Desarrollo e implementación de herramientas de simulación de modelos para sistemas embebidos

***Abstract*— uModel Factory es un software didáctico-profesional de modelado de aplicaciones para sistemas embebidos, desarrollado en la UTN FRBA en el marco de un proyecto de Investigación y Desarrollo del Departamento de Ingeniería Electrónica (PID UTN1562, Resolución CD FRBA N°2040/11). Permite la creación de un diagrama de estados a través de su interfaz gráfica, como así también la generación automática de código C portable y toda la documentación asociada, manteniendo en todo momento sincronismo entre modelo, código generado y documentación. En este trabajo se presenta el diseño e implementación de herramientas de software para simular y validar diagramas de estados orientados a sistemas embebidos, con el objetivo de integrarlas en una nueva versión del software de modelado uModel Factory.**

Keywords—UML; sistemas embebidos; modelo ; simulación

I. INTRODUCCIÓN

Cada vez es más frecuente el uso de modelos para describir el software de los sistemas embebidos, ya que ayudan a comprender el sistema y a diseñar con un nivel de abstracción superior al de los lenguajes de programación. En cierta forma, la migración de una metodología de programación basada en lenguaje C hacia el desarrollo de software basado en modelos supone un incremento en nivel de abstracción y en productividad similar al producido al cambiar desde lenguaje assembler hacia lenguaje C.

Un modelo es una representación simplificada de un sistema que contempla las propiedades importantes del mismo desde un determinado punto de vista. El uso de modelos es una actividad arraigada en técnicos e ingenieros y probablemente tan antigua como la propia ingeniería.

Además de servir para lograr un conocimiento más profundo del problema, favorecen el intercambio de ideas entre las personas involucradas en el diseño. La mayoría de los enfoques actuales en el desarrollo de software basado en modelos coinciden en [1,2]:

- Utilizar una representación gráfica del sistema a desarrollar
- Describir el sistema con un cierto grado de abstracción.
- Generar código ejecutable para el sistema embebido partiendo del propio modelo

Tanto en la industria como en el ámbito académico se han utilizado durante mucho tiempo los diagramas de flujo (data flow diagram -dfd-) ya que permiten una rápida visualización de la solución propuesta para un problema permitiendo seguir la lógica desarrollada en forma simple [3,4].

Dentro de las representaciones también encontramos, las máquinas de estados finitos (FSM por Finite State Machine) las cuales constituyen una herramienta gráfica que ha sido utilizada tradicionalmente para modelar el comportamiento de sistemas electrónicos e informáticos. Como una amplia extensión al formalismo convencional de estas máquinas surgieron los diagramas de estado (Statecharts) los cuales se convirtieron en parte del estándar UML (Unified Modeling Language) para describir el comportamiento de sistemas o de modelos abstractos.

Los diagramas de estado muestran el conjunto de estados por los cuales pasa un objeto durante su vida [5] en una aplicación, para el pasaje de este objeto por los estados del modelo se analiza su respuesta a eventos (por ejemplo, mensajes recibidos, tiempos cumplidos o errores) y se vincula con sus respuestas y acciones (figura 1).

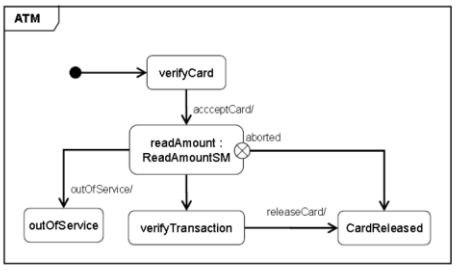

Figura 1 – Diagrama de estados de un cajero automático (ATM)

Estos diagramas normalmente contienen estados, transiciones, eventos, acciones y actividades. Un evento es una ocurrencia que puede causar la transición de un estado a otro del sistema. Esta ocurrencia se puede deber a distintas condiciones como puede ser: una condición que toma el valor de verdadero o falso (expresión booleana); la recepción de una señal o mensaje externo; o el paso de cierto período de tiempo. Una acción es una operación atómica, que no se puede

interrumpir por un evento y que se ejecuta hasta su finalización.

En este trabajo se presenta el desarrollo e implementación de herramientas de software que permitan la simulación y validación de diagramas de estados orientados a sistemas embebidos. El objetivo final es la integración de estas nuevas herramientas con el software uModel Factory.

uModel Factory es un software didáctico-profesional de modelado de aplicaciones para sistemas embebidos, desarrollado en la UTN FRBA en el marco de un proyecto de Investigación y Desarrollo del Departamento de Ingeniería Electrónica (PID UTN1562, Resolución CD FRBA N°2040/11).

Permite la creación de un diagrama de estados a través de su interfaz gráfica, como así también la generación automática de código C portable y toda la documentación asociada, manteniendo en todo momento sincronismo entre modelo, código generado y documentación (figura 2).

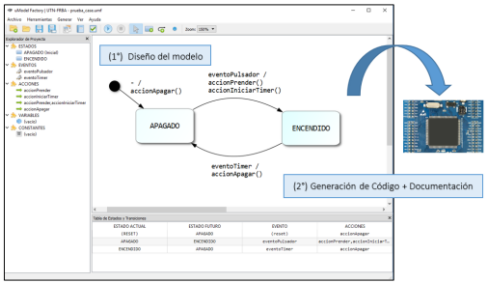

Figura 2 – Interfaz principal de uModel Factory v1.0

Si bien existen otras herramientas que posibilitan la generación de código en diferentes lenguajes a partir de la representación de su modelo [6,7] poseen una curva de aprendizaje pronunciada. Es por ello que esta herramienta posee un enfoque didáctico ya que genera el código en lenguaje C en diferentes implementaciones, lo cual favorece la discusión y el análisis dentro del aula [8]. Este software se ha utilizado en la cursada de Informática II como herramienta didáctica durante los ciclos lectivos 2014 y 2015.

II. ALCANCE Y OBJETIVOS

La versión 2014 de uModel Factory permitía la creación de diagramas de estados a través de su interfaz gráfica pero no contaba con ninguna herramienta para que el usuario pudiese simular o validar el modelo realizado. Esta limitación implica que para poder analizar el comportamiento del modelo resulta necesario hacerlo directamente sobre el sistema embebido, y en el caso de que no funcione como se esperaba se debe volver al primer paso y editar el modelo.

El objetivo principal del proyecto es la nueva versión de uModel Factory (v2.0) la cual permita simular el modelo de un sistema de acuerdo a periféricos de entrada/salida y que el usuario pueda generar el código resultante una vez atravesado el proceso de depuración.

A partir del desarrollo se prevé que se podrán generar eventos o disparar condiciones particulares para guiar la ejecución hacia los puntos a analizar, y ejecutar el modelo en tiempo real para permitir la detección de inconvenientes en el funcionamiento.

~~(NO ENTIENDO BIEN ESTE PARRAFO)~~

Con formato: Color de fuente: Rojo

Los objetivos específicos son:

A. Desarrollar el software que permita la Simulación y Depuración del diagrama de estados.

B. Elaborar los algoritmos de software necesarios que permitan:

   1. Evaluar el comportamiento real del modelo a partir de su representación gráfica.

   2. Incorporar elementos gráficos de entrada y salida de información (Pulsadores, Teclados, Leds, Displays, etc.)

C. Ampliar la capacidad de generación de código C, para nuevas plataformas de hardware.

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

E. Garantizar la participación de la comunidad educativa a los efectos de favorecer la difusión y el crecimiento de uModel Factory.

LIMITACIONES ACTUALES Y PROPUESTAS DE MEJORA

Para que el usuario pudiera desarrollar una aplicación con la versión 2014 de uModel Factory (v1.0) debía, en primer lugar, diseñar el diagrama de estados del sistema en forma gráfica, declarando estados, transiciones, eventos y acciones. Luego debía seleccionar el modo de implementación para iniciar la generación automática de código C. A partir de los archivos .c y .h generados, era necesario utilizar algún entorno de desarrollo apropiado para compilar el código y poder bajar el programa al microcontrolador. Por último, para poder analizar y validar el funcionamiento del modelo, debía ejecutar la aplicación diseñada directamente sobre el sistema embebido (figura 3).

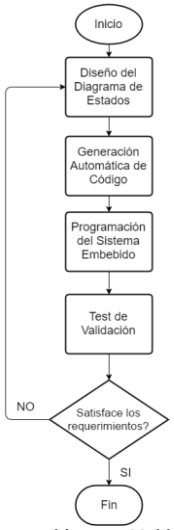

Figura 3 – Secuencia básica con uModel Factory v1.0

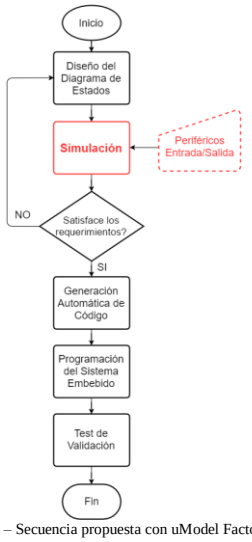

Figura 4 – Secuencia propuesta con uModel Factory v2.0

En el caso de que la aplicación no funcionara como se esperaba, para realizar modificaciones había que volver al primer paso y editar el modelo. Luego se debían recorrer nuevamente todas las etapas siguientes hasta llegar a realizar una nueva prueba. Es en este punto donde aparece la necesidad de contar con una herramienta para poder validar el comportamiento del sistema, desde las primeras etapas del diseño.

Las mejoras propuestas se focalizaron en diseñar las herramientas de software necesarias que permitan la simulación, depuración y validación del modelo de estados planteado por el usuario (figura 4). Contar con la posibilidad de validar que la aplicación se comporta de la forma deseada, antes de generar el código y programar el sistema embebido, es una ventaja importante en lo que respecta a reducir el tiempo de desarrollo y a la localización de errores.

III. RESULTADOS

A continuación se muestra una comparación entre las versiones v1.0 y v2.0 (figura 5 y 6), destacando las mejoras más importantes:

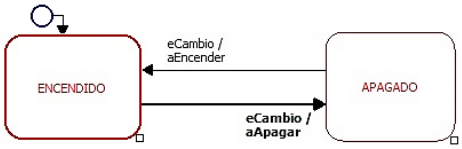

Figura 5 – Diagrama de estados en la versión 2014 de uModel Factory

Limitaciones de la versión

- Opciones limitadas
- Generación de código básica
- Interfaz con el usuario poco amigable
- Sin simulación ni validación
- Pocas posibilidades de editar el diagrama

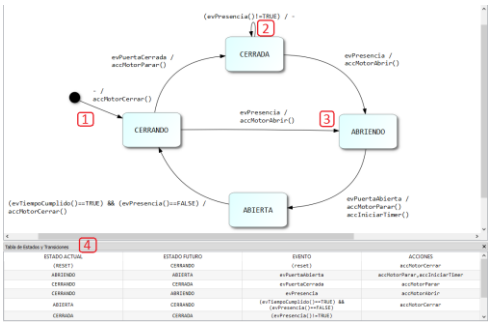

Figura 6 – Diagrama de estados en la nueva versión de uModelFactory

Mejoras incluidas en la nueva versión

1. Mejoras en el Estado Inicial del diagrama. Incorporación de Acciones asociadas al reset, y adaptación de la estructura XML a los nuevos requerimientos.

2. Mejoras en el aspecto visual de las auto-transiciones. Mayor facilidad para seleccionarlas.

3. Mejoras en el aspecto gráfico de los estados. Cambios en la estructura XML.

4. Tabla de Estados y Transiciones. Actualización dinámica con el diagrama.

~~Pondría algo de la depuración~~

Se desarrolló también en esta etapa un editor de transiciones de forma que no sea necesario borrarlas y definirlas nuevamente si se quiere cambiar el modelo, y se mejoró la interfaz para declarar acciones (figura 7).

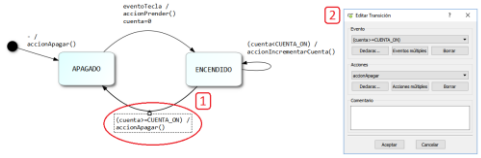

Figura 7 – Editor de Transiciones en la nueva versión de uModel Factory

A su vez, se incorporó la posibilidad de organizar las acciones por prioridad y la asignación de variables, esto se puede realizar por medio de un editor que facilita dicha asignación (figura 8).

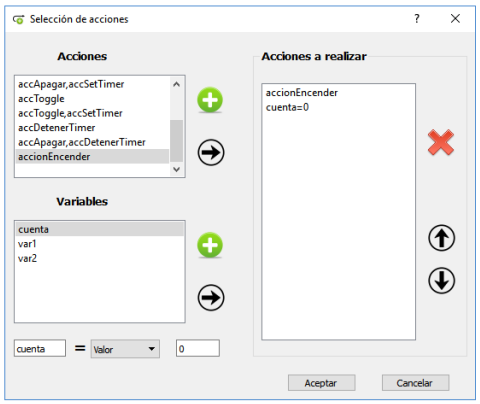

Figura 8 – Acciones múltiples en la nueva versión de uModel Factory

Implementación de la simulación

Para la implementación de la simulación se comenzó con la elaboración de algoritmos para poder simular el sistema a partir de la representación gráfica del modelo. Se estudiaron las posibilidades que ofrecía el framework QtCreator y se optó por generar un "modo" de Simulación que le permite al usuario disparar eventos y visualizar la evolución del sistema entre los distintos estados, para poder validar el correcto funcionamiento del modelo.

En la figura 9 se muestra la barra de herramientas de uModel Factory y se destacan los nuevos controles para ingresar y para salir del modo de Simulación.

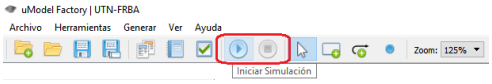

Figura 9 – Controles para el modo Simulación en uModelFactory

Una vez que el usuario inicia la simulación, tiene disponibles una serie de herramientas (figura 10) para guiar la ejecución hacia los puntos que considere críticos y que sea importante validar.

Dichas herramientas permiten ver el estado del sistema, los eventos disponibles en función del estado actual y las acciones asociadas a la última transición realizada. De este modo, el usuario puede chequear el funcionamiento de la lógica planteada y corregir los posibles errores desde las primeras etapas del diseño, ahorrando tiempo en las etapas siguientes.

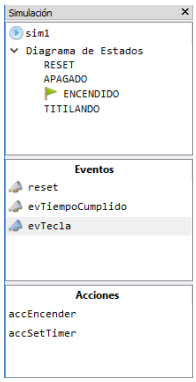

Figura 10 – Herramientas para el modo Simulación

Para el aspecto visual del diagrama de estados, se decidió utilizar distintos colores para hacer énfasis en el estado actual y en las transiciones realizadas. A continuación se muestra la evolución de un sistema simple (figura 11a y 11b), como consecuencia de la interacción del usuario disparando los eventos correspondientes con el estado actual y la transición realizada resaltada como se expresó anteriormente.

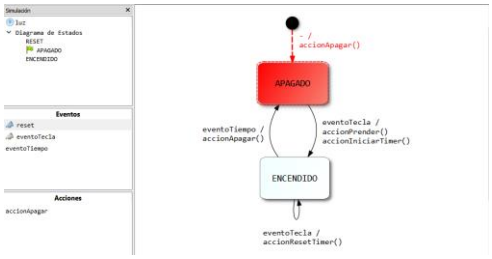

Figura 11a – Simulación, transición entre estados

(RESET a APAGADO)

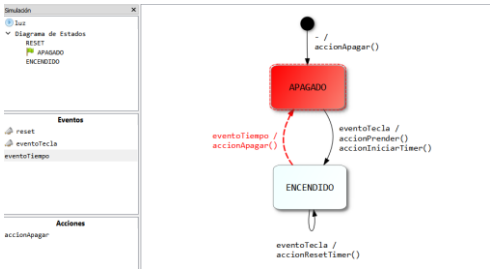

Figura 11b – Simulación, transición entre estados

(ENCENDIDO a APAGADO)

Chequeo de errores

Como complemento a la simulación, se desarrolló una herramienta para poder validar el modelo planteado por el usuario (figura 12).

Figura 12 – Controles para iniciar la validación del modelo

La validación del sistema consiste en evaluar si existen estados sin salida, estados que nunca se alcanzan, y eventos, variables o acciones declaradas que no se utilizan. Con el conjunto de estas nuevas herramientas, se puede visualizar el comportamiento del modelo y encontrar errores que no se habían detectado durante el diseño (figura 13a y 13b).

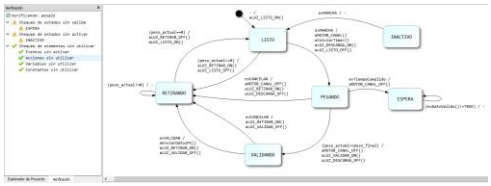

Figura 13a – Informe de validación del modelo, con errores

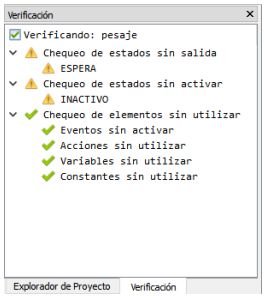

Figura 13b – Informe de validación del modelo, con todos los ítems correctos

IV. TRABAJOS A FUTURO

El objetivo futuro es poder asociar los eventos y las acciones declaradas con entradas y salidas que representen periféricos físicos, de modo tal que pueda comprenderse aún mejor el sistema que se está diseñando (figura 14).

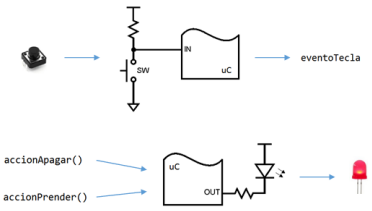

Figura 14 – Periféricos de entrada y salida, vinculados con eventos y acciones

Por ello la estructura de código que controla la simulación fue diseñada para poder integrarse con estos módulos de elementos gráficos que representen periféricos (hardware) de entrada y salida (Pulsadores, relés, displays, teclados matriciales, etc.).

V. CONCLUSIONES

En el presente trabajo se diseñaron e implementaron herramientas de software para simular y validar diagramas de estados orientados a sistemas embebidos, y se integraron en una nueva versión del software de modelado uModel Factory.

Para lograr los objetivos propuestos, fue necesario trabajar sobre distintas áreas vinculadas tanto a la programación de sistemas embebidos, como a la modelización aplicada en el desarrollo de software, sin dejar de lado el aspecto educativo y formativo de su uso como aplicación didáctica.

A su vez, se estudiaron distintas posibilidades y se desarrollaron las soluciones para incorporar nuevas funcionalidades a uModel Factory. Se implementaron e integraron las herramientas de simulación y de validación, las cuales permiten que el usuario pueda generar código luego de validar el modelo. Durante el desarrollo y la codificación, se tuvo en consideración lo importante que resulta que el código incorporado permita que se puedan seguir agregando nuevas prestaciones a futuro y que la aplicación siga creciendo.

El trabajo realizado generó como resultado una nueva versión de uModel Factory para ser utilizada durante el próximo ciclo lectivo 2016 en el contexto de la materia Informática II perteneciente al Departamento de Ingeniería Electrónica (UTN FRBA).

VI. REFERENCIAS

[1] G. Booch, J. Rumbaugh, I. Jacobson. “El Lenguaje Unificado de Modelado”. Addison-Wesley 2nd Edition, 2006.

[2] G. Booch, J. Rumbaugh, I. Jacobson. “El Proceso Unificado de Desarrollo de Software”. Addison-Wesley 1st Edition, 2000.

[3] P. Ward." The transformation schema: An extension of the data flow diagram to represent control and timing". IEEE Transactions on Software Engineering, 1986.

[4] T. Arndt, A. Guercio. "Decomposition of data flow diagrams". Software Engineering and Knowledge Engineering, 1992.

[5] C. Larman. “UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado”. Prentice-Hall, 2003.

[6] Visual Paradigm. Referencia: https://www.visual-paradigm.com/features/code-engineering/

[7] Altova UModel. Referencia: http://www.altova.com/es/umodel/uml-code-generation.html

[8] N. González, L. Sugezky, M. Prieto, M. Giura, Y. Kuo, M. Trujillo, J.M. Cruz. “Evaluación del software uModelFactory como herramienta didáctica”. IEEE Argencon, 2016.