Traductor de Formalismos para Modelado y Simulación de Sistemas Híbridos

Pedro Rodríguez and Rodrigo Castro

Departamento de Computación amd Instituto de Ciencias de la Computación,
Facultad de Ciencias Exactas y naturales, Universidad de Buenos Aires.
{perodriguez,rcastro}@dc.uba.ar

Resumen El estudio moderno de sistemas complejos interdisciplinarios mediante modelado y simulación suele requerir la capacidad de expresar dinámicas híbridas (continuas y discretas). Pocas herramientas ofrecen esta posibilidad garantizando a la vez un tratamiento correcto y eficiente de interacciones entre dichas dinámicas ya que es un problema no trivial. Desarrollamos un traductor automático de formalismos System Dynamics a DEVS, valiéndonos de los estándares XMILE (XML Interchange Language) y DEVSML (DEVS Modeling Language) diseñados con el propósito de intercambiar modelos entre simuladores de uno y otro formalismo, aunque por separado. Mediante nuestro traductor, modelos System Dynamics y DEVS pueden coexistir e interactuar bajo un mismo simulador DEVS. Mostramos la utilidad del traductor en una aplicación de modelo híbrido complejo combinando dinámicas macroeconómicas y modelos de opinión.

Keywords: Simulación, Sistemas híbridos, Ststem Dynamics, DEVS

1. Introducción

SystemDynamics (SD) es una metodología de modelado desarrollada desde finales de los años 50 en la MIT School of Industrial Management por el ingeniero Jay Wright Forrester [1].

SD se desarrolló como una estrategia para el modelado inductivo, es decir, para modelar sistemas cuyos componentes principales son gobernados por metaleyes desconocidas. Sistemas típicos que se describen con SD son aquellos cuya complejidad no permite una descripción general adecuada por medio de principios basados en el dominio del problema. Por ejemplo, en sistemas demográficos, biológicos, económicos, ambientales, de salud y sistemas de gestión. Así, la estructura y el comportamiento de un sistema se construyen esencialmente a partir de observaciones empíricas. Algunos casos de estudio en la literatura que vale la pena destacar son: en la simulación de portfolios de negocios [2], construcción de autopistas en China [3], proceso de desarrollo de productos [4] y el famoso modelo de límites de crecimiento (*Limits to growth*) World3 [5], entre otros.

De forma paralela se desarrolló otra metodología de simulación, introducida por el matemático Bernard Zeigler a partir de la creación del formalismo DEVS

[6]. DEVS es más flexible y poderoso que SystemDynamics y puede ser utilizado como formalismo universal para el modelado y simulación de sistemas.

Sin embargo, si bien ha habido algunos intentos de combinar SD con simulación de eventos discretos (DES) (ver [7]) no se le ha dado demasiada importancia a lograr la unificación de la representación de modelos SD y DES bajo un mismo formalismo. Por su capacidad de representar tanto dinámicas discretas como dinámicas continuas, el formalismo DEVS aparece como el candidato ideal para encarar dicho problema.

En este trabajo nos proponemos implementar un *software traductor* de modelos SD en modelos DEVS que permita la combinación de ambas metodologías de modelado.

2. Motivación y formulación del problema

Existe una gran variedad de herramientas de simulación basadas en el formalismo System Dynamics. Algunos ejemplos son: Dynamo [8], Stella [9], Vensim [10], Simile [11], PowerSim [12]. De igual manera, se pueden encontrar muchas herramientas de simulación basadas en el formalismo DEVS: PowerDEVS ([13]), CD++ ([14]), DEVSJAVA ([15]) ó PythonDEVS ([16]) entre otras. Sin embargo, creemos que en parte por razones históricas y en parte por su estrategia de apartarse de los detalles de la matemática subyacente y enfocarse en la sencillez visual de especificación de modelos, SD alcanzó un nivel de adopción mucho más alto.

Hasta hace pocos años los entornos de simulación para sistemas SD eran incompatibles entre sí. Es decir, no había un formato estándar para representar los modelos que éstos permitían simular, a pesar de que su paradigma visual sí lo era. Es decir, dado un modelo de un mismo sistema, para simularlo con herramientas distintas era necesario reescribirlo cada vez.

Este estado de la situación se modifica en el año 2013 cuando la IEEE (Institute of Electrical and Electronics Engineers) en conjunto con la empresa IBM, establecen un estándar para representar modelos SD utilizando XML denominado XMILE . El estándar permite que los modelos puedan intercambiarse entre las distintas herramientas sin la necesidad de una reescritura.

Históricamente también hubo esfuerzos por encontrar un formato estándar para la representación de modelos DEVS con el fin de poder intercambiar modelos entre las distintas herrramientas de simulación. En esta tesis nos referiremos a los trabajos desarrollados en DEVSML [17].

Dada la vasta disponibilidad de modelos SD desarrollados durante décadas, es relevante contar con la posibilidad de reutilizar y hacer coexistir modelos SD en el universo de simulación a eventos discretos, para el cual DEVS representa un formalismo universal.

En la literatura han habido esfuerzos por lograr la integración entre SD y DES, a fin de enriquecer la variedad de modelos que se puedan simular. En esos tipos de integraciones (por ejemplo, ver [18]), las características de DES se

utilizan para representar elementos del sistema que no se capturan con el nivel de granularidad suficiente dentro del modelo SD [7].

La simulación híbrida (definida como el modelado que combina dos de los siguientes métodos: SD, DES y simulación basada en agentes) ha experimentado un crecimiento casi exponencial en popularidad de las últimas dos décadas. Es por esta razón que este tipo integraciones se están volviendo muy necesarias para el análisis de sistemas complejos, como es en el área de investigación operativa.

Sin embargo al momento de realizar este tipo de integración de los formalismos SD y DEVS se deben tener en cuenta por lo menos las siguientes consideraciones [18]:

- Elegir las partes del modelo SD que se desean pasar a modelos DES y generar dichos modelos. Estos modelos se ejecutarán de manera independiente
- Definir el particionamiento del tiempo (*time bucket*) a utilizar para sincronizar entre el modelo SD y los modelos DES. Lo cual es una decisión que afecta a la precisión y eficiencia del modelo
- Convertir los valores de la variables compartidas entre modelos, tanto de SD a DES como de DES a SD
- Crear un mecanismo o proceso para el intercambio de las variables entre los modelos en tiempo de simulación

Estos requerimientos no estarían presentes si se utilizara un único modelo que sea capaz de integrar ambos tipos de formalismos

Una alternativa a la integración entre SD y DES que al momento no ha sido explorada es mediante traducción de modelos SD a DES utilizando el formalismo híbrido DEVS.

Según nuestro relevamiento al inicio del desarrollo de este trabajo, no existía ningún trabajo que presentara un algoritmo para traducir entre los estándares en formato XML de SD (XMILE) y DEVS (DEVSML).

Dicho algoritmo permitiría la reutilización de modelos en SD ya implementados y la interconexión de estos con otros modelos DEVS más avanzados. Así, se podría obtener lo mejor de ambos mundos para la creación de modelos de sistemas dinámicos: por un lado, la comodidad de generación de módulos (o partes) de un modelo en herramientas avanzadas que trabajen con SystemDynamics y por el otro, contar con las todas las ventajas provistas por el formalismo integrador DEVS.

3. Objetivos

El objetivo de esta tesis es la implementación de un traductor que permita transformar un modelo SD utilizando el estándar XMILE en un modelo DEVS expresado en un formato derivado del formato DEVSML, al que llamaremos muDEVSML. El traductor facilitará la reutilización de modelos preexistentes ya estudiados en la literatura y permitirá su modificación y adaptación a modelos nuevos, aprovechando todas las ventajas de DEVS.

Un modelador especialista en SD podrá exportar sus modelos de forma automática a DEVS, sin tener que aprender las bases de dicho formalismo, ni ninguna herramienta DEVS adicional. Se logrará así la integración de modelos desarrollados en herramientas que implementen SystemDynamics con sistemas de eventos discretos generalizados (que son representables mediante el formalismo DEVS).

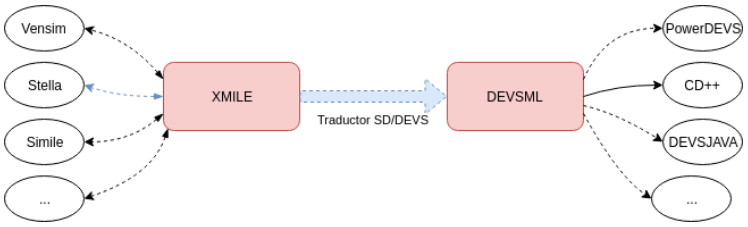

Figura 1. Capacidades del traductor: generar archivos en la especificación muDEVSML partiendo de una representación XMILE del modelo SD implementado originalmente en *Stella*. La flecha azul más gruesa representa la implementación *core* de esta tesis, mientras que la negra sin puntear representa nuestra implementación de la transformación de muDEVSML al lenguaje de programación de la herramienta CD++, necesaria para testear los resultados de los modelos traducidos. En línea negra punteada quedan representadas otras implementaciones posibles, y en línea azul punteada implementaciones ya existenes hechas por terceros.

El traductor deberá ser capaz de cumplir con los siguientes requisitos:

1. Interpretar y traducir una variedad de modelos SD clásicos, posibilitando que la simulación del modelo traducido replique los resultados de la simulación generada para éstos por simuladores comerciales utilizados por la comunidad SD
2. Ser fácilmente extensible, permitiendo la traducción de nuevas funcionalidades que puedan agregarse a *XMILE* de forma rápida y sencilla
3. Permitir la interconexión de modelos preexistentes que no sean representables en SD (por ejemplo, modelos de autómatas celulares) con modelos de ecuaciones diferenciales ordinarias que sí puedan ser expresados en SD

4. Simulación de sistemas dinámicos

La simulación es una metodología para estudiar sistemas complejos. El proceso de simulación comienza siempre con un problema que se quiere resolver o comprender. En muchos casos este proceso comienza con la observación de un sistema real a partir del cual se identifican entidades y sus interacciones a partir de las cuales se construye un modelo del mismo. Una vez que el modelo está definido, se experimenta con él mediante un simulador, el cual lo ejecuta. Los

resultados de la simulación son comparados con los datos que se tienen del sistema real a fin de validar el modelo. En la mayoría de los casos no es posible crear un modelo que tenga en cuenta todos los aspectos del sistema real. Es por ello que se define un marco experimental, en el cual se delinean los objetivos del modelado y el alcance del modelo. La figura 2 muestra la relación entre estos conceptos.

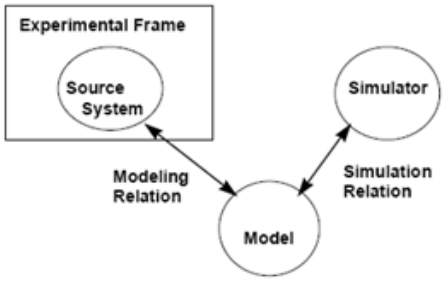

Figura 2. Relaciones de modelado y simulación: Se cuenta con un Modelo, y con un Sistema origen encuadrado en un Marco Experimental, que tiene una relación de modelado con el modelo. Por otro lado con un Simulador, que tiene un relación de simulación con el Modelo.

Formalmente podemos definir un sistema como una entidad real o artificial que representa una parte de una realidad y está restringida por un entorno. Puede decirse que un sistema es un conjunto ordenado de objetos lógicamente relacionados que atraviesan actividades, interactuando para cumplir los objetivos propuestos [19]. Por otro lado, un modelo es una representación inteligible (abstracta y consistente) de un sistema. Se pueden distinguir dos grandes grupos de métodos para modelar sistemas complejos a partir de un sistema real: los modelos analíticos y los modelos basados en simulación.

Los modelos analíticos están basados en el razonamiento deductivo y permiten obtener soluciones generales al problema. Un formalismo analítico muy difundido para el modelado de problemas es el uso de ecuaciones diferenciales. Un problema de estos métodos es que al considerar sistemas complejos éstos (con pocas excepciones) serán analíticamente intratables y numéricamente prohibitivos de evaluar. La utilización de estos métodos requiere de la simplificación del modelo, alejándolo demasiado de lo que sucede en la realidad [19].

Los modelos basados en simulación suelen utilizarse en los casos en que no es posible o conveniente modelar el sistema mediante métodos analíticos. En este tipo de modelos no se intenta buscar una solución general al problema, sino que se buscan soluciones particulares. Algunos problemas con estse tipo de modelos son el tiempo que lleva su desarrollo, la necesidad de validar los resultados y de recopilar datos para poder reproducir el comportamiento del sistema. Algunas

ventajas de realizar simulaciones en lugar de experimentación directa sobre el sistema son [19]:

Repetitividad: Una simulación puede ser realizada tantas veces como sea necesario.
Experimentación Controlada: La simulación de un sistema no afecta al sistema real.
Compresión del tiempo: En muchos casos una simulación demanda menos tiempo en su ejecución que la realización de un experimento en el sistema real.
Análisis de sensibilidad: El modelador decide cuál es el marco experimental (leyes que rigen la simulación) de cada experimento de simulación.
Automatización: se puede automatizar la simulación para hacer una experimentación exhaustiva y así encontrar los resultados deseados.

Como muestra la figura 3, los modelos pueden clasificarse básicamente en cuatro categorías, a partir de la observación de su base de tiempo y el valor de sus variables.

Con respecto a la base de tiempo, pueden ser de:

- Tiempo continuo, donde el tiempo evoluciona de forma continua
- Tiempo discreto, donde el tiempo avanza por saltos de un valor entero a otro

Respecto a los conjuntos de valores de las variables descriptivas, pueden ser de:

- Estados o eventos discretos, donde el valor de las variables pertenecen a conjuntos discretos
- Continuos, las variables son números reales
- Mixtos, variables tanto discretas como continuas

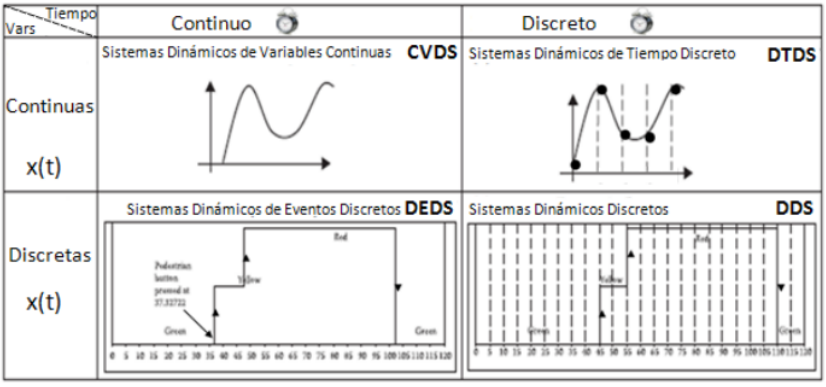

Figura 3. Clasificación de modelos según su base de tiempo y el valor de sus variables [19]

Por ejemplo, los modelos descriptos con ecuaciones diferenciales se encuentran dentro del cuadrante superior izquierdo, mientras que dentro del superior derecho se discretiza el tiempo obteniendo ecuaciones en diferencia, es decir, aproximaciones a tiempo discreto de las ecuaciones diferenciales. En el cuadrante inferior derecho (variables y tiempo discretos) se encuentran modelos como las máquinas de estados finitos, redes de Petri, autómatas celulares y otros. Finalmente en el inferior izquierdo se muestran los paradigmas de variables discretas a tiempo continuo. Tales sistemas reciben el nombre de **Sistemas Dinámicos de Eventos Discretos** (**DEDS** – Discrete Events Dynamic Systems). En este trabajo utilizaremos dos formalismos matemáticos: System Dynamics, típicamente utilizado para modelar sistemas de ecuaciones diferenciales, utiliza **variables continuas** y **tiempo discreto**; y por otro lado DEVS, que es utilizado para modelar sistemas generales y si bien podría categorizarse como un DEDS, DEVS también puede representar variables continuas. Diremos que utiliza **variables discretas y continuas** y **tiempo continuo**.

5. Relación entre Integradores SD y DEVS: idea motivadora

Como ya vimos, la estructura elemental clave de SD consiste en un stock y sus flujos de entrada y salida controladas. Para concebir una representación DEVS de esta estructura un primer paso razonable es encontrar una unidad básica de comportamiento en DEVS que sea equivalente al submodelo de "stock-and-flow"de SD. En términos teóricos lo que se desea es establecer un isomorfismo SD-a-DEVS en el nivel de Acoplamiento de Componentes.

En la figura 2.3 se puede ver un posible modelo basado en DEVS que es isomórfico con el submodelo de stock y flujo de SD. Se ven cuatro modelos atómicos DEVS: un integrador dinámico (por ejemplo, usando QSS) y tres funciones sin memoria: una función derivada (F_der), una función tasa de incremento (F_incr) y una función de tasa de decremento (F_decr).

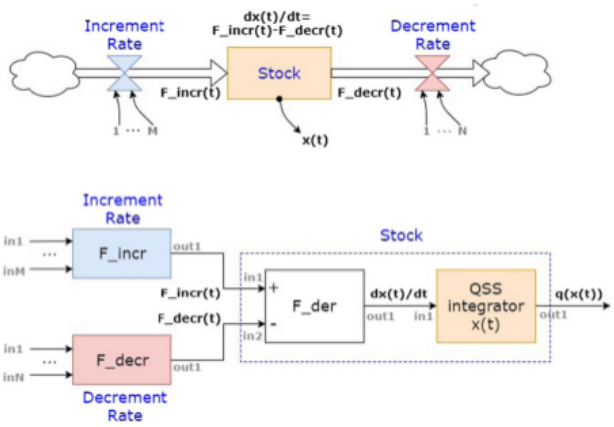

Figura 4. La estructura básica de elementos **stock** y **flow**. SD (arriba) y DEVS (abajo).

6. Algunos Casos de Estudio: Modelo SIR

Una clase de modelos ampliamente conocidos en el mundo SD es el modelo de contagio tipo *Modelo SIR*. En la 5 se muestra una representación gráfica generada en Stella de un *Modelo SIR* (Susceptible-Infected-Recovered). En la ecuación 1 se detalla el modelo en formato de ecuaciones diferenciales (ODEs).

$$
\begin{cases}
Susceptibles_0 = 995 \\
Susceptibles(t) = Susceptibles(t - dt) - Succumbing \cdot dt \\
Infected_0 = 5 \\
Infected(t) = Infected(t - dt) + (Succumbing \cdot dt - Recovering \cdot dt) \\
Recovered_0 = 0 \\
Recovered(t) = Recovered(t - dt) + Recovering \cdot dt \\
InfectionRate = 0{,}3 \\
TotalPopulation = 1000 \\
Duration = 5 \\
Succumbing(t) = \dfrac{Susceptibles(t) * Infected(t)}{InfectionRate * TotalPopulation} \\
Recovering(t) = Infected(t)/Duration
\end{cases}
\tag{1}
$$

En la 5 pueden verse tres parámetros (constantes) en el modelo, representados mediante los **aux**: el total de la población, **TotalPopulation**, la tasa de infección, **InfectionRate**, y la duración de la infección, **Duration**. Los **stock** del sistema son: la cantidad de población aún sana, **Susceptibles**(t), la cantidad de población infectada, **Infected**(t), y la cantidad de población recuperada **Recovered**(t). Hay dos **flow**: la *tasa de contagio*, **Succumbing**(t), y la *tasa de*

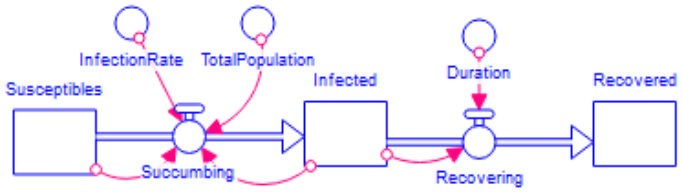

Figura 5. Modelo gráfico del *Modelo SIR* generado en Stella

recuperación, **Recovering**. Según este modelo, una vez que alguien se recupera, ya no puede contagiarse otra vez.

En la figura 6 se observa la evolución de las tres variables más importantes del modelo (**Susceptibles**(t), **Infected**(t) y **Recovered**(t)), tanto para el modelo en SystemDynamics simulado en `Stella` y como para el modelo DEVS obtenido de la traducción simulado en `CD++` en el en el intervalo de tiempo $[0,14]$. Observándose igual comportamiento y valores similares en ambos modelos.

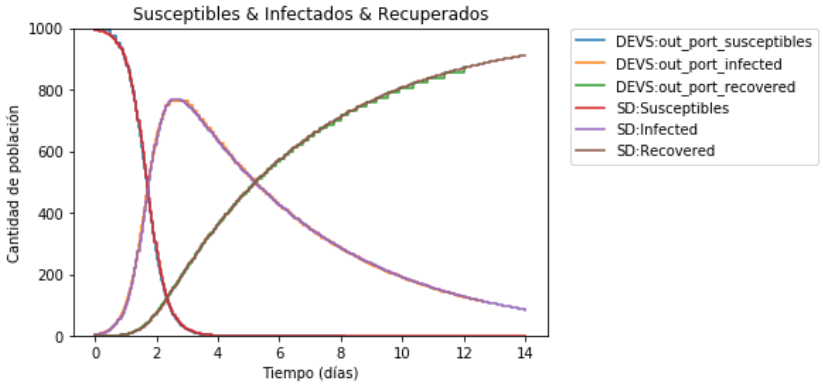

Figura 6. Evolución de **stock**s del *Modelo SIR* para el modelo simulado en Stella y en CD++.

En la figura 7 hacemos un corte del intervalo de tiempo $[1,3]$ de la simulación para mostrar la diferencia entre la forma de aproximar las curvas de cada una de los **stock** utilizando la herramienta `Stella` (discretiza el tiempo) y `CD++` (discretiza el valor de las variables). En particular, se puede comparar las líneas naranja (**out_port_infected**, `CD++`) y violeta (**Infected**, `Stella`). Y ver cómo para una misma variación en el valor de dicha variable, en un caso se hacen dos saltos y en otro solamente uno.

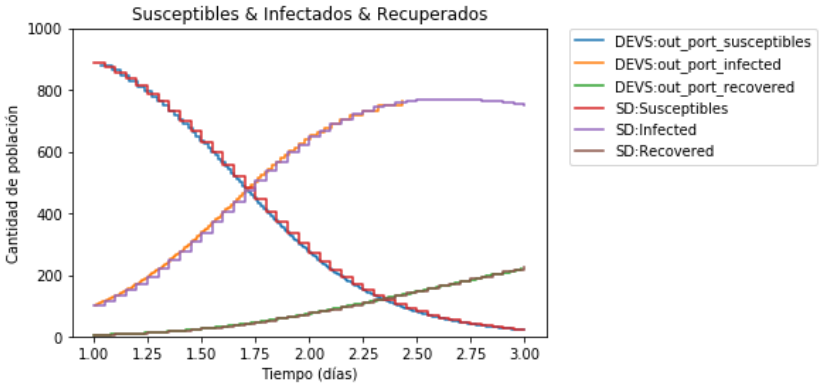

Figura 7. Evolución de **stock**s del *Modelo SIR* en el intervalo de tiempo $[1{,}3]$ para el modelo simulado en Stella y en CD++.

7. Modelo STEP: limitaciones y problemas con SD

Para entender mejor uno de los beneficios relevantes que ofrece DEVS en comparación a SD, diseñamos un nuevo modelo al que llamamos *Modelo STEP*. El objetivo es mostrar como DEVS maneja de manera eficiente y correcta la combinación de dinámicas continuas y discretas. En Fig. 8 se puede ver el modelo gráfico en Stella.

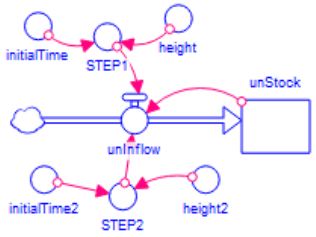

Figura 8. Teacup + Step Function

Para el experimento, configuramos utilizamos los siguientes valores para los parámetros: $DQ = 0{,}01$, $DREL = 0{,}01$ y $DT = 0{,}01$. De esta forma, se va a hacer evidente un caso de no detección de los eventos STEP en Stella. En la Fig. 9 se muestra el instante de tiempo en que los generadores STEP accionan sobre el sistema.

En 2 se detalla el sistema de ecuaciones correspondiente a este modelo.

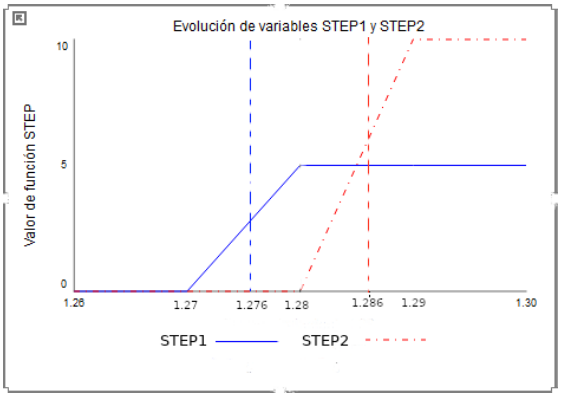

Figura 9. Funciones STEP. Stella no es capaz de detectar los instantes de tiempo exactos en que suceden los eventos STEP1 y STEP2 ($t = 1{,}276$ y $t = 1{,}286$ minutos, marcados con líneas verticales azul y roja punteadas respectivamente)

$$\begin{cases} initialTime = 1{,}276 \\ initialTime2 = 1{,}286 \\ height = 5 \\ height2 = 10 \\ STEP1 = STEP(initialTime, height) \\ STEP2 = STEP(initialTime2, height2) \\ unStock_0 = 1 \\ unStock(t) = unStock(t - dt) + unInflow * dt \\ unInflow = unStock + STEP1 + STEP2 \end{cases} \tag{2}$$

En este ejemplo se hace evidente una debilidad importante de la simulación de sistemas en SD: cuando se combina con la aparición de eventos discretos no se puede saber *a priori* qué valor configurarle a DT de modo tal de evitar errores.

Para distintos parámetros de un mismo modelo podría ser necesario configurar valores muy distintos a DT para que la simulación sea lo suficientemente precisa en los momentos de aparición de los eventos. Además, no existe ningún cálculo que provea un valor para DT (sólo se pueden usar heurísticas, que pueden ser en algunos contextos muy molesto o restrictivo de utilizar).

En la 10 se puede apreciar el desfasaje que se genera en las curvas de la variable **unStock** para el modelo simulado con CD++ (línea azul) en comparación con el simulado en Stella (línea naranja). Este desfasaje puede acumular error en el tiempo de manera no controlada, haciendo que la simulación SD termine dando resultados inaceptablemente incorrectos.

8. Implementación de un modelo híbrido complejo

En esta última sección nos proponemos poner a prueba la utilidad del sistema de traducción para un caso más complejo que solamente traducir de SD a

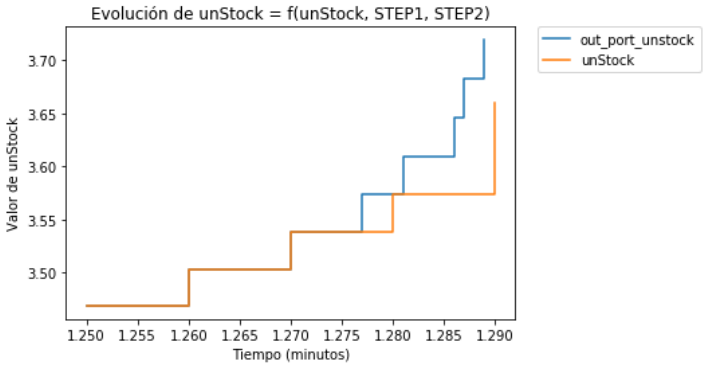

Figura 10. *Modelo STEP*: se hace zoom en el intervalo $t = [1{,}25, 1{,}29]$. Puede observarse cómo ambas trayectorias evolucionan a la par hasta que en t=1.276 y t=1.286 la línea azul (correspondiente a la simulación de CD++) detecta los eventos STEP de manera exacta, mientras que esto no sucede en el caso de la línea naranja (simulación de Stella)

DEVS. Partiendo de dos modelos preexistentes sobre dinámicas socioeconoómicas, intentaremos obtener un nuevo sistema que vincule sus dinámicas utilizando interconexión de puertos de entrada/salida y definición de nuevas dinámicas.

Uno de los modelos de partida describe la macroeconomía de un país. El otro es un modelo de opinión, que representa dinámicas de influencia mutua y cambio de preferencia entre votantes de una sociedad.

Para el desarrollo del nuevo modelo híbrido nos planteamos la siguiente hipótesis de trabajo.

Asumiendo que:

- Es posible obtener una descripción formal del modelo macroeconómico en forma de ODEs.
- Es posible utilizar la herramienta *Stella* para generar un modelo SD equivalente al existente en la herramienta Minsky.
- Es posible utilizar nuestro nuevo traductor para producir un modelo DEVS equivalente.
- Al obtener un modelo DEVS es natural hacer la interconexión del modelo macroeconómico con un modelo basado en Cell-DEVS (como es el caso de modelo de intercambio de opinión).

Entonces, nuestra hipótesis es que:

- Es posible obtener un modelo híbrido combinando SD con autómatas celulares, en el cual la dinámica de un submodelo puede influir sobre la dinámica del otro.
- El esfuerzo de obtener dicha combinación se concentra mayoritariamente en el diseño conceptual y modelado de las nuevas interacciones, ocultando toda problemática subyacente referida a sincronización entre un modelo continuo

a parámetros concentrados (como SD) y un modelo de agentes a eventos discretos y espacialmente explícito (como Cell-DEVS).

Con estas ideas en mente propondremos un caso de prueba para nuestro nuevo modelo híbrido y analizaremos los resultados del mismo.

8.1. Modelo de intercambio de opinión

Este modelo fue tomado del trabajo [20], el cual fue desarrollado como trabajo práctico para el curso *Simulación de Eventos Discretos*, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, UBA.

Su objetivo es el estudio del resultado final (asintótico) de una elección entre dos partidos, sometida a un libre intercambio de opiniones, y se inspira en un trabajo sobre dinámicas de opinión [21].

Cada **votante** V que participa de la elección tiene una preferencia y cierto **grado de convicción C** representada como un número real en el intervalo $\left[\text{-3,3}\right]$, la cual clasificaremos de la siguiente manera:

- $\left[\text{-3,-2.8}\right)$: Votante Influyente Negativo V_I^-
- $\left[\text{-2.8}, -1\right)$: Votante Pasivo Negativo V_P^-
- $\left[\text{-1,1}\right]$: Votante Indeciso V_{ind}
- $\left(\text{1,2.8}\right]$: Votante Pasivo Positivo V_P^+
- $\left(\text{2.8,3}\right]$: Votante Influyente Positivo V_I^+

Definimos que existe un partido **oficialista** y uno **opositor** y por lo tanto la connotación de **Negativo** y **Positivo** se refiere a si un votante se **opone** o **apoya** al partido gobernante, respectivamente. El grado de convicción C le da la característica de **Pasivo** o **Influyente**.

Los votantes se distribuyen espacialmente en una grilla regular de tamaño N x N e intercambian opinión con sus vecinos pertenecientes a una vecindad de Von Neumann.

La dinámica del intercambio de opinión se da de a *pares de votantes vecinos*, a los que denominamos genéricamente votante A (**vA**) y votante B (**vB**).

8.2. Modelo Goodwin-Keen

Steve Keen es un economista australiano, profesor de economía y director de la School of Economics, History and Politics en Kingston University, Londres.

Su trabajo académico se ha concentrado en formular una crítica a la interpretación neoclásica de la macroeconomía por carecer de fundamento empírico. Keen se especializa en inestabilidad financiera. En 1995 publicó "Finance and Economic Breakdown". En 2008 escribió "en diciembre de 2005, casi dos años antes de que golpeara la crisis, me di cuenta que se acercaba una seria crisis financiera. Estaba preocupado de la probable severidad y de la falta de conocimiento sobre ella entre los actores políticos, por lo que tomé el riesgo (para un

académico) de volverme muy público sobre mis puntos de vista. Empecé a comentar sobre política económica en los medios, comencé el DebtWatch Report, registré una página web con el oportuno nombre de www.debtdeflation.com, y establecí el blog Steve Keen's Oz Debtwatch. [22].

Según Keen, la incapacidad de los economistas neoclásicos para reconocer los problemas que ha ocasionado la crisis financiera iniciada en 2007 y que desembocó en la Gran Depresión confirmaría que los postulados de la economía neoclásica son falsos.

Para dar sustento a sus argumentos decidió modelar de forma clara y reusable sus modelos económicos no ortodoxos y para ello desarrolló el software *Minsky*. Según Keen, representar estos modelos mediante herramientas clásicas de SystemDynamics es muy complicado visualmente mientras que Minsky permite realizar estas representaciones de forma más entendible (ver 11).

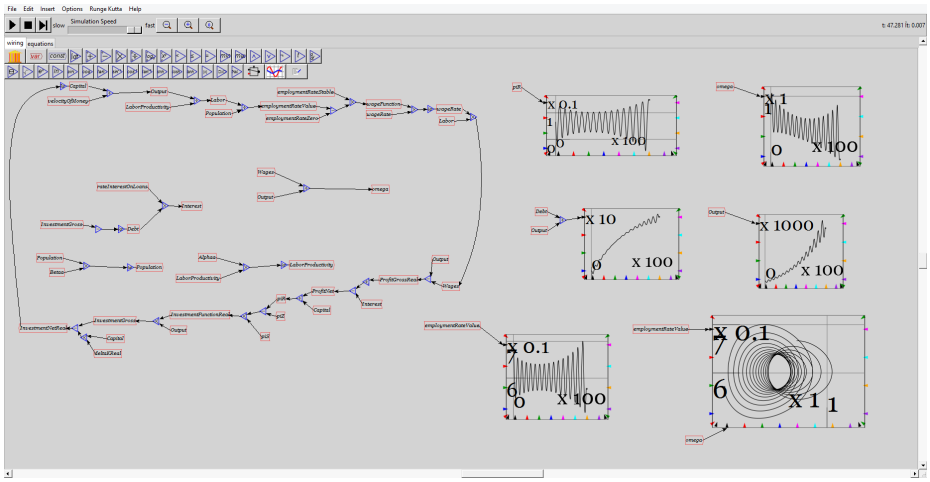

Figura 11. Modelo Goodwin-Keen modelado y ejecutado en el software Minsky.

Su modelo más conocido, en el cual según Keen se basó para poder predecir la crisis financiera del 2008, es una extensión al clásico **Modelo Goodwin**, que es un *modelo cíclico de crecimiento* de la economía [23].

Keen explica la teoría y cómo modelar y simular este modelo en Minsky en videos publicados en la web, como por ejemplo el titulado *Modeling Financial Instability using Minsky*[1]. Allí detalla cómo el modelo está basado en ideas de Marx (1867), quien planteaba el siguiente circuito macroeconómico:

Suba de salarios → Baja de inversión → Baja de crecimiento → Suba de desempleo → Baja en demanda de aumento de salarios → Suba de ratio de ganancias → Suba de la inversión → Suba de crecimiento → Baja del desempleo → Suba de salarios → ...

[1] https://www.youtube.com/watch?v=zpbMd9QW7VM

8.3. Modelo híbrido macroeconomía-opinión

Ya estamos en condiciones de implementar un *modelo multinivel* como se muestra en la 12 que combina ambos modelos: se generan *shocks* de acuerdo a la evolución del **Modelo Goodwin-Keen** y la relación entre sus variables accionando sobre el **Modelo de Opinión** y modificando su dinámica.

Como puede verse en la 12, en la capa superior se ubica el *Modelo Goodwin-Keen*, con su dinámica de evolución interna. Sus variables de output son **EmploymentRate**(t) y **Wages**(t). Éstas son recibidas por el modelo isntermedio *Modelo de Shocks*.

El *Modelo de Shocks* define los **criterios de shocks** en función de los valores que recibe del *Modelo Goodwin-Keen*. El **Modelo Opinión** puede recibir los shocks en cualquier instante de tiempo t (tiempo continuo).

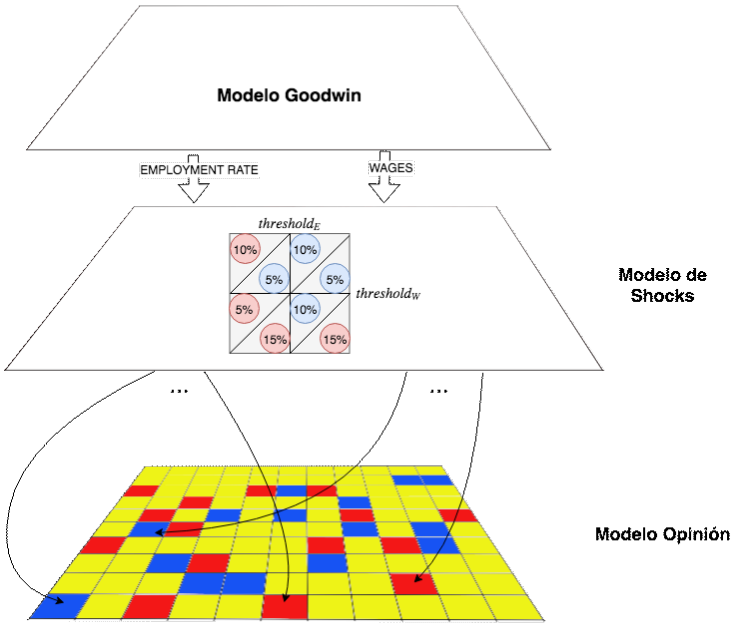

Figura 12. Modelo DEVS multinivel:

Para este nuevo modelo híbrido definimos que el intervalo entre interacciones de a pares de votantes es de 1 mes. En el **Modelo Goodwin-Keen** la unidad

de tiempo es en **años**. La simulación del modelo utilizando los parámetros que Keen propone en su artículo tiene una duración total de 50 años (hasta que el sistema se vuelve inestable, momento de supuesto colapso económico).

Está claro que **Modelo Opinión** fue pensado originalmente para una situación de *ballotage* en un intervalo de tiempo corto, mientras que el **Modelo Goodwin-Keen** intenta modelar un comportamiento macro de la economía a lo largo de períodos mucho más amplios (en el orden de algunas décadas).

Creemos que el modelo combinado resultante podría aceptar interpretaciones realistas bajo ciertas suposiciones. Por ejemplo, considerando que los intercambios de opinión entre agentes en una sociedad se manifiestan más fuertemente en forma mensual, bajo la hipótesis de que el debate se intensifica a partir de la publicación mensual de estadísticas e índices (por ejemplo inflación o tasa de desempleo).

8.4. Criterios de shocks

En 13 y 14 se muestra la evolución de las variables **EmploymentRate**(t) y **Wages**(t), simulado con *Stella* y luego con *CD++* (es decir, luego de la traducción).

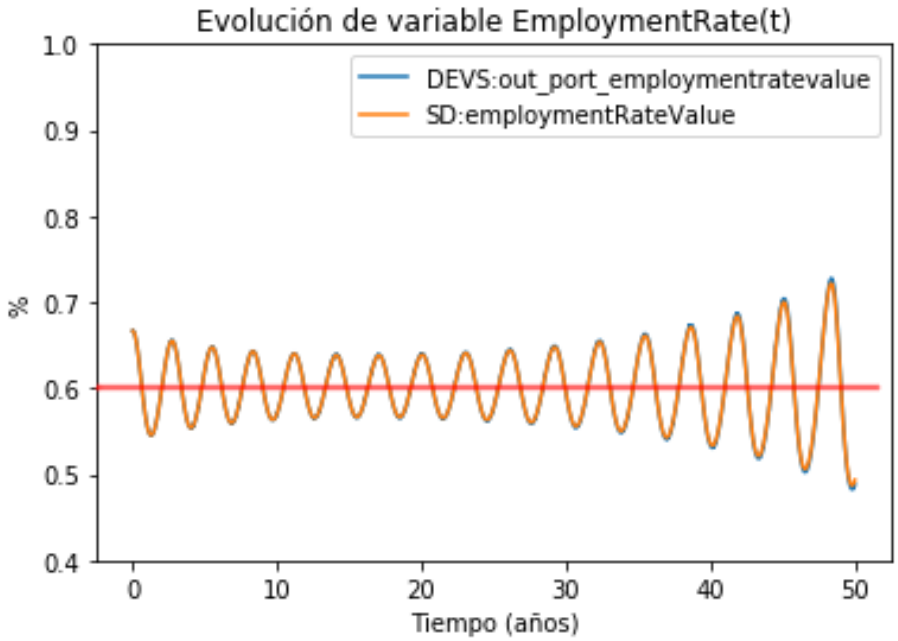

Figura 13. Evolución del empleo a lo largo de 50 años según el modelo Goodwin-Keen. La línea roja marca el umbral utilizado en el ejemplo de próximas secciones.

0.8[h!]

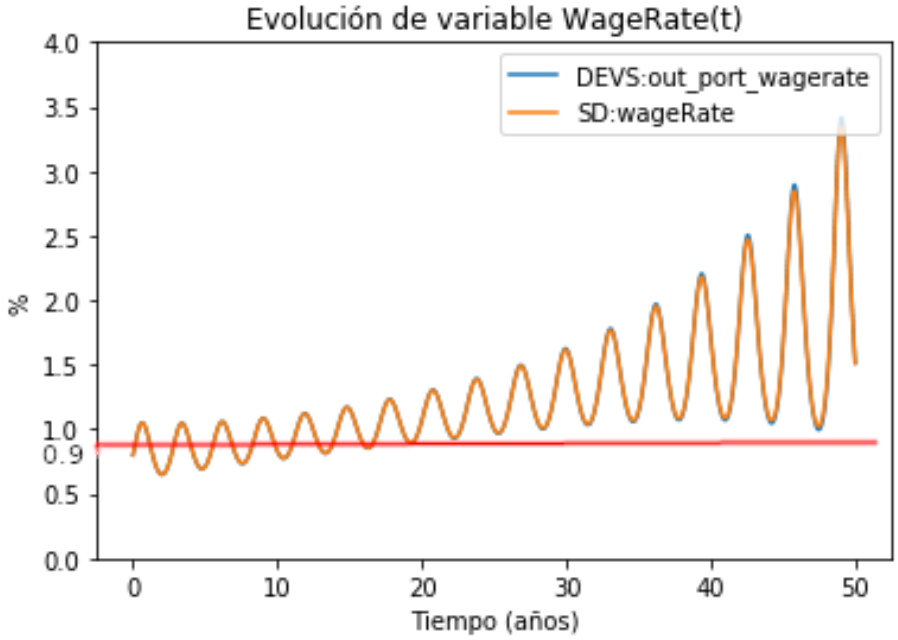

Figura 14. Evolución de los salarios a lo largo de 50 años según el modelo Goodwin-Keen. La línea roja marca el umbral utilizado en el ejemplo de próximas secciones.

El **Shocker** implementado maneja dos *umbrales*, uno para el valor del **EmploymentRate**, al que denominaremos $threshold_E$ y otro para el de **WagesRate**, al que denominaremos $threshold_W$. Utilizamos la palabra *threshold* para nombrar a las variables de los umbrales por consistencia con la denominación en inglés de las variables del modelo de Keen.

Cuando **EmploymentRate** o **WagesRate** atraviesen sus *umbrales* el **Shocker** enviará un *shock de opinión* que tendrá las siguientes propiedades de acuerdo al valor de las variables frente a sus umbrales:

- **EmploymentRate** $> umbral_E$ y **WagesRate** $> umbral_W$: a un 10 % de las celdas se les envía un *shock positivo*, y a otro 5 % se les envía otro *shock positivo*
- **EmploymentRate** $> umbral_E$ y **WagesRate** $< umbral_W$: a un 10 % de las celdas se les envía un *shock positivo*, y a otro 15 % se les envía un *shock negativo*
- **EmploymentRate** $< umbral_E$ y **WagesRate** $> umbral_W$: a un 10 % de las celdas se les envía un *shock negativo*, y a otro 5 % se les envía otro *shock positivo*
- **EmploymentRate**$< umbral_E$ y **WagesRate** $< umbral_W$: a un 5 % de las celdas se les envía un *shock negativo*, y a otro 15 % se les envía otro *shock negativo*

Los valores de los porcentajes son arbitrarios y se eligieron a título ilustrativo.

Todas las celdas son afectadas por el shock en simultáneo (es decir, en el mismo instante del tiempo todas las celdas afectadas actualizan su estado de acuerdo a dicho shock).

La elección de las celdas a ser afectadas por el **Shocker** se realiza de manera aleatoria uniforme al momento de enviar cada shock, y una misma celda no puede ser afectada al mismo tiempo por dos tipos diferentes de shock.

Para definir el comportamiento de cada votante ante la recepción de un *shock* utilizamos la variable δ del **Modelo Opinión**, y agregamos una nueva variable *qshock*, que define en cuántos *deltas* modificará el *shock* la opinión del votante:

Shock positivo: opinión del votante se ve modificada en $+qshock \cdot \delta$
Shock negativo: opinión del votante se ve modificada en $-qshock \cdot \delta$

8.5. Experimentación

En las siguientes figuras se muestra la evolución del Modelo de Opinión con y sin shocks, para 10 escenarios distintos.

En todos los escenarios el estado inicial de la grilla contiene:

- 10 % de votantes influyentes negativos V_I^-
- 10 % de votantes influyentes positivos V_I^+
- 5 % de votantes pasivos negativos V_P^-
- 5 % de votantes pasivos positivos V_P^+

- 70 % de votantes indefinidos V_{ind}

Como se muestra en los ejemplos de la 18, la distribución de los votantes sobre la grilla cambia de escenario a escenario, manteniendo constante la cantidad de votantes pertenecientes a cada grupo, pero distribuyéndolos espacialmente de diferente manera cada vez.

En la 18 los valores de la convicción se varía de escenario a escenario: en 16 hay un agente con opinión -3, pero no así en 15.

Por otra parte, en cada escenario se utilizan los siguientes *umbrales* para las variables **employmentRate** y **WageRate**:

- $umbral_E = 0{,}6$
- $umbral_W = 0{,}9$

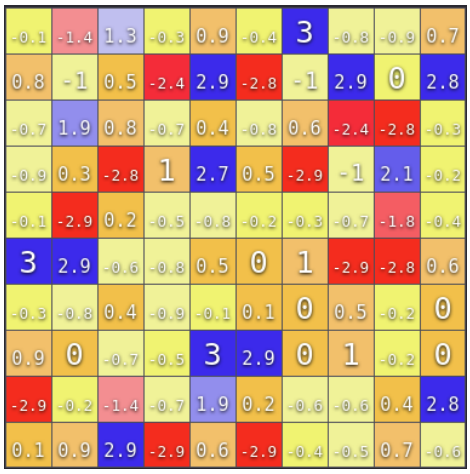

Figura 15. Configuración inicial del Escenario 1

Para la realización de estos gráficos, se subdividió el intervalo de 60 años en 100 ventanas de igual longitud. Tanto para el modelo con shocks como sin shocks, para cada agente, en cada una de dichas ventanas, se identificó cuál era la opinión del agente al comenzar ese intervalo de tiempo. Hecho esto, para cada intervalo se calcula el promedio de la convicción de todos los agentes de la grilla, y se graficó acorde se muestran las figuras.

Para cada escenario, desde un punto de vista global (a partir del cálculo del promedio del nivel de opinión en toda la grilla), en un caso es claramente afectado por los shocks externos y en el otro no.

La comparación entre cada curva de un mismo color en las figuras 19 y 20 y otra figura muestra que el *Modelo Goodwin-Keen* puede cambiar cualitativamente los resultados a largo plazo del modelos de opinión.

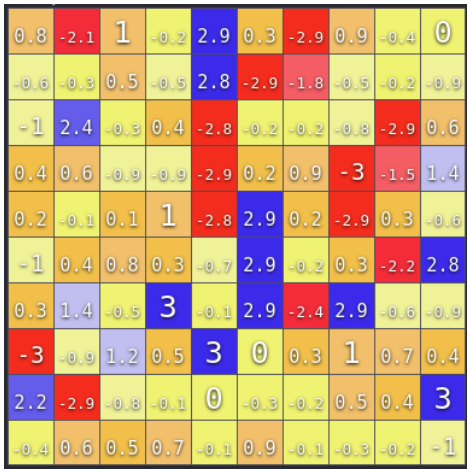

Figura 16. Configuración inicial del Escenario 2

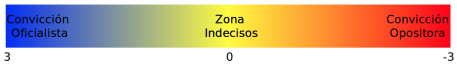

Figura 17. Escala de colores utilizada en 15 y 16.

Figura 18. Visualización de la configuración inicial. Ejemplo para dos de los escenarios utilizados en la experimentación. En ambos casos hay la misma proporción de cada tipo de votante, pero distribuidos espacialmente de diferente manera en la grilla y no necesariamente con exactamente los mismos valores absolutos de convicción.

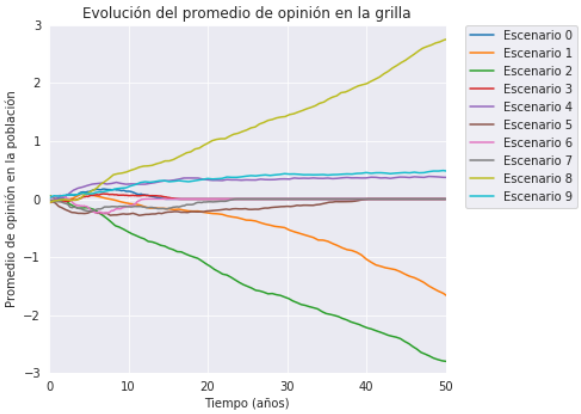

Figura 19. Resultados **sin** shocks.

Por ejemplo, para el Escenario 9, la presencia de shocks llevó a la población de un estado global *indiferente* hacia una preferencia sostenida por el *partido*

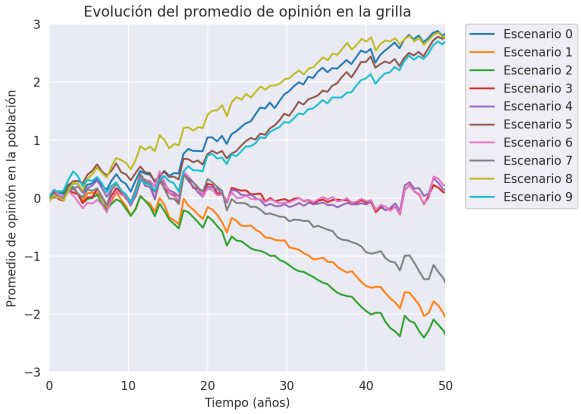

Figura 20. Resultados **con** shocks.

oficialista. De observar el efecto sobre varios escenarios, una conclusión parcial posible es que la distribución espacial de los influyentes tiene efectos determinantes, lo cual es consistente con resultados conocidos en teoría de grafos (influencia de la ubicación de hubs dentro de un grafo).

9. Conclusiones

Implementamos un traductor de modelos SystemDynamics a DEVS basándonos en el formato de especificación XMILE para la descripción de modelos SystemDynamics. Luego de efectuar la traducción del modelo, lo hemos descripto en un nuevo formato propio muDEVSML para realizar su especificación como modelo DEVS.

Los ejercicios realizados con algunos modelos clásicos del mundo de la simulación en SD como Teacup, SIR, Lokta-Volterra y Supply Demand mostraron que las traducciones son correctas en cuanto a que se obtienen los resultados esperados. Esto es, se replican los resultados de manera indistinguible simulando en una herramienta como Stella para SD y como CD++ para DEVS.

Implementamos extensiones de algunos de estos modelos clásicos: Supply Demand Extendido y Lotka-Volterra Extendido para demostrar el potencial del traductor usando funcionalidades avanzadas como la modularización de modelos en varias componentes.

En particular fue relevante confirmar un resultado esperable en términos de sistemas híbridos: al ser SD un formalismo intrínsecamente basado en tiempo discreto, presenta problemas para lidiar con eventos discretos que ocurren en instantes arbitrarios de tiempo. Pudimos comprobar que mientras una herramienta para SD falla en el tratamiento del evento discreto, el modelo traducido a DEVS maneja correctamente el evento gracias a las propiedades asíncronas de

los integradores tipo QSS (basados en DEVS) frente a los integradores clásicos de SD que utilizan un valor fijo de DT.

También mostramos que es posible traducir modelos de simulación basados en ecuaciones diferenciales ordinarias implementados originalmente en otros programas de simulación, como el Modelo Goodwin-Keen en Minsky.

Mostramos que el traductor permite la implementación de modelos DEVS híbridos, a partir de la composición de modelos previamente desarrollados en SD y en DEVS. En particular mostramos que es posible integrar modelos celulares Cell-DEVS con modelos SystemDynamics. De esta forma mostramos que de forma directa y a través de un único formalismo es posible integrar modelos SD y DES mediante el formalismo DEVS, en lugar de tener que recurrir a la comunicación, sincronización e intercambio de mensajes entre distintos programas de simulación.

Referencias

1. J. Forrester, *Industrial Dynamics*. Waltham, MA: Pegasus Communications, 1961.
2. P. P. Merten, R. Löffler, and K. Wiedmann, "Portfolio simulation: A tool to support strategic management," *System Dynamics Review*, vol. 3, pp. 81 – 101, 1987.
3. X. Honggang, A. N. Mashayekhi, and K. Saeed, "Effectiveness of infrastructure service delivery through earmarking: the case of highway construction in china," *System Dynamics Review*, 1998.
4. D. N. Ford and J. Sterman, "Dynamic modeling of product development processes," *System Dynamics Review*, 1997.
5. D. H. Meadows, J. Randers, and D. L. Meadows, *Limits to Growth: The 30-Year Update*. Chelsea Green, 2004.
6. B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. London, UK: Academic Press, 2nd ed., 1976.
7. J. S. Morgan, S. Howick, and V. Belton, "A toolkit of designs for mixing discrete event simulation and system dynamics," *European Journal of Operational Research*, vol. 257, no. 3, pp. 907 – 918, 2017.
8. A. Pugh, *DYNAMO user's manual*. M.I.T. Press, 1963.
9. isee systems, "Stella Professional."
10. Ventana Systems, Inc. of Harvard, Massachusetts, "Vensim."
11. Simulistics Ltd, "Simile."
12. Powersim Software AS, "Powersim Studio 10."
13. F. Bergero and E. Kofman, "Powerdevs: a tool for hybrid system modeling and real-time simulation," *SIMULATION*, vol. 87, no. 1-2, pp. 113–132, 2011.
14. G. A. Wainer, "Cd++: a toolkit to define discrete-event models," *Software, Practice and Experience. Wiley*, vol. 32, pp. 1261–1306, November 2002.
15. H. S. Sarjoughian and B. P. Zeigler, "Devsjava : Basis for a devs-based collaborative m & s environment," 2014.
16. Y. Van Tendeloo, "Activity-aware devs simulation," *Master's Thesis, University of Antwerp, Antwerp, Belgium*, 2014.
17. S. Mittal, J. L. Risco-Martín, and B. P. Zeigler, "Devsml: automating devs execution over soa towards transparent simulators," *SpringSim '07 Proceedings of the 2007 spring simulation multiconference*, vol. 2, pp. 287 – 295, 2007.

18. M. Helal, L. Rabelo, J. Sepúlveda, and A. Jones, "A methodology for integrating and synchronizing the system dynamics and discrete event simulation paradigms," 07 2007.
19. G. A. Wainer and P. J. Mosterman, *Discrete Event Modeling and Simulation: Theory and Applications*. Montreal, Quebec: CRC Press, 1st ed., 2011.
20. G. Dima and E. Sosa, "Dinámica de intercambio de opinión (trabajo práctico para la asignatura simulación de eventos discretos, departamento de computación, facultad de ciencias exactas y naturales, universidad de buenos aires)," 2016.
21. P. Balenzuela, J. P. Pinasco, and V. Semeshenko, "The undecided have the key: interaction-driven opinion dynamics in a three state model," *PloS one*, vol. 10, no. 10, p. e0139572, 2015.
22. D. Bezemer, "No one saw this coming. understanding financial crisis through accounting models," workingpaper, University of Groningen, SOM research school, 2009.
23. S. Keen, "A monetary minsky model of the great moderation and the great recession," *Journal of Economic Behavior & Organization*, vol. 86, pp. 221 – 235, 2011.