CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Generación de documentación automática para sistemas embebidos utilizando uModelFactory

Nicolas Tobías Almaraz[1]

[1] Universidad Tecnológica Nacional - Facultad Regional Buenos Aires, CABA 1179, ARG
nalmaraz@frba.utn.edu.ar

Abstract. 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. La aplicación tiene la finalidad de ser utilizada tanto en un ámbito industrial como en un ámbito educativo. De hecho, la cátedra de la asignatura Informática II decide implementarla como herramienta didáctica. En base a ello es que fue necesario la funcionalidad de documentar los proyectos desarrollados a partir de la aplicación. La idea fue que el usuario tenga la posibilidad de generar un documento editable con toda la información necesaria acerca del proyecto a partir de unos pocos clics. En otras palabras, que el usuario genere un archivo DOCX que cumpla con cierto template y contenga toda la información previamente configurada. La información a escribir implica todo lo referido al desarrollo del sistema embebido, por ejemplo autor, descripción del proyecto, máquinas de estados con sus respectivos diagramas y tablas, listas de variables, eventos y acciones, implementaciones en C, etc.

Palabras clave: máquinas de estado, documentación, software

1. Antecedentes de la propuesta

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.

En versiones anteriores la funcionalidad de documentar automáticamente estaba resuelta pero no generaba archivos editables. Pues, en un principio generaba una página web HTML en donde se presentaba la información. Dado que el objetivo era poder generar un template para posibles informes referidos al proyecto esta opción no terminaba de cumplir su objetivo, dado que los archivos HTML no son editables.

Es por ello que nace el desarrollo de una mejora en las funcionalidades de documentación. La propuesta fue generar un archivo DOCX o en su defecto ODT de tal modo que el usuario pueda, o bien, utilizarlo directamente tal como lo generó uModelFactory, o bien modificarlo en función de las necesidades del proyecto.

2. Diseño de la Propuesta

El objetivo del informe es documentar el proyecto, es por ello que incluye autor, fecha, descripción del proyecto, introducción teórica, máquinas de estados (cada una de ellas con su respectivo diagrama y tabla) y finalmente las implementaciones en C.

El usuario tiene la posibilidad de configurar la inclusión o no de algunos de los campos mencionados. Además, se tomó la decisión de conservar las funcionalidades de exportar como HTML para darle al usuario otra opción.

3. Resultados

Luego de desarrollar todo lo mencionado en la sección III, el resultado es un nuevo botón en la barra de herramientas (Fig. 1) que brinda la posibilidad de generar documentación.

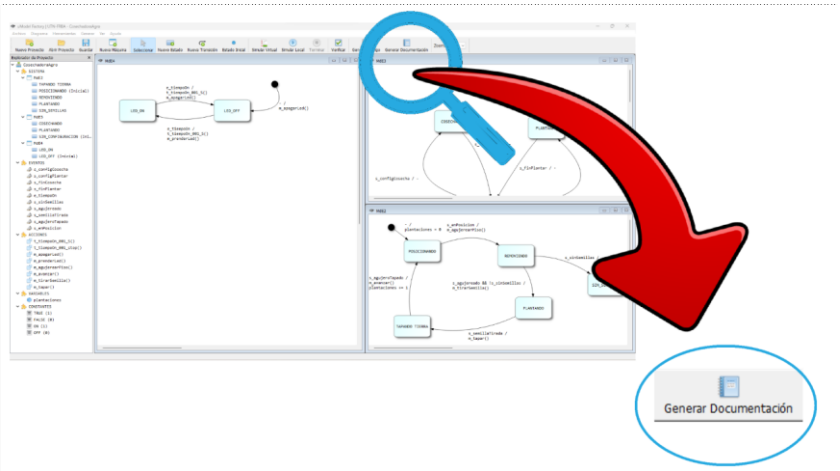

Fig 1. Acceso a funcionalidades desde la barra de tareas.

Este abrirá un menú de configuración en donde el usuario puede configurar qué secciones va a incluir en el informe y si desea exportarlo como DOCX o como HTML (Fig. 2).

Cabe destacar que algunos de los campos se configuran automáticamente, como por ejemplo el nombre del proyecto o el autor (configurado en la creación del proyecto).

Luego, una vez que se configura todo se presiona el botón “Ok” y si todo está bien se abrirá una ventana de confirmación (Fig. 3). En ella indicará la ruta en donde se exportó el archivo y nos facilitará un botón para acceder a dicho directorio.

Finalmente, se obtiene un informe como se detalla a continuación. Particularmente en el ejemplo que se va a desarrollar se configuró sin la introducción teórica y consta de un proyecto de una sola máquina de estados.

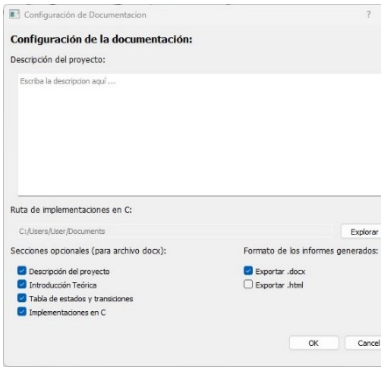

Fig 2. Interfaz de usuario encargada de configurar la documentación automática.

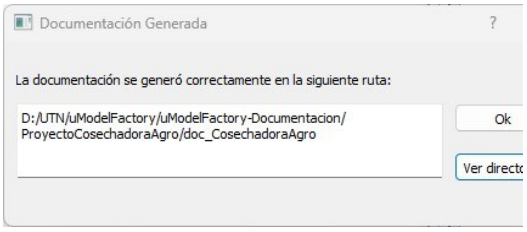

Fig 3. Ventana emergente de confirmación.

En primer lugar, en la Fig. 4 muestra la primera página del informe, en esta podemos ver el encabezado y pie de página con los datos de la universidad y el rótulo de uModelFactory, además de la fecha y datos del autor. También incluye una descripción del proyecto (configurada por el usuario en la Fig. 2) y las listas de acciones, eventos y variables que usa el sistema.

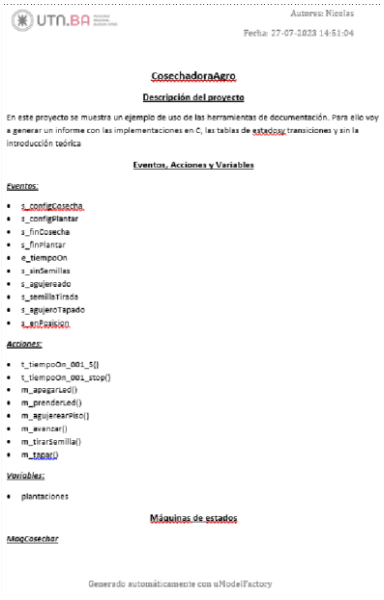

Fig 4. Primera página del informe generado.

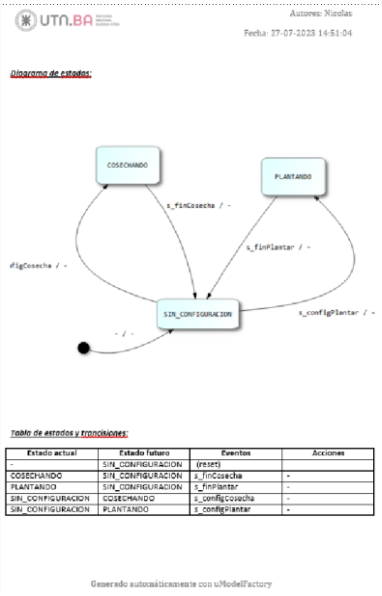

Fig 5. Segunda página del informe.

Por otro lado, la Fig. 5 muestra la segunda página del informe, en esta podemos ver la información referida a la máquina de estados que tiene el proyecto. Aquí se incluye el nombre de la máquina, diagrama y tabla de estados y transiciones. Es importante

aclarar que si el proyecto tuviese otras máquinas van a incluirse una a continuación de la otra hasta terminar de adjuntar la información de todas.

Finalmente, se incorpora una sección del informe que incluye las implementaciones en C configuradas por el usuario. Esta funcionalidad se encargará de cargar el contenido de todos los archivos que se encuentren en el directorio que configuró el usuario en la Fig 2. Cabe destacar que la aplicación tiene implementada la funcionalidad de generación de código automática. Sin embargo, al ser una ruta genérica admite códigos que no fueron exportados directamente desde uModelFactory.

4. Implementación

Todo el proyecto está desarrollado con el framework Qt. Sin embargo, no hay clases desarrolladas para la escritura de archivos DOCX directamente de Qt. Es por ello que la implementación de la escritura se realizó desde Python utilizando el módulo python -docx [4].

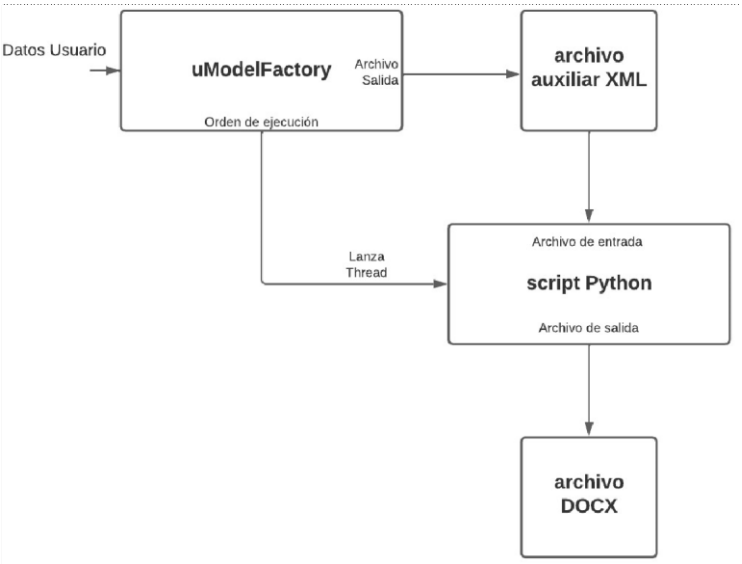

Fig 7. Esquema de funcionamiento.

La estrategia fue escribir un documento auxiliar XML desde la aplicación con toda la información necesaria y luego mediante un nuevo proceso llamar al script de Python para que lea dicho archivo auxiliar y escriba todo lo solicitado (Fig. 7).

Por otro lado, la implementación dentro del bloque uModelFactory se divide en tres niveles jerárquicos principales.

- En el de más alto nivel se arma el template del informe a realizar y recibe como argumentos el contenido del informe
- En el nivel intermedio se hacen métodos para poder escribir más fácilmente el

template, como por ejemplo definir el formato de los encabezados, títulos, subtítulos, etc.
- En el más bajo nivel se encuentran las funcionalidades más básicas como escribir texto con formato, imágenes, tablas, listas, encabezado y pie de página, etc.

5. Conclusiones

Con el desarrollo de este módulo del proyecto se cubren las necesidades planteadas. Es decir, contar con documentación generada a partir del modelo en forma automática, incorporando elementos y recursos de base como también opcionales, en función de la necesidad del usuario o del proyecto. Dicha documentación se encuentra en un formato editable y sirve de base para poder complementar la documentación con otros módulos que se desarrollen o de terceros. En este caso, el módulo de documentación fue también pensado como un aliado para los estudiantes de Informática II, que deben realizar un trabajo práctico integrador, no solo diseñando una solución para un sistema embebido sino también en el soporte para la continuidad de dicho proyecto. En un futuro se tendrá una devolución por parte de los usuarios, cohorte 2023, correspondiente a los estudiantes de la asignatura mencionada, perteneciente al segundo nivel de la carrera de Ingeniería Electrónica de la Universidad Tecnológica Nacional Facultad Regional Buenos Aires.

6. Trabajos a futuro

Como desarrollo a futuro está planificado una serie de mejoras en las funcionalidades. En particular poder seleccionar archivos fuente y elegir qué funciones principales vale la pena poder incluir y cuáles no serán necesarias. En ese sentido, el próximo desafío es poder interpretar el código de los archivos fuente seleccionados para poder permitirle al usuario incluir solamente algunas funciones o solamente algunos de los archivos que se encuentren en la ruta indicada.

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.
4. “Analysis of a UML-based embedded system modeling software application”. Congreso Argentino de Sistemas Embebidos, 2014.
5. python-docx, "python-docx documentation", Versión 0.8.11, python-docx, 2023. Disponible en: https://python-docx.readthedocs.io/en/latest/# . [Consultado el: 20/07/2023].