Herramientas de apoyo al estudio de sistemas embebidos utilizando uModelFactory

Almaraz, Nicolás Tobías
Estudiante de Ing. Electrónica
UTN FRBA
CABA, Argentina
nalmaraz@frba.utn.edu.ar

Nirino, Felipe
Estudiante de Ing. Electrónica
UTN FRBA
CABA, Argentina
fnirino@frba.utn.edu.ar

***Resumen* — A partir del trabajo realizado desde el departamento de ingeniería electrónica de la UTN FRBA este equipo de trabajo desarrolló un software llamado uModelFactory. El mismo permite el diseño, depuración y simulación de máquinas de estados finitas tipo Moore. Dicha herramienta fue concebida para ser utilizada a la hora de diseñar sistemas embebidos. Es por ello que la cátedra de la asignatura Informática II decide implementarla como herramienta didáctica. Sin embargo, los datos provenientes de encuestas realizadas al alumnado revelan que este software no tenía una interfaz lo suficientemente intuitiva o cómoda para desarrolladores noveles. Es por ello, que el equipo de desarrollo decide lanzar una sección de ayuda dinámica que facilite el acceso a la información acerca de conceptos teóricos. Para esto se implementa un sistema con tres niveles de ayuda que muestran información acerca de lo que el cursor del mouse esté apuntando. Los resultados de esta nueva funcionalidad fueron positivos y motivan a futuros desarrollos.**

Palabras Clave — Máquinas de estados, Sistemas embebidos, Enseñanza y Aprendizaje.

I. Introducción

El software "uModelFactory" [1] se presenta como una herramienta para asistir en la elaboración de diagramas y Máquinas de Estado Finitas (MEF) tipo Moore gobernados por eventos. A partir de trabajos anteriores, se ha analizado el uso de la misma en el ámbito educativo [2], particularmente en la cátedra de Informática II de la Universidad Tecnológica Nacional - Facultad Regional Buenos Aires, como una tecnología que facilite la enseñanza y el aprendizaje del proceso de diseño y diagramado de MEF con complejidad creciente.

De la retroalimentación obtenida por parte del alumnado general, recolectada a partir de instrumentos como encuestas cerradas o de las opiniones expresas por parte de los respectivos docentes, se realizó un análisis de las falencias, defectos y limitaciones a la funcionalidad del software. Los resultados revelaron en esta instancia una prominente manifestación de falta de intuitividad y de facilidad de uso que perjudica la apropiación de conocimiento y causa frustraciones en el conjunto de los estudiantes.

Teniendo en cuenta la dificultad, se optó por incluir un apartado de ayuda dinámica que favorezca la interacción con el programa y presente al usuario diferentes niveles de información relativa a los elementos que componen los diagramas.

II. Marco Teórico

A. Concepción de máquinas de estados

Una máquina de estados es una estrategia de modelizar el comportamiento de cualquier tipo de sistema (por ejemplo, una cafetera, un proceso de fabricación industrial, las condiciones de aprobación de un examen, etc).

El proceso de diseñar una máquina de estados consiste en dividir el sistema en diferentes instancias llamadas "estados" que cada uno de ellos representa una etapa del proceso. Debido a las excitaciones o estímulos de entrada podremos pasar de un estado a otro. A este cambio de estado se lo denomina transición y aquí se deben aclarar qué salidas del sistema deben cambiar frente a la excitación que la produjo.

B. Componentes de una máquina de estados

Las máquinas de estados pueden descomponerse en cuatro pilares esenciales (Fig. 1):

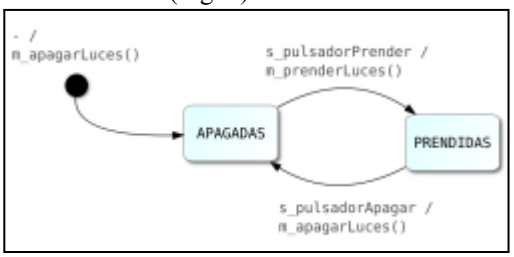

Fig. 1. Diagrama con los pilares de una máquina de estados elaborado en "uModelFactory"

- Estados: Son las diferentes etapas por las que tiene que pasar un sistema para completar un proceso. Para avanzar de un estado a otro se hace a través de las transiciones. En los diagramas son representados por recuadros (estados "prendidas" y "apagadas").
- Transiciones: Es el paso de un estado a otro. Estas se dan cuando se cumplen ciertas condiciones de entrada (eventos) y son representadas por flechas que van desde un estado de origen a un estado de destino. Se dice que una transición se dispara frente a un evento determinado. Por ejemplo: Pasar de "prendidas" a "apagadas" al recibir el evento "s_pulsadorApagar". En estas transiciones se modifican las salidas o alguna variable interna en caso de ser necesario (acciones). A la hora de representarlas en el diagrama se debe detallar "eventos / acciones", es decir, qué evento la disparó y qué acción debe ejecutarse. Por ejemplo, si se presiona el pulsador de apagado, entonces apagar las luces.
- Eventos: Representan los estímulos externos o internos que recibe la máquina de estados, y a los que la misma responderá. Por ejemplo, la pulsación de un botón que resulta en la generación de un evento "s_pulsadorApagar".
- Acciones: Determinan los comportamientos al producirse una transición de un estado a otro. Por ejemplo, encender o apagar las luces descritas en los ejemplos anteriores.

C. *Estados compuestos y máquinas de estados concurrentes*

Los sistemas a describir pueden tener un alto nivel de complejidad, por lo que suelen agregarse a los pilares fundamentales herramientas más elaboradas, como ser los estados compuestos (estados que poseen máquinas de estados dentro de sí) y las máquinas concurrentes (aquellas que coexisten en un mismo período temporal, procesando eventos en paralelo).

Estas dos configuraciones son soportadas por "uModelFactory", permitiendo no solo el uso sino la simulación de las mismas, presentándose entonces como un software que goza de un gran poder de aplicación.

D. *Áreas de aplicación*

Uno de los campos en los que se utiliza la metodología de máquinas de estado es el de los sistemas embebidos, donde deben describirse sistemas que respondan a estímulos y lleven a cabo determinadas tareas. Este método permite ordenar el procedimiento de cada tarea a realizar, facilitando la implementación del sistema y reduciendo la posibilidad de fallos o comportamientos erráticos.

El software "uModelFactory" ha sido diseñado especialmente para dicho campo, contando con la posibilidad tanto de simular la ejecución como de generar el código necesario para la correcta implementación en un microcontrolador programable [3].

III. Materiales y Métodos

Las opiniones del alumnado han sido registradas por los docentes de cada curso a partir de encuestas anónimas, de manifestaciones expresas de los alumnos, e inferencias basadas en consultas y dudas recurrentes durante el periodo de clases. La encuesta anónima incluía tanto aspectos personales (tiempo dedicado a estudio, facilidad de seguir las explicaciones del docente, posesión de hardware necesario para correr el software, etc.) como aspectos relativos a facilidad de uso y efectividad en la comprensión de conceptos a partir del uso de la plataforma. Ejemplos de algunas de las preguntas utilizadas se presentan a continuación:

- ¿Cómo calificarías el uso del software para el modelo de MEF?
- ¿Cómo calificarías su interfaz gráfica y experiencia de usuario?
- ¿Cómo calificarías la generación de código brindada?

Con esta información, se procedió a elaborar un sistema de ayuda que brinde información sobre conceptos clave del marco teórico asociado a los diagramas o las MEF. Para determinar su contenido, el cuerpo docente relevó información detallada sobre los diferentes cursos. A esta se aplicó un filtro con el fin de obtener las críticas hechas pura y exclusivamente a las características y las formas de presentación del software.

Analizando los resultados se optó para proceder un desarrollo segmentado en torno a tres conceptos principales elaborados a continuación: División cualitativa de los temas de ayuda en acotados grupos o "niveles", fácil traducción y/o alteración, y una interfaz efectiva e intuitiva.

Para el desarrollo del proyecto se utilizó el framework de Qt. La elección fue dada por contar con una amplia variedad de bibliotecas fácilmente adaptables para resolver las diferentes problemáticas que se describen. Por otro lado, tiene una muy detallada documentación y ejemplos que facilitan la tarea del programador.

IV. Resultados

En la captura de pantalla mostrada en la Fig. 2 se puede ver que cuando el mouse se encuentra sobre los diferentes ítems de la máquina en la ventana de la derecha nos mostrará la información referida a lo que se encuentre en la posición del cursor. A su vez, en la Fig. 3 se detalla la ayuda correspondiente a los elementos seleccionados del diagrama.

Fig. 2. Captura de pantalla de la aplicación con el panel de ayuda desplegado y resaltado en rojo.

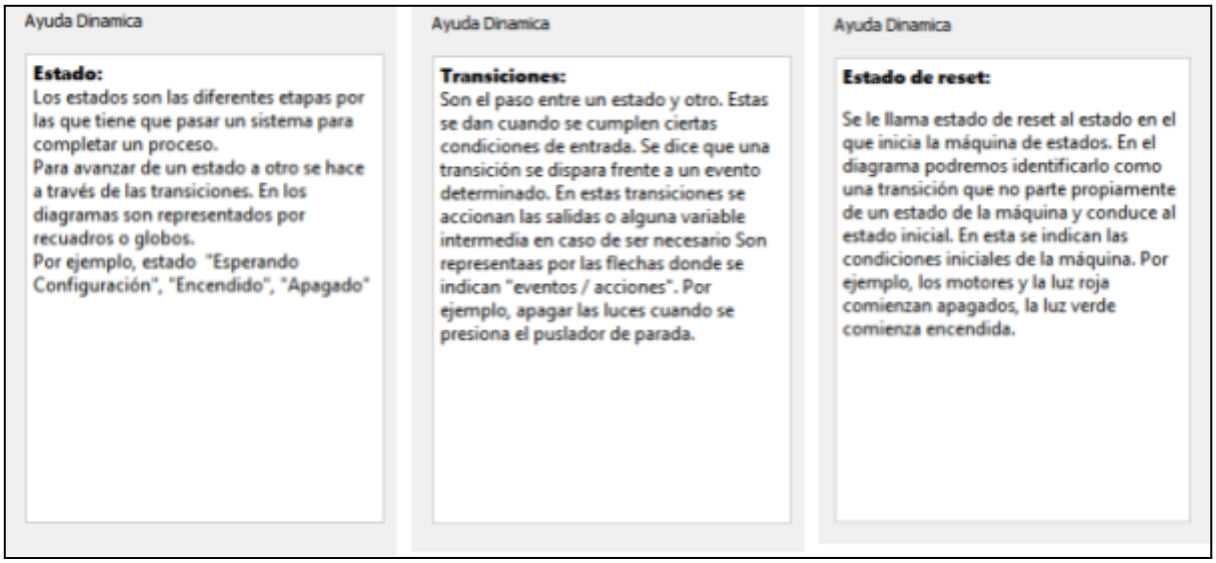

Fig. 3. Capturas de diferentes textos de ayuda. De izquierda a derecha: Estado, Transiciones y Estado de reset o estado inicial

A. *División cualitativa de la ayuda*

Desde el inicio se planteó la necesidad de poder alterar la cantidad de ayuda mostrada, con el principal objetivo de permitir un avance progresivo en términos educativos del alumno y permitir a usuarios intermedios o avanzados obtener una referencia rápida de conceptos clave que pudiesen ser olvidados.

Se confeccionaron tres categorías o niveles de visualización de ayuda nombrados de acuerdo a la clasificación del usuario: “Básico” (el nivel más abarcativo, útil para los primeros usos), “Medio” (reduce ciertos aspectos del nivel Básico, apropiado para usuarios intermedios) y “Avanzado” (provee una referencia rápida y limitada). Además, se incluye una opción para desactivar completamente el subsistema de ayuda, lo que brinda mejores posibilidades en cuanto a organización del entorno de trabajo.

Esta configuración podremos hacerla desde la barra de herramientas en la opción de ayuda (Fig. 4).

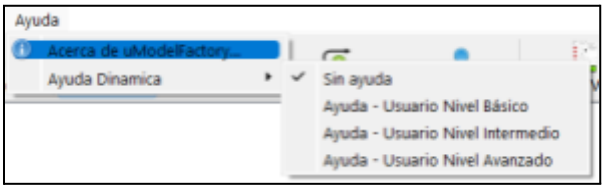

Fig. 4. Menú desplegable con opciones de ayuda dinámica

B. *Fácil traducción y/o alteración del contenido*

Con este punto se buscó favorecer el proceso de modificación y mejora de las recomendaciones mostradas por la ayuda dinámica, incluyéndose en el diseño la posibilidad de llevar a cabo una traducción del software a otros idiomas. En pro de ello, se dictaminó el uso de un archivo con sintaxis XML (eXtensible Markup Language) [4][5] para almacenar tanto los textos de ayuda como metadatos correspondientes a versionado y lenguaje en el que se encuentran los mismos.

Se decidió utilizar la presente estructura dada por la sintaxis XML por la facilidad de implementación en cuanto a código (la complejidad agregada por otros mecanismos de codificación opaca las ventajas de los mismos respecto a un mejor aprovechamiento de recursos como espacio de almacenamiento), la claridad intuitiva con la que permite organizar la información, y la facilidad en términos de edición (los archivos XML se agrupan dentro de los archivos de caracteres, donde la información se encuentra codificada en forma de texto y son legibles por cualquier persona sin la necesidad de utilizar programas especializados).

C. *Interfaz intuitiva y efectiva*

Luego de probar con varios diseños se optó por implementar la ayuda en un panel (una sección vertical presente dentro de la ventana del programa principal) pasible de ser movida y redimensionada por el usuario a gusto, apelando al confort del mismo y evitando frustraciones innecesarias por una interfaz no adaptable.

En función de la posición del mouse en el diagrama se presenta un mensaje de ayuda dinámica con información del objeto señalado por el mouse. En nuestro caso podremos encontrar el estado de reset, estados y transiciones. El mensaje cambiará en función del nivel de ayuda seleccionado.

D. *Devolución por parte de los alumnos y ex-alumnos*

Los resultados de las encuestas a los ex-alumnos y/o alumnos recursantes de la materia fueron que con la funcionalidad de ayuda dinámica podría haber sido mucho más fluida la etapa de aprendizaje.

Por otro lado, los alumnos que cursaron la asignatura con esta nueva versión de uModelFactory no manifestaron problemas o sugerencias de accesibilidad a la información en las encuestas.

V. Conclusiones y Trabajos a Futuro

Dado que el impacto de estas nuevas funcionalidades fueron positivos se llega a la conclusión de que cuanto mayor sea la cantidad de herramientas que el alumno tenga, mejor será su rendimiento.

En consecuencia, esto motiva a desarrollar nuevas herramientas que faciliten el proceso de aprendizaje. De hecho, actualmente, se está trabajando en un menú de ayuda que cuente con un buscador de tal manera que allí se pueda acceder fácilmente a una descripción de cada funcionalidad de la aplicación. Así como diferentes aspectos teóricos.

Al igual que en la ayuda dinámica se está implementando con un archivo XML para añadir la posibilidad de configurar el idioma en versiones posteriores, además de contar con un sistema de hipervínculos que puedan facilitar información acerca de herramientas que podrían llegar a ser útiles según la funcionalidad buscada.

Al igual que en las herramientas de ayuda ya desarrolladas, estas nuevas herramientas serán configurables para que se adapten en función de las necesidades del usuario y no estorben en caso de no ser requeridas.

Referencias

[1] L. Sugezky, M. Prieto, N. González, M. Giura, Y. Kuo, M. Trujillo, J.M. Cruz. "Desarrollo e implementación de herramientas de simulación de modelos para sistemas embebidos". Congreso Argentino de Sistemas Embebidos, 2016.

[2] 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.

[3] N. Gonzalez, J. Cruz, L. Sugezky, M. Giura, M. Trujillo, M. Prieto. "Analysis of a UML-based embedded system modeling software application". Congreso Argentino de Sistemas Embebidos, 2014.

[4] B. Schnabel. "Enabling Language Translation with XML Tools and Standards". The Center For Information-Development Management, 2015. Disponible en: http://www.oasis-open.org/committees/xliff/documents/translation_wxml_tools.pdf

[5] A. Zydron, (2004). "Translating XML Documents with xml" Disponible en: https://www.xml.com/pub/a/2004/01/07/xmltm.html