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

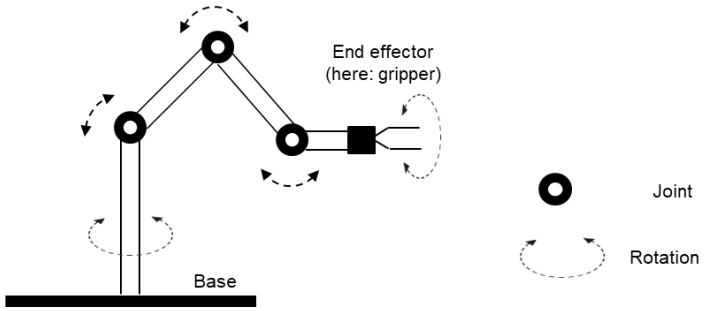

Figure 1.1 – An example robot arm

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):

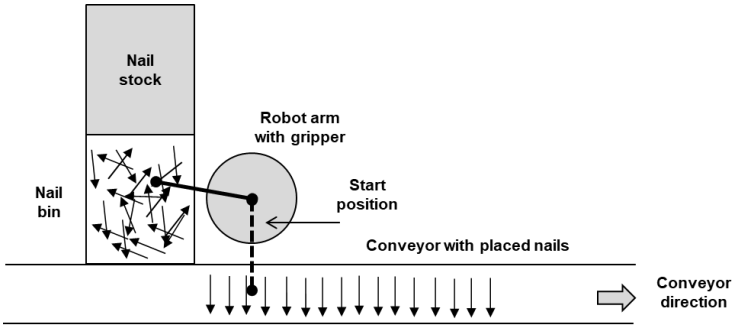

Figure 1.2 – The robot in its environment

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.

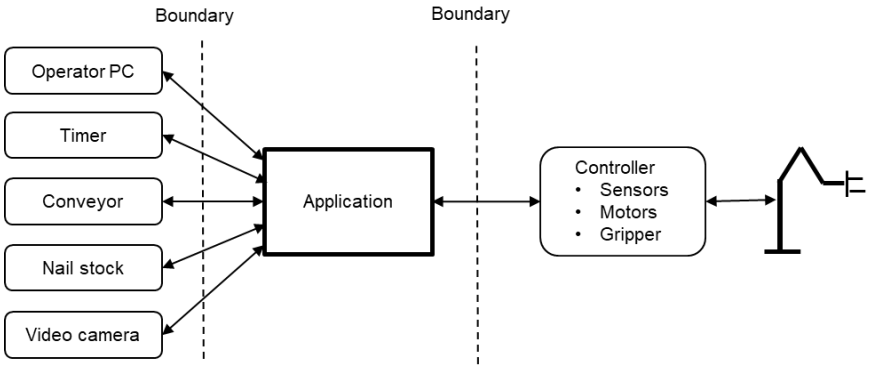

Figure 2.1 – Context diagram of the robot software

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

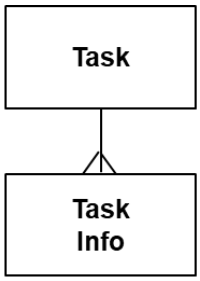

Figure 3.1 – Data model

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.

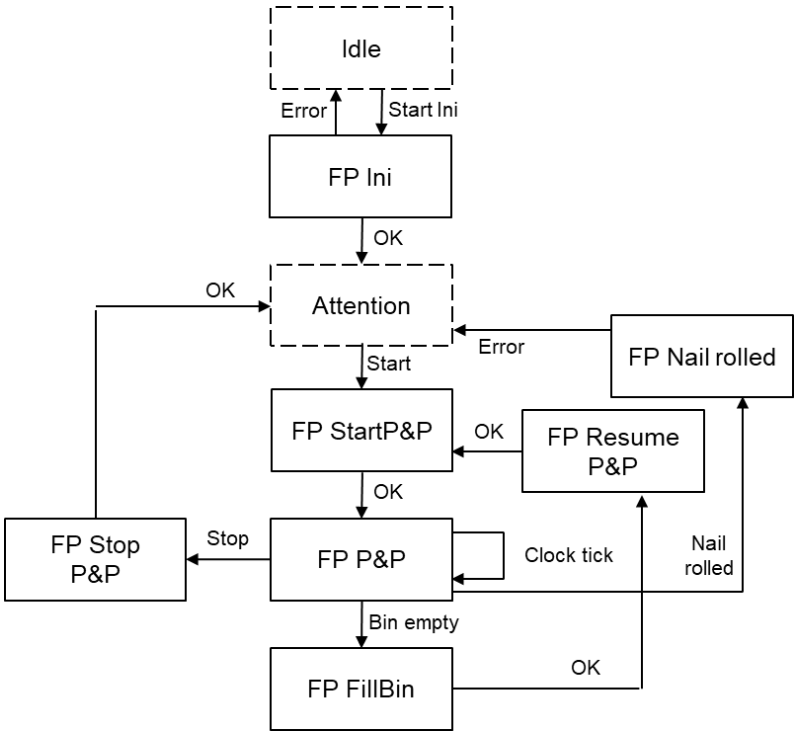

Figure 3.1 – The nail picking process as a finite state automaton

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.

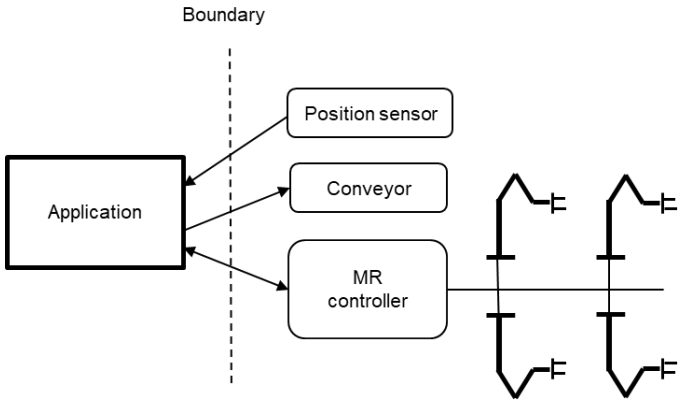

Figure 5.1 – Spot-weld car frame context diagram

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.

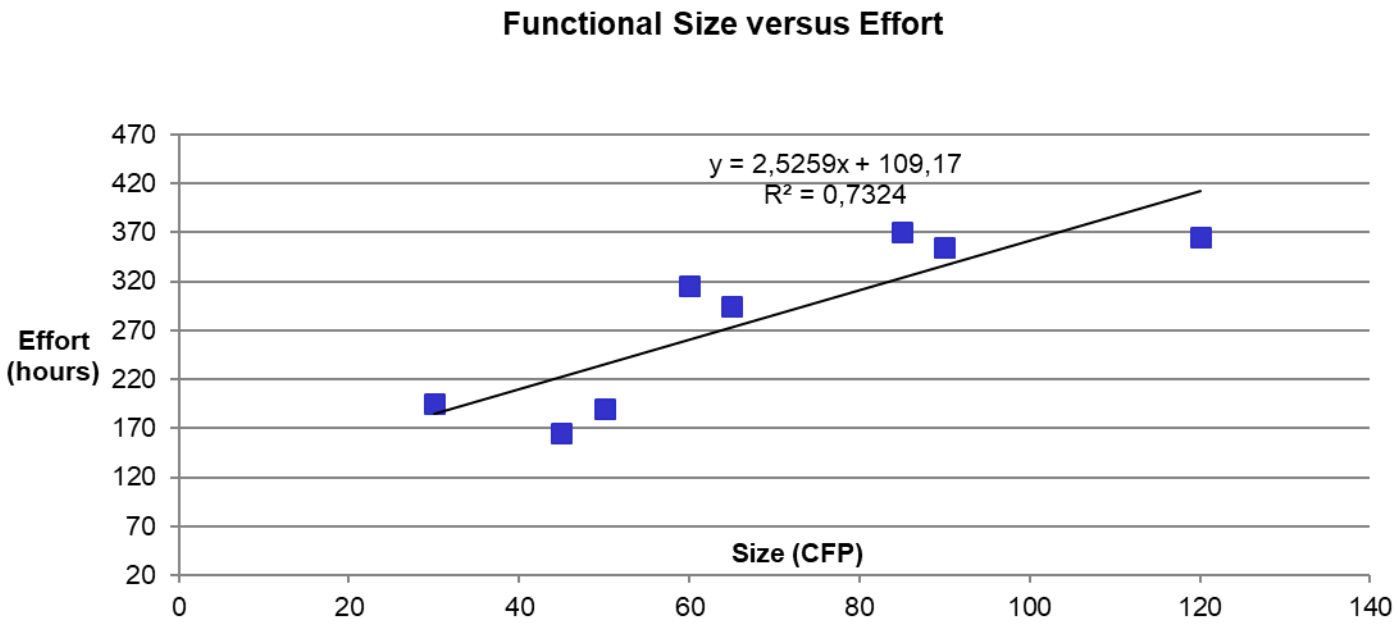

Figure B.1 – The estimation model

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.