INTEGRACIÓN DE FACTORY IO Y 4DIAC-FORTE PARA LA VALIDACIÓN DE SOFTWARE DE CONTROL EN LA NORMA IEC 61499

Andrés Tendero Vegas, Oscar Miguel-Escrig, Julio-Ariel Romero-Pérez,
Departamento de Ingeniería de Sistemas y Diseño. Universitat Jaume I, España.
Bianca Wiesmayr, Virendra Ashiwal, Alois Zoitl
LIT Cyber-Physical Systems Lab Johannes Kepler University Linz, Austria

Resumen

En este trabajo se presenta la integración de 4diac-FORTE y Factory IO mediante OPC-UA. Dicha integración permite verificar software de control desarrollado según la norma IEC 61499 y de esta forma validar de metodologías para el desarrollo de dicho *software*. Se presenta un ejemplo sencillo que ilustra detalles importantes a tener en cuenta en la configuración adecuada del servidor OPC-UA que permite esta integración.

Palabras clave: IEC 61499, Factory IO, 4diac-FORTE.

1. INTRODUCCIÓN

La transformación digital de la industria plantea nuevos requerimientos a los sistemas de control automático, ya que de ellos depende en gran medida el desempeño de los procesos industriales. En este contexto, los sistemas de control adquieren un protagonismo aún mayor, pues el éxito de las estrategias de planificación global pasa inexorablemente por mantener una buena regulación sobre las variables de los procesos. Las demandas de este nuevo paradigma hacen necesario un replanteamiento de los sistemas de regulación y control tradicionales, tanto en su funcionamiento, como en la incorporación de nuevas tecnologías que permitan su integración en un entorno caracterizado por una alta conectividad, reconfiguración y manejo de grandes volúmenes de información. Los cambios productivos impuestos por los requerimientos de adaptabilidad deben ser asumidos en tiempos razonables y una vez alcanzados mantener un funcionamiento robusto.

En cuanto al desarrollo del software de control automático hay que decir que en la actualidad la mayor parte de los sistemas de automatización industrial se programan según el estándar IEC 61131, [5], ampliamente usado en la programación de Autómatas Programables Industriales (APIs). Sin embargo, en los últimos años se ha estado desarrollando el estándar IEC 61499, [4], más orientado a la programación de sistemas de control distribuidos y reconfigurables. Dicho estándar introduce conceptos novedosos respecto al IEC 61131, los cuales permiten el diseño de aplicaciones cuya ejecución puede ser distribuida entre distintos dispositivos de control los cuales no necesariamente deben ser del mismo fabricante o modelo, lo cual mejora de forma considerable la interoperabilidad del software de control.

Otra de las diferencias respecto al IEC 61131 es que el nuevo estándar cuenta con mecanismos para la modificación de las aplicaciones sin necesidad de detener su ejecución, lo cual facilita la reconfiguración de los sistemas de control. Las características antes mencionadas hacen que el IEC 61499 esté siendo actualmente el centro de atención de numerosos trabajos para su uso en diferentes problemas de automatización y control industrial. En los últimos años numerosos trabajo de investigación abordan la implementación de algoritmos de control usando dicho estándar, ver por ejemplo [8, 10, 9, 6, 2].

Una revisión actualizada del estado del arte del estándar IEC 61499 se puede consultar en [7]. Muestra del interés por el nuevo estándar entre los desarrolladores de equipos de control es el hecho de que ya existen entornos de programación comerciales que lo soportan como Isagraf de Rockwell Automation, nxtSTUDIO de nxtCONTROL y EcoStruxure Automation Expert de Schneider Electric. Además, la reciente creación de la organización sin ánimo de lucro UniversalAutomation (UniversalAutomation.org), cuyo objetivo es favorecer el desarrollo de sistemas de programación para la automatización basado en la norma IEC 61499, a la cual se han unido importantes empresas, es un indicativo claro del interés que está despertando este estándar.

En este trabajo se describe como realizar la integración entre 4diac-FORTE y Factory IO, lo cual permite probar software de control desarrollado según la norma IEC 61499. Dicha integración abre la posibilidad de validar metodologías para el desarrollo de *software* de control bajo dicha norma, para facilitar la aceptación de la misma en el entorno industrial. La estructura del trabajo es como sigue. En la sección 2 se describen los conceptos

https://doi.org/10.17979/spudc.9788497498418.0949

básicos de la norma IEC 61499. Después se realiza un breve descripción de Factory IO y 4diac-FORTE. En la sección 4 se presentan las ideas fundamentales para la integración entre Factory IO y 4diac-FORTE y se ilustran dicha integración mediante un ejemplo. Finalmente, en la sección 5 se resumen las principales conclusiones del trabajo.

2. IEC 61499. BREVE DESCRIPCIÓN.

La primera versión del estándar IEC 61499 fue publicada en 2005 y la segunda versión en 2012, en ella se define una arquitectura y unos modelos de software para el desarrollo de aplicaciones reconfigurables de control distribuido. La arquitectura está organizada jerárquicamente mediante los modelos de sistema, dispositivo y recurso y sobre estos los modelos de Aplicación y Bloque de Función, siendo este último una extensión de los bloques de funciones definidos en la norma IEC 61131.

El estándar se caracteriza por una arquitectura que se compone fundamentalmente por un conjunto de modelos de referencia que representan los principales elementos y reglas que intervienen en un sistema de control distribuido para que este se pueda llevar a cabo correctamente su ejecución. En el centro de la norma se encuentra el concepto de bloque de funciones (se nombrará por las siglas FB del inglés: *Function Block*), los cuales se construyen mediante el modelado o modelo de bloques de funciones (se nombrará por las siglas FBM del inglés: *Function Block Model*), siendo los FB la unidad funcional de software de menor nivel, con su propia estructura de datos que puede ser manipulada por uno o más algoritmos.

En el FB de la norma IEC 61499 confluyen dos tipos de parámetros, los eventos y los datos, como se muestra en la Figura 1. La ejecución de un FB viene regulada por la recepción de eventos, pero también dependen de los datos de entrada, de salida y de las variables internas del propio bloque. Un FB puede tanto recibir como enviar ambos tipos de parámetros, y en concreto, la recepción o envío de un evento se corresponde con el refresco de los datos de entrada o salida asociadas con dicho evento.

En cuanto al comportamiento e interfaz con el resto de elementos de la norma existen tres tipos de FB:

- Los bloques de funciones básicos (se nombrará por las siglas BFB del inglés: *Basic Function Block*). Los BFB se caracterizan por tener una estructura y comportamiento están definidos por medio de un gráfico de control de ejecución (se nombrará por sus siglas ECC del inglés: *Execution Control Chart*).

- Los bloques de funciones compuestos (se nombrará por sus siglas CFB del inglés: *Composite Function Block*). Los CFB se caracterizan por tener su estructura interna compuesta por instancias de otros FB y sus respectivas conexiones.

- Los bloques de funciones de interfaz de servicio (se nombrará por sus siglas SIFB del inglés: *Service Interface Function Block*). Los SIFB proveen una interfaz entre la norma y servicios externos como puede ser una comunicación entre FB en diferentes dispositivos (i.e. gestión de la comunicación según el protocolo entre dispositivos).

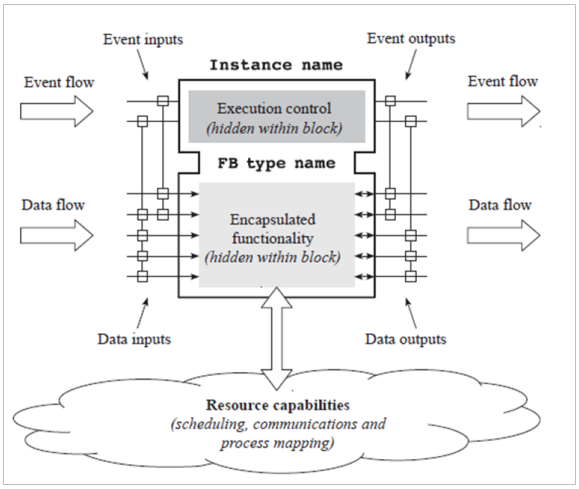

Figura 1: Esquema del un FB.

Con respecto a los BFB, se ha mencionado que se caracterizan por tener una estructura interna caracterizada por un ECC. Un ECC es un gráfico que regula la ejecución y el comportamiento del FB. Los ECC se componen de varios estados, en los que se puede tanto enviar eventos de salida como ejecutar algoritmos, y a los que se llega por medio de unas transiciones, las cuales se franquean mediante distintas condiciones, las cuales pueden ser simplemente la recepción de eventos o condiciones lógicas con datos. En la Figura 2 podemos ver un ECC con las partes que hemos descrito, en azul los estados y las transiciones, en amarillo el algoritmo a ejecutar y en verde los eventos a enviar.

Los FB se agrupan en un modelo que configura una aplicación, el modelo de aplicación (a partir de ahora se nombrará por sus siglas AM del inglés:

https://doi.org/10.17979/spudc.9788497498418.0949

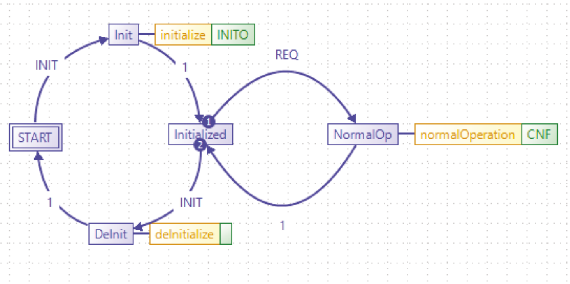

Figura 2: Ejemplo de ECC.

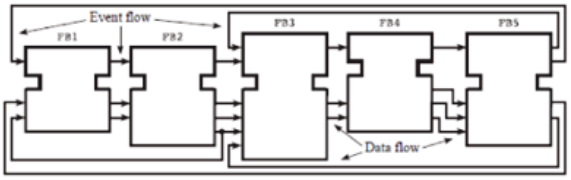

Figura 3: Modelo de aplicación.

Application Model). Una aplicación queda determinada y definida en términos de comportamiento y estructura por los FB que la componen y de la interconexión de eventos y datos de los mismos, siendo esta configuración la que dota de funcionalidad total al programa sin necesitar de variables globales o locales fuera de los FB (a excepción de los bloques que ya hemos explicado anteriormente) ya que el comportamiento del algoritmo debe regularse íntegramente dentro de estos. En la Figura 3 podemos ver un ejemplo de aplicación por diferentes FB.

En las aplicaciones de la norma IEC 61499 los eventos son generalmente conexiones de punto a punto, pero en algunos casos se permiten conexiones de una salida de evento a varias entradas, aunque esta práctica no es deseable y debería ser evitada siempre que fuera posible. Sin embargo, para los datos sí que está permitido conectar un dato de salida a varios de entrada, no siendo esto posible al contrario, es decir, no se puede conectar diferentes datos de distintas salidas a un dato de entrada ya que el FB receptor no dispondrá de la información del origen del dato y no podrá tratar correctamente la información. Cabe destacar que los datos y los eventos pueden tener su origen en diferentes FB pero confluir en otro.

La norma IEC 61499 presenta como ventaja respecto a la norma IEC 61131 justamente que se centra en el AM y no en los recursos que posee el sistema, lo cual repercute en la posibilidad de poder evaluar el funcionamiento de una aplicación en etapas tempranas del proceso de desarrollo de la aplicación. Los AM en esta norma pueden desarrollarse en uno o varios dispositivos o recursos que pueden estar conectados entre ellos según una serie de protocolos o redes. El modelo que describe y conecta los dispositivos es el modelo del sistema (a partir de ahora se nombrará por sus siglas SM del inglés: *System Model*). En este modelo se presentan los diferentes dispositivos así como las diferentes redes que pueden interconectarlos, asignando además algunas propiedades concretas en cuanto a la relación del dispositivo con la red (i.e. Asignación de IP's a un dispositivo conectado vía Ethernet). En la Figura 4 vemos un esquema de un SM.

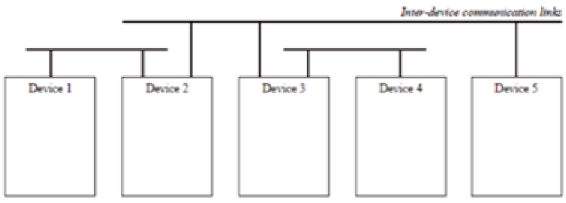

Figura 4: Modelo de sistema.

Además, cada dispositivo está modelado por una estructura interna llamada modelo del dispositivo (traducción del original *Device Model*) que le permite ejecutar las redes de FB de la norma. El objetivo de este modelo es proveer de una infraestructura al dispositivo para soportar uno o más recursos. Los recursos de la norma se definen de forma similar a los de la norma IEC 61131, ya que estos permiten la ejecución de los FB. Sin embargo, a diferencia de la norma anterior, los recursos del nuevo estándar no están ligados necesariamente a una unidad de ejecución, es decir, un recurso es para esta norma una separación lógica dentro de un dispositivo y cada uno proporciona una ejecución y un control de las redes de FB independiente.

Como ya hemos comentado anteriormente, uno de los objetivos fundamentales de la norma es facilitar la implementación de aplicaciones distribuidas. Para este fin la norma cuenta con el modelo de distribución, el cual conecta la aplicación o partes de la aplicación (modelada en el AM), la cual a priori es independiente del sistema o sistemas que la ejecutan, con el modelo del sistema (SM). Este modelado trata, por tanto, de asignar partes de la aplicación a diferentes dispositivos y recursos y de configurar los dispositivos que ejecutan la aplicación. Según la norma IEC 61499 las aplicaciones distribuidas deben distribuirse a nivel de la unidad fundamental de programación de la norma, es decir, el FB. De lo cual se desprende, que una aplicación distribuida para la norma consiste en una red de FB los cuales funcionan en diferentes dispositivos. En la Figura 5 podemos ver un ejemplo de un modelo de distribución.

https://doi.org/10.17979/spudc.9788497498418.0949

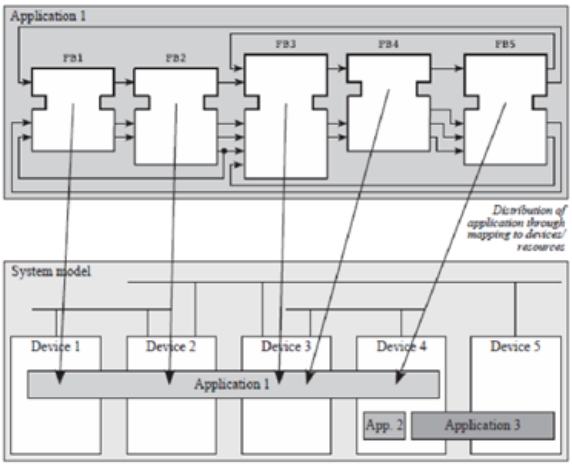

Figura 5: Distribución de una aplicación.

Una particularidad del SM y del modelo de distribución del estándar es que permiten la ejecución de más de una aplicación en cada dispositivo, pudiendo además, asignar más de una aplicación a un mismo recurso. Para ello, en la concepción del modelo del dispositivo, el cual no vamos a detallar porque resultaría excesivo para una explicación introductoria de la norma, se ha previsto que se pueda cargar y borrar las aplicaciones de un dispositivo sin perturbar el funcionamiento de las otras aplicaciones en curso. Hay que tener en cuenta para el desarrollo de aplicaciones distribuidas que se debe prestar atención a las características del recurso y del dispositivo, ya que el rendimiento de la aplicación dependerá del recurso donde esta haya sido mapeada y de las comunicaciones entre los recursos.

En las aplicaciones distribuidas, las conexiones entre FB situados en diferentes recursos se puede decir que se cortan y que además no se pueden añadir FB en el AM que proporcionen una configuración específica de la comunicación entre dispositivos porque el AM tiene que ser independiente de los sistemas que lo ejecutan. Para solventar estos problemas, la norma permite añadir FB de configuraciones específicas de los sistemas dentro de los recursos, de esta forma la aplicación se mantiene independiente de los sistemas y se asegura la funcionalidad de la aplicación.

Para finalizar, se debe tratar uno de los objetivos principales en la concepción del estándar IEC 61499: facilitar el desarrollo de aplicaciones reconfigurables. Se entiende por reconfigurabilidad a la capacidad de un algoritmo de cambiar su comportamiento, estructura y/o los datos que maneja, sin necesidad de la intervención del programador, es decir, hacer este cambio dinámicamente. En este punto se debe hacer una distinción entre lo que se considera programación reconfigurable y paramétrica, la clave consiste en que en la programación paramétrica todo está programado antes de la ejecución y esta depende de unos parámetros, por el contrario, en la programación reconfigurable lo que se programa son los mecanismos que permiten al programa auto-reprogramarse a medida que se desarrolla la ejecución. Para las aplicaciones reconfigurables, se proveen una serie de FB que permiten gestionar la instanciación, destrucción e interconexión de otros FB y los datos que manejan, iniciar la ejecución de FB como parte de aplicaciones distribuidas (i.e. Activar partes de la aplicación que normalmente no se activan), cambiar el estado en el que se encuentran los FB y proveer servicios a demandas por parte de las redes de comunicación.

3. FACTORY IO

Factory IO está diseñado específicamente para la formación en la programación de sistemas de automatización y en concreto de los Autómatas Programables Industriales (API). Se pueden crear modelos 3D de plantas industriales incluyendo toda la sensorización y realizar la programación de los API para la automatización de dichas plantas. Para ello, Factory IO se integra con *softwares* de programación de API como por ejemplo Codesys o Unity-Pro. Además, soporta otros métodos de intercambio de información como OPC-UA o MODBUS que permiten su integración con otras herramientas para el desarrollo de *software* de control.

En la actualidad, la validación de metodologías de desarrollo de software de control se realizan usando sistemas sencillos desarrollados con fines docentes o de investigación en las universidades. Dichos sistemas no consideran la complejidad de los procesos industriales reales. Factory IO permite plantear problemas de automatización industrial complejos con un coste mínimo al ser desarrollados en un entorno virtual. Para ello consta de una gran variedad de elementos que pueden ser dispuesto de forma conveniente para formar sistemas de diferentes tipos que pueden incluir sensores y actuadores digitales o analógicos. Cada sistema creado en Factory IO recibe el nombre genérico de escena.

Todo lo anterior, unido a la simulación realista que el programa realiza de todos sus elementos que forman una escena, de manera similar a un vídeojuego, hacen que Factory IO una herramienta ideal para la validación de metodologías para el desarrollo de *software* de control.

https://doi.org/10.17979/spudc.9788497498418.0949

3.1. 4DIAC-FORTE

4diac [3] es un entorno de desarrollo *opensource* extensible a través de plugins, basado en el reconocido entorno de desarrollo Eclipse. La aplicación es muy robusta y muy intuitiva, tiene soporte oficial y saca dos actualizaciones del entorno por año. Permite la creación de bloques básicos, sistemas aplicaciones y el resto de modelos. Además, permite distribuir muy fácilmente las aplicaciones entre diferentes recursos de forma gráfica. Cuenta con la funcionalidad de *test* y *debug* que permite fijar y leer los valores de las variables de forma remota. Se ha demostrado su portabilidad con nxtStudio y con FBDK.

FORTE es el *runtime* de ejecución para aplicaciones IEC 61499 desarrolladas en 4DIAC. FORTE está implementada en C++ y es de código abierto. Esta especialmente desarrollado para ejecutarse en sistemas embebidos de tiempo real, ya que tiene los mecanismos necesarios para poder asegurar la ejecución en tiempo real y la gestión de prioridades. Otro de los puntos fuertes de FORTE es que ha sido diseñada para ser independiente de la plataforma sobre la que se ejecute, para poder ser ejecutada en diferentes hardwares y sistemas operativos. En la actualidad FORTE se ha llevado a diferentes sistemas operativos como Linux, windows, threadX y eCos. FORTE se puede configurar con 4diac pero también es compatible con otros entornos como FBDK o nxtOne.

4. INTEGRACIÓN de 4DIAC-FORTE y Factory IO

La integración entre una aplicación IEC 61499 que se ejecuta en uno o varios *runtimes* FORTE y una escena de Factory IO que simula el sistema a controlar se puede realizar de forma relativamente sencilla mediante OPC-UA. Factory IO tiene la opción de comunicarse con un servidor OPC-UA. Por otro lado, es posible definir un servidor OPC-UA en FORTE usando los bloques de funciones PUBLISH y SUBSCRIBE de 4DIAC. Por ello, la comunicación mediante OPC-UA es la forma más simple de intercambiar datos entre ambas aplicaciones.

El esquema general se muestra en la Figura 6. Se puede ver que el servidor OPC-UA se ejecuta como parte del FORTE. En la aplicación IEC 61499 los bloques de funciones PUBLISH y SUBSCRIBE se usan para escribir y leer variables del servidor desde la aplicación.

Aunque en este trabajo nos centramos en el caso de aplicaciones centralizadas, cabe destacar que también es posible la integración mediante OPC-UA en el caso de aplicaciones IEC 61499 distribuidas en varios *runtimes* de FORTE.

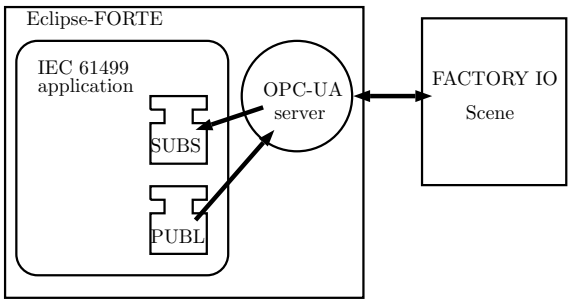

Figura 6: Esquema de integración de FORTE y Factory IO mediante servidor OPC-UA.

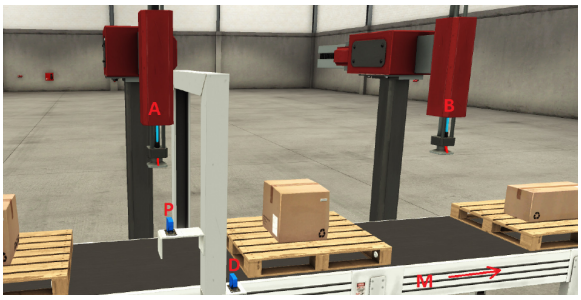

Figura 7: Escena de Factory IO del sistema.

4.1. EJEMPLO

En esta sección mostramos un ejemplo sencillo para ilustrar el uso de estas herramientas.

Considérese el sistema de la Figura 7. Por una cinta accionada por el motor M circulan palets espaciados regularmente, de forma que cuando hay un palet en A, hay otro en B. Sobre estos palets puede haber dos tipos de cajas: altas y bajas. En las cajas altas hay que realizar las operaciones A y B (podrían ser marcado, etiquetado, etc). El detector inductivo D detecta el paso de los palets, mientras que la fotocélula P detecta la presencia de caja alta sobre el palet. Las operaciones A y B duran 0.2 y 0.5 segundos respectivamente. Se dispone de un pulsador de MARCHA/PARADA para iniciar o detener el funcionamiento del sistema.

La Figura 8 muestra la aplicación IEC 61499 para el control del sistema. En ella se puede apreciar el uso de los tipos de bloques de función `SUBSCRIBE_3` (instancia `SUBSCRIBE_3_OPC`) y `PUBLISH_3` (instancia `PUBLISH_3_OPC`) para la lectura y escritura de variables en el servidor OPC-UA. El resto de bloque de función conforman el algoritmo de control secuencial que define el comportamiento del sistema.

https://doi.org/10.17979/spudc.9788497498418.0949

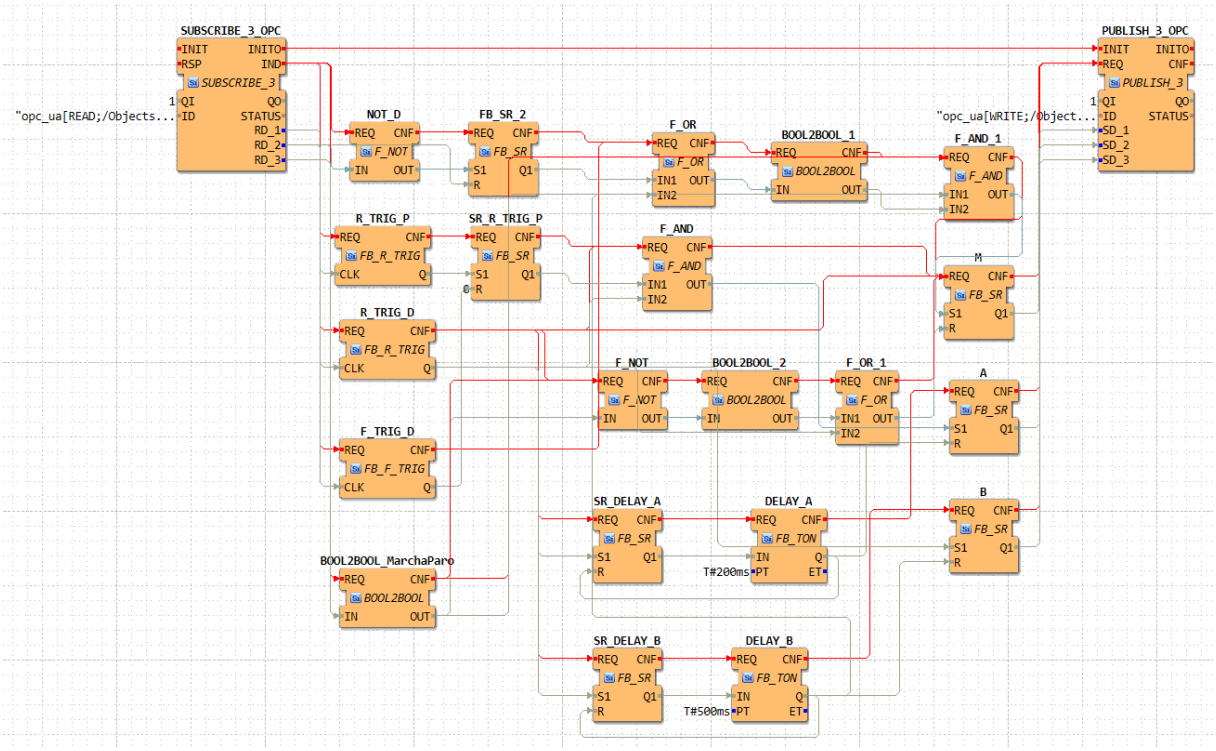

Figura 8: Aplicación IEC 61499 desarrollada en 4DIAC para el control de la escena de la Figura 7.

El parámetro de entrada ID de los bloques de función `SUBSCRIBE_3_OPC` y `PUBLISH_3_OPC` define la forma en que las variables van a ser creadas en el servidor y la manera en que la aplicación de FORTE accederá a las mismas. Para este ejemplo el parámetro ID del FB `SUBSCRIBE_3_OPC` es:

```
"opc_ua[READ;
/Objects/fio_P,1:s=fio_P;
/Objects/fio_D,1:s=fio_D;
/Objects/fio_MARCHA_PARO,1:s=fio_MARCHA_PARO]"
```

Este parámetro indica que se realizará la lectura de las variables `fio_P`, `fio_D` y `fio_MARCHA_PARO`. Estas variables se crearán en la carpeta `Objects` del servidor. Los resultados de las variables son devueltos por el las salidas RD en el mismo orden en que aparecen en parámetro `ID`, en este caso `fio_P`, `fio_D` y `fio_MARCHA_PARO` estarán disponibles en las salidas `RD_1`, `RD_2` y `RD_3` respectivamente.

Por otro lado, el parámetro ID del bloque `PUBLISH_3_OPC` es:

```
"opc_ua[WRITE;
/Objects/fio_M,1:s=fio_M;
/Objects/fio_A,1:s=fio_A;
/Objects/fio_B,1:s=fio_B]"
```

Este parámetro indica que se realizará la escritura de las variables `fio_M`, `fio_A` y `fio_B`. Estas variables se crearán en la carpeta `Objects` del servidor y el valor que será escrito en cada una se especifica por las entradas `SD` en el mismo orden en que las variables aparecen definidas en la entrada `ID`, en este caso, las entradas `SD_1`, `SD_2` y `SD_3` reciben los valores de las variables `fio_M`, `fio_A` y `fio_B` respectivamente. Más detalles de la configuración de servidores OPC-UA en FORTE pueden ser consultados en [1].

En cuanto a la configuración de la conexión del Factory IO con el servidor OPC-UA, esta se realiza mediante una interfaz gráfica, tal como se muestra en la Figura 9. El trabajo con dicha interfaz se describe de forma detallada en la documentación de ayuda que está disponible en la web de este *software*.

5. CONCLUSIONES

En este trabajo se ha presentado la integración de Factory IO y 4diac-FORTE mediante OPC-UA. Dicha integración permite realizar la verificación de software de control desarrollado según el estándar IEC 61499. Con ello se abre la posibilidad de validar nuevas metodologías para el diseño de aplicaciones en dicho estándar. Además, puede ser una valiosa herramienta en la enseñanza para favorecer la introducción del IEC 61499 en el entorno industrial.

https://doi.org/10.17979/spudc.9788497498418.0949

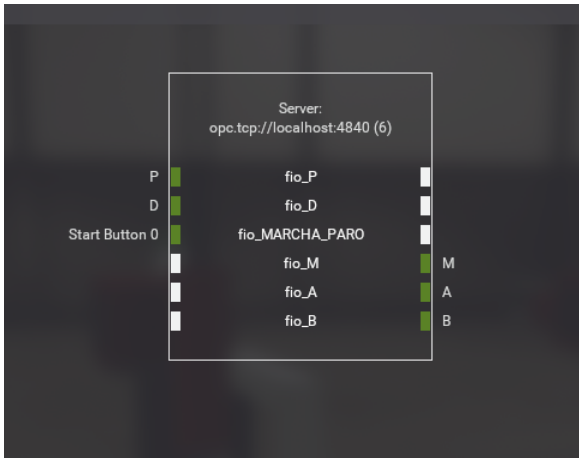

Figura 9: Configuración en Factory IO de la conexión con el servidor OPC-UA que se ejecuta en FORTE.

Agradecimientos

Este trabajo ha sido financiado por la Universitat Jaume I mediante el proyecto de investigación UJI-B2021-45.

English summary

INTEGRATION OF FACTORY IO AND 4DIAC-FORTE FOR THE VALIDATION OF CONTROL SOFTWARE IN THE IEC 61499 STANDARD

Abstract

In this paper, the integration of 4diac-FORTE and Factory IO through OPC-UA is presented. This integration allows verifying control software developed according to the IEC 61499 standard and opens the possibility of validating methodologies for the development of control applications under that standard. A simple example is presented that allows illustrating important details to take into account in the proper configuration of the OPC-UA server that allows this integration.

Keywords: IEC 61499, Factory IO, 4diac-FORTE.

Referencias

[1] OPC UA with IEC 61499 Tutorial, 2022.

[2] Yue Cao, Yusheng Liu, Bo Huang, Ganming Huang, and Xiaoping Ye. Transformation from system design models in sysml to executable iec 61499 function block models. In *2021 6th International Conference on Control, Robotics and Cybernetics (CRC)*, pages 200–206, 2021.

[3] Eclipse 4diac. Eclipse 4diac - the open source environment for distributed industrial automation and control systems, 2020.

[4] IEC. IEC 61499: Function blocks for industrial-process measurement and control systems, 2012.

[5] IEC. IEC 61131 - programmable controllers, part 3: Programming languages, 2013.

[6] Pranay Jhunjhunwala and Valeriy Vyatkin. Proposing and prototyping an extension to the adapter concept in the iec 61499 standard. *IEEE Access*, 10:2564–2577, 2022.

[7] Guolin Lyu and Robert William Brennan. Towards iec 61499-based distributed intelligent automation: A literature review. *IEEE Transactions on Industrial Informatics*, 17(4):2295–2306, 2021.

[8] Oscar Miguel-Escrig, Julio-Ariel Romero-Pérez, Bianca Wiesmayr, and Alois Zoitl. Distributed implementation of Grafcets through IEC 61499. In *2020 25th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2020.

[9] Lisa Sonnleithner and Alois Zoitl. A software measure for iec 61499 basic function blocks. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 997–1000, 2020.

[10] Bianca Wiesmayr, Alois Zoitl, Oscar Miguel-Escrig, and Julio-Ariel Romero-Perez. Distributed implementation of hierarchical grafcets through iec 61499. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2021.