Desarrollo y utilización en la enseñanza de grado de herramientas de simulación y depuración para el modelado e implementación de sistemas embebidos: La investigación al servicio de la didáctica.

M. Giura, N. González, M. Trujillo, L. Sugezky, M. Prieto, J. Cruz

Departamento de Ingeniería Electrónica
Facultad Regional Buenos Aires, Universidad Tecnológica Nacional
Medrano 951, mgiura@frba.utn.edu.ar

Resumen

El uso de modelos para describir el software en sistemas embebidos es una metodología cada vez más frecuente. El desarrollo de un software libre que permita la creación de modelos, su simulación, su depuración en tiempo real y su representación en lenguaje C eligiendo diferentes estrategias de programación, resulta de gran interés para su implementación en el ámbito académico y, eventualmente, en el ámbito profesional. Enmarcado en la cátedra de la asignatura Informática II de la carrera Ingeniería Electrónica en la FR Buenos Aires de la UTN, cuatro profesores nos propusimos realizar un desarrollo experimental en el marco de un PID cuyo producido sea un software de las características mencionadas. Su utilización en el contexto del dictado de la asignatura posibilita a los docentes contar con una nueva herramienta didáctica adecuada a las necesidades de la cátedra que facilita la aprehensión del conocimiento debido a su flexibilidad y potencialidad para modelar situaciones problemáticas reales y obtener así diferentes implementaciones de software, así como también, la documentación asociada al modelo.

Palabras clave: *sistemas embebidos, modelos, didáctica.*

1. Introducción

La experiencia de cátedra aquí presentada constituye un subproducto del proyecto de investigación y desarrollo denominado "Desarrollo de herramientas de depuración para el modelado e implementación de sistemas embebidos utilizando uModelFactory". Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Código: EIUTNBA0004734.

Este proyecto ha sido incubado dentro de la Cátedra de la asignatura Informática II de la carrera de Ingeniería Electrónica de la institución.

El proyecto actual es la continuidad de dos proyectos anteriores que se tomaron como punto de partida [10].

2. Marco teórico

El desarrollo ya mencionado, contaba con un entorno integrado de desarrollo (IDE, del inglés Integrated Development Environment) a nivel didáctico, que permitía el modelado gráfico de diagramas de estado y la simulación sobre el propio modelado para su posterior implementación en aplicaciones industriales sobre sistemas embebidos.

Se trata de un software para PC denominado *uModel Factory* con interfaz gráfica, multiplataforma, open source y de uso libre; que oficia de herramienta para el modelado, y posterior simulación del funcionamiento de una aplicación de control en sistemas embebidos; y cuyo producto final es –una vez definida la plataforma de hardware objetivo- la codificación en lenguaje de programación C que la represente.

Para llegar a ello, partimos del conocimiento que los modelos, desde la perspectiva de la programación, no se los utiliza para visualizar código, sino para representar un sistema con un nivel de abstracción superior al de los lenguajes de programación. Los

modelos ayudan a comprender el sistema a diseñar y favorecen el intercambio de ideas.

Un **modelo** es una **representación simplificada** de un sistema que contempla las propiedades importantes del mismo desde un determinado punto de vista.

Desde una mirada constructivista [3] la promoción de situaciones de trabajo colectivas que fomenten la discusión, el análisis crítico y la evaluación de diferentes formas de resolver problemas facilitarán la aprehensión del conocimiento.

En ese sentido, las herramientas de simulación son una pieza fundamental y cumplen un rol significativo en los procesos de enseñanza-aprendizaje. [8]

En la actualidad, el estándar UML (Unified Modeling Language) se ha convertido en una herramienta usual para programadores e ingenieros en sistemas de información. El uso principal de UML es el modelado de sistemas e incluye tanto el análisis como el diseño [7].

Resulta interesante destacar que *uModel Factory* fue concebido bajo una perspectiva didáctica que se aprecia en el hecho que genera tres implementaciones diferentes del código que representa el modelo creado (if anidados, switch-case y arreglo de punteros a función, en lenguaje C), lo cual favorece la discusión y el análisis dentro del aula[5].

Su desarrollo incluyó una interfaz gráfica a los fines de priorizar la claridad de conceptos, facilitar el uso y potenciar el análisis de comportamiento.

Dicha interfaz está basada en tres pilares:

- Modelado gráfico del algoritmo de control.
- Incorporación de periféricos terminales.
- Generación de código C para diferentes plataformas con absoluta transparencia.

Como característica destacable, cabe señalar que *uModel Factory* mantiene sincronizado, en tiempo de desarrollo, el modelo, código generado y documentación.

La sincronización señalada se produce a partir del diagrama de estados y transiciones que una vez concluido permite obtener el código C y la documentación pertinente. Si se requieren cambios, y mantener la sincronización, debe modificarse el diagrama para obtener el nuevo código C y su documentación asociada.

Los objetivos de la investigación en esa primera etapa fueron:

a) Analizar el software de modelado disponible y su potencialidad para educación.
b) Reconocer sobre dicho relevamiento las limitaciones y/o ventajas que pudiesen presentar.
c) Determinar las necesidades formativas de docentes para su uso en el aula.

Luego de realizado el análisis de diferentes software de modelado disponibles [1, 11], se encontró que, si bien existe una variedad de herramientas para desarrollar aplicaciones utilizando una metodología gráfica, a la hora de realizar una aplicación embebida, todos tienen restricciones de algún tipo y, por sobre todas las cosas, ninguno permite enfocar la lógica a un diagrama de estados.

Sin excepción, todas las aplicaciones relevadas se encuentran disponibles en inglés, que, si bien es el lenguaje técnico por excelencia, termina agregando un escalón más de dificultad en la curva de aprendizaje.

La gran mayoría de las aplicaciones existentes poseen una serie de desventajas comunes:

- Focalización al diseño de aplicaciones de alto nivel para el ambiente de sistemas.
- Inadecuación del código fuente al uso en plataformas embebidas.
- Desaprovechamiento de recursos a la hora de generar el código final.
- Utilización desmedida de librerías para realizar acciones básicas con el consecuente desaprovechamiento de recursos.
- Curvas de aprendizaje muy extensas o empinadas.
- Dificultad de utilización como

herramienta de enseñanza.

- En el caso de poseer una orientación a la enseñanza, no se genera código susceptible de ser utilizado en proyectos profesionales.
- Imposibilidad de expansión.
- Código cerrado.
- Licencias pagas.
- Compatibilidad sólo con Windows.
- Sólo con interfaz en inglés.

Una de las herramientas evaluadas es el framework RKH diseñado por el Ing. Leandro Francucci de la Universidad Nacional de Mar del Plata. [4]

El framework RKH es un entorno de trabajo open-source para el desarrollo de sistemas reactivos, que consiste en un framework multiplataforma para desarrollo de software embebido basado en diagramas de estado y el paradigma de la programación gobernada por eventos. Como limitación de esta herramienta encontramos que no posee una interfaz de usuario que facilite el trabajo de estudiantes noveles.

En ese sentido, el equipo de trabajo que desarrollo este framework, de fuerte perfil profesional, se sumó al presente proyecto de forma de concebir una herramienta que junte las fortalezas de ambos emprendimientos y dé a luz un nuevo desarrollo que no solo apunte al enfoque didáctico sino también al uso intensivo profesional en la industria del sector.

La posibilidad de incorporar el framework RKH le ha dado a *uModel Factory* un salto de amplio espectro puesto que RKH tiene resuelta la generación de código sobre numerosas plataformas de hardware.

Para finalizar esta sección introductoria, cabe señalar que una funcionalidad poco implementada en las herramientas comerciales evaluadas es la incorporación de algún mecanismo de depuración en tiempo real que, desde el punto de vista pedagógico, consideramos vital pues es un fuerte integrador de varios conocimientos que subyacen en el trabajo con sistemas embebidos.

Al respecto, cabe decir que existen diferentes enfoques orientados a la depuración en sistemas embebidos [2,6,9], principalmente pueden clasificarse en intrusivos o no intrusivos. A su vez, podemos evaluar los mismos como de tiempo real o de tiempo diferido.

El enfoque adoptado en el desarrollo aquí presentado es intrusivo y de tiempo real.

3. Objetivos y Metodología

Esa versión anterior de la aplicación no contaba con la posibilidad de modelar máquinas de estado concurrentes, ni tampoco permitía la depuración del modelo en tiempo real, lo cual constituía una limitante desde la perspectiva didáctica que nos ocupa.

Estas funcionalidades fueron incorporadas en la versión actual de *uModel Factory*.

Los objetivos específicos que nos propusimos fueron:

a) Desarrollar un software que permita la depuración (además de la simulación) del modelo previamente realizado.

b) Elaborar los algoritmos de software necesarios que permitan:

b1. Evaluar el comportamiento real del modelo a partir de su gráfico.

b2. Generar archivos con una lista de eventos temporizados para simular funcionamiento continuo.

b3. Realizar el software necesario para la incorporación de elementos gráficos de entrada y salida de información (Teclados, display, entre otros)

c) Evaluar el funcionamiento del modelo.

d) Generar código C para nuevas plataformas de hardware.

d1. Implementación del software utilizando código ansi C compatible. d2. Generación del código utilizando el framework RKH.

e) Fortalecer los aspectos didácticos en el diseño de la interfaz de usuario, asegurando que los conceptos principales se pongan en juego en cada operación.

f) Evaluar el desarrollo en su entorno de aplicación.

g) Favorecer la participación de la comunidad disciplinar a los efectos de beneficiar su difusión.

Para establecer los objetivos específicos del proyecto se realizaron numerosos encuentros de cátedra tanto con profesores como con auxiliares a los efectos de evaluar los avances, así como también recepcionar ideas a partir de la experiencia que cada equipo docente viene teniendo en el uso de la herramienta desde sus primeros inicios en 2014.

4. Resultados

La interfaz principal permite diseñar gráficamente diagramas de estados correspondientes a máquinas concurrentes (figura 1).

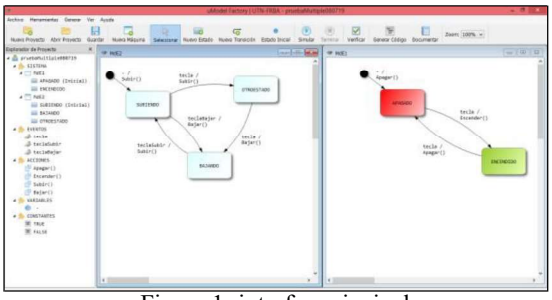


Figura 1: interfaz principal

Dentro del entorno de desarrollo encontramos una barra de menús y herramientas que permiten realizar las acciones más habituales (figura 2) entre las cuales podemos destacar:

- Crear, abrir y guardar proyecto
- Crear, Editar y Borrar estado
- Generar código de acuerdo al método elegido
- Simular el o los modelos propuestos
- Depurar el o los modelos propuestos
- Generar documentación

Figura 2: barra de menus y herramientas

Una vez definidos los estados pertenecientes al diagrama, el software permite definir las transiciones entre estados. La versión actual permite definir nuevos eventos (funciones booleanas o cambios en variables) y acciones (llamada a función o actualizaciones de variables) en relación con el proyecto o situación problemática (figura 3) a través de la generación de eventos. A su vez, también permite redefinir el nombre del estado y su color. (figura 4)

Para definir las transiciones de un estado, se deberán seleccionar los siguientes ítems:

- El estado al que se desea llegar.
- El evento que genera la transición (condición)
- El conjunto de acciones asociadas al evento.

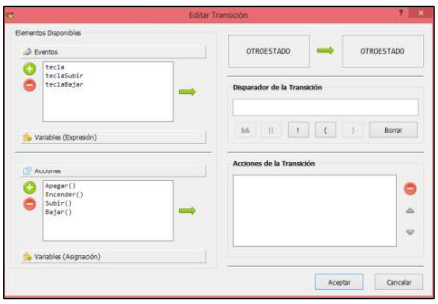
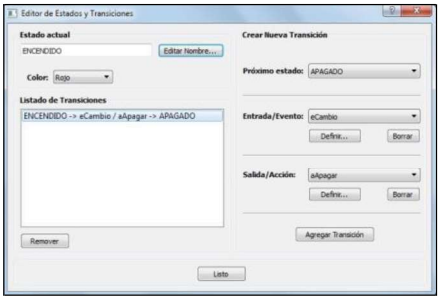


Figura 3: transiciones entre estados



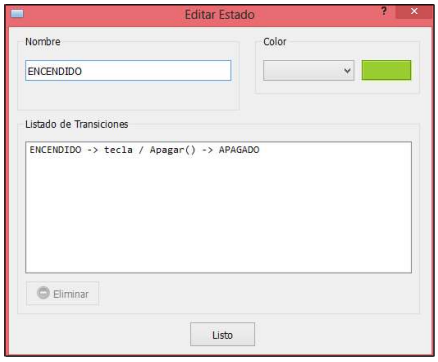
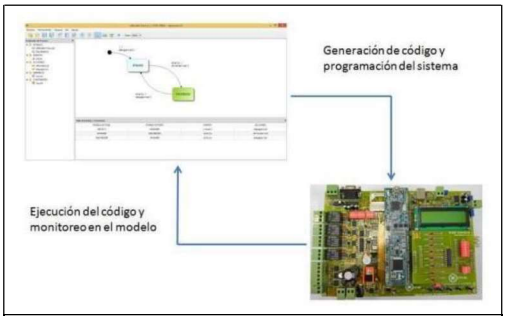


Figura 4: ajustes sobre las características del estado



Con relación al código generado, se agrega al ANSI C existente, la posibilidad de producir código compatible con el framework RKH, de amplia utilización en la industria.

Por otro lado, se adaptó el módulo generador de documentación dinámica del proyecto a los efectos de que contemple potencialmente N máquinas de estado. (figura 5)

Dentro de la documentación generada se encuentra:

- Nombre del proyecto
- Autores
- Fecha de modificación
- Diagrama de estados
- Listado de eventos
- Listado de acciones
- Tabla de estados asociada

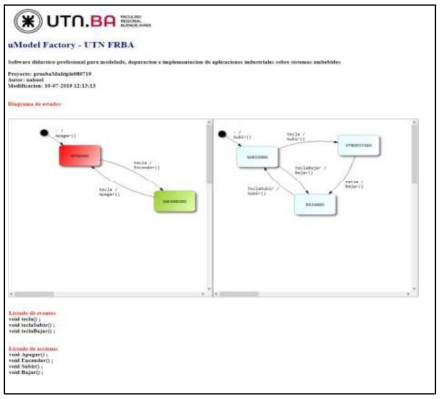


Figura 5: generación de documentación

Si bien el proceso de simulación permitía detectar errores previos al momento de ejecución sobre el hardware, aún no era posible monitorear el funcionamiento del sistema embebido en operación. (figura 6)

Figura 6: depuración del modelo

La aplicación aquí presentada se desarrolló en su totalidad utilizando el framework Qt (de uso libre) y se codificó en modo portable (Linux-Windows) en C++. En todo momento del desarrollo no se perdió de vista los aspectos didácticos en el diseño de la interfaz de usuario, asegurando que los conceptos principales se pongan en juego explícitamente en cada operación.

5. Conclusiones

Desde la perspectiva del desarrollo del producto, *uModel Factory* cumple con los objetivos propuestos.

Desde la perspectiva didáctica, los profesores creemos que su utilización favorece la aprehensión de un tema central de la asignatura como lo es la modelización por medio de máquinas de estados, tal como lo prueban los resultados de una encuesta realizada entre el profesorado, luego de una jornada de capacitación sobre la herramienta realizada en Mayo de este año:

- El 58% utilizó herramientas similares previamente (IAR visualSTATE).
- El 71% consideró que presenta una interfaz más simple que facilita la creación de modelos en relación con la herramienta que conocían.
- El 91% considera que la herramienta presentada resultará útil para los estudiantes de Informática II del ciclo lectivo 2019.
- El 50% considera que es una herramienta útil para complementar la preparación del examen final.

Adicionalmente, la incorporación de la funcionalidad de depuración [2,6,9], de uso habitual en la industria permite, no solo llevar a un nivel superior la intensidad de la formación al poner en juego la teoría con la práctica real y concreta, sino también abordar otro tema central de la asignatura como lo es la comunicación asincrónica serial, por medio de la cual se realiza el proceso de depuración.

Como trabajos a futuro se propone:

- Liberar los fuentes del software con la documentación necesario para el desarrollo de plugins que amplíen sus funcionalidades.

  Rediseñar dispositivos de aprendizaje durante el segundo semestre 2019 a fin de analizar el nivel de recepción de los estudiantes de las nuevas funcionalidades incorporadas.
- Implementar estos dispositivos y evaluar su impacto en el trayecto académico de los estudiantes.

Referencias

[1] Altova UModel. Disponible en: http://www.altova.com/es/umodel/uml-code-generation.html (última fecha de acceso: julio de 2019) .

[2] Campbell, J. - Kazantsev, V. - O'Keeffe, H. (2014). Real-time Trace: *A Better Way to Debug Embedded Applications*. White paper. Synopsys.

[3] Diaz Santana (2004). *Enfoque constructivista como herramienta para el aprendizaje*. Centro de Competencias de la Comunicación. Universidad de Humacao. Puerto Rico.

[4] Framework RKH: https://www.vortexmakes.com/que-es/ (última fecha de acceso: julio de 2019)

[5] Guía de trabajos prácticos de clase 6. Informática II. Departamento de Ing. Electrónica. UTN – FRBA.

[6] Gracioli, G. - Fischmeister, S. (2012). *Tracing Interrupts in Embedded Software*. Journal of Systems Architecture. Volume 58, Issue 9.

[7] Grady Booch - Rumbaugh, J – Jacobson, I. (1998). *The Unified Modeling Language User Guide*, Addison-Wesley. ISBN 0-201-57168-4

[8] Joaquim, P. – Gonzalez, N. – Navarro, C. (2012). *Influencia del software de simulación en la Aprehensión del Conocimiento*. II Jornadas de Enseñanza de la Ingeniería (JEIN).

[9] Kraft, J. – Wall, A. – Kienle, H. (2010). *Trace Recording forEmbedded Systems: Lessons Learned from Five Industrial Projects*. En: Barringer H. et al.(eds) Runtime Verification. RV

[10] PIDs UTN1562/2012 y PID UTN2436/2015

[11] Visual Paradigm. Disponible en: https://www.visual-paradigm.com /features/code-engineering/(última fecha de acceso: julio de 2019)