Size Measurement Tool for Real-Time Embedded Software Requirements Expressed Using a Simulink Model", MathWorks Automotive Conference, Stuttgart.

20. RTCA Special Committee 167 (1992), "Software considerations in airborne systems and equipment certification", Technical Report DO-178B, RTCA.

21. Soubra H, Abran A and Ramdane-Cherif A (2012), "A Refined Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed Using the Simulink Model", Proceedings of the 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. IEEE Computer Society, Assisi, Italy, pp. 70-77.

22. Soubra H, Abran A, Stern S. and Ramdane-Cherif A (2011), "Design of a Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed using the Simulink Model." Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA). IEEE, Nara, Japan, pp. 76-85.

23. Soubra H and Chaaban K (2012), "Functional Size Measurement of Electronic

Control Units Software Designed Following the AUTOSAR Standard: A Measurement Guideline Based on the COSMIC ISO 19761 Standard", Proceedings of the 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. IEEE Computer Society, Assisi, Italy, pp. 78-84.

24. Soubra H and Ebert C (2013), "Towards a Method for Specifying and Estimating Functional Change in Real-Time Embedded Systems A Guideline Based on COSMIC ISO 19761 with an AUTOSAR Example", Metrikon, Kaiserslautern.

25. Soubra H (2013), "Fast Functional Size Measurement with Synchronous Languages: An Approach Based on LUSTRE and on the Cosmic ISO 19761 Standard", Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on. IEEE, Ankara, Turkey, pp. 3-8.

26. Talib M A., Khelifi A, Abran A and Ormandjieva O (2007), "A case study using the COSMIC-FFP Measurement Method for Assessing Real-Time System Specifications", Proceedings of the IWSM-Mensura, Palma de Mallorca, Spain.

27. Turetken O, Demirors O, Gencel C, Top O O and Ozkan B (2008), "The Effect of Entity Generalization on Software Functional Sizing: A Case Study", In Product-Focused Software Process Improvement, Springer Berlin Heidelberg, pp. 105-116.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com