Análisis sobre plataformas de Digital Twins

Sebastián Rodriguez
[sebastian.rodriguez.gonzalez@fing.edu.uy]
Esteban Leopardi
[esteban.leopardi@fing.edu.uy]
Sebastián Álvarez
[sebastian.alvarez@fing.edu.uy]
Lucas Talín
[lucas.talin@fing.edu.uy]

Estudiantes de Facultad de Ingeniería.
Universidad de la República Montevideo, Uruguay

ABSTRACT

En los tiempos que corren las nuevas tecnologías afloran a un ritmo acelerado, tal es el caso de la fusión entre entidades físicas y virtuales, en lo que se conoce como los gemelos digitales (Digital Twins). El trabajo realizado a lo largo de este artículo será analizar distintas plataformas de Digital Twins evaluando mediante el método T-Check sus prestaciones para luego responder las preguntas de ¿Qué tan bien cumplen las hipótesis planteadas? y en base a esto explorar la interrogante de ¿Cuál es el estado actual de las plataformas, la tecnología y el concepto de Digital Twins?.
Se definirá el concepto de qué es un Digital Twin, los requisitos necesarios que una plataforma que trate con esta tecnología debería poseer, y posteriormente estableceremos una comparativa de cumplimientos entre las mismas.
Finalmente se culminará presentando las razones por las cuales, a nuestro entender, el concepto de Digital Twins no ha alcanzado un estatus consolidado, existiendo discrepancias entre distintas fuentes, y las plataformas aún tienen un camino que recorrer antes de llegar a cumplir todas las hipótesis planteadas.

Palabras clave: Digital Twin, IoT, Simulación, T-Check, Plataformas, SaaS

INTRODUCCIÓN

El presente informe tiene como objetivo el análisis tanto del concepto de Digital Twins como de las plataformas de Digital Twins que se encuentran disponibles a la fecha, aplicando para cada una el método T-Check.

Se presenta en la sección **Marco Conceptual** los conocimientos previos más relevantes que permiten al lector entrar en contexto para facilitar el abordaje de los conceptos manejados y del trabajo realizado a lo largo del documento, presentando además una definición propia sobre qué es un Digital Twin y qué características deben poseer. Luego en la sección **Plataformas Analizadas**, se realiza una breve descripción de cuáles serán las plataformas alcanzadas por este análisis y se describe resumidamente cuáles son sus principales características. En esta ocasión serán analizadas las siguientes cuatro plataformas: Eclipse Ditto, Prespective, Azure Digital Twins y AWS IoT TwinMaker. Seguido de esto, en la sección **Hipótesis**, se plantean una serie de hipótesis cuyo cumplimiento será puesto en evaluación para cada una de las plataformas, según los lineamientos generales dictados por el método T-Check. En la sección **Comparativa**, se realiza una breve comparación entre las plataformas indicando tanto sus puntos coincidentes como aquellos en los que se diferencian. Finalmente, en la sección **Conclusiones** se da una recapitulación general tanto del concepto Digital Twin así como de las plataformas analizadas, en base a lo que se ha podido investigar y poner a prueba en el marco de este análisis.

MARCO CONCEPTUAL

Según IBM [1] un Digital Twin es un modelo virtual diseñado para reflejar de manera precisa las características de un objeto físico. El objeto estudiado está provisto de varios sensores en áreas vitales de su funcionamiento. Estos sensores producen datos de diferentes aspectos del desempeño del objeto físico, tales como potencia generada, temperatura, condiciones climáticas, entre otros.
Estos datos son luego transmitidos a un sistema de procesamiento y aplicados a la copia digital. Teniendo estos datos, el modelo digital puede ser usado para correr simulaciones, estudiar aspectos de rendimiento y generar posibles mejoras, todo esto con el objetivo de generar valiosas lecciones que posteriormente pueden ser aplicadas de nuevo al objeto físico.

En nuestra definición de Digital Twins, un digital twin es un modelo virtual de una entidad o proceso del mundo real con suficiente precisión potencial. A la entidad o proceso le llamamos parte física y al modelo virtual le llamamos parte virtual.
Precisión actual es la capacidad que tiene la parte virtual de representar fielmente a la parte física en un momento dado. La misma mejora mediante la obtención de datos en tiempo real de la parte física. Por otro lado, la **precisión potencial** es la máxima precisión actual que teóricamente puede alcanzar la parte virtual si está todo bien configurado, funcionando correctamente y ha tenido suficiente tiempo para estabilizarse. Esta mejora mediante intervención humana.

Continuando nuestra definición, introducimos el término de "**Caso ideal de uso**" que define a un Digital Twin, la cantidad de ocurrencias del caso ideal depende del contexto de uso. Pero para decir que estamos trabajando con un digital twin debe ocurrir o estar planeado su ejecución al menos una vez. En el caso de ejecución ideal la parte física alimenta (a través de un canal bidireccional) a la parte virtual con datos en tiempo real (obtenidos de sensores) los cuales la parte virtual utiliza para mejorar su precisión actual. Al mismo tiempo la parte virtual retroalimenta a la parte física con datos (obtenidos de simulaciones en tiempo real) y comandos para mejorar su rendimiento. La información obtenida a partir de una ejecución ideal en un entorno de desarrollo además podrá indicar modificaciones posibles al diseño de la parte física para mejorar alguno de sus atributos.

El orden de creación de la parte física y la parte virtual, no está predeterminado. Tanto la parte virtual como la parte física pueden ser creadas antes que su contraparte. Si existe la parte física, se puede desarrollar la parte virtual en base a esta esperando obtener una mejora de rendimiento. Por otro lado, si no existe la parte física y nuestro objetivo es crearla podemos realizar un ahorro al disminuir la cantidad de prototipos físicos que pueden ser muy costosos.

Las utilidades que uno le puede dar al uso de Digital Twins es amplia y abarca varias áreas, como por ejemplo: Mejorar el entendimiento de la parte física. Esto es muy valioso en la toma de decisiones en la etapa de desarrollo de un producto.
Mejorar la eficiencia de procesos productivos existentes. Ayuda a establecer plazos para el mantenimiento de ciertos objetos, como motores o maquinaria industrial. Correr simulaciones sobre una entidad o proceso [1].

El uso de Digital Twins provee una serie de beneficios en varias áreas de ingeniería, desde diseño, producción hasta mantenimiento y el final de vida útil de un producto. Es altamente aplicable a cualquier entidad o proceso en el mundo físico. Sin embargo es necesario evaluar si la complejidad de la entidad/proceso amerita el uso de los recursos necesarios para desarrollar y utilizar un Digital Twin, y también revisar si es financieramente rentable su uso, ya que (actualmente) la elaboración de un digital twin es generalmente costosa [2].

IoT significa "Internet of Things" o internet de las cosas y se describe como la red de objetos físicos ("cosas") que llevan incorporados sensores, software y otras tecnologías con el fin de conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet [3].

SaaS cuyas siglas refieren a "Software as a Service" ,o software como servicio, es un modelo de entrega de software basado en la nube en el que el proveedor de la nube desarrolla y mantiene el software de las aplicaciones en la nube, proporciona actualizaciones automáticas del mismo y lo pone a disposición de sus clientes a través de Internet con un sistema de pago por uso [4].

El método T-check es un método desarrollado por el Carnegie Mellon Software Engineering Institute (SEI) empleado para evaluar si una tecnología aplica al contexto en donde se planea usar, o para verificar si determinadas afirmaciones hechas acerca de la tecnología son efectivamente ciertas. El mismo consiste en la generación de hipótesis y posteriormente criterios de evaluación, que puedan ser probados individualmente para finalmente realizar una evaluación acerca de la utilidad de la tecnología basándose en el cumplimiento de dichas hipótesis y criterios [5].

PLATAFORMAS ANALIZADAS

ECLIPSE DITTO

Eclipse Ditto es un framework open source que tiene como objetivo ayudar a construir Digital Twins para dispositivos conectados a internet. Es agnóstico al dominio en su diseño y por ende puede ser utilizado en dominios industriales, residenciales, agrícolas y cualquier otro dominio de IoT [6].

En pocas palabras, ditto actúa como un middleware para IoT proveyendo una capa de abstracción para soluciones de IoT interactuando con dispositivos físicos a través del patrón de Digital Twins [7]. Para la implementación de estas soluciones se proveen dos librerías (SDK's) una para java y la otra para javascript, las cuales proveen una interfaz programática para interactuar con Ditto sin necesidad de comprender el protocolo Ditto. Por otro lado, los dispositivos se integran con Ditto a través de capas de conectividad como Eclipse Hono o brokers MQTT como Eclipse Mosquitto [8].

Un escenario sencillo de uso de Ditto puede ser: varios dispositivos conectados con un broker MQTT, publicando mensajes y suscritos a cambios que indican instrucciones. Por otro lado, Ditto está conectado a este mismo broker recibiendo los mensajes, mapeandolos y actualizando su bbdd con la información recibida de los dispositivos. Finalmente el cliente solución IoT está conectado a Ditto y recibe, a través de él, notificaciones en tiempo real de los cambios en la información de los dispositivos, con esta puede realizar una simulación y enviarle a Ditto información de control sobre el dispositivo. Dicha información luego se publica como un cambio al broker y el dispositivo suscrito a estos cambios puede alterar su funcionamiento en base a estas instrucciones. Resumidamente, ditto actúa como un middleware que facilita las conexiones bidireccionales entre dispositivo/s físicos (parte física) y su/s modelo/s virtuales (parte virtual). Durante todo este proceso además provee acceso autenticado a los dispositivos así como la persistencia del último estado (dado por el último mensaje) conocido del dispositivo.

PRESPECTIVE

Prespective es una plataforma que funciona integrada con Unity. Para ponerlo en contexto, vale la pena antes mencionar qué es Unity. Unity es una plataforma que permite, entre otras cosas, crear videojuegos para diversas plataformas como pueden ser PC, android o consolas. Dentro de las características que más destacan a Unity están su motor gráfico, que permite renderizar gráficos en 2D y 3D; su motor físico, que permite simular de manera bastante realista las leyes de la física; su interfaz, que provee una manera bastante gráfica de conectar a todos los componentes del sistema de una manera relativamente sencilla [9].
Algo bastante interesante, es que Unity integra en su IDE la posibilidad de implementar scripts empleando el lenguaje C#. Esto abre una infinidad de posibilidades ya que permite vincular de manera lógica diversos elementos del proyecto, o definir ciertos comportamientos. Por ejemplo, es posible desencadenar acciones o eventos en base a la interacción del usuario con alguno de los elementos que aparecen en pantalla o en base a interacción entre componentes dentro del propio sistema [10].
Finalmente, es preciso para este caso mencionar también que Unity cuenta con un catálogo de componentes (Unity asset store) en el cual es posible encontrar diversos recursos tales como modelos en 3D, proyectos, recursos gráficos, plugins o scripts, tanto gratuitos como pagos, que pueden ser sencillamente importados a cualquier proyecto.

Habiendo hecho esta introducción, se puede decir entonces que Prespective funciona como un componente que puede ser encontrado en la asset store de Unity. Basta con importarlo a un nuevo proyecto para comenzar a trabajar con él.
A todas las funcionalidades ya mencionadas que ofrece Unity, Prespective provee mecanismos para comunicarse con sistemas externos mediante protocolos como puede ser MQTT, por ejemplo. De esta forma, es posible recibir señales externas de control y reflejarlas en el modelo, permitiendo la representación digital de una o varias entidades físicas, o en otras palabras, permitir así la creación de la parte digital de un Digital Twin.

AZURE DIGITAL TWINS

Es una plataforma de Internet of Things (IoT), la cual permite la representación digital de distintos tipos de entidades del mundo real, como por ejemplo procesos empresariales, máquinas o personas. La misma está integrada dentro de la plataforma Azure de Microsoft y se trata de una plataforma de SaaS, el estar dentro de Microsoft Azure le permite poder integrarse con otros servicios ofrecidos por Microsoft, como lo es Azure Data Explorer, Azure Active Directory, entre otros [11].

Siendo Azure Digital Twins una SaaS, se encontrará que para poder utilizarla será necesario tener conexión a internet, para acceder a la visualización de los Digital Twins se utilizará la interfaz de Azure Digital Twins Explorer, la misma se levanta mediante la consola de comandos desde la carpeta del proyecto, y con la cual se podrá visualizar gráficamente a los DTs, las relaciones entre los mismos y los valores de sus propiedades, además de también importar nuevos modelos y esquemas de

datos a la plataforma siguiendo los formatos Digital Twin Description Language (DTDL) para los modelos y excel para las tablas de datos.

AWS IOT TWIN MAKER

Aws es una plataforma de computación en la nube. La misma provee muchos servicios entre los que se encuentra AWS IOT Twin Maker , que es el que permite la creación de los digital twin de sistemas reales del mundo real, como puede ser una fábrica, equipos industriales,etc. Lo importante a destacar de AWS Iot Twin Maker es que la misma necesita de otros servicios provistos por AWS para su funcionamiento, por ejemplo un S3. Con AWS Iot Twin Maker se puede usar conectores ya creados , o crear unos rápidamente , para usar información de sensores, procesos, equipamiento ,etc. Luego uno puede modelar en 3D su fábrica, proceso o equipo industrial , con el fin de crear una conexión entre el mundo real y el modelo 3D , con el cual AWS permite crear alarmas para cuando un resultado derive de lo esperado, y así poder editar en tiempo real los datos que sean necesarios.
La caracterización de AWS frente a otras plataformas es que se encuentra en la nube, por lo que no requiere de un hardware potente(a nivel local) para la creación de digital twins [12].

HIPÓTESIS

Hipótesis 1: La plataforma debe permitir la modificación e incorporación de componentes físicos y virtuales del Digital Twin sin necesidad de hacer un re-deploy.

- **Eclipse Ditto:** Dada la arquitectura de eclipse ditto decimos que **cumple** con este criterio ya que es posible modificar en tiempo real la estructura de datos que representa al digital twin dentro de Ditto. Aun así, cabe aclarar que no es posible modificar en tiempo real el código del cliente websockets o http (que utiliza estos datos para tomar decisiones y controlar actuadores) sin recompilar y establecer una nueva conexión. Adicionalmente, si agregamos un componente físico nuevo al sistema (es decir otro cliente mqtt o similar) la conexión entre Ditto y este va a poder ser configurada a través de comandos devops piggyback en tiempo real. Pero de nuevo, el cliente websockets o http sigue teniendo el mismo problema si se quiere incorporar a este componente en sus decisiones. Consideramos igualmente que cumple ya que este cliente está por fuera de la plataforma Ditto en sí y además es posible desarrollar con el cliente dado (librería maven) un cliente que si permita el re-deploy.
- **Prespective:** Es posible realizar modificaciones en la definición del digital twin sin necesidad de apagarlo, pero no todos estos cambios quedan guardados sino que los cambios hechos en el modelo se mantienen por el tiempo en el que dure la ejecución. Es decir, que una vez detenida la ejecución estos cambios se revierten y es necesario volver a hacerlos para poder guardarlos. Sin embargo, esto no sucede con los cambios que se hacen en los scripts. Se entiende que **se cumple parcialmente** con la hipótesis por no poder persistir todos los cambios hechos en tiempo de ejecución.
- **Azure Digital Twins:** Es posible realizar modificaciones a la definición de un Digital Twin sin interrumpir la ejecución del mismo. Utilizando el Digital Twin Explorer uno puede subir una nueva versión del modelo o agregar relaciones entre DTs, y estos cambios quedan reflejados en Microsoft Azure sin la necesidad de hacer un re-deploy o detener el DT. De la misma manera que uno puede usar el Digital Twin Explorer para actualizar un modelo, se puede subir un nuevo modelo sin la necesidad de interrupciones en el sistema. Por lo tanto decimos que se **cumple**.
- **AWS IoT Twin Maker:** Decimos que AWS Iot Twin Maker **no cumple** con esta hipótesis, ya que a pesar de que se puede editar los digital twin cuando no se están ejecutando, no se ha encontrado la posibilidad de hacerlo mientras se están ejecutando, por lo tanto se debe hacer Re-Deploy.

Hipótesis 2: La plataforma permite exportar / importar componentes en y desde la plataforma o mediante un catálogo de componentes, la orquestación de estos y la utilización de librerías externas.

- **Eclipse Ditto:** Se **cumple muy parcialmente**, es posible exportar las estructuras de datos que representan los digital twins (como jsons) y los comandos utilizados para establecer las conexiones. También es posible exportar el código fuente java que utiliza el cliente (o el binario compilado). Pero no existe una herramienta de la plataforma que integre todo, uno debe exportar todo manualmente. Es decir no se puede exportar todo esto junto y claramente tampoco importarlo junto. En el caso de importar no se puede crear el digital twin en ditto, establecer las conexiones entre el dispositivo físico y ditto, y también correr el cliente ditto java todo junto. Esto se puede hacer con un script pero no sería una funcionalidad de la plataforma. Por otro lado, no existen apenas ejemplos de código, mucho menos catálogos que provean todo lo necesario para levantar un ejemplo completo de digital twins como anteriormente fue mencionado. Adicionalmente, si consideramos la parte virtual del digital twin al cliente java, y si tenemos muchos

clientes distintos entonces decimos que ditto no brinda ninguna herramienta para orquestar a estos clientes. Como mucho provee un sistema de usuarios con autenticación para los mismos. Finalmente, la lógica del digital twin en sí está implementada en un cliente java, entonces se puede importar cualquier librería de un repositorio maven por ejemplo. El cliente ditto en sí es una librería en un repositorio maven.

- **Prespective:** Se logró comprobar que es posible importar y exportar componentes (assets) creados en la plataforma. Al ser componentes creados en Unity, el mecanismo usado para importar/exportar es el mismo: a través del menú *assets > import/export*.
  Complementariamente, la plataforma cuenta con una librería llamada "Prespective industrial assets", en donde se ofrecen modelos 3D que pueden ser fácilmente incluídos al proyecto en el que se esté trabajando. Vale la pena destacar que esta opción está disponible solamente en una versión beta de la plataforma y no ha podido ser verificada de manera empírica. Se pudo comprobar que la plataforma permite orquestar diferentes digital twins, ya que dadas dos entidades virtuales creadas en en esta, las mismas pueden interactuar entre sí, e incluso el usuario puede modificar valores como velocidad o movimiento (en el ejemplo sobre el cual se trabajó). Finalmente, la plataforma permite la importación de modelos CAD en 3D para ser utilizados en los proyectos. Si bien esta funcionalidad no ha sido probada, es sabido que en Unity esto es posible y por lo tanto aplicaría también para Prespective. Se entiende que esto es suficiente evidencia para decir que la plataforma **cumple parcialmente** con la hipótesis.
- **Azure Digital Twins: Cumple parcialmente,** es posible importar/exportar "grafos" que involucran a los digital twins y las relaciones entre los mismos, hablando del modelo que define a un digital twin en sí, existe la posibilidad de importar modelos de DTs desde el directorio de carpetas, siendo estos creados por el usuario o pertenecientes a catálogos llamados "ontologías" que hacen referencia a un dominio determinado. En cuanto a exportar los mismos, se exportan directamente desde el directorio del proyecto. La plataforma permite crear relaciones entre distintos digital twins con los mencionados "grafos". No posee capacidad para utilizar librerías externas en los digital twins debido a que el código de los modelos no se escribe dentro de la plataforma.
- **AWS IoT Twin Maker: Cumple parcialmente**. Se evidenció que se pueden importar y exportar componentes, ya que los mismos están en formato json. Los mismos se pueden reutilizar siempre y cuando uno los tenga de un proyecto anterior o de los ejemplos dados en la guía de aws. A su vez existen varios componentes en los cuales se pueden definir relaciones entre dos entidades diferentes, como por ejemplo el connector. Lamentablemente no se pudo encontrar la posibilidad de importar librerías externas que no sean de AWS.
  Se pueden importar y exportar componentes, ya que los mismos son json. Cabe aclarar que los componentes de aws son json que representan por ejemplo un motor de una entidad en general.

Hipótesis 3: La plataforma provee características de seguridad que permiten mantener la integridad, disponibilidad y confidencialidad de la información. Deben proveerse funcionalidades como protección de datos en tránsito, uso de canales encriptados y almacenamiento de datos de forma encriptada. Además, las acciones sobre el sistema y los objetos (o partes de ellos) deben soportar autenticación de usuarios y manejar autorización de funcionalidades.

- **Eclipse Ditto:** Se **cumple**, se puede configurar la conexión MQTT para que funcione sobre TLS y de esta forma asegurar el canal entre la parte física y ditto. Para esto se necesita crear certificados para cliente y servidor, instalarlos y además instalar un certificado de CA. Esto se debe hacer tanto en ditto, como en el broker mqtt como en el cliente mqtt (parte física). También se pueden asegurar las conexiones HTTP y Websockets, la forma de realizar esto es asegurando el módulo de gateway de ditto. Configurando la api http para que funcione con https y la api websockets para que funcione sobre TLS. Esto se hace modificando el componente de reverse proxy nginx. No se pudo testear ya que es una configuración manual ardua sin apoyo de la plataforma. También se puede asegurar la información en reposo modificando mongod (daemon de la base de datos usada por ditto) para que realice este encriptado. Esto no se pudo testear ya que el motor de encriptado de mongodb es una funcionalidad enterprise. Por otro lado, la plataforma flojea en cuanto a autenticación sobre acciones del sistema. Para las acciones devops sobre el sistema (manejar conectividad, manejar logs, etc) no hay autorización y la autenticación es mono usuario. Solo maneja un usuario que puede modificar el sistema en tiempo de ejecución, Por defecto devops:foobar. Se puede cambiar la contraseña utilizando una variable de entorno en el contenedor donde corre el servicio de gateway. Por otro lado, para las acciones sobre los objetos del dominio (things, features y policies) no se puede especificar quién tiene permiso para crear. Pero si se puede especificar quién tiene permiso a leer, modificar, o eliminar dichos objetos. Incluso con detalle de granularidad, es decir a cada parte de estos se le puede asignar distintos permisos, como se detalla a continuación. Para la autenticación de estos usuarios se puede utilizar un token jwt de google o otro proveedor de openid connect (lo cual no se testeo). También existe la posibilidad de agregar usuarios al reverse proxy nginx. Por defecto sólo existe el usuario ditto:ditto pero se pueden agregar más. Este reverse proxy asegura la autenticación y luego le envía al servicio de gateway el nombre del usuario autenticado solo si la autenticación fue satisfactoria. Luego se ejecuta el esquema de autorización. Se determina a qué recurso está intentando acceder el usuario y con qué operación (read, write, execute) y luego se determina si la política que domina el acceso a ese recurso le permite a este usuario

realizar esta operación sobre este recurso. Se indica qué política determina el acceso a un recurso (por ejemplo una thing) al momento de crearlo. En la misma se pueden especificar sujetos (usuarios) y sus determinados privilegios (operación, parte del recurso).

- **Prespective:** En materia de canales de comunicación seguros, la plataforma permite el uso de MQTT con usuario y contraseña, y existe la opción de utilizar TLS. Respecto a la protección de los datos, no existen mecanismos que permitan almacenar datos en forma encriptada de manera nativa. Sí, podría llegar a ser posible si se implementase un script en C# que logre almacenar la información deseada y encriptarla, lo cual es en teoría posible, pero nuevamente, no tendría validez en este caso ya que no se trataría de una funcionalidad provista de manera nativa.
  En lo que tiene que ver con el control de acceso y la protección de la información, la plataforma no maneja ningún sistema de permisos sobre objetos o digital twins. Es más, la plataforma no cuenta con ningún mecanismo de autenticación que sea requerido para, por ejemplo, abrir un proyecto y realizar cambios al mismo. En conclusión, la plataforma **cumple muy parcialmente** la hipótesis.
- **Azure Digital Twins:** Se **cumple**, ADT utiliza lo que se conoce como "rutas de eventos" para enviar datos internamente y a consumidores externos. Una ruta de eventos permite enviar datos de eventos desde un digital twin en ADT a otros DTs o a puntos de conexión definidos por el usuario en las suscripciones de manera segura. Actualmente se admiten tres servicios de Azure para puntos de conexión: Event Hubs, Event Grid y Service Bus. ADT permite el cifrado de los datos en reposo y en tránsito a medida que se almacenan en los centros de datos de Microsoft y los descifra automáticamente mientras se accede a ellos. Este cifrado se produce mediante una clave de cifrado administrada de Microsoft. La plataforma brinda soporte de autenticación de usuarios mediante Azure Active Directory (Azure AD). El cual trabaja con tokens de OAuth 2.0 para controlar los accesos a la aplicación. ADT permite el control de acceso preciso sobre datos, recursos y acciones específicas en su implementación. Para ello, se usa una estrategia pormenorizada de roles y permisos que se conoce como Control de acceso basado en roles de Azure (Azure RBAC), la cual es proporcionada en ADT mediante la integración con Azure AD.
- **AWS IoT Twin Maker:** Se **cumple** la hipótesis Toda la información enviada en tránsito a AWS Iot Twin Maker es enviada mediante una conexión tls usando protocolos https, ademas de que toda la información recopilada por los DT es guardada en el servicio de aws S3 donde el mismo por defecto tiene la funcionalidad de encriptación de datos, además de ser un claro ejemplo de un servicio que apoya ACLs. Dado lo anterior podemos decir que toda la información manejada entre digital twins es segura y está encriptada. Además de todo esto, AWS provee el servicio de IAM (Identity and Access Management), el cual permite crear,editar y controlar usuarios,sus roles y permisos, por lo que fácilmente uno puede crear usuarios restringiendo su control y visualización sobre los DT.

Hipótesis 4: La plataforma cuenta con documentación sobre todas las funcionalidades que provee, algún mecanismo de soporte brindado por alguna empresa para poder escalar los problemas, foros activos para dudas, tutoriales o cursos remotos o presenciales, asistentes de inicio rápido y la última versión fue liberada hace menos de seis meses.

- **Eclipse Ditto:** Se **cumple muy parcialmente**. Ditto en sí provee documentación [8] pero está muy fragmentada. Por ejemplo, para realizar una conexión simple mqtt si uno va a la sección que explica esto no se va a entender cómo realizar esta conexión, ya que no se da un ejemplo entero. Uno antes tiene que ir a la sección de comandos donde explican los comandos piggyback y ahí descubrir que comando utilizar. Por otro lado, la documentación sobre el cliente java utilizado no solo es escasa, sino que también es errónea. Los ejemplos dados en la página de ditto no funcionan, probablemente estén desactualizados. No provee mecanismo de soporte, lo más cercano es un chat gitter.im donde cualquiera puede escribir. El foro de eclipse ditto está inactivo y solamente cuenta con un total de ocho posts durante un periodo de cuatro años (2018-2022). No provee cursos o tutoriales. Lo más cercano a asistentes de inicio rápido son los ejemplos en [13] que resultaron indispensables para tener algo funcional. Finalmente la versión 2.4 de ditto en docker compose salió hace 3 meses (integra todos los componentes que forman a ditto que tienen releases asíncronas).
- **Prespective:** La documentación provista de manera pública no explica en detalle todas las funcionalidades de la plataforma. (por ejemplo, cómo conectarse a dispositivos IoT). No existe documentación acerca de todas las versiones que pueden descargarse desde la web oficial. Si bien existen medios para obtener soporte oficial por parte de la empresa, este es exclusivo para clientes con licencias pagas.
  Por otro lado, no se han encontrado foros oficiales ni secundarios, ni comunidades en donde se de un intercambio entre usuarios. La última versión liberada data del año 2021.
  Finalmente, si bien la plataforma cuenta con tutoriales básicos con proyectos de ejemplo que pueden ser descargados, no se proveen *wizards o* asistentes de inicio rápido.
  Se concluye por lo tanto que la plataforma **no cumple** con esta hipótesis.
- **Azure Digital Twins:** La plataforma cuenta con documentación que describe sus funcionalidades, la cual se puede encontrar en [14]. También cuenta con un servicio propio de Microsoft de soporte, en la página de Microsoft Azure en la sección de Ayuda y Soporte Técnico donde uno puede reportar los problemas surgidos. Existe un foro de

Microsoft Q&A, donde se plantean discusiones de manera activa y se ha visto cómo las preguntas encuentran respuesta en cuestión de días. Se encontraron tutoriales en la página de documentos de Microsoft y en YouTube en la cuenta oficial de Microsoft Azure. No se proveen asistentes de inicio rápido, solamente documentación. La plataforma en la cual opera ADT (Azure Microsoft) tiene actualizaciones mensualmente. Se **cumple parcialmente** debido a que no se posee un asistente de inicio rápido y esto afecta considerablemente a la hora de empezar a trabajar con la plataforma.

- **AWS IoT Twin Maker:** Se **cumple parcialmente**. La documentación provista por aws iot twin maker está en [15], en la misma también se encuentra un tutorial de como usar AWS IoT Twin Maker y los pasos que se deben seguir para armar un ambiente de Digital Twins. Debido a que aws es una plataforma en la nube ya cuenta con otros servicios que permiten manejar la escalabilidad de los problemas, por ejemplo dependiendo de la cantidad de información que uno quiera manejar, el S3 puede ser modificado para tener más espacio, o espacio a demanda. Además de esto AWS tiene un foro dedicado a consultas de cualquier servicio que proveen. La última versión de AWS Iot Twin Maker y de su API fue el 28 de junio de 2022, como se puede ver en [16]. Lo único que no cumple de la hipótesis es que no se puede armar un ambiente en minutos, ya que el mismo lleva un tiempo considerable, incluso habiendo tenido experiencia con AWS Iot Twin Maker.

Hipótesis 5: La plataforma garantiza un flujo de comunicación bidireccional entre la parte física y la virtual. Permite que a partir de los datos obtenidos del gemelo físico se realicen predicciones utilizando el Digital Twin y en base a las mismas se modifica/configura el Physical Twin. Esta secuencia debe repetirse de manera continua para mejorar la semejanza entre ambos modelos (físico y digital). Se debe poder integrar con emuladores que simulan datos de sensores o mediante la importación de un conjunto de datos. Soporta conexiones mediante al menos dos protocolos de comunicación (por ejemplo HTTP, MQTT, etc) y permite la integración directa con otras plataformas de IoT, IA, Análisis de datos o BigData.

- **Eclipse Ditto:** Se **cumple**. Se puede realizar esto conectando los dispositivos físicos a un broker mqtt y luego ditto a este mismo broker. Finalmente se conecta un cliente ditto a ditto a través de websockets. Existe una comunicación bidireccional a través del canal twin que permite la comunicación a través de una estructura de datos común a ambos. El funcionamiento es el siguiente: los dispositivos físicos envían su información a ditto a través de un broker mqtt en tiempo real, luego cualquier cliente suscrito (por websockets) a cambios en este dispositivo recibe esta información. Finalmente, con esta información el cliente websockets puede modificar valores de control en la estructura de datos de ditto para el dispositivo y de esta forma el dispositivo suscrito a estos cambios puede alterar su funcionamiento. También existe el canal live que permite la comunicación directa entre cliente y dispositivo a través del envío de mensajes que no se testeo. Para realizar una emulación del dispositivo uno puede simplemente enviar mensajes mqtt al broker con cualquier cliente para realizar esta función. Aunque este broker debe configurarse manualmente y no viene con ditto. Se testeo el soporte de los protocolos HTTP, Websockets y MQTT, la plataforma además soporta otros protocolos no testeados. Es posible integrarse con otras plataformas ya que uno puede conectarse con la api http o websockets. Pero debe realizarse de forma manual y caso a caso.
- **Prespective:** No existen mecanismos para realizar predicciones. Sin embargo, la plataforma soporta conexiones mediante protocolos MQTT, AMQP, TWINCAT_ADS, OPC_UA y ACTIVE MQ. Por cuestiones técnicas, sólo se ha podido establecer una conexión mediante TWINCAT_ADS.
  Por otro lado, no es posible la integración directa con algún tipo de plataforma de IoT o IA.
  Respecto a emuladores, se probó que la plataforma puede conectarse con TwinCAT XAE. Esta herramienta permite la simulación de datos de control, como por ejemplo velocidad de motores.
  En base a estos datos, se concluye que la plataforma **cumple muy parcialmente** con esta hipótesis.
- **Azure Digital Twins:** Se **cumple**. Es posible establecer un flujo bidireccional de datos entre la parte física y la digital mediante las “rutas de eventos”, pero tanto la modificación del Digital y Physical Twin tienen que realizarse manualmente. Es posible importar conjuntos de datos en forma de plantillas de excel. La comunicación se da vía las ya mencionadas “rutas de eventos”, las cuales pueden utilizar por ej. Azure Event Hub como plataforma de streaming de datos y eventos, los publicadores de eventos pueden publicar eventos mediante HTTPS, AMQP 1.0 o Apache Kafka (1.0 y posterior). Azure Digital Twins ya funciona dentro del marco de Azure IoT por lo tanto viene integrada con la misma, también se permite la integración con la plataforma de análisis de Big Data Azure Data Explorer vía Event Hubs.
- **AWS IoT Twin Maker: Cumple** con la hipótesis. Además de que exista una bidireccional entre los digital twin del entorno de aws iot twin, los mismos pueden contener alarmas que puedan disparar un evento de cambio de uso de batería, ver ejemplo en [17]. Al mismo tiempo uno puede usar datos seteados en un S3, con el objetivo de simular que pasaría si se tuviesen esos datos en tiempo real, y cómo esto afectaría al DT. Por otro lado, respecto a las conexiones, AWS Iot Twin Maker utiliza AWS Iot Site Wise (para la obtención de datos de sensores del mundo físico), que a su vez puede utilizar AWS Iot Core, donde el mismo puede utilizar MQTT (Message Queuing and Telemetry Transport), MQTT over WSS(Websockets Secure), HTTPS (Hypertext Transfer Protocol - Secure),

LoRaWAN (Long Range Wide Area Network) para sus conexiones. AWS permite la integración con otras plataformas IOT, por ejemplo AWS IOT Site Wise.

Hipótesis 6: La plataforma ofrece una interfaz gráfica que permite visualizar la información del sistema. Cuenta con servicios para realizar simulaciones sobre escenarios hipotéticos de contexto y configuración de la instancia física. Permite simular el comportamiento futuro a partir de los datos obtenidos en tiempo real. Provee al menos de una herramienta de análisis de datos y servicios de inteligencia artificial para realizar este análisis.

- **Eclipse Ditto: No se cumple**. La forma más gráfica de observar la información almacenada en el sistema es a través de una interfaz web swagger y leyendo un json (esta información almacenada es actualizada en tiempo real). Por otro lado la información en tiempo real (es decir mensajes enviados directamente desde el sensor hacia un cliente) solo se puede acceder de forma programática, por ejemplo en el cliente java. Por otro lado, no se cuenta con servicios de la plataforma para realizar simulaciones de la instancia física, como se mencionó en la hipótesis 5 esto se puede simular utilizando un cliente mqtt que no es parte de la plataforma. Adicionalmente las simulaciones deben realizarse íntegramente desde cero sin ninguna asistencia por parte de la plataforma además de la recepción de los datos en tiempo real. No cuenta con servicios de inteligencia artificial ni análisis de datos. Si uno quisiera analizar los datos lo mejor que podría hacer sería implementar un cliente java que se suscriba a todos los cambios y luego los analice. No existe una implementación dada por la plataforma para hacer esto, es recomendable utilizar alguna librería java para simplificar este análisis.
- **Prespective:** Es posible monitorear los datos que llegan en tiempo real. Sin embargo, no es posible visualizar qué datos están siendo almacenados en el sistema. Vale la pena mencionar que mostrar estos datos requiere de una programación a bastante bajo nivel (no es una función nativa, que permite mostrar las entradas al modelo). Además, para desplegar los datos en la vista 3D, es necesario tener un buen dominio de Unity para generar los componentes visuales necesarios.
  En materia de simulaciones de casos hipotéticos, no existe un servicio específicamente para esto. Podría llegar a ser posible hacerlo si se utilizan datos de entrada simulados. De hecho esto fue lo que se hizo para probar la integración con la herramienta TWINCAT XAE.
  Respecto a herramientas de análisis de datos, no se provee ninguna de manera nativa. De todas formas, es posible programar scripts en C# que procesen la información obtenida en tiempo real, abriendo un abanico de posibilidades tan amplio como se quiera. Nuevamente, al tratarse de algo que requiere de un esfuerzo adicional y al no ser una funcionalidad ofrecida de manera nativa, se entiende que no es válido decir que la plataforma provee herramientas de análisis de datos.
  En resumen, la plataforma **no cumple** con la hipótesis planteada.
- **Azure Digital Twins: Cumple parcialmente.** En el Azure Digital Twins Explorer se puede visualizar la información en tiempo real del sistema. Se usa Azure Event Hubs y Azure Data Explorer para transmitir y almacenar datos históricos de Azure Digital Twins. Es posible configurar y setear los DTs sin usar datos en tiempo real a modo de simular escenarios hipotéticos. La plataforma no proporciona herramientas para predecir comportamiento futuro ni tiene servicios de inteligencia artificial para el análisis de datos. Azure cuenta con la posibilidad de integrar ADT con Azure Data Explorer para el análisis de datos, donde por ejemplo se pueden realizar gráficas usando los datos históricos del sistema, los cuales incluyen timestamp, de esta forma se pueden ver los valores de entrada para un momento dado.
- **AWS IoT Twin Maker: Cumple parcialmente**. La plataforma permite visualizar la información obtenida a través de servicios como AWS Site Wise, que permite simplificar la recopilación, organización y el análisis de datos de equipos industriales. Un ejemplo de esto sería cortar una línea de suministros a una fábrica, e indicar en tiempo real cómo cambia el flujo de los pedidos y que otra fábrica se hace cargo del mismo. Además permite definir por ejemplo funciones de cálculo de datos en particular, con la cual uno puede predecir qué puede pasar a futuro con un digital twin, activando una alarma que pueda avisar al usuario. AWS cuenta con servicios de inteligencia artificial para el análisis de datos, sin embargo no se ha encontrado evidencia de que pueda aplicarse a los aws iot twin maker. SiteWise es un servicio que además de muchas otras cosas tiene la capacidad de analizar datos de equipos industriales. Lamentablemente no se encontró evidencia para poder realizar una snapshot.

Hipótesis 7: La plataforma ofrece una interfaz para permitir que otros Digital Twins envíen o soliciten información a través de protocolos de comunicación estándar entre Digital Twins.

- **Eclipse Ditto:** Se **cumple parcialmente**, otros digital twins pueden fácilmente utilizar la api http para enviar o recibir información. También pueden utilizar una librería de java y utilizar la api websockets. Sino también se pueden comunicar con un broker mqtt y hablar protocolo ditto. Estas últimas opciones son más complejas y menos documentadas. El problema es que la forma de comunicación no es estándar. Claramente el protocolo ditto es utilizado por eclipse ditto y no por otros sistemas de digital twins. Adicionalmente, las otras api 's también son particulares y no estándar.
- **Prespective: No cumple**, no se puede conectar directamente con otros Digital Twin. No existe dicha funcionalidad, además no está mencionada en la documentación oficial.
- **Azure Digital Twins:** Se **Cumple parcialmente** la hipótesis. La comunicación entre DTs se da vía las mencionadas "rutas de eventos", las cuales pueden utilizar por ej. Azure Event Hub como plataforma de streaming de datos y eventos, los publicadores de eventos pueden publicar eventos mediante HTTPS, AMQP 1.0 o Apache Kafka (1.0 y posterior). También se cuenta con la posibilidad de enviar o consultar datos a través de la API Rest de ADT. Pero los formatos de los mensajes no llegan a ser un estándar.
- **AWS IoT Twin Maker: Cumple parcialmente** la hipótesis. El intercambio de información de un DT en AWS IoT Twin Maker es a través de una request en formato json donde para cada una el json tiene un formato determinado. Además la api de aws iot twin maker utiliza el protocolo de comunicación HTTP. Pero, el formato de estos json no es estándar entre plataformas.

Hipótesis 8: La plataforma cuenta con herramientas amigables para los operadores o expertos de dominio como soporte para lenguajes de modelado, herramientas de visualización o monitoreo en 2D y visualización 3D y es compatible con programas CAD para diseñar los digital twins.

- **Eclipse Ditto: No se cumple**. No brinda soporte para lenguajes de modelado e incluso el diseño de las estructuras de datos es a nivel de escribir un json. Para monitoreo lo mejor que se provee es una interfaz swagger donde se puede imprimir los json que representan al digital twin de forma explícita. No existe visualización 3D de ningún tipo ni es compatible con programas CAD para diseñar los digital twin. En el mejor de los casos se puede tratar de integrarlo manualmente con el cliente java (libreria maven).
- **Prespective:** La plataforma se integra con Unity, por naturaleza. Por lo tanto, admite los mismos tipos de modelos que admite Unity y es clara la posibilidad de obtener visualizaciones 2D o 3D de los modelos. Por esta misma razón, también es compatible con programas CAD de modelado. En base a esto, se puede decir que la plataforma **cumple** con esta hipótesis.
- **Azure Digital Twins:** Se **cumple muy parcialmente**. El lenguaje de modelado que presenta la plataforma es DTDL (Digital Twins Definition Language) el cual es un lenguaje basado en JSON y no es exclusivo a ADT, pero no se considera amigable para un experto de dominio. La plataforma soporta la visualización y monitoreo de los digital twins en 2D, se visualizan los DTs y las relaciones entre los mismos en lo que se llaman "twin graph" y "model graph". Los digital twins se muestran como nodos de un grafo y las relaciones como aristas que los unen. Vale la pena mencionar que las ventanas de visualización cuentan con la capacidad de correr querys para filtrar los resultados. No cuenta con soporte para visualización en 3D, para brindar esto es necesario integrar la misma a otra plataforma (por ej. [18]) que sí soporte modelos 3D. No puede trabajar directamente con modelos CAD.
- **AWS IoT Twin Maker: Cumple muy parcialmente** El lenguaje basado para modelar las entidades / componentes / atributos son con el formato JSON pero no se considera amigable para un experto de dominio. A su vez, a pesar de no poder visualizar en 2D existe una visualización en 3D, que utiliza el formato GLB para la importación de objetos 3D. A pesar de que en la documentación diga que es compatible con CAD se, intentó importar un archivo dwg y no lo pudo reconocer. Los únicos tipos que acepta de importación son los siguientes: BIN, GLB, GLTF, PNG, PDF, JPG, JPEG y MP4.

Hipótesis 9: La plataforma permite fácilmente importar/exportar datos desde CSV, JSON, Excel, BD SQL u otro formato estándar/accesible. Soporta la consulta/envío de datos a través de API REST, API SOAP o WEBSOCKETS.

- **Eclipse Ditto:** Se **cumple**, se puede exportar la información actual de datos del digital twin. Pero solo los datos y solo los actuales. No se puede exportar ni configuraciones de conexión ni los clientes ditto con la lógica ni datos históricos del digital twin. Estos datos se pueden acceder a través de una API REST que incluso tiene un swagger en donde se puede ver su definición. También se pueden acceder a través de websockets utilizando un cliente java (con la librería maven correspondiente). Por otro lado, la plataforma no provee una interfaz SOAP. Finalmente con esta API REST o Websockets se pueden importar estos mismos datos.
- **Prespective:** No es posible importar/exportar datos de manera nativa. Tendría que evaluarse la manera de hacerlo mediante scripts programados en C#, lo cual requiere un conocimiento sólido tanto de C# como de Unity. De todas formas, esto dejaría de calificar como un soporte "nativo", a pesar de que pueda llegar a hacerse, por lo que no se considera un medio válido. Además de esto, el envío de datos mediante web services no está soportado por la plataforma. En base a esto se concluye que **no cumple** con la hipótesis.
- **Azure Digital Twins:** Se **cumple,** la plataforma permite importar modelos de digital twins en lenguaje DTDL, que está basado en JSON. Además permite importar/exportar "grafos" de DTs y sus relaciones en formato Excel. La plataforma soporta consulta y envío de datos vía API REST, permite operaciones como por ejemplo crear, obtener o eliminar instancias. No soporta consulta/envío de datos vía API SOAP ni WEBSOCKETS.
- **AWS IoT Twin Maker: Cumple.** Es posible la exportación / importación en formato json. Además soporta la consulta a través de una API REST, ya que en la API de aws iot twin maker se utiliza una consulta del siguiente formato:
    - GET /workspaces/workspaceId/entities/entityId HTTP/1.1

  A través de una API similar soporta la creación de Digital Twins.
    - POST /workspaces/workspaceId/entities HTTP/1.1
    - Content-type: application/json
    - La misma utiliza un JSON como parámetro.

  A pesar de no soportar envío de datos mediante SOAP, soporta mediante WEBSOCKETS , ya que el servicio AWS IoT Core soporta el protocolo MQTT over WSS(Websockets Secure).

Hipótesis 10: La plataforma debe soportar la verificación y validación de los modelos con soporte para tests de componentes individuales. Permitiendo la creación de tests automatizados, la posibilidad de hacer pruebas con diferentes sets de datos y brindando soporte de debugger al modelo.

- **Eclipse Ditto: No se cumple**. En el lugar que uno querría realizar estas pruebas (el cliente java) uno estaría tentado a utilizar algo como junit para realizar estos tests. Pero el cliente no provee ningúna forma de configurarlo para que funcione con datos stub. Entonces, es necesario tener ditto funcionando para correr estos tests, es más, es necesario configurar otros programas (como puede ser un cliente mqtt) para que envíe la información proveniente de los sensores a un broker mqtt (otro programa). Todo esto debe realizarse de forma manual y no existe una funcionalidad de la plataforma que permita unificar esto. Una forma de realizar tests junit implica utilizar una librería mqtt java para que programáticamente simule los datos provenientes de sensores y los recibidos por actuadores y luego implementar un test complejo que debe considerar la concurrencia del sistema. Un ejemplo de un test más sencillo podría ser: ejecutar el cliente websockets y luego utilizar un test junit que envíe mensajes mqtt y luego analice las respuestas mqtt que recibe del cliente y determine si están en un rango adecuado. Finalmente, si cuenta con soporte de debugger, uno podría correr el cliente java en modo debug.
- **Prespective:** No existen funcionalidades de testing, por lo tanto la plataforma **no cumple** con esta hipótesis.
- **Azure Digital Twins: No se cumple**, no tiene capacidad de crear tests para componentes individuales. No es posible realizar pruebas automatizadas del estilo JUnit o similares. La plataforma no tiene un debugger.
- **AWS IoT Twin Maker**: **No se cumple**, no existen funcionalidades dedicadas especialmente para testing.

TABLA COMPARATIVA

Hipótesis	Eclipse Ditto	Pres-pective	Azure Digital Twins	AWS IoT Twin Maker
01. La plataforma debe permitir la modificación e incorporación de componentes físicos y virtuales del Digital Twin sin necesidad de hacer un re-deploy.	🟩	🟨	🟩	🟥
02. La plataforma permite exportar / importar componentes en y desde la plataforma o mediante un catálogo de componentes, la orquestación de estos y la utilización de librerías externas.	🟧	🟨	🟨	🟨
03. La plataforma provee características de seguridad que permiten mantener la integridad, disponibilidad y confidencialidad de la información. Deben proveerse funcionalidades como protección de datos en tránsito, uso de canales encriptados y almacenamiento de datos de forma encriptada. Además, las acciones sobre el sistema y los objetos (o partes de ellos) deben soportar autenticación de usuarios y manejar autorización de funcionalidades.	🟩	🟧	🟩	🟩
04. La plataforma cuenta con documentación sobre todas las funcionalidades que provee, algún mecanismo de soporte brindado por alguna empresa para poder escalar los problemas, foros activos para dudas, tutoriales o cursos remotos o presenciales, asistentes de inicio rápido y la última versión fue liberada hace menos de seis meses.	🟧	🟥	🟨	🟨
05. La plataforma garantiza un flujo de comunicación bidireccional entre la parte física y la virtual. Permite que a partir de los datos obtenidos del gemelo físico se realicen predicciones utilizando el Digital Twin y en base a las mismas se modifica/configura el Physical Twin. Esta secuencia debe repetirse de manera continua para mejorar la semejanza entre ambos modelos (físico y digital). Se debe poder integrar con emuladores que simulan datos de sensores o mediante la importación de un conjunto de datos. Soporta conexiones mediante al menos dos protocolos de comunicación (por ejemplo HTTP, MQTT, etc) y permite la integración directa con otras plataformas de IoT, IA, Análisis de datos o BigData.	🟩	🟧	🟩	🟩
06. La plataforma ofrece una interfaz gráfica que permite visualizar la información del sistema. Cuenta con servicios para realizar simulaciones sobre escenarios hipotéticos de contexto y configuración de la instancia física. Permite simular el comportamiento futuro a partir de los datos obtenidos en tiempo real. Provee al menos de una herramientas de análisis de datos y servicios de inteligencia artificial para realizar este análisis.	🟥	🟥	🟨	🟨
07. La plataforma ofrece una interfaz para permitir que otros Digital Twins envíen o soliciten información a través de protocolos de comunicación estándar entre Digital Twins.	🟨	🟥	🟨	🟨
08. La plataforma cuenta con herramientas amigables para los operadores o expertos de dominio como soporte para lenguajes de modelado, herramientas de visualización o monitoreo en 2D y visualización 3D y es compatible con programas CAD para diseñar los digital twins.	🟥	🟩	🟧	🟧
09. La plataforma permite fácilmente importar/exportar datos desde CSV, JSON, Excel, BD SQL u otro formato estándar/accesible. Soporta la consulta/envío de datos a través de API REST, API SOAP o WEBSOCKETS.	🟩	🟥	🟩	🟩
10. La plataforma debe soportar la verificación y validación de los modelos con soporte para tests de componentes individuales. Permitiendo la creación de tests automatizados, la posibilidad de hacer pruebas con diferentes sets de datos y brindando soporte de debugger al modelo.	🟥	🟥	🟥	🟥

(🟩: Cumple 🟨: Cumple parcialmente 🟧: Cumple muy parcialmente 🟥: No Cumple)

COMPARATIVA

Resulta interesante ver que el cumplimiento de hipótesis es muy similar entre **Azure Digital Twins** y **IoT Twin Maker**, es decir proveen funcionalidades muy similares. A nuestro entender esto es algo razonable ya que estas son las dos plataformas pagas líderes que se encuentran en competencia directa. Por otro lado, **Ditto**, a grandes rasgos, tiene un subset de las funcionalidades de estas. Lo cual también es lógico ya que es una plataforma libre y open source que provee un framework con las funcionalidades mínimas e indispensables (según sus desarrolladores) para poder implementar digital twins o la parte más fundamental de estos. El resto de las funcionalidades no vienen con la plataforma y deben ser trabajadas de forma heterogénea.
Por otro lado, se observa que **Prespective** se enfoca más en el desarrollo de simulaciones en 3D, mientras que las demás se enfocan más en la parte de conectividad. Por este motivo hay muchas funcionalidades que no tiene en comparación con las otras plataformas, como contraparte a esto en su área de especialización es la más desarrollada.
También resulta notorio ver que en general las plataformas cumplen con las hipótesis 3 y 5, es decir que todas interpretan que la parte de conectividad y seguridad es indispensable en estas plataformas. Finalmente, es interesante observar que las plataformas cumplen parcialmente con muchas hipótesis y ninguna de las plataformas cumple con la última hipótesis, lo cual deja en evidencia que las capacidades actuales de dichas plataformas todavía tienen mucho espacio para mejorar.

CONCLUSIONES

Primero que nada vale la pena detenerse un momento en lo que es el concepto de Digital Twin y cómo es el panorama respecto al mismo al día de hoy. Un aspecto que se ha podido observar a lo largo de la investigación realizada para cada una de las plataformas, es que cada proveedor maneja la definición de lo que es un Digital Twin a su propia manera. Esto lo que provoca es que, si bien existe un marco teórico común, existen también ciertos aspectos que no quedan del todo definidos como por ejemplo las características mínimas con las que debe contar una plataforma de Digital Twins. A pesar de esto, en esta sección se intentará consolidar los conocimientos adquiridos hasta el momento para poder dar una idea concisa acerca del "estado del arte" de las tecnologías que implementan este concepto.

En lo que hace referencia a las plataformas de Digital Twins, o al menos a las que se han sido alcanzadas por este estudio, se puede concluir que efectivamente no existe un único criterio respecto a cuáles son las funcionalidades que una plataforma debe de poder brindar para satisfacer las necesidades tanto técnicas como administrativas en lo que a creación, manejo y gestión de Digital Twins se refiere.
Esto provoca que se den diferencias notorias al momento de evaluar una funcionalidad en particular, como por ejemplo, la seguridad y protección de la información, o la capacidad de visualizar los componentes del sistema de manera más o menos gráfica. Estas diferencias se hicieron evidentes en particular para los casos de estudio tratados. Al realizar un balance general y contrastando las diferentes características ofrecidas por cada una de las plataformas analizadas, se ha podido observar un desbalance bastante notorio en materia de prestaciones técnicas y funcionalidades provistas de forma nativa por cada una de ellas.

Se entiende además que, el hecho de que el concepto de Digital Twins no se encuentre aún tan normalizado como sucede con otros conceptos como pueden ser Inteligencia Artificial o Machine Learning, y adicionalmente como es un concepto relativamente nuevo provoca que las plataformas que existen al día de hoy no se encuentren en un estado de madurez avanzado. Uno de los aspectos más notorios de esto, es el hecho de que prácticamente no existen (o son muy pocos) los sitios como foros o páginas web en donde se dé un intercambio entre desarrolladores o usuarios de dichas plataformas. Lo mismo sucede a la hora de buscar información técnica como tutoriales o guías. Existe una cantidad relativamente muy reducida de información, si, nuevamente, lo comparamos con otro tipo de tecnologías o conceptos ya más consolidados.

En conclusión, y en base a lo expuesto anteriormente, se podría decir que el concepto de Digital Twins como tal, se encuentra aún en un proceso de madurez intermedio. Es decir, se entiende que aún queda mucho por recorrer tanto en materia de funcionalidades como de adopción de la tecnología y de sus potenciales beneficios.
Lo mismo aplica en lo que tiene que ver con las plataformas. En otras palabras, existen algunas que parecen ser más completas que otras, o que parecen estar en un estado más avanzado de desarrollo, pero en general ninguna de las evaluadas en el marco de esta investigación se encuentra totalmente consolidada sino que están todas en constante evolución.

REFERENCIAS

[1] https://www.ibm.com/topics/what-is-a-digital-twin (1/7/2022)
[2] https://www.networkworld.com/article/3280225/what-is-digital-twin-technology-and-why-it-matters.html (30/6/2022).
[3] https://www.oracle.com/mx/internet-of-things/what-is-iot/ (29/6/2022).
[4] https://www.oracle.com/mx/applications/what-is-saas/ (29/6/2022).
[5] https://resources.sei.cmu.edu/asset_files/Brochure/2010_015_001_512005.pdf (30/6/2022).
[6] https://www.eclipse.org/ditto/index.html (1/7/2022).
[7] https://www.eclipse.org/ditto/intro-digitaltwins.html (1/7/2022)
[8] https://www.eclipse.org/ditto/intro-overview.html (1/7/2022).
[9] https://www.masterd.es/blog/que-es-unity-3d-tutorial (28/6/2022).
[10] https://gamedevacademy.org/what-is-unity/ (28/6/2022).
[11] https://azure.microsoft.com/es-es/services/digital-twins/#overview (1/7/2022).
[12] https://docs.aws.amazon.com/iot-twinmaker/index.html (27/6/2022).
[13] https://github.com/eclipse/ditto-examples (1/7/2022).
[14] https://docs.microsoft.com/es-es/azure/digital-twins/overview (1/7/2022).
[15] https://docs.aws.amazon.com/iot-twinmaker/latest/guide/what-is-twinmaker.html (27/6/2022).
[16] https://docs.aws.amazon.com/iot-twinmaker/latest/apireference/Welcome.html (27/6/2022).
[17] https://aws.amazon.com/es/blogs/iot/l2-informative-digital-twins/ (27/6/2022).
[18] https://techcommunity.microsoft.com/t5/internet-of-things-blog/visualizing-azure-digital-twins-in-3d/ba-p/2898159 (29/6/2022).