L. Ordinez, G. Eggly, M. Micheletto and R. Santos, "Using UML for Learning How to Design and Model Cyber-Physical Systems" in IEEE-RITA, vol. 15, no. 1, pp. 50-60, FEB. 2020. doi: 10.1109/RITA.2020.2978416

Utilizando UML para el aprendizaje del modelado y diseño de sistemas CPS

L. Ordinez, G. Eggly, M. Micheletto and R. Santos

***Resumen*—Este trabajo presenta una metodología para la enseñanza del modelado de sistemas empotrados o de manera más genérica CPS. Para esto se toma un subconjunto de herramientas de UML que se encadenan de manera intuitiva desde la descripción informal del sistema hasta los detalles de implementación, dejando la codificación final abierta de acuerdo al lenguaje de programación que se utilice. El método propuesto se utiliza desde hace varios años reuniendo experiencia sobre la adaptabilidad del mismo por parte de los alumnos y la facilidad para discriminar los distintos tipos de requerimientos, actores y funciones.**

***Abstract*—In this paper a methodology for teaching and learning the modelling of embedded systems and in a more generic vision cyber-physical systems 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 have 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 elicitate the different types of requirements, actors and functions.**

***Index Terms*—Sistemas CPS, educación, requerimientos.**

I. Introducción

En el comienzo del siglo XXI, la generalización de los microcontroladores y el desarrollo del paradigma de hardware y software abierto sumado a la cada vez mayor capacidad de comunicación inalámbrica de los dispositivos, generó un nuevo ámbito de desarrollo denominado sistemas CPS [1]–[3]. Este paradigma se plantea como una etapa superadora de aquel de los sistemas empotrados. La definición aceptada en este momento refiere a la integración de sistemas de computación, comunicación y control con procesos físicos. Es importante destacar que el análisis, estudio, modelado e implementación de esta clase de sistemas se realiza incluyendo todos los aspectos involucrados como las señales sensadas, los actuadores, el procesamiento de la información y obviamente el proceso físico con el que se interactúa. Esto involucra un proceso de desarrollo que atraviesa áreas de la Ingeniería de Software, el Control Automático y las Comunicaciones.

Los sistemas CPS (CPS) son de importancia en muchas áreas de la ingeniería e impactan sobre la vida cotidiana de las personas aún en espacios en los cuales no los pensaríamos como presentes. De hecho los mismos son parte de la Internet de las Cosas (IoT por su sigla en inglés) y contribuyen a la realización de ambientes inteligentes y cooperativos hasta renovadas aplicaciones militares y espaciales [4].

En la currícula universitaria de las carreras de Ingeniería Electrónica o Ingeniería de Computación, todavía los docentes encontramos dificultades para transmitir los conocimientos de manera integral. Por el contrario, en general nos sentimos cómodos separando saberes en compartimentos estancos facilitándonos así la transferencia de conocimientos en áreas específicas sin considerar las interacciones con otros aspectos. Eventualmente, en el desarrollo de un proyecto asociado a las prácticas de laboratorio, un docente puede incursionar someramente en algún espacio aledaño, sin que esto signifique profundizar en los mismos. Así al enseñar arquitecturas de microprocesadores nos centramos en el tipo de lenguajes, la orientación del mismo, el número de registros y su programación de bajo nivel. Si corresponde la realización de un práctico experimental de interfaz con el medio físico se explican los detalles de implementación y el software necesario para su funcionamiento. Desde las áreas de ingeniería de software por el otro lado se enseñan metodologías de resolución y abstracción de problemas con técnicas de modelado para lograr una dinámica de programación. Hay por lo tanto una falta de visión conjunta del problema en la cual los actores intervinientes pueden ser módulos de software, de hardware, de interacción con el medio o incluso humanos.

La aparición de plataformas de hardware y software abierto como Arduino o Raspberry PI simplificaron en gran medida las cuestiones relacionadas con la implementación de los sistemas CPS en los laboratorios de las materia de grado. Sin embargo, como sostiene [5], lo que puede ser una ventaja en un aspecto puede constituir una desventaja por la simplificación extrema que no responde a las exigencias reales de diseño. Por este motivo, la incorporación de técnicas de modelado simples pero efectivas generan una mejor perspectiva el diseño del sistema, lo cual beneficia los procesos de enseñanza-aprendizaje y da mayor confianza y seguridad a los alumnos al momento de tomar decisiones de implementación.

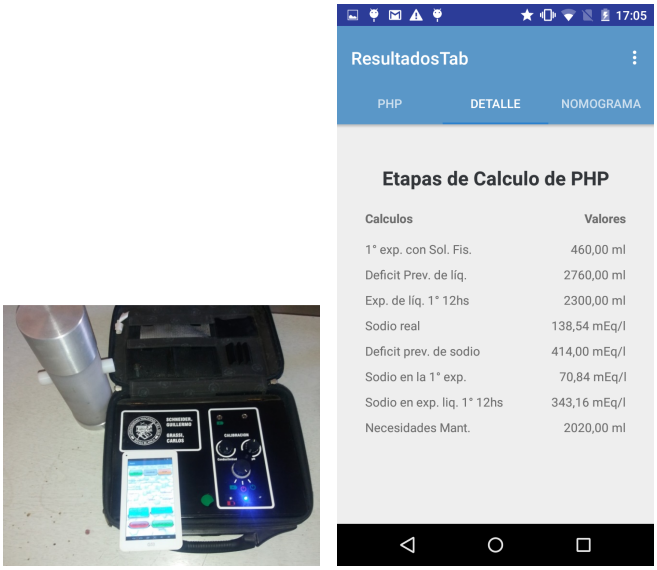
En este trabajo, se propone una metodología de enseñanza de los métodos de modelado y diseño de CPS, utilizando para ello un subconjunto de las herramientas provistas por el Lenguaje de Modelado Unificado (UML) [6]. La propuesta se instrumenta en cursos de cuarto y quinto año de Ingeniería Electrónica de la Universidad Nacional del Sur, Argentina, como son Computadoras Digitales y Proyecto. En las mismas, los alumnos deben modelar, diseñar e implementar un sistema

a nivel de prototipo que cumpla con los requerimientos planteados. El desarrollo del trabajo es guiado por los docentes de la cátedra.

Como se plantea en [7], las características no funcionales de los CPS apenas se abordan en los planes de estudio y cursos de CPS. La autora, en su estudio, ha revelado que, si bien existe una brecha en las competencias de los ingenieros de CPS relacionadas con las características no funcionales de CPS, los planes de estudio y cursos actuales de CPS apenas se centran en este tema. Asimismo, los estudios planteados en [8] muestran que una combinación de Ingeniería Basada en Modelos (MDE) y CPS se puede enseñar de manera efectiva en las clases centradas en proyecto [9]. En relación a MDE, los autores de [10] proporcionan tres conclusiones principales, que son que (1) la enseñanza de la MDE no puede reutilizar el conocimiento previo desde enfoques centrados en el código; (2) hay una falta de herramientas eficientes; y (3) todavía no hay un "buen libro de texto". En relación a esto, tanto en el curso Computadoras Digitales como en Proyecto, se instrumentan estrategias pedagógicas centradas en la resolución de problemas reales, mediante el desarrollo de proyectos. En particular, aquí se sistematiza la experiencia de trabajo a lo largo de los años 2013 a 2018 en el curso Computadoras Digitales con un promedio de 15 alumnos por curso. A la vez, para el caso de Proyecto la cantidad de proyectistas a lo largo de los años 2012 a 2018 fue de 12. En todos los casos, se busca que los proyectos tengan una complejidad media y que estén fuertemente ligados a aplicaciones del mundo real. El término *media* intuitivamente expresa un nivel suficiente para ser representativo del dominio de aplicación de los CPS y acotado, para cumplir los objetivos pedagógicos de los contenidos curriculares. Por ejemplo, en la Figura 1a se puede ver un monitor mutiparamétrico para muestrear espejos y cursos de agua, mientras que en la Figura 1b se muestra una captura de una aplicación para calcular la compensación por deshidratación en Cetoacidosis Diabética, una condición que en pediatría puede tener severas consecuencias si no es tratada adecuadamente.

Este artículo propone un método basado en requerimientos para desarrollar un CPS pequeño. En primer lugar, el método se centra en modelar los requerimientos. Esto se realiza a través de una adaptación de los Diagramas de Casos de Uso UML [6], en particular proponiendo nuevos estereotipos asociados a las relaciones. En segundo lugar, el método utiliza el análisis, proporcionando a los CPS una manera sistemática de implementar casos de uso mediante diagramas de Actividad, diagramas de Clases y diagramas de Despliegue. El método está diseñado para ser simple, enfatizando un buen nivel de facilidad de uso y en línea con los principios ágiles de desarrollo [8].

Este artículo, se enfoca en sistemas que tengan un bajo número de Casos de Uso, no más de 30, ya que superando esa cantidad el modelado de las relaciones entre los mismos se torna complejo. Aunque es difícil clasificar un sistema basado en computadora de acuerdo con su tamaño, aquí se esboza una clasificación aproximada. Dado que uno de los componentes principales del enfoque propuesto son los casos de uso UML, se utilizará una métrica basada en ellos para medir el tamaño



(a) (b)

Figura 1: Ejemplos de proyectos realizados.

del sistema. La métrica elegida es Use Case Points (UCP) [11] [12]. Se basa en dos entradas principales que son la complejidad de los actores, según la interfaz de comunicación entre los actores y los casos de uso; y la complejidad de los casos de uso medida en términos de las transacciones de casos de uso. En [13] los autores realizan una descripción detallada de cómo identificar las transacciones en los casos de uso. Sobre la base de esas entradas, la métrica UCP establece diferentes resultados parciales que son sumas ponderadas de las entradas, ajustadas por factores predeterminados. Los factores involucran aspectos tanto técnicos como ambientales. El último producto en la cadena de sumas es el punto de caso de uso (UCP), que luego se multiplica por un *Factor de productividad* (PF) para obtener un *Estimación total* (TE) de las horas requeridas para completar el proyecto. Con esto, los sistemas se pueden clasificar en simples, medios y complejos o grandes. Basado en la métrica UCP, un sistema CPS de tamaño pequeño a mediano es uno que se encuentra dentro de los siguientes parámetros:

- No hay más de un par de decenas de actores.
- La mayor parte de los actores tienen una complejidad baja o media y sólo algunos de ellos pueden tenerla alta.
- La cantidad de Casos de Uso se limita a unos pocas decenas.
- La complejidad de los Casos de Uso es media o baja con sólo algunos elevada.
- El proyecto se lleva a cabo con equipos reducidos de no más de cuatro personas.

Contribución: Se presenta un método basado en UML para el modelado, diseño e implementación de sistemas CPS que utiliza un número reducido de herramientas y permite una rápida validación. Con este método se enseña a los estudiantes a determinar de manera práctica y sencilla los requerimientos funcionales y no funcionales, especificar las relaciones entre los distintos actores y el modo en que pueden

ser codificados en el caso de software o implementados en el caso de elementos de hardware. Desde el punto de vista pedagógico, el enfoque ha mostrado ser apropiado, ya que favorece la obtención de una perspectiva global del problema y acelera los procesos madurativos de comprensión y solución de la problemática.

II. Trabajos Previos

Este trabajo presenta una metodología de enseñanza para el modelado, diseño e implementación de sistemas CPS utilizada en la Universidad Nacional del Sur, Argentina. El método propuesto fue desarrollado en primera instancia como parte de los trabajos de tesis doctoral del Dr. Ordinez y presentada como tal en [14] y [15]. Sin embargo en este trabajo se extienden los resultados presentados allí y se explica la metodología de enseñanza.

En general los trabajos mencionados a continuación refieren a métodos de diseño o determinación de requerimientos pero no abordan el problema de la enseñanza dentro de las materias de grado. Así en [16] se relevan diferentes técnicas de modelado y diseño. Los autores mencionan distintos lenguajes y herramientas y los principales problemas asociados a la semántica del sistema y las comunicaciones. En [17] se introduce un sistema bidimensional para calidad de servicio en arquitecturas orientadas a servicios y un middleware de tiempo real. Aunque en algunos puntos se podría asemejar al planteo que aquí se introduce, el trabajo no apunta a la enseñanza. En [18], los autores presentan una metodología de elicitación de requerimientos. Al igual que los casos anteriores no refieren a un método aplicado al aprendizaje. Brown [19] presenta un sistema de modelado y diseño de software basado en cuatro aspectos que responden a las 4 "C" que refieren a Contexto, Contenedores, Componentes y Código. El método intenta simplificar las herramientas habituales como UML o SySML que por su complejidad muchas veces son dejadas de lado. Sin embargo no es una herramienta que plantee su utilización en la enseñanza.

Hay varios trabajos referidos a la determinación de requerimientos de tiempo real en sistemas empotrados como [20]–[22]. En ellos se introducen diferentes variantes en la definición de los actores, llegando a la conceptualización de un actor virtual y la inclusión de Casos de Uso especiales. Nuevamente, los trabajos abordan la metodología pero no la enseñanza. El caso de [23] es interesante pues proponen el uso de UML para el co-diseño hardware/software que resulta similar a la propuesta que se realiza en este trabajo.

Dado que el método presentado aquí utiliza dos de los diagramas principales de UML, también puede tomar las extensiones propuestas para UML. En este caso, se puede incluir el perfil MARTE [24], [25] o incluso AADL [26], [27]. En [28], los autores proponen la realización de contratos que reduzcan el salto entre las disciplinas de control e ingeniería de software. A pesar de que pareciera cercano en la idea a lo propuesto en este trabajo, los autores aclaran que el método de contratos es útil una vez que todos los requerimientos y funciones han sido elaborados por lo que no aplica la metodología al diseño sino a la implementación.

En el caso de SysML [29], este lenguaje incluye un tipo de diagrama para los requisitos. Sin embargo, se limita a proporcionar un artefacto para detallar texto y líneas para vincular diferentes requisitos. En cualquier caso, el método propuesto aquí puede adaptarse a SysML y viceversa, ya que la plantilla de texto proporcionada puede incorporarse en los diagramas de requisitos de SysML. De esta manera, el enfoque propuesto, con respecto a los requisitos, es más expresivo y fácil de visualizar. Además, AADL, UML, SysML y MARTE tienen ciertas limitaciones [30], que incluyen la falta de abstracción de los componentes del sistema operativo en tiempo real y la ausencia de modelos de comportamiento (AADL); la dificultad en el tratamiento de un gran número de diagramas (UML y MARTE); la restricción al dominio de ingeniería de sistemas (SysML) y al metamodelo subyacente complejo (MARTE). En todos los casos, el enfoque propuesto aquí puede complementarse con cualquiera de las anteriores, bajo las circunstancias que se requiera.

Finalmente se mencionan algunos trabajos que se centran en la enseñanza de los sistemas CPS a partir de distintas herramientas de modelado y diseño [7], [8], [31]–[33].

III. Revisión de Casos de Uso

El caso de los CPS plantean una relación estrecha entre los elementos mecánicos, de hardware y de software que interactúan con el medio físico. Por este motivo es necesario un enfoque integrador que permita analizar las interacciones entre todos ellos. Los requerimientos entonces pueden ser analizados a partir del objetivo del sistema, es decir qué se espera que realice y desde un punto de vista del escenario en el cual se desenvolverá, cómo se espera que responda ante determinada situación. En un enfoque primario se definen entonces cuáles serán las variables de entrada al sistema y cuáles serán las de salida que permitan alcanzar el objetivo deseado en el ambiente determinado. Para esta primera etapa los Casos de Uso son una herramienta sencilla de implementar, que permiten al estudiante discernir rápidamente el modo en que debe interactuar el sistema con el medio físico para lograr el cometido. Los Casos de Uso tienen dos etapas: gráfica y textual. La primera facilita la visualización de las interacciones entre los diferentes elementos en las distintas situaciones planteadas. Mientras que la segunda, permite una descripción detallada de las mismas facilitando la inclusión de detalles poniendo énfasis en los aspectos computacionales y las interfaces entre el dispositivo y el mundo real.

III-A. Casos de Uso gráficos

En el estándar UML [6] se determina que los diagramas de Casos de Uso cuentan con cuatro constructores: sistema, actores, el Caso de Uso y las relaciones de comunicación que expresan las asociaciones entre los constructores, ver la Figura 2. En el caso de los CPS, la relación entre el medio físico y el mundo computacional es muy estrecha y debe ser considerada de manera especial al modelar los requerimientos. Para incorporar esta característica se propone un nuevo "constructor" que se representa por una línea punteada que encierra al *sistema*. Mientras que en los Casos de Uso tradicionales sólo

se contemplan los elementos computacionales, en este caso se incluyen todas las entidades o elementos que conforman el sistema. Este constructor no tiene una interpretación semántica o sintáctica, sino que simplemente sirve para visualizar la real composición del sistema a modelar. Esto facilita, en los estudiantes, la comprensión de un CPS al no discriminar el mundo físico (actores) del computacional (casos de uso). De modo similar, los *requerimientos* se encuadran en otro rectángulo que limita su alcance, demarcando así el límite explícito de las operaciones de software. En lo que sigue algunos constructores tradicionales de UML son replanteados en el contexto de los CPS.

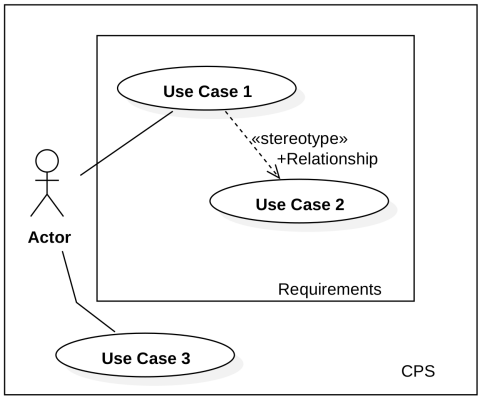


Figura 2: Elementos de los Casos de Uso

1. **Actor:** En este elemento, introducimos una extensión al concepto generalmente aceptado de "actor" en la bibliografía para incorporar a los componentes que interactúan con el exterior, llamando a todos ellos actores. Un actor es una parte interna del sistema. Puede ser un rol desarrollado por el sistema, un sub-sistema sobre el cual no se tiene control directo (sistema operativo, controlador de red, dispositivo de comunicación), pero que brinda funciones. También puede ser una parte del proceso físico que el sistema debe supervisar o controlar. Al mirar a los actores en función de su rol dentro del sistema, es imposible distinguir desde afuera si uno en particular se refiere al espacio computacional o físico. Esta visión facilita a los estudiantes una primera aproximación al problema, mediante el enfoque de *Caja Negra* [34].
2. **Caso de Uso:** Las funcionalidades del sistema se definen por diferentes Casos de Uso que representan cada uno un flujo de acciones o actividades específicas. Cada uno representa entonces una actividad identificable en el sistema que puede interactuar con otras. Al mismo tiempo, esta actividad puede intercambiar información con el mundo físico. A su vez los Casos de Uso representan el propósito del sistema y de esta manera el conjunto de todos ellos constituye la descripción computacional completa del sistema.
   Nótese la diferencia entre el caso de uso **Use Case 1** (o **Use Case 2**) y el **Use Case 3**. Mientras el primero es una funcionalidad implementable por software, el segundo representa un proceso físico. Esta distinción visual favorece en los alumnos la identificación de los procesos involucrados y sus abordajes.
3. **Requerimientos:** Este artefacto se utiliza para rodear los requerimientos computacionales de los sistemas CPS. Todo lo que se encuentra dentro se relaciona con la computación empotrada: software, interfaces, etc.
4. **CPS:** Dado que los sistemas CPS integran los procesos de computación con los físicos, es mejor analizar los dos aspectos simultáneamente. Dentro de este constructor todos los aspectos relacionados son incluidos, los roles, los dispositivos y las interfaces. Este constructor define el *alcance* del sistema
5. **Asociación:** Una asociación entre un actor y un Caso de Uso indica que el primero provee o requiere una funcionalidad expresada en el Caso de Uso. Respecto de los Casos de Uso, una asociación entre ellos muestra que hay funcionalidades que dependen de otras para poder implementarse, se dice que entre Casos de Uso, la asociación es *directa*. Un caso especial de asociación es la *generalización*. Este tipo de relación se permite únicamente entre actores y su semántica es similar al concepto de herencia en la programación orientada a objetos.

Además, las asociaciones pueden ser calificadas por un *estereotipo* que describe la clase de relación entre los constructores.

6. **Estereotipo:** Es un mecanismo que extiende a UML y puede ser utilizado para ajustar el modelo al dominio particular de la aplicación [35]. Los siguientes estereotipos son especiales en las asociaciones entre Casos de Uso. En general, determinan el tipo de relación entre dos Casos de Uso por medio del propósito de la misma. Es decir, que capturan el espíritu o la necesidad de la relación. Es importante notar que el estereotipo es una asociación que está direccionada de uno a otro Caso de Uso indicando la orientación de la misma.
   - `<<modeling>>` Representa a las leyes físicas o naturales que rodean al sistema. En forma genérica podemos indicar en este estereotipo a todas aquellas relaciones y elementos que no constituyen en sí la parte computacional y mecánica del sistema.
   - `<<communication>>` Indica que la asociación refiere a un aspecto de comunicación, por ejemplo un servicio de red o acceso a un puerto serie.
   - `<<call>>` Indica el llamado a un Caso de Uso auxiliar que realiza una tarea específica.
   - `<<syscall>>` Se utiliza para indicar la llamada a un servicio provisto por un software de otra jerarquía como un sistema operativo o monitor.
   - `<<sync-syscall>>` Se usa en el caso de llamadas al sistema que requieren de eventos de sincronización como el acceso a memoria compartida, los semáforos o los mailboxes.

   Los siguientes estereotipos se refieren a asociaciones entre Casos de Uso y actores. Aquí no requieren orientación en el establecimiento de la relación, aunque de

	NOMBRE DEL CASO DE USO A DESCRIBIR
NOMBRE	El nombre del Caso de Uso.
ACTORES	Los actores asociados directamente al Caso de Uso, sino hubiera, queda en blanco.
DESCRIPCIÓN	Qué hace el Caso de Uso. La respuesta debe ser lo más detallada posible. En el caso de utilizar el estereotipo <<analysis>> se deben indicar los procedimientos que realiza y las variables que involucra; <<h-mi>> la descripción de las interfaces usadas; <<communication>> una indicación claro de la clase de comunicaciones implementadas y la importancia que tienen dentro del Caso de Uso; además claro de las llamadas a sistema que pudiera requerir si fueran necesarias, <<syscall>>, <<sync-call>>, <<hw-call>>.
FLUJO DE EJECUCIÓN	Es la secuencia normal de ejecución.
FLUJO DE EXCEPCIÓN	Situaciones anormales que requieran de procedimientos excepcionales.
VARIABLES MONITOREADAS	En el caso de que el Caso de Uso tenga una asociación de entrada del tipo <<sensing>>, se describen cuales son las variables monitoreadas, si hubiera un estereotipo <<analysis>>, las variables derivadas para el procesamiento que son utilizadas de manera indirecta. La descripción debe incluir rangos, precisión, unidades y todo aquello que pueda ser considerado relevante.
VARIABLES CONTROLADAS	Similar al caso anterior pero sobre el estereotipo <<actuating>>.
REQ. NO FUNCIONALES	Refiere a las restricciones que tenga el sistema. Pueden ser de memoria, de temporizado, de condiciones de uso (temperatura, humedad), consumo de energía, tamaño, peso, mecánicas, tipos de componentes, entre otros.

Cuadro I: Plantilla Casos de Uso

todos modos identifican el propósito de la misma.

- <<requirer>> El actor *requiere* una funcionalidad expresada en el Caso de Uso. Describe la relación por medio de un Caso de Uso entre dos actores, uno que requiere y otro que provee. En cualquier caso es asimilable a los conceptos de cliente/servidor o *request/response*.
- <<provider>> Es el anverso del anterior.

Estos dos estereotipos son importantes ya que ayudan a definir la arquitectura, cuando correspondiera, de cliente/servidor por medio de la identificación de servicios provistos y requeridos.

- <<h-mi>> Indica que el Caso de Uso asociado representa una funcionalidad de interacción con una persona (interfaz hombre-máquina). Es indistinto del tipo de interfaz: teclado, pantalla táctil, sonido, etc.
- <<sensing>> Expresa que el Caso de Uso destino involucra una acción de sensado para lograr su objetivo. Es uno de los estereotipos que representan la interacción entre el mundo físico y el computacional.
- <<actuating>> Expresa que el Caso de Uso destino involucra la ejecución de una acción por medio de un actuador. Como en el estereotipo anterior, sirve para establecer la interacción con el medio físico.
- <<analysis>> Expresa la funcionalidad del Caso de Uso que refiere específicamente al procesamiento y análisis de los datos. No hay interacción directa con el medio.

III-B. Casos de Uso textuales

Los Casos de Uso gráficos permiten una rápida identificación de los actores y las relaciones y funcionalidades de cada elemento del sistema, pero no brindan los detalles necesarios para realizar un análisis de requerimientos. Para este fin se incorpora la descripción textual de los mismos en la cual se pueden agregar o expresar cuestiones como el tamaño de la memoria necesaria, restricciones temporales, precisión de los sensores o consumo de energía. En [36]–[38] se explicitan mejor estos aspectos.

El Cuadro I muestra un modelo de Caso de Uso que se adopta en el curso de Computadoras Digitales de la Universidad Nacional del Sur. El hecho de redactar un texto, aunque pequeño y acotado, promueve aspectos positivos en los estudiantes como la organización de la información, la claridad de conceptos, la síntesis y la necesidad de utilizar un vocabulario preciso y amplio.

IV. Modelado de Requerimientos

En esta sección se desarrollan los pasos a seguir para el modelado. Es una guía que en cada paso permite a los estudiantes construir un modelo del sistema con todos los requerimientos. Busca evitar el avance desorganizado y ordenar en cambio el proceso de manera sencilla.

1. **Definir el sistema** El primer paso consiste en una adecuada definición del sistema a realizar. Las descripciones suelen ser abiertas y plenas de ambigüedades que deben ser resueltas. Para ello hay que escribir claramente los propósitos del sistema, es decir qué debe hacer y una descripción que complete lo anterior con cómo lo debería hacer. En el Cuadro II se presenta una base que sirve para esto.
2. **Determinar los actores** En el contexto de este trabajo, los actores representan sistemas externos, entidades externas o internas con las que se intercambian acciones o datos. De este modo se incorpora al medio físico en la definición del sistema ya que se transforma en un actor con el cual se interactúa de manera tanto directa como indirecta. El Cuadro III muestra una plantilla que los estudiantes pueden completar una vez que han identificado a los actores.
3. **Determinar los Casos de Uso** Las funcionalidades del sistema, es decir, aquellas acciones que realiza tanto interna como externamente, se representan en los Casos de Uso. De este modo a cada funcionalidad se le asocia un Caso de Uso determinado. El proceso para llegar a la determinación de los Casos de Uso es iterativo y depende de los interesados (*stakeholders*) en

NOMBRE DEL SISTEMA	
OBJETIVOS	Identifique los objetivos, propósitos o principales funcionalidades del sistema. Enumérelas con el mayor detalle y evite ambigüedades.
DESCRIPCIÓN	Escribir la descripción del sistema de la manera más completa posible, respondiendo la pregunta de qué hace su sistema.

Cuadro II: Plantilla del Sistema

NOMBRE DEL SISTEMA	
NOMBRE DEL ACTOR	El nombre debe ser breve y representativo.
FUNCIONALIDADES PROVISTAS	Principales características ofrecidas por el actor al sistema.
FUNCIONALIDADES REQUERIDAS	Principales características requeridas por el actor al sistema.
DESCRIPCIÓN	Describir el rol que lleva adelante el actor dentro del sistema.

Cuadro III: Plantilla Actor

el sistema quienes son los que finalmente imponen los requerimientos del mismo. En el caso de los estudiantes este proceso se realiza en sucesivas entrevistas con la cátedra, que busca de este modo delimitar los alcances del proyecto que se va a desarrollar. Los nombres de los casos de uso deben reflejar a los actores involucrados y la funcionalidad que se desarrolla en el mismo. En [22] se presentan reglas para la nominación de los Casos de Uso que se adapta en este caso de la siguiente manera:
Para un Caso de Uso asociado a un actor con el estereotipo `<<requirer>>`:
"*El* `<nombre del sistema>` *es requerido por* `<nombre actor>` *para* `<nombre Caso de Uso>`."
Para Casos de Uso asociados a actores con el estereotipo `<<provider>>`:
"*El* `<nombre actor>` *ofrece* al `<nombre del sistema>` *una funcionalidad para* `<nombre Caso de Uso>`."
Para Casos de Uso que no tienen asociacón directa con actores no hay reglas explícitas para su nominación.

4. **Determinar entradas (*variables a monitorear*) y salidas (*variables controladas*) para cada Caso de Uso**
En general, las variables monitoreadas se asocian a sensores o información externa provista al Caso de Uso. Las variables controladas en cambio se asocian a actuadores o información interna. En algunos casos, una variable puede ser simultáneamente monitoreada y controlada (posición del rotor de un motor). En este punto, no es necesario describir completamente las variables, alcanza con enumerarlas. Más tarde cuando se realiza la descripción textual del Caso de Uso se deben incorporar todos los detalles correspondientes.
5. **Resolver los Casos de Uso superpuestos** En muchas ocasiones, dos o más actores pueden requerir el mismo Caso de Uso o varios Casos de Uso pueden contenerse parcialmente entre sí. En ambas situaciones, la estrategia de *refactorización* de los Casos de Uso [39]–[41], resulta una alternativa adecuada y promovida desde diferentes enfoques de Ingeniería de Software.
Desde el punto de vista pedagógico, la refactorización de los casos de uso favorece la comprensión del problema por parte de los estudiantes, quienes se ven forzados a repensar el problema, discutirlo y proponer una nueva mirada sobre los requerimientos.
6. **Escribir los Casos de Uso de manera textual** Los Casos de Uso de UML son un método ágil para describir requerimientos de un sistema. Sin embargo, no pueden capturar algunos aspectos importantes que deben incorporarse para completarlos [42]. Con la descripción textual propuesta en la sección III, se puede incorporar todo aquéllo que no puede ser representado en los gráficos como son los requerimientos no funcionales referidos a características temporales, consumo de energía o disponibilidad de memoria.

Los pasos mencionados deben repetirse iterativamente hasta alcanzar una descripción satisfactoria.

V. Diseño de Software basado en Requerimientos

Una vez que los objetivos fueron identificados y los requerimientos modelados, el siguiente paso es describir las funcionalidades de manera precisa. Para esto se usan los Diagramas de Actividades de UML. A fin de poder expresar correctamente todos los estereotipos planteados en este trabajo se realizan algunas consideraciones que se explican a continuación.

V-A. Diagrama de Actividades: aproximación básica

El Diagrama de Actividades se compone de diversos constructores y describe una unidad lógica de trabajo [6]. Se puede dividir en *acciones*. Una acción es la unidad de trabajo más pequeña o atómica ya que no puede ser descompuesta en otras. La secuencia de acciones es controlada por aristas direccionadas y condiciones de guarda que habilitan o no la transición marcada. Las acciones entonces son enlazadas por las aristas que representan el flujo del proceso o los eventos que se suceden. Se pueden transmitir y recibir señales para realizar llamadas al sistema. Cuando hay procesos auxiliares, se propone el uso de estados de *sub-actividades*.

V-B. Método de Diseño

El proceso de describir Casos de Uso se divide en dos partes. En la primera se realiza un mapeo entre las situaciones que surgen en los diagramas de los Casos de Uso en relación con sus correspondientes en los Diagramas de Actividades. Esto se realiza mostrando como las relaciones de los estereotipos se reflejan en los Diagramas de Actividades. En la segunda parte, un método para mapear los Casos de Uso en los Diagramas de Actividades se presenta. El resultado de este método es una descripción de cómo el Caso de Uso realiza su objetivo.

La descripción del Caso de Uso en la forma de un Diagrama de Actividades es directa si se consideran las representaciones

textuales de cada uno de ellos. En particular, el flujo normal de ejecución y el excepcional constituyen el conjunto de acciones que determina que el Caso de Uso alcanzó su objetivo o no. Como paso intermedio, entre los Casos de Uso y los Diagramas de Actividades, se pueden utilizar Diagramas de Secuencia [43], que representan interacciones entre entidades (actores, clases, etc.), mediante el pasaje de mensajes. Esta estrategia se recomienda aplicarla ante la dificultad de los alumnos de realizar el pasaje en forma natural. En este caso, se facilita el proceso de maduración de los conceptos, mediante la apoyatura de los Diagramas de Secuencia, como estadio intermedio. A continuación describimos los estereotipos mencionados.

La Figura 3 muestra como el estereotipo «communication» se mapea en un Diagrama de Actividades. Como se puede ver, se incluye un objeto "flujo" que es simplemente un camino por el cual se mueven datos.

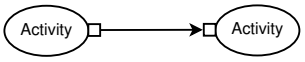


Figura 3: Representación del estereotipo «communication» en un Diagrama de Actividad

Una situación común en un software es la llamada a una función auxiliar. La Figura 4 muestra un esquema para mapear el esterotipo «call». Se puede ver que hay una conexión entre dos nodos de actividades.

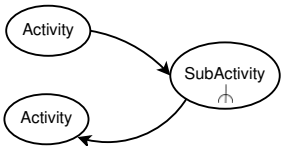


Figura 4: Estereotipo «call» representado en un Diagrama de Actividad

Los estereotipos «syscall» y «sync-syscall» son similares al anterior, pero la sub-actividad la realiza una entidad diferente por lo que se ejecuta en un carril distinto (*swimlane*).

En este trabajo extendemos la clasificación que realiza [22] en cuanto a los Casos de Uso. Si bien en el trabajo mencionado se utiliza el concepto de *objetos*, aquí se extiende al de *tareas* que resulta más abarcativo en el contexto de los sistemas CPS. Las *tareas de control* corresponden a los Casos de Uso con estereotipos «syscall», «sync-syscall», y a aquéllos que brindan funcionalidades a otros Casos de Uso en sub-actividades. Las *tareas de interfaz* se refieren a los Casos de Uso que brindan servicios de hardware y comunicación con otros Casos de Uso. Finalmente, las *tareas de entidades* reúnen a los Casos de Uso que desarrollan el resto de las funcionalidades.

El proceso por el cual se describen las tareas es iterativo. Los Casos de Uso textuales se revisan y corrigen, se aclara y mejora continuamente hasta lograr una descripción completa, sin ambigüedades y ordenada de lo que se pretende del sistema. Basado en el método propuesto por Kimour y Meslati [44] se siguen los siguientes pasos:

1. **Diagrama de Actividades:** Cada Caso de Uso textual se transforma en un Diagrama de Actividades. Es importante realizar tanto el flujo normal como el excepcional de acciones. En este paso hay que verificar si hay puntos de sincronización y establecer las condiciones de guarda necesarias.
   - En caso de dificultad para realizar la transformación, se deben utilizar Diagramas de Despliegue como pasos intermedios.
2. **Acciones:** En este paso se especifican las acciones a realizar. Se deben considerar las variables de entrada y salida involucradas y las condiciones que las habilitan.
3. **Consistencia:** En este paso se revisa la consistencia de las acciones y el flujo de las mismas en el Diagrama de Actividades a fin de que sea factible.

VI. Organización arquitectural

Para la descripción de la arquitectura del software se utilizan Diagramas de Clases y de Paquetes de UML, los cuales exponen los datos y funciones asociados a cada módulo de software. Estos aspectos son comunes a la práctica de Ingeniería de Software, por lo que no se profundizarán. En la Figura 5 se muestra un ejemplo genérico.

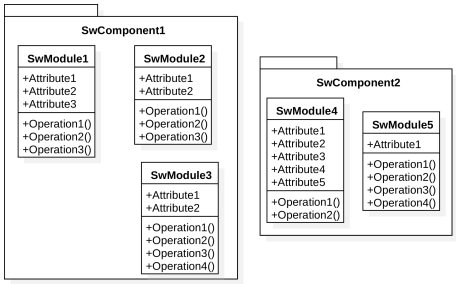


Figura 5: Diagrama de Clases y Paquetes

A la vez, para la descripción de la distribución física de los componentes de software en los nodos de hardware, se utilizan Diagramas de Despliegue, los cuales materializan en diseño los estereotipos de «communication», «h-mi», «sensing», «actuating» y «analysis». En la Figura 6 se muestran cómo se ubican en *nodos* físicos los *componentes* de software, así como los *protocolos de comunicación* y los dispositivos de sensado, actuación e interfaz hombre-máquina.

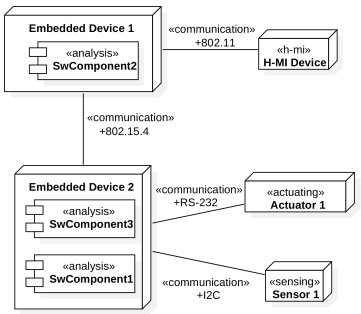


Figura 6: Diagrama de Despliegue

L. Ordinez, G. Eggly, M. Micheletto and R. Santos, "Using UML for Learning How to Design and Model Cyber-Physical Systems" in IEEE-RITA, vol. 15, no. 1, pp. 50-60, FEB. 2020. doi: 10.1109/RITA.2020.2978416

VII. Ejemplo sistema Flow-Batch

En la química analítica es común ver analizadores que utilizan la metodología Flow-Batch (FB) [45], [46] para la determinación de componentes en distintos compuestos. Éstos son sistemas automáticos que permiten una gran mejoría en el desempeño de los procesos, teniendo un aumento significativo en la frecuencia, precisión y exactitud de los análisis. Además, permiten la manipulación de sustancias inestables, tóxicas, explosivas y hasta radioactivas, reduciendo los costos para grandes volúmenes de análisis y control de los equipamientos con una mínima intervención de los operadores. En un sistema FB se introduce la muestra, el medio y el reactivo en una cámara de mezclado, de forma secuencial o bien simultáneamente.

Los elementos que componen un sistema FB son: la unidad de propulsión (bomba peristáltica con múltiples canales), que tiene como misión establecer un flujo de soluciones y/o reactivos constante y, sobre todo, perfectamente reproducible; y un sistema de multiconmutación implementado con válvulas solenoides de tres vías para un control total de aquellos fluidos que deben ingresar a la cámara de mezclado. Teniendo un control del tiempo y la rotación de la bomba, se puede calcular con precisión el volumen total inyectado. El proceso es controlado por un sistema embebido que por medio de una interfaz hombre-máquina (teclados y monitores) puede ser configurado por el usuario. En el Cuadro IV se muestra la plantilla del sistema propuesto.

Nombre	Sistema Flow-Batch
Objetivo	Administrar el funcionamiento general de un sistema FB genérico para el análisis de sustancias químicas.
Descripción	A partir de la configuración introducida por el usuario en la interfaz hombre-máquina, el controlador maneja el flujo de los componentes por medio de la bomba peristáltica, la apertura y cierre de válvulas, la aplicación de señales eléctricas y electrónicas y la captura en sensores de los resultados.

Cuadro IV: Plantilla del Sistema Flow-Batch

Una vez definido el sistema, se identifican los siguientes tres actores: el usuario, el sistema embebido en el rol de interfaz hombre-máquina (HMI), y el sistema embebido en el rol de controlador. En la Figura 7 se muestra un diagrama de casos de uso simplificado del sistema.

En los Cuadros V y VI se detallan dos casos de uso representativos.

En la Figura 8 se presenta como ejemplo un diagrama de despliegue para la interfaz hombre máquina con sus funciones principales.

La Figura 9 muestra el diagrama de actividades para el procedimiento que ejecuta la rutina Flow-Batch.

En la Figura 10 se presenta un ejemplo de Diagrama de Clases para el caso de los controladores de las válvulas y la bomba peristálticas. Al compartir ambos atributos, se implementa el Patrón Estrategia para mejorar el diseño.

VIII. Evaluación empírica

El método propuesto se presentó durante el cursado de dos materias de grado en la carrera de Ingeniería Electrónica de la Universidad Nacional del Sur, en Argentina, habiendo sido utilizado por más de 100 alumnos. En la primera de ellas, Computadoras Digitales, se introduce a los alumnos en las arquitecturas básicas de microprocesadores y microcontroladores. Para aprobar el cursado los estudiantes deben desarrollar un sistema a propuesta de la cátedra o por los mismos alumnos. Estos sistemas deben reunir ciertas características ya que en la realización de su implementación los estudiantes deben adquirir las competencias necesarias para trabajar con estos dispositivos, sus interfaces y el procesamiento de los datos asociados. La segunda materia sobre la que se aplica el método es el Proyecto Final de carrera. En este caso, se trata de una *tesina* que los alumnos desarrollan al terminar la carrera y en la cual deben integrar los conocimientos adquiridos en el transcurso de la misma. La materia es abierta en cuanto cualquier profesor puede dirigir este trabajo final y por ende hay diversas temáticas. En el caso de los Proyectos dirigidos en el ámbito del Laboratorio de Sistemas Digitales, se trata de desarrollar sistemas CPS o empotrados que tengan como fin la instrumentación o el control aplicado. Nuevamente aquí se hace imperioso un proceso de diseño que permita ir definiendo de un modo sistemático requerimientos funcionales y no funcionales hasta llegar al modelo final. Algunos de los sistemas modelados y diseñados fueron un sensor multiparamétrico portátil para espejos y cursos de agua que registra caudal, turbidez, pH, temperatura, conductividad; un controlador flow-batch para la automatización de ensayos químicos utilizando los principios de la química verde; un analizador de espectro; controladores de temperatura en hornos de cerámica, entre otros.

En los últimos seis años se enseñó esta metodología permitiendo a los alumnos adquirir destrezas en la definición de problemas, identificando actores, pautando requerimientos y obteniendo en forma ágil modelos convergentes en poco tiempo. Dentro de los proyectos abordados entre las dos materias se pueden mencionar el desarrollo de equipos de medición de potenciales eléctricos sobre micro muestras de productos apícolas que permiten la identificación de contaminantes como plomo, la utilización de sensores de ultrasonido para la identificación de glicerol en biodiesel, el filtrado por medio de imágenes de longitudes de onda particulares para la detección de nano partículas contaminantes en muestras de agua, el sensado de gases por medio de sensores especiales en biodigestores, el desarrollo de equipos de medición multiparamétricos integrados para mediciones *in situ* de cursos de agua, la construcción de registradores de esfuerzo y sensores de cobertura de suelos, entre otros.

Antes de la instrumentación del método propuesto el desarrollo del trabajo en la materia Computadoras Digitales demandaba tres de los cuatro meses de cursado. Los alumnos realizaban el trabajo de manera intuitiva requiriendo muchas iteraciones para alcanzar los objetivos planteados. A medida que los docentes instalamos la metodología de diseño en un modo más sistemático, los sistemas desarrollados se pudieron hacer más complejos y reducir además en un mes el tiempo necesario para llevarlos adelante. En la actualidad, los alumnos comienzan el planteo de requerimientos y definición del sistema a fines de abril para presentar el sistema funcionando en las últimas semanas de junio. Por otro lado, al disponer de

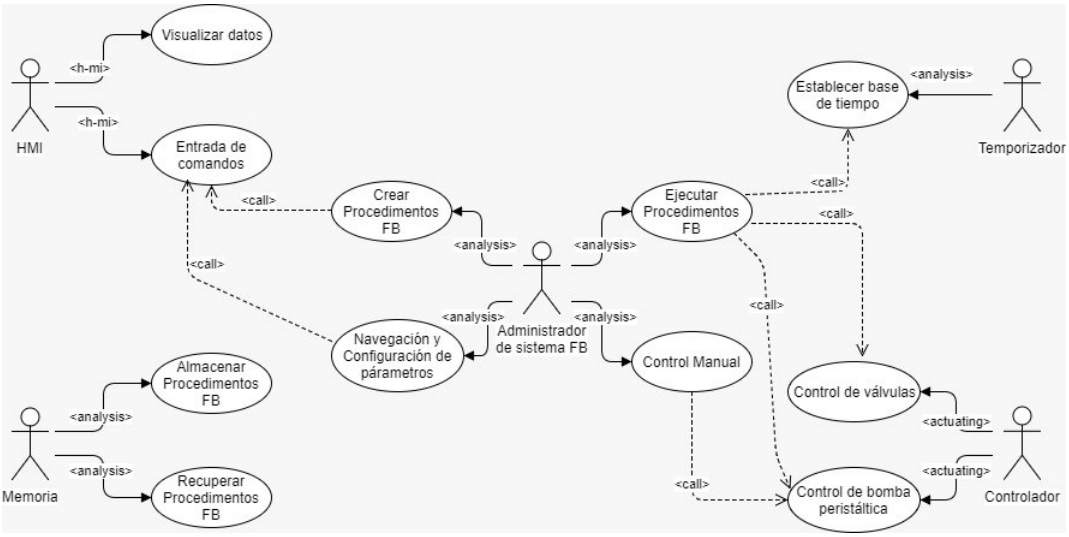


Figura 7: Diagrama de casos de uso simplificado

	Controlador Flow-Batch
NOMBRE	Entrada de comandos
ACTORES	HMI y Administrador Flow-Batch
DESCRIPCION	El Usuario a través de los botones del panel frontal puede navegar por los distintos menús como también configurar parámetros del sistema.
FLUJO DE EJECUCIÓN	1. Leer opción de usuario. 2. Identificar opción seleccionada. 3. Ejecutar la opción seleccionada
FLUJO DE EXCEPCIÓN	2.1. La opción seleccionada es incorrecta. 2.2. Volver a 1.
VARIABLES MONITOREADAS	Entradas digitales de los botones K1, K2, K3, K4 y K5.

Cuadro V: Caso de Uso Textual para Entrada de comandos

	Controlador Flow-Batch
NOMBRE	Ejecutar procedimiento flow-batch.
ACTORES	Administrador de Sistema FB, Controlador y Temporizador.
DESCRIPCION	El controlador ejecuta las tareas preestablecidas para lograr una secuencia flow-batch. En base a la cantidad de pasos de la secuencia y los tiempos de permanencia en cada uno, el controlador establece las RPM de la bomba y la salida activa en cada caso. Una vez finalizada la secuencia, el sistema vuelve al estado inicial a la espera de una nueva orden.
FLUJO DE EJECUCIÓN	1. Ajustar salidas. 2. Encender bomba. 3. Finaliza tiempo. 4. Siguiente circuito. 5. Volver a 1.
FLUJO DE EXCEPCIÓN	3.1. Tiempo no finaliza. 3.2. Volver a 2. 4.1 Último circuito. 4.2. Desactivar salidas.
VARIABLES CONTROLADAS / MONITOREADAS	Se controlan las salidas digitales para accionar las válvulas (O1, O2, ..., O8), así como también las salidas que controlan el motor paso a paso (STEP, M1, M2).

Cuadro VI: Caso de Uso Textual para Ejecutar procedimiento flow-batch.

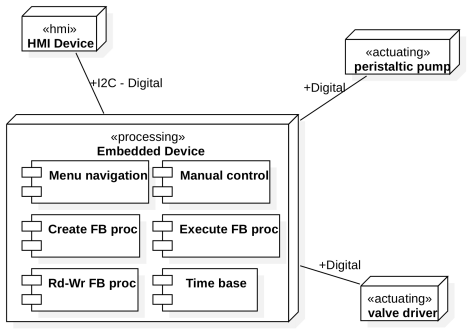


Figura 8: Diagrama de despliegue para la interfaz hombre máquina

un método de trabajo, las iteraciones también se redujeron significativamente. Es importante notar, que previo al año 2013, muchos alumnos no lograban terminar el trabajo de acuerdo a los requerimientos del sistema planteado. Era común que sólo presentaran las diversas partes del mismo pero sin integración. De esta forma mostraban que podían realizar la lectura de un conversor analógico-digital o producir un cambio de estado en una variable. Sin embargo, no podían enlazar los diferentes procesos mostrando una carencia en la visión integral del sistema que requería por ejemplo de acuerdo al valor obtenido de la lectura analógica realizar una acción u otra sobre un actuador. El 20 % de los alumnos llegaba al final del curso en estas condiciones habiéndose reducido en la actualidad a menos del 5 %, lo que representa un avance. Vale destacar que tanto el número promedio de alumnos de la materia Computadoras Digitales, (el cual es 15), como la conformación de los grupos de trabajos (de entre dos y cuatro estudiantes), se ha mantenido inalterado a lo largo de los seis años de implementación del método.

En la otra materia donde se aplica el método propuesto, Proyecto Final de Carrera, es una tesina de grado para acceder

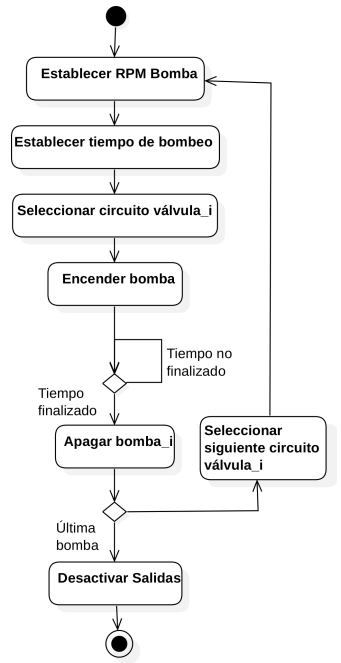


Figura 9: Diagrama actividades procedimiento del proceso que ejecuta el control Flow-Batch

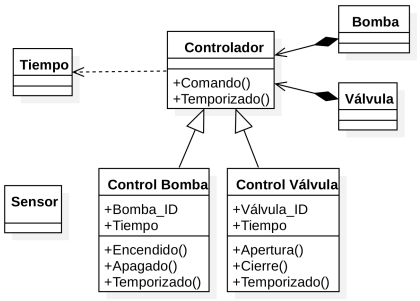


Figura 10: Diagrama de Clase del sistema.

al título de Ingeniero Electrónico. Este trabajo final tiene varias etapas de desarrollo. En la primera se plantean los objetivos del mismo y se planean las actividades a realizar con los tiempos estimados para cada etapa. Luego se desarrolla el trabajo y al término del mismo se debe presentar una memoria técnica descriptiva de lo realizado que debe ser defendida ante un tribunal que evalúa tanto el trabajo realizado, como la memoria escrita y la presentación oral. En este caso los alumnos se enfrentan por primera vez en el marco de la carrera al desafío de plantear un problema, sus posibles soluciones técnicas, su realización, presentación de informe y defensa. Los pasos de evaluación de soluciones técnica y realización son en los cuales la metodología propuesta es aplicada. Aunque la materia debe concluirse en el plazo de un cuatrimestre, era usual que los estudiantes demoraran mucho tiempo en la definición de los requerimientos y en la instrumentación de soluciones ya que no contaban con un método sistemático de trabajo. Esto significaba retrasos considerables que en algunos casos llegaba a duplicar y más el tiempo establecido siendo el promedio de duración cercano a los nueve meses. Con la incorporación de esta propuesta, los alumnos redujeron significativamente el tiempo de cursado de esta materia logrando realizarla en los tiempos previstos.

La enseñanza del método se realiza de manera intuitiva sin introducir formalmente los pasos a seguir. De este modo, se pretende que los alumnos internalicen el procedimiento de modo natural. La utilización del método no se plantea como un algoritmo que repiten sobre diferentes problemas para luego ser evaluados en los pasos que siguen. Por el contrario, el método se presenta como un conjunto de herramientas ordenadas de las que van haciendo uso a medida que avanzan en la definición del problema, los requerimientos y las estrategias de implementación. La definición de los actores y la especificación de los Casos de Uso son las partes que más tiempo demoran porque obligan a la conceptualización del sistema y sus requerimientos. Sin embargo esta etapa se realiza con lenguaje natural evitando el uso de tecnicismos lo que permite una elicitación de requerimientos más precisa. Los diagramas de despliegue, actividades y clases les resultan más sencillos ya que tienen características procedimentales y los acerca a la codificación, que es la última etapa que no describimos en este trabajo.

IX. Conclusiones

La enseñanza en ingeniería electrónica y de computación como quizás en ningunas otras áreas está sujeta a continua revisión dado que los avances continuos en la tecnología obligan a una actualización permanente. En este trabajo, se presentó una metodología basada en conceptos de UML para brindar un método concreto, sencillo, ágil y seguro para el modelado y diseño de sistemas CPS o empotrados.

La currícula actual enseña en forma compartimentada por un lado las arquitecturas de los procesadores, sus interfaces, los modelos de programación, las leyes de control y los mecanismos de comunicación, perdiendo de vista la integración que existe en la actualidad entre los diferentes campos. La propuesta entonces presenta desde la descripción prácticamente en lenguaje natural de los actores, sus relaciones, requerimientos y funcionalidades hasta los detalles de implementación por medio de Diagrama de Actividades.

La propuesta se puso en práctica en diversos cursos de grado y el resultado fue exitoso en cuanto al tiempo invertido en la etapa de definición del sistema, sus requerimientos y diseño inicial evitó muchas iteraciones y ajustes que se realizaban antes de adoptarlo. Con esto la eficiencia mejoró pero también, y más importante aún, la interpretación global del problema.

En los próximos cursos introduciremos en el proceso herramientas de Sysml a fin de evaluar si mejora aún más el proceso de aprendizaje de modelado y definición de requerimientos.

Referencias

[1] K. Kim, “Challenges and future directions of cyber-physical system software,” in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, pp. 10 –13, july 2010.

[2] M. Broy, “Requirements engineering for embedded systems.”

[3] E. A. Lee, “Cyber physical systems: Design challenges,” in *Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing*, (Washington, DC, USA), pp. 363–369, IEEE Computer Society, 2008.

[4] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems," *Computer*, vol. 38, pp. 23–31, November 2005.

[5] J. C. Martínez-Santos, O. Acevedo-Patino, and S. H. Contreras-Ortiz, "Influence of arduino on the development of advanced microcontrollers courses," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 12, pp. 208–217, Nov 2017.

[6] "Unified modeling language." http://www.uml.org, Last visited 20th October 2014.

[7] E. Makio-Marusik, "Current trends in teaching cyber physical systems engineering: A literature review," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pp. 518–525, July 2017.

[8] J. O. Ringert, B. Rumpe, C. Schulze, and A. Wortmann, "Teaching agile model-driven engineering for cyber-physical systems," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*, ICSE-SEET '17, (Piscataway, NJ, USA), pp. 127–136, IEEE Press, 2017.

[9] L. Ordinez and O. Alimenti, "A constructivist approach for teaching embedded systems," *IEEE Latin America Transactions*, vol. 11, 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 *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, ITNG '09, (Washington, DC, USA), pp. 719–724, IEEE Computer Society, 2009.

[11] S. Diev, "Use cases modeling and software estimation: Applying use case points," *SIGSOFT Softw. Eng. Notes*, vol. 31, pp. 1–4, Nov. 2006.

[12] K. G., *Use Case Points - Resource Estimation for Objectory Projects, Objective Systems SF AB*. PhD thesis, copyright owned by Rational Software, 1993.

[13] R. Collaris and E. Dekker, "Software cost estimation using use case points: Getting use case transactions straight," tech. rep., IBM, The 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 *Proceedings of WER'10 - Workshop em Engenharia de Requisitos*, 2010.

[15] L. Ordinez, O. Alimenti, and L. Calles, "Eliciting requirements in small cyber-physical systems," in *Proceedings of CLEI'11 - XXXVII Conferencia Latinoamericana de Informatica*, 2011.

[16] P. Derler, E. Lee, and A.-S. Vincentelli, "Modeling cyber physical systems," *Proceedings of the IEEE*, vol. 100, pp. 13–28, Jan 2012.

[18] B. J. B. Durán Toro, Daniel, "Metodología para la elicitación de requisitos de sistemas software," Tech. Rep. LSI–2000–10, Universidad de Sevilla, 2000.

[19] "The c4 model for software architecture." https://c4model.com/. https://c4model.com/.

[20] S. Goldsack and A. Finkelstein, "Requirements engineering for real-time systems," *Software Engineering Journal*, vol. 6, 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, 1991.

[22] D. D. Zhang, "Use case modeling for real-time application," in *WORDS '99: Proceedings of the Fourth International Workshop on Object-Oriented Real-Time Dependable Systems*, (Washington, DC, USA), IEEE Computer Society, 1999.

[23] N. Bolloju and S. X. Sun, "Benefits of supplementing use case narratives with activity diagrams—an exploratory study," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2182 – 2191, 2012. Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).

[24] "Modeling and analysis of real-time and embedded systems." http://www.omgmarte.org/ Last visited 20th October 2014.

[25] L. Ordinez, D. Donari, R. Santos, and J. Orozco, "Time is not enough: Dealing with behavior in real-time systems," *jucs*, vol. 17, pp. 1572–1604, jul 2011.

[27] S. Turki, E. Senn, and D. Blouin, "Mapping the marte uml profile to aadl," in *MoDELS 2010 ACES-MB Workshop Proceedings*, 2010.

[26] F. Mallet, C. Andre, and J. DeAntoni, "Executing aadl models with uml/marte," in *Proceedings of the 2009 14th IEEE International Conference on Engineering of Complex Computer Systems*, ICECCS '09, (Washington, DC, USA), pp. 371–376, IEEE Computer Society, 2009.

[28] P. Derler, E. A. Lee, S. Tripakis, and M. Törngren, "Cyber-physical system design contracts," in *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, ICCPS '13, (New York, NY, USA), pp. 109–118, ACM, 2013.

[29] "Omg systems modeling language." http://www.omgsysml.org/, Last visited: march 2019.

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

[31] J. Makio, E. Mäkiö-Marusik, E. Yablochnikov, V. Arckhipov, and K. Kipriianov, "Teaching cyber physical systems engineering," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 3530–3535, Oct 2017.

[32] J. I. Sosa, R. R. Recanzone, F. Ardoli, A. Aresi, C. Bender, N. Blet, and J. L. Simón, "Industrial plant at academic level for teaching industrial informatics in an electronic engineering undergraduate degree," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 12, pp. 1–9, Feb 2017.

[33] H. Posadas and E. Villar, "Using professional resources for teaching embedded sw development," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 11, 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 *2016 IEEE Biennial Congress of Argentina (ARGENCON)*, pp. 1–7, June 2016.

[35] D. Riesco, P. Martellotto, and G. Montejano, "Uml and the unified process," ch. Extension to UML Using Stereotypes, pp. 273–293, Hershey, PA, USA: IGI Global, 2003.

[36] "Usecases.org." http://www.usecases.org/.

[37] A. Cockburn, "Structuring use cases with goals," *Journal of Object-Oriented Programming*, Sep-Oct, Nov-Dec 1997.

[38] I. Jacobson and P.-W. Ng, *Aspect-Oriented Software Development with Use Cases*. Addison-Wesley Professional, 2004.

[39] K. Rui and G. Butler, "Refactoring use case models: The metamodel," in *Proceedings of the 26th Australasian Computer Science Conference - Volume 16*, ACSC '03, (Darlinghurst, Australia, Australia), pp. 301–308, Australian Computer Society, Inc., 2003.

[40] J. Xu, W. Yu, K. Rui, and G. Butler, "Use case refactoring: a tool and a case study," in *11th Asia-Pacific Software Engineering Conference*, pp. 484–491, Nov 2004.

[41] C. Marcos, A. Rago, and J. A. D. Pace, "Improving use case specifications by means of refactoring," *IEEE Latin America Transactions*, vol. 13, no. 4, pp. 1135–1140, 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," Tech. Rep. DIAB-11-01-1, Universidad de Castilla - La Mancha, 2011.

[43] J. M. Almendros-Jiménez and L. Iribarne, "Describing use-case relationships with sequence diagrams," *Comput. J.*, vol. 50, pp. 116–128, Jan. 2007.

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

[45] G. Eggly, M. Blackhall, A. de Araújo Gomes, R. Santos, M. de Araújo, and M. Pistonesi, "Emitter/receiver piezoelectric films coupled to flow-batch analyzer for acoustic determination of free glycerol in biodiesel without chemicals/external pretreatment," *Microchemical Journal*, vol. 138, pp. 296–302, 2018. cited By 1.

[46] G. Eggly, M. Nabaes, M. Di Nezio, M. Centurión, R. Santos, and M. Pistonesi, "Embedded flow-batch system with electrochemical detection for the determination of lead in propolis samples," *International Journal of Environmental Analytical Chemistry*, vol. 97, no. 10, pp. 922–934, 2017. cited By 0.