Conciencia Tecnológica No. 47, Enero-Junio 2014

Uso de DevC++® con Delmia Quest® para Optimizar Simulaciones

Reporte de Proyecto

Mtro. Ricardo Pérez Rodríguez[1], Dr. S. Jöns Sánchez Aguilar[2,3], Dr. Arturo Hernández Aguirre[4].

[1]Posgrado Interinstitucional en Ciencia y Tecnología, PICYT.
Centro de Innovación Aplicada en Tecnologías Competitivas, CIATEC A.C.
Omega 201, Fracc. Industrial Delta, León, Gto. México.
e-mail: rperez.picyt@ciatec.mx.

[2]Consejo Nacional de Ciencia y Tecnología, CONACYT.

[3]Dirección General de Educación Superior Tecnológica, DGEST.

[4]Centro de Investigación en Matemáticas, CIMAT, A.C.

Resumen

En este artículo se presentan los elementos básicos e imprescindibles utilizados para lograr comunicación entre cualquier algoritmo de optimización que se programe en el entorno de programación DevC++® y que sea utilizado para optimizar modelos de simulación en Delmia Quest® versión R19 o superiores. El objetivo de este artículo es facilitar el aprendizaje para comunicar ambas plataformas con el fin de sistematizar y automatizar la evaluación de las soluciones desde un punto de vista práctico.

El aporte central de este artículo es el desarrollo y validación de un tutorial para demostrar el proceso de intercomunicación entre Delmia Quest® y DevC++®.

La idea global se refiere a construir un procedimiento que enlaza las instrucciones del algoritmo de optimización programado en DevC++® la cuales son ejecutadas por el lenguaje de simulación Delmia Quest®, el cual devuelve algún parámetro con la información pertinente de nuevo al algoritmo de optimización. El procedimiento se repite iterativamente de acuerdo a las especificaciones del algoritmo de optimización. Con el presente tutorial, los usuarios podrán iniciar rápidamente la programación de sus propias rutinas de comunicación inclusive en otros lenguajes de programación como Java® ahorrado recursos en costosos cursos de capacitación.

Finalmente, 10 estudiantes de ingeniería probaron la efectividad del documento, al menos 50% de ellos consumieron entre 12 y 17 minutos para terminar exitosamente la programación del tutorial propuesto, donde tradicionalmente como factor barrera se pueden requerir horas e inclusive días *ya que no existe bibliografía accesible y clara en el idioma español*.

Palabras clave: algoritmos, modelos de simulación, optimización de simulaciones, DevC++®, Delmia Quest®.

Abstract

This paper shows some elements used to achieve communication between any optimization algorithm programmed in DevC++® to optimize any simulation model built on Delmia Quest® platform. The aim of this article is to facilitate the learning process to communicate both platforms, i.e., the goal is to systematize the evaluation of solutions from a practical approach.

The contribution of this paper is the development and validation of a tutorial to demonstrate the process of communication between Delmia Quest® and DevC++®.

The global idea refers to build a procedure that uses the instructions of the optimization algorithm programmed in DevC++® and these are executed by the simulation language Delmia Quest®, which returns a relevant output parameter to the algorithm optimization. The procedure is repeated iteratively according to the specifications of the optimization algorithm. With this tutorial, the users can quickly program their own procedures of communication even in other programming languages such as Java® without squandering resources on expensive training courses.

Finally, 10 engineering students tested the effectiveness of the document, at least 50% of them consumed between 12 and 17 minutes to successfully complete the proposed programming tutorial where traditionally as barrier factor may require hours and even days because *there is no bibliography accessible and clear in Spanish language*.

Key words: algorithms, simulation models, simulation optimization, DevC++®, Delmia Quest®.

Introducción

La simulación de eventos discretos es una herramienta probada para evaluar sistemas complejos de manufactura, tales como, producción de componentes de autopartes, ensamble de autos, fabricación de productos metalmecánicos, producción de compuestos químicos, entre otros, reduciendo el riesgo de una planeación limitada y prediciendo con suficiente anticipación impactos no solo en la programación de los trabajos sino en la producción misma [1].

Generalmente los procesos industriales tienen un alto contenido de trabajo en cada producto a elaborar, por lo que dichos procesos poseen una gran cantidad de variables a considerar al modelarlos; además, se presentan sucesos o eventos inesperados, y la variabilidad e interdependencias entre subprocesos son altas. Lo anterior justifica que resultaría fructífero modelar dichos procesos usando simulación de eventos discretos [2].

La utilización de modelos de simulación para resolver problemas y tomar decisiones ha ido en aumento. Los usuarios y quienes los construyen toman decisiones con base en los resultados que arroja el modelo según Sargent [3]. A menudo se realizan experimentos de prueba y error, o bien se depende mucho de la experiencia del personal involucrado en los procesos y fenómenos de estudio.

Existen un gran número de simuladores disponibles en el mercado, como ProModel®, Arena®, Simul8®, Delmia Quest® entre otros. Sin embargo, es importante resaltar las ventajas de utilizar Delmia Quest® con respecto a otras plataformas existentes en el mercado nacional mencionadas anteriormente. Estas son:

- Rapidez para integrar o eliminar diversos elementos y componentes en el modelo.
- Facilidad para modificar las dimensiones y características de los elementos o componentes en el modelo.
- Integración visual 3D capaz de importar y exportar entidades de variadas fuentes o para otra aplicación.
- Flexibilidad para utilizar elementos que permiten realizar la transferencia y el manejo de materiales de manera práctica y realista.
- Accesible y asequible a cualquier institución educativa a través de licencias académicas que ofrece el proveedor a nivel nacional para realizar actividades de docencia e investigación. Además, los estudiantes interesados pueden acceder por medio del patrocinio de la institución educativa a la que pertenecen, obteniendo así los mismos beneficios que la institución.
- Posibilidad de interactuar con otras aplicaciones o lenguajes a través de interfaces construidas por el propio usuario de manera práctica.

La optimización de simulaciones está basada en ver al modelo de simulación como una "caja negra" que evalúa la función a optimizar. Bajo este enfoque el algoritmo de optimización utilizado escoge un conjunto de valores para los parámetros de entrada en el modelo de simulación (por ejemplo factores o variables de decisión) y utiliza la respuesta generada obtenida en el modelo para tomar decisiones sobre la selección del siguiente conjunto de valores para los parámetros de entrada a probar.

Los líderes actuales en lenguajes de simulación comerciales proporcionan herramientas de optimización a sus usuarios dentro del simulador. Hoy en día casi todos los lenguajes de simulación comerciales contienen un módulo de optimización que realiza algún tipo de búsqueda de los mejores valores para los parámetros de entrada y no solamente la estimación estadística tradicional. Sin embargo dichos lenguajes de simulación comercial requieren una considerable cantidad de habilidades técnicas por parte del usuario, en ocasiones requieren un tiempo computacional excesivo y solamente ofrecen un método de optimización. Por lo que en esta investigación se utiliza Delmia Quest®, pues permite reducir errores en la implementación, tiempo en la construcción del modelo significativamente y flexibilidad de comunicación con diferentes lenguajes de programación a fin de evaluar el desempeño de diferentes métodos de optimización en la solución.

Un inconveniente que tiene la plataforma Delmia Quest®, son los altos costos de capacitación para su aprendizaje que todo usuario puede requerir. La capacitación tiene un alcance elemental y no siempre responde a las necesidades de modelado para casos particulares. Además, las referencias bibliográficas actuales se encuentran escritas en el idioma inglés, lo que podría dificultar la rápida difusión del uso de la plataforma. Por estas razones el presente documento será un catalizador para acelerar el aprendizaje de los métodos más ampliamente utilizados y probados para lograr comunicación entre algoritmos de optimización programados en DevC++® o alguna variante de lenguaje C y C++ y modelos de simulación de eventos discretos en áreas tales como manufactura, logística, tráfico, entre otras. El aporte central de este artículo es el desarrollo y validación de un tutorial para demostrar el proceso de intercomunicación entre Delmia Quest® y DevC++®.

En el tutorial se describen los mecanismos más simples, básicos e imprescindibles utilizados para ayudar a desarrollar dichas comunicaciones a fin de obtener optimización de simulaciones.

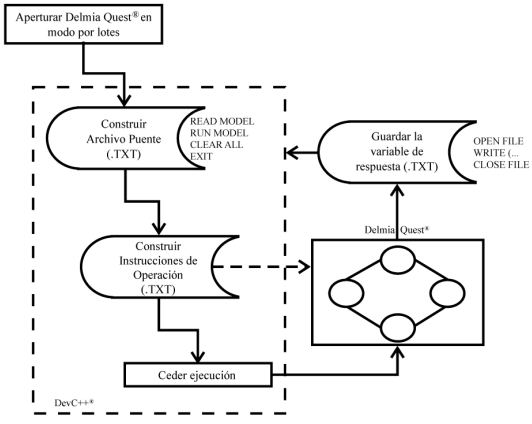
En este artículo se muestra cómo comunicar e interactuar instrucciones de algún algoritmo de optimización programado en DevC++® y modelos de

simulación de eventos discretos construidos en Delmia Quest®, el cual es un lenguaje de simulación que permite este tipo de escenarios [4].

Este tutorial no está relacionado con fundamentos de simulación. Definiciones importantes sobre modelos y diversos tipos de ellos pueden encontrarse en Bisschop [5]. Además, la bibliografía recomendada sobre simulación de eventos discretos sería: Harrell [6], García *et al*. [7] y Banks *et al*. [8]. Asimismo, existe una referencia introductoria para el software Delmia Quest® en Barnes [9]. Se recomienda que los lectores estén familiarizados con estos lenguajes (DevC++® y Delmia Quest®) para aprovechar al máximo este tutorial.

Comunicación DevC++® y Delmia Quest®

Teniendo un modelo de simulación construido en Delmia Quest® es posible realizar optimización de simulaciones y comunicarlo con algún algoritmo de optimización definido para ello que haya sido programado en lenguaje C, C++ o Java®.
La Figura 1 detalla el proceso global de comunicación que se detalla posteriormente.



Paso 1. Construir una copia del archivo ejecutable.
Primeramente se debe construir una copia del archivo ejecutable del simulador Delmia Quest® con extensión .BAT que servirá para realizar su ejecución cuando este sea llamado por el algoritmo de optimización. Una vez hecho esto se debe modificar la línea de comandos relacionada a la forma en que será abierto el simulador. La modificación puede consultarse en la referencia [10]. A continuación se describe el cambio a realizar en la línea de comandos.

Encontrar la siguiente línea en el archivo

```
start/max %DENEB_PRODUCT%.exe %1 %2 %3 %4 %5 %6 %7 %8 %9
```

Modificar dicha línea por la siguiente

```
quest -b < %1
```

La letra 'b' indicada en la línea anterior condiciona al simulador que será aperturado en modo por lotes y el número 1 implica que algún archivo contendrá las instrucciones de ejecución para el simulador. Cuando se realiza esta modificación, la posibilidad de abrir a Delmia Quest® en modo por lotes se logra de manera tal que este espera a ser llamado por el sistema operativo y no por el usuario directamente.

Paso 2. Construir un archivo puente.
La interacción entre el algoritmo de optimización y el modelo de simulación puede realizarse cuando este último ejecuta una serie de comandos definidos por el lenguaje de control por lotes BCL (por sus siglas en inglés Batch Control Language) que tiene provisto Delmia Quest®. Los comandos a ejecutar por el modelo de simulación deben ser escritos en un archivo con extensión TXT. Los comandos que deben ser programados en el algoritmo de optimización para lograr la integración con el modelo de simulación al menos son:

-Comando de apertura del modelo y un ejemplo

```
READ MODEL 'c:/deneb/QUESTLIB/MODELS/mi_primer_modelo.mdl'
```

-Comando de ejecución (corrida) de simulación y un ejemplo

```
RUN 100000
```

-Comando de cierre del modelo

```
CLEAR ALL
```

-Comando de cierre del simulador

```
EXIT
```

Nota: este último comando cierra la aplicación del simulador abierto previamente en modo por lotes y devuelve la ejecución en tiempo real al lenguaje DevC++® para continuar el proceso.

Paso 3. Programar instrucciones de operación.
La interacción entre el algoritmo de optimización y el modelo de simulación debe ser prevista desde el lenguaje de programación DevC++®.
Cuando se tiene definido el conjunto de valores de los parámetros de entrada para el modelo de simulación, estos deben ser guardados en archivos con extensión TXT. Los elementos involucrados con dichos parámetros de entrada en el modelo de simulación tales como locaciones, máquinas, buffers, operadores, partes, entidades, fuentes y sumideros deben ser asociados en su proceso lógico con los archivos con extensión TXT mencionados.

A modo de ejemplo, la siguientes líneas en DevC++® pueden ser programadas para almacenar la secuencia de producción de partes que serán creadas en el modelo

de simulación por una fuente constructora de partes. Así las partes se construyen en base a la secuencia definida por el algoritmo de optimización. La secuencia de partes funge como un parámetro de entrada para el modelo de simulación.

```
FILE *schptr;
schptr = fopen("c:/deneb/QUESTLIB/SCHEDULES/mi_secuencia.txt", "w")
fprintf(schptr, "%s%s%s\n", "ORDEN ","MATERIAL ","CANTIDAD ");
fprintf(schptr, "%d%s%s%s%d\n",1," ","Material A "," ",50);
fprintf(schptr, "%d%s%s%s%d\n",2," ","Material B "," ",30);
fprintf(schptr, "%d%s%s%s%d\n",3," ","Material C "," ",10);
....
....
fclose(schptr);
```

Paso 4. Ceder la ejecución al simulador.
El mejor momento de ceder por parte del algoritmo de optimización la ejecución al simulador en tiempo real es cuando este último debe entregar la respuesta que se requiere para tomar decisiones por el algoritmo de optimización. Para poder realizar esto, a continuación la siguiente línea cede la ejecución del algoritmo al simulador en tiempo real por medio de una ventana de comandos, la cual se direccionará al archivo puente mencionado en el paso 2 de este tutorial.

```
system("c:\\deneb\\quest\\mi_copia_quest.bat C:\\deneb\\QUESTlib\\
BCLMACROS\\mi_archivo_puente.txt");
```

Paso 5. Obtener la variable de respuesta.
Una vez que la ejecución del simulador ha sido realizada. Este debe guardar el valor de la variable de respuesta esperado en algún archivo con extensión TXT. Las siguientes líneas ofrecen un ejemplo de almacenar la cantidad de partes que han salido de un elemento en el modelo al momento de finalizar la corrida de simulación.

```
OPEN FILE 'c:/deneb/QUESTLIB/SCHEDULES/mi_archivo_respuesta.txt'
FOR OUTPUT AS 1
WRITE(#1,PART_OUT_COUNT(any, celem),cr)
CLOSE #1
```

Los pasos anteriores deben ser integrados y estructurados en el algoritmo de optimización a fin de que pueda ser ejecutado el modelo de simulación tantas veces se necesite y así obtener las respuestas necesarias.

Resultados y Discusión

El tutorial previamente descrito fue puesto a prueba y analizado por 10 estudiantes del posgrado PICYT en la sede del CIATEC.

El Gráfico 1 muestra el tiempo en minutos que requirió cada estudiante para implementar exitosamente dicho tutorial.

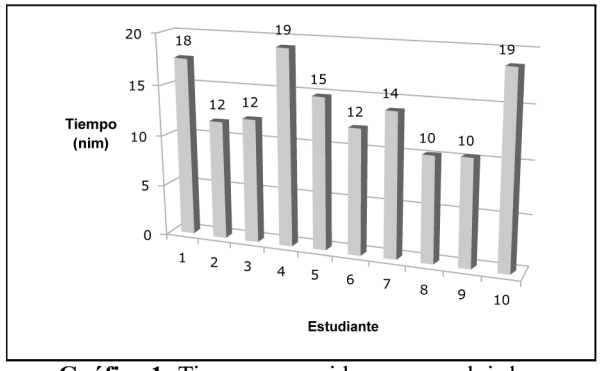


Gráfico 1. Tiempo requerido para concluir la comunicación .

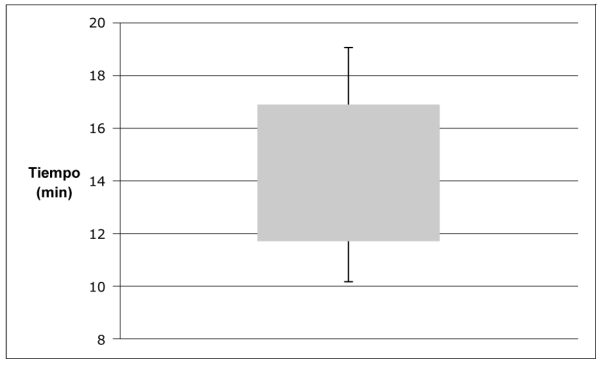


Gráfico 2. Diagrama de caja para el Gráfico 1.

El Gráfico 2 muestra que los estudiantes consumieron un promedio de 14 minutos con una desviación estándar de 3.36 minutos. Asimismo, se interpreta que al menos el 50% de los estudiantes se tardaron entre 12 y 17 minutos en reconstruir exitosamente los pasos descritos en este documento.

Por lo anterior, se observa una diferencia significativa que este tutorial ayuda a disminuir el tiempo requerido de los estudiantes para aprender a comunicar algoritmos de optimización programados en DevC++® con modelos básicos de simulación en la plataforma Delmia Quest® por que tradicionalmente se pueden requerir horas e inclusive días para dicha capacitación, *ya que no existe bibliografía accesible y clara en español.*

Por último los resultados de este tutorial mantienen consistencia con los obtenidos en Pérez *et al*. [11], reduciendo el tiempo requerido en la curva de aprendizaje de este simulador.

Conclusiones

Consideramos que la simulación de eventos discretos es útil para resolver problemas en sistemas donde existen interdependencias y variabilidad dentro del mismo sistema. Sin embargo, pese a estas evidencias de éxito aún se hallan algunas barreras para que su uso sea habitual. Entre las barreras más comunes están:

- Escaso material documental especializado para usuarios potenciales de habla hispana.
- Altos costos de entrenamiento lo que se hace difícil el acceso a capacitación actualizada.
- Inversión de tiempo considerable para el entendimiento de nuevas plataformas.
- Nula difusión de nuevas plataformas de simulación en el entorno industrial.
- Profesionistas sin entrenamiento en herramientas de optimización.

El presente artículo es una propuesta para disminuir algunos de los factores barrera tales como el escaso material documental especializado escrito en el idioma español, tiempo excesivo en la comprensión de nuevas plataformas y los altos costos de capacitación que ofrecen los distribuidores autorizados. Los autores del presente trabajo consideran que con la difusión de este tipo de tutoriales la simulación de eventos discretos debiera ser utilizada en la industria mexicana de formas más común.

Finalmente, los proponentes del tutorial están convencidos que con este tipo de documentos, se puede difundir la mecánica de construcción y comunicación de modelos de simulación en la plataforma novedosa Delmia Quest®, reduciendo el tiempo de aprendizaje de los usuarios de este software.

En un futuro no muy lejano el entrenamiento en técnicas de optimización y simulación a profesionistas en el ámbito industrial debe ser considerado una prioridad y una necesidad inevitable para garantizar mejores soluciones para cualquier proceso industrial.

Agradecimientos

Los autores agradecen las sugerencias y el apoyo incondicional de Jonathan Fournier y a Martin Barnes. Del mismo modo, un reconocimiento a todas las personas que revisaron en numerosas ocasiones este tutorial.

Referencias

[1] Taylor, SJE., Robinson, S., (2006), "So where the next? A survey of the future for discrete-event simulation", *Journal of Simulation* 0, pp. 1-6.

[2] Pérez, R., Sánchez, J., Gómez, J., Ochoa, C., (2010), "Best Practices for modeling the manufacture of steel doors using Quest®", *IIE-IERC Annual Conference & Expo 2010* (Cancún, México., 5-9 Junio, 2010), pp. 140.

[3] Sargent, R.G., (1996), "Verification and Validation of Simulation Models", *Proceedings of 1996 Winter Simulation Conference*, (Coronado, California, USA., 8-11 Diciembre, 1996), pp. 55-64.

[4] Dagpunar, J.S., (2007), *Simulation and Monte Carlo*, John Wiley & Sons, Ltd. (England).

[5] Bisschop, J., (2007), *AIMMS Optimization Modeling*, Paragon Decision Technology B.V. (USA).

[6] Harrell, C.R., (1995), *Simulation using ProModel®*, McGraw-Hill (EUA).

[7] García, E., García, R., Cárdenas, L., (2006), *Simulación y Análisis de Sistemas con ProModel®*, Pearson Educación (México).

[8] Banks, J., Carson II, J.S., Nelson, B.L., Nicol, D. M., (2001), *Discrete-Event System Simulation*, Tercera edición, Prentice-Hall (USA).

[9] Barnes, M.R. (1997), "An Introduction to Quest", S. Andradóttir, K.J. Healy, D.H. Withers, and B.L. Nelson (Eds.) *Proceedings of the 1997 Winter Simulation Conference*, (Atlanta, Georgia, USA., 7-10 Diciembre, 1997), pp. 619-624.

[10] Delmia Quest® User Manual 5_17, (2006), Delmia Corporation, Auburn Hills, MI.

[11] Pérez, R., Jöns, S., Hernández, A., Young, D., (2011), "Tutorial de Simulación Básica utilizando Quest®", *Conciencia Tecnológica* 41, pp. 28-34.

Recibido: 26 de agosto de 2013

Aceptado: 24 de enero de 2014