Integración de ROS y Tecnomatix para el desarrollo de gemelos digitales en sistemas de manufactura flexible

Carolina Saavedra Sueldo[1], Sebastián A. Villar[1], Mariano De Paula[1,2], Silvia B. Urrutia[2], Gerardo G. Acosta[1]

[1] INTELYMEC, Centro de Investigaciones en Física e Ingeniería del Centro CIFICEN – UNICEN – CICpBA – CONICET, Argentina, Avenida del Valle 5737, B7400JWI, Olavarría, Argentina
[2] Departamento de Ingeniería Industrial, Facultad de Ingeniería, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina, Avenida del Valle 5737, B7400JWI, Olavarría, Argentina

{carolina.saavedra, svillar, mariano.depaula, surrutia, ggacosta}@fio.unicen.edu.ar

Resumen. Los gemelos digitales emplean la simulación en conjunto con una variedad de datos provenientes de diferentes equipos y sistemas físicos de planta, para mantener actualizados continuamente sus modelos digitales del mundo en un esquema de retroalimentación en un entorno virtual que facilita la toma de decisiones. La heterogeneidad de hardware y software existente requiere del desarrollo de arquitecturas de software para que los diversos componentes se integren e interactúen mediante el intercambio de información. En este trabajo se propone el diseño y construcción de una arquitectura de software que integra un simulador de procesos de manufactura con el sistema operativo de robots (ROS-*Robot Operating System*). La propuesta se prueba con el simulador *Tecnomatix* de *Siemens*® y la distribución libre ROS *Melodic*. Se presenta una instancia de la arquitectura de software para un caso de estudio complejo típico de plantas de manufactura y se demuestra su fácil integración con un sistema autónomo de toma de decisiones.

Palabras Claves: *Gemelo Digital, Industria 4.0, ROS, Tecnomatix*

1 Introducción

Debido al crecimiento exponencial de la tecnología, los hábitos de consumo globales manifiestan una tendencia cambiante que exige un ritmo cada vez más rápido de adaptación de las industrias de manufactura para poder atender una demanda variable que requiere multiplicidad de productos para ser satisfecha y una producción más personalizada. Esto lleva a los sistemas productivos a buscar una alta flexibilidad y capacidad de reconfiguración, por ejemplo utilizando recursos y/o servicios rentados [1]. En contrapartida, estos sistemas son cada vez más complejos, ya que generalmente reúnen otros sub-sistemas automatizados y/o robóticos para la realización de múltiples tareas a lo largo de todo el proceso productivo [2]. Por lo tanto, en esta nueva

configuración de sistemas de producción existen inevitablemente interferencias dinámicas asociadas con diversas fuentes, como por ejemplo la aleatoriedad de los pedidos de los clientes, la dinámica de la demanda de producción, la baja repetitividad del proceso productivo, el uso de servicios y recursos externos, etc. que hacen que la producción sincronizada pueda verse perturbada en detrimento de la efectividad de respuesta del sistema productivo [3].

Para analizar el comportamiento y el desempeño de los sistemas de manufactura, se ha adoptado la simulación como base para representar sistemas complejos a través de modelos simples y fáciles de comprender [4]. Una variedad de modelos de simulación han sido construidos como base para la toma de decisiones en la industria de manufactura [5]. Así, la simulación se convirtió en una de las herramientas utilizadas, por excelencia, para analizar el impacto de las decisiones tomadas sin afectar la operación del sistema real [6].

En las configuraciones actuales de los sistemas productivos intervienen una variedad de equipos, recursos de software y hardware y, también, recursos humanos que deben actuar coordinadamente y de forma autónoma para hacer frente a la variación y alta frecuencia de las tareas de producción causadas por los requisitos personalizados de los clientes [7]. En este contexto, el paradigma de simulación clásico (dinámica y de eventos discretos) ha dejado de ser el componente central para la formulación de sistemas autónomos de toma de decisiones debido a su característica estática. Por ello, y dada la fuerte irrupción de internet industrial de las cosas junto a los sistemas ciberfísicos [8], su lugar ha sido ocupado por el concepto de gemelo digital (GD) [9]. Sin embargo, los métodos y técnicas de simulación de sistemas tienen un papel preponderante para la abstracción y representación de los sistemas productivos [10]. En este sentido, los GDs emplean la simulación en un esquema de intercambio de información con una variedad de datos provenientes de los diferentes equipos y sistemas físicos, para mantener actualizados continuamente los modelos digitales con el fin de reflejar cualquier cambio que ocurra en los equivalentes físicos en el tiempo[11]. De esta manera, se puede lograr un esquema de retroalimentación en un entorno virtual adaptativo que facilita la toma decisiones apoyándose en estos GDs.

Hasta ahora, la integración entre diferentes tecnologías para el desarrollo de los GDs es una cuestión que ha sido abordada de una forma heterogénea y algunos trabajos han puesto el foco en organizar algunos conceptos claves [11], [12], cuestión que es de vital relevancia para la formulación de propuestas que empleen el uso de gemelos digitales, como por ejemplo en el desarrollo de sistemas de manufactura flexible [13]. Recientemente en el trabajo [14], se formuló una propuesta de arquitectura modular para implementar GDs, probándolo en un clasificador (*sorter*) pequeño y analizando los tiempos de la comunicación entre partes; y en el trabajo [15] realizan una propuesta de integración entre un simulador de eventos discretos y un software de dinámica de sistemas para resolver un caso de un problema de control. En este trabajo, se presenta una arquitectura software para el desarrollo de GDs mediante la integración del bien conocido sistema operativo de robots (ROS) y un simulador de procesos de manufactura flexible, el *Tecnomatix Plant Simulation – Siemens*® (TPS).

2 Metodología

En esta sección introducimos brevemente los principales componentes metodológicos hacia la formulación de un GD para un sistema de manufactura flexible. Para más detalle, el lector podrá referirse a las citas oportunamente indicadas.

2.1 Simulación

La metodología de simulación se ha transformado en una herramienta de análisis indispensable para evaluar el impacto de cambios sin que sea necesario modificar el sistema real. Particularmente, los sistemas de producción manufacturera son un caso especial de sistemas dinámicos híbridos, conocidos como sistemas de eventos discretos [14]. En este sentido, pueden ser modelados y simulados bajo el paradigma de simulación de eventos discretos [6]. La simulación de eventos discretos es una herramienta de análisis ampliamente difundida en el ambiente empresarial y ha comprobado su utilidad para apoyar la toma de decisiones relacionadas con la planeación de la producción e inventarios así como con el diseño de los sistemas de producción y sus cadenas de suministro [5].

Los modelos de simulación representan procesos y situaciones complejas de la realidad en función de los eventos que puedan ocurrir, por lo que pueden representar, tanto situaciones típicas de un proceso productivo (rotura de inventarios, fallas de equipos, aparición de imponderables), como así también estimar un determinado nivel de utilización de recursos relacionados con dichas situaciones. Asimismo, contempla las probabilidades de que ocurran cada uno de los eventos según se asocian con las características de cada uno de los elementos que intervienen. En su conjunto, todos estos elementos simulan la evolución natural del proceso analizado. De esta forma los modelos permiten conceptualizar sobre un proceso productivo y su gestión en términos de los eventos que puedan suceder y cuyo impacto afecta tanto a los clientes como a otros componentes del sistema (p. ej., el uso de recursos). En definitiva, la simulación de eventos discretos es una técnica que se puede utilizar para estudiar los sistemas y procesos cuyos estados van cambiando con el tiempo de forma discreta.

2.2 Gemelos digitales

La irrupción de los sistemas ciberfísicos ha cambiado paradigmáticamente la forma en que se conciben los sistemas industriales, permitiendo la integración de todas las entidades que intervienen en un sistema productivo [15]. Así, es posible lograr la integración y la interacción en tiempo real entre los diferentes componentes de un sistema productivo lo que hace posible gestionar las entidades físicas de manera confiable, segura, colaborativa, robusta y eficiente [16]. Desde esta perspectiva, los sistemas de fabricación integrados podrían pensarse compuestos por dos partes principales: una parte física y una parte digital. Como se muestra en la Fig. 1, la parte física comprende todas las entidades y recursos empleados para la fabricación mientras que la parte digital contiene aplicaciones y servicios ubicuos, incorporando subsistemas con capa-

cidades para el procesamiento y gestión de datos, simulación y análisis junto a sistemas inteligentes para la toma de decisiones.

En la parte física, representada esquemáticamente en la parte inferior de la Fig. 1, se encuentran interconectadas las diferentes entidades involucradas en el proceso de producción y, en una clasificación general, pueden ser agrupadas como Equipos, Materiales, Operarios y Herramientas. Todas estas entidades pueden compartir información y mantener actualizado su estado en tiempo real gracias a las diferentes tecnologías actuales como, por ejemplo, los protocolos Ethernet, ROS, sistemas de georeferenciación, etiquetas/lectores RFID, etc. De esta forma, en la parte física no sólo se recopilan e intercambian datos, sino que también se comunica con la parte digital donde se generan decisiones o acciones que posteriormente se ejecutan en la parte física. Es decir, mediante esta comunicación, la parte digital puede afectar los procesos físicos y viceversa.

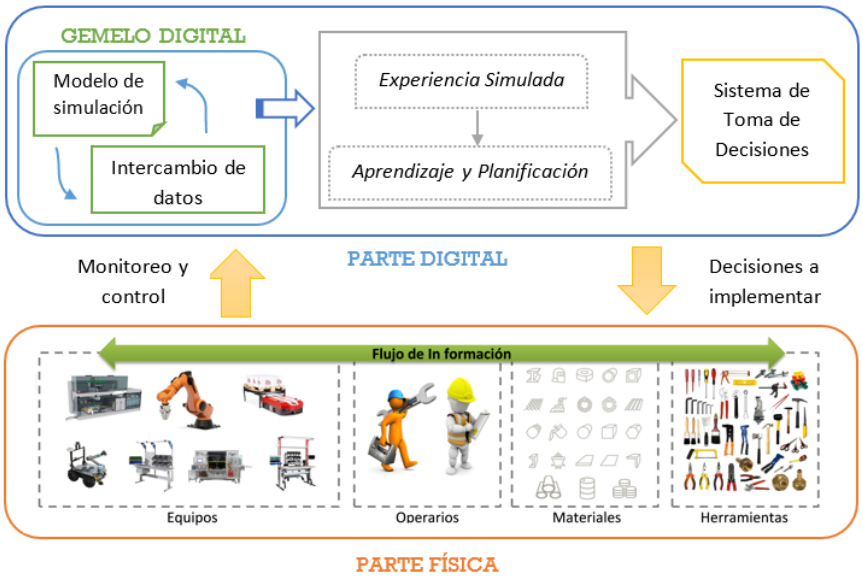


Fig. 1. Entidades y relaciones para un sistema de manufactura flexible.

La parte digital, representada en el fragmento superior de la Fig. 1, contiene distintos subsistemas, servicios y aplicaciones que desarrollan distintas funciones que permiten a las entidades intervinientes en la fabricación operar coordinadamente para optimizar el desempeño del sistema productivo en su conjunto. Actualmente, muchos campos de investigación enfocan sus esfuerzos hacia el desarrollo de propuestas para este sistema virtual. Particularmente, todos ellos buscan aprovechar las bondades y beneficios de diferentes metodologías para generar propuestas que sustenten el desarrollo de sistemas autónomos de toma decisiones, los cuales deben establecer las acciones que se deben ejecutar para gestionar el sistema físico de producción [18]. En este sentido, el concepto de GD ocupa un lugar clave y está centrando la atención de

muchas investigaciones para el desarrollo de metodologías y técnicas que permitan la creación de manera fiable de tales GDs [19].

La función principal de los GDs es proporcionar una descripción física, funcional e integral del sistema en su conjunto. Los GDs son modelos virtuales para reproducir de manera realista las propiedades, comportamientos y reglas de las entidades de la parte física [20]. En otras palabras, los modelos virtuales y las entidades físicas tienen una apariencia y comportamientos similares, es decir, son *gemelos*.

Para el desarrollo de los GDs, las técnicas de simulación tienen un rol central y deben operar en un esquema de intercambio de información con una variedad de datos, provenientes de los diferentes equipos y sistemas físicos de planta, para mantener actualizados continuamente los modelos digitales con el fin de reflejar cualquier cambio que ocurra en los equivalentes físicos en el tiempo. La interacción entre el mundo físico y los GDs tiene una importancia clave debido a que los modelos virtuales y el mundo físico deben evolucionar conjuntamente, ya que este último posee un comportamiento dinámico y, por ejemplo, una misma entidad puede mostrar diversas propiedades en diferentes momentos. Por esto, los datos recogidos en tiempo real del mundo físico se deben compartir con el mundo virtual en un esquema sincronizado para mantener actualizados los modelos virtuales que permiten simular el proceso físico y su evolución.

2.3 Jerarquía de los Gemelos Digitales

Desde una perspectiva jerárquica, los GDs pueden pensarse compuestos por tres niveles diferentes: Entidades, Información y Sistema.

En un primer nivel se encuentran los modelos de todas las entidades que intervienen en el sistema físico. En este nivel, cada una de las entidades debe ser modelada digitalmente con la mayor fidelidad posible. Actualmente, existen muchas soluciones y plataformas disponibles empleadas en el ámbito industrial que proveen soluciones para cada una de las entidades. En concreto, para diferentes tipos de entidades se han desarrollado potentes simuladores computacionales que ya han dado cuenta de su eficacia. También, es común que intervengan personas para desarrollar tareas específicas, los cuales pueden ser modelados digitalmente, empleando un simulador adecuado [7]. De la misma forma, cuando intervienen entidades que realizan operaciones unitarias pueden ser modeladas mediantes entornos de simulación específicos. En definitiva, una vez que se identifican todas las entidades intervinientes en el proceso productivo, cada entidad puede ser comprendida como un subsistema dentro de otro sistema más amplio, y a su vez, existen herramientas computacionales de simulación que permiten crear los modelos digitales más apropiados de cada una de ellas.

Cada uno de los subsistemas modelados no opera de manera aislada, sino que tiene asociado un comportamiento en el que realiza intercambios (por ej. de materiales, energía, insumos, datos, etc.) con otras entidades en el proceso productivo. Es decir, cada una de las entidades presentes tiene un propósito específico y, en consecuencia, tienen asociados un conjunto de funciones con lo cual continuamente están generando datos y/o información que pueden ser almacenados y/o compartidos entre las distintas entidades físicas y sus modelos digitales.

En el sistema productivo, las entidades están relacionadas y comparten datos e información, es decir tienen una relación sinérgica y son parte de una estructura sistémica. Por ello, cada una de estas entidades debe estar adecuadamente modelada y vinculada entre sí cuando se genera el modelo del sistema productivo en su conjunto. En otras palabras, de nada sirve, por ejemplo, modelar perfectamente y de forma aislada un centro de trabajo, un robot manipulador y un operario si no se lo hace con todas las relaciones y propiedades requeridas en el entorno fabril en el que se están empleando.

Dada la heterogeneidad y la complejidad de las entidades que pueden intervenir en un sistema de manufactura flexible y por lo tanto, para generar los mejores GDs, es conveniente utilizar los simuladores específicos y adecuados para cada entidad acorde a su complejidad y especificidad. En el desarrollo de GDs para sistemas completos de producción manufacturera, la vinculación entre diferentes entornos de simulación y las entidades físicas ha sido poco desarrollada en la literatura. Por ello, en este trabajo se propone una arquitectura de software que permite integrar diferentes entornos de simulación, útiles para modelar cada una de las entidades comúnmente usadas en los entornos de manufactura flexible, así como con los dispositivos físicos propiamente dichos. Esta propuesta se desarrolla en la próxima sección.

3 Integración Tecnomatix & ROS

Hoy en día, es muy común que los fabricantes de robots y equipos automatizados para la industria provean simuladores computacionales de sus productos, los cuales son la base para la creación de los GD de cada equipo o entidad en cuestión. Sin embargo, muchas de estas soluciones disponibles no ofrecen una solución integral desde la perspectiva del sistema industrial. La diversidad y complejidad de cada una de las entidades que intervienen en un proceso productivo hacen que los simuladores destinados a modelar íntegramente un sistema productivo (por ejemplo los simuladores de eventos discretos), puedan no reflejar con suficiente grado de fidelidad el sistema en su conjunto ya que tienden a simplificar la dinámica de las entidades que lo componen, lo cual va en detrimento de la calidad de un GD del sistema industrial que se pretenda crear. Por esto, la integración de un simulador de eventos discretos con las entidades reales junto con sus simuladores especializados, representa una opción prometedora para la creación de un verdadero GD de un sistema industrial.

3.1 El Sistema Operativo Robot - ROS

En el pasado, la interacción y la comunicación con y entre los robots eran complejas y dependientes de la plataforma robótica empleada. Durante la última década, esta situación ha sido facilitada y potenciada debido al creciente desarrollo del Sistema Operativo Robot, conocido como ROS[1].

[1] https://www.ros.org/

ROS consiste de un conjunto de procesos heterogéneos e independientes denominados nodos ejecutados al mismo tiempo y teniendo la capacidad de intercambio de mensajes entre ellos. La conexión entre los nodos se denomina tópico. Un nodo puede publicar mensajes o subscribirse a un determinado tópico [21]. ROS es ampliamente utilizado en la comunidad de diseñadores y fabricantes de robots debido a sus características de software abierto, que permite encapsular nodos desarrollados en diferentes lenguajes de programación, facilitando enormemente el crecimiento incremental del sistema y su prototipado. En efecto, la incorporación de nuevos nodos que manejen nuevos dispositivos, tanto sensores como actuadores, se realiza de forma inmediata y homogénea, aún para dispositivos con drivers realizados en software propietario y muy diferente entre sí. Estas características le confieren a toda la arquitectura una gran versatilidad, fundamentalmente para el trabajo de investigación tecnológica.

Resumidamente, las principales características y ventajas de ROS son: (1) abstracción de hardware y control de dispositivos de bajo nivel; (2) intercambio de mensajes entre procesos; (3) gestión de paquetes; (4) herramientas para desarrollar, probar y ejecutar código en varios tipos de lenguajes de programación; (5) computación distribuida; (6) reutilización de software; y (7) pruebas rápidas. En este sentido, todos los robots equipados con ROS tienen el mismo criterio de comunicación.

Para la creación de los GDs, la estandarización de los protocolos de comunicación juega un papel central. Teniendo en mente que los GDs están constituidos por modelos simulados, con fusión de datos reales, es atinado pensar en la creación de estos gemelos considerando la posibilidad de interactuar con las entidades reales.

3.2 Simulador de planta *Tecnomatix*

En esta sección presentamos brevemente el paradigma de programación del simulador de procesos de manufactura TPS.

Una particularidad que, en principio, vuelve atractivo al software *Tecnomatix* es que, por una parte, se pueden explorar sus capacidades con licencias de prueba libres y con un número máximo de hasta ochenta entidades, lo cual muchas veces es suficiente para modelar, por ejemplo, un sistema productivo de una pequeña/mediana empresa. Por otra parte, permite la interconexión con software y *hardware in the loop* tipo PLC (*Programable Logic Controller*) y la integración con los productos Siemens es aún más sencilla. En la práctica esto puede ser ventajoso dado que es muy común la existencia de este tipo de equipamiento en las plantas industriales.

Finalmente otra de las ventajas que brinda el software del TPS a la hora de modelar es que tiene una interface gráfica amigable que permite modelar un sistema de manufactura simplemente seleccionando cada uno de sus componentes, ubicándolo y estableciendo las relaciones con los demás componentes del sistema. También permite programar lógicas de actuación a cada instancia utilizada con su lenguaje propietario. Asimismo, otra cualidad que lo vuelve aún más atractivo es que cada una de esas instancias permite una conexión directa con Python. Para nuestra propuesta, esto es de suma importancia dado que es posible la conexión con otras instancias cuyo comportamiento sea el del elemento real o su homólogo simulado mediante su específico simulador dinámico.

3.3 Arquitectura de Integración

La Fig. 2, esquematiza la combinación de los elementos individuales descriptos en la secciones previas para desarrollar la arquitectura de software propuesta que integra ROS y el simulador de procesos de manufactura TPS, con la posibilidad del agregado de un sistema de decisión autónoma para optimización de procesos.

Observando en la parte superior izquierda de la Fig. 2, tenemos encerrado en una línea de color gris el ROS. Este componente se encuentra integrado por dos nodos, o procesos heterogéneos e independientes, denominados *roscore* y *Simulación de Planta*. El nodo *roscore* representa generalmente una colección de procesos y programas del ROS que permite que una red heterogénea de nodos puedan comunicarse entre sí. El nodo *Simulación de Planta* representa el proceso que tiene la responsabilidad de dirigir una planta de manufactura simulada. A su vez, este proceso puede comunicarse con otros nodos por medio de tópicos del tipo suscriptores o publicadores. Por ejemplo, el conocimiento de optimización de procesos obtenido en el nodo de *Simulación de Planta* en base a simular una planta de manufactura puede ser publicado directamente a través de un tópico que modifique el comportamiento de una entidad de planta real. Por otro lado, el nodo *Simulación de Planta* puede suscribirse a un tópico de datos proporcionado por una entidad real que altere el proceso de optimización de una planta de manufactura en particular simulada, proporcionando conocimiento que nuevamente puede ser aplicado a una entidad real, teniendo como resultado un GD para un sistema industrial.

Por otro lado en la parte superior derecha de la Fig. 2, se encuentra encerrado en una línea de color celeste el simulador de procesos de manufactura TPS. Con este software es posible diseñar plantas de manufactura adaptadas a cada problemática en particular con el objetivo de simular sus procesos de producción. Cada una de las plantas de manufactura se almacena en archivos externos con extensión .spp. Por ejemplo, en la Fig. 2 se muestra el modelo desarrollado en el simulador TPS para un GD de un problema puntual de manejo de materiales en un piso de planta flexible. Este caso de estudio se aborda en detalle en la Sección 4.

En la parte central de la Fig. 2, delimitada por una línea de color verde, se encuentra la integración entre ROS y el software TPS. Como se puede ver, se propone un patrón de diseño de clases siguiendo el paradigma de programación orientada a objetos para el desarrollo de la arquitectura de software propuesta en este trabajo. El diseño propuesto se implementó y testeo íntegramente utilizando el lenguaje de programación Python. La comunicación entre ROS y el software TPS se realiza mediante la clase denominada *Interfaz de Conexión*, que tiene la responsabilidad de proporcionar puntos de acceso externos que realizan alteraciones dentro del software de simulación. Estos puntos de acceso comprenden la carga de una planta en particular para ser procesada por el simulador, establecer y obtener datos de una entidad cualquiera simulada, comenzar, parar y pausar simulaciones, entre otras alteraciones permitidas por el software TPS. Para implementar esta comunicación se utilizó la librería *win32com* la cual permite intervenir de forma externa cualquier sistema que sea ejecutado dentro del sistema operativo de Windows®. La clase *Interfaz de Conexión* representa el patrón de diseño *Mediator* utilizado comúnmente en el desarrollo de

software de la programación orientada a objetos, el cual tiene la responsabilidad de encapsular la comunicación entre los objetos [22].

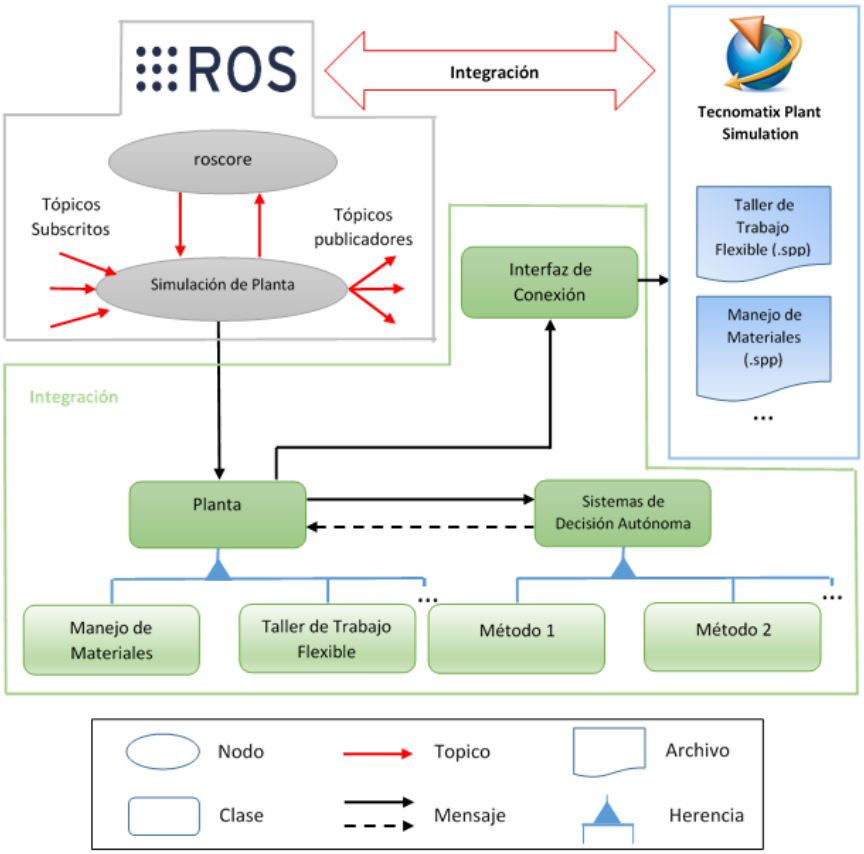


Fig. 2. Arquitectura de software propuesta para integrar ROS y un simulador de procesos de manufactura TPS, con la posibilidad del agregado de un sistema de decisión autónoma.

Siguiendo con el diseño de la arquitectura de software propuesta se observa que el nodo *Simulación de Planta* del ROS ejecuta procesos mediante el pasaje de mensajes, de una planta en particular utilizando la clase *Planta*. La clase *Planta* representa una clase abstracta y sigue el patrón de diseño denominado herencia simple, la cual abstrae todo el comportamiento general de cualquier planta de manufactura simulada [23]. Precisamente, la clase *Planta* ejecuta procesos mediante el pasaje de mensajes por medio de la clase *Interfaz de Conexión*, la cual representa un comportamiento general que requiere cualquier planta de manufactura simulada coordinada externamente. En esta jerarquía simple se tienen por ejemplo dos clases de plantas de manufactura concretas denominadas *Taller de Trabajo Flexible* y *Manejo de Materiales*, entre otras plantas que pueden ser implementadas. Cada planta concreta implementa el comportamiento de coordinación de una planta simulada por el software de simula-

ción. Por ejemplo, la clase concreta *Manejo de Materiales* tiene la responsabilidad de cargar dentro del software de simulación el archivo *Manejo de Materiales* con extensión spp, cambiar el comportamiento de la planta simulada ante establecimiento de datos a entidades simuladas, comenzar, parar y extraer resultados de una simulación, siempre a través de los puntos de accesos brindados por la *Interfaz de Conexión*. Se debe tener en cuenta que cada comportamiento específico de cada clase concreta de planta depende de la simulación abordada para cada planta en particular. Asimismo, debe notarse que la extensión de una nueva planta concreta que dirija de forma externa una planta simulada por el software TPS es directa, simplemente se debe crear una nueva clase que herede de la clase *Planta* y codificar el comportamiento simulado requerido, promoviendo la escalabilidad de la arquitectura de software propuesta.

Paralelamente a la jerarquía de planta, se tiene una herencia simple con la clase abstracta *Sistemas de Decisión Autónoma*. Esta clase abstrae todo el comportamiento general de cualquier método de toma de decisiones autónoma provista por la inteligencia artificial. Una clase concreta, y como se mostrará en la Sección 4, representa la aplicación de un método particular de aprendizaje por refuerzo. Notar, que el agregado de un nuevo método de decisión autónomo es directo, simplemente se debe crear una clase que herede de la clase *Sistemas de Decisión Autónoma* y codificar el método en particular, nuevamente fomentando la escalabilidad de la arquitectura de software propuesta.

Estas jerarquías paralelas interrelacionadas entre las clases abstractas *Planta* y *Sistemas de Decisión Autónoma* representa una combinación de patrones de diseño conocidos como *Factory Method* y *Observer* o *Publish-Suscribe* utilizados comúnmente en el desarrollo de software de la programación orientada a objetos [22]. El patrón *Factory Method* tiene la responsabilidad de construir o fabricar objetos de un subtipo determinado, es decir, plantas simuladas, en la que se le puede o no aplicar dentro de cada subtipo un método en particular, en este caso, un proceso del sistema de decisión autónoma. Por otro lado, el patrón y *Observer* o *Publish-Suscribe* define una dependencia entre el observador (la clase *Planta*) y observado (la clase *Sistemas de Decisión Autónoma*). De esta manera, una planta cualquiera, es decir, el observador, ejecuta un proceso en particular del sistema de decisión autónoma, es decir, el observado (ver Fig. 2 el paso de mensaje de una flecha color negra). Luego, cuando el observado cambia de estado, por ejemplo, cuando el método de sistema de decisión autónoma obtiene un resultado en la optimización de un proceso en particular, el observador es notificado automáticamente (ver Fig. 2 el paso de mensaje de una flecha color negra a trazo), teniendo nuevamente el control de la planta, y a su vez, el resultado del proceso de optimización para que seguidamente altere con el resultado obtenido la planta simulada en cuestión.

Observando con una mirada abstracta la Fig. 2 podemos notar que desde cualquier nodo del ROS podemos crear numerosas plantas de manufactura a la vez de igual o de distinto tipo, que interactúen con el simulador de procesos de manufactura TPS, y a su vez a cada tipo de planta se le puede aplicar o no métodos de un sistema de decisión autónoma.

3.4 Gemelo digital para la TDDs

Como ya hemos señalado, el paradigma de fabricación centrado en el cliente requiere que las empresas se esfuercen en buscar alternativas para mejorar su competitividad. El hecho de desarrollar sistemas autónomos de toma de decisiones (SATDD), capaces de optimizar productividad global del sistema de fabricación y consecuentemente el uso de recursos, es una ventaja distintiva para la determinación de una variedad de decisiones que deben tomarse constantemente, y en tiempo real, para hacer frente tanto a las variaciones de demanda de clientes, como así también, a las perturbaciones propias de un sistema productivo. Los SATDDs, para ser eficaces, necesitan información precisa y en tiempo real para determinar su política de actuación [8].

En el pasado, los modelos digitales se usaban casi exclusivamente para analizar y validar los comportamientos de los sistemas e instalaciones industriales de producción altamente automatizados. En la actualidad, las tecnologías modernas, la digitalización y la inteligencia artificial fomentan la necesidad de desarrollar aún más las herramientas de planificación digital a la vez que promueven el desarrollo de métodos SATDD

La propuesta de desarrollo de GDs para sistemas de manufactura, es justamente, uno de los pilares para el desarrollo de SATDD basados en el paradigma del aprendizaje por refuerzos (AR) [24]. En este esquema un agente inteligente (SATDD) interactúa con su entorno (GD) eligiendo acciones cuyo objetivo es maximizar la recompensa esperada. Formalmente, este problema se describe como un problema de decisión de Markov el cual requiere que se defina un vector de estado del sistema, el conjunto de acciones posibles a tomar en cada estado y una función de recompensa que premie los efectos de las acciones tomadas. En la Sección 4.3 se detalla cada uno de los elementos en base al caso de estudio abordado.

4 Caso de estudio

En los sistemas de manufactura flexible el manejo de materiales es un problema difícil y que afecta directamente la eficiencia y productividad del sistema de fabricación. Mejorar la eficiencia de los procesos complejos de fabricación, en particular la logística y el manejo de materiales, es un problema difícil y que ha recibido especial atención, principalmente para la planificación de tareas en esquemas de producción estáticos. Sin embargo, no hay demasiada evidencia para esquemas de producción dinámicos, cuando cambian por ejemplo, alguna de las condiciones de los pisos de planta flexibles y/o también varía la demanda. En esta sección presentamos la creación de un GD para un sistema de manufactura de manejo de materiales. La creación del modelo digital en el software TPS y su integración con ROS para el desarrollo de los correspondientes GDs, se realizó siguiendo los pasos descriptos en las secciones anteriores. Finalmente, se presentará una propuesta de integración con un sistema de toma de decisiones que emplea un agente artificial de AR.

En un problema de manejo de materiales se busca minimizar las distancias recorridas, el uso de los medios de transporte y las demoras en las entregas que interrumpan el sistema productivo. Durante la resolución de un problema de este tipo, se debe evitar la interrupción del flujo de trabajo asegurando la provisión en tiempo y forma

de los insumos necesarios para las diferentes estaciones de trabajo procurando lograr el normal funcionamiento del sistema de producción.

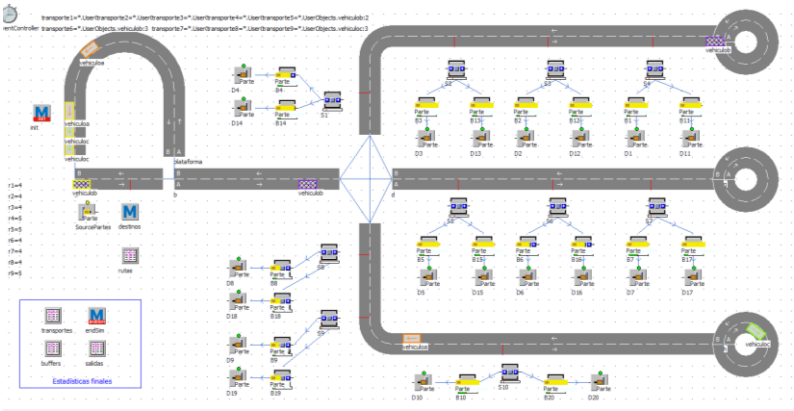


Fig. 3. Modelo de simulación en TPS.

4.1 Definición del modelo de simulación

Particularmente, para el análisis se toma el caso de estudio presentado en el trabajo de [25]. Para más detalles se recomienda recurrir al trabajo mencionado.

El sistema está compuesto por diez estaciones de trabajo ($S_i \; \forall \; i = 1, 2, \dots, 10$) que consumen dos tipos de parte (P_1, P_2) cada una simultáneamente; en cada estación existen buffers ($B_k \; \forall \; k = 1, 2, \dots, 20$) de almacenamiento para cada parte cuyas capacidades son iguales al doble de la cantidad de partes de cada tipo consumidas en una hora; hay nueve vehículos de tres tipos diferentes ($V_{je} \; \forall \; j \in \{1, 2, 3\} \; \vee \; e \in \{1, 2, 3\}$ y se desarrollan distintos planes de manejo de materiales. Cada plan de manejo de materiales tiene asociado una ruta a seguir (W_c), un transporte específico (T_d), partes y cantidades de partes a llevar a las estaciones. Los equipos de transporte no son todos iguales, es decir, cada tipo de vehículo tiene su propia capacidad de carga y puede llevar cualquier tipo de parte.

La Fig. 3 muestra el modelo desarrollado en el software TPS. Como se puede observar, en este modelo están presentes todas las entidades que forman el sistema en estudio. Asimismo se aprecian, las vías de circulación con sus respectivos sentidos, los puntos de carga/descarga, los equipos de transporte y las partes.

4.2 Integración digital

El modelo en el simulador TPS queda definido mediante un conjunto de parámetros, los cuales pueden accederse externamente mediante el nodo ROS creado. La Fig. 4 muestra los resultados finales de una simulación que sirven de entrada a un sistema de toma decisiones. Para poder cambiar los planes se recolectan: distancias recorridas por cada medio de transporte y nivel de ocupación de los mismos (Fig. 4a), nivel máximo de llenado de los buffers respecto a la capacidad definida y porción del tiempo

en que se llenan completamente (Fig. 4b) y porcentajes de trabajo y espera en cada salida que consume partes (Fig. 4c).

string	string 0	real 1	real 2
		Distancia recorrida	Porcion ocupado
1	Transporte1	611.93	0.11
2	Transporte2	612.33	0.10
3	Transporte3	628.73	0.11
4	Transporte4	613.13	0.15
5	Transporte5	533.20	0.10
6	Transporte6	629.93	0.10
7	Transporte7	534.00	0.10
8	Transporte8	614.73	0.11
9	Transporte9	1266.86	0.17

(a)

string	string 0	integer 1	integer 2	real 3
	Buffer	Capacidad	Max número partes	Porción lleno
1	B1	136	31	0.00
2	B2	178	35	0.00
3	B3	184	86	0.00
4	B4	50	29	0.00
5	B5	80	55	0.00
6	B6	132	32	0.00
7	B7	40	40	0.06
8	B8	30	14	0.00
9	B9	44	12	0.00
10	B10	60	42	0.00
11	B11	158	26	0.00
12	B12	74	31	0.00

(b)

string	string 0	real 1	real 2
	Salida	Working	Waiting
1	D1	0.85	0.15
2	D2	0.77	0.23
3	D3	1.00	0.00
4	D4	1.00	0.00
5	D5	1.00	0.00
6	D6	0.86	0.14
7	D7	1.00	0.00
8	D8	1.00	0.00
9	D9	0.91	0.09
10	D10	1.00	0.00
11	D11	0.72	0.28
12	D12	1.00	0.00

(c)

Fig. 4. Resultados obtenidos en el simulador TPS.

4.3 SATDD para el manejo de materiales

A continuación se presentan los elementos para la formulación del problema de AR. A menos que se diga lo contrario, se omite el subíndice t de representación del tiempo a efectos de simplificar la notación.

Estado del sistema

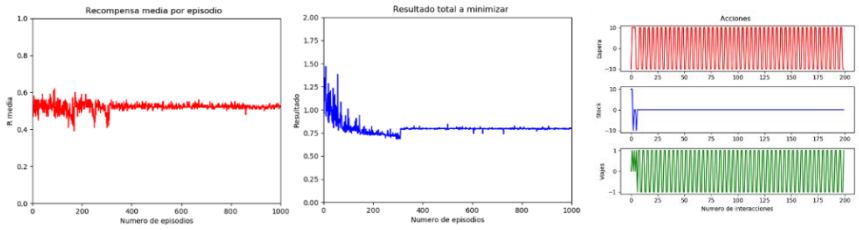
Para la representación del estado, del sistema se consideran las magnitudes de tres de las variables a saber: el tiempo de espera al cargar (T_l), el stock inicial (s_0) y la cantidad de viajes por vehículos (M), manteniéndose constantes los demás parámetros de sistema. De esta manera, el vector de estado del sistema es $\mathbf{s}_t = (T_l, s_0, M)$. Para la discretización del estado del sistema, el tiempo T_l se discretiza con un paso de 10 segundos en el intervalo [60, 300]; el stock s_0 puede tomar valores entre [10, 50] con un paso de 10 unidades; la cantidad de viajes por vehículos debe estar en el intervalo de enteros [1, 5].

Acciones

En cada estado en que se encuentre el sistema el agente podrá aumentar/disminuir en 10 segundos el tiempo de espera en cargar, aumentar/disminuir en 10 unidades el stock inicial y aumentar/disminuir en 1 los viajes por vehículo. En todos los casos, para cada variable, el agente también puede tomar la decisión nula, es decir, no modificar alguna (o todas) las variables de plan. De esta forma, el vector de acción tomado por el agente es $\mathbf{u} = (u_1, u_2, u_3)$, tal que $u_1 = \{-10; 0; 10\}$, $u_2 = \{-10; 0; 10\}$, $u_3 = \{-1; 0; 1\}$.

Función de recompensa

En este problema, se busca optimizar el desempeño del sistema en base al grado de movimientos, uso de los *buffers* y tiempos de espera. De esta forma la función de recompensa planteada devuelve un valor + 1 sí y sólo sí el plan mejora y 0 de otra forma. Para poder evaluar la mejora del plan se comparan resultados en dos momentos sucesivos: $\mathcal{L}(\mathbf{u}|\mathbf{s}) = C_1 w_1 + C_2 w_2 + C_3 w_3$; tal que C_1 es el porcentaje de ocupación de los transportes; C_2 es la porción de tiempo en que los buffers se llenan completamente y C_3 es el porcentaje de tiempo de espera total para consumir las partes. Los factores de peso w_1, w_2 y w_3 se fijaron en 0.3, 0.3 y 0.4, respectivamente. Si el resultado disminuye respecto al anterior, entonces se confirma la mejora del plan.



(a) (b) (c)

Fig. 5. Resultados del algoritmo Q-*learning*. **a**) Evolución de la recompensa media por episodio. **b**) Evolución de la función $\mathcal{L}$ a minimizar **c**) Acciones tomadas por el agente para una prueba de 200 interacciones.

Para el entrenamiento del agente artificial se aplicó el conocido algoritmo Q-*learning* usando una estrategia de exploración $\epsilon - greedy$, con un decaimiento de ϵ con el transcurso de los episodios desde $\epsilon = 1$ hasta un valor mínimo $\epsilon = 0.01$. La tasa de aprendizaje usada es $\alpha = 0.1$ y el factor de descuento es $\gamma = 0.9$. Los resultados obtenidos se muestran en la Fig. 5. La Fig. 5a muestra la recompensa promedio obtenida por el agente durante 1000 episodios de entrenamiento. Cada episodio contempla 100 instancias de interacción. Como puede verse, el agente logra converger a un valor medio de recompensa a lo largo de los episodios. La Fig. 5b muestra la minimización de la función $\mathcal{L}$ durante los mismos 1000 episodios y por último, la Fig. 5c detalla las acciones escogidas por el agente en una prueba de testeo de 200 interacciones. En conclusión este ejercicio demuestra la factibilidad de integración entre un agente de toma de decisiones autónomo y un gemelo digital creado mediante la arquitectura de integración presentada.

5 Conclusión

En este trabajo se presentó una arquitectura de software para el desarrollo de gemelos digitales mediante la integración del sistema operativo de robots y un simulador de procesos de manufactura flexible. La misma permite la interconexión e interacción entre diferentes componentes (tanto de subsistemas de hardware como de software) que pueden estar presentes en un sistema de fabricación flexible. También, se presentó una metodología sistemática para el desarrollo de un gemelo digital de un sistema de manufactura y su integración con un sistema autónomo de toma de decisiones y de esta manera, queda habilitado un entorno que, conectándose naturalmente a través de ROS con la planta de manufactura real, permitirá realizar optimizaciones con el paradigma de GD presentado. Adicionalmente, se desarrolló un sistema de toma de decisiones autónomo basado en un algoritmo de aprendizaje por refuerzos para el problema de manejo de materiales en piso de planta. En un futuro inmediato se tiene como objetivo ampliar los desarrollos de este trabajo a casos con más componentes de hardware y software para abordar otros casos de estudio en vistas a analizar el desempeño de la arquitectura de software propuesta.

Referencias

[1] A. Kusiak, “Smart manufacturing,” *Int. J. Prod. Res.*, vol. 56, no. 1–2, pp. 508–517, Jan. 2018.

[2] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, “Intelligent Manufacturing in the Context of Industry 4.0: A Review,” *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[3] I. Zolotová, P. Papcun, E. Kajáti, M. Miškuf, and J. Mocnej, “Computers & Industrial Engineering Smart and cognitive solutions for Operator 4 . 0 : Laboratory H-CPPS case studies,” *Comput. Ind. Eng.*, no. xxxx, pp. 1–15, 2018.

[4] J. Sterman, *Business dynamics : systems thinking and modeling for a complex world*. New York: McGraw-Hill, 2000.

[5] D. Mourtzis, M. Doukas, and D. Bernidaki, “Simulation in Manufacturing: Review and Challenges,” *Procedia CIRP*, vol. 25, pp. 213–229, Jan. 2014.

[6] N. Furian, M. O’Sullivan, C. Walker, S. Vössner, and D. Neubacher, “A conceptual modeling framework for discrete event simulation using hierarchical control structures,” *Simul. Model. Pract. Theory*, vol. 56, pp. 82–96, Aug. 2015.

[7] S. Baskaran *et al.*, “Digital human and robot simulation in automotive assembly using siemens process simulate: A feasibility study,” *Procedia Manuf.*, vol. 34, pp. 986–994, 2019.

[8] K. Ding, J. Lei, F. T. S. Chan, J. Hui, F. Zhang, and Y. Wang, “Hidden Markov model-based autonomous manufacturing task orchestration in smart shop floors,” *Robot. Comput. Integr. Manuf.*, vol. 61, no. September 2018, pp. 1–9, 2020.

[9] S. Haag and R. Anderl, “Digital twin – Proof of concept,” *Manuf. Lett.*, vol. 15, pp. 64–66, Jan. 2018.

[10] J. Cheng, H. Zhang, F. Tao, and C. F. Juang, “DT-II:Digital twin enhanced Industrial Internet reference framework towards smart manufacturing,” *Robot. Comput. Integr. Manuf.*, vol. 62, p. 101881, Apr. 2020.

[11] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the Digital Twin: A systematic literature review,” *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, 2020.

[12] J. Wang, C. Xu, J. Zhang, J. Bao, and R. Zhong, “A collaborative architecture of the industrial internet platform for manufacturing systems,” *Robot. Comput. Integr. Manuf.*, vol. 61, Feb. 2020.

[13] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, “Digital Twin in manufacturing: A categorical literature review and classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[14] R. P. Rolle, V. D. O. Martucci, and E. P. Godoy, “Architecture for Digital Twin implementation focusing on Industry 4.0,” *IEEE Lat. Am. Trans.*, vol. 18, no. 5, pp. 889–898, 2020.

[15] Q. M. G. Laurindo, T. A. Peixoto, and J. J. de A. Rangel, “Communication mechanism of the discrete event simulation and the mechanical project softwares for manufacturing systems,” *J. Comput. Des. Eng.*, vol. 6, no. 1, pp. 70–80, 2019.

[16] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science+Business Media, 2008.

[17] Q. Qi *et al.*, “Enabling technologies and tools for digital twin,” *J. Manuf. Syst.*, no. August, pp. 0–1, 2020.

[18] M. Kunath and H. Winkler, “Integrating the Digital Twin of the manufacturing system into a decision support system for improving the order management process,” *Procedia CIRP*, vol. 72, no. June, pp. 225–231, 2018.

[19] X. Sun, J. Bao, J. Li, Y. Zhang, S. Liu, and B. Zhou, “A digital twin-driven approach for the assembly-commissioning of high precision products,” *Robot. Comput. Integr. Manuf.*, vol. 61, no. March 2019, 2020.

[20] R. He, G. Chen, C. Dong, S. Sun, and X. Shen, “Data-driven digital twin technology for optimized control in process systems,” *ISA Trans.*, vol. 95, pp. 221–234, 2019.

[21] A. Koubaa, *Robot Operating System (ROS) The Complete Reference*, 1st ed. Springer, 2016.

[22] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Upper Saddle River, NJ, 1996.

[23] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison Wesley, 2013.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[25] S. Govindaiah and M. D. Petty, “Applying Reinforcement Learning to Plan Manufacturing Material Handling Part 1 : Background and Formal Problem Specification,” pp. 168–171, 2019.