Introducción al Modelado Basado en Agentes

Imanol Garnelo Pérez

Octubre de 2025

¿Por qué el Modelado Basado en Agentes?

Esta serie de sesiones ofrece una introducción a la metodología del modelado basado en agentes (ABM) y muestra cómo esta herramienta puede ayudarnos a comprender con mayor profundidad tanto los sistemas naturales como los sociales, además de servir como apoyo en el diseño de soluciones a problemas colectivos. Antes de explicar por qué el ABM resulta relevante, conviene aclarar brevemente en qué consiste. Un agente es una entidad autónoma —puede representar a una persona, un animal, una organización o incluso un recurso— que en el modelo se traduce en un objeto computacional, es decir, una unidad programada en el software con propiedades y reglas de comportamiento definidas. (Wilensky & Rand, 2015, p. 2).

Ventajas de las representaciones computacionales

Estas representaciones computacionales —o objetos dentro del modelo— son dinámicas y ejecutables, lo que permite una interacción directa entre el usuario y la simulación.
Aún más importante, los modelos basados en agentes ofrecen ventajas particulares: resultan intuitivos y relativamente fáciles de comprender. (Wilensky & Rand, 2015, p. 2).

ABM vs. Modelos Matemáticos

Los modelos basados en agentes ofrecen una intuición y comprensión más claras que las representaciones matemáticas del mismo fenómeno.
Esto se debe a que se construyen a partir de individuos con reglas simples de comportamiento, lo que facilita visualizar cómo sus interacciones producen resultados colectivos. En cambio, los modelos matemáticos tradicionales se expresan mediante símbolos y ecuaciones abstractas, lo que puede resultar menos accesible.
En la vida cotidiana solemos pensar y hablar en términos de personas que interactúan entre sí, no en función de tasas de cambio agregadas como las que describen las ecuaciones diferenciales. (Wilensky & Rand, 2015, p. 2).

Modelo Matemático (Modelo SIR simplificado) (Wilensky & Rand, 2015, p. 2)

Qué es:

- Es un **modelo matemático** que describe cómo se propaga una enfermedad en un grupo de personas.
- Representa **grupos** de personas, no individuos.
- Variables:
  - x = número de personas sanas
  - y = número de personas infectadas

Fórmula del modelo:

$$\frac{dy}{dt} = \beta xy - \gamma y$$

- β: rapidez con la que los sanos se contagian al interactuar con infectados.
- γ: rapidez con la que los infectados se recuperan.
- La fórmula indica cómo cambia el número de infectados (y) con el tiempo.

Modelo Basado en Agentes (ABM) (Wilensky & Rand, 2015, p. 2)

Qué es:

- Es un **modelo basado en agentes**, que representa a cada persona individualmente.
- Cada agente puede estar:
  - Sano (x)
  - Infectado (y)
- Los agentes interactúan: si un sano se encuentra con un infectado, puede contagiarse.
- Los infectados pueden recuperarse con el tiempo.
- Más fácil de entender, porque representa interacciones reales y no solo fórmulas.

Diferencia clave entre ambos modelos (Wilensky & Rand, 2015, p. 2)

- **Modelo Matemático:** describe cómo cambian los grupos de personas usando ecuaciones; es **abstracto** porque no representa individuos, solo promedios y tasas de cambio.
- **ABM:** describe cómo cambian individuos a través de interacciones; más intuitivo y fácil de comprender, porque vemos lo que hace cada agente.

Herramientas para estudiar sistemas complejos (Wilensky & Rand, 2015, pp. 6)

Idea principal: El avance del conocimiento impulsa el desarrollo de herramientas más potentes.

- El aumento del poder de cómputo nos permite:
  - **Modelar** sistemas con muchos elementos interconectados
  - **Simular** cómo interactúan dichos elementos
  - **Analizar** los patrones y fenómenos que emergen de esas interacciones
- De esta manera se consolida el campo de los **sistemas complejos**.

Definición de sistemas complejos (Wilensky & Rand, 2015, pp. 6)

Características principales:

- Los sistemas complejos están formados por **muchos elementos** que se afectan entre sí.
- El comportamiento total del sistema **no se puede predecir** solo mirando cada elemento por separado.
- De estas interacciones surgen **patrones o fenómenos nuevos** que no estaban planeados.
- Esta aparición de patrones inesperados se llama **emergencia**, y es lo que define a los sistemas complejos.

Emergencia en sistemas complejos (Wilensky & Rand, 2015, pp. 6-7)

Aspectos clave de la emergencia:

- Los patrones globales aparecen de manera espontánea a partir de interacciones locales.
- No existe un control central; el sistema se **autoorganiza**.
- Las reglas simples a nivel micro generan estructuras ordenadas a nivel macro.
- Las macroestructuras son **dinámicas**: pueden deshacerse y volver a formarse.
- Cada reestructuración es distinta, influida por el **azar y la probabilidad**.

Agentes computacionales e IA

Los agentes computacionales han sido un tópico activo de exploración durante las primeras seis décadas de la inteligencia artificial (IA). En los años setenta y ochenta, la investigación se concentró en aplicaciones como la representación del conocimiento, sistemas expertos, aprendizaje de máquinas e interpretación del lenguaje natural.
Una pequeña comunidad conformó la inteligencia artificial distribuida (IAD), definida como el estudio de entidades autónomas que cooperan y se comunican para desarrollar tareas específicas (Ontiveros & Calvo, 2017, p. 24).

Preambulo sobre NetLogo

Introducción al lenguaje NetLogo:

- Creación de agentes e interacción con su entorno.
- Procesos que permiten la interacción y los cambios en el entorno.
- Ejemplo sencillo de programación en NetLogo y construcción de interfaz gráfica.

(Ontiveros & Calvo, 2017, p. 24).

Un viaje rápido por NetLogo

- Descargar e instalar NetLogo desde su página oficial.
- Interfaz gráfica con pestañas: “Ejecutar”, “Información” y “Código”.
- Biblioteca de modelos con ejemplos prefabricados.
- Uso de la pestaña “Información” siguiendo el protocolo ODD.
- Ventana de código que compila y verifica sintaxis.

Estas secciones permiten entender NetLogo como un lenguaje, un ambiente y un sistema de programación (Ontiveros & Calvo, 2017, pp. 41-48).

El ambiente en NetLogo

En la pestaña **“Ejecutar”**, el ambiente del modelo ya está construido. Este mundo se compone de *patches* (parcelas), que son agentes sin movilidad pero con ubicación, color y propiedades definidas por el usuario.
Por defecto, el mundo mide 33×33 parcelas, es decir, 1 089 *patches*. El espacio es toroidal: no tiene límites verticales ni horizontales, ya que se cierra sobre sí mismo (Ontiveros & Calvo, 2017, p. 54).

Construcción del laboratorio en NetLogo

En la ventana **"Ejecutar"** se pueden añadir botones que activan procedimientos. Por ejemplo, un botón "Inicializar" se vincula al procedimiento `setup`.

- El comando `ask` indica qué agentes ejecutarán las acciones.
- El comando `pcolor` define el color de cada parcela.
- El comando `set` establece la acción o propiedad.
- Comandos clave: `clear-all`, `n-of`, `in-radius`.

Esto permite transformar el mundo artificial para experimentar con diferentes escenarios (Ontiveros & Calvo, 2017, pp. 55-58).

Estructura básica de un modelo en NetLogo

NetLogo se organiza en distintos niveles de agentes y variables:

- **Globals:** variables compartidas por todos los agentes.
- **Patches:** representan el espacio o territorio del modelo.
- **Turtles:** agentes móviles que interactúan con el entorno y entre sí.
- **Breeds:** subtipos de tortugas con funciones o comportamientos diferenciados.

Globals

- Son variables **globales**, visibles y modificables por todos los agentes del modelo.
- Se definen con:

  ```
  globals [ comida-total ]
  ```

- Sirven para almacenar información compartida, como recursos, tiempo o listas especiales de parches.
- Ejemplo conceptual:
  - `comida-total` = cantidad de comida disponible en todo el mundo.
  - Las tortugas pueden decidir moverse o quedarse según el valor de `comida-total`.
  - Cada tortuga que consume comida disminuye el valor de este global, afectando a todas las demás.

Breeds

- Permiten crear distintos **tipos de tortugas** (agentes móviles).
- Se definen con:

```
breed [ agricultores agricultor ]
breed [ comerciantes comerciante ]
```

- Facilitan asignar reglas y comportamientos específicos a cada tipo.

Patches

- Representan el **espacio** o el entorno del modelo.
- Cada patch tiene coordenadas fijas y puede tener variables propias.
- Ejemplo de comando:

```
ask patches [ set pcolor green ]
```

- Se usan para definir zonas, recursos, o condiciones ambientales.

Turtles

- Son los **agentes móviles** que actúan dentro del entorno.
- Se crean con:

```
create-turtles 100 [ set color blue ]
```

- Pueden moverse, cambiar de color, interactuar con patches o con otras tortugas.
- Ejemplo:

```
ask turtles [ fd 1 rt random 30 ]
```

Simulación de gobernanza territorial (ABSOLUG)

Referencia: von Essen & Lambin (2023) — *Agent-Based Simulation of Land Use Governance (ABSOLUG)*

Objetivo: Mostrar cómo interactúan gobierno, productores y ONGs en zonas tropicales, y cómo diferentes políticas afectan la deforestación y el estado del bosque.

Elementos principales del modelo:

- Cuatro tipos de agentes: grandes y pequeños productores, ONG y gobierno.
- Parches de suelo: bosque, cultivos y áreas protegidas.
- Las decisiones de los agentes dependen de beneficios, costos y sanciones.
- El modelo se ajusta con datos reales de fronteras agrícolas tropicales.

Resultados e importancia del modelo ABSOLUG

Resultados clave:

- Sin intervención del gobierno: casi toda la cubierta forestal se pierde.
- Con políticas activas: la deforestación se reduce significativamente.

Por qué es importante: Permite **probar virtualmente políticas públicas** y comparar los resultados con **datos reales**, fortaleciendo la relación entre simulación y evidencia empírica.

Fuente: von Essen & Lambin (2023), *Journal of Artificial Societies and Social Simulation*, 26(1):5.

Modelos basados en agentes y evidencia empírica

Integración entre simulación y datos reales:

- Los **modelos basados en agentes (ABM)** no sustituyen la realidad, sino que la **recrean y exploran** bajo distintos escenarios hipotéticos.
- Permiten probar políticas o dinámicas sociales donde los **experimentos reales serían imposibles** o éticamente problemáticos.
- Los modelos se **calibran y validan** con datos empíricos: censos, imágenes satelitales, encuestas o registros administrativos.
- Esta combinación fortalece la **validez externa** del modelo y mejora su utilidad para la toma de decisiones.
- En proyectos como ABSOLUG, los datos observados sirven para ajustar parámetros (por ejemplo, tasas de deforestación o costos de transporte).

Basado en von Essen & Lambin (2023).

Ejemplo: Primera simulacion

Modelo simple de 100 agentes (*turtles*) que se distribuyen aleatoriamente y simulan dinámicas de migración.

```
to setup
  clear-all
  create-turtles 100 [
    setxy random-xcor random-ycor
    set color blue
  ]
  reset-ticks
end
```

Visualización esperada: Una cuadrícula donde 100 puntos azules representan agentes migrantes en distintas coordenadas. El comportamiento cambia al modificar parámetros como densidad o velocidad.

Modelo de Migración Versión 1

- Mostrar de manera sencilla cómo los agentes (personas) **se mueven hacia zonas más atractivas**.
- Representar visualmente un proceso de **migración territorial** por desigualdad en oportunidades.
- Introducir los conceptos básicos de programación en NetLogo: `setup`, `go`, `ask`, `patches` y `turtles`.

Código base en NetLogo

```
globals [
  atractivos  ;; parches atractivos
]

to setup
  clear-all
  ask patches [ set pcolor gray ]          ;; parches neutros
  set atractivos n-of 40 patches           ;; 40 atractivos
  ask atractivos [ set pcolor red ]        ;; zonas atractivas
  create-turtles 80 [
    setxy random-xcor random-ycor
    set color blue
    set shape "person"
  ]
  reset-ticks
end
```

Comportamiento de los agentes

```
to go
  ask turtles [
    ;; Si hay una zona atractiva cerca, moverse hacia ella
    if any? atractivos in-radius 5 [
      face one-of atractivos in-radius 5
    ]
    ;; Si no hay, moverse al azar
    rt random 30 - random 30
    fd 1
  ]
  tick
end
```

Explicación del comportamiento

- Los **parches rojos** representan zonas atractivas (mejor empleo, servicios, etc.).
- Las **tortugas azules** representan personas que se desplazan buscando esas oportunidades.
- Si una tortuga detecta una zona atractiva cercana, se mueve hacia ella.
- En caso contrario, sigue un movimiento aleatorio.

Resultados esperados

- Con el paso del tiempo, las tortugas tienden a **agruparse alrededor de las zonas rojas**.
- Se visualiza un proceso de **migración dirigida por el atractivo territorial**.
- El modelo es ideal para discutir:
  - Desigualdad territorial
  - Concentración poblacional
  - Atracción y movilidad social

Modelo de Migración Avanzado

Modelo: Migración por Desigualdad Territorial

- Representar un **flujo migratorio parcial y diverso** hacia zonas más atractivas.
- Incorporar **decisiones individuales**: no todas las personas migran.
- Simular un proceso más realista con **destinos variados** y trayectorias diferenciadas.

Estructura general del modelo

```
globals [
  atractivos     ;; parches atractivos (destinos)
]

turtles-own [
  quiere-migrar?   ;; decisión individual
  destino           ;; destino elegido
]
```

Conceptos básicos

- Cada tortuga representa a una persona.
- Cada parche rojo representa una **zona atractiva**.
- El modelo combina factores estructurales (zonas) e individuales (decisiones).

Inicialización: setup

```
to setup
  clear-all

  ;; Colorear regiones
  ask patches [
    if pycor < 0 [ set pcolor blue ]      ;; zona de origen
    if pycor >= 0 [ set pcolor gray ]     ;; zona neutra
  ]

  ;; Zonas atractivas (rojas)
  set atractivos n-of 40 patches with [pycor > 4]
  ask atractivos [ set pcolor red ]

  ;; Crear personas en el origen
  create-turtles 80 [
    setxy random-xcor random -10
    set color white
    set shape "person"
    set size 1.5
    set quiere-migrar? (random-float 1 < 0.6)
```

Comportamiento: go

```
to go
  ask turtles [
    if not quiere-migrar? [ stop ]          ;; no migra
    ifelse [pcolor] of patch-here = red [
      set color green                       ;; llegó al destino
      set quiere-migrar? false
    ][
      face destino                          ;; avanzar hacia su de
      fd 0.7
      rt random 10 - random 10              ;; pequeña variación
    ]
  ]
  tick
end
```

Lógica y dinámica social

- La población inicial se encuentra en la **zona azul** (origen).
- Solo una parte (60 %) decide migrar, simulando **factores sociales, económicos o personales**.
- Cada migrante elige un destino diferente dentro de las **zonas atractivas** (rojas).
- Las trayectorias son diversas, con pequeñas variaciones aleatorias.

Interpretación visual del modelo

- Zona azul: área de origen (baja oportunidad).
- Zona gris: región de tránsito.
- Zona roja: destinos con mayor atractivo.
- Tortugas blancas: migrantes en tránsito.
- Tortugas verdes: personas establecidas.

El resultado es un patrón espacial de asentamiento diferenciado.

Ejemplo 1: Turtles Circling Simple (Wilensky & Rand, 2015, pp. 8)

Sistema de tortugas en un círculo:

- Cada tortuga sigue una regla simple.
- Inicialización: tortugas distribuidas en un círculo, orientadas en sentido horario.
- Comportamiento: avanzar y girar en cada tick del reloj.
- Pregunta: ¿Qué patrón global emergerá?
- Resultado: el círculo puede mantenerse regular o deformarse según los parámetros ajustados.

Control mediante sliders (deslizadores)

Sliders en NetLogo:

- Controles interactivos que permiten al usuario **modificar valores dinámicamente** durante la simulación.
- En este modelo se usan sliders para:
  - `initial-radius` = radio inicial del círculo de tortugas.
  - `fd-step` = distancia que cada tortuga avanza en cada tick.
- Los sliders no tienen unidades fijas; se pueden ajustar libremente para explorar distintos patrones.
- Ajustar los sliders permite observar cómo pequeñas variaciones afectan el patrón global y la **emergencia**.

Relación para un círculo perfecto

Idea principal: Para que las tortugas describan un círculo perfecto, el avance (`fd-step`) debe coincidir con el arco del círculo definido por (`initial-radius`).

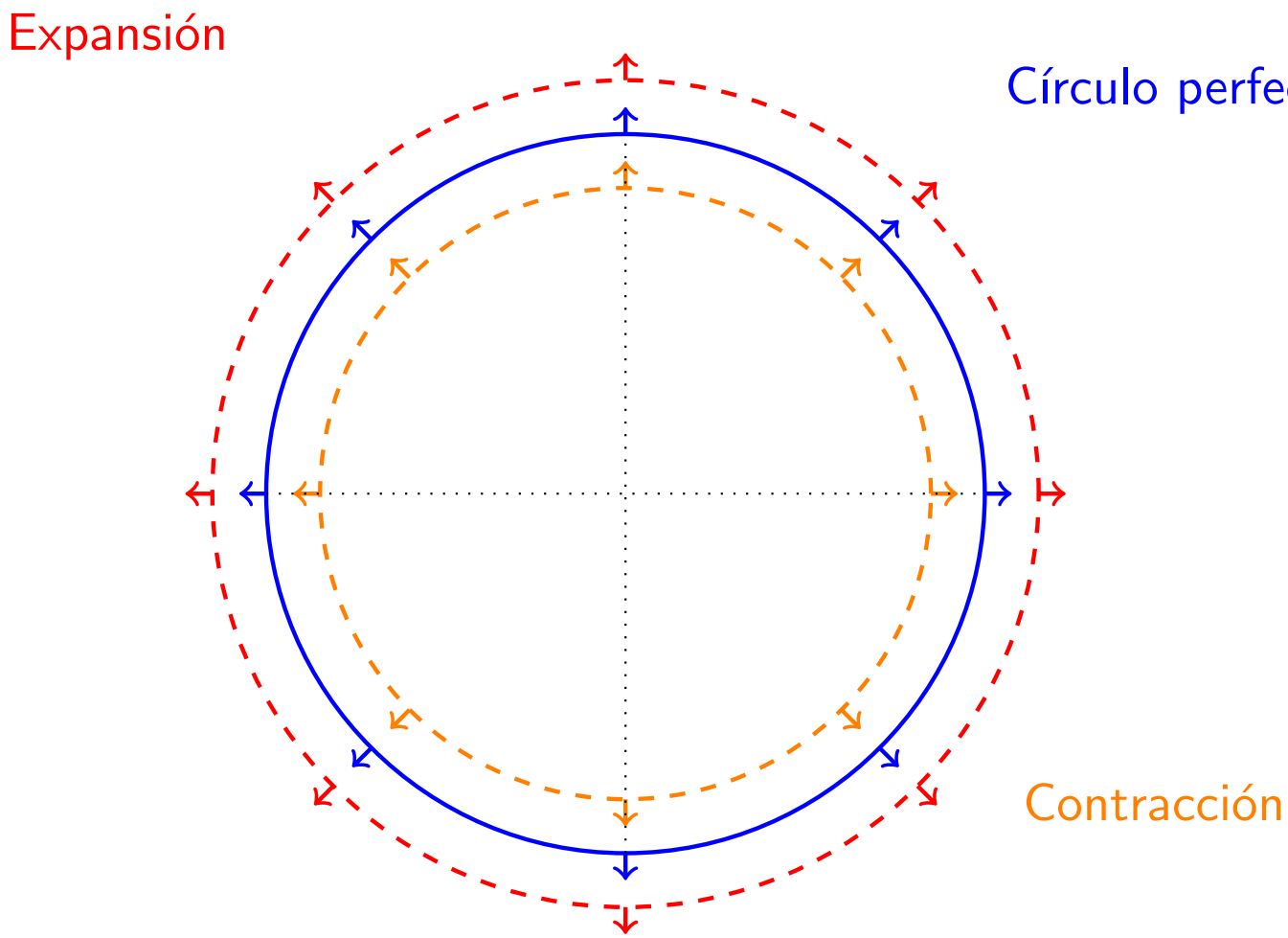
$$\text{Ángulo de giro (turn-step)} \approx \frac{\texttt{fd-step}}{\texttt{initial-radius}} \cdot \frac{180}{\pi}$$

Interpretación del evento emergente

Deformación del círculo:

- Cada tortuga sigue inicialmente un arco de círculo definido por el radio inicial.
- Si la distancia y la orientación no coinciden exactamente con el arco, el círculo se deforma.
- Esto provoca que el círculo parezca expandirse o contraerse según cómo se acumulen los movimientos de cada tortuga.
- Este efecto no está programado explícitamente, sino que surge de la interacción entre los pasos y la orientación: es un **evento emergente**.

Visualización del círculo con flechas



Flechas avanzando sobre segmentos curvos del círculo

Referencias I


Ontiveros, A. A., & Calvo, M. P. (2017). *Introducción al modelado basado en agentes: Una aproximación desde NetLogo*. El Colegio de San Luis. https://colsan.repositorioinstitucional.mx/jspui/handle/1013/792


Wilensky, U., & Rand, W. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. The MIT Press.