Using UML for Learning How to Design and Model Cyber-Physical Systems

Leo Ordinez, Gabriel Eggly, Matías Micheletto, and Rodrigo Santos

***Abstract*— In this paper a methodology for teaching and learning the modeling of embedded systems and, in a more generic vision cyber-physical systems (CPS) is presented. To this end, a subset of tools from UML is used in an intuitive and ordered way starting with an informal description of the system until implementation details are obtained. However, the codification of the system is left out as the programming language depends on the hardware platform to be used. The method has been used in grade courses for several years now with an important accumulated experience that shows how students are able to adopt it and learn to elicit the different types of requirements, actors and functions.**

***Index Terms*— Cyberphysical systems, educations, requirements.**

I. Introduction

AT THE beginning of the 21^{st} Century, micro-controllers generalization and the development of the open hardware and software paradigms together with the increase in the wireless communication capabilities of the devices have created a new concept for the design and implementation of embedded systems that transformed them into CPS [1]–[3]. CPS definition is indeed an evolution of the embedded systems one. In fact, the most accepted definition for CPS is the integration of computing, communication and control systems with physical process. It is important to remark that the analysis, modeling, and implementation of this kind of systems is done considering all the aspects involved such as the signals to be sampled, the actuators, the information processing and obviously the physical process with which the system interacts. For this reason, the development includes different areas like software engineering, automatic control and communications.

CPS systems are important in different engineering areas and have impact in people's daily lives, even in certain situations where people are not aware of their presence. In fact, they are closely related to different domains such as Internet of Things (IoT), ambient intelligence (smart cities, smart transport, etc), collaborative systems and space and military applications [4].

In the curricula for Electronic Engineering and Computer Engineering careers, teachers still have problems to transmit the knowledge in an integral way. In general, professors feel more comfortable by splitting the reality in sealed compartments, thus facilitating the knowledge transfer in specific areas without considering the interactions with other aspects. Sometimes, during a lab project, a teacher may consider the interactions with different elements but these are not further deepened. While delivering a class on microprocessors architectures the main concerns are the assembly languages, registers sizes, buses, the interactions between registers and memory and the programming model. During a lab practice, the students may interface the microprocessor with some external device using a I/O device. The explanation will probably be restricted to the way in which the device is programmed and not on the physical aspects related to it. In the software engineering courses, different methodologies for solving problems using modeling techniques are taught to achieve dynamic programming skills. Both approaches, the microprocessor architecture course or the software engineering one, lack a global vision of the problem in such a way that all the actors involved, either software or hardware modules interacting with the environment or users can be analyzed together.

The open hardware and software platforms like Arduino or Raspberry Pi have simplified the implementation of CPS in the labs of undergraduate courses. However, as [5] explains, what may be considered an advantage is also a disadvantage as students work with an extremely simply platform that will not be available later for complex designs. For this reason, it is important to incorporate simple and effective modeling techniques that provide students with a good perspective of the system's design methodologies. This is important for the teaching-learning experience and contributes to their confidence at the moment of taking decisions for the implementation.

In this paper, a methodology for the teaching of modeling and design techniques for CPS, using a reduced set of tools of the Unified Modeling Language (UML) [6], is presented. The proposed methodology is used in courses of fourth and fifth year of the Electronic Engineering career at National University of the South (Bahía Blanca, Argentina), in particular, Digital Computers and Final Project courses. In both of them, the students have to model, design and implement a system at a prototype level that satisfies the requirements. The work is guided by teachers.

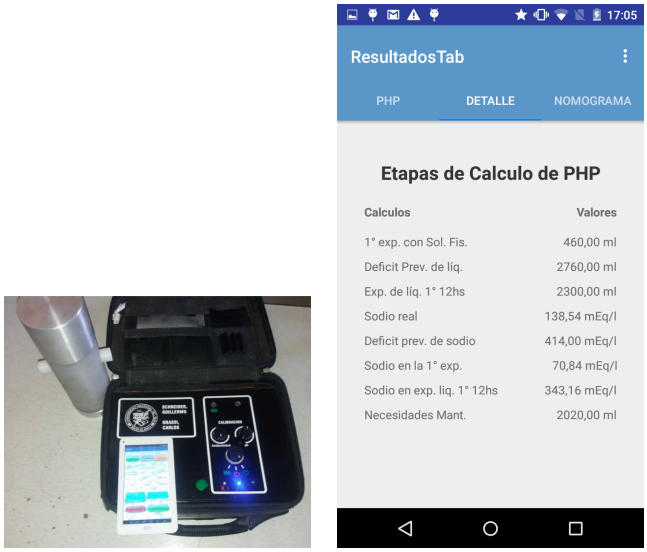
Manuscript received July 23, 2019; revised October 8, 2019 and November 4, 2019; accepted November 20, 2019. Date of publication March 4, 2020; date of current version April 6, 2020. (Spanish version received December 18, 2018; revised February 13, 2019; accepted May 21, 2019.) *(Corresponding author: Leo Ordinez.)*

Leo Ordinez is with the Laboratorio de Investigación en Informática (LINVI), Departamento de Informática, Universidad Nacional de la Patagonia San Juan Bosco (UNPSJB), Puerto Madryn 9120, Argentina (e-mail: leo.ordinez@gmail.com).

Gabriel Eggly, Matías Micheletto, and Rodrigo Santos are with the Departamento de Ingeniería Eléctrica y Computadoras, Instituto de Ciencias e Ingeniería de Computación, CONICET, Universidad Nacional del Sur (UNS), Bahia Blanca 8000, Argentina (e-mail: gmeggly@uns.edu.ar; matias.micheletto@uns.edu.ar; ierms@criba.edu.ar).

There exists a Spanish version of this article available at http://rita.det.uvigo.es/VAEPRITA/V8N1/A7.pdf

Digital Object Identifier 10.1109/RITA.2020.2978416



(a) (b)

Fig. 1. Examples of projects finished with this methodology. (a) Multiparametric sensor. (b) Android application for Diabetic Ketoacidosis.

As stated in [7], the non-functional characteristics of the CPS are not considered in the curricula. The author remarks in her study that even when it is clear that there are problems with the skills or competencies of engineers working with CPS, the actual courses barely focus on this subject. In the same line, the studies of [8] show that a combination of Model-Driven Engineering (MDE) and CPS can be taught in an effective way in the courses based on projects [9]. With respect to MDE the authors of [10] arrived to three main conclusions: first, the teaching of MDE can not reuse previous knowledge from code-centric approximations; second, there is lack of efficient tools; third, there is still missing a good course text-book. In this concern, in both courses (Digital Computers and Final Project), the solving of real problems is used as a pedagogical strategy for teaching. In this way, the students should implement a system that performs a real job learning with the technique of *hands on*. In particular, in this paper the teaching experience in the period 2013-2018 is presented in a systematic way for the Digital Computers course with 15 students in the average and for another 12 students in the period 2012-2018 that finished their Final Project. In all cases, the problems to be solved have a medium level complexity and represent real world situations. The medium level complexity expresses a sufficient level to be representative of the domain of application of the CPS and, at the same time, should be bounded, to meet the pedagogical objectives of the curricular contents. For example, in Figure 1a it is shown a multi parametric sensor to sample water quality in rivers, lakes or even the sea and in Figure 1b it is shown a screenshot of an Android application that is used to compute the compensation for dehydration in Diabetic Ketoacidosis, a condition that in pediatrics can have severe consequences if not properly treated.

This paper proposes a method based on requirements to develop small CPS. In it, the requirements are modeled in the first place. This is done through an adaptation of the use case diagrams of UML [6], in particular introducing new stereotypes associated to the relations. After this, the method performs the analysis, providing the CPS with a systematic fashion of implementing the use cases through activity diagrams, class diagrams and deployment diagrams. The method was thought to be simple, reinforcing the idea that it should be easy to use and aligned to the principles of *agile* programming [8].

The paper is oriented to systems with a low number of use cases, no more than thirty, since over that number the modeling of relationships among them is too complex. Although it is difficult to classify a computing system based on its size, we introduce an approximate classification. Since use cases are one of the main components of the proposed method, a metric based on them is used to determine the size of the system. The Use Case Points (UCP) [11], [12] metric is chosen. It uses two inputs that are the complexity of the actors measured according to the communication interface between them and the use cases and the complexity of the use cases measured in terms of the transactions associated to each one. In [13], the authors made a detailed description of how these transactions should be identified. Using the mentioned two inputs, the UCP metric establishes partial results from weighted sums of the inputs. The weights are related to technical and ambient aspects. The result of the sum is the Use Case Point (UCP), which is then multiplied by a productivity factor (PF) and this is the Estimated Time (ET) of the hours/man necessary to finish the project. With this, the systems are classified as small, medium or big. Using the UCP metric, a CPS is considered small or medium according to these parameters:

- No more than twenty actors.
- Most of the actors have low or medium complexity and only a few of them may have a high one.
- The number of use cases is limited to a few tens.
- The complexity of the use cases is low or medium with only a few of them with high one.
- The project can be carried out with a reduced team of developers of no more than four persons.

Contribution: A method based on UML is presented for the modeling, design and implementation of CPS that uses a reduced number of tools and allows a quick validation. With this method, students learn to determine in a practical and simple way the functional and non-functional requirements. They also learn to specify the relations between the actors and the mode in which they can be coded if it is software or implemented in the case of hardware. From the pedagogical point of view, the approach was proved to be suitable, since it promotes a global perspective of the problem and consequently accelerates the intellectual processes of students to understand and overcome the situation.

II. Previous Work

This paper presents a methodology for teaching the modeling, design and implementation of CPS used at National University of the South, Argentina. This method was developed as part of the doctoral thesis of Dr. Ordinez and presented in [14] and [15]. However, this paper extends the results presented there and explains the teaching methodology.

The following papers present design methods or requirements determination but are not oriented to the

teaching experience of them in the undergraduate courses. In [16], different techniques for modeling and design are discussed. The authors mention languages and tools and the main problems associated to the semantic of the systems and the communications. In [17], a bi-dimensional system for the quality of service in service oriented architectures and a real-time middleware are introduced. Although some points are similar to what is presented here, the paper is not oriented to teaching. In [18], the authors present a methodology for the elicitation of requirements. Like in the previous cases, it is not oriented to the learning experience. Brown [19] presents a software modeling and design system based on four aspects that respond to the four "C": Context, Containers, Components and Code. The method tries to simplify the usual tools like UML and SysML because of their complexity. However, this tool is not presented to be used in the teaching.

There are several papers about requirements determination in real-time embedded systems [20]–[22]. These papers introduce different variations in the definition of actors like the conceptualization of virtual actors and special use cases. Like in the previous papers, the authors explain the methodology or the tools proposed but are not oriented to the learning process. In [23], the authors propose to use UML for the co-design of hardware/software, this is somewhat similar to the methodology introduced in this paper.

The method here proposed is based on two of the main UML diagrams, thus it is possible to incorporate the extensions like the MARTE profile [24], [25] or even AADL [26], [27]. In [28], the authors proposed the use of contracts to reduce the gap between the control and software engineering. Although this looks similar to the methodology presented in this paper, the authors indicate that the contracts are useful for the implementation phase once the requirements and functions of the system have been defined and not in the modeling and design ones.

In SysML [29] there is a special diagram to depict requirements. However, it is limited to provide an artifact where the designer can detail in textual form the different requirements and link them with lines to functions. The method proposed in this paper can be adapted to use the SysML tools and vice versa, since the textual description proposed for the use cases can be incorporated in the requirement diagrams of SysML. In this way, the proposed methodology in relation to the requirements elicitation is more expressive and simple to visualize. Moreover, AADL, UML, SysML and MARTE have certain limitations [30], that include the lack of abstractions for certain components like the operating system in real-time and the absence of behavior models (AADL); difficulties to handle a large amount of diagrams (UML and MARTE); the restriction to the utilization in the systems engineering domain only (SysML) and the complex underlying metamodel (MARTE). If necessary, the proposed methodology in this paper can be complemented with any of the mentioned modeling methodologies.

Finally, some papers related to the teaching of CPS are mention that use different tools for modeling and design [7], [8], [31]–[33].

III. Use Cases Revisited

In the case of CPS there is a close relation among mechanic, hardware and software elements that interact with the physical environment. For this reason, it is necessary a global vision that integrates all the aspects involved and their interactions. The requirements can then be elicited from the purpose of the system, that is what is expected to be done. Additionally, requirements can be evaluated from the point of view of the environment where the system will work, that is how should it respond to certain situation. In a first approximation the input and output variables or signals of the system that are needed to achieve the desired functionality are determined. For this first stage, use cases are a simple tool that help the student to discriminate quickly how the system interacts with the physical environment. use cases have two stages: graphic and textual. The first one eases the visualization of the interactions among the involved elements in the different situations. The second one, allows a description of the interactions with details such as the computational aspects and the interfaces between the system and the real world.

A. Graphic Use Cases

In the UML standard [6], the use case diagrams have four constructs: system, actors, use cases and the communications links that express the associations among the constructors as can be seen in Figure 2. In the case of CPS, the relation among the physical environment and the computational world is very close and should be considered in a special way when modeling the requirements. In order to add this characteristic, a new "element" is added. It is represented by a dotted line that encloses the *system*. While in traditional use cases, only the computational elements are considered, in this case all the entities or elements that are part of the system are included. This element has no semantic or syntactic interpretation, it is only used to visualize the whole composition of the system to be modeled. This provides the students a broader understanding of the CPS by not discriminating the physical world (actors) from the computational one (use cases). In a similar way, the *requirements* are enclosed in another box that limit its range, making explicit the limit for the software operations. Some traditional UML elements are shown in the context of CPS.

1) **Actor:** In this element, an extension to the generally accepted concept of "actor" in the literature is introduced to incorporate the input/output (I/O) elements, naming all of them actors. An actor is an inner part of the system. It can be a role played by the system, a subsystem over which there is no direct control (operating system, network controller, communication device), but that provides functionality. It can also be a part of the physical process that the system is supervising or controlling. When looking at the actors from outside the system it should be impossible to determine whether they are within the computational or physical space. This vision is useful for the students to understand the problem with a *black box* approach [34].

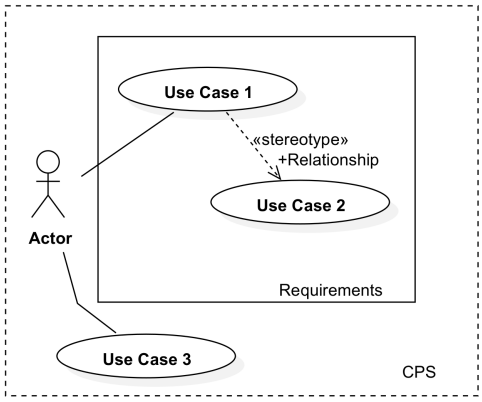


Fig. 2. Use case elements.

2) **Use case:** The functionalities of the system are defined by different use cases that represent each one a flow of actions or specific activities. Each use case represents a complex but identifiable activity in the system that can interact with others. At the same time, this activity can exchange information with the physical world. It should be noted that the use cases represent the purpose of the system and in this way the set of all them constitutes the complete computational description of the system. It is important to remark the differences among the use cases **Use Case 1** (or **Use Case 2**) with the **Use Case 3**. While the first ones are functionalities that can be implemented in software, the third represents a physical process. This visual distinction facilitates the students the identification of the processes involved and the way they are solved.
3) **Requirements:** This artifact is used to enclose the computational requirements of the CPS. All the characteristics that are inside are related to the embedded system: software, interfaces, etc.
4) **CPS:** Since CPS integrate the computational and physical processes, it is better to analyze both aspects simultaneously. Within this constructor all the related elements are included, the roles, devices and interfaces. This constructor defines the scope of the system.
5) **Association:** An association between an actor and a use case indicates that the first one provides or requires a functionality expressed in the use case. When the association is between use cases, it shows that some functionalities require others to be implemented, in this case it is a *direct* association. Another special type of association is *generalization*. This one has semantics similar to the inheritance concept in the object oriented programming. Besides, the associations can be qualified by a *stereotype* that describes the kind of relationship among the elements.
6) **Stereotype:** This is a mechanism that extends UML and can be used to adapt the model to the particular application domain [35]. The following stereotypes are special for the associations among use cases. In general, they determine the type of relation between two use cases through the purpose of it. That is, they capture the spirit or need behind the relation. It is important to notice that the stereotype is an association that is directed from one use case to another one.
   - ≪modeling≫ It represents the physical or natural laws around the system. In a generic way, this stereotype can be applied to all the relations that are not a computational or mechanical part of the system.
   - ≪communication≫ It indicates that the association is related to a communication aspect, for example a network service or the access to a serial port.
   - ≪call≫ Indicates the call to an auxiliary use case that performs a specific task.
   - ≪syscall≫ It is used to indicate the call to a service provided by a software of a higher hierarchy like an operating system or monitor.
   - ≪sync-syscall≫ It is used when the call to the operating system involves synchronization like in the case of access to shared memory, semaphores or mailboxes.

   The next stereotypes are referred to associations between use cases and actors. Although they identify the purpose of the association, they are not directed like the previous ones.
   - ≪requirer≫ The actor *requires* a functionality expressed in the use case. It describes the relation through a use case between two actors, one is the provider and the other the requirer. This is similar to the client/server or *request/response* concepts.
   - ≪provider≫ Like the previous one but on the other side of the relation.

   These two stereotypes are important because they help to define the client/server architecture by identifying the required services and the eventual providers.
   - ≪h-mi≫ This stereotype indicates that the use case associated represents an human-machine interaction functionality. It is common to any interface like keyboards, sounds, displays or touch screens among others.
   - ≪sensing≫ This stereotype expresses that the destination use case involves a sensing action to achieve its objective. This is one of the stereotypes used to represent the interaction between the physical and computational worlds.
   - ≪actuating≫ This stereotype indicates that the use case pointed involves the execution of an action through an actuator. It is used to represent the interaction with the physical world.
   - ≪analysis≫ This stereotype is used to represent data analysis and information processing within the computational world, there is no interaction with the environment.

B. Textual Use Cases

Graphical use cases allow a quick identification of the actors and the relations and functionalities of each element in the

TABLE I
USE CASE GENERAL TEMPLATE

	USE CASE NAME TO BE DESCRIBED
NAME	Use case name.
ACTORS	The actors associated directly to the use case, if there are not actors, the field is left empty.
DESCRIPTION	What does the use case do? In the case of using the stereotype <<analysis>> the procedures that are performed and the variables involved should be indicated; <<h-mi>> the description of the used interfaces; <<communication>> a clear indication of the communications implemented and the importance they have within the use case; besides the system calls that may be necessary, <<syscall>>, <<sync-call>>.
EXECUTION FLOW	Normal execution sequence.
EXCEPTION FLOW	Abnormal situations that require exceptional procedures
MONITORED VARIABLES	If the use case has associated an input of the type <<sensing>>, the monitored variables are described, if there is an stereotype <<analysis>>, the derived variables used for the processing should be explained. The description has to include range, precision, units and everything that may be significant.
CONTROLLED VARIABLES	Similar to the previous item but for the stereotype <<actuating>>
NON FUNCTIONAL REQ	This field refers to the restrictions that the system has. They can be of memory, response time, size, energy consumption, mechanic, footprint, weight, type of components, conditions of use (temperature, humidity, radiation, etc)

TABLE II
SYSTEM TEMPLATE

SYSTEM NAME	
OBJECTIVES	Identify the objectives, purposes or main functionalities of the system. Enumerate them with as much detail as possible and avoid ambiguities.
DESCRIPTION	Write the system's description as complete as possible, answering the question of what the system does.

TABLE III
ACTOR TEMPLATE

SYSTEM NAME	
ACTOR NAME	The name should be short and represent the actor.
PROVIDED FUNCTIONALITIES	Main characteristics provided by the actor to the system.
REQUESTED FUNCTIONALITIES	Main characteristics requested by the actor to the system.
DESCRIPTION	Describe the role the actor has within the system.

system, but they do not provide the necessary details to do a requirements analysis. The textual use cases are introduced to complement the description by incorporating the possibility of expressing different aspects like the execution flow and the steps involved in it, along with the size of the required memory, temporal restrictions, the precision of the sensors or energy demand among others. In [36]–[38] this is further discussed.

Table I shows a generic use case model adopted in the Digital Computers course at National University of the South. The fact of writing a text, although small and limited, promotes positive aspects in students such as the organization of information, the clarification of concepts, the synthesis and the need to use a precise and comprehensive vocabulary.

IV. REQUIREMENTS MODELING

In this section the different steps for the modeling are developed. It is a guide that allows the students to build the system model with all the requirements in an organized and simple fashion.

1) **Define the system:** The first step consists of a suitable definition of the system to be built. Usually, the descriptions are plenty of ambiguities that should be solved. For this it is necessary to write the purpose of the system clearly answering two basic questions: what the system does and how the system should do that. Table II shows a template for this.
2) **Identify the actors:** In this paper, actors are defined as external systems, external and internal entities with which actions and data are exchange. In this way, the physical environment is incorporated in the system definition as it is an actor with which there are direct and indirect interactions. Table III shows a template that the students can fill once they have identified the actors.
3) **Identify the use cases:** The functionalities of the system, that is the actions it performs either internally or externally, are represented by the use cases. In this way, for each functionality a specific use case is associated. The process to identify all the use cases is iterative and depends on the (*stakeholders*) of the system that are who in the end define the requirements. In the case of the students, this process is made interacting with the professors and assistant professors that help them to bound the complexity of the project. The names of the use cases should reflect the actors involved and the functionality they perform. In [22] a set of rules for the naming is presented and it is adapted here in the following way:
   For a use case associated with an Actor under the stereotype ≪requirer≫:
   "*The* <system' s name> *is required by* <actor's name> *for* <use case's name>."
   For a use case associated with an Actor under the stereotype ≪provider≫:
   "*The* <actor's name> *provides* to <system's name> *a functionality for* <use case's name>."
   For use cases that do not have a direct association with actors there are no explicit rules for their naming.
4) **Determine inputs (*monitored variables*) and outputs (*controlled variables*) for each use case:** In general,

the monitored variables are associated to sensors or external information provided to the use case. The controlled variables are usually associated to actuators or internal information that is shared with other use cases. In some cases, a variable can be monitored and controlled at the same time, for example the position of a motor's rotor. At this point, it is not necessary to completely describe the variables, it is enough to enumerate them. Later, in the textual description of the use cases all the details should be included.

5) **Solve the superposition of use cases:** Many times, two or more actors may require the same use case or several use cases may have overlapped functionalities. In both situations, the *refactoring* of use cases [39]–[41] is a suitable strategy promoted from different areas in Software Engineering. From a pedagogical point of view, the refactoring of the use cases eases the understanding of the problem by the students. They are forced to rethink the problem again, discuss and propose a new vision over the requirements.
6) **Write the textual use cases:** UML use cases are an agile method to describe the system requirements. However, they can not capture some important aspects that are necessary and should be included to complete them [42]. With the textual description proposed in section III, all those things that can not be represented in graphics can be incorporated. This involves non-functional requirements such as timing constraints, energy consumption and memory footprint.

The method is iterative until a satisfactory description is achieved.

V. Requirements-Based Software Design

Once the objectives are identified and the requirements are modeled, the next step is to describe the functionalities in a precise way. For this, the UML activity diagrams are used.

A. Activity Diagram: Basic Approach

The Activity Diagram is composed of several constructors and describes a basic logic unit of work [6]. It can be divided in *actions*. An action is the smallest or atomic work unit, which cannot be further divided. The sequence of actions is linked by the edges that represent the process flow or the successive events. Signals to make system calls can be transmitted or received. When there are auxiliary processes, it is proposed the use of *sub-activities* state.

B. Design Method

The process of describing use cases is divided in two parts. In the first one, a mapping between the situations that arise in the use case diagrams in relation to those in the activity diagrams is made. This is done by showing how the relations in the stereotypes are reflected in the activity diagrams. In the second part, a method to map the use cases into activity diagrams is presented. The result of this method is the description of how the use case achieves its objective.

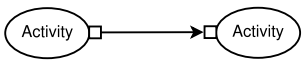


Fig. 3. Representation of the stereotype ≪communication≫ in an activity diagram.

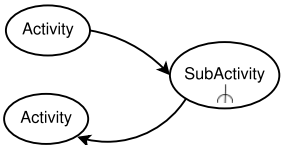


Fig. 4. The stereotype ≪call≫ represented in an activity diagram.

The Activity Diagram can be derived in a direct way from the textual description of the use case. In particular, the normal execution flow and the exception one represent the set of actions performed by the use case to achieve its purpose. As an intermediate step, between the use case and the activity diagrams, the sequence diagrams can be used [43]. They represent the interactions between the entities (actors, classes, etc), by passing messages. This strategy is recommended if the students have problems to translate the text into activities in a natural way. In this case, the intermediate step facilitates the understanding of the concepts involved. In what follows the stereotypes mentioned are described.

Figure 3 shows how the stereotype ≪communication≫ is mapped in an activity diagrams. As can be seen, a "flow" object is included as a path through which data is moved.

A common situation is a procedure or function call. Figure 4 shows a scheme to map the stereotype ≪call≫. It can be seen that there is a connection between two activity nodes.

The stereotypes ≪syscall≫ and ≪sync-syscall≫ are similar to the previous one, but the sub-activity is made by a different entity, thus it is executed in a different *swim lane*.

In this paper, the classification proposed in [22] is extended in relation to the use cases. In that paper, the author used the concept of *objects* but here it is extended to *tasks* that is more inclusive in the context of CPS. The *control tasks* are related to the use case with stereotypes ≪syscall≫, ≪sync-syscall≫, and those that provide functionalities to other use cases in sub-activities. The *interface tasks* are related to use cases that provide hardware services and communications with other use cases. Finally, the *entity tasks* group the use cases that perform the rest of the functionalities.

The textual use cases are revised and corrected, they are improved continuously until a plain ordered description without ambiguities of what is expected from the system is achieved. Based on the method proposed by Kimour and Meslati [44] the following steps are followed:

1) **Activity diagrams:** Each textual use case is transformed into an Activity Diagram. It is important to implement both the normal flow and the exception one. In this step it is necessary to verify if there are synchronization points and to establish in such a case the *guard* conditions.
   - If there are difficulties to do the transformation, deployment diagrams are useful as intermediate steps.

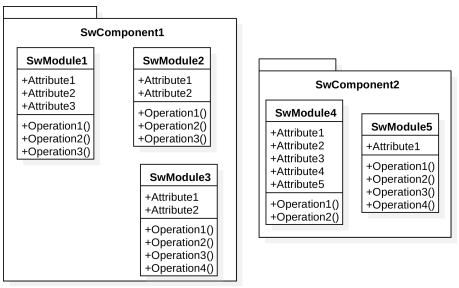


Fig. 5. Package and class diagrams.

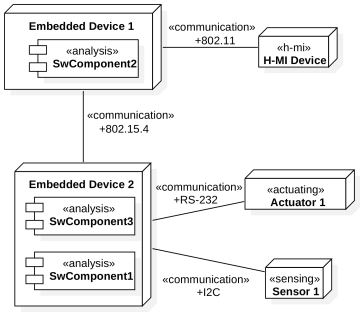


Fig. 6. Deployment diagram.

2) **Actions:** In this step the actions to be done are specified. It is necessary to consider both input and output variables involved and the conditions that enable them.
3) **Consistency:** In this step the consistency between the actions and their flow in the Activity Diagram is checked to guarantee that the Activity Diagram is feasible.

VI. Architectural Organization

Class and package diagrams from UML are used for describing the software architecture. They are used to showing functions and data associated to each software module. These aspects are common in the software engineering practice so no further comments are given here. Figure 5 presents a generic example.

The physical distribution of the software components in the hardware nodes is described through deployment diagrams that show how the stereotypes ≪communication≫, ≪h-mi≫, ≪sensing≫, ≪actuating≫ and ≪analysis≫ are actually implemented. In Figure 6, shows where the software *components* are placed in the physical *nodes*, along with the *communication protocols*, the sensors, the actuators and the human-machine interface.

VII. Example: Flow-Batch System

In the analytic chemistry it is common to see analyzers based on the Flow-Batch (FB) methodology [45], [46] for the determination of some components in different compounds. These are automatic systems that allow an improvement in the performance of the analytic process, allowing a higher frequency, precision and confidence in the studies. Moreover, they allow the manipulation of unstable, toxic, explosive or even radioactive substances, in a more secure way, as they reduce the volume of the compounds under analysis and the interaction that these may have with persons handling them in the process. In a FB system, the sample and the reactive are introduced in a mixing chamber in a sequential way or simultaneously by commanding a set of valves and peristaltic pumps that guarantee the repeatably of the experiment.

TABLE IV
Flow-Batch System Description

NAME	Flow-Batch system
OBJECTIVE	To handle a generic FB system for the analysis of chemical substances.
DESCRIPTION	Once the configuration of the system has been set by the user through the human-machine interface, the controller handles the components flows by commanding the peristaltic pump and the valves by issuing the adequate electric and electronic signals. It also collects the data sampled by the sensors.

There are different elements in a FB system. The propulsion unit (peristaltic pump with multiple pipes) is in charged of keeping constant the flow of solutions or reactives, and most important, to be able to repeat in exactly the same way the action. A multi switching system implemented with solenoid valves with three paths for a complete control of the fluids that should get into the mixing chamber. These fluids are injected into the mixing chamber by controlling the time and the rotation of the pump and in this way the volume injected is precisely determined. The system is controlled by an embedded system that through a human-machine interface (HMI) (keyboard and display) can be configured by the user. Table IV shows the template for the proposed system.

Once the system is defined, three actors are identified: the user, the embedded system in the role of the HMI, and the embedded system in the controller role. Figure 7 presents a simplify use case diagram.

In Tables V and VI two representative use cases are detailed. In Figure 8 a Deployment Diagram to show the HMI and its main functions is presented and Figure 9 shows the activity diagram for the procedure that executes the Flow-Batch process.

In Figure 10 a Class Diagram is presented for the valve controllers and the peristaltic pump. As they share both attributes, an Strategy Pattern is implemented to improve the design.

VIII. Experimental Evaluation

The proposed methodology was implemented in two undergraduate courses of the Electronic Engineering career at National University of the South in Argentina, and has been used for more than a hundred students. In the first of these courses, Digital Computers, the students get in touch for the first time with the basic microprocessor and microcontroller architectures. During the course, the students have to develop a system that is proposed by the teachers or by them spontaneously. These systems must have certain characteristics in order to guarantee that the students acquire the necessary skills to work with these devices, the hardware and software interfaces and the information processing associated to them.

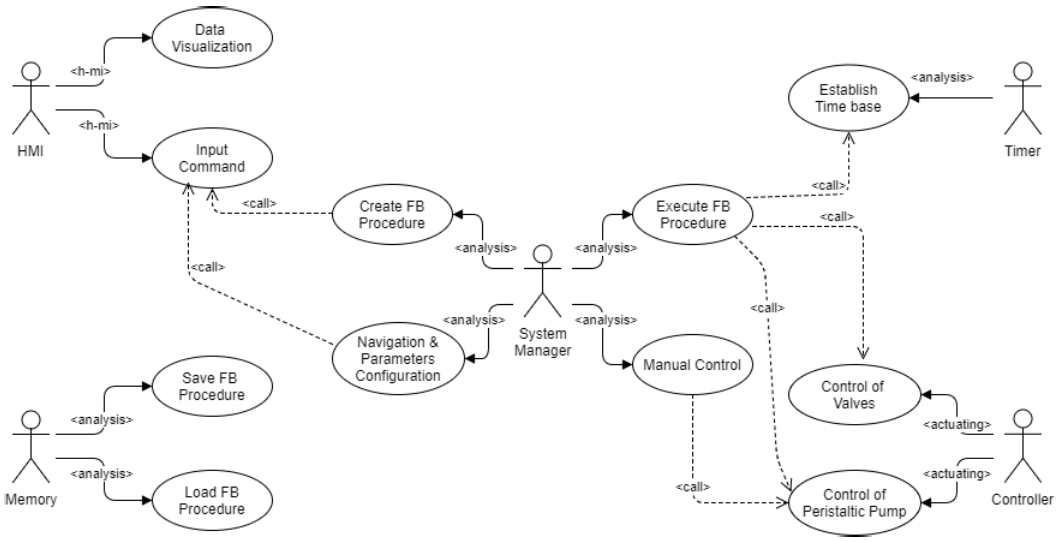


Fig. 7. Simplified use case diagram.

TABLE V
TEXTUAL USE CASE FOR THE INPUT COMMANDS

	Flow-Batch Controller
NAME	Input Commands
ACTORS	HMI and Flow-Batch handler
DESCRIPTION	The user sets the configuration by moving through the menus using the buttons in the front panel.
EXECUTION FLOW	1. Read the user's option 2. Identify the selected option 3. Execute the selected option.
EXCEPTION FLOW	2.1. The selected option is wrong 2.2 Return to 1.
MONITORED VARIABLES	Digital inputs from buttons K1, K2, K3, K4 and K5.

TABLE VI
TEXTUAL FLOW-BATCH USE CASE FOR THE EXECUTION OF THE PROCEDURE

	Flow-Batch controller
NAME	Execute flow-batch procedure.
ACTORS	FB System administrator, Controller, Timer.
DESCRIPTION	The controller executes the predefined tasks to obtain the requested FB sequence. Considering the amount of steps in the sequence and the time indicated for each one, the controller sets up the RPM for the pump and the opening of the corresponding valve. Once the sequence is finished, the system returns to the initial state awaiting for a new program.
EXECUTION FLOW	1. Adjust the outputs. 2. Turn-on the pump 3. Elapsed time 4. Choose next circuit 5. Return to 1.
EXCEPTION FLOW	3.1. Time is not elapsed 3.2 Return to 2. 4.1 Last circuit 4.2 Deactivate outputs.
CONTROLLED/MONITORED VARIABLES	The digital outputs that control the valves and the stepper motor (O1, O2, ...O8) and (STEP, M1, M2).

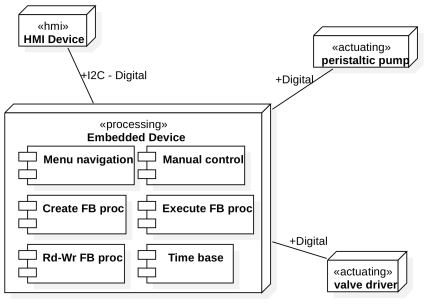


Fig. 8. Deployment diagram for the HMI.

The second course in which the method is applied is Final Project. In this case the students have to elaborate a thesis at the end of the career in which they show the knowledge acquired along the career. Any professor of the career can supervise this work, thus there are multiple areas to work. In the case of the Final Projects directed in the area of the Digital Systems Laboratory, they are oriented to the development of CPS or embedded systems for instrumentation or automation. Like in the other course, it is necessary to guide the students through the process of modeling as to elicit both functional and non-functional requirements in a systematic way until a satisfactory model is obtained. Some systems that have been modeled, designed and implemented through the years are: a portable multi parametric sensor to measure flow, pH, temperature, conductivity and turbidity for rivers, lakes or sea; a flow-batch controller for the automatizing of chemical experiments using the principles of the green chemistry; a spectral analyzer, temperature controllers for ceramic ovens, among others.

Before 2013, when the method was not taught to the students, the project in the Digital Computers course used to take about three out of fourth months. The students made the job in an intuitive way requiring several iterations until the objectives were achieved. As the teachers incorporated

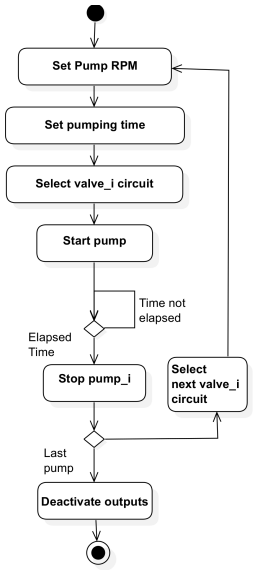


Fig. 9. Activity diagram for the procedure that executes the Flow-Batch process.

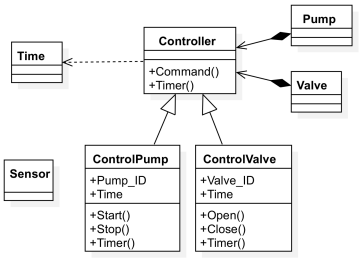


Fig. 10. Class diagram for the controller.

the proposed methodology the design adopted a systematic approach, the complexity of the system was increased and the time demanded to finish the project was reduced in one month. Nowadays, the students begin the project in the last days of April and present the operating prototype at the end of June. As the students have a methodology to follow, the number of iterations has been reduced significantly. Before the introduction of the method, about 20% of the students was not able to finish on time the project according to the requirements of the system. At the end of the course they just presented different operating parts but not integrated in a full system. For example, they showed that they were able to read an analog/digital converter, or change the state of an output variable but these functionalities were not linked into a system. This fact showed that the students were lacking the global vision of the system needed to modify the output value of a variable according to the reading of an input one. After the implementation of the methodology, only 5% of the students fails to present a working system. It is important to remark that the number of students that take the course (15) and the way in which they are divided in groups to do the projects (between two and four students) have not changed in the last six years.

The other course in which the method is implemented, Final Project, the students should define, model, design and implement a working system, and they should write and defend a report of their work to achieve the Electronic Engineering degree. Although it is a four months course, before the introduction of this methodology, the students used to take an average of nine months, that is more than the double of time assigned to the course. With the introduction of the method, the students finished it in most of the cases within the expected time of four months.

The method is taught without formally introducing the steps to be followed. In this way, it is expected that the students can internalize it naturally. The method is not presented as an algorithm to follow that students repeat over different problems to be finally evaluated. On the contrary, the method is presented as a set of ordered tools that they should use as needed while they advance in the definition of the system, the requirements elicitation and the implementation strategies. The identification and definition of the actors and the specification of the use cases are the steps that require more time because they oblige the students to conceptualize the system and its requirements. However, these steps are made using natural language avoiding the technicalities, consequently facilitating a more precise requirements elicitation. The deployment, activities and classes diagrams seem simple to the students, since they have procedural characteristics and bring them closer to coding, which is the last stage that we do not describe in this work.

IX. Conclusion

Electronic and Computer Engineering teaching is continuously under revision and update as there are advances in the associated technology every day turning mandatory the curricula actualization to keep the courses in line. In this paper, a methodology based on UML concepts to provide a simple, secure and agile mechanism for the modeling and design of CPS and embedded system was presented.

The curricula divides the learning into separated areas losing the global vision of problems. The processor architecture, their interfaces, programming models, control laws and communication protocols are taught as independent parts. The methodology here proposed presents a complete real problem to be solved starting with a description in natural language of the actors, their relations, requirements and functionalities until the implementation details with activity diagrams.

The proposed methodology was introduced in 2013 in different undergraduate courses with excellent results specially in reducing the time needed to define the system, its requirements and design avoiding unnecessary iterations and subsequent adjustments that were common. With the implementation, not only the performance of the students was improved but what is more important, the global interpretation of the problem.

In the next courses, tools from SysML will be incorporated to the methodology in order to evaluate if there are improvements in the learning process by using other type of diagrams.

References

[1] K. H. Kim, "Challenges and future directions of cyber-physical system software," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf.*, Jul. 2010, pp. 10–13.

[2] M. Broy and T. Stauner, "Requirements engineering for embedded systems," *Inf. Technol.*, vol. 41, no. 2, pp. 7–11, 1999.

[3] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Symp. Object Oriented Real-Time Distrib. Comput.* Washington, DC, USA: IEEE Computer Society, May 2008, pp. 363–369.

[4] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems," *Computer*, vol. 38, no. 11, pp. 23–31, Nov. 2005.

[5] J. C. Martinez-Santos, O. Acevedo-Patino, and S. H. Contreras-Ortiz, "Influence of arduino on the development of advanced microcontrollers courses," *IEEE Revista Iberoamericana Tecnologias Aprendizaje*, vol. 12, no. 4, pp. 208–217, Nov. 2017.

[6] (Oct. 20, 2014). *Unified Modeling Language*. [Online]. Available: http://www.uml.org

[7] E. Makio-Marusik, "Current trends in teaching cyber physical systems engineering: A literature review," in *Proc. IEEE 15th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2017, pp. 518–525.

[8] J. O. Ringert, B. Rumpe, C. Schulze, and A. Wortmann, "Teaching agile model-driven engineering for cyber-physical systems," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng., Softw. Eng. Edu. Training Track (ICSE-SEET)*, Piscataway, NJ, USA, May 2017, pp. 127–136, doi: 10.1109/ICSE-SEET.2017.16.

[9] L. Ordinez and O. Alimenti, "A constructivist approach for teaching embedded systems," *IEEE Latin Amer. Trans.*, vol. 11, no. 1, pp. 572–578, Feb. 2013.

[10] A. Hamou-Lhadj, A. Gherbi, and J. Nandigam, "The impact of the model-driven approach to software engineering on software engineering education," in *Proc. 6th Int. Conf. Inf. Technol., New Generat. (ITNG)* Washington, DC, USA: IEEE Computer Society, 2009, pp. 719–724, doi: 10.1109/ITNG.2009.160.

[11] S. Diev, "Use cases modeling and software estimation: Applying use case points," *ACM SIGSOFT Softw. Eng. Notes*, vol. 31, no. 6, pp. 1–4, Nov. 2006, doi: 10.1145/1218776.1218780.

[12] G. Karner, "Use case points: Resource estimation for objectory projects Objective Systems SF AB," Ph.D. dissertation, Rational Softw., San Jose, CA, USA, 1993.

[13] R. Collaris and E. Dekker, "Software cost estimation using use case points: Getting use case transactions straight," *IBM, Rational Edge*, 2009.

[14] L. Ordinez, D. Donari, R. M. Santos, and J. Orozco, "From user requirements to tasks descriptions in real-time systems," in *Proc. Workshop Engenharia Requisitos (WER)*, 2010.

[15] L. Ordinez, O. Alimenti, and L. Calles, "Eliciting requirements in small cyber-physical systems," in *Proc. CLEI 37th Conferencia Latinoamericana Inf.*, 2011.

[16] P. Derler, E. Lee, and A.-S. Vincentelli, "Modeling cyber physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, Jan. 2012.

[17] M. García-Valls, P. Basanta-Val, M. Marcos, and E. Estévez, "A bi-dimensional QoS model for SOA and real-time middleware," *Int. J. Comput. Syst. Sci. Eng.*, vol. 267, p. 6192, 2013.

[18] A. D. Toro and B. B. Jiménez, "Metodología para la elicitación de requisitos de sistemas software," Univ. Sevilla, Seville, Spain, Tech. Rep. LSI-2000-10, 2000.

[19] *The C4 Model for Software Architecture*. Accessed: Mar. 23, 2020. [Online]. Available: https://c4model.com/

[20] S. J. Goldsack and A. C. W. Finkelstein, "Requirements engineering for real-time systems," *Softw. Eng. J.*, vol. 6, no. 3, pp. 101–115, May 1991.

[21] M. S. Jaffe, N. G. Leveson, M. P. E. Heimdahl, and B. E. Melhart, "Software requirements analysis for real-time process-control systems," *IEEE Trans. Softw. Eng.*, vol. 17, no. 3, pp. 241–258, Mar. 1991.

[22] D. D. Zhang, "Use case modeling for real-time application," in *Proc. 4th Int. Workshop Object-Oriented Real-Time Dependable Syst. (WORDS)*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 56–64.

[23] N. Bolloju and S. X. Y. Sun, "Benefits of supplementing use case narratives with activity diagrams—An exploratory study," *J. Syst. Softw.*, vol. 85, no. 9, pp. 2182–2191, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016412121200129X

[24] (Oct. 20, 2014). *Modeling and Analysis of Real-Time and Embedded Systems*. [Online]. Available: http://www.omgmarte.org/

[25] L. Ordinez, D. Donari, R. Santos, and J. Orozco, "Time is not enough: Dealing with behavior in real-time systems," *J. UCS*, vol. 17, no. 11, pp. 1572–1604, Jul. 2011.

[26] F. Mallet, C. André, and J. DeAntoni, "Executing AADL models with UML/MARTE," in *Proc. 14th IEEE Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 371–376, doi: 10.1109/ICECCS.2009.10.

[27] S. Turki, E. Senn, and D. Blouin, "Mapping the MARTE UML profile to AADL," in *Proc. MoDELS 2010 ACES-MB Workshop*, 2010, pp. 11–20.

[28] P. Derler, E. A. Lee, S. Tripakis, and M. Törngren, "Cyber-physical system design contracts," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2013, pp. 109–118, doi: 10.1145/2502524.2502540.

[29] (Mar. 2019). *OMG Systems Modeling Language*. [Online]. Available: http://www.omgsysml.org/

[30] K. Evensen and K. Weiss, "A comparison and evaluation of real-time software systems modeling languages," in *Proc. AIAA Infotech@Aerospace*, Apr. 2010, p. 3504.

[31] J. Mäkiö, E. Mäkiö-Marusik, E. Yablochnikov, V. Arckhipov, and K. Kipriianov, "Teaching cyber physical systems engineering," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2017, pp. 3530–3535.

[32] J. I. Sosa *et al.*, "Industrial plant at academic level for teaching industrial informatics in an electronic engineering undergraduate degree," *IEEE Revista Iberoamericana Tecnologias Aprendizaje*, vol. 12, no. 1, pp. 1–9, Feb. 2017.

[33] H. Posadas and E. Villar, "Using professional resources for teaching embedded SW development," *IEEE Revista Iberoamericana Tecnologias Aprendizaje*, vol. 11, no. 4, pp. 248–255, Nov. 2016.

[34] R. Santos, L. Ordinez, and G. Eggly, "El enfoque de Cajas Negra y Blanca para la enseñanza de sistemas embebidos," in *Proc. IEEE Biennial Congr. Argentina (ARGENCON)*, Jun. 2016, pp. 1–7.

[35] D. Riesco, P. Martellotto, and G. Montejano, "Extension to UML using stereotypes," in *UML and the Unified Process*, L. Favre, Ed. Hershey, PA, USA: IGI Global, 2003, pp. 273–293. [Online]. Available: http://dl.acm.org/citation.cfm?id=953192.953207

[36] Usecases.org. *Matthew Tippett's Blog, Musings on Software Engineering, Management & Other Stuff*. Accessed: Mar. 23, 2020. [Online]. Available: https://use-cases.org/

[37] A. Cockburn, "Structuring use cases with goals," *J. Object-Oriented Program.*, vol. 10, no. 5, pp. 56–62, Sep./Dec. 1997.

[38] I. Jacobson and P.-W. Ng, *Aspect-Oriented Software Development With Use Cases*. Reading, MA, USA: Addison-Wesley, 2004.

[39] K. Rui and G. Butler, "Refactoring use case models: The metamodel," in *Proc. 26th Australas. Comput. Sci. Conf. (ACSC)*, vol. 16. Darlinghurst, NSW, Australia: Australian Computer Society, Inc., 2003, pp. 301–308. [Online]. Available: http://dl.acm.org/citation.cfm?id=783106.783140

[40] J. Xu, W. Yu, K. Rui, and G. Butler, "Use case refactoring: A tool and a case study," in *Proc. 11th Asia–Pacific Softw. Eng. Conf.*, Nov. 2004, pp. 484–491.

[41] C. Marcos, A. Rago, and J. A. Diaz Pace, "Improving use case specifications by means of refactoring," *IEEE Latin Amer. Trans.*, vol. 13, no. 4, pp. 1135–1140, Apr. 2015.

[42] M. A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. Gonzalez, "An empirical evaluation of requirement engineering techniques for collaborative systems," Univ. Castilla-La Mancha, Ciudad Real, Spain, Tech. Rep. DIAB-11-01-1, 2011.

[43] J. M. Almendros-Jimenez and L. Iribarne, "Describing use-case relationships with sequence diagrams," *Comput. J.*, vol. 50, no. 1, pp. 116–128, Oct. 2007, doi: 10.1093/comjnl/bxl053.

[44] M. T. Kimour and D. Meslati, "Deriving objects from use cases in real-time embedded systems," *Inf. Softw. Technol.*, vol. 47, no. 8, pp. 533–541, Jun. 2005.

[45] G. M. Eggly, M. Blackhall, A. de Araújo Gomes, R. Santos, M. C. U. de Araújo, and M. F. Pistonesi, "Emitter/receiver piezoelectric films coupled to flow-batch analyzer for acoustic determination of free glycerol in biodiesel without chemicals/external pretreatment," *Microchem. J.*, vol. 138, pp. 296–302, May 2018.

[46] G. M. Eggly, M. Nabaes, M. S. Di Nezio, M. E. Centurión, R. Santos, and M. F. Pistonesi, "Embedded flow-batch system with electrochemical detection for the determination of lead in propolis samples," *Int. J. Environ. Anal. Chem.*, vol. 97, no. 10, pp. 922–934, Aug. 2017.

Leo Ordinez received the degree in computer engineering and the Ph.D. degree in engineering from the Universidad Nacional del Sur. He is currently a Professor with the Universidad Nacional de la Patagonia San Juan Bosco. His research interests are oriented to the incorporation of technology for the improvement of life in cities, which could be named as smart cities. His work is focused on interdisciplinary projects covering areas such as software engineering, requirements engineering, urban planning, local development, and the Internet of Things.

Gabriel Eggly received the degree in electronic engineering and the Ph.D. degree from the Universidad Nacional del Sur, where he currently holds a Postdoctoral scholarship with CONICET. He is an Adjunct Professor with the Universidad Nacional del Sur. His research interests are in embedded systems and instrumentation oriented to metrology applied to the study and analysis of analytical chemistry and environmental variables.

Matías Micheletto received the degree in electronic engineering from the Universidad Nacional del Sur, where he currently holds a Ph.D. Scholarship with CONICET. His research interests are in the fields of embedded systems and scheduling optimization applied to precision farming, among others.

Rodrigo Santos is an Adjunct Professor with the Universidad Nacional del Sur and an Independent Researcher with CONICET, Argentina. He has been the President of the Center of Latin–American Studies in Informatics and the Vice-Chair of the IFIP WG 6.9 Communications for Developing Countries. His research interests are in the fields of real-time systems, embedded systems, and collaborative systems.