# Estado del arte de los *frameworks* para la implementación de herramientas de simulación.

Rolando J. Andrade Fernández
*White paper*
contact@rolandoandrade.me

***Resumen*—La simulación permite experimentar con modelos que representan parte o la totalidad de un sistema. El estudio por modelos dio origen a los modelos matemáticos y estos a su vez modelos de simulación. Tras años de enfrentamientos militares y revoluciones tecnológicas, surgieron los primeros simuladores computacionales para el área de la aviación y salud. La evolución de los sistemas de simulación, también ha significado un avance en la manera en la que se desarrollan, siendo los elementos comunes y repetitivos los que fomentaron la implementación de *frameworks* para la implementación de herramientas de simulación. Entre las herramientas actuales son relevantes DesmoJ y SimPy, no obstante el futuro promete novedades en el entendimiento de la simulación, la realización de cómputos y lenguajes de programación utilizados para la realización de los mismos.**

***Palabras clave*—simulación, teoría de sistemas, *frameworks*, marcos de trabajo, estado del arte, sistemas de eventos discretos y continuos.**

## I. Introducción

Una simulación es la imitación del funcionamiento de un proceso o sistema del mundo real a lo largo del tiempo con el fin de llegar a conclusiones que permitan identificar sus características [1].

La simulación ha ido evolucionando a lo largo de los años estableciéndose como un área de estudio muy importante en la actualidad, cuyas aplicaciones han revolucionado las metodologías de experimentación científica, permitiendo la demostración y el descubrimiento de soluciones a problemas complejos.

La evolución de los sistemas de simulación, también ha significado un avance en la manera en la que se desarrollan.

Ante esto, el presente artículo se destinará a dar un breve contexto sobre los orígenes e ideas que fundaron los conceptos que se manejan en la actualidad. Posteriormente, se hará énfasis en la relevancia del tema con el fin de definir el estado de arte de los *frameworks* para la implementación de las herramientas de simulación, describiendo las herramientas de código abierto más destacadas de la actualidad, los problemas que quedan por afrontar y las posibles novedades que depara el futuro.

Finalmente ser realizará una valoración objetiva sobre el tema y la repercusión que éste podría tener en el futuro inmediato.

## II. Antecedentes

Antes de estudiar las herramientas utilizadas actualmente para crear software de simulación, es necesario abordar los inicios y conceptos que obligaron a ello.

### *II-A. Estudio de los sistemas como modelos*

Un sistema es "conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a determinado objeto" [2].

Es común que los sistemas sean afectados por el ambiente donde se encuentra. Para el estudio de sistemas se crea un límite del sistema, donde se agrupan todos los factores en factores propios y externos. Un mismo factor, puede ser propio o externo para un mismo sistema dependiendo del límite del sistema establecido en el caso de estudio [1].

Dependiendo de la naturaleza de los cambios de estado, los sistemas han podido ser catalogados como discretos o continuos, pese a que en la naturaleza es casi imposible encontrar alguno que sea perteneciente completamente a uno de estos grupos [3].

Ante el problema de estudiar los sistemas complejos, con el tiempo se empezaron a desarrollar sistemas más sencillos que los representaran de manera fiel, surgiendo los modelos. Crear un modelo, ha permitido a lo largo de la historia estudiar los sistemas en caso de que no existan, sea complicado o costoso experimentar con él [1].

El estudio por modelos dio origen a los modelos matemáticos y físicos, donde los modelos matemáticos son representados por notación simbólica y ecuaciones matemáticas [1], mientras que los físicos son representaciones reales de una parte o la totalidad de un sistema[1] [4].

### *II-B. Orígenes de la simulación por computadora*

"Una simulación es una imitación de una operación de un proceso o sistema del mundo real a través del tiempo" [1].

Mucho antes de que existieran las computadoras digitales, computadoras analógicas fueron utilizadas para ayudar a los cálculos en física y matemática. Con el tiempo, tras muchos avances tecnológicos durante los siglos XIX y XX, las computadoras fueron mejorando en su rapidez y precisión en la realización de operaciones complejas, siendo las necesidades militares de la Segunda Guerra Mundial, las que finalmente aceleraron el paso de las computadoras análogas a digitales, iniciando la Era de la Información que se extiende desde 1970 hasta el presente [5].

Los orígenes de la simulación por computadora se remontan a las simulaciones de vuelo, ya que, a principio del siglo XIX

[1]Como lo pueden ser maquetas, simulaciones por túneles de viento, colisionador de partículas, etc.

EL 90 % de las catástrofes aéreas eran responsabilidad del piloto. De esta forma, en 1931 se patenta un dispositivo de entrenamiento para aviadores[2], siendo el primer simulador con una utilidad, logrando entrenar a más de 10.000 soldados de los aliados durante la Segunda Guerra Mundial en operaciones de vuelo y aterrizaje con clima desfavorable [5].

En cuanto a la guerra, desde la antigüedad el hombre desarrolló juegos de guerra, como medio de estrategia y simulación. Siendo el escenario bélico de mediados principios del siglo XX, el detonante para la realización de simuladores computarizados [6].

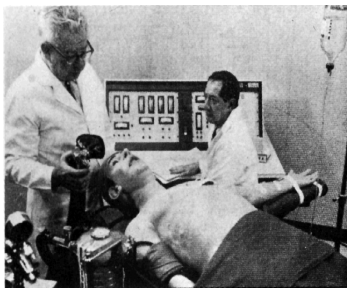

Figura 1. Sim One fue un modelo físico (híbrido) cuyo comportamiento era simulado por computadora (matemático). Extraído de [7].

Pese a que la medicina se valió durante muchos años de modelos físicos como maniquíes y estructuras anatómicas, a finales de los años 60 fue pionera en la simulación de alta precisión, surgiendo el primer simulador de anestesia "Sim One", el cual era un maniquí dirigido por computado a escala real de un paciente. Los futuros avances en el área se trasladaron a otras áreas de la salud como la veterinaria y la odontología [5] [6].

A medida que la tecnología fue avanzando, más métodos y procesos fueron trasladados a entornos virtuales o basados por computadora [6], surgiendo simuladores de todo tipo y tareas, como el estudio del cosmos, simulación de partículas, análisis de procesos productivos e incluso desarrollo de videojuegos.

### *II-C. Surgimiento de frameworks de simulación*

La flexibilidad de las herramientas de simulación, hizo que se incluyera en dominios del conocimiento muy variados, lo que dio origen a distintos desarrollos.

Un *framework*, ambiente de desarrollo, entorno de desarrollo, marco de trabajo o entorno de trabajo; modela un dominio específico o un aspecto importante del mismo [8].

Los ambientes de desarrollo representan el dominio a través de un diseño abstracto, basado en clases abstractas o interfaces, definiendo además como las mismas colaboran entre sí durante el tiempo de ejecución. Actúa como un esqueleto que determina cómo los objetos se relacionan unos con otros [8].

Los conceptos de modelado monolítico de simuladores que fueron empleados en las primeras herramientas de simulación, cambiaron de paradigma rápidamente a uno modular debido a la necesidad de dividir las operaciones complejas en piezas manejables para tratar los problemas uno a la vez, además de permitir la reutilización de parte o la totalidad de los componentes del modelo a desarrollar [2].

A medida que se desarrollaban más herramientas de simulación, se descubrió que todos tenían elementos en común, siendo principalmente un sistema dinámico, un software para construir los modelos, un motor de simulación para computar las entradas y salidas de la simulación, y un medio de control y observación de simulaciones [2].

El principal objetivo de los *framework* es permitir la reutilización y así aumentar la productividad, proveyendo clases abstractas e implementaciones concretas, con la definición de primitivas que delegan la implementación crítica a las subclases [8].

La caracterización del sistema dinámico es el marco conceptual que engloba a todos los *frameworks* de simulación, pues es el conjunto de reglas para determinado dominio.

Al lograr abstraerse el concepto de sistema dinámico a través de un mecanismo de transición de estado para el sistema, y entenderse que la interconexión de modelos en red del sistema es invariante a sus dinámicas, se ha podido crear un mismo marco de trabajo para los distintos tipos de simulación existentes [2].


Figura 2. Modelo atómico [2].

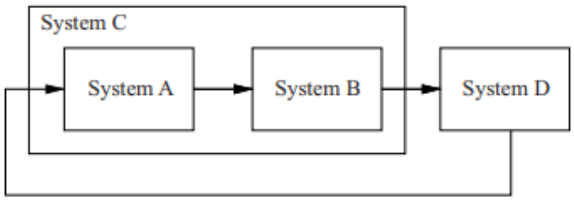


Figura 3. Modelo de red, compuesto por tres modelos atómicos [2].

En la figura 3 se puede ver como un modelo de red, tiene el mismo comportamiento que un modelo atómico (mostrado en la figura 2): entrada, estado y salida. Donde el sistema C, comprende los sistemas A y B, dando una salida hacia D, el cual también es una de sus entradas, las cuales enrruta a A. Al no depender de la dinámica, los modelos pueden ser discretos o continuos sin implicar un cambio en el *framework*.

## III. Usos prácticos de los marcos de trabajo para la implementación de herramientas de simulación

Un ambiente de desarrollo para la implementación de herramientas de software de simulación tiene la capacidad de ser una solución para los diversos problemas que se afrontan a la hora de diseñar e implementar un simulador desde cero.

[2]El nombre de la patente fue *Combination Training Device for Student Aviators and Entertainment*, registrada en septiembre de 1931.

El la administración, ventas, procesos, infraestructura, auditoría, educación y un sin fin de ramas más, se utilizan la simulación y proyección estadística para realizar proyecciones y evaluar el rendimiento de las actividades realizadas.

Para realizar las simulaciones, las empresas optan por adquirir un simulador, o por desarrollar uno propio. Entre las ventajas de utilizar uno propio estaría la capacidad de adaptarlo específicamente a sus necesidades e incluso poder ahorrar costos. No obstante, es posible que para proyectos muy específicos no exista el software de simulación, y el desarrollo de uno sea muy complicado, como por ejemplo un simulador de ambientes de conducción para vehículos autónomos o un motor de videojuegos.

Un entorno de desarrollo permite tener una misma base para realizar los distintos proyectos de simulación que puedan surgir, permitiendo reutilizar módulos y enfocándose solo en las nuevas funcionalidades. Un ejemplo de las ventajas que puede traer este marco de trabajo se puede ver en los problemas que atacan la mayoría de simuladores, ya que, la mayoría de las veces no requieren cambios en las operaciones de cómputo, reportes o control, lo que suele variar es la interfaz y el tipo de sistema que se desea simular, por lo que existe la posibilidad de volver a usar los otros módulos ya existentes sin necesidad de realizar ningún cambio, y los que cambiaron, no hace falta crearlos desde cero sino que al formar parte de un ambiente de desarrollo están abiertos a la extensión para soportar las nuevas funcionalidades [2].

El uso de un *framework* que pueda ser usado para próximos proyectos contribuye al seguimiento de buenas prácticas de Ingeniería de Software, generando productos estandarizados, robustos y repetibles, que ayuden en un futuro al estudio e investigación en el área de la simulación.

Sin embargo, pese a ser una necesidad tan relevante para el presente y futuro de las organizaciones, el estudio de los marcos de trabajo está bastante atrasado respecto a los nuevos paradigmas y avances tecnológicos, lo que obliga a limitarse a las capacidades de los ambientes de desarrollo existentes.

## IV. Marcos de trabajo para la implementación de herramientas de simulación

A lo largo de los últimos 20 años se han creado varios *frameworks* para la implementación de herramientas y simulación, donde se puede destacar DesmoJ y SimPy por su relevancia en el presente.

### *IV-A. DesmoJ*

DesmoJ es un conocido ambiente de desarrollo de simulación de eventos discretos desarrollado en Java. Entre las ventajas que provee están [9]:

- Abstracciones de modelo, entidad, evento y proceso.
- Componentes como colas, muestreo de distribuciones y contenedores de datos.
- Elementos de infraestructura como *scheduler*[3], lista de eventos y reloj.

[3]Programa y ejecuta los eventos del sistema a partir de la lista de eventos.

El modelado de un sistema en DesmoJ utilizando el enfoque de eventos y se realiza implementando clases derivadas de *Model*, *Event* y *Entity*.

*Model* es una clase abstracta que contiene el modelo del sistema a simular. En este se definen las colas de espera, las entidades y eventos, y las distribuciones que rigen los aspectos aleatorios del sistema [9].

*Entity* es la superclase de todas las entidades del sistema [9].

*Event* es la superclase de los eventos de un sistema. Pueden ser internos si necesitan una entidad para ser invocados, o externos que son los que provienen de una entrada al sistema y no necesitan ser invocados por una entidad [9].

Para la ejecución de simulaciones, se define un modelo y un experimento. En el modelo se describe el sistema y en el experimento los elementos de infraestructura.

Con base en Alaggia [9] el algoritmo de simulación usado por el *framework* para la simulación de modelos sigue los siguientes pasos:

1. Se crea una instancia del modelo a simular.
2. Se crea una instancia de experimento.
3. Se conecta el modelo con el experimento.
4. Desde el modelo se invoca la operación de inicio del experimento.
   *a*) El experimento invoca el método para programar los primeros eventos del modelo.
   *b*) Se invocan varios métodos intermedios para iniciar la simulación.
   *c*) Procesa el siguiente evento o proceso de la simulación mientras sea posible.

Además de la visión orientada a eventos, DesmoJ posee un paradigma orientado a procesos. Para crear una simulación a procesos se debe implementar una clase derivada de *SimProcesss* [9].

Finalmente, posee algunas desventajas como [9]:

- Carencia de documentación técnica sobre el funcionamiento del *framework* lo que limita su extensión.
- No es posible hacer que soporte eventos condicionales.
- Al manejar múltiples paradigmas, fue diseñado violando principios de responsabilidad única que complican el entendimiento del funcionamiento de algunas estructuras.

### *IV-B. SimPy*

Briceño [10] tras analizar los antecedentes de los distintos ambientes de desarrollo y paquetes orientados a resolver problemas de simulación, indica que estos son muy costosos y difíciles de modificar, por lo que sugiere el uso de SimPy, el cual es gratis y al ser de código abierto, es fácil de extender hacia nuevas funcionalidades.

SimPy se define como un “marco de simulación de eventos discretos basado en Python estándar” [10]. Se basa en un paradigma funcional, proveyendo al programador de una gran cantidad de recursos para hacer software de simulación de manera muy rápida.

Entre sus ventajas se encuentran [10]:

- Usa procesos paralelos para modelar componentes activos tales como mensajes, individuos, piezas
- Abstracciones de modelo, recursos, niveles y almacenes.
- Desarrollado por módulos independientes y bien definidos.

Los elementos básicos de un modelo en SimPy son objetos de tipo proceso[4]. Los procesos son retardados por tiempos fijos o aleatorios de acuerdo a las especificaciones del modelo conceptual y colocados en cola para el uso de algunos de los recursos.

SimPy maneja la declaración de una o más instancias de procesos y la creación de una serie de objetos a partir de estos. Cada objeto que extiende de *Process* ejecuta su Método de Ejecución de Proceso[5] (PEM). Cada PEM se ejecuta en paralelo invocando las diferentes rutinas de los subprocesos y, además puede interactuar con varios PEM de otros objetos de tipo *Process*.

SimPy está compuesto por módulos de simulación, trazas, velocidad de la simulación, realización de simulación paso a paso, gráfica de resultados, interfaces gráficas y formato de visualización [10].

Uno de los algoritmos posibles para realizar una simulación es el siguiente [10]:

1. Importar la librería de funciones.
2. Definir una clase de procesos.
3. Formular un modelo.
   *a*) Inicializa las funciones de ejecución de SimPy.
   *b*) Genera una o más instancias de la clase de procesos definida.
   *c*) Activa las instancias.
   *d*) Comienza la ejecución de la simulación.
4. Definir los valores de las entradas de la simulación.
5. Ejecutar el experimento.
6. Analizar los resultados.

Pese a ser una herramienta bastante útil tiene algunas desventajas [10]:

- No puede aprovechar la capacidad de correr múltiples procesos en paralelo utilizando múltiples procesadores.
- Su diseño fue realizado con un paradigma funcional que dificulta su extensión.
- No posee documentación de implementación ni guías de usuario.

### *IV-C. Problemas comunes de frameworks para la implementación de software de simulación*

La mayoría de herramientas para la implementación de software de simulación sufren varios problemas producto de su diseño y tecnologías utilizadas [9]:

- En su mayoría están dirigidos hacia un dominio específico como la simulación de redes, procesos industriales, demanda, videojuegos o tráfico, lo que limita su uso para otros ámbitos, o la necesidad de incluir varios marcos de trabajo en un mismo proyecto.
- Casi todos los ambientes de desarrollo y librerías de simulación están orientados a sistemas discretos, lo que da errores cuando se desea simular sistemas continuos.
- No existe un consenso o protocolo general entre *frameworks*, lo que hace que cada uno sea diferente en diseño y comportamiento.
- La mayoría de los proyectos de código abierto están abandonados, con falta de documentación y guías de usuario.
- Los paradigmas y decisiones de diseño empleadas no permiten la fácil extensión de los mismos para adaptarse a los nuevos descubrimientos en el área.

## V. Nuevos paradigmas para el desarrollo de herramientas para la implementación de software de simulación

Los avances tecnológicos y la inclusión de la Ingeniería de Software en el área de la simulación, ha significado cambios de ideas sobre los futuros *frameworks* a desarrollar, a fin de corregir los errores de sistemas anteriores y aprovechar las capacidades tecnológicas de los últimos años.

### *V-A. Implementación de los frameworks*

A lo largo de la historia se fueron creando varios lenguajes de modelado y diseño de simulaciones como PSM++ o SIMSCRIPT que con el tiempo fueron remplazados por lenguajes de programación orientados objetos como C++ y Java. Tener las mismas estructuras de datos requeridas y poder reutilizar los componentes por medio de la inclusión de librerías, además de poseer las características de un lenguaje de propósito general, permitió que a finales de los 90 se realizara este traslado [2].

Con el impacto de los datos en la actualidad, se espera que en los próximos años las herramientas de simulación estén directamente integradas a herramientas de hojas de cálculo y lenguajes de programación estadísticos como R y MATLAB a fin de agilizar el proceso de obtención, manipulación, procesamiento y pronóstico de los datos [2].

Por otra parte, se empezó a ver un retraso notable en los simuladores de código abierto respecto a la infraestructura existente, pues no se adaptan ni en tiempos, precisión y rendimiento al estado del arte de la infraestructura, pudiendo quedar obsoletos en los próximos años [11].

El futuro de los diseños de los frameworks estarán basados en los mecanismos de inyección por módulos y complementos, lo que permitirá ensamblar simuladores con poco esfuerzo [11].

Finalmente, para la implementación de herramientas completas de simulación, se espera la conjunción de lenguajes de modelado de simulación como Modelica, con lenguajes de programación de propósito general como C++, a través de compiladores híbridos, lo que facilitará tanto los procesos de simulación como los procesos de realización de apelaciones para usuarios finales [2].

[4]Extienden de la clase *Process*.

[5]Traducción del inglés de *Process Execution Method*.

### V-B. Computación paralela

Los investigadores en los próximos años se enfocarán en la realización de herramientas de código abierto para la implementación de software de simulación desacoplados de la infraestructura y que permitan correr simulaciones en en mononúcleo, multinúcleo y multihilo [11].

Con cada generación de computadoras, se prevé el crecimiento de la complejidad computacional de los modelos mientras que los tiempos para su ejecuxión cada vez se vuelven más razonables. Con la popularización de los mecanismos de cómputo en red y sistemas distribuidos, cada vez más surge la necesidad de adaptar las herramientas a los procesos de computación paralela [2].

El paradigma usado actualmente, solo explota los eventos que ocurren simultáneamente para procesarlos en paralelo, sin embargo, estadísticamente hablando son muy raros, por lo que este mecanismo de paralelismo es poco efectivo [2].

La nueva generación de ambientes de desarrollo para la implementación de herramientas de simulación, pese a no poder paralelizar los procesos de trayectorias por las relaciones de causalidad, promete ser más eficiente al paralelizar los procesos de cambio de estado, al realizar el cómputo de las variables que se alteran de manera paralela, disminuyendo los tiempos de cálculo numérica [2].

No obstante, algunos investigadores han querido optimizar aún más los tiempos a costa de la precisión de los resultados, proponiendo modelos conservativos y optimistas, caracterizados por ser predecibles bajo cierto margen de error, lo que reduciría el número de iteraciones de las operaciones, además de permitir el cómputo paralelo de muchos estados en distintos tiempos al existir independencia de la causalidad [2]. Estos modelos estadísticos, también quieren ser llevados a microarquitecturas para generar trazas sintéticas en tiempos razonables, lo que implicará un cambio drástico en los usos de la simulación en tiempo real [11].

El principal problema que afrontan estas propuestas, y que se está buscando resolver, es la trazabilidad de los datos, puesto que los estados al usar una estructura de datos almacenada en memoria y ser reversibles, pueden generar inconsistencias que no han logrado tener soluciones prácticas [2].

Ante el estancamiento de las velocidades secuenciales en un solo procesador, el límite de velocidad para la mayoría de las simulaciones actuales ya está determinado, por lo que es previsible que el siguiente paso sea hacia la resolución de los problemas de paralelismo [2].

### V-C. Combinación de modelos de simulación

Desde los inicios de la simulación, se ha discutido sobre la mejor forma de simular. El debate entre simulación orientada por eventos, procesos y actividades se ha extendido hasta los días de hoy, llegando a existir herramientas que cubren cada uno de los paradigmas [1].

Pese al orden establecido, con los estudios matemáticos en el área de los últimos años, ha crecido el número de modelos analíticos para representar los sistemas dinámicos como Redes de Petri y Autómatas Celulares Asíncronos [2].

Todos los modelos propuestos presentan redescubrimientos y similitudes que están bajo investigación, con el fin de conocer las relaciones entre los mismos para saber el significado de sus transformaciones y combinaciones.

### V-D. Inteligencia Artificial

Desde finales del siglo pasado se han investigado mecanismos y *frameworks* para la integración de la simulación con la inteligencia artificial.

Los primeros intentos de fusión fueron propuestas de utilizar la inteligencia artificial para optimizar el procesamiento de operaciones y el modelado de sistemas [12].

No obstante, pese a realizarse muchas investigaciones al respecto desde esos primeros artículos, a día de hoy no existen herramientas de código abierto que integren las ideas, por lo que es posible que con el nuevo auge de la inteligencia artificial, el área pueda volver a considerarse para la simulación asistida.

A pesar del estancamiento en el área de apoyo a la simulación por inteligencia artificial, se han propuesto y desarrollado *frameworks* de inteligencia artificial que permiten realizar pronósticos a través de simulaciones, lo que promete ser una alternativa a una mejor toma de decisiones [13].

Finalmente, se ha visto el potencial de los entornos virtuales simulados para el entrenamiento de modelos de inteligencia artificial, por lo que se han diseñado ambientes de desarrollo como OpenAI Gym para el entrenamiento de agentes por medio de aprendizaje reforzado, por lo que se espera que en los próximos años siga habiendo innovaciones en el área [14].

## VI. Conclusiones

La simulación ha sido un proceso que ha perdurado y evolucionado en el tiempo. Desde los orígenes de la humanidad, se intentó describir sistemas complejos a través de modelos y cálculos. La insaciable codicia del hombre por predecir situaciones reales lo llevó a simular todo tipo de escenarios.

Con la llegada de la computación, el hombre pudo expandir los horizontes de lo que podía llegar a ser la simulación, juntando modelos físicos de hardware a computadoras con software de simulación, se crearon los primeros simuladores de vuelo y anestesia médica.

Al haber tantas áreas que explotar en la simulación por computadora, se desarrollaron una gran cantidad de simuladores híbridos y de software desde cero, lo que llevó a la identificación de elementos comunes que fundaron las ideas base de los *frameworks* de simulación. Saber que todos los simuladores poseen al menos un módulo de sistema dinámico, construcción de modelos, motor de simulación y control de simulaciones, potenció la idea de reutilización de los mismos entre diferentes proyectos y la estandarización del desarrollo.

La simulación es práctica para casi todas las áreas del saber, sin embargo, no hace falta desarrollar un simulador para todos los casos, pues existe una variedad incontable de soluciones ya desarrolladas para casi cualquier área. No obstante, hay casos donde sí podría ser interesante desarrollar un simulador, ya sea por motivos tecnológicos, recursos o necesidades. Sin

embargo, desarrollar los simuladores desde cero implica un costo y esfuerzo mayor al que se podría requerir para el desarrollo al haber muchas funcionalidades comunes a otros desarrollos, por lo que se vuelve interesante la idea de utilizar un *framework*.

Entre los ambientes de desarrollo más relevantes se encontró DesmoJ y Simpy, donde se destacaron sus características como paradigmas y algoritmos utilizados, así como ventajas y desventajas.

Tras analizar más *frameworks*, se listaron los problemas comunes encontrados, como el sesgo hacia la simulación de sistemas discretos y el abandono de los proyectos.

En base a los problemas comunes encontrados, se estudió el futuro y el rumbo a tomar por las nuevas herramientas, obteniéndose un resultado alentador.

Para la implementación de futuras herramientas, los lenguajes y tecnologías son inciertas, pero se cree que estarán dirigidas más hacia el ámbito de manejo y manipulación de datos, y creación de aplicaciones, por lo que el estado del arte apunta a lenguajes estadísticos y de alto nivel, unidos a lenguajes de modelado de sistemas por medio de compiladores híbridos.

Se espera que en los próximos años se resuelva e integren soluciones adaptadas al hardware actual, pues el estancamiento de las velocidades de procesadores únicos obliga a los investigadores a buscar soluciones que integren el paralelismo de procesamiento. No obstante, la dificultad de manejar los recursos compartidos y el principio de causalidad que rige el universo, complican los avances en el área.

En el área de investigación han surgido nuevos modelos para caracterizar a los sistemas dinámicos que pueden facilitar y alterar el entendimiento de los futuros modelos de simulación. La combinación entre los mismos, permitirá crear herramientas más complejas y flexibles a todo tipo de simulación.

La introducción de la inteligencia artificial para la asistencia en el desarrollo de herramientas de simulación ha sido tímida, más no está descartada del todo. Mientras tanto, se han empezado a crear *frameworks* para la creación de entornos de simulación que puedan utilizar agentes de inteligencia artificial para su entrenamiento.

Con el pasado, presente y futuro a la vista, no cabe duda de que el campo de la simulación y la implementación de herramientas para la realización de las mismas, todavía tiene mucho terreno que explorar y ramas en las que puede contribuir.

El avance acelerado de las demás tecnologías es posible que fomente y se integre con la simulación para así lograr objetivos incluso superiores a la realización de proyecciones y evaluación de procesos.

Es de vital importancia seguir con la investigación en el campo para la resolución de problemas, y la inclusión de disciplinas como la Ingeniería de Software con el fin de crear nuevas herramientas siguiendo las mejores prácticas que permitan su evolución y uso extendido por la comunidad.

## REFERENCIAS


[1] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*. Pearson Prentice-Hall, 4ta ed., 2005.

[2] J. Nutaro, *Building software for simulation*. Hoboken, N.J.: John Wiley & Sons, Inc., 2011.

[3] J. Banks, *Handbook of simulation: principles, methodology, advances, applications, and practice*. Hoboken, N.J.: John Wiley & Sons, Inc., 1998.

[4] A. Law and W. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 2000.

[5] K. Rosen, "The history of simulation," in *The comprehensive textbook of healthcare simulation*, pp. 5–49, Springer, 2013.

[6] M. Aebersold, "The history of simulation and its impact on the future," *AACN advanced critical care*, vol. 27, no. 1, pp. 56–61, 2016.

[7] Rahmi M. Koç Academy of Interventional Medicine Education and Simulation, 2017. *History of Medical Simulation*. [Imagen en línea]. RMK AIMES. Disponible en: http://aimes.org//Uploads/Files/HistoryofSimulation/7.jpg [Accedido 01-08-2021].

[8] D. Riehle, *Framework design: A role modeling approach*. Diss. ETH Zurich, 2000.

[9] S. Alaggia, "Relevamiento de lenguajes de simulación," 2005.

[10] E. Briceño, "Estudio comparativo del paquete de simulación orientado a eventos discretos sympy. desarrollo de un manual de usuario con ejemplos resueltos." trabajo de fin de grado, Universidad de Los Andes, 2007.

[11] J. Y. Joshua, L. Eeckhout, D. J. Lilja, B. Calder, L. K. John, and J. E. Smith, "The future of simulation: A field of dreams," *Computer*, vol. 39, no. 11, pp. 22–29, 2006.

[12] G. I. Doukidis and M. C. Angelides, "A framework for integrating artificial intelligence and simulation," *Artificial Intelligence Review*, vol. 8, no. 1, pp. 55–85, 1994.

[13] H. J. Briegel and G. De las Cuevas, "Projective simulation for artificial intelligence," *Scientific reports*, vol. 2, no. 1, pp. 1–16, 2012.

[14] P. Gawłowicz and A. Zubow, "Ns-3 meets openai gym: The playground for machine learning in networking research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 113–120, 2019.