UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Carrera de Ingeniero Informático

SIMULACIÓN HÍBRIDA DE LA DIFUSIÓN DE LA INNOVACIÓN CON DINÁMICA DE SISTEMAS Y SISTEMAS MULTIAGENTES

JOSU TAMAYO ZAMORANO

Dirigido por: Dr. Alfonso Urquía

Curso: 2014-15 (convocatoria de defensa)

SIMULACION HÍBRIDA DE LA DIFUSIÓN DE LA INNOVACIÓN CON DINÁMICA DE SISTEMAS Y SISTEMAS MULTIAGENTES

Proyecto de Fin de Carrera de modalidad *oferta específica*

JOSU TAMAYO ZAMORANO

Dirigido por: Dr. Alfonso Urquía (firma)

Tribunal calificador:

Presidente: D./Dª. .....
(firma)

Secretario: D./Dª. .....
(firma)

Vocal: D./Dª. .....
(firma)

Fecha de lectura y defensa: .....

Calificación: .....

RESUMEN

El propósito de este proyecto, es ilustrar cómo implementar un modelo de la difusión de la innovación, un clásico de la literatura de Dinámica de Sistemas (DS), integrándolo con un sistema multiagente (MA) y sistemas de información geográfica (GIS – Geographic Information System) utilizando el entorno de simulación híbrido NetLogo. Para facilitar la comprensión del desarrollo realizado, se ha elaborado un tutorial del entorno de programación NetLogo. También se ha estudiado el estado del arte de las herramientas de simulación híbridas y se ha realizado una comparativa de los entornos NetLogo, Repast y AnyLogic.

La difusión de la innovación (DOI - Diffusion Of Innovation) es una teoría que intenta explicar por qué y cómo se difunden las nuevas tecnologías a través de las diferentes culturas. Pocas teorías de las ciencias sociales tienen una historia de estudio conceptual y empírico como la difusión de la innovación. La robustez y longevidad de esta teoría está basada en la cantidad de campos y disciplinas en las que se ha estudiado.

La elección de este modelo está motivada por que la teoría de la difusión de la innovación se puede descomponer en dos niveles claramente diferenciados: por un lado, un nivel del lado del productor, donde se analizan las políticas y estrategias de la empresa, y un nivel individual, el del consumidor, que, en función de sus estados internos y las interacciones con el entorno, decide adoptar la innovación tecnológica. Este nivel, además puede incorporar una dimensión espacial gracias a la aparición de entornos de simulación con soporte GIS.

Consideramos que aplicar una metodología multinivel a sistemas con múltiples niveles es el enfoque más adecuado. Al simular un modelo multinivel haciendo uso de dos metodologías diferentes se aumenta la complejidad del modelo y se exige al modelador un esfuerzo extra, al tener que trabajar con diferentes paradigmas y herramientas. Por suerte, empiezan a aparecer entornos de simulación, como NetLogo, Repast, AnyLogic que intentan facilitar el desarrollo multinivel agrupando varios paradigmas de modelado en la misma herramienta.

La irrupción de sistemas de información geográfica, por su parte, permite realizar simulaciones con información geoespacial asociada, simulaciones que necesitan de un entorno de simulación flexible y que facilite la transición de los modelos existentes a estas nuevas metodologías.

Haciendo uso de estas herramientas y gracias a los sistemas multiagentes, hemos extendido el modelo DOI añadiendo una dimensión espacial.

LISTADO DE PALABRAS CLAVE

Modelado multinivel

Dinámica de Sistemas

Sistemas basados en agentes

Sistemas de información geográfica

Modelado de sistemas multiagentes

Difusión de la innovación

NetLogo

ABSTRACT

This work aims to explain the process of making a multimethod model of the classical Diffusion of innovation System Dynamics model, using System Dynamics and Multiagent systems methodologies. The resulting multimethod model will integrate (GIS – Geographic Information System) *shapes* using NetLogo modeling environment. In order to understand the development processes, we have made a NetLogo modeling environment tutorial. We also studies the state of art of hybrid simulation frameworks, by making a comparison of AnyLogic, NetLogo and Repast programing environments.

The diffusion of innovation (DOI) is a theory that seeks to explain how and why new ideas spread through cultures. The election of this model is motivated because the diffusion of innovation can be separated in two differentiated levels, a macro level for producer side where political decisions are adopted and a second micro level the market which is the aggregation of many individuals, this level can incorporate a geographic dimension using GIS tools.

Using a multimethod metodology, System Dynamics and multiagent systems, we have extend the DOI model to incorporte a geographic dimension.

KEYWORDS

Multilevel modeling

System Dynamics

Multiagent systems

Geographic information systems

Multiagent system modeling

Diffusion of innovation

NetLogo

Índice

Lista de Figuras

Lista de Tablas

Índice de Código

1 Introducción, Objetivos y Estructura

"Hasta ahora hemos mantenido estable la situación..., pero, por primera vez en la historia del Plan Seldon, es posible que los actos inesperados de un solo individuo lo destruyan".

Segunda fundación, Isaac Asimov

1.1 Introducción

En el mundo de la simulación existen diferentes enfoques de modelado como, por ejemplo, la Dinámica de Sistemas (DS), de Jay Forrester, los sistemas dinámicos de eventos discretos (DEDS – Discrete Events Dynamic Systems) y los sistemas multiagentes (SMA).

Como señalan Meadows, Dennis L (1985), "Cada metodología de modelado, está basada en un modelo de cómo debería ser el modelado". Cada metodología, se basa en unas asunciones subyacentes únicas, que son constantemente usadas, pero raramente analizadas por la comunidad modeladora. Es por ello, que a estas asunciones se las llama paradigmas.

La dinámica de sistemas es una metodología de tiempo continuo, cuya formulación se expresa en ecuaciones diferenciales. Se utiliza desde una perspectiva "top-down", de arriba abajo, y es útil para modelar sistemas desde un nivel macro (colectivo - sociedad). Su concepto subyacente principal, el bucle de realimentación, ha permitido aplicar esta metodología a multitud de sistemas causales: desde sistemas sociales, economía, cadenas de suministros, producción industrial.

Los sistemas multiagentes son una metodología de simulación proveniente de la Inteligencia Artificial de tiempo discreto, que se utiliza desde una perspectiva "bottom-up", de abajo arriba, y que resulta útil para modelar sistemas desde un nivel micro (individuo – agente). Los sistemas multiagentes han venido siendo utilizados en numerosos campos, como la ecología,la biología,la economía,el trafico, el sector militar, etc. Los sistemas multiagentes que se centran en la simulación de fenómenos sociales, son conocidos como SSBA: Simulación social basada en agentes (ABSS Agent based social simulation). Los sistemas multiagentes, tienen varias desventajas si los comparamos con los modelos tradicionales de Dinámica de Sistemas: mayor cantidad de datos a ser calibrados, mayor coste computacional, resultados más difíciles de tratar, etc.

Pero no todo son desventajas: un punto clave que distingue y potencia los sistemas multiagentes frente a la Dinámica de Sistemas es el concepto de *emergencia,* procesos o propiedades de un sistema no reducibles a las propiedades o procesos de sus partes constituyentes. Un ejemplo sorprendente es el de las dinámicas de segregación Schelling (1971), donde Schelling explicaba cómo se pueden obtener patrones claros de segregación no deducibles a partir de la psicología individual,

pero sí a partir de las iteraciones locales de individuos con tendencias segregacionistas débiles.

A lo largo del tiempo, estas dos técnicas se han ignorado la una a la otra, plasmándose en una carencia de modelos híbridos. No deja de ser chocante, que, teniendo una multitud de ámbitos de estudio comunes y solapados, apenas podamos encontrar ejemplos. Se han desarrollado modelos combinando DS y herramientas GIS en ámbitos como el crecimiento de áreas residenciales, el desarrollo urbano, etc. En cuanto a los SMA y GIS, existen modelos forestales, logísticos.

Como Lorenz y Jost señalaban en la 24ª Conferencia de Dinámica de Sistemas, generalmente los desarrolladores de modelos suelen pasar por alto otros métodos de modelado, pues no son capaces de diferenciar y aplicar métodos alternativos diferentes a los que conocen. Siempre que en un modelo encontremos estas perspectivas un enfoque de alto nivel y otro de bajo nivel, y el objetivo del modelado lo permita, un enfoque multimétodo puede resultar más conveniente que aplicar un único método exclusivamente.

Lorenz & Jost (2006a) también señalaban allá por el año 2006 que existía una carencia de herramientas de simulación multimétodo, lo que justificaba la casi nula existencia de modelos multimétodo o híbridos. Parece que, después del llamamiento realizado, en los últimos cinco años han surgido varias herramientas (Repast 2012, NetLogo 2012, AnyLogic 2012), que van a facilitar el desarrollo de modelos multimétodo. Estos entornos, aparte de permitir modelar utilizando múltiples paradigmas, permiten importar información geográfica (polígonos, puntos , vectores), proveniente de herramientas GIS, al modelo y representarla de forma visual en él, ampliando aun más las posibilidades de modelado.

Los Sistemas de Información Geográfica (GIS – Geographical Information Systems) son un conjunto de herramientas informáticas, que permiten la organización y almacenamiento de grandes cantidades de datos del mundo real, asociados a sus coordenadas geográficas y que facilitan la toma de decisiones (sociales, económicas, ambientales, etc.).

La difusión de la innovación es una teoría que intenta explicar por qué y cómo se difunden las innovaciones a través de las diferentes culturas. Normalmente, la innovación se presenta en forma de producto o servicio innovador, por lo que la difusión de nuevos productos o servicios es solo

equiparable a esta teoría si el producto es innovador. Los profesionales en gestión estratégica de las empresas, muestran un gran interés por este campo, ya que la comercialización exitosa de innovaciones determina si la empresa puede generar una ventaja competitiva y asegurar el éxito a largo plazo.

Modelos agregados, tales como el modelo de difusión de (Mansfield 1961) o (Bass 1969), proporcionan una generalización empírica que ayuda a tomar decisiones pero hasta cierto límite, ya que estos modelos no se diseñaron para responder a preguntas del tipo ¿qué sucedería si...?. Además, los modelos agregados no consideran la heterogeneidad de los compradores y la complejidad de los procesos sociales que dan forma a la difusión.

La irrupción de nuevos canales de información sociales, tales como Twitter, Facebook, Tuenti o Whatsapp, y la pérdida de peso progresiva de los medios de comunicación tradicionales han provocado que la investigación en materia de difusión de productos innovadores , adopte la simulación multiagente. Una de las principales razones de usar esta metodología abajo arriba, del inglés “bottom-up”, es que permite modelar fenómenos complejos de emergencia, como la difusión de la innovación.

El proceso de difusión, resulta muy adecuado para ser modelado desde un enfoque híbrido, ya que, por un lado, contiene un nivel empresarial, donde se adoptan las decisiones estratégicas para producir y comercializar la innovación con una perspectiva “top-down”, y, por otro lado, existe otro nivel, una sociedad compuesta de individuos entre los cuales se generan una serie de relaciones personales que pueden favorecer la difusión de la innovación.

1.2 Objetivos

Respondiendo al llamamiento realizado por Lorenz y Jost (2006b), este proyecto pretende realizar el estudio de las herramientas y teorías subyacentes, que permiten construir modelos multimétodo y contribuir en la difusión de este tipo de modelado.

Para ello, hemos reimplementado un modelo teórico de Dinámica de Sistemas, el modelo de

difusión de la innovación publicado por Ahmadian (2008), utilizando la herramienta NetLogo. Mediante una serie de refinamientos que mostramos en la Figura 1.1, hemos obtenido un modelo multimétodo aplicable a un ámbito geográfico, gracias a la introducción de restricciones geográficas obtenidas mediante la herramienta de información geográfica QGIS.

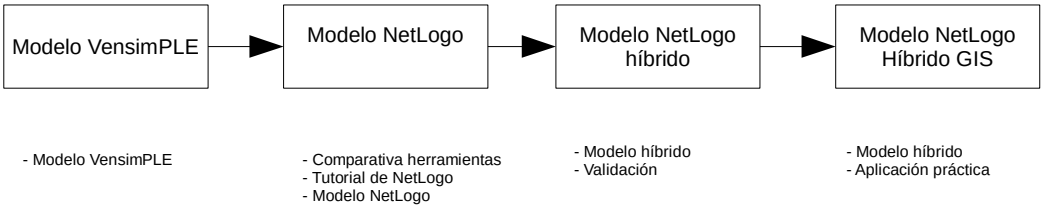


Figura 1.1: Fases del proyecto y entregables

El proyecto, según se definió en el anteproyecto, ha generado los siguientes entregables:

1. Implementación del modelo de difusión de la innovación en VensimPLE.
2. Estudio comparativo de las herramientas de simulación híbrida.
3. Tutorial de NetLogo.
4. Modelo NetLogo de la difusión de la Innovación.
5. Modelo híbrido en NetLogo.
6. Modelo híbrido aplicable a diferentes ubicaciones geográficas haciendo uso de la extensión GIS de NetLogo.
7. Caso practico, aplicación del modelo a la difusión de energía verde caso Som Energia.
8. Memoria.

Creemos que los entregables, así como sus correspondientes análisis realizados en la memoria, servirán para entender cómo se pueden combinar las metodologías de Dinámica de Sistemas y sistemas multiagentes, para realizar modelos híbridos en los nuevos entornos de modelado multiparadigma, como NetLogo.

A continuación detallamos los objetivos que hemos perseguido al generar los entregables.

Se ha elaborado una comparativa de los entornos de simulación multimétodo AnyLogic, Repast y NetLogo, con el objetivo de facilitar a los modeladores la decisión de qué herramienta adoptar a la hora de realizar un modelado multimétodo.

Se ha realizado un tutorial sobre la herramienta NetLogo, con el objetivo de tener una visión sobre el entorno de desarrollo y el lenguaje de programación NetLogo, con el que se han elaborado los modelos de este proyecto.

Se ha elaborado un tutorial con el objetivo de entender la integración del entorno NetLogo mediante su extensión GIS, con fuentes de datos provenientes de herramientas GIS. Con el objetivo de facilitar la comprensión de los conceptos clave de este último punto, se ha realizado un pequeño tutorial sobre el entorno de desarrollo QGIS Valmeira, donde se expone *grosso modo* qué son los sistemas GIS, cuáles son los conceptos fundamentales y la forma de trabajo con este tipo de herramientas, junto a ejemplos reales utilizados en la obtención de *rasters* para el modelo del proyecto.

Partiendo de la memoria de la tesis de Ahmadian (2008), hemos construido su modelo utilizando la herramienta de simulación de Dinámica de Sistemas VensimPLE, con el objetivo de realizar una conversión completa a NetLogo de este modelo, utilizando el modelador de Dinámica de Sistemas de NetLogo.

Partiendo de este modelo, se discute la idoneidad y motivación de reimplementar el modelo utilizando un enfoque multimétodo. Proponemos una división del modelo en dos partes: la parte empresarial, que se ha modelado mediante Dinámica de Sistemas, y la parte del mercado, que se ha simulado mediante un sistema multiagente.

Una vez dispuesto el modelo híbrido, se ha añadido una serie de restricciones espaciales importando al modelo *rasters* y *shapes*, obtenidas mediante la herramienta SIG QGIS y la extensión GIS de NetLogo.

1.3 Estructura de la memoria

La memoria está estructurada en nueve capítulos:

- El primer capítulo es esta introducción, donde se explican los objetivos que persigue el proyecto y muestra cómo se ha estructurado la memoria.

- En el segundo capítulo presentaremos las metodologías y herramientas que se han utilizado para desarrollar el proyecto. Primeramente se estudia la Dinámica de Sistemas, después los sistemas multiagentes, para, una vez entendidas las diferencias entre ambas metodologías, realizar, en tercer lugar, un análisis de las diferentes formas de combinar ambas metodologías y una comparativa de las herramientas de simulación multimétodo AnyLogic, NetLogo, Repast; por último, se analizan trabajos similares.

- En el tercer capítulo, hemos confeccionado un tutorial sobre el lenguaje y entorno de programación NetLogo con el que se ha desarrollado el proyecto. También se dedicará un apartado a explicar la extensión de NetLogo (GIS[1]), que permite importar capas vectoriales procedentes de herramientas GIS a NetLogo. Para facilitar la comprensión del punto anterior y con la intención de introducir los principales conceptos manejados en los sistemas de información geográfica, hemos confeccionado un breve tutorial sobre la herramienta QGIS, con ejemplos reales utilizados en la confección de la información geográfica utilizada en este proyecto.

- En el cuarto capítulo, presentaremos y analizaremos el modelo de difusión de la innovación, que se ha simulado de una manera híbrida usando las metodologías de DS y SMA. En el último apartado de este capítulo también realizaremos una validación del modelo híbrido.

- En el quinto capítulo se explicará cómo y por qué se ha subdividido el modelo de difusión de la innovación en dos niveles: un nivel macro, donde se definen las políticas de producción y comercialización, que se rige por un modelo DS, y un nivel micro, compuesto por los usuarios que se modelarán utilizando una metodología multiagente. El último apartado de este capítulo se centrará en validar el modelo híbrido frente al modelo de Dinámica de Sistemas.

1 El nombre oficial de la extensión en el repositorio es GIS

- El sexto capítulo, se centrará en explicar cómo se puede incorporar información geográfica proveniente de herramientas GIS. Para este proyecto se ha utilizado la herramienta QGIS y se ha confeccionado información a medida para el modelo de difusión de la innovación. También se explicará detalladamente el uso de la extensión GIS de NetLogo que nos ha permitido importar ficheros ESRI (.shp) generados mediante QGIS al modelo.
  Se han realizado una serie de hipótesis y modificaciones en el modelo híbrido para analizar su comportamiento acotado a un ámbito geográfico determinado.

- En el séptimo capítulo discutimos la planificación y el coste del proyecto.

- En el octavo capítulo, ponemos a prueba el modelo desarrollado, aplicándolo a la difusión de un servicio de comercialización de energía verde que se esta realizando por la cooperativa Som Energia.

- El noveno y último capítulo está dedicado a las conclusiones y trabajos futuros.

1.4 Estructura del CDROM

El CDROM se ha estructurado mediante carpetas, de la siguiente manera:

1. **Carpeta Instaladores**. Se han incluido en el CD las herramientas con las que se ha desarrollado la práctica al permitirlo su licencia de distribución.

   a) **Carpeta NetLogo**.
   Instaladores de NetLogo[1] para diferentes plataformas.

   NetLogo5.1.0Installer.exe (Windows), NetLogo-5.1.0.tar.gz (GNU), NetLogo205.1.dmg (Mac) .

1 Copyright 1999-2014 by Uri Wilensky.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

b) **Carpeta QGIS.**

Instaladores de QGIS[1] para diferentes plataformas.

QGIS-OSGeo4W-2.2.0-1-Setup-x86_64.exe (Windows 64), QGIS-2.6.1-2.dmg (Mac)

Librería GDAL [2], requerido para instalar QGIS en Mac OSX.

GDAL_Complete-1.11.dmg (Mac)

2. Carpeta Memoria.

a) *Memoria.pdf*. Memoria en formato PDF (Portable Document Dormat).
b) *Memoria.odt*. Memoria en formato ODT (Open Document Type).

3. Carpeta Modelos.

a) **Carpeta Vensim**.

SD.mdl modelo de difusión de la innovación desarrollado utilizando VensimPLE.

b) **Carpeta NetLogo**

SD.nlogo.

Implementación del modelo *SD.mdl* de Vensim en NetLogo, correspondiente al Capítulo 4.

1 Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)
You are free to:
Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the materialfor any purpose, even commercially.
The licensor cannot revoke these freedoms as long as you follow the license terms.
2 http://opensource.org/licenses/mit-license.php
The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

SDMA.nlogo

Implementación del modelo de Dinámica de Sistemas con sistemas multiagentes, correspondiente al Capítulo 5.

SDMAGIS.nlogo

Implementación del modelo de Dinámica de Sistemas con sistemas multiagentes añadiendo información proveniente de herramientas GIS, correspondiente al Capítulo 6.

SOM.nlogo

Aplicación del modelo SDMAGIS a la cooperativa Som Energia.

2 Metodologías y herramientas

2.1 Introducción

En este capítulo vamos a presentar las metodologías de modelado y las herramientas que hemos utilizado en el desarrollo del proyecto: Dinámica de Sistemas, sistemas multiagentes, QGIS. No vamos a realizar una introducción a NetLogo ya que el Capítulo 3 entero se dedica a su estudio.

El último apartado de este capítulo se destina a describir trabajos similares realizados utilizando una combinación de las metodologías y herramientas aquí descritas.

2.2 Dinámica de Sistemas

2.2.1 Introducción

Un sistema es un conjunto de partes o elementos organizadas y relacionadas que interactúan entre sí para lograr un objetivo. Los sistemas reciben (entrada) datos, energía o materia del ambiente y proveen (salida) información, energía o materia[1].

Todo sistema posee las siguientes características:

1. Un conjunto de partes o elementos, estos pueden ser concretos (planetas, tráfico) o abstractos (finanzas, otros sistemas).
2. Las partes del sistema están organizadas.
3. Las partes se influencian entre sí.
4. Tienen unos límites, fronteras.

Un sistema dinámico es un sistema que evoluciona con el tiempo.

La Dinámica de Sistemas es una metodología que permite modelar el comportamiento de sistemas complejos a lo largo del tiempo. Según su creador Jay Forrester[2] "La Dinámica de Sistemas versa sobre cómo cambian las cosas a lo largo del tiempo, lo que incluye casi todo lo que la gente considera importante. Usa la simulación por computador para obtener el conocimiento que tenemos

1 Diccionario Informático http://www.alegsa.com.ar/Dic/sistema.php
2 Jay Forrester también lideró en el MIT el grupo que más tarde crearía la memoria RAM

acerca de los detalles del mundo que nos rodea y nos muestra por qué nuestros sistemas físicos y sociales se comportan como lo hacen" [1].

La Dinámica de Sistemas resulta útil cuando:

1. Nos interesamos por la evolución de un sistema complejo.
2. Cuando no tenemos por objetivo determinar situaciones óptimas, si no comprender el funcionamiento del sistema y poder responder a cuestiones del tipo ¿que pasaría sí..?.
3. Probar hipótesis, probar escenarios.
4. Evaluar políticas de gestión de crisis en escenarios simulados.

La Dinámica Industrial (nombre que adoptaba la DS en los 50) fue creada por Jay Forrester en el MIT (Massachusetts Institute of Technology) para resolver un problema en concreto: el que presentaba una empresa de productos electrónicos que, teniendo poco volumen de negocio, registraba considerables oscilaciones en la línea de producción (Forrester 1962). Forrester llegó a la conclusión de que las oscilaciones eran debidas a la combinación de estructuras de realimentación y retrasos en la transmisión de la información a lo largo de la empresa.

Al llevar a cabo este análisis, sentó las bases del método que hoy conocemos como Dinámica de Sistemas, que cristalizó en su obra "Industrial Dynamics" (Forrester 1962). Más tarde, Forrester extendió este método a otros ámbitos de aplicación gracias a sus obras "Urban Dynamics" (Forrester 1969), que estudia los problemas de la sociedad urbana, y "World Dynamics" (Forrester 1971), que se centraba en estudiar los problemas asociados al crecimiento demográfico y la contaminación. En 1970 El Club de Roma presentó el modelo del mundo, coordinado por el matrimonio Meadows (Meadows D.H., Meadows D.L., Randers J. 1972), utilizando la DS; gracias a este trabajo y su difusión, se popularizo la DS.

2.2.2 Estructuras de Realimentación

Un bucle de realimentación es una cadena cerrada de acciones elementales entre los elementos de un sistema, como por ejemplo:

- Proceso de llenado de un vaso de agua.

1 De la lista de correo electrónico de la System Dynamics Society

- Sistema automático de regulación de la temperatura mediante termostato.

Para poder examinar las relaciones de influencia que se dan entre los elementos de un sistema y poder entender la estructura, estas relaciones se suelen representar mediante diagramas causales:

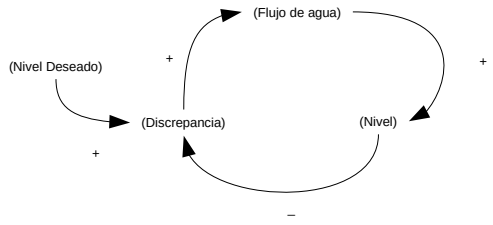


Figura 2.1: Diagrama básico del proceso de llenado de un vaso

Reglas para realizar un Diagrama Causal:

- Las relaciones entre elementos se representan mediante una flecha.
- Sobre la fecha se indica el tipo de influencia mediante el símbolo + si la influencia es positiva y – si la influencia es negativa.

En la Figura 2.1 se muestra el bucle de realimentación del proceso de llenado de un vaso. En el diagrama se muestran las influencias que se producen entre los distintos elementos que intervienen en el proceso. Mediante los signos + y – se indica si la influencia es positiva o negativa. En este ejemplo hemos descompuesto los elementos esenciales y las relaciones de influencia entre ellos; también encontramos un bucle de realimentación, ya que se produce una transmisión de información entre los elementos de forma circular.

El bucle de realimentación está presente en una infinidad de sistemas y es el origen de comportamientos complejos que estudia la Dinámica de Sistemas.

Comportamiento anti-intuitivo

En los sistemas simples la relaciones causa efecto son cercanas en el tiempo, pero en los sistemas complejos la causa y el efecto pueden estar muy alejadas en el tiempo.

Ejemplo: Curva de Laffer[1]

1 Aunque el economista Arthur Laffer no reclama haber inventado el concepto de curva de Laffer, se popularizó a raíz de una reunión con funcionarios de la Administración Ford en 1974. Laffer trataba de explicar a Dick Cheney y Donald Rumsfeld las ventajas de una rebaja fiscal y para ilustrar sus ideas, tiró de una servilleta, y dibujó su famosa gráfica.

Subida IVA → Precios más altos → Menos Ventas → Menos Recaudación

Al subir los impuestos indirectos se pretende aumentar la recaudación, pero en determinadas ocasiones este aumento puede producir una disminución en la recaudación, ya que, aunque se recaude más por cada venta, el nivel total de recaudación por ventas puede caer, puesto que, al incrementarse los precios debido al aumento del IVA, la gente puede dejar de comprar.

Insensibilidad

La sensibilidad es la medida en la que se altera el comportamiento de un sistema al cambiar sus parámetros. La existencia de bucles de realimentación reduce la sensibilidad del sistema (Aracil & Gordillo 2005).

Bucle de realimentación negativa

A los bucles de realimentación negativa se les llama de diferentes maneras: estabilizadores, equilibradores, balanceadores, reguladores o autorreguladores, y son la base de cualquier sistema regulador. Los bucles de realimentación negativa tienden a buscar un equilibrio. Una variación en un elemento provoca un efecto que contrarresta la variación. Normalmente a este tipo de comportamiento se le conoce como “Goal Seeking”; en la Figura 2.2 podemos ver como existe un objetivo que alcanzar.

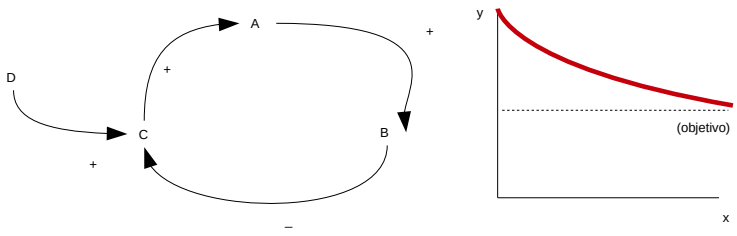


Figura 2.2: Diagrama genérico de realimentación negativago (Aracil & Gordillo 2005)

En la Figura 2.2 también podemos ver el diagrama de un sistema de regulación de la temperatura mediante termostato. Si la temperatura se desvía del valor deseado, se produce una discrepancia y se activa la calefacción. El bucle de realimentación negativa es un comportamiento en el que el sistema corrige las perturbaciones que hacen que no sea el deseado. A este comportamiento también se le conoce como comportamiento regulador.

Bucle de realimentación positiva

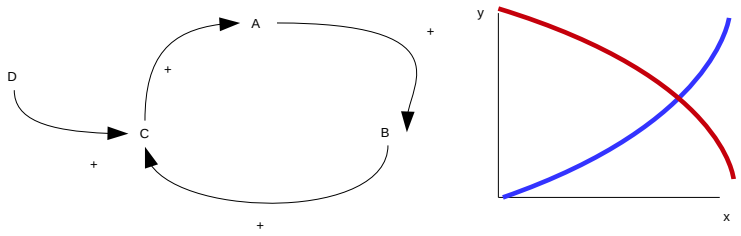


Figura 2.3: Diagrama genérico de realimentación positiva (Aracil & Gordillo 2005)

Los bucles de realimentación positiva hacen que la perturbación de un elemento tienda a reforzarse a lo largo del tiempo, por eso también son conocidos como bucles reforzadores, o más descriptivamente, de efecto bola de nieve. El término "positivo" puede llevar a confusión ya que la perturbación puede ser tanto un incremento como un decremento. En la Figura 2.3 podemos ver los efectos de un bucle de realimentación positiva representados en el diagrama cartesiano de la izquierda: una respuesta explosiva (azul) o depresiva (rojo) exponencial.

Un ejemplo de este bucle puede ser el crecimiento de una población en un entorno sin limitaciones de recursos y espacio.

Retrasos

Un retraso es el tiempo que transcurre entre una causa y sus efectos. Los retrasos pueden tener una influencia notable en el comportamiento de un sistema, por eso, al construir un diagrama de influencias de un sistema, hay que tener en cuenta si la relación es inmediata o tiene un retardo.

El ejemplo más ilustrativo es el del efecto de la bajada del precio sobre las ventas: aunque se baje un precio, no implica que automáticamente las ventas aumenten, hay que esperar a que los compradores lo perciban y eso requiere un tiempo adicional.

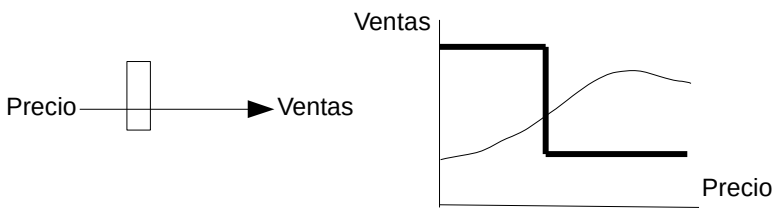


Figura 2.4: Retraso en las ventas provocado por la bajada de precios

Existen dos tipos de retrasos (Aracil & Gordillo 2005):

1. Retraso en la transmisiones materiales.
   Se produce cuando existen elementos en el sistema que almacenan temporalmente el flujo de entrada.
2. Retrasos en la transmisión de la información.
   Se produce cuando hay que conservar y almacenar información del sistema antes de proceder a una toma de decisiones.

2.2.3 Diagramas de Forrester

Los Diagramas de Forrester, llamados Diagramas de niveles y flujos (*Stock and Flow diagrams*), son una representación simbólica del modelo, que está a un paso intermedio entre un Diagrama Causal y el sistema de ecuaciones diferenciales de primer orden que le corresponde (Aracil & Gordillo 2005). Para realizar esta conversión, hay que clasificar los tipos de variables que intervienen en las relaciones, en estos tres tipos básicos:

1. **Variables de nivel**. Son variables que acumulan los resultados de acciones tomadas en el pasado. Tomando el ejemplo del llenado de un vaso, el volumen de agua sería una variable de nivel; apréciese la coincidencia: variable de nivel – nivel del agua. Las variables de nivel suelen representarse mediante un rectángulo (ver Figura 2.5).
2. **Variables de flujo**. Son las variables que miden las variaciones en las variables de nivel del sistema, las cuales se acumulan en el nivel correspondiente. Tomando el ejemplo del llenado de un vaso, la variable de flujo es el caudal de agua. Como veremos más adelante a la hora de convertir un diagrama de Forrester en ecuaciones, son las derivadas de los niveles respecto al tiempo. Se representan mediante un reloj de arena (ver Figura 2.5).
3. **Variables auxiliares**. Las variables auxiliares son variables intermedias que representan un paso en el cálculo de una variable de flujo. Se suelen representar mediante un círculo (ver Figura 2.5).

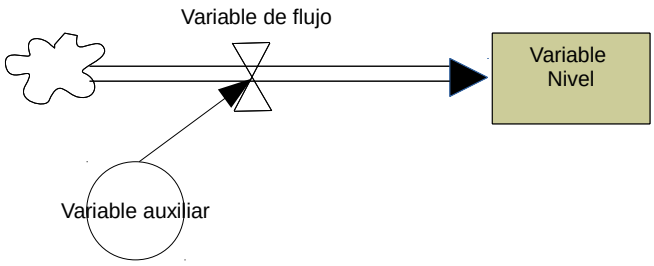


Figura 2.5: Diagrama de Forrester de ejemplo

Los niveles pueden alimentarse a través de un flujo, desde otro nivel o bien desde una fuente exterior al sistema. Esta fuente, que se considera infinita, se representa por una nube. Así mismo, los niveles se vacían a través de un flujo sobre otro nivel o sobre un sumidero exterior al sistema, el sumidero se supone de capacidad infinita y también se representa mediante la nube.

Las variables se relacionan por medio de canales, hay dos tipos:

1. Canales materiales, los cuales se representan mediante un trazo.
2. Canales de información, los cuales se representan mediante un trazo discontinuo.

Esta distinción se realiza a nivel explicativo; en el desarrollo del proyecto no se usan, ya que los paquetes de simulación modernos no realizan esta distinción.

Las **tablas** se usan para representar aquellas relaciones entre variables que son no lineales, este tipo de comportamientos se pueden observar por ejemplo entre la sensación de hambre y la cantidad de alimento consumido; a medida que se come la sensación de hambre disminuye, pero no proporcionalmente a la cantidad de alimento consumido. También se usan para representar multiplicadores, que son las relaciones entre variables que no permanecen constantes a través del tiempo. Ejemplo de ello sería una tasa de interés variable.

Los **retrasos** se representan mediante un rectángulo con 3 rectángulos interiores.

2.2.4 Representación matemática de los Diagramas de Forrester

Como ya mencionamos, detrás de los Diagramas de Forrester y su metáfora hidrotérmica

(niveles, flujos, sumideros) existe una intencionalidad: enmascarar el aparato matemático del cálculo diferencial propio de los sistemas de control, facilitando la comprensión y manejo de los modelos de simulación dinámica.

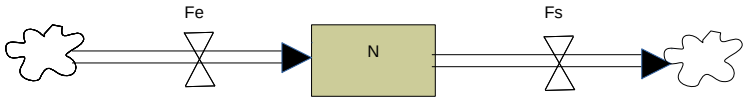


Figura 2.6: Diagrama de Forrester de la ecuación de nivel básica

A cada nivel $N(t)$ se le puede asociar un flujo de entrada $F_e(t)$ y salida, de acuerdo con

$$N(t)=N(t_0)+\int_{t_0}^{t}(F_e(t)-F_s(t))\,dt \tag{2.1}$$

O representado en forma diferencial:

$$\frac{dN}{dt}=F_e(t)-F_s(t) \tag{2.2}$$

A cada flujo $F(t)$ se le asocia una ecuación llamada ecuación de flujo o función de decisión que admite como variables de entrada niveles, auxiliares y constantes, en la forma general

$$F(t)=TN\times M(t)\times N(t) \tag{2.3}$$

Siendo TN el flujo normal (constante), $M(t)$ el multiplicador del flujo normal (auxiliar) y $N(t)$ (nivel). El diagrama de Forrester de la ecuación de flujo es el que se muestra en la Figura 2.7.

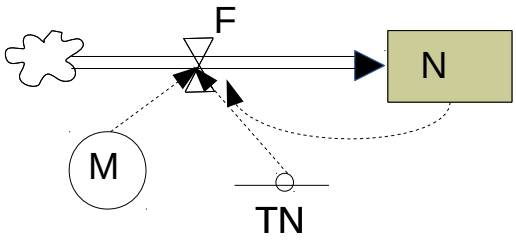


Figura 2.7: Diagrama de Forrester de la Ecuación de Flujo

Los retrasos en la transmisión de material se representan mediante la Figura 2.8.

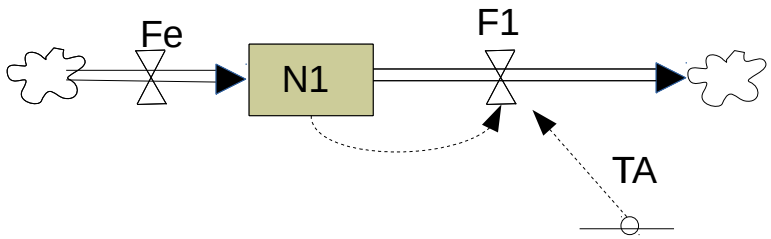


Figura 2.8: Diagrama de Forrester de un retardo en la transmisión material

Se puede traducir en la siguiente ecuación:

$$N_1(t)=N_1(0)+\int_0^t Fe-F_1(t)\,dt \quad \text{donde} \quad F_1(t)=\frac{N_1(t)}{TA} \tag{2.4}$$

Los retrasos de información se representan mediante el siguiente diagrama de Forrester.

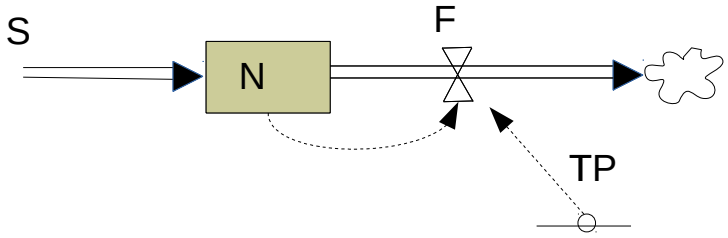


Figura 2.9: Diagrama de Forrester de un retardo en la transmisión de la información

Y su ecuación es

$$N(t)=N(0)+\int_0^t S(t)-F(t)\,dt \quad \text{donde} \quad F(t)=\frac{N(t)}{TP} \tag{2.5}$$

Los diagramas de Forrester representan modelos continuos, para poder resolverlos en un ordenador tenemos que realizar una simulación discreta. Para ello, la ecuación (2.4) se debe representar de manera discreta:

$$N(t+\Delta t)=N(t)+(F_e(t)-F_s(t))\Delta t \tag{2.6}$$

Para poder realizar el cálculo de esta ecuación de nivel mediante un lenguaje iterativo debemos ir acumulando su valor a través de un bucle como el siguiente:

```
dt=1
for (t=inicio; t<=fin; t=t+dt)
{
   N[t] = N[t-1] + (Fe-Fs);
}
```

En este caso, la variable N acumula en el intervalo (inicio, final) la diferencia entre los flujos de entrada y salida.

Para poder simular un modelo de Dinámica de Sistemas, tenemos que desarrollar un programa iterativo que realice todos los cálculos en el intervalo (inicio, final), y que en cada iteración incrementa el tiempo (t). La forma de realizar los cálculos tiene un orden, ya que un nivel es función exclusiva del propio nivel y de los flujos asociados, pero el cálculo de las variables de flujo y auxiliares es más elaborado pues depende de los niveles, de las variables auxiliares y los flujos previos. También hay que tener en cuenta que los niveles se miden en los puntos de muestreo (t o t_{-1}) mientras que los flujos se calculan en los intervalos existentes entre dos puntos de muestreo (Δt o Δt_{-1}).

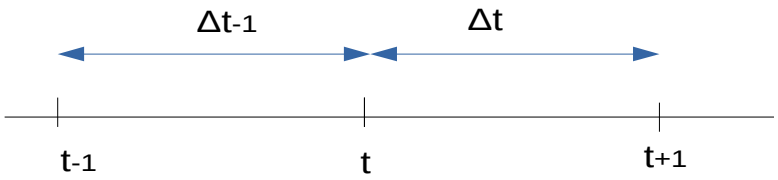


Figura 2.10: Visión del tiempo en el motor de simulación de DS

```
Inicializar parámetros;
Inicializar  Δt;
t = inicio;
Inicializar niveles en t; /* N[t] = N_inicial; */
while (t <= final) {
      Calcular variables auxiliares en t;
           /* aux[t] = f(N[t],aux[t -1],flujo[ Δt -1])); */
      Calcular flujos en  Δt;
         /* flujo[ Δt] = f(N[t],aux[t],flujo[ Δt -1])); */
      Calcular niveles en t +1;
                 /* N[t +1] = N[t] +  Δt*f(flujo[ Δt]); */
      Desplazar: [t] → [t -1]; [t +1] → [t]; [ Δt] → [ Δt -1];
      t = t + Δt;
}
```

Código 2.1: Algoritmo DS (Morlán Santa Catalina 2010)

Teniendo en cuenta todas estas consideraciones, podemos presentar el algoritmo del motor de simulación de un modelo DS:

2.2.5 Fases del Modelado

Según Aracil (1995), el proceso de modelado consiste en el conjunto de operaciones mediante el cual, tras el oportuno estudio y análisis, se construye el modelo. El proceso de modelado tiene las siguientes fases:

1. Definición del problema.

   Se trata de entender cuál es el problema y cuál es el propósito de modelar un sistema. Todo modelo es una representación de un sistema, pero para que un modelo sea útil debe intentar solucionar un problema, no ser un mero reflejo del sistema entero. Como dice Sterman (2000), "el arte del modelado es saber qué es lo que hay que cortar y el propósito del modelo sirve como una cuchilla lógica".

2. Conceptualización del sistema.

   Se trata de descomponer el sistema en sus componentes variables, relaciones, constantes, y describir el comportamiento con un lenguaje preciso. El resultado de esta fase es el diagrama de influencias.

3. Formalización.

   Se trata de, partiendo del diagrama de influencias, concretar el diagrama de Forrester y obtener una representación matemática del modelo. En esta fase debe procederse a asignar valores a los parámetros que intervienen en el modelo. La fase de formalización concluye cuando se dispone de un modelo del sistema en forma de ecuaciones programadas en un ordenador.

4. Comportamiento.

   Consiste en simular el modelo en un ordenador y ver los resultados que genera comparados con el comportamiento actual del sistema. En esta fase se prueban los modelos, no solo con datos actuales, sino que se comparan con datos históricos y condiciones extremas.

5. Definición de políticas y Validación.

   La definición de políticas es mucho más que cambiar parámetros. Consiste en crear

una serie completa de nuevas estrategias, estructuras y reglas.

2.2.6 Herramientas

Cuando Jay Forrester escribió su libro I*ndustrial Dynamics,* utilizó el lenguaje de simulación DYNAMO (DYNAmic MOdels), lenguaje desarrollado bajo su supervisión en el MIT. Se utilizó desde los años 60 hasta los 80 pasando por varias adaptaciones y versiones. Aunque hoy en día no se utiliza, lo mencionamos por su importancia histórica.

Según la System Dynamics Society, los tres paquetes de software más utilizados son (System Dynamics Society s. f.)

- Ithink y STELLA[1] (http://www.iseesystems.com).
- Powersim Studio (http://www.powersim.com)
- Vensim (http://vensim.com/)

Los tres paquetes tienen en común estar basados en los conceptos estándares de Dinámica de Sistemas: flujos de entrada y niveles planteados por Forrester. El desarrollo de un modelo mediante estas herramienta consiste en:

1. Representar el diagrama de Forrester del modelo que se va a simular en el editor gráfico del entorno.

2. Introducir las ecuaciones de las diferentes variables intervinientes.

1Ithink y STELLA son versiones con diferentes funcionalidades y licenciamientos de la misma plataforma de desarrollo.

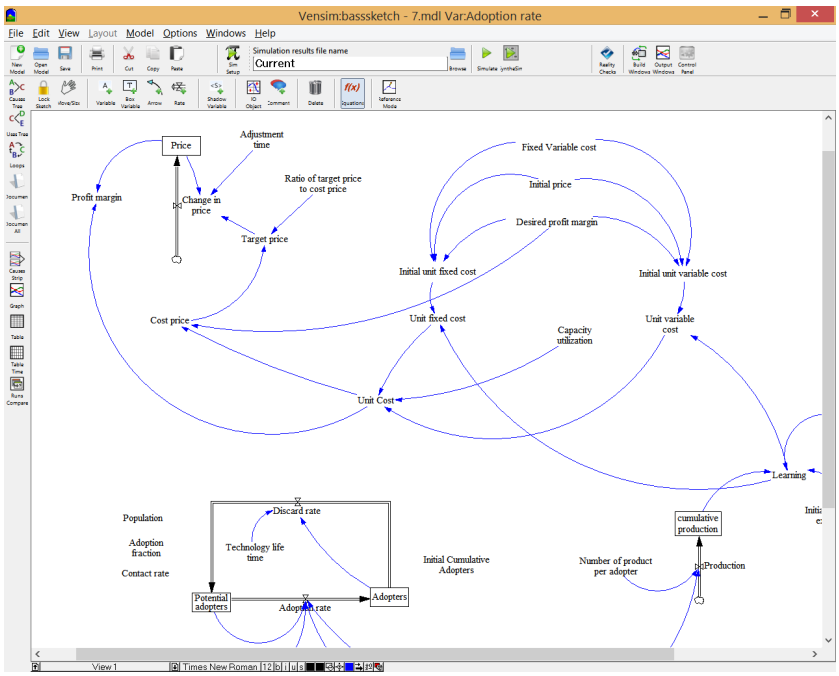


Figura 2.11: Editor de diagramas VensimPLE

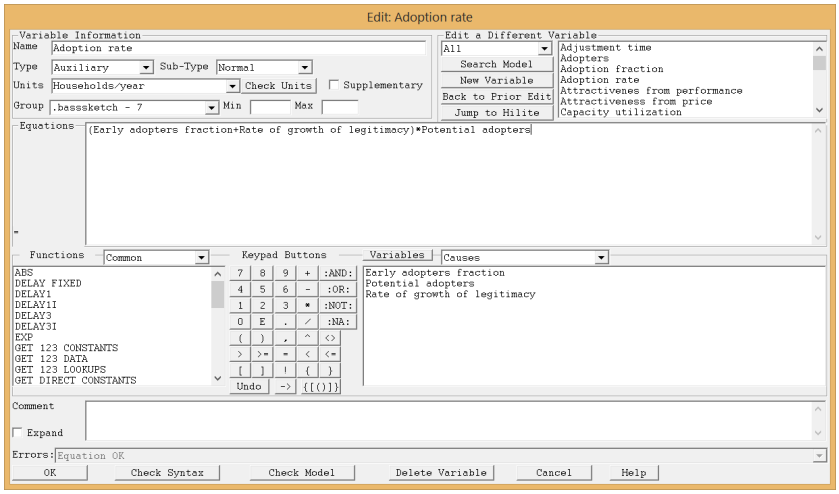


Figura 2.12: Editor de ecuaciones de VensimPLE

Una vez introducidas las ecuaciones, los entornos realizan una validación del modelo y ejecutan la simulación. Los resultados de la simulación, se pueden visualizar mediante tablas de datos o mediante gráficas en los propios entornos.

Las principales funcionalidades de estas herramientas son:

- Editor de diagramas de flujo y de nivel.
- Editor de ecuaciones.
- Validador de unidades.
- Editor de diagramas causales.
- Identificador de bucles.
- Gráficos.
- Importación/exportación de/a datos externos.
- Comparador de simulaciones.

Como indicábamos en el Capítulo 1, en este proyecto hemos utilizado la herramienta VensimPLE; ya partimos de un modelo de la difusión de la innovación que fue elaborado usando esta herramienta. Al no disponer del modelo original en formato .mdl[1], hemos tenido que reconstruir todo el modelo a partir de la Tesis de Máster en Ciencias de (Ahmadian 2008).

1 La extensión .mdl es la que se utiliza para nombrar los modelos de VensimPLE

2.3 Sistemas Multiagentes

2.3.1 Introducción

Existen muchas definiciones de lo que es un sistema multiagente, pero vamos a plantear una, inspirada en las definiciónes dadas por (Benenson & Torrens 2004) y (Wooldridge et al. 1995), que se adecua al propósito de este trabajo: un sistema multiagente es un sistema compuesto por un entorno donde múltiples agentes interactúan. Estas interacciones están influenciadas por el comportamiento del resto de agentes y el entorno.

Los sistemas multiagentes se utilizan en:

- Sistemas industriales de control de procesos.
- "Workflows" y procesos de negocio.
- Agentes de Interfaz.
- Comercio electrónico, (compra, venta, subastas..).
- Recuperación y clasificación de información (spiders, buscadores, agregadores).
- **Simulación Social Basada en Agentes (ABSS Agent-based social simulation).**

Este proyecto se enmarca dentro de la **Simulación Social Basada en Agentes**, que se centra en simular el comportamiento social a través de modelos de ordenador basados en sistemas multiagentes. En este tipo de simulaciones las personas o grupos de personas son representadas por agentes que forman la sociedad.

En su libro "Growing Artificial Societies" (Epstein & Axtel 1996), una de las principales referencias en la literatura de la simulación basada en agentes, presentaban su modelo Sugarscape**,** como un banco de pruebas multiuso para probar teorías sociales.

El modelo "Life and Death on Sugarscape" se considera un paradigma dentro de los modelos de simulación basado en agentes y nos va a servir para introducir esta disciplina.

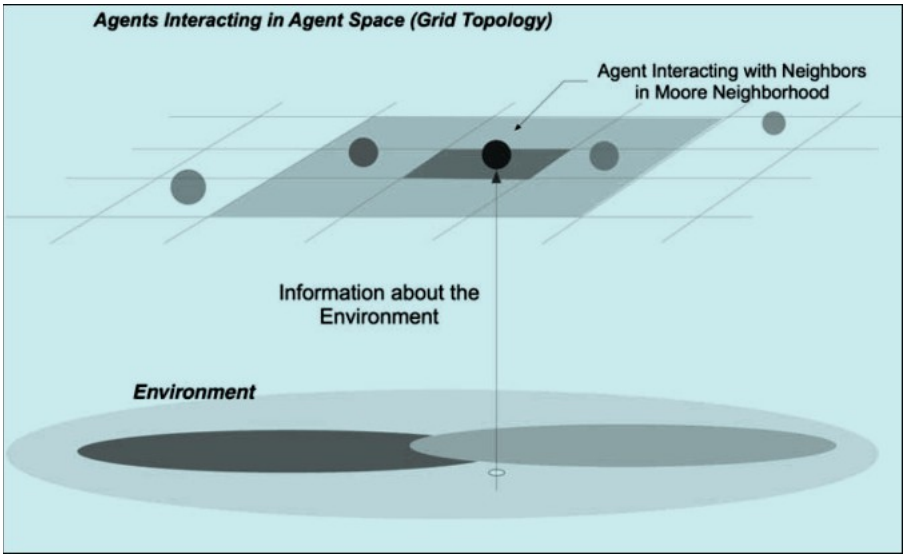


Figura 2.13: Sugarscape Epstein&Axtell (1996)

El modelo original está basado en una matriz de 51x51, donde cada celda puede tener una cantidad de azúcar diferente. En cada paso, los agentes miran a su alrededor para buscar la celda más cercana con azúcar; una vez encontrada, se desplazan y la metabolizan. Las cañas de azúcar vuelven a crecer al cabo del tiempo.

A lo largo del libro, se van introduciendo nuevas reglas a los agentes, permitiendo el comercio, la lucha, la reproducción, etc. Este modelo permite comprender cómo surgen las estructuras sociales, las instituciones , las costumbres y las conductas a partir de las interacciones de los agentes.

"Our rules create the clifs we drive off"

"Computational systems such a Sugarscape can offer 'headlights', if you will, by permiting us to project, however crudely, the evolutionary consequences of certain rules".

Epstein&Axtell

Su libro representa una buena guía para la comprensión y el desarrollo de propuestas de ecosistemas artificiales.

Para poder desarrollar el modelo de este proyecto correctamente, a lo largo de este apartado

vamos a estudiar las características de los agentes de manera individual, su organización y su comportamiento cuando se agrupan formando un SMA, e iremos señalando los principios e ideas que hemos adoptado para realizar nuestro modelo.

2.3.2 Definición de Agente Inteligente

Es importante aclarar que no existe una definición comúnmente aceptada de qué es un Agente Inteligente. Wooldridge planteó una definición basándose en las propiedades que caracterizan un agente de software, y que es la candidata a convertirse en la más aceptada. Según él, "Un agente es un sistema informático situado en un entorno concreto y dotado de comportamiento autónomo orientado a alcanzar unos objetivos predefinidos" (Wooldridge et al. 1995)

- **Reactivo**. Todo agente mantiene una interacción constante con su entorno y es capaz de reaccionar frente a cambios sucedidos en éste en el tiempo suficiente para ser útil.
- **Proactivo**. A su vez, un agente es capaz de tomar la iniciativa; no reacciona únicamente frente a cambios en el entorno, sino que identifica oportunidades y actúa en pro de sus objetivos.
- **Autonomía**. Los agentes operan sin la intervención directa de los seres humanos, tienen un control interno de sus acciones. (Covrigaru, A. 1991) presentan una categorización más detallada de la autonomía, para que un agente sea considerado autónomo debe de:
- **Sociabilidad**. Es la habilidad que tiene un agente de interactuar con otros agentes o con humanos. Para que esta interacción sea satisfactoria, dicho agente tiene que ser capaz de cooperar, coordinar y negociar. Una descripción más formal sería: "Un agente es un sistema informático situado en un entorno concreto y dotado de comportamiento autónomo orientado a alcanzar unos objetivos predefinidos", (Wooldridge et al. 1995).
    1. Disponer de un comportamiento guiado por metas y tiene que ser capaz de seleccionar la meta que va a procesar en cada instante.
    2. Su existencia se extiende un poco más que el tiempo de consecución de sus metas.
    3. Es suficientemente robusto como para adaptarse a cambios en el ambiente.
    4. Puede interaccionar con su entorno en la modalidad de procesamiento de la

información.

5. Es capaz de exhibir una variedad de respuestas, incluyendo movimientos de adaptación fluidos; y su atención a los estímulos es selectiva.
6. Ninguna de sus funciones, acciones o decisiones está totalmente gobernada porun agente externo.
7. Una vez en ejecución, el agente no necesita ser programado nuevamente por un agente externo.

Aparte de estas características básicas, vamos a indicar otras características que no son ni necesarias, ni suficientes para un agente, pero que permiten categorizar los tipos de agentes (Rodríguez Anaya 2012);

- **Movilidad**. Es la habilidad para desplazarse por una red electrónica. Algunos agentes tienen la posibilidad de ser ejecutados en diferentes ubicaciones físicas.
- **Veracidad**. Se trata de la suposición de que los agentes no comunican información falsa.
- **Benevolencia**. Es la suposición de que el agente no tiene objetivos contradictorios y siempre realiza la tarea asignada.
- **Inteligencia**
  - **Racionalidad.** Similar a la característica en los humanos, el agente tiene unos conocimientos de su entorno, unos objetivos y unas reglas que determinan cómo alcanzar los objetivos a partir del conocimiento que maneja.
  - **Coherencia**. La base de datos de conocimiento del agente es coherente.
  - **Adaptabilidad**. El agente adapta su base de conocimientos y su comportamiento a partir de los cambios que se producen en el entorno y sus comportamientos anteriores, el agente aprende.
  - **Aprendizaje**

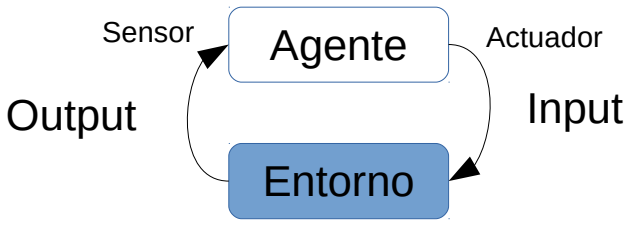


Figura 2.14: El agente y su entorno

En la Figura 2.14 apreciamos como el agente interacciona con su entorno con ánimo de provocar un efecto. Aunque el agente no tiene un control absoluto del entorno, se considera que puede influenciarlo. Esto implica que dos inputs sobre el entorno del mismo tipo en diferentes momentos no van a producir el mismo output.

Esta interacción con el entorno es tan determinante para el agente que su arquitectura queda determinada por el tipo de entorno en el que se mueve. Los entornos poseen las siguientes características según (Russell & Norvig 2009):

- **Accesibilidad**. Un entorno es accesible cuando un agente puede obtener información completa, precisa y actualizada de su estado. Cuanto más accesible sea su entorno, más fácil resulta desarrollar un agente capaz de operar en él.
- **Determinismo**. Un entorno determinista es un entorno donde se sabe con antelación el efecto que se va a provocar al realizar una acción, no existe incertidumbre sobre el resultado de la acción.
- **Episódico / no episódico**. En los entornos episódicos la experiencia de un agente se divide en episodios, las percepciones y acciones que realiza el agente se restringen al episodio actual sin tener en cuenta las acciones / percepciones tomadas en anteriores episodios. Los entornos episódicos son mas simples que los no episódicos, ya que el agente no tiene que basarse en la historia para adoptar decisiones.
- **Dinamismo**. Un entorno es dinámico cuando sobre él pueden actuar varios procesos simultáneamente, y no solo es el agente el que influencia el entorno.
- **Continuidad**. Se trata de entornos donde existe un número finito de acciones o efectos que se pueden realizar por parte de un agente.

2.3.3 Qué no es un Agente

Como indicaba el profesor Rodríguez, muchas veces, otras metodologías y formas de trabajar pueden provocar errores a la hora de entender que es un agente (Rodrigez 2012).

Objetos

Para los programadores acostumbrados a la Programación Orientada a Objetos (POO), puede ser tentador asociar las características de un agente con las de un objeto. Los objetos son entidades

software que encapsulan algún estado y pueden realizar acciones mediante métodos sobre este estado. Los objetos también pueden comunicarse mediante el paso de mensajes. Las instancias de un objeto, además, poseen una identidad, como los agentes, que les permite ser distinguidos entre las otras instancias. Pero la diferencia principal estriba en que los objetos poseen métodos públicos, acciones que realizan para otros objetos, y esto es lo que los hace perder la característica de autonomía.

La distinción entre agentes y objetos se resume en la siguiente frase (Wooldridge et al. 1995):

"Objects do it for free; agents do it because they want to"

La segunda diferencia entre objetos y agentes es referente a la noción de flexibilidad (reactivo, pro-activo, social). El modelo estándar de objeto no permite definir cómo se implementan estas características en un objeto.

La tercera diferencia es que cada agente dispone de un hilo de ejecución propio, mientras que los objetos, aunque también pueden disponer de un hilo propio si así se lo permite el lenguaje de POO en el que se implementan, no tienen porqué tenerlo.

El que los objetos no sean agentes no implica que un agente no pueda ser programado con técnicas de programación orientadas a objetos.

Sistemas Expertos

Los sistemas expertos, como el nombre indica, son sistemas que emulan el comportamiento de un experto en un dominio concreto. Un ejemplo clásico de sistema experto es MYCIN, una herramienta que permitía asistir en el tratamiento de las infecciones. MYCIN interacciona con el médico mediante una serie de preguntas y respuestas y obtiene unas conclusiones; MYCIN funcionaba como un consultor externo. La principal diferencia entre un objeto y MYCIN (u otro sistema experto) es que MYCIN no interacciona con ningún entorno, no obtiene ninguna información a través de sensores. Otra diferencia fundamental es que los sistemas expertos no pueden cooperar con agentes.

Inteligencia Artificial

Los agentes son un producto derivado de la Inteligencia artificial. La IA tiene como meta construir sistemas que entiendan el lenguaje natural, que piensen de modo creativo, que usen el sentido común, una de sus realizaciones son los Agentes. Los agentes son parte de la IA, pero la IA no

se centra exclusivamente en crear agentes.

2.3.4 Taxonomía de los agentes software y Arquitecturas de agentes

Existen varias dimensiones para clasificar el software de agentes existente (Nwana 1996). Una primera dimensión puede ser la movilidad: si los agentes son móviles o estáticos. Podemos clasificar los agentes en función de su comportamiento: si es deliberativo o reactivo. En los agentes deliberativos el comportamiento y conocimiento de los agentes están explícitamente representados mediante un modelo simbólico, además de esto en la arquitectura deliberativa las decisiones de los agentes son hechas mediante razonamiento lógico o pseudológico. Los agentes reactivos, al contrario, no tienen un modelo simbólico interno de su entorno y actúan utilizando un comportamiento de estimulo respuesta para responder al estado presente en el entorno.

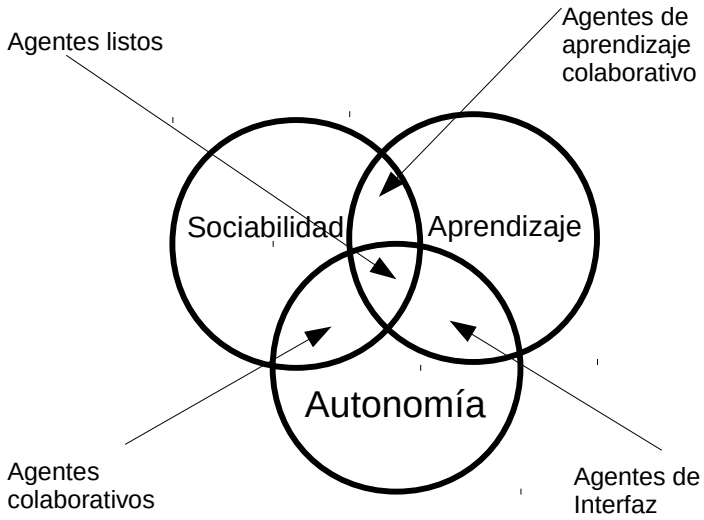


Figura 2.15: Tipología de los agentes (Nwana 1996)

Agentes Colaborativos

Estos agentes poseen autonomía y capacidades sociales que los habilitan para ejecutar las tareas de sus usuarios en colaboración con otros agentes. Los agentes colaborativos son capaces de actuar autónomamente y con un comportamiento racional en ambientes donde existen más agentes y restricciones de recursos. “Crear un sistema que interconecte agentes colaborativos permite al conjunto funcionar más allá de las capacidades de cualquiera de sus miembros”. Esto se expresa de la siguiente manera (Huhns, M. N. & Singh 94d. C.):

$$F\left(\sum agente_i\right) = max\left(F\left(agente_i\right)\right) \tag{2.7}$$

donde F es la función de utilidad y se puede expresar diciendo que la función de utilidad de todos los agentes es siempre superior al máximo de la función de utilizad de cualquiera de los agentes.

Los agentes colaborativos se utilizan para resolver problemas como (Nwana 1996):

1. Cuando los problemas son demasiado complejos para ser resueltos por un agente central.
2. Permitir la interconexión de sistemas de IA existentes como sistemas expertos, sistemas de soporte de decisión.
3. Proporcionar soluciones a problemas inherentemente distribuidos.
4. Proporcionar soluciones que simulen recursos de información distribuidos.
5. Incrementar la flexibilidad, confiabilidad, velocidad modularidad y reutilización en sistemas informáticos.

Agentes de interfaz

Un agente de interfaz se destaca por la autonomía y el aprendizaje.

Un agente de interfaz básicamente presta soporte y asistencia a un usuario que está aprendiendo una aplicación o nuevos conceptos. El ejemplo más claro de este tipo de agentes es el odioso *Clipo* de Microsoft Office. El agente observa y monitoriza, las acciones que hace el usuario en el programa, aprende y sugiere la mejor forma de realizar la tarea.

El agente aprende mediante alguna de las siguientes cuatro técnicas (Nwana 1996):

1. Por observación e imitación del usuario.
2. A través de una retroalimentación por parte del usuario.
3. Recibiendo instrucciones explícitas del usuario.
4. Colaborando con otros agentes.

Los agentes de interfaz se pueden catalogar por el rol que adoptan (Nwana 1996):

- Asistentes de calendario. Puede aprender los hábitos del usuario y no poner reuniones los lunes por las mañanas etc.
- Asistentes de navegación web, como Irene la asistente de RENFE que nos ayuda en el uso de la web.
- Sugeridores. Quizás LastFM sea el más conocido, recibe información sobre nuestros gustos musicales y nos sugiere música de nuestro interés.
- Agentes de Noticia / Filtrado.

Agentes móviles

Los agentes móviles son aplicaciones software capaces de moverse por una red de datos, realizando acciones en nombre del usuario en nodos remotos, y retornando para informar del resultado de sus acciones.

Las ventajas que proporcionan los agentes móviles son (Nwana 1996):

- Reducen el ancho de banda. En algunos casos en los que la tarea del agente implica importantes movimientos de datos, resulta más conveniente ejecutar el código en la fuente de datos y devolver el programa con el resultado deseado.
- Reduce el consumo de recursos. Al poder enviar agentes, no hacen falta recursos *hardware*: memoria, disco.
- Ejecución asíncrona. El cliente lanza un agente móvil y prosigue realizando otras tareas; el agente puede volver en cuanto haya concluido su tarea.

Agentes de información

Debido al nacimiento y crecimiento de la red de redes, han surgido este tipo de herramientas que permiten el manejo y la recuperación de grandes cantidades de información. Nos podemos encontrar en esta categoría con motores de búsqueda (Google, DuckDuckGo), directorios (Yahoo), metabuscadores (Seguros, Viajes).

Agentes reactivos

LA IA tiene mucho camino que recorrer para construir arquitecturas deliberativas eficientes, es por ello por lo que se ha cuestionado la viabilidad del enfoque simbólico y se ha desarrollado un enfoque reactivo. Los agentes reactivos han surgido inspirados por las comunidades sociales de insectos.

Los agentes reactivos, no disponen de estados internos, ni de procesos deliberativos; se rigen por un modelo de estímulo-respuesta. Brooks (1986) propuso tres ideas básicas para desarrollar este tipo de agentes:

1. La conducta inteligente puede ser generada sin usar representaciones explícitas del tipo que usa la IA simbólica.
2. La conducta inteligente puede ser generada sin usar razonamiento abstracto explícito del tipo usado por la IA simbólica.
3. La inteligencia es una propiedad emergente de ciertos sistemas complejos.

Agentes híbridos

Un agente híbrido es aquel que combina varias filosofías. Para algunas aplicaciones los beneficios que puede aportar la combinación de varias filosofías puede ser superior al que está restringido a una sola filosofía. Un ejemplo podría ser un agente colaborativo (deliberativo) y reactivo al mismo tiempo.

Teniendo en cuenta los diferentes tipos de agentes presentados en este apartado, nos damos cuenta de que la implementación de agentes puede ser un proceso nada trivial, ya que, dependiendo del tipo de agente, su estructura interna puede variar sustancialmente. Surge la necesidad de desarrollar metodologías para la construcción de agentes, que especifiquen cómo descomponer el agente, el conjunto de módulos que lo compondrán y su modelo de interacciones. Las arquitecturas incorporan las técnicas y algoritmos que dan soporte a estas metodologías.

Aunque este proyecto se centra en la simulación social y los agentes que se van a desarrollar son tremendamente simples, es conveniente repasar los tipos de arquitecturas para poder comprender el siguiente apartado, ya que la arquitectura interna de los agentes es clave para entender el comportamiento de los sistemas multiagentes.

Una primera clasificación de las arquitecturas puede ser (Nwana 1996):

1. Arquitecturas horizontales. Todos los módulos tienen acceso a los sensores y actuadores. Esto dota de un mayor paralelismo a costa de un mayor coste de control en cada módulo.
2. Arquitecturas verticales. Solo la capa más baja tiene acceso a los sensores y actuadores.

Arquitectura basada en la lógica

Los estados internos de los agentes se representan mediante un conjunto de sentencias de lógica de primer orden, a la hora de tomar decisiones se utilizan las reglas de deducción lógica sobre las sentencias. Su ciclo de funcionamiento se resume, en que los sensores generan una percepción, el componente de revisión de estado consulta la base de datos para ver si existe alguna regla para esa percepción y actualiza el estado, por último, se realizan las acciones asociadas al estado del agente.

Arquitecturas deliberativas

Estos agentes disponen de un modelo simbólico del mundo. Las decisiones se realizan a través del razonamiento lógico basándose en el reconocimiento de patrones y manipulación simbólica. Se basa en el paradigma clásico de la planificación en IA detectar-planificar-actuar.

Arquitectura BDI Believes Desires Intentions

Desarrollada por Michael E. Bratman (1999), esta arquitectura se centra en los roles de las intenciones en el razonamiento práctico. Según este paradigma, los agentes tienen creencias (conocimiento del entorno), un conjunto de deseos, que pueden ser surrealistas e inconsistentes , unos objetivos, que son un subconjunto de los deseos, pero esta vez realistas y consistentes, y unas intenciones, que son un subconjunto de objetivos que el agente ha decidido acometer. Por último, se necesita un plan, un conjunto de acciones necesarias, para lograr las intenciones surgidas de las creencias del agente.

Arquitecturas reactivas

Los agentes disponen de una representación muy sencilla del entorno y disponen de mecanismos que permiten reaccionar instantáneamente ante una percepción. Se basan en el paradigma de comportamiento.

Cada agente está compuesto de módulos autónomos que se encargan de realizar una tarea determinada: sensores, controles, cálculos; existe una mínima comunicación entre módulos y, por supuesto, no hay un modelo global, ni planificador.

En cuanto a los datos que se obtienen por los sensores, no existe un tratamiento complejo de estos datos, se trabaja con los datos en crudo. El comportamiento de estos agentes se puede resumir en que cada agente analiza sus sensores y, si se cumplen las condiciones, actúa.

Arquitecturas de subsunción[1]

Brooks, como ya mencionamos al analizar los agentes reactivos, construyó varios sistemas (robots) basados en arquitecturas de subsunción. Una arquitectura de subsunción es una jerarquía de comportamientos que permiten cumplimentar tareas. Cada comportamiento compite con el resto para tomar control del agente, ya que es posible que se apliquen varios comportamientos simultáneamente.

1 DRAE
subsumir.
(De sub- y el lat. sumĕre, tomar).
1. tr. Incluir algo como componente en una síntesis o clasificación más abarcadora.
2. tr. Considerar algo como parte de un conjunto más amplio o como caso particular sometido a un principio o norma general.

Los diferentes comportamientos se distribuyen en capas: las capas más bajas representan comportamientos primitivos (evitar obstáculos): las capas superiores representan comportamientos más complejos (identificar objetos). Los comportamientos de las capas inferiores tienen precedencia sobre el de las capas superiores (evitamos chichones). Los sistemas resultantes son extremadamente simples, pero pueden impresionar a sistemas desarrollados con IA simbólica.

Arquitecturas híbridas

Según numerosos autores, ni la arquitectura deliberativa ni la arquitectura reactiva son suficientes a la hora de crear un agente, por eso sugieren combinar las dos alternativas en una. Las arquitecturas híbridas se basan en construir agentes con dos subsistemas de diferente nivel cada uno, uno deliberativo y otro reactivo. Normalmente, la capa/subsistema reactivo tiene cierta precedencia sobre la deliberativa.

2.3.5 Principios del modelado con sistemas multiagentes

Macal y North, en el tutorial de modelado y diseño basado en agentes de la revista JOS (Journal of Simulation) (Macal & North 2010), planteaban que en el diseño de todo modelo basado en agentes se tenían que plantear una serie de preguntas:

1. ¿Qué problema quiere resolver el modelo? ¿Qué pregunta quiere responder?
2. ¿Quiénes son los agentes en el modelo?
3. ¿Cuál es el entorno de los agentes?¿Cómo interaccionan los agentes con el entorno?
4. ¿Cuáles son los comportamientos de interés de los agentes?
5. ¿Cómo interaccionan los agentes?
6. ¿De dónde, especialmente debido al comportamiento de los agentes, pueden provenir los datos para este modelo?
7. ¿Cómo podemos validar el modelo, especialmente el comportamiento de los agentes?

Helbing, en su libro “Agent Based Simulations and Experiments to Study Social Behavior” (2012), planteaba una serie de principios, que literalmente recogen la esencia de las cuestiones anteriores[1]:

1 Heldbing en su libro no cita a Macal

1. Hay que describir claramente el fenómeno que se va a reproducir.
2. Hay que explicar cuál es el propósito de la simulación, entender el fenómeno, realizar predicciones, desarrollar una aplicación.
3. Hay que seleccionar correctamente a los agentes; para simular un mercado, por ejemplo, en un modelo empresarial ¿conviene seleccionar un agente por empresa o un agente por cada empleado de la empresa?
4. Una vez especificados los agentes, se deberán formular las hipótesis que rigen el proceso que se va a simular.
5. A la hora de especificar los mecanismos que rigen la simulación multiagente , no se deben poner en el modelo las asunciones que se quieren explicar.
6. Se deben validar los resultados de la simulación con la evidencia empírica.

2.3.6 Herramientas de modelado de sistemas multiagentes

Como hemos visto en la Figura 2.14, todo modelo multiagente está compuesto de tres elementos:

- Un conjunto de agentes con sus atributos y comportamientos.
- Un conjunto de relaciones y modos de interacción (lenguajes).
- Un entorno donde los agentes interactúan.

Para desarrollar un modelo multiagente, todo modelador debe identificar y programar estos tres elementos. Para poder programar estos tres elementos, podemos utilizar un lenguaje de programación de propósito general, o utilizar kits de herramientas o *software* específicamente diseñado para el modelado de agentes.

Si el modelador decide implementar un sistema multiagente mediante lenguajes de programación de propósito general como Java, C++, Python o sistemas de cálculo simbólico como MATLAB o Mathematica, el coste de desarrollo se puede disparar al tener que programar innumerables servicios, que son ofrecidos por plataformas especializadas en el desarrollo de sistemas multiagentes. La gran mayoría de modelos multiagentes a gran escala se desarrollan utilizando *toolkits* o entornos de desarrollo, ya que estos entornos proporcionan:

- Servicios de especificación de proyecto.
- Velocidad de desarrollo.
- Usabilidad.
- Ejecución multiplataforma.
- Capacidades de integración con herramientas GIS.
- Servicios para especificar agentes/entorno.
- Servicios para ejecución del modelo.
- Servicios de análisis de datos.
- Servicios de empaquetado y distribución de modelos.

Los servicios de especificación de proyecto proveen al desarrollador de una forma de determinar qué conjunto de recursos (normalmente ficheros) forman parte del modelo. En función de cuánto soporte ofrece el entorno al modelador, los servicios de especificación pueden proporcionarse mediante:

1. Un conjunto de librerías.
2. Un entorno de desarrollo integrado IDE.
3. La combinación de los dos anteriores: un IDE y un conjunto de librerías.

Cuando se proporcionan un conjunto de librerías, la programación de agentes se realiza siguiendo una API. Los modeladores realizan programas que hacen una serie de llamadas a los objetos y funciones que proporciona la librería. Es responsabilidad del modelador el asegurarse de que la secuencia de llamadas a las librerías se realiza de forma correcta, y de que el conjunto de ficheros necesarios está presente. A cambio, el programador de modelos obtiene una total flexibilidad a la hora de definir sus modelos. En esta categoría de herramientas podemos encontrar MASON (GMU (George Mason University) 2014), Swarm (SGD 2014).

Cuando se proporciona un IDE, se proporciona un editor que permite organizar la construcción del modelo y, normalmente, un compilador o intérprete, que automatiza el proceso de compilación y ejecución. En esta categoría podemos encontrar NetLogo (Wilensky 1999) .

Por último, podemos encontrar entornos de desarrollo que integran una librería con un IDE,

en este caso los IDE suelen ser entornos de software libre como Eclipse. Nos podemos encontrar con herramientas como AnyLogic (XT Technologies 2014) o Repast Simphony (ROAD (Repast Organization for Architecture and Design) 2014).

Los servicios para especificar agentes, proveen al modelador de medios, a la hora de definir los atributos y el comportamiento de los agentes. Estos servicios pueden utilizar lenguajes de propósito general, como C++ o DSLs, visuales, como el editor de flujos de Repast; también pueden proporcionar un soporte especial a la hora de definir características como la adaptación o el aprendizaje (por ejemplo, redes neuronales), optimización (por ejemplo, algoritmos genéticos), redes sociales, Dinámica de Sistemas o integración con herramientas GIS.

2.4 Combinación de las metodologías SD y SMA

2.4.1 Diferencias entre las metodologías

Como hemos visto en las secciones anteriores, ambas metodologías tienen similitudes y diferencias importantes; es precisamente la combinación de estas diferencias las que pueden dar como resultado un marco de simulación híbrido que supere a ambas metodologías por separado en determinados ámbitos. Nadine Schieritz y Peter M. Milling planteaban una interesante comparativa (Schieritz & Milling 2003), que resumimos y ampliamos.

	Dinámica de Sistemas	**Sistemas multiagentes**
Estructura básica	Flujo de realimentación	Agente
Perspectiva	Agregada	Individual
Unidad de análisis	Estructura	Reglas
Soporta la heterogeneidad	No	Sí
Soporte problemas espaciales	No	Sí
Nivel de modelado	Macro	Micro
Tipo de entidades	Interactivas	No interactivas
Perspectiva	Arriba abajo	Abajo Arriba
Comportamiento	Centralizado	Descentralizados
Coste computacional	Bajo	Alto
Tiempo	Continuo	Discreto
Formulación matemática	Ecuaciones integrales	Reglas

Tabla 2.1: Comparación entre SD y SMA

- **Estructura básica: flujo de realimentación / agente**

  La estructura de realimentación nos lleva a un comportamiento generado exógenamente, el tipo de comportamiento con el que trabaja la Dinámica de Sistemas. En la simulación multiagente, la estructura básica es el agente. Un modelo, consiste en una multitud de agentes heterogéneos y su entorno, que en ocasiones suele ser modelado como otro agente más. Cada agente tiene una serie de reglas para interaccionar con el resto de agentes.

- **Perspectiva agregada / individual**

  En los modelos de Dinámica de Sistemas las variables son siempre agregadas, recordemos que un nivel es el acumulador de un flujo. En un modelo multiagente, puede haber variables individuales y variables agregadas, que se obtienen de un grupo de entidades (agentes).

- **Unidad de análisis Estructura/Reglas**

  El comportamiento de un modelo de dinámica de sistemas se determina por la estructura del modelo, es decir, es necesario definir la estructura antes de comenzar la simulación. En los sistemas multiagentes las reglas de cada agente determinan sus interacciones con otros agentes, lo cual hace emerger el comportamiento del sistema macro.

- **Heterogeneidad**

  Los sistemas multiagentes, pueden contener diferentes clases de agentes completamente diferentes y que en determinadas circunstancias pueden interaccionar entre sí.

- **Soporte problemas espaciales**

  Cada agente puede tener atributos espaciales que permitan ubicarlo en un marco de referencia espacial, dando pie a simular fenómenos espaciales tráfico, dispersión, etc.

- **Nivel de modelado Macro / Micro**

  El término macrosimulación puede dar lugar a equívocos, no implica que la Dinámica de Sistemas solo pueda utilizarse para modelar únicamente problemas macroeconómicos o macrosociales. Las variables de dinámica de sistemas siempre son agregadas, esto es un efecto colateral de la técnica de simulación utilizada; un nivel es un acumulador de un flujo, los componentes individuales de ese flujo no pueden ser identificados.

  Los sistemas multiagentes se concentran en modelar poblaciones, centrándose en modelar el comportamiento individual y permitiendo que los individuos interactúen.

- **Comportamiento Centralizado / Descentralizado**

  Los sistemas multiagentes son mayoritariamente descentralizados, ya que el modelador no sitúa en ningún lugar centralizado la definición del comportamiento del sistema; el comportamiento emerge a través de las reglas individuales de los diferentes tipos de agentes y sus relaciones. Como hemos visto previamente la Dinámica de Sistemas necesita conocer y describir la estructura del fenómeno que se va a modelar.

- **Perspectiva Arriba abajo / Abajo arriba**

  En los sistemas multiagentes los fenómenos emergentes son de naturaleza abajo arriba, el comportamiento global viene establecido por los fenómenos a nivel micro que se han definido en cada tipo de agente.

  Si la Dinámica de Sistemas tuviera que estudiar un fenómeno emergente, analizaría sus propiedades, su estructura analizaría el fenómeno en sí.

- **Tiempo continuo / discreto**

  En la literatura sobre agentes no se establece una posición clara al respecto, pero normalmente

se suele aplicar un modelo de tiempo discreto.

- **Formulación matemática**

  Un modelo de Dinámica de Sistemas es un sistema de ecuaciones diferenciales de primer orden.

  En cuanto a los SMAs, no existe un formalismo matemático comúnmente aceptado , la mayoría de los formalismos suelen ser lógicos, como vimos al repasar las arquitecturas lógicas, reactivas, deliberativas.

2.4.2 Diferentes tipos de combinaciones

Gebetsroither (2010) señala que existen cuatro tipos de posibles combinaciones entre las dos metodologías.

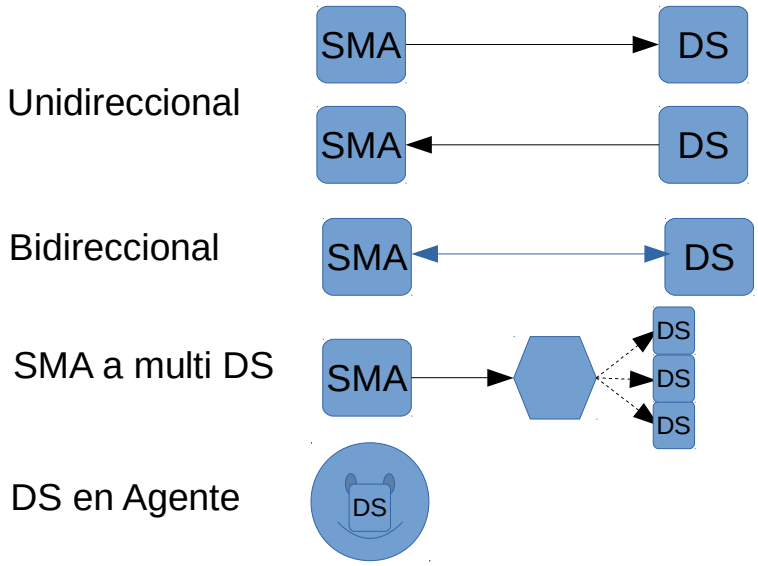


Figura 2.16: Tipos de combinaciones

1. **Unidireccional**

   Este tipo de combinación es el más simple. Se trata de considerar que el modelo está compuesto de dos modelos independientes, tanto a nivel de tiempo como de espacio. Solamente hay que gestionar la introducción de los datos de un modelo a otro. La

validación y verificación, usando una combinación unidireccional, es muy sencilla.

2. **Bidireccional**

   Se trata de la combinación más compleja, donde pueden emerger flujos de realimentación complejos. En este tipo de combinación, ambos modelos deben estar sincronizados, ya que cada sistema influencia al otro. La validación y verificación resulta compleja, debido a que en cada paso el *input* de un modelo SMA a uno de DS varía debido a la naturaleza estocástica del SMA.

3. **SMA a múltiples DS**

   Se trata de una combinación unidireccional a múltiples modelos de DS. A diferencia del modelo unidireccional, no todas las relaciones son simultáneas. Si analizamos la Figura 2.16, el hexágono intermedio vendría a ser una puerta OR, que en función de una serie de condiciones o eventos informa al modelo DS correspondiente.

4. **Dinámica de Sistemas en agentes**

   En este caso, el propio agente dispone de un modelo en dinámica de sistemas para hacer cambiar sus estados. El coste comunicacional es mayor que en el caso bidireccional y, en algunos casos como en los SSBA, puede resultar no asumible. La validación y verificación solo es posible para la combinación de ambos modelos, no se puede realizar por separado como en el caso unidireccional.

2.5 Comparación de entornos de simulación híbridos

2.5.1 Introducción y objetivos

En este apartado vamos a realizar una presentación de las principales herramientas de simulación híbrida con soporte GIS, continuaremos con una comparativa de las plataformas AnyLogic, NetLogo y Repast.

2.5.2 AnyLogic

AnyLogic es una herramienta de simulación multimétodo que soporta tres paradigmas de modelado: la Dinámica de Sistemas, simulación de eventos discretos y la simulación multiagente. Un único modelo de AnyLogic admite cualquier combinación de estos enfoques.

AnyLogic incluye un lenguaje de modelado visual y permite la extensión de los modelos mediante el lenguaje de programación Java.

2.5.3 NetLogo

Inicialmente conocida como StartLogoT, es una plataforma de alto nivel que proporciona un potente y sencillo lenguaje de programación, dispone de un interfaz sencillo y amplia documentación. Su facilidad de uso y su pequeña curva de aprendizaje lo convierten en líder como herramienta educativa.

Se creó con un modelo específico en mente, agentes móviles que interaccionan localmente en una matriz a lo largo del tiempo. Aunque resulta sencillo implementar modelos de este tipo, NetLogo no solo está limitado a ellos y, por sus características y extensa documentación, resulta una herramienta tremendamente profesional.

2.5.4 Repast Recursive Porous Agent Simulation Toolkit

Repast surgió como una implementación JAVA de SWARM[1], sistema multiagente pionero creado por el Santa Fe Institute (SFI), posteriormente divergió significativamente.

Permite el estudio del comportamiento de sistemas complejos a través de experimentos informáticos replicables. Repast, provee al usuario de un cuerpo de clases para el desarrollo de simulaciones basadas en agentes y una colección de librerías para la presentación de tablas, diagramas, etc. También tiene la posibilidad de integrar datos provenientes de herramientas GIS directamente en la simulación.

Repast también pone a disposición del usuario una implementación DS con integración mediante Euler y Runge-Kutta (RK4).

2.5.5 Otros entornos

MASGISmo

MASGISmo[2] cuyo nombre proviene de "Multimethod Agent-based System Dynamics, GIS modeling platform", fue creado por la confluencia de varios proyectos de investigación para permitir combinar una simulación *bottom up* usando un sistema multiagentes, y un método *top down* usando Dinámica de Sistemas, y para permitir tratar información GIS.

La plataforma está programada en JAVA y se conecta con otras herramientas y tecnologías: usa como gestor de base de datos PostgreSQL[3], usa VensimPLE como herramienta de Dinámica de Sistemas utilizando su capacidad de conexión mediante dll, y usa Repast como corazón del sistema multiagente. En cuanto a su capacidad GIS, permite la integración con datos y funcionalidades PostGIS, así como importar y exportar capas vectoriales ESRI[4].

Al tratarse de un marco de trabajo dependiente de otras tecnologías, no consideramos que se

1 http://savannah.nongnu.org/projects/swarm
2 La herramienta no dispone de una página web, más info en la web del Austrian Institute of Technology http://www.ait.ac.at/
3 http://www.postgresql.org/
4 El formato ESRI Shapefile (SHP) es un formato de archivo informático propietario de datos espaciales desarrollado por la compañía ESRI, quien crea y comercializa software para Sistemas de Información Geográfica como Arc/Info o ArcGIS.

pueda catalogar como herramienta de simulación híbrida con capacidad GIS y por eso la hemos excluido de esta comparativa.

GAMA

Gama[1] es una plataforma completa de modelado y simulación que ofrece un entorno de desarrollo para generar simulaciones multiagente espaciales. La característica principal es la posibilidad de usar datos provenientes de aplicaciones GIS como entorno de los agentes. GAMA propone un lenguaje de modelado orientado a objetos, llamado GAML, para modelar agentes y entornos. Al estar integrado con Eclipse, propone un editor gráfico basado en el plugin de Eclipse Graphiti que permite definir el modelo mediante un interfaz gráfico, también permite crear diagramas gráficos a partir de modelos Gaml.

Gama se define como un entorno multinivel capaz de realizar modelos multiparadigma (máquinas de estados finitos, arquitecturas de control, ecuaciones matemáticas), por lo cual lo mencionamos en este apartado, aunque debido a su no integración explícita con la Dinámica de Sistemas lo hemos dejado de lado en la comparativa.

2.5.6 Comparación de características

La comparativa que se presenta a continuación, se ha realizado consultando las *webs*, los manuales de las herramientas y examinando cada uno de los entornos.

Algunos de los criterios de la comparativa están inspirados en el artículo Tools of the Trade: A Survey of Various Agent Based Modeling Platforms (Nikolai & Madey 2009), y otros han sido propuestos de acuerdo a los objetivos de este proyecto. Las características que se examinan en la comparativa son: los paradigmas soportados, el tipo de licencia, tratamiento del tiempo, lenguaje de programación, posibilidad, uso de diagramas, sistema operativo, ámbito de utilización, soporte, curva de aprendizaje, soporte a cambios en caliente, soporte GIS.

1 https://code.google.com/p/gama-platform/

1. Paradigmas soportados

	Dinámica de Sistemas	Modelado basado en agentes	DEVS
AnyLogic	✓	✓	✓
NetLogo	✓	✓	✗
Repast	✓	✓	✗

Tabla 2.2: Paradigmas soportados

Aunque tanto en NetLogo como en Repast son simuladores de tiempo discreto y se pueden implementar modelos DEVS, el entorno no ofrece soporte a entidades como colas o bloques de servicio, por lo que hay que crearlos a partir de cero.

2. Tipo de Licencia

- **AnyLogic**. Tiene una licencia propietaria.
- **NetLogo**. GNU General Public License v2[1] o superior.
- **Repast.** La suite Repast y su documentación están licenciadas con la licencia New BSD[2].

3. Tratamiento del Tiempo

- **AnyLogic**. El tiempo no es discreto como en NetLogo, en AnyLogic los agentes pueden programar tareas y ejecutarlas en cualquier momento.
- **NetLogo**. El tiempo es discreto, los agentes solo pueden realizar acciones entre los ticks.
- **Repast.** Es un simulador de tiempo discreto. Al igual que NetLogo, su unidad de tiempo se llama tick.

4. Lenguaje de programación

- **AnyLogic.** AnyLogic incluye un lenguaje de modelado gráfico y también permite que los

1 GNU GPL v2 http://www.gnu.org/licenses/gpl-2.0.html
2 New BSD o BSD 3- Clause License http://opensource.org/licenses/BSD-3-Clause

usuarios puedan ampliar los modelos de simulación con código de Java. La aplicación también permite exportar sus modelos a applets Java.

- **NetLogo.** Tiene su propio dialecto del lenguaje de programación Logo.
- **Repast.** RePast cuenta con su propio lenguaje de programación para principiantes llamado Relogo, que permite un prototipado rápido para desarrollar modelos basados en agentes de forma rápida. Relogo fue escrito usando el lenguaje de programación Groovy[1] con lo que también permite código en Groovy.
- Repast permite programar sus modelos indistintamente en Relogo o en Java haciendo uso del entorno de programación Eclipse. Si necesitamos correr un modelo en un Supercomputador, Repast dispone de un *toolkit* (Repast HPC) para desarrollar modelos en plataformas distribuidas de alto rendimiento. Repast HPC puede programarse usando C++ o un "l*ogo-style*" C++.

  Existen otras implementaciones de Repast: RepastPy (basada en el lenguaje Python), Repast.Net, implementado en el lenguaje C#. La versión que estamos comparando es Repast Symphony, basada en Java y diseñada para estaciones de trabajo y pequeños clústeres.

5. Uso de diagramas

Este criterio de comparación permite conocer si se pueden diseñar y ejecutar modelos de simulación mediante diagramas (sin necesidad de programar). Es importante destacar que todos los entornos disponen de un diseñador de Dinámica de Sistemas; este criterio se refiere al uso de diagramas a la hora de diseñar el modelo en global.

- **Repast**. Se permite diseñar los estados de los agentes mediante diagramas, así como sus propiedades y comportamientos.
- **NetLogo**. No soportado.
- **AnyLogic.** Realizar modelos mediante diagramas, es la forma habitual de trabajar con AnyLogic.

1 http://groovy.codehaus.org/

6. Sistema Operativo entornos de ejecución

Al estar los tres entornos comparados basados en Java, pueden correr en cualquier sistema operativo en el que el entorno de ejecución JAVA esté disponible.

- **AnyLogic**. Java Runtime Enviroment (JRE) 1.5.0 o superior.
- **NetLogo.** Java Development Kit (JDK) 1.5.
- **Repast.** Java 6, parece que Java 7 tiene problemas.

7. Ámbitos de utilización

- **AnyLogic.** Propósito general basado en agentes.
- **NetLogo.** Propósito general ideal para enseñanza de modelado.
- **Repast**. Especialmente indicado para ciencias sociales.

8. Soporte / Comunidad

En este apartado analizaremos qué soporte hay en cuanto a la calidad de la documentación, ejemplos, lista de distribución, actividad de la comunidad.

	AnyLogic	NetLogo	Repast
API	-	Notable	Notable
Documentación	Notable	Notable	Notable
Wiki	-	No funciona	-
FAQ	Notable	Notable	-
Ejemplos		Notable	-
Tutoriales, Videotutoriales	Notable	Notable	Notable
Extensiones	No	Notable	Notable
Listas de distribución, foros	Sí	Sí	-
BugList	No	Sí	Sí
Consultoría	Sí	No	-
Biblioteca de modelos	Bien	Sobresaliente	Apto
Publicaciones de Referencia	Sí	-	-

Tabla 2.3: Comparativa documentación

9. Curva de Aprendizaje

    - **AnyLogic** Aunque en principio se pueden diseñar modelos sin necesidad de introducir código, gracias al diseñador de interfaces gráfico, la interfaz y sus elementos resultan complejos y es necesario invertir bastante tiempo para dominarla; además, para poder realizar cualquier modelo medianamente complejo, resulta imprescindible tener amplios conocimientos de Java y de las librerías de AnyLogic.
    - **NetLogo**

      El diseñador de interfaces, es mucho más sencillo que AnyLogic ya que solo contiene un grupo de controles reducidos y elementales. El lenguaje, por su parte, al ser procedimental resulta más sencillo de aprender que Java.
    - **Repast**

      Repast se programa utilizando el IDE eclipse, con lo que, si no se conoce bien este editor, hay un coste adicional, aunque, si se conoce, parte del trabajo está hecha. En cuanto al lenguaje, tanto Groovy como Java son más complejos que NetLogo, e iguales que AnyLogic.

10. Cambios en caliente

    Nos referimos a la capacidad que tienen los entornos de cambiar los parámetros de una simulación en ejecución sin tener que detenerla.

    - **AnyLogic**. Permite cambiar los parámetros del modelo en caliente.
    - **NetLogo**. No soportado.
    - **Repast**. Permite cambiar las propiedades de los agentes, las ecuaciones de comportamiento y los parámetros del momento en tiempo de ejecución.

11. Soporte GIS

    - **AnyLogic**

      AnyLogic soporta mapas provenientes de OpenMap(TM)[1] que estén definidos con el

1 https://code.google.com/p/openmap/

sistema de coordenadas WGS 84[1]. Openmap es un conjunto de herramientas de programación de código abierto en Java que permite añadir a las aplicaciones capacidades de visualización de mapas GIS.

- **NetLogo**

  Gracias a la extensión GIS de NetLogo, podemos importar vectores y *rasters* en ficheros shape (.shp) con formato ESRI en nuestro proyecto. La extensión GIS no soporta todos los sistemas de coordenadas, solamente se soportan los sistemas GEOGCS.

- **Repast**

  Existen dos sistemas GIS que se pueden usar con Repast ESRI ArcMap y OpenMap. ArcMap está relacionado con el software GIS comercial ArcGIS[2] y OpenMap con el proyecto OpenStreetMap[3]. La diferencia entre estos sistemas se basa en el tipo de objeto geometría que se utiliza.

  - Si elegimos ArcGIS, se utilizará **`com.vividsolutions.jts.geom`** y el entorno nos proporcionará la clase de utilidad GeotoolsData para leer y escribir vectores y las utilidades básicas para crear polígonos, líneas, posicionar agentes, etc. Es necesario disponer de ArcMap[4] y su respectiva licencia para poder trabajar de este modo.
  - Si elegimos OpenMap, se utilizará **`com.bbn.openmap.omGraphics.OMGraphi`** y el entorno proporcionará la clase OpenMapData. Repast trabaja en este modo con el JavaBean OpenMap(tm). Al igual que en el caso anterior, la biblioteca OpenMapData ofrece las utilidades básica ofrecidas por la anterior.

2.5.7 Conclusiones

No podemos concluir que una de las plataformas comparadas sea la mejor, pero sí podemos concluir que cada plataforma puede ser más adecuada en función del modelo que se va a simular y el modelador que lleve a cargo el modelado. Por este motivo las conclusiones que vamos a ofrecer en

1 http://www.anylogic.com/anylogic/help/
2 https://www.arcgis.com/features/
3 http://www.openstreetmap.org/ Los datos de OpenStreetMap son abiertos (libres)
4 ArcMap es el componente principal de la suite Geospacial ArcGIS

esta sección se hacen para cada plataforma.

AnyLogic

El lenguaje visual de programación que recuerda a UML facilita el desarrollo, los diagramas de estado facilitan la definición del comportamiento de los agentes y los diagramas de acción son útiles para definir algoritmos. Los usuarios pueden escribir siempre código Java para modelar algo más específico. La integración con modelos de Dinámica de Sistemas y eventos discretos es directa. La plataforma está bien documentada y hay múltiples recursos en la web.

El punto débil de esta plataforma es el licenciamiento, no existe una licencia libre y la licencia básica para estudiantes tiene un coste elevado.

NetLogo

NetLogo es citado habitualmente como la plataforma más apropiada para el ámbito educativo, dispone de una excelente documentación. Su lenguaje resulta atractivo para los simuladores sociales, que no dominan la programación orientada a objetos.

Es recomendable para simulaciones de agentes en un entorno de matriz no demasiado complejo. También puede resultar recomendable como plataforma de prototipo para después realizar modelos con más detalle. Aunque puede que el rendimiento no sea su punto fuerte, el ahorro de tiempo que se produce a la hora de programar lo compensa.

Los programadores experimentados pueden sentirse incómodos al tener que programar todo en el mismo fichero; además, aunque el entorno es muy completo, carece de un depurador de código.

Repast

Repast se centra en el modelado del comportamiento social aunque se puede extender a otros ámbitos fácilmente.

Repast es una plataforma muy completa y dispone de una amplia biblioteca que cubre casi la totalidad de funciones exigidas a un entorno de programación de este tipo. Al estar basado en Groovy /Java, la curva de aprendizaje se acentúa, especialmente en aquellos simuladores sociales sin experiencia en programación.

2.6 Trabajos similares

Como mencionábamos en el Capítulo 1, existen muy pocos ejemplos de modelos que combinan varios paradigmas y hacen uso de información espacial.

Si realizamos un repaso, en los repositorios de modelos, de los entornos de desarrollo que vamos a comparar en el siguiente capítulo, nos encontramos con:

NetLogo

En la biblioteca de modelos de NetLogo http://ccl.northwestern.edu/netlogo/models/ nos encontramos con un único modelo híbrido Tabonuco yagrumo Hybrid (Wilensky 2006a). Este modelo simula la competición existente entre dos especies de árboles por ocupar los claros del bosque. La especie tabonuco crece más lentamente que el yagrumo, pero vive más tiempo. La competencia comienza cuando un huracán destruye el bosque dejando claros. El modelo dispone de dos variables de nivel, tabonuco y yagrumo que representan las poblaciones de sendas especies y que se alimentan de un sistema basado en agentes que simula el crecimiento y las catástrofes en el entorno. Por otro lado, se han incorporado al modelo, los niveles de carbono y nitrógeno generados por el bosque y que se calculan utilizando un modelo de Dinámica de Sistemas sobre las variables de nivel tabonuco y yagrumo.

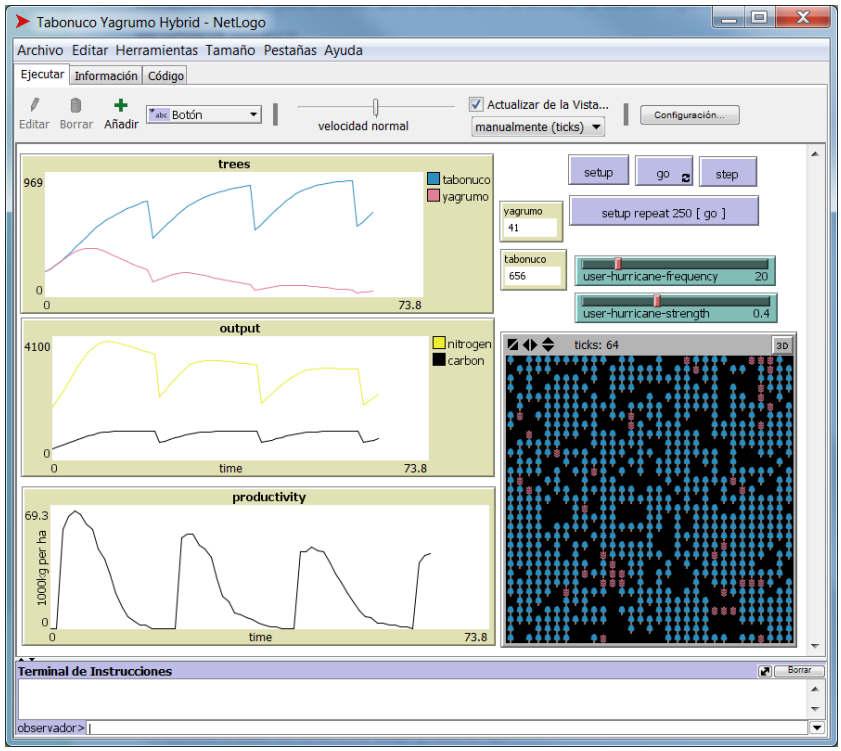


Figura 2.17: Modelo Tabonuco Yagrumo

AnyLogic

En la biblioteca de modelos podemos encontrar un par de ellos:

1. El primer modelo es el utilizado en el tutorial "How to Build a Combined Agent Based / System Dynamics Model in AnyLogic" (Anón 2008). En este modelo existen dos productos alternativos manufacturados por dos empresas rivales, que cuentan con su propia cadena de distribución cada una. Los consumidores, al inicio, no utilizan ninguno de los productos, pero son influenciados por la publicidad y el efecto boca a boca. Ambos productos tienen una durabilidad y, al finalizar, el consumidor puede optar por la otra marca en función de la disponibilidad en la cadena de distribución. El mercado es modelado usando una metodología basada en agentes y las cadenas de distribución se modelan usando Dinámica de Sistemas.
2. El segundo muestra la dinámica de una población de animales marinos en un área pequeña. El área de interés está dividida en una matriz 2D de celdas. La población de individuos maduros de cada celda está influenciada por la tasa de maduración,la muerte local así como por la migración desde otras celdas. Cada celda es un agente cuya dinámica se caracteriza mediante un modelo de Dinámica de Sistemas. Las celdas distribuidas en una

matriz tienen contacto con sus vecinos en las cuatro direcciones cardinales.

Si ampliamos las miras y no nos limitamos a las herramientas analizadas en este proyecto, el trabajo más reciente e interesante es el modelo multimétodo de simulación para el soporte en las decisiones de planificación Urbana (Gebetsroither 2013). En este capítulo se explican cuáles son las ventajas y desventajas de combinar varios métodos de modelado y se presenta la plataforma de simulación especializada MASGISmo, una herramienta que permite la integración con VensimPLE de herramientas GIS y sistemas basados en agentes. La discusión teórica se sustenta en resultados obtenidos en diferentes simulaciones. El capítulo ilustra cómo un modelo multimétodo de desarrollo urbano puede contribuir a que los planificadores urbanos y legisladores realicen planes urbanísticos más robustos y argumentados.

Siguiendo en el mismo campo, podemos analizar el trabajo realizado por la Universidad de Riga, donde se nos presenta un modelo de desarrollo comunitario sostenible, COMMOD (comunity-modelling). En este trabajo se propone un marco de trabajo que integra Dinámica de Sistemas, sistemas multiagentes y sistemas de información geográficos, con el propósito de ayudar a evaluar planificaciones urbanísticas sostenibles para la región de Kurzeme.

2.7 Conclusiones

En este capítulo hemos resumido las metodologías de Dinámica de Sistemas y sistemas multiagentes, haciendo hincapié en sus diferencias y la diversas maneras de integración que se pueden plantear.

Hemos analizado y comparado varias plataformas de simulación multimétodo con soporte GIS y hemos concluido que no existe una plataforma más ventajosa que el resto, cada plataforma puede resultar más adecuada en función del sistema que se va a simular.

También hemos realizado un tutorial del entorno de modelado NetLogo, así como de su extensión GIS que permite importar datos en formato (.shp). Hemos hecho una breve introducción y un tutorial de la herramienta Quantum GIS, que permite confeccionar y modificar *shapes* para su posterior uso en NetLogo.

Hemos realizado una búsqueda de modelos multimétodo con soporte GIS y hemos visto que existen muy pocas referencias. Creemos que es debido a la complejidad que supone realizar un modelo que implica tantas herramientas y tecnologías.

3 Introducción a NetLogo

3.1 Introducción

Este capítulo está dedicado a introducir NetLogo, la herramienta con la que se ha confeccionado el modelo de este proyecto. Aunque este capítulo está íntimamente ligado con el Capítulo 2, se ha confeccionado de una manera desacoplada del resto de la memoria, ya que uno de los objetivos del proyecto es la generación de un tutorial sobre NetLogo. El capítulo se ha redactado siguiendo un estilo de tutorial y teniendo siempre en mente su posible publicación en Internet de manera independiente del resto del trabajo.

NetLogo es un lenguaje de programación basado en un dialecto del conocido lenguaje de programación Logo y también un entorno de simulación.

NetLogo permite simular sistemas complejos mediante agentes interdependientes y ver cómo evoluciona el sistema desde un punto de vista macroscópico a partir de una programación microscópica, es decir, del individuo o del agente.

NetLogo ha sido diseñado teniendo en mente que sus usuarios son desde jóvenes aprendices de programación hasta expertos profesionales en simulación. Su curva de aprendizaje es extremadamente corta y sus funcionalidades son amplias y permiten desarrollar simulaciones complejas en ámbitos tan diferentes como economía, psicología, biología, química, informática, ingeniería, etc.

NetLogo dispone de un creador de interfaces que permite modificar los parámetros y visualizar los resultados de la simulación, así como numerosos complementos que permiten enlazar con otras muchas herramientas informáticas.

NetLogo, proporciona una librería de modelos funcionales listos para ser ejecutados o modificados por el usuario de una forma sencilla. Al estar basado en Java, los modelos de NetLogo pueden ejecutarse en cualquier navegador con soporte Java, facilitando la creación y distribución de modelos.

3.2 El entorno de simulación NetLogo

El entorno de NetLogo, está compuesto por una barra de menús y tres vistas: “Ejecutar”, “Información” y “Código”, accesibles mediante las pestañas que se encuentran bajo la barra de menús.

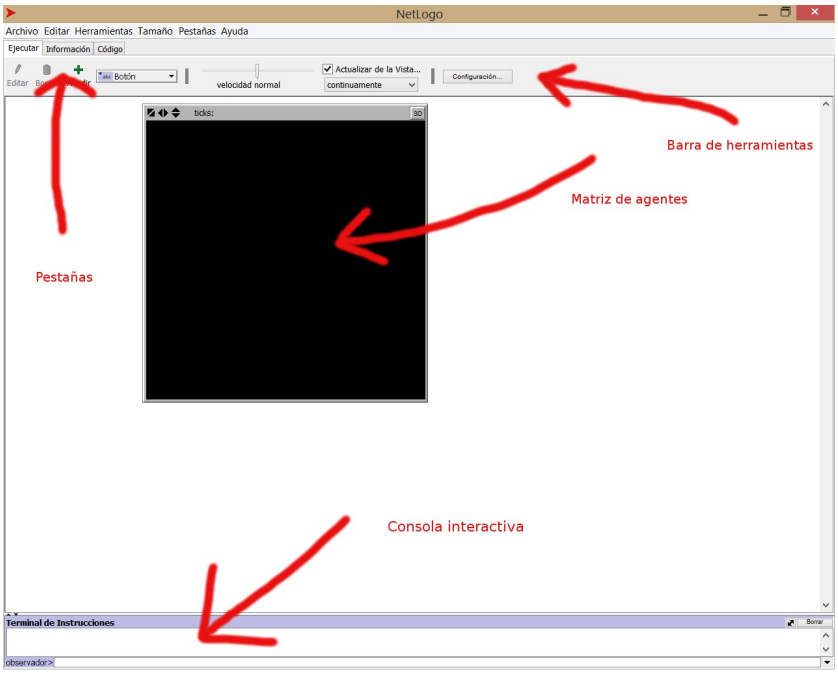


Figura 3.1: Interfaz principal de NetLogo

1. La primera pestaña, “Ejecutar”, muestra la interfaz de ejecución, donde se muestran los controles que dispone nuestro simulador y una barra de controles que permite arrastrar nuevos controles al simulador.
2. La vista Información es un editor de texto que permite ver o modificar información sobre el modelo en el que estamos trabajando.
3. Por último, la tercera vista “Código” nos da acceso al código del modelo.

La pestaña de “Ejecutar” a su vez se divide en:

1. Una consola interactiva en la parte inferior, donde podremos ejecutar comandos en tiempo real para realizar diversas tareas, como obtener valores de variables en ejecución, etc.

2. La matriz de agentes, en la parte central, llamada mundo ,y, a su alrededor, espacio en blanco donde podremos situar los controles que parametricen la ejecución de la simulación.
3. En la parte superior disponemos de una barra de herramientas, que nos permite insertar diferentes controles en la parte central.

La pestaña “Información”:

Nos muestra información sobre el modelo con el que estamos trabajando. El botón “editar” nos permite editar el contenido de esta ficha directamente. El editor utiliza un lenguaje de marcado ligero, llamado Markdown, que se utiliza en diferentes ámbitos; es similar al lenguaje de marcado MediaWiki utilizado por la Wikipedia.

Ejemplos de Markdown:

```
# Encabezamiento de primer nivel
### Encabezamiento de tercer nivel
* Elemento de una lista desordenada
* Otro elemento
2. Elemento de una lista enumerada
2. Otro elemento ordenado
`código`
```
Código en varias líneas
```
>Citas
[Texto de un enlace] (URL enlace “titulo del enlace”)
![Texto alternativo de una imagen] (URL de la imagen “Título de la imagen”)
*cursiva*
**negrita**
```

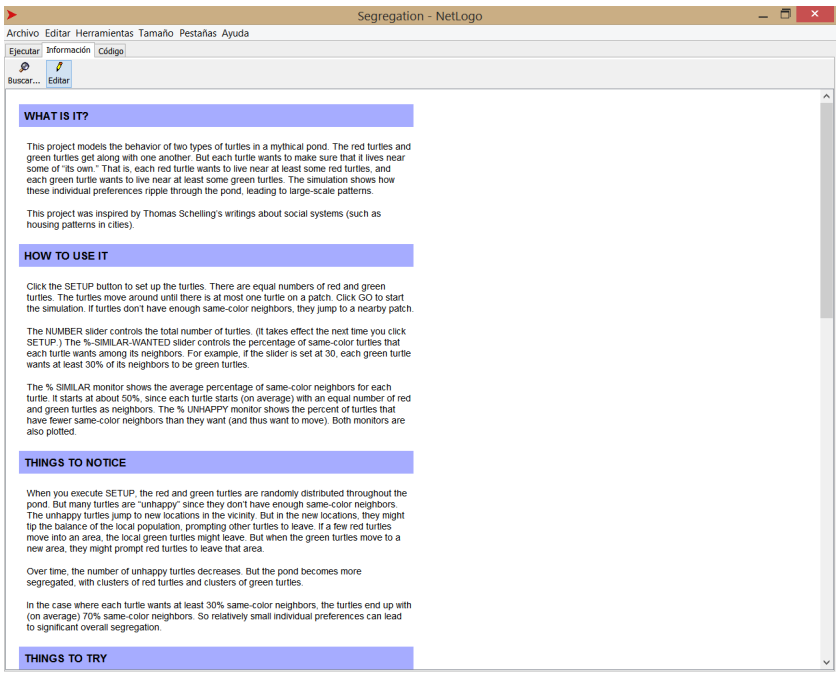


Figura 3.2: Pestaña de Información

La pestaña “Código”:

Como el nombre indica, permite editar el programa en el lenguaje NetLogo. Dispone a su vez de:

1. Un botón **Check,** que permite realizar una validación sintáctica del código introducido.
2. Un selector de procedimientos, que nos da acceso directo a los diferentes procedimientos disponibles en el código.

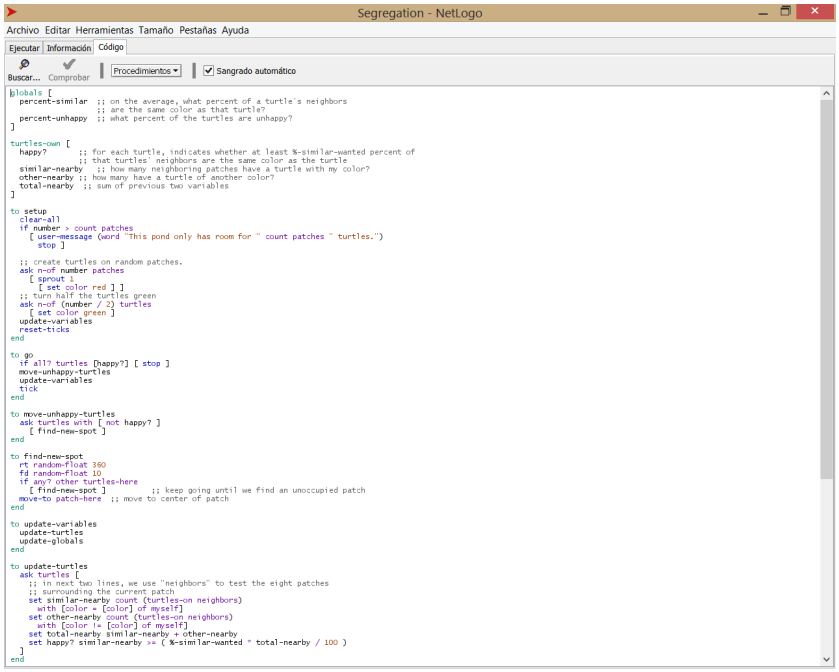


Figura 3.3: Pestaña código.

3.3 Componentes de NetLogo

3.3.1 El Mundo

El elemento central de la pestaña ejecución es el mundo. Se trata de una matriz donde los diferentes tipos de agentes interaccionan entre sí. El mundo está compuesto por baldosas “patches” por donde las tortugas se desplazan. Las baldosas pueden ser localizadas mediante un sistema de coordenadas. El origen, por defecto 0,0, se sitúa en el centro (este punto es configurable como se puede apreciar en la Figura 3.4: El mundo, la matriz de agentes.). El tamaño de cada eje, se establece con el parámetro max-pxcor, máxima coordenada x para las baldosas, y en el caso de que situemos el eje en 0,0 y establezcamos un max-pxcor=55 y maxpycor=55, el mundo tendrá unas dimensiones de

(55x2+1)^2 baldosas 111x111.

Entre las opciones más destacables que se pueden definir, se puede establecer si el mundo se comporta como un toroide tanto horizontal como vertical. También se puede establecer el tamaño de parcela, lo cual es muy útil ya que permite escalar las parcelas para que mundos de tamaño grande puedan visualizarse en pantallas estrechas.

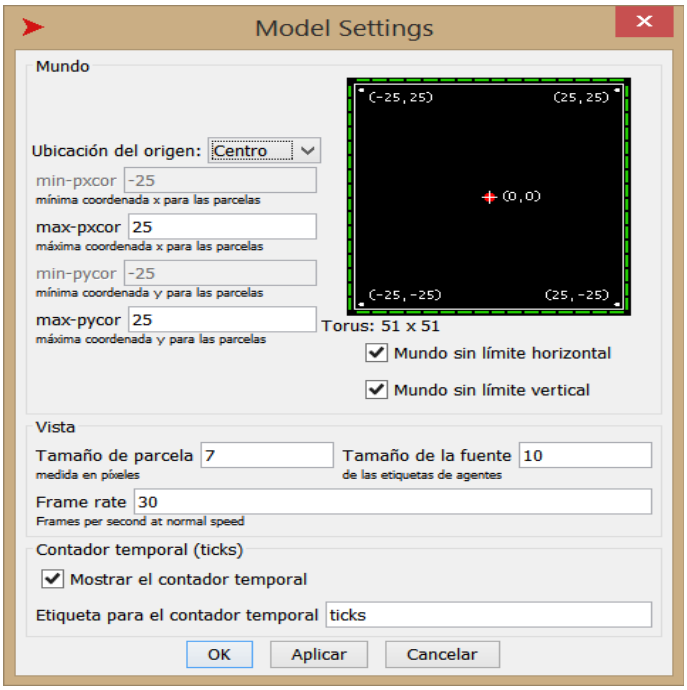


Figura 3.4: El mundo, la matriz de agentes.

3.3.2 Los Agentes

En NetLogo existen cuatro tipos de agentes:

1. Las tortugas “turtles”: son los agentes móviles que se mueven por el mundo.
2. Las baldosas “patches”: son las casillas que componen el mundo como; hemos visto antes, componen un tablero por donde las tortugas se desplazan.
3. Los enlaces “links”: son las conexiones orientadas o no que se establecen entre dos tortugas.
4. El observador.

Tortugas

Los agentes móviles en NetLogo, son denominados tortugas, ya que en el idioma original Logo al cursor se le llamaba tortuga.

Toda tortuga dispone de 13 atributos básicos, a los que se les puede añadir un número indeterminado de atributos personalizados (Wilensky 2006b).

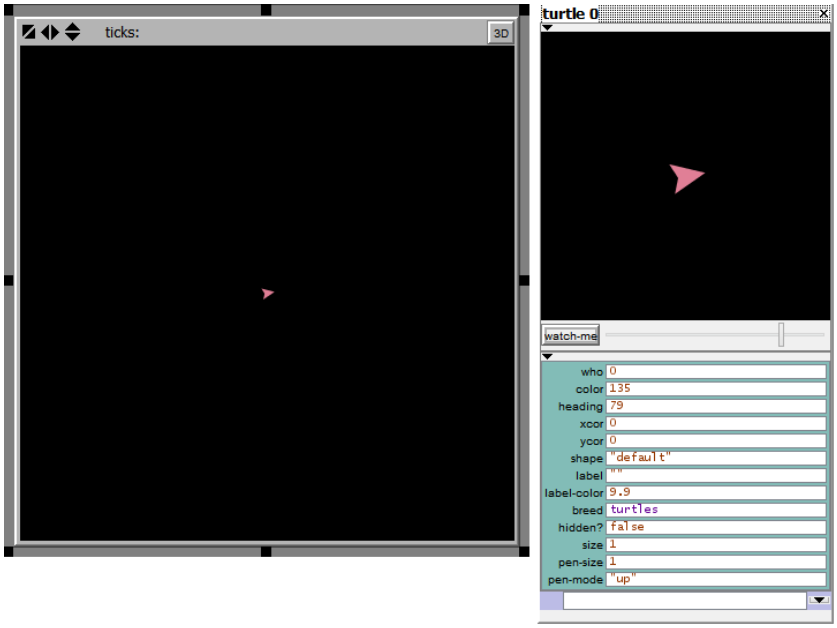


Figura 3.5: Propiedades de una tortuga.

who	Identificador numérico de la tortuga
color	
heading	Orientación
xcor	Posición en el eje de abscisas
ycor	Posición en el eje de ordenadas
shape	Forma
label	Etiqueta
Label-color	Color de la etiqueta
breed	Raza a la que pertenece la tortuga
hidden?	Estado visible/invisible
size	Tamaño
pen-size	Tamaño de la etiqueta
pen-mode	Posición de la etiqueta

Tabla 3.1: Propiedades de las tortugas.

A cada tortuga, se le pueden añadir nuevos atributos mediante la expresión turtles-own [atributonuevo];

```
turtles-own [ edad energia ]
```

Podemos inspeccionar los atributos individuales de una tortuga; pulsando encima de ella, se nos mostrará una ventana como la 3.5, donde aparecerán todas las propiedades del agente seleccionado.

Comandos básicos

ht Oculta la tortuga

st Muestra la tortugas

pu Levanta el lápiz de la tortuga (si lleva lápiz para dibujar el camino que recorre)

pd Baja el lápiz

pe pone borra-goma

crt X crea X tortugas

Para indicar a una tortuga que ejecute una sentencia debemos seleccionarla.

```
ask turtle 5 [ pd fd 5 pu ]
```

Comandos de movimiento

```
ask turtle 5 [ move-to patch 0 0 ]
ask turtle 5 [ set xcor 0 set ycor 0 ]
ask turtle 5 [ setxy 0 0 ]
ask turtle 5 [ face patch 5 5 ];;indicamos a la tortuga 5 que mira a la baldosa
5 5
ask turtle 5 [ facexy 5 5 ]
ask turtle 0 [ set heading 45 ]
ask turtle 10 [ fd 10 ] ;; 10 pasos hacia adelante
ask turtle 10 [ bk 10 ] ;; 10 pasos hacia atrás
ask turtle 10 [ rt 30 ] ;; giro derecha 30 grados
ask turtle 10 [ lt 45 ] ;; giro izquierda 45 grados
```

Código 3.1: Comandos de movimiento de NetLogo

Baldosas

Las baldosas o “patches” son un tipo de agente inmóvil, por donde las tortugas se desplazan. Vienen a ser las celdas de un tablero de ajedrez por donde las fichas se mueven.

Como en el caso de las tortugas, cada baldosa dispone de 5 atributos:

- `pxcor`
- `pycor`
- `pcolor`
- `plabel`
- `plabel-color`

Para añadir atributos adicionales a las baldosas, deberemos ejecutar el comando

```
patches-own [ penergia pprofundidad ptemperatura ]
```

Para alterar los atributos, utilizaremos el comando set VARIABLE EXPRESSION

```
;; establcemos que todos los patches tienen una pprofundidad de 100
ask patches [ set pprofundidad random 100 ]
```

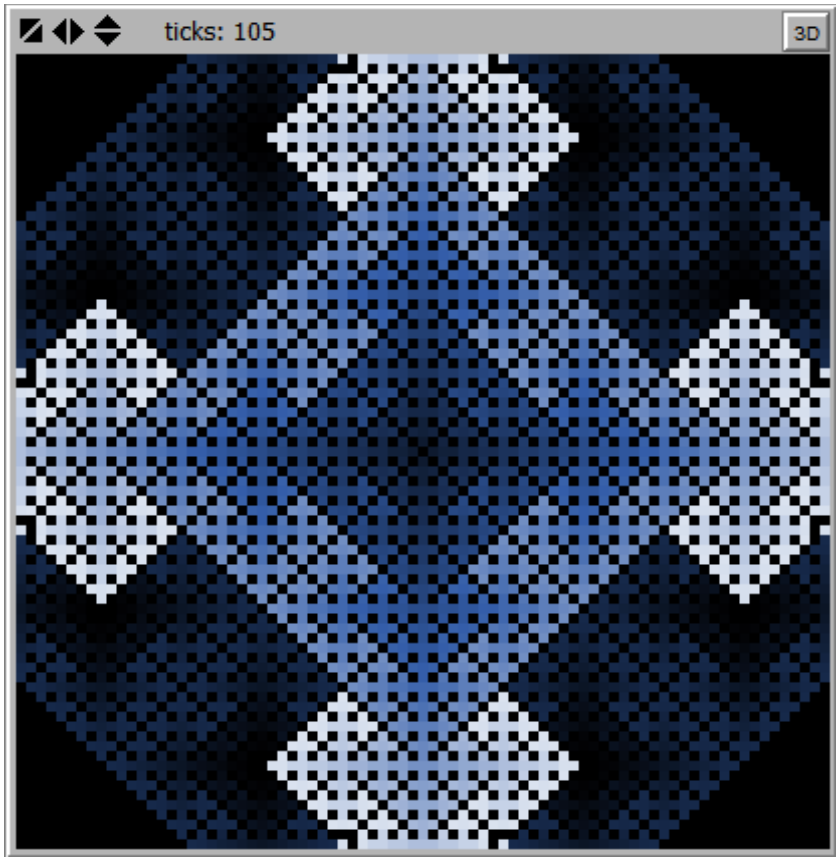


Figura 3.6: Patches o baldosas que componen el mundo – Cells model (Wilensky 2014c)

Podemos inspeccionar cualquier celda haciendo clic sobre ella, NetLogo nos mostrará una ventana con las propiedades de la celda seleccionada.

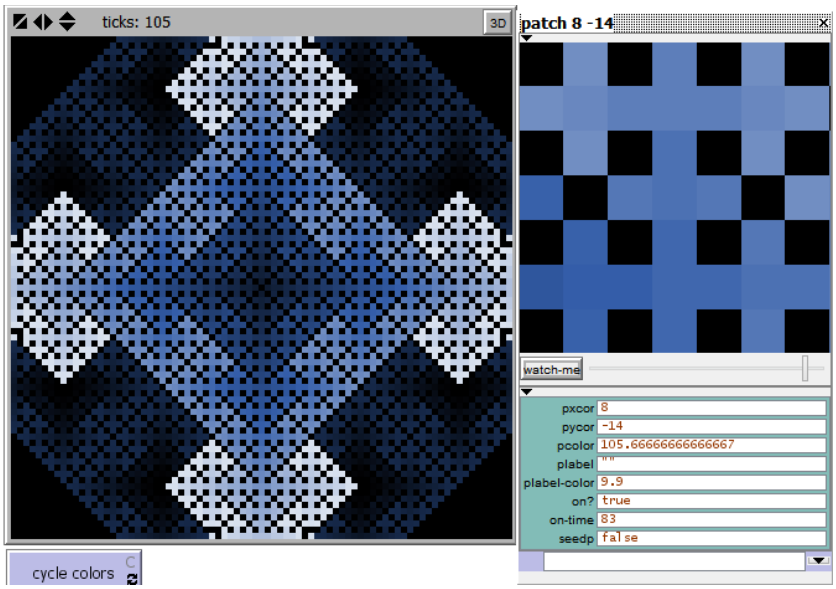


Figura 3.7: Propiedades de las baldosas – Cells model (Wilensky 2014c)

En este ejemplo, las coordenadas iniciales comienzan en (-16,16), en la esquina superior izquierda, y terminan en (16,-16), en la esquina inferior derecha. Pero, como ya hemos visto, el tamaño del mundo se puede modificar pulsando sobre el mundo con el botón derecho del ratón.

Aunque las baldosas no se pueden mover, como agentes que son, poseen atributos y pueden ejecutar procedimientos.

```
ask patch 0 [ advertise sell restock ]
```

Las baldosas están íntimamente ligadas a las baldosas que las rodean, existen multitud de comandos para relacionar las baldosas, algunos ejemplos:

```
ask patches in-radius 5 [set pcolor red ]
ask turtles in-radius 5 [die]
ask turtles-here ;; invocamos algo sobre la tortuga que se encuentra en esta
baldosa
ask patches in-radius 5 ;;Elegimos Todas las baldosas a una distancia igual a 5
```

Código 3.2: Comandos básicos sobre baldosas NetLogo

Enlaces

El tercer tipo de agentes que vamos a ver son los enlaces. Los enlaces son la unidad necesaria para crear redes. Los enlaces son la conexión que se realiza entre tortugas. Siguiendo la terminología de la teoría de grafos, las tortugas serían el nodo y los enlaces, los arcos.
El comando que utilizamos para crear un enlace es:

```
create-link
```

Para crear una única conexión utilizamos el comando create-link with nombre-tortuga. Para crear más de una conexión, utilizaremos el comando create-link-with seguido de un agentset.

```
crt 5
ask turtles [ setxy ramdom-xcor ramdom-ycor create-links-with other turtles ]
```

En la siguiente figura podemos ver el resultado del comando anterior; se crean 5 tortugas y enlaces aleatorios entre las diferentes tortugas.

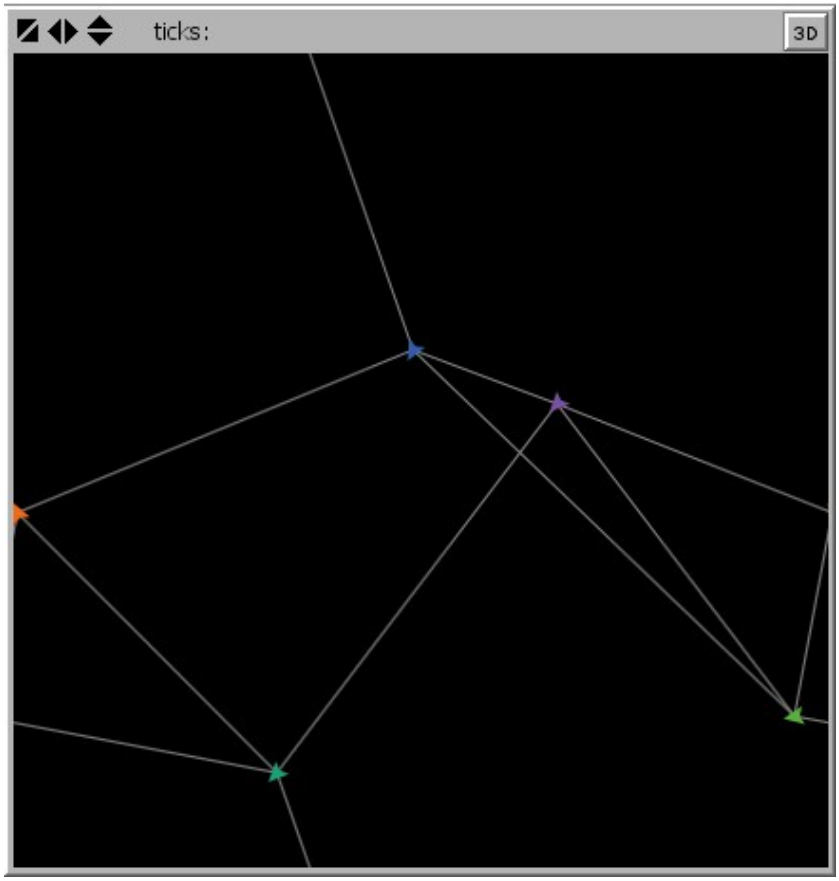


Figura 3.8: Enlaces aleatorios entre tortugas

Una vez creada la red, podemos pedir a los agentes que se posicionen en una formación determinada respecto a la red.

```
layout-radial
layout-spring
layout-tutte
layout-radial turtles links ( turtle 0 ) ;posiciona la tortuga 0 en el centro y a su alrededor las restantes
```

Código 3.3: Comandos enlaces NetLogo

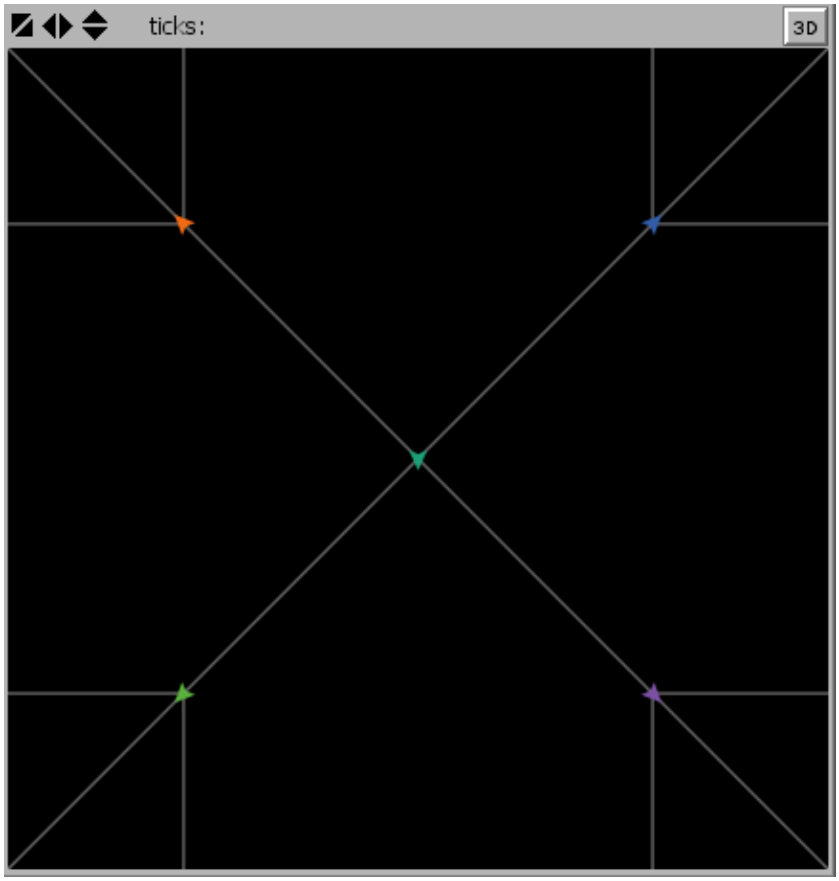


Figura 3.9: Posición del resto de tortugas

Por ahora, hemos visto que los enlaces no contienen dirección. Si queremos crear enlaces con dirección, deberemos utilizar las palabras FROM y TO con el comando create-links.

```
ask turtle 0 [ create-link-to turtle 2 ]
```

Los enlaces, al igual que el resto de agentes, tienen atributos. Los atributos por defecto son:

- end1
- end2
- color
- label
- label-color
- hidden?
- Breed
- thikness
- shapeness
- tie-mode

Para poder añadir nuevos atributos a los arcos, utilizaremos la instrucción:

```
links-own [ atributo ]
```

3.3.3 Agentsets, conjunto de agentes

Los conjuntos de agentes, agrupan a un subconjunto de agentes del mismo tipo (tortugas, baldosas o enlaces). Los agentes del conjunto no se ordenan de ninguna manera, es más, siempre están ordenados aleatoriamente.

Existen multitud de maneras de crear agentsets, a continuación mostramos unos pocos comandos que permiten crear conjuntos:

```
patches at-points [[1 0] [0 1] [-1 0] [0 -1]] ;; las cuatro baldosas al norte sur oeste este
neighbors4 ;; abreviatura para el comando anterior
turtles with [color = red] ;; todas las tortugas rojas
other turtles-here with [color = red] ;; las demás tortugas rojas de la baldosa
patches with [pxcor > 0] ;; baldosa con coordenada x positiva
turtles in-radius 6 ;; todas las tortugas a 3 pasos de distancia
[my-links] of turtle 0 ;; todos los enlaces conectados a la tortuga 0
```

Código 3.4: Comandos para crear conjuntos de agentes en NetLogo

Existen multitud de comandos para poder operar con los conjuntos de agentes. El más usado es el comando **ask**, que solicita a los agentes de un conjunto que realicen algo.

```
ask n-of 50 patches [ set pcolor green ] ;; 50 baldosas elegidas aleatorias cambian el color a verde
```

any? para saber si un agentset esta vacío.

```
if any? turtles with [color = red]
 [ show "por lo menos una tortuga es roja!" ]
```

nobody sirve para indicar que no se encontró ningún agente

```
set target one-of other turtles-here
if target != nobody
 [ ask target [ set color red ] ]
```

all? para saber si todos los agentes del conjunto tienen una propiedad.

```
if all? turtles [color = red]
 [ show "todas las tortugas son rojas!" ]
```

member? para saber si un agente pertenece a un conjunto de agentes.

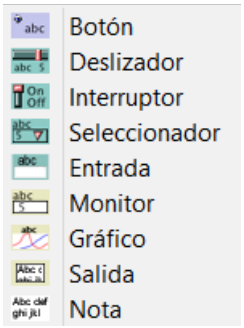
n-of <tamaño> <lista/agentset> crea un subconjunto aleatorio de n agentes del agentset

```
ask n-of 50 patches [ set pcolor green ] ;; 50 baldosas elegidas aleatoriamente cambian
el color a verde
```

one-of <lista/agentset> obtiene un agente aleatorio del conjunto de agentes.

```
ask one-of patches [ set pcolor green ] ;; un baldosa se vuelve verde
```

3.3.4 Controles de NetLogo



- **Botón**. Existen dos tipos de botones: el botón “por una vez” y el “botón permanente”. Cuando pulsamos un botón “por una vez”, ejecuta las instrucciones asociadas al botón una única vez, mientras que cuando pulsamos un botón permanente ejecuta las instrucciones una y otra vez hasta que se pulsa el botón otra vez. Para diferenciar los dos tipos de botones, en la esquina superior izquierda nos aparecerá un check para indicar el tipo.

- **Deslizador**. Como mencionaremos más adelante, los deslizadores son un tipo de variables globales accesibles por todos los agentes, permiten cambiar el valor de una variable sin tener la necesidad de recodificar cada vez.

- **Interruptor**. Como el deslizador, se trata de una variable global booleana accesible desde cualquier parte del entorno.
- **Seleccionador**. Permite asignar un valor a una variable global, restringido a un rango fijado de antemano; los valores pueden ser numéricos, de tipo texto o booleanos.
- **Entrada**. Permite introducir texto mediante el teclado.
- **Monitor**. Muestra por pantalla el valor de una función.
- **Salida**. Es un área con scroll que permite añadir un log al modelo.
- **Nota**. Permite añadir etiquetas de texto a la interfaz.
- **Gráfico**. Muestra datos que el modelo está generando.

3.4 Modelador de dinámica de sistemas de NetLogo

NetLogo dispone de un modelador de dinámica de sistemas que, de una forma gráfica, utilizando diagramas de Forrester, nos permite introducir las ecuaciones. Para acceder al modelador, debemos ir al menú Herramientas-> Modelador de dinámica de sistemas, o pulsando el Ctrl+Mayús+D.

La interfaz del modelador se compone de dos pestañas principales la pestaña “diagrama”, donde se puede componer el modelo de manera gráfica arrastrando los controles deseados; y la pestaña “código” donde se muestran las ecuaciones en NetLogo correspondientes al modelo gráfico.

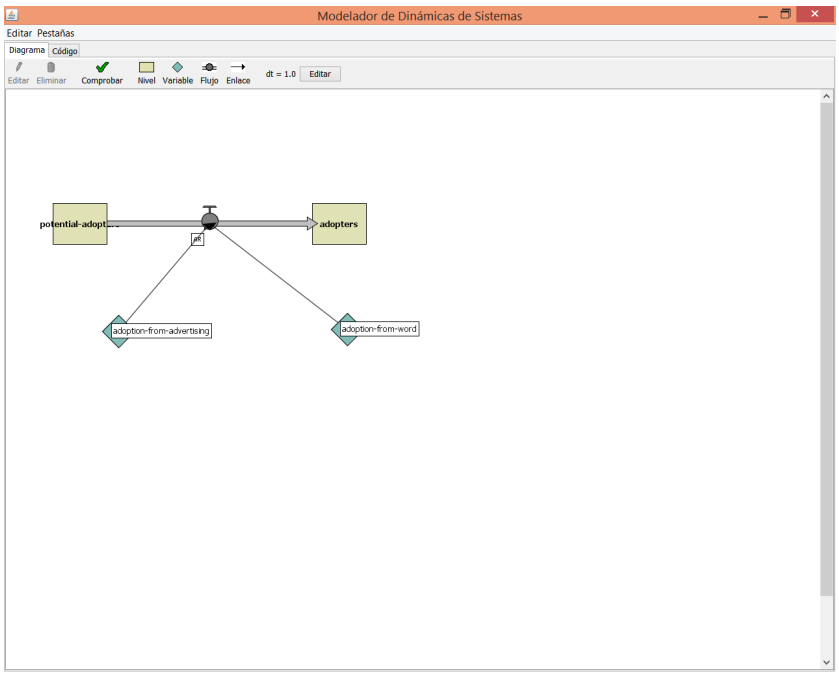
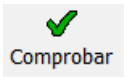


Figura 3.10: Modelador de Dinámica de sistemas

El modelador de dinámica de sistemas de NetLogo no tiene más que un ápice de las funcionalidades que tiene un entorno clásico de dinámica de sistemas como Stella o VensimPLE.



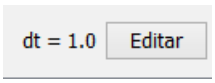
- El control comprobar valida las ecuaciones introducidas hasta el momento.

- El control nivel crea una nueva variable de nivel. Estas variables suelen representar un Stock como por ejemplo la población, volumen de agua acumulado etc.

- El control variable permite introducir una nueva variable auxiliar, puede ser una ecuación o una constante .

- El control flujo permite crear una nueva variable de flujo.

- El enlace proporciona un valor de una parte del diagrama a otra parte.



- Por último, en el extremo izquierdo de la barra de controles tenemos el *delta time*, abreviado como dt. Este parámetro gobierna todo el modelo y especifica la frecuencia en la que se realizarán los cálculos en el modelo. Por defecto tiene el valor de 1.0, lo cual hace que el modelo de Dinámica de Sistemas y el *tick* del modelo NetLogo estén sincronizados.

Cada vez que se edita un nivel, haciendo doble clic sobre una variable, o se pulsa el botón "check", NetLogo genera el código correspondiente al actual diagrama; a la vez intenta compilar este código. Si existe algún error en el modelo, la pestaña de código se pondrá en color rojo y se nos indicará en qué línea existe un error.

Una característica de la pestaña de código es que no se puede editar directamente el código.

La única forma de cambiar las ecuaciones es mediante el diagrama.

La transcripción del diagrama a ecuaciones se realiza de la siguiente manera (ver 2.2.4):

1. Las variables de nivel son inicializadas según el valor o expresión recogida en el campo valor inicial. Cada variable de nivel se actualiza en cada paso en función de sus flujos de entrada y salida.
2. Los flujos contendrán un procedimiento que contiene la expresión introducida en el campo expresión.
3. Las variables pueden ser variables globales o procedimientos. Si la expresión que se ha introducido es una constante, la variable generada será global. Si la expresión introducida es más compleja, se creará un procedimiento como en el caso de un flujo.
4. Por último, se añadirán automáticamente tres procedimientos: **system-dynamics-setup**, **system-dynamics-go**, *y* **system-dynamics-do-plot**. El primer procedimiento **system-dynamics-setup** se encarga de inicializar el modelo fijando el dt o paso, también resetea los ticks e inicializa las variables globales y niveles. **System-dynamics-go** se encarga de realizar el cálculo por cada paso, es decir, calcula el valor de los flujos y variables, y actualiza los niveles. El último procedimiento **system-dynamics-do-plot** se utiliza para imprimir resultados en la ventana principal de NetLogo.

3.5 El lenguaje NetLogo

3.5.1 Declaración y utilización de variables

Las variables se pueden declarar en diferentes sitios:

1. En la interfaz gráfica en los controles (nota, selector, botón, etc.).
2. Al principio del programa como variables globales, en la lista globals.

```
globals [ tasadecambio precio ]
```

3. Al principio del programa como perteneciente a un agente.

```
vendedores-own [ coste precio beneficio ]
```

4. Como variable local en un procedimiento o bloque de código.

```
let nombrevaraible
```

Los valores de una variable se establecen mediante la instrucción set;

```
set precio-inicial 200
set precios [199 200 300 ]
set nombre “Paco”
set precio-inicial  ( item 0 precios )
```

3.5.2 El tiempo en NetLogo

Los modelos de NetLogo transcurren en eventos de tiempo discreto, que avanza en pasos llamados ***ticks***. NetLogo dispone de un contador general, que memoriza el número de *ticks* transcurridos. Para obtener el tiempo transcurrido desde el inicio de la simulación, podemos ejecutar la instrucción **let este-periodo** ticks. Si deseamos resetear el contador, podemos hacerlo mediante la instrucción **reset-ticks**. Si deseamos aumentar su valor en una unidad, podemos ejecutar el comando **tick**.

El contador de *ticks* comienza con el valor 0 y se incrementa una unidad en cada paso de tiempo. También es posible que el incremento sea un número de coma flotante. Para que el contador avance una fracción, debemos utilizar el comando **tick-advance,** este comando toma como parámetro un valor numérico que indica cuánto será el incremento en el tiempo.

3.5.3 Estructura de un modelo

Cualquier modelo de NetLogo dispondrá como mínimo de:

- Una definición de los diferentes tipos de agentes tortugas **breed**, opcionalmente podrá

contener variables globales, variables de agente.

- Procesos de inicialización. Estos procesos, por convención, serán activados mediante el pulsado de un botón llamado **setup**.
- Definiciones de los procedimientos que describen la dinámica del sistema. Por convención, la simulación se inicializará pulsando un botón llamado **go**.

3.5.4 Procedimientos y funciones (commands y reporters)

Los procedimientos son un conjunto de instrucciones que se agrupan para poder ser llamadas varias veces. Los procedimientos se utilizan para cambiar las variables globales o las variables que son pasadas como argumentos.

Los **reporters** son funciones que devuelven un valor al final de su ejecución.

La declaración de un procedimiento comienza por la palabra **to** seguida del nombre de procedimiento y finaliza por la palabra **end;** los argumentos, si los tiene, se añaden detrás del nombre mediante una lista entre corchetes, a continuación se añaden las instrucciones del procedimiento.

```
to nombre-procedimiento [ argumento1 argumento2 ]
        instrucción1
        instrucción2
end
```

Código 3.5: Reporters de NetLogo (Procedimientos)

Para poder llamar a un procedimiento, solo hace falta escribir el nombre del procedimiento seguido de sus argumentos, si los tiene.

NetLogo permite la llamada a procedimiento a través del control botón. Si editamos un botón en el campo de instrucciones, solo debemos poner el nombre del procedimiento que queremos llamar. Observamos que en esta misma ventana podemos indicar que las instrucciones o el procedimiento se ejecuten continuamente (cada *tick*).

La declaración de funciones se realiza con la palabra ***to-report,*** igual que los procedimientos,

pero entre las instrucciones una función debe contener la instrucción ***report,*** que devuelve el valor resultado de la ejecución de la función.

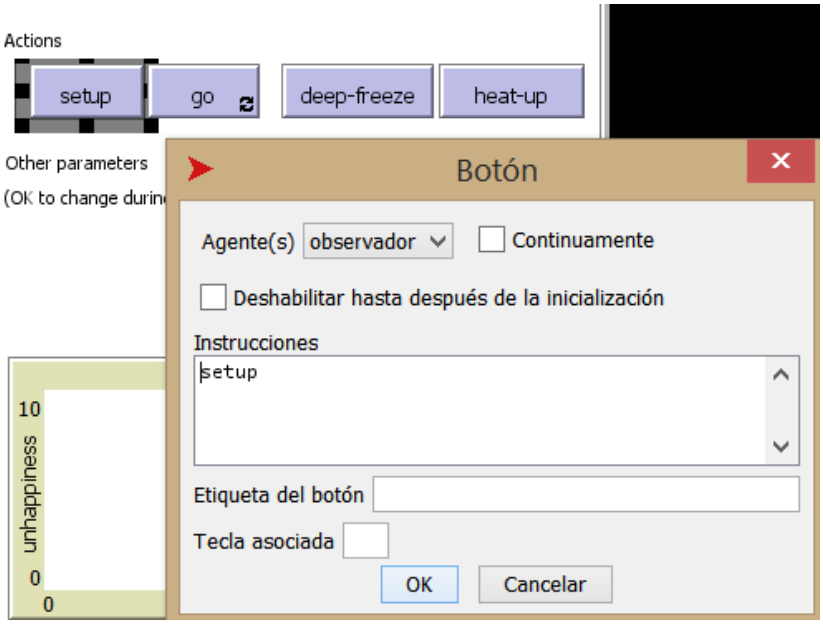


Figura 3.11: Llamada a procedimientos desde botones

3.5.5 Bloques de control

Bucles [1]

La instrucción ***while*** permite repetir un conjunto de instrucciones mientras la condición sea verdadera

```
while [ condicion ] [ instrucciones ]

while [ any other turtles-here]

[ fd 1 ] ;; las tortugas se mueven hasta que encuentran una
baldosa donde no  hay otra tortuga.
```

Código 3.6: Bucles en NetLogo

La primitiva ***repeat*** permite repetir un número n de veces un conjunto de instrucciones

```
repeat n [ instrucciones ]

let %ct 0 repeat 50 [show %ct set %ct (%ct + 1) stop ]
```

La primitiva ***loop*** permite repetir un número de instrucciones indefinidamente. Obviamente, es una construcción peligrosa, aunque sí contiene la instrucción ***stop*.**

```
loop [ comandos ]
```

Condiciones

If: el operador **If** de NetLogo solo puede trabajar con variables booleanas.

```
If semáforo = rojo [ para-coche ]
```

Ifelse: el operador if de NetLogo solo permite validar una condición y ejecutar una acción, pero en numerosas ocasiones se debe evaluar una condición y la contraria y ejecutar una acción apropiada para cada condición.

El operador Ifelse permite realizar esto de la siguiente manera:

```
Ifelse tiempo = lluvioso [ ir-al-cine ] [ playa ]
```

3.6 Gráficas en NetLogo

Como vimos, NetLogo dispone de un componente para mostrar gráficos. Cuando creamos una gráfica, automáticamente nos aparecerá el cuadro de diálogo del componente. La mayoría de los campos se autoexplican: Nombre, Etiquetas.

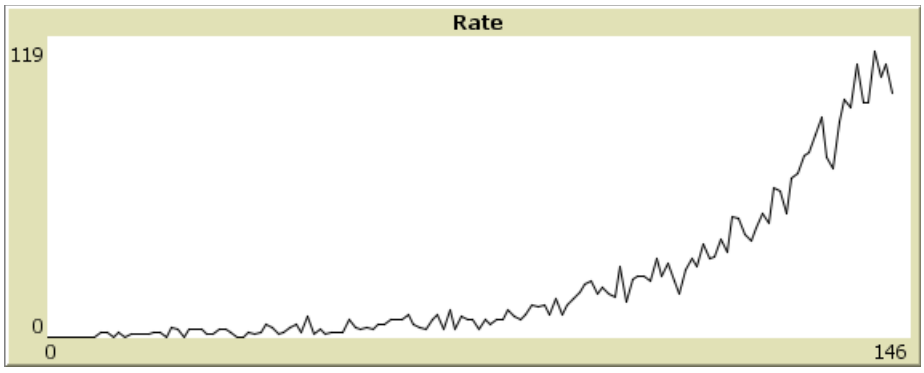


Figura 3.12: Componente Gráfico de NetLogo

Los campos importantes son los que aparecen en la tabla Trazos del Gráfico:, por un lado, el "Nombre del Trazo" (*Pen name*) y, por otro, "Instrucciones de Actualización de Trazos". Cada trazo corresponde a una serie de datos. Su valor se obtiene ejecutando las "Instrucciones de Actualización de Trazos" cada tick. Como podemos ver en la Figura 3.13, se trata de comandos de NetLogo que devuelven el valor numérico que corresponde al valor alcanzado por la variable que queremos representar.

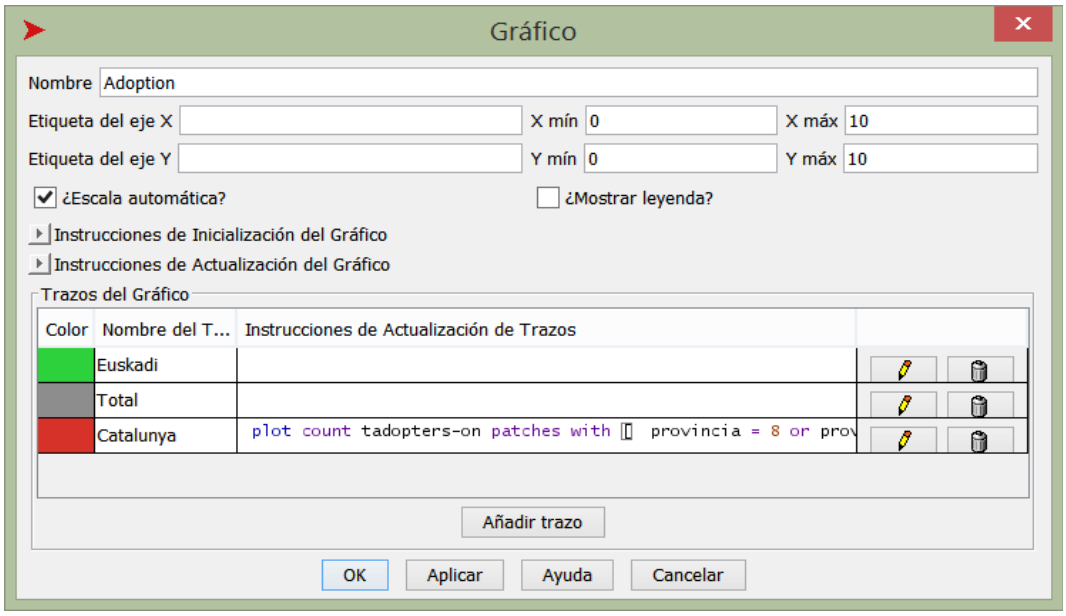


Figura 3.13: Dialogo Gráfica

NetLogo provee de comandos para poder actualizar las gráficas directamente en el código, sin tener que introducir las instrucciones de actualización de los trazos.

```
set-current-plot "Adoption"
 set-current-plot-pen "Total"
  plot count tadopters
```

Código 3.1: Código actualización de gráficas en NetLogo

Como podemos ver en el código, la actualización se realiza en tres pasos:

1. Seleccionamos el gráfico mediante el comando `set-current-plot`.
2. Elegimos el trazo que queremos actualizar mediante el comando `set-current-plot-pen`.
3. Mediante el comando `plot`, establecemos las instrucciones de actualización de la gráfica.

Por último, nos gustaría señalar que, cuando se realiza un modelo de Dinámica de Sistemas

en NetLogo, el modelador de Dinámica de Sistemas genera automáticamente un bloque de código (reporter), llamado `system-dynamics-do-plot`, donde, para cada variable de nivel, se actualiza la gráfica si existe.

```
to system-dynamics-do-plot
 if plot-pen-exists? "Marketing_performance" [
  set-current-plot-pen "Marketing_performance"
  plotxy ticks Marketing_performance
 ]
end
```

Código 3.7: Bloque de código autogenerado NetLogo

3.7 Ejemplo difusión de una epidemia

Una vez vistas las características del lenguaje y los elementos que componen NetLogo, vamos a desarrollar un caso práctico, un modelo de difusión epidémica en NetLogo.

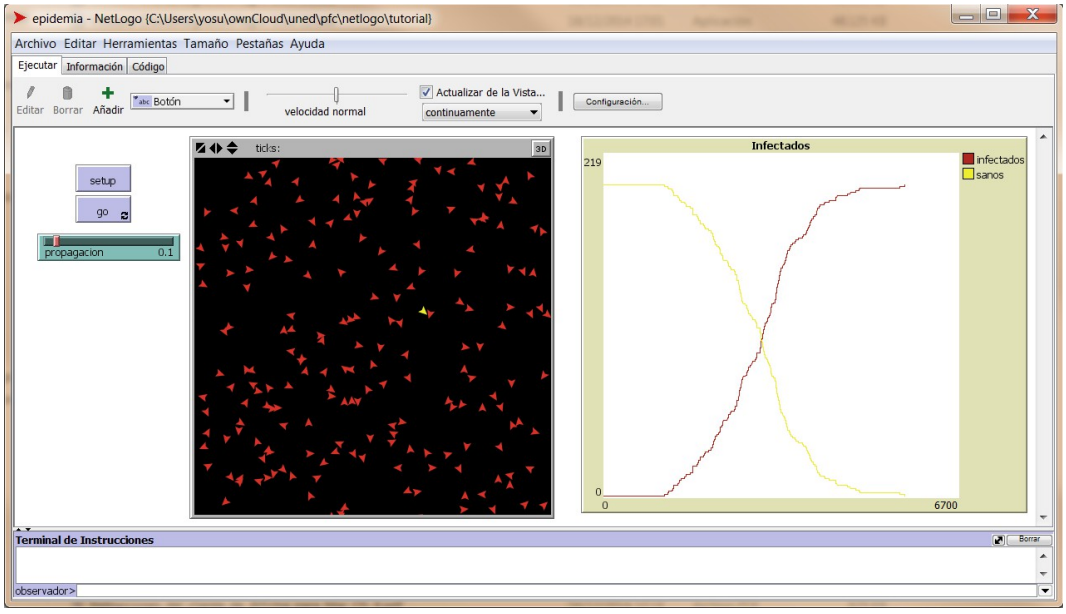


Figura 3.14: Ejemplo difusión de una epidemia

Supondremos que tenemos una población de individuos sanos (amarillos) y un infectado (rojo). Cuando el infectado entre en contacto con una persona, la podrá infectar o no. El proceso de

infección se regirá por una función de probabilidad marcada por el parámetro propagación (deslizador) que determinará la velocidad del contagio.

La interfaz contará con un deslizador para configurar el parámetro propagación y una gráfica donde veremos la evolución de la infección, aparte de los botones **setup** y **go,** que sirven para inicializar y comenzar la simulación.

Creación de la Interfaz

El deslizador lo definiremos de la siguiente manera:

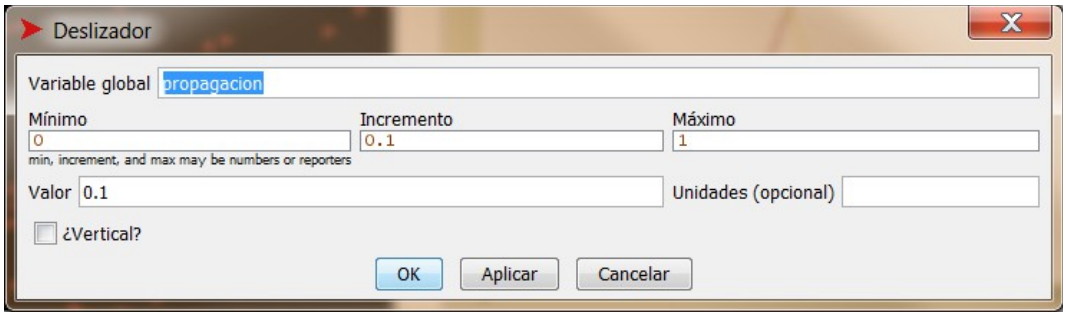


Figura 3.15: Deslizador modelo epidémico

El gráfico se definirá de la siguiente manera:

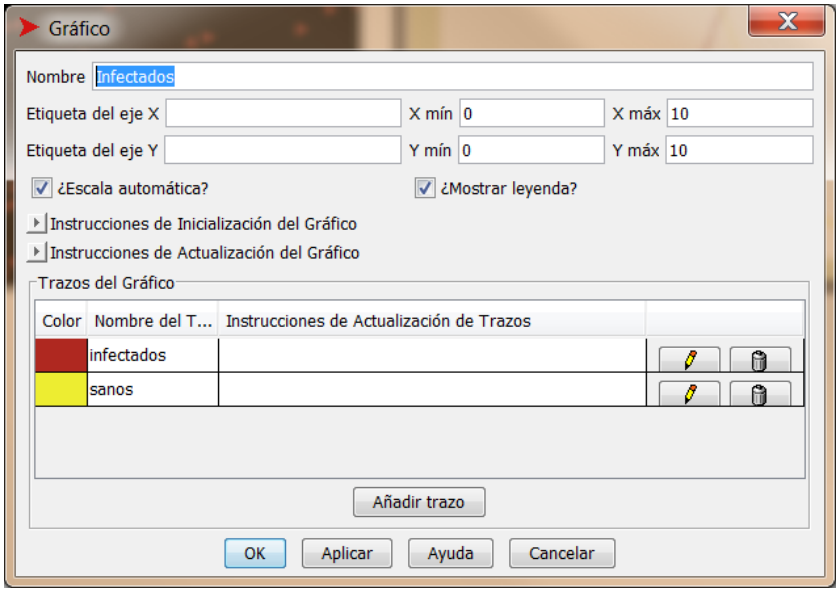


Figura 3.16: Gráfico infectados

Código del modelo

Como en todo modelo NetLogo, lo primero que haremos es crear el código de inicialización asociado al botón **setup**.

```
to setup
 ca                                ;reset
 crt 200                           ;creamos 200 tortugas
 ask turtles [
  fd random 100                    ;Las distribuimos aleatoriamente
  set color yellow                 ;Les ponemos el color amarillo (no infectadas)
  set heading random 360           ;Las orientamos aleatoriamente
  if who = 1 [ set color red]      ;Elegimos una tortuga y la infectamos
  ]
end
```

Código 3.2: Código botón setup

El botón **go,** que como sabemos, se ejecuta cada paso, realizará tres tareas:

1. Mover las tortugas
2. Infectarlas a razón del parámetro propagación
3. Actualizar el gráfico

Cada tarea se implementará en un procedimiento

```
to go
 if ( count turtles with [ color = yellow] > 1 ) ;mientras queden tortugas sanas
 [
  mueve
  infecta
  pinta-grafica
 ]
end

to mueve
 ask turtles [
  right random 40
  left random 40
  forward 1

  ]
end

to infecta
 ask turtles with [color = red ]
  [ask other turtles-here with [color = yellow ]
   [if (random 100) < propagacion
    [
     set color red
    ]

    ]

   ]
end

to pinta-grafica
 set-current-plot "Infectados"      ;Indicamos con que gráfica queremos trabajar
 set-current-plot-pen "infectados"    ;Indicamos que trazo queremos actualizar
 plot count turtles with [ color = red] ;Indicamos el valor del trazo
 set-current-plot-pen "sanos"
 plot count turtles with [ color = yellow ]
end
```

Código 3.3: Código botón go

3.8 Integración con Herramientas GIS

En este apartado vamos a estudiar la extensión GIS de NetLogo. Esta extensión permite importar información proveniente de herramientas GIS en NetLogo y trabajar con ella.

Para facilitar la comprensión de los conceptos clave necesarios para entender el proceso de

importación, vamos a realizar en el siguiente apartado una breve introducción a una herramienta GIS de código abierto (Qgis s. f.), herramienta que hemos utilizado en el proyecto para confeccionar capas para el modelo de Difusión de la Innovación.

3.8.1 QGIS

"Quantum GIS" es un Sistema de Información Geográfica (SIG) de código libre multiplataforma. Permite manejar *rasters* y *shapes* a través de la biblioteca GDAL/OGR[1] – Geospatial Data Abstraction Library, así como múltiples bases de datos.

Para poder entender cuál es la función de una herramienta como ésta, tenemos que explicar necesariamente qué son los "*rasters*" y qué son los "*shapes*".

Un fichero *raster* es una matriz de píxeles o celdas organizadas en filas y columnas en las que cada celda almacena datos sobre esa región; normalmente, suelen ser fotos aéreas, de satélite, o mapas escaneados. Un *raster* suele estar formado por:

1. Una imagen, fotografía o imágenes escaneadas de un mapa.
2. Datos continuos, como la elevación o datos espectrales.
3. Datos discretos que representan, por ejemplo, el destino al que se dedica el suelo, si es suelo rústico o urbano, etc.

Un fichero shape es un fichero que contiene información de tipo vectorial (puntos, líneas, polígonos) asociadas a su ubicación geométrica. En este tipo de ficheros también se suelen incluir bases de datos que contienen información sobre atributos adicionales que se pueden asociar a los *shapes*.

Gracias a la librería OGR, se permite trabajar con los siguientes formatos de *shapes* ESRI, MapInfo Microstation Autocad DXF, PostGIS, SpatiaLitem Oracle Spatial, MSSQL Spatial y otros muchos.

En este proyecto se han confeccionado una serie de *shapes* para ubicar los agentes, por lo que a continuación explicaremos los procedimientos que hemos utilizado en la confección de dichos *shapes* en formato tutorial, para poder tener una visión de cómo y para qué se utilizan este tipo de

1 OGR forma parte del árbol de desarrollo de GDAL y provee funciones sencillas para rasters.

herramientas.

3.8.2 Trabajando con Shapes

Importar Shape

Una vez arrancado QGIS, podemos importar un *shape* pulsando el botón  "Añadir capa vectorial".

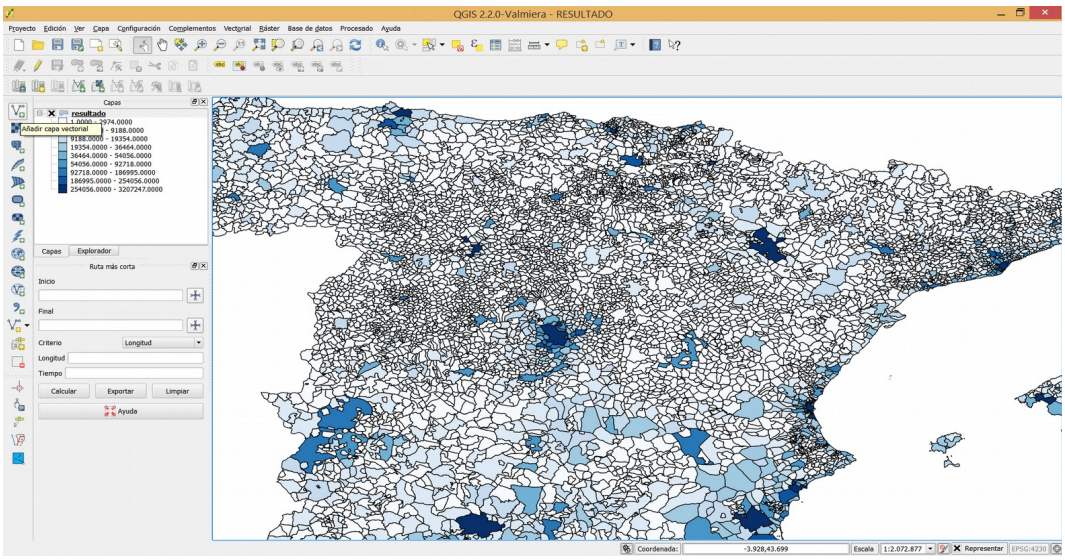


Figura 3.17: Interfaz principal del QGIS

Una vez pulsado debemos indicar la ruta donde se encuentra el fichero shp.

Figura 3.18: Ventana de dialogo Añadir capa vectorial QGIS

Una vez cargado el fichero, si dispone de información vectorial, ésta se dibujará en el marco principal de la interfaz, como se muestra en la Figura 3.17. Además, en la ventana "capas" se añadirá una nueva capa con el *shape* que hemos importado. Como hemos indicado en la introducción, un

shape no solo está compuesto de puntos, líneas o polígonos, también contiene atributos que son bases de datos asociados a la información vectorial. Si deseamos ver/manipular esta información, debemos elegir la capa que se va a visualizar en la ventana de capas, esquina superior izquierda en la Figura 3.17. Después pulsaremos el botón derecho del ratón, donde aparecerá un menú flotante en el que tendremos que seleccionar la opción “propiedades”.

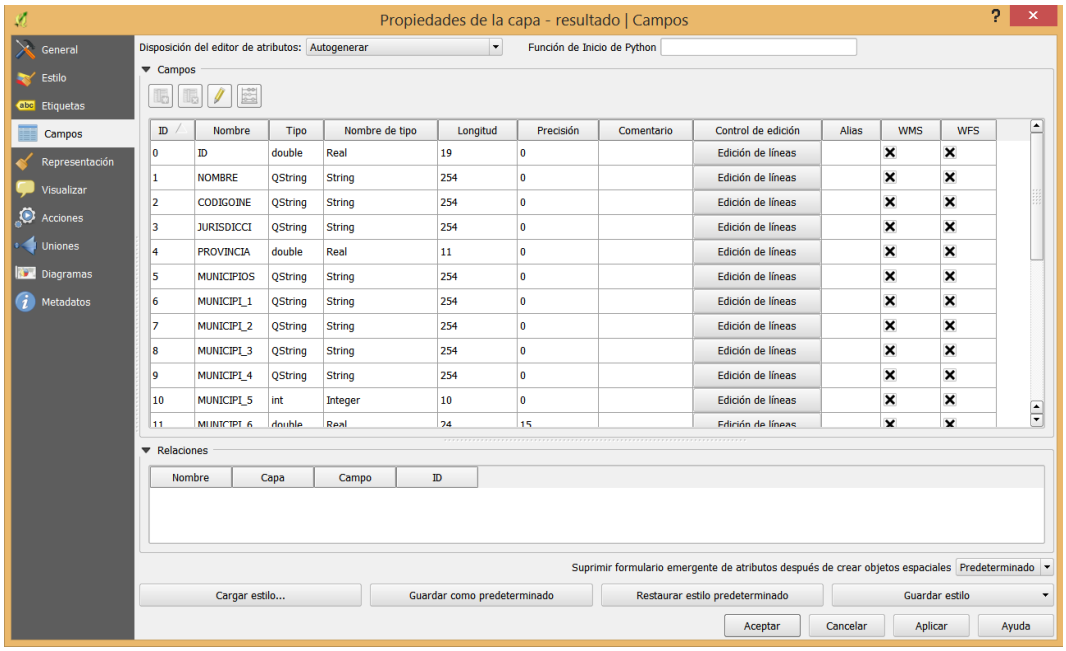
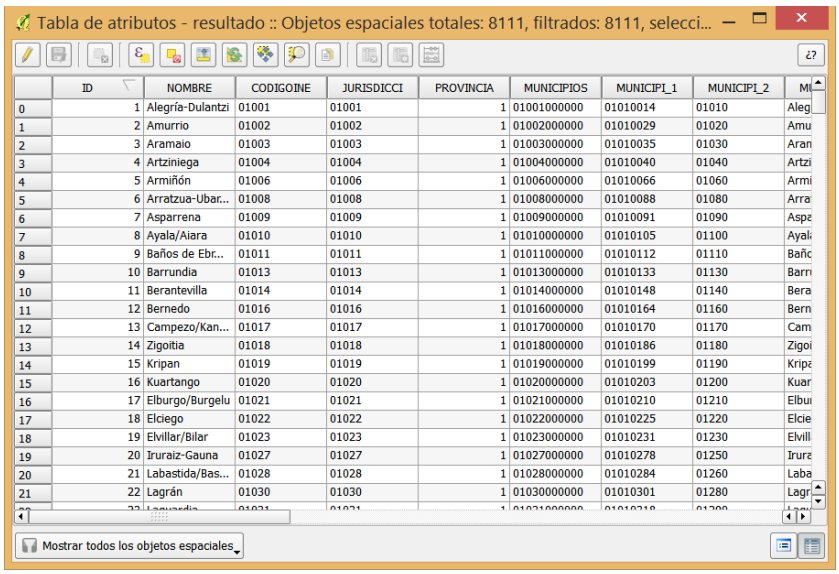


Figura 3.19: Ventana propiedades de la capa QGIS

La ventana de propiedades tiene muchas opciones, pero la que nos interesa es “Campos”, como se muestra en la Figura 3.19. En esta vista percivimos que se muestra la estructura de la base de datos asociada al *shape*: los campos, tipos de datos y demás atributos; desde esta ventana también podremos modificar la estructura añadiendo, eliminando o modificando los campos de la tabla.

Si lo que deseamos es ver los datos que contiene la tabla y no su estructura, tendremos que elegir el *shape* en la ventaba de capas y pulsar el botón  “Abrir tabla de atributos” de la interfaz principal, parte superior derecha en la Figura 3.17.



	ID	NOMBRE	CODIGOINE	JURISDICCI	PROVINCIA	MUNICIPIOS	MUNICIPI_1	MUNICIPI_2	MU
0	1	Alegría-Dulantzi	01001	01001	1	01001000000	01010014	01010	Aleg
1	2	Amurrio	01002	01002	1	01002000000	01010029	01020	Amu
2	3	Aramaio	01003	01003	1	01003000000	01010035	01030	Aran
3	4	Artziniega	01004	01004	1	01004000000	01010040	01040	Artzi
4	5	Armiñón	01006	01006	1	01006000000	01010066	01060	Armi
5	6	Arratzua-Ubar...	01008	01008	1	01008000000	01010088	01080	Arra
6	7	Asparrena	01009	01009	1	01009000000	01010091	01090	Aspa
7	8	Ayala/Aiara	01010	01010	1	01010000000	01010105	01100	Ayal
8	9	Baños de Ebr...	01011	01011	1	01011000000	01010112	01110	Bañ
9	10	Barrundia	01013	01013	1	01013000000	01010133	01130	Barr
10	11	Berantevilla	01014	01014	1	01014000000	01010148	01140	Bera
11	12	Bernedo	01016	01016	1	01016000000	01010164	01160	Bern
12	13	Campezo/Kan...	01017	01017	1	01017000000	01010170	01170	Cam
13	14	Zigoitia	01018	01018	1	01018000000	01010186	01180	Zigoi
14	15	Kripan	01019	01019	1	01019000000	01010199	01190	Kripa
15	16	Kuartango	01020	01020	1	01020000000	01010203	01200	Kuar
16	17	Elburgo/Burgelu	01021	01021	1	01021000000	01010210	01210	Elbu
17	18	Elciego	01022	01022	1	01022000000	01010225	01220	Elcie
18	19	Elvillar/Bilar	01023	01023	1	01023000000	01010231	01230	Elvill
19	20	Iruraiz-Gauna	01027	01027	1	01027000000	01010278	01250	Irura
20	21	Labastida/Bas...	01028	01028	1	01028000000	01010284	01260	Laba
21	22	Lagrán	01030	01030	1	01030000000	01010301	01280	Lagr

Figura 3.20: Tabla de atributos de la capa QGIS

El trabajo con los datos no se realiza como en un gestor de base de datos, mediante un lenguaje de consultas como SQL (por sus siglas en inglés *Structured Query Language*). A modo ilustrativo y como explicación de una tarea realizada para la confección del *shape* utilizado en este proyecto, vamos a explicar cómo se pueden borrar datos nulos de una columna y cómo añadir una nueva columna donde se añade la superficie del polígono asociado.

Eliminación de nulos de un campo

1. Elegir los campos con null.
    i. Seleccionamos el icono  y rellenamos la expresión de búsqueda.

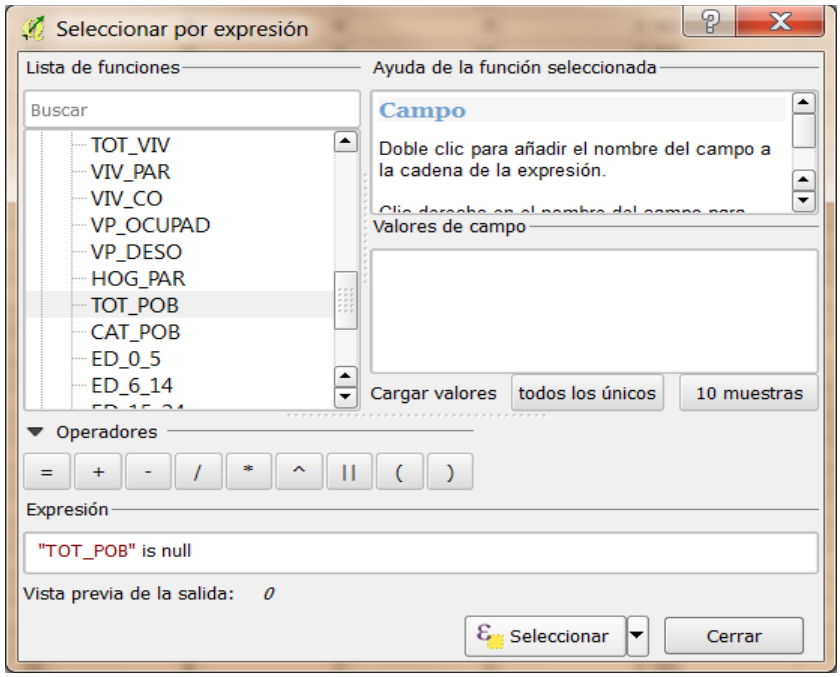


Figura 3.21: Ventana Seleccionar por expresión

Los campos nulos aparecen seleccionados en la tabla de atributos.

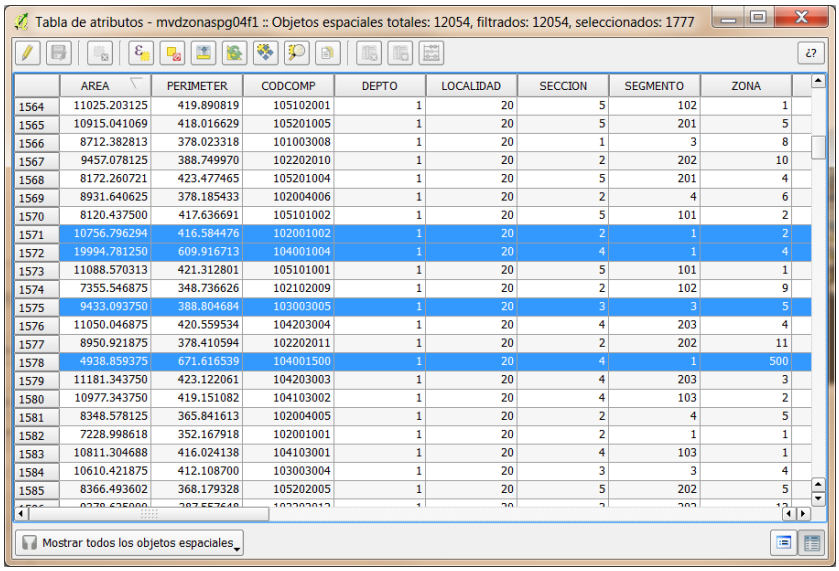


Figura 3.22: Campos seleccionados tras aplicar una selección.

2. Sustituimos los campos nulos por el valor deseado, en este caso el valor 0.
   i. Activamos el botón de edición 
   ii. Pulsamos en el botón calculadora de campos  (Ábaco).
   iii. Señalamos que queremos cambiar un campo existente, solo para los campos

seleccionados.

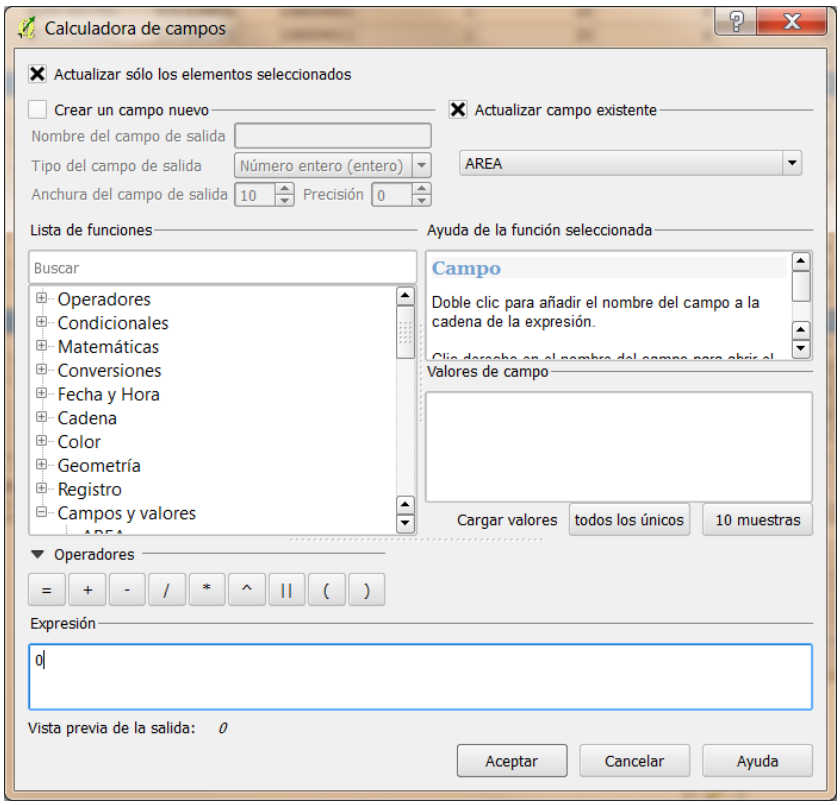


Figura 3.23: Ventana Calculadora de campos

Añadiendo un campo que mida el área de la superficie de un polígono

1. Pulsamos en calculadora de campos.
2. Pulsamos editar.
3. Y añadimos un campo que recoja la siguiente expresión: $area.

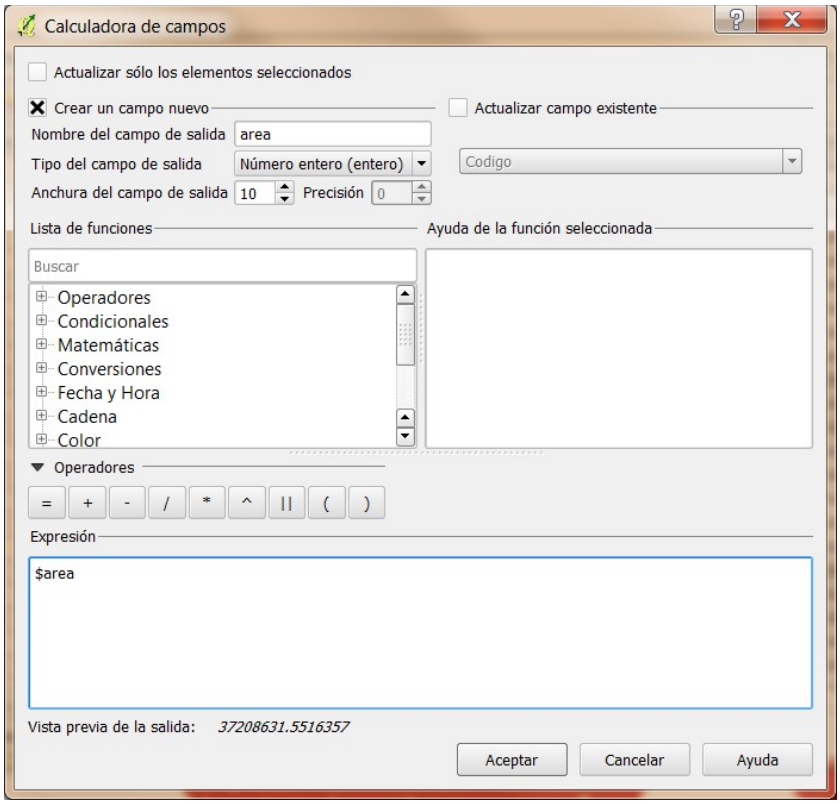


Figura 3.24: Ventana de Calculadora de campos con expresión

Uniendo campos importados desde Excel

Los ficheros xls (Excel) pueden ser añadidos a un proyecto QGIS como otra capa vectorial más.

1-Al añadirlos nos aparecerá una capa Consulta asociada al Excel en la ventana de capas.

2-Pulsamos sobre la capa.shp con la que queremos combinar los datos del Excel.

3- Pulsamos en propiedades y asociamos qué columnas queremos unir.

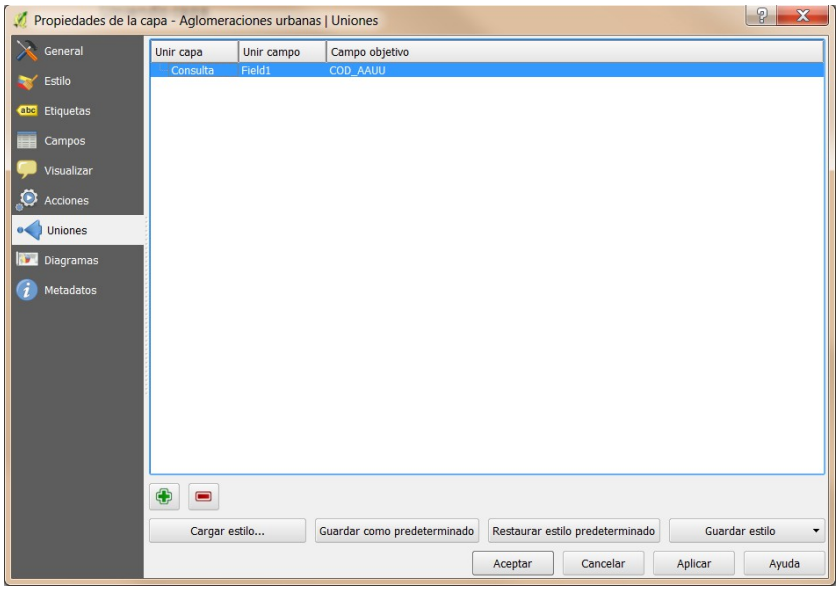


Figura 3.25: Ventana de unión de campos

3.8.3 Extensión GIS para NetLogo

La extensión GIS de NetLogo (Russell & Edelson 2012) nos permite importar vectores y *rasters* en ficheros *shape* (.shp) con formato ESRI en nuestro proyecto. Para realizar la importación de un *shape* debemos seguir los siguientes pasos:

1. Cargar la extensión GIS en NetLogo.
2. Definir una transformación entre el espacio GIS y el espacio NetLogo.
3. Cargar los *datasets* o ficheros .shp que utilizaremos en nuestro proyecto.
4. Aplicar una cobertura.
5. Mostrar el *shape* en NetLogo.

Vamos a detallar los pasos que hemos seguido en nuestro proyecto y explicar los conceptos necesarios para realizar correctamente la importación.

El primer paso es cargar la extensión GIS en NetLogo;

```
extensions [ gis ]
```

Después tenemos que definir una transformación entre el espacio GIS y el espacio NetLogo.

```
gis:load-coordinate-system "resultado.prj"
```

El fichero resultado.prj, generado en QGIS al confeccionar el *shape* para el proyecto, solamente contiene una línea que indica el sistema geodésico de referencia del *shape* que vamos a importar.

```
GEOGCS["ED50",DATUM["D_European_1950",SPHEROID["International_1924",6378388,297]],PRIME
M["Greenwich",0],UNIT["Degree",0.0174532925199433295]]
```

El siguiente paso consiste en cargar los *datasets* mediante la primitiva gis:load-dataset. Esta primitiva carga un vector o un *shape* y se lo asigna a una variable.

```
set distritos gis:load-dataset "resultado.shp"
```

Una vez cargado el dataset, lo podemos visualizar en NetLogo mediante las siguientes primitivas:

```
gis:set-drawing-color white
gis:draw distritos 1
```

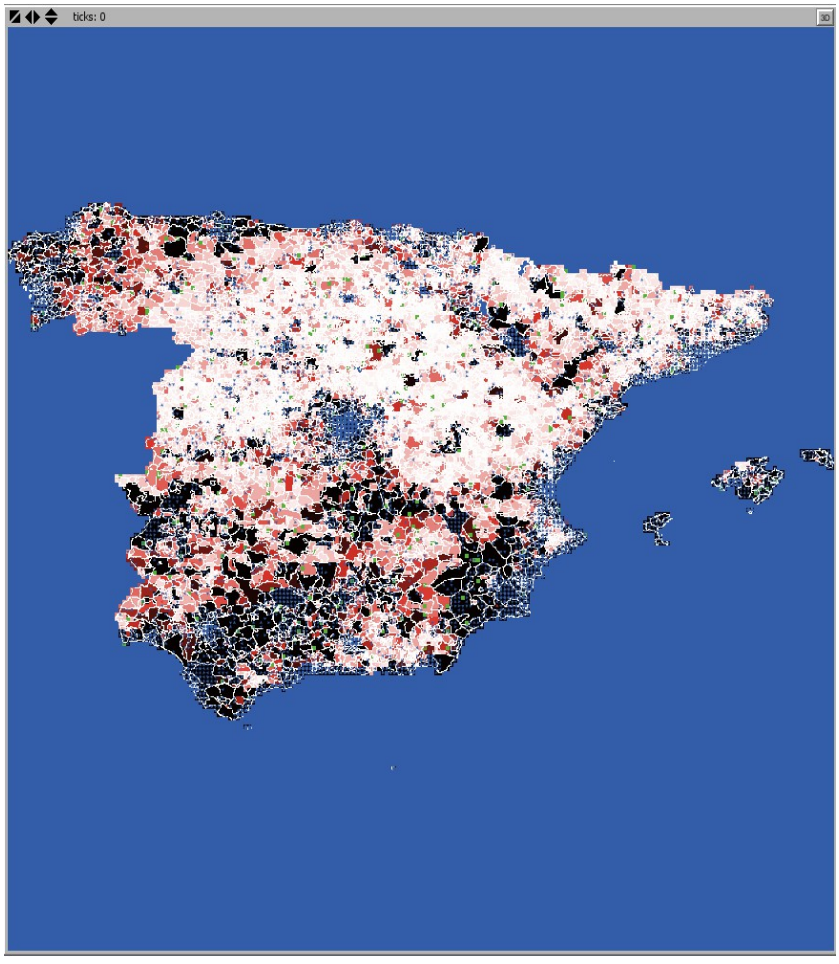


Figura 3.26: Shape visualizado en NetLogo

Por último, el paso más importante es aplicar una cobertura, que consiste en asociar atributos del *shape* a su correspondiente atributo de la baldosa en NetLogo. Para cada baldosa, se realiza una intersección con los atributos del *shape;* si el atributo es un campo de texto, asignará el más repetido; si el atributo es numérico, calculará una media ponderada para la baldosa. La primitiva que utilizaremos es **gis:apply-coverage** *VectorDataset property-name patch-variable,* que, como se puede observar, contiene tres argumentos: el *shape*, el nombre del atributo del *shape* y, por último, la variable de la baldosa a la que se asociará la información.

```
gis:apply-coverage distritos "MUNICIPI_5" poblacion
gis:apply-coverage distritos "MUNICIPI_6" area
```

En el ejemplo anterior asociamos el atributo MUNICIPI_5 del *shape* (la población) al atributo de la baldosa población. A partir de este momento, podemos trabajar en NetLogo con los nuevos atributos asociados a las baldosas. En la siguiente figura mostramos los atributos importados de área y densidad para una de las baldosas correspondientes al término municipal de El Burgo de Osma.

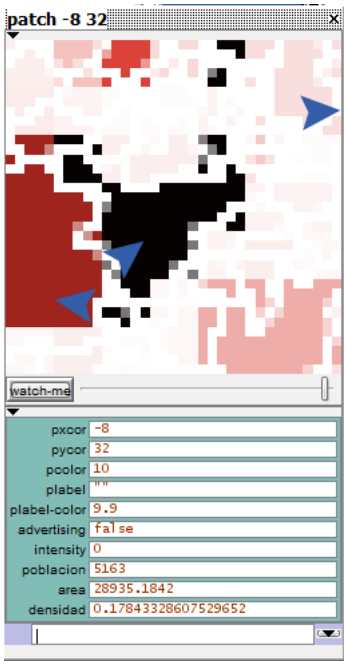


Figura 3.27: Ventana de baldosa con atributos importados desde QGIS

3.9 Cómo seguir estudiando NetLogo

El usuario que quiera profundizar en el aprendizaje de NetLogo cuenta con innumerables tutoriales y documentación en Internet.

Como veíamos en la comparativa de herramientas del Apartado 2.5, la documentación oficial de NetLogo (Wilensky 1999) así como sus preguntas frecuentes faq (Wilensky 2014b), son el punto de partida para continuar el aprendizaje.

La documentación oficial solo está disponible en inglés, pero contamos con un excelente tutorial en castellano realizado por (J. Poza García s. f.)

El usuario más avanzado puede analizar modelos en la biblioteca oficial de NetLogo incluida en la propia aplicación (Wilensky 2014c). En caso de querer analizar más modelos o contribuir con la comunidad, NetLogo cuenta con una biblioteca *on line* con más modelos (Wilensky n.d.).

Para finalizar, el usuario que quiera trabajar con información GIS en NetLogo puede encontrar toda la documentación referente al *plugin* GIS en la documentación oficial del *plugin* (Russell & Edelson 2012) .

3.10 Conclusiones

El entorno de programación NetLogo es sobrio y sencillo. Su lenguaje, no orientado a objetos, basado en el lenguaje Logo, facilita el desarrollo a programadores poco experimentados. La sencillez del lenguaje no impide que se puedan realizar modelos complejos, siendo una herramienta ideal para el prototipado rápido de modelos.

Hemos estudiado la posibilidad de integrar información proveniente de aplicaciones GIS en NetLogo. Para ello, hemos realizado una pequeña introducción a las herramientas GIS y hemos presentando el entorno QGIS, entorno cuyo estudio excede el ámbito de este proyecto, ya que, como hemos visto al analizar las posibilidades de la herramienta (solo hemos visto la punta del iceberg), los sistemas de información geográfica, son una disciplina que se sustenta en tecnologías como los sistemas de bases de datos, los sistemas de coordenadas cartográficas, gráficos vectoriales, etc .

4 Modelo de la difusión de la Innovación

4.1 Introducción

Entendemos por innovación la aplicación original y portadora de progreso de un descubrimiento, de una invención o, simplemente, de una idea.

La difusión, por su parte, es el proceso que recoge la expansión de una innovación desde el primer seguidor hasta su aceptación masiva por el mercado.

La difusión de la innovación es un modelo que pretende explicar cómo, por qué y a qué velocidad se difunden las nuevas tecnologías o ideas entre las diferentes culturas. Esta teoría se popularizó gracias al libro “Diffusion of Innovations” (Rogers 1962) escrito por el sociólogo y estadístico americano Everett Rogers. Si bien su teoría se ha cuestionado con la aparición de ejemplos de innovación radicales, como los teléfonos móviles, sigue siendo un modelo válido y ampliamente utilizado.

En este capítulo vamos a estudiar dos modelos de difusión clásicos: el modelo de Rogers y el de Bass para después analizar y comprender el modelo de Ahmadian, el cual, como casi todos los modelos de difusión, se inspiran en los anteriores. De entre los diferentes modelos de difusión clásicos : (Fourt & Woodlock 1960) , (Mansfield 1961) , (Rogers 1962) ,(Bass 1969) hemos elegido estos dos modelos porque, además de ser los más citados, parecen haber sido concebidos para ser desarrollados empleando la metodología multiagente estudiada en el Capítulo 2.3. El modelo híbrido de difusión que plantearemos en el siguiente capítulo está basado en los modelos estudiados en este capítulo, por lo que sugerimos leer este capítulo teniendo siempre en mente el posible uso de estas teorías en la confección de un modelo híbrido.

4.2 La teoría de la difusión de la innovación (Rogers 1962)

Rogers estudió cuáles eran las condiciones que contribuían a la adopción de una innovación, particularmente entre los profesores. Su teoría detecta cinco características de una innovación para cinco tipos de usuarios y en cinco fases, que se resumen en la siguiente tabla.

Características de la innovación	Tipos de usuarios	Fases de adopción
Ventaja relativa	Innovadores	Conocimiento
Compatibilidad con los valores del grupo de pertenencia	Primeros siguientes	Persuasión
	Mayoría precoz	Decisión
Complejidad	Mayoría tardía	Implantación
Posibilidad de evaluación	Rezagados	Confirmación
Visibilidad		

Tabla 4.1: Teoría de la adopción Rogers

Según Rogers, en un entorno social no todos los individuos toman las decisiones sincronizadamente, sino que cada individuo, según sus características, va adoptando la innovación a su ritmo. Rogers categorizó los usuarios a la hora de adoptar una innovación de la siguiente manera:

Categoría	Descripción
Innovadores	Tienen un interés especial por las nuevas ideas. Son ellos los que presentan una innovación al resto de los miembros de la sociedad.
Primeros seguidores	Más integrados en el sistema social que los innovadores, son líderes de opinión que son consultados por otros miembros acerca de la idoneidad de la innovación.
Mayoría precoz	Representan un tercio de la sociedad. Adoptan una innovación justo antes de la media de la sociedad. Sirven de correa de transmisión entre los miembros que adoptan la innovación relativamente temprano y aquellos que posponen la decisión.
Mayoría tardía	Representa también otro tercio de la sociedad. Adoptan una innovación justo después de la media de la sociedad. Consienten la adopción cuando las incertidumbres referentes a la innovación han sido disipadas por la comunidad.
Rezagados	Resistentes a los cambios y anclados en el pasado, quieren asegurarse el éxito de la innovación antes de adoptarla.

Tabla 4.2: Descripción de las categorías de seguidores según Rogers

En un sistema social las decisiones no se toman colectivamente, sino que cada individuo adopta su propia decisión de innovar. Los individuos van pasando por diferentes fases hasta poder confirmar la adopción de la innovación; Rogers categorizó estas fases de la siguiente manera:

Fase	Proceso de adopción
Conocimiento	El individuo investiga la innovación y adquiere nociones básicas sobre su funcionamiento.
Persuasión	El individuo comienza a interesarse en adoptar la innovación.
Decisión	El individuo realiza una serie de acciones con el objeto de decidir si aceptar o rechazar la innovación.
Implementación	El individuo utiliza diariamente la innovación y evalúa sus ventajas.
Confirmación	El individuo intenta obtener información para reforzar su decisión.

Tabla 4.3: Fases de adopción de la innovación Rogers

El proceso de adopción a lo largo del tiempo sigue una gráfica de distribución normal o campana de Gauss, y el número de adoptadores adopta una forma de S.

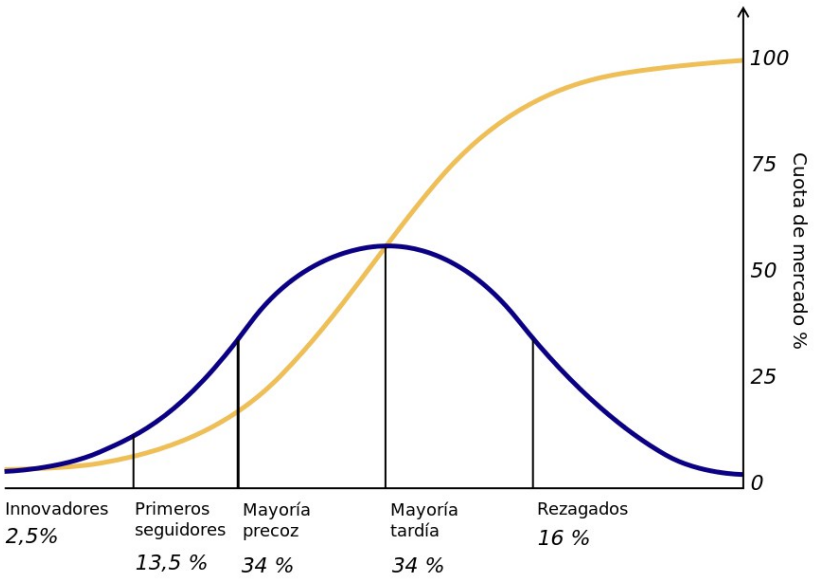


Figura 4.1: Curva de adopción (fuente Wikimedia)

Desde la perspectiva de la Dinámica de Sistemas, la difusión de un nuevo producto puede verse como un fenómeno epidemiológico donde los usuarios de la nueva tecnología contagian a los clientes potenciales mediante el EBB (efecto boca a boca) (Sterman 2000). La tasa de adopción de la nueva tecnología depende de: 1) el número de los clientes actuales, 2) el número de clientes potenciales y 3) la capacidad de persuasión de los clientes para convencer a los clientes potenciales de pasar a la nueva tecnología.

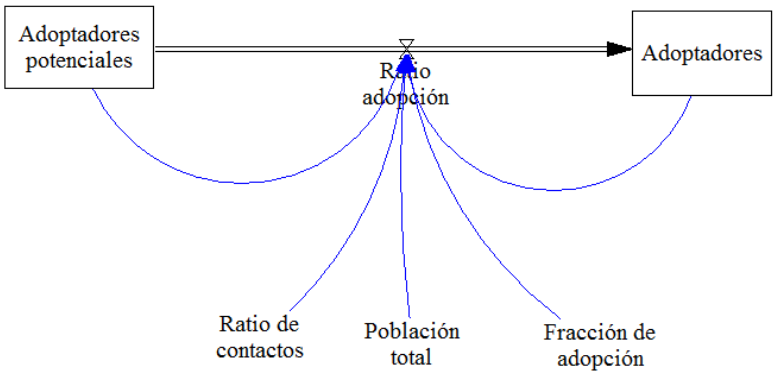


Figura 4.2: Modelo de difusión epidemiológico

Existen dos bucles de realimentación; uno el positivo, el efecto boca a boca, y otro, el negativo, la saturación del mercado. Al contrario que en una epidemia real, el modelo epidemiológico de difusión no necesita para infectar el que dos individuos entren en contacto físico, el contagio se puede realizar por teléfono, *email,* etc.

Uno de los problemas de este tipo de modelos es el inicio. Sin información acerca de la innovación, ningún individuo se atreve a adoptarla por muy innovador que sea. Y es que en este tipo de modelos la adopción 0 es un punto de equilibrio. Para subsanar este problema, Bass (1969) desarrolló un modelo que solucionaba el problema de "arranque".Modelo de Difusión de Bass

4.3 Modelo de Difusión de Bass

El modelo original de Bass estaba destinado a estimar las ventas de nuevos productos, partiendo del modelo epidemiológico; este modelo describe la difusión de innovaciones tecnológicas en una población mediante la siguiente ecuación.

$$\frac{dN(t)}{dt}=(p+(\frac{q}{m})N(t))(m-N(t)) \quad \text{o} \quad \frac{dF(t)}{dt}=(p+qF(t))(1-F(t)) \tag{4.1}$$

Donde N(t) es el acumulado de seguidores en el momento t, m es el techo de población, p es el coeficiente de innovación o de influencia externa y q es el coeficiente de imitación o de influencia interna.

El número de individuos que adoptan la innovación en un intervalo de tiempo *dt* es el resultado de la suma de: 1) Un fenómeno de contagio que depende del número de individuos que hayan adquirido la innovación y 2) un fenómeno de saturación que está ligado al tamaño del mercado potencial.

El modelo es exactamente igual al epidemiológico, pero, gracias al parámetro p, coeficiente de innovación, que mide la propensión de un individuo a adoptar una innovación por influencia externa (precio, publicidad, necesidades), se soluciona el problema de inicio del modelo de difusión epidemiológico.

El parámetro *q* (positivo) representa la influencia de los poseedores de la innovación sobre los clientes potenciales. Cuando *q* es superior a *p*, las ventas de la innovación aumentan , hasta llegar a un máximo, y comienzan a disminuir. Cuando *q* es inferior a *p,* las ventas disminuyen de manera monótona a partir de t=0, no existe difusión.

Al inicio, los únicos compradores son los que se convencen mediante fuentes de información externa (publicidad, *marketing*). Cuando el número de compradores aumenta, el efecto de la publicidad empieza a disminuir frente al efecto del boca a boca. Tan pronto como el efecto boca a boca es dominante, el modelo, empieza a comportarse como un modelo de difusión logístico.

4.4 Modelo de difusión de tecnología de Ahmadian

En este apartado vamos a analizar una implementación del modelo de difusión realizada por Ali Ahmadian, en la Universidad Chlamers de Gotemburgo (Ahmadian 2008), utilizando la herramienta de simulación Vensim PLE. Como indicábamos en el Capítulo 1, partimos de una implementación completa, que en el capítulo siguiente descompondremos para poder reimplementar este mismo modelo, utilizando una metodología híbrida que permita integrar al modelo resultante restricciones geográficas y acotarlo a un ámbito geográfico concreto.

A diferencia de los modelos analizados hasta ahora, este modelo recoge un sinfín de variables relacionadas con el proceso de producción y que, como veremos en el Capítulo 5 resulta, propicio para ser descompuesto en dos partes que se simularán utilizando dos metodologías diferentes.

Al tratarse de una tesis en inglés, hemos traducido el nombre de las variables al castellano para facilitar la comprensión del modelo en la memoria, si bien en las implementaciones realizadas en NetLogo hemos mantenido el nombre de las variables en inglés para facilitar el desarrollo. Al final del capítulo ofrecemos la Tabla diccionario 4.12 con la traducción del nombre de las variables.

Por otro lado, hay que indicar que solo hemos dispuesto de la tesis de Ahmadian y no del modelo en VensimPLE (.mdl). Ha sido necesario realizar la implementación del modelo en el entorno VensimPLE a partir de la tesis, y después hemos realizado la conversión del modelo de VensimPLE a NetLogo utilizando el modelador de Dinámica de Sistemas.

En el modelo inicial la población total era de 100 millones de individuos, este es el único parámetro que hemos modificado ya que las limitaciones de recursos *hardware* para este proyecto no nos permiten ejecutar simulaciones mayores. Hemos optado por simular un mercado compuesto por diez mil agentes. Todos los resultados y figuras que se muestran en este capítulo han sido confeccionados con la población modificada de diez mil agentes.

4.4.1 Principales bucles de realimentación en el modelo

Este modelo de la difusión de la innovación contempla las principales dinámicas generadas

por los factores subyacentes en la difusión de una nueva tecnología. El modelo diferencia entre la dinámica propia de la empresa a la hora de difundir una novedad tecnológica y la dinámica que se genera entre la gente que la adopta.

Existen dos bucles de realimentación positiva que dirigen el comportamiento de la difusión de una tecnología.

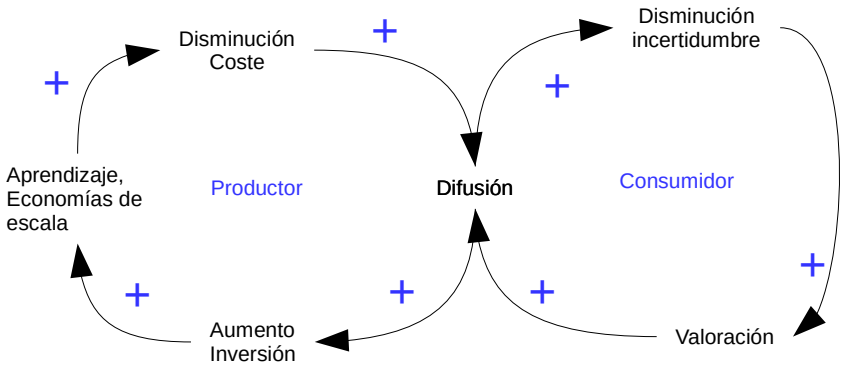


Figura 4.3: Bucles de realimentación en el modelo de difusión de la tecnología

Del lado del productor, el aumento en la inversión permite mejorar la calidad y los costes a través de mecanismos como las economías de escala en la producción. Por otro lado, el de los consumidores, el aumento en el uso de la tecnología produce que la incertidumbre disminuya entre aquellos consumidores indecisos favoreciendo la adopción. De entre estos dos bucles, el bloque que impera a la hora de difundir una tecnología es el del consumidor.

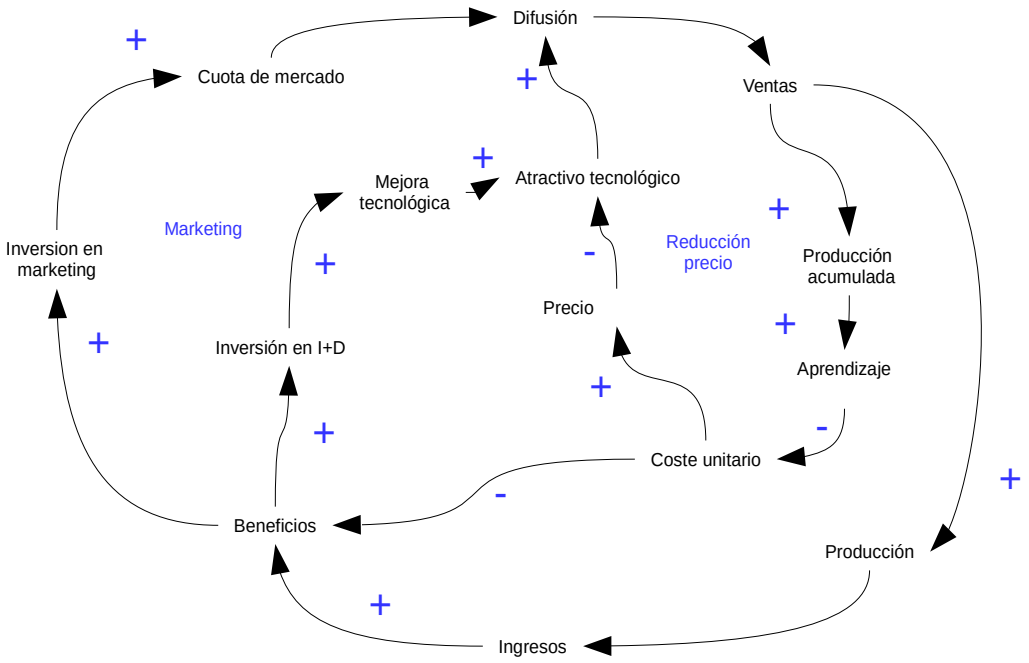


Figura 4.4: Bucles de realimentación positiva en el lado de la producción

Bucles de realimentación positiva en el lado de la producción

La Figura 4.4 muestra los principales bucles de realimentación. Se considera que la variable más importante en este bucle es el aprendizaje, un concepto que implica que los costes unitarios disminuyen gracias al aprendizaje que se obtiene por la producción acumulada.

El primer bucle que analizaremos es la reducción de costes, que, como hemos comentado, es el más importante de todos. La repetición de los procesos de fabricación deriva en una optimización y mejora que permiten abaratar los costes unitarios de producción. A este proceso se le conoce como *learning-by-doing (Arrow 1962)*, que en un primer momento se aplicaba al capital humano (manufacturas) y que después se popularizó gracias a los sistemas de gestión de la calidad *Kaizen* en los procesos de producción de Toyota.

El segundo bucle ilustra otro concepto diferente, *learning-by-researching,* que considera que la I+D (investigación y desarrollo) permite no mejorar el proceso de producción, sino la tecnología en sí, aumentando el atractivo de esta tecnología y, en consecuencia, su difusión.

Por último, como vimos al analizar el modelo de difusión de Bass, existe un tercer bloque importante para poder comenzar a difundir cualquier innovación: el *marketing*.

Bucles de realimentación positiva en el lado del consumidor

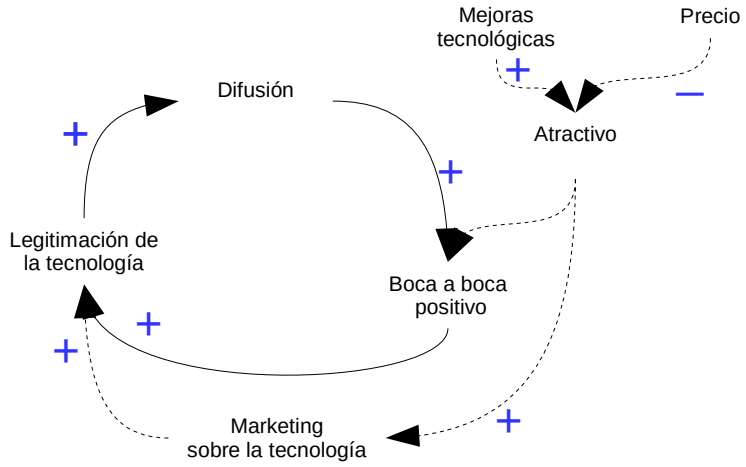


Figura 4.5: Bucles de realimentación ampliados en el lado del consumidor

Cuando una nueva tecnología se difunde en un mercado, genera unos beneficios a los usuarios, lo cual hace disminuir la incertidumbre a la hora de adquirirla. Bajando el precio de la innovación y mejorando la tecnología, la empresa consigue mejorar el atractivo tecnológico. Un aumento del atractivo tecnológico permite la adopción por parte del consumidor y su difusión.

En este modelo el boca a boca y el *marketing* se consideran los actores principales a la hora de legitimar una tecnología nueva. Una mayor difusión aumenta la probabilidad de que se dé un boca a boca positivo; a su vez, cuanto mayor es el nivel de ventas, mayor beneficio, lo cual puede llevar a que se incremente el gasto en *marketing* y los usuarios estén más familiarizados con la nueva tecnología.

En el siguiente diagrama se pueden ver cuáles son las variables determinantes del comportamiento final del modelo; si nos fijamos, la tasa de adopción es una función sobre el

crecimiento de la legitimidad del efecto boca a boca y sobre el crecimiento de la legitimidad por *marketing*, variables que se asemejan a la q (coeficiente de imitación o de influencia interna) y la p (coeficiente de innovación o de influencia externa) del modelo de Bass. Estas dos variables, vistas al analizar los bucles de realimentación positiva en el lado del consumidor, a su vez, están influenciadas por la variable atractivo tecnológico visto en el bucle del lado del productor.

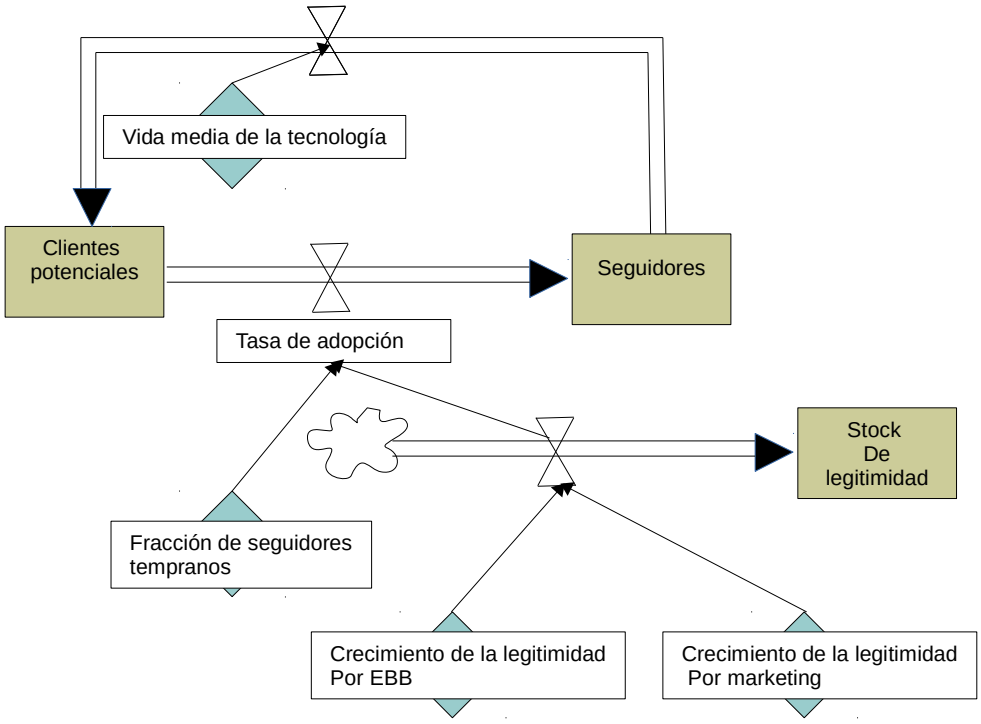


Figura 4.6: Determinación de la tasa de adopción

4.4.2 Descomposición y análisis del modelo

Para tener una visión más amplia del modelo, ha sido dividido en nueve submodelos, empezando por las dinámicas del lado del productor y finalizando con los flujos y niveles del lado del consumidor. En esta sección vamos a analizar los siguientes nueve submodelos:

1. Curva de aprendizaje de la tecnología.
2. Coste de la tecnología.
3. Precio de la tecnología.
4. Ingresos e inversiones.
5. Mejora en la eficacia del *marketing*.
6. Mejora del rendimiento de la tecnología.
7. Atractivo de la tecnología.
8. Actitud del consumidor y legitimidad de la tecnología.
9. Adopción de la tecnología.

Curva de aprendizaje de la tecnología

El coste unitario de la nueva tecnología decae según se va adquiriendo experiencia en su producción. Como el aprendizaje depende en la acumulación de experiencia y no del tiempo, se mide como una función sobre la producción acumulada de la tecnología.

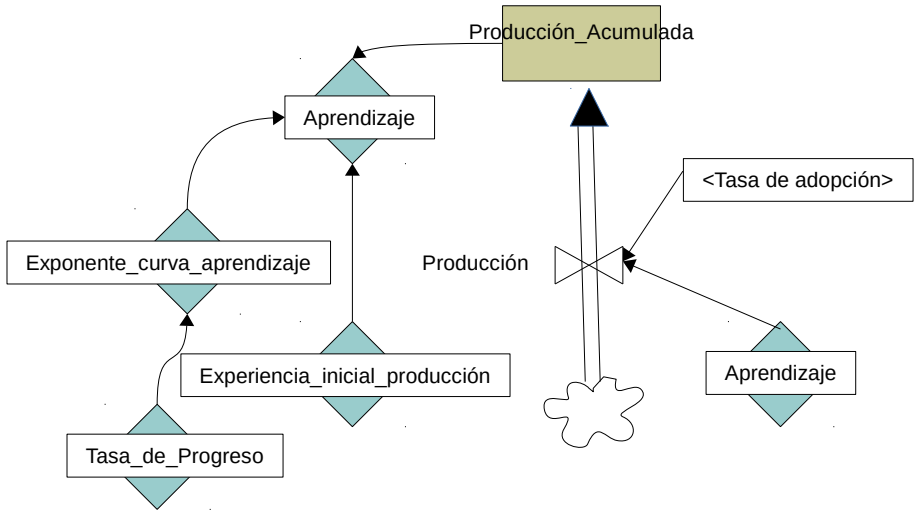


Figura 4.7: Diagrama submodelo curva de aprendizaje

Variable	Notación	Unidad	Valor
Producción	δ_Q	u/año	(4.2)
Número de productos por usuario	q	u/casa	1
Producción acumulada	Q	u	(4.3)
Exponente curva de aprendizaje	S_L	Indef	(4.5)
Experiencia inicial en producción	Q_i	Indef	1000
Aprendizaje	L_c	Indef	(4.4)
Tasa de progreso	ε_c	Indef	0,8

Tabla 4.4: Variables submodelo curva de aprendizaje

A continuación se muestran las ecuaciones que rigen el submodelo de aprendizaje:

$$\delta_Q = q \times R_a \tag{4.2}$$

$$Q = Q_i + \int \delta_Q \, dt \tag{4.3}$$

$L_C = \left(\frac{Q}{Q_i}\right)^{S_L}$ Ecuación formulada por (Kahouli-Brahmi 2008). (4.4)

$$S_L = \frac{\ln(\varepsilon_c)}{\ln(2)} \tag{4.5}$$

ε_c Es la tasa de progreso asociado a la tecnología, que básicamente mide el efecto del aprendizaje sobre el coste.

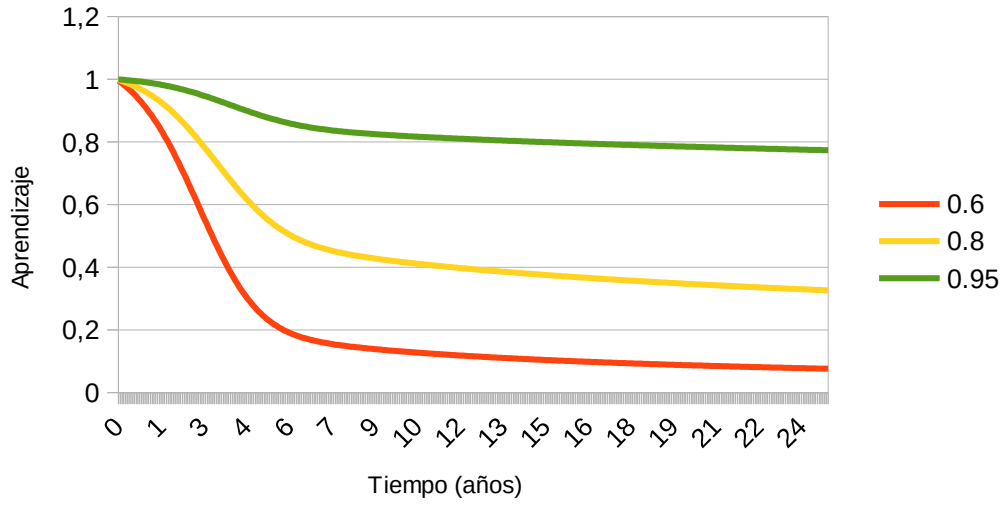


Figura 4.8: Curva de aprendizaje para diferentes tasas de aprendizaje

Coste de la tecnología

En economía, siempre se han considerado dos tipos de costes asociados a la producción: los costes fijos y los costes variables. Los costes fijos son los costes independientes del volumen de producción, no varían sea cual sea la cantidad producida. Los costes variables varían según el volumen de producción: a mayor producción, mayores costes variables.

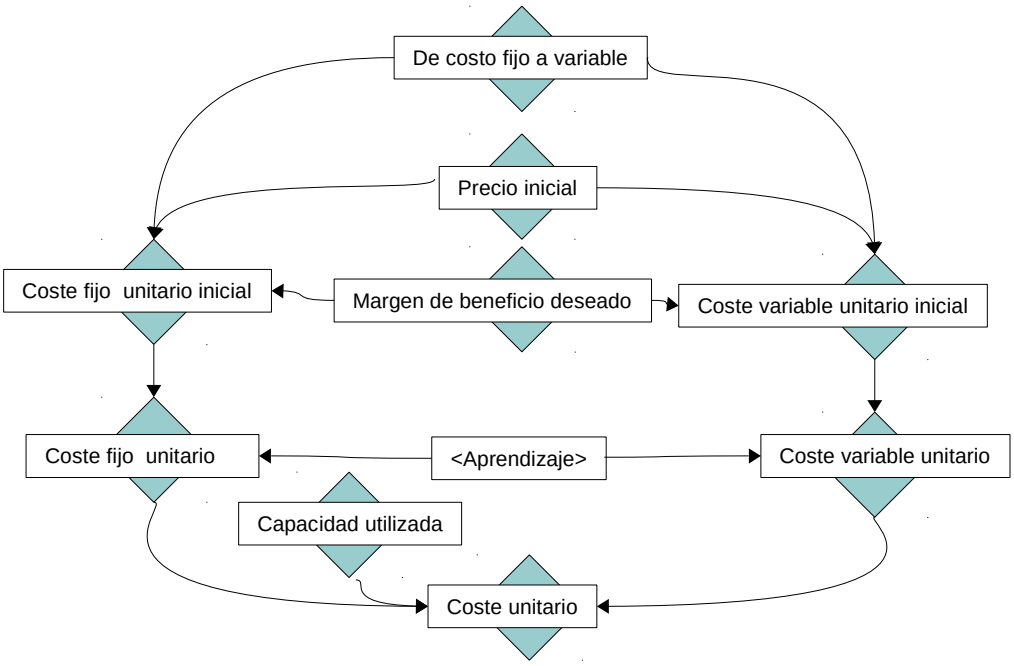


Figura 4.9: Submodelo costes de la tecnología

Variable	**Notación**	**Unidad**	**Valor**
De coste fijo a variable	γ	Indef	3
Precio inicial	P_i	$/u	1000
Coste fijo inicial	C_{fi}	$/u	(4.7)
Coste variable inicial	C_{vi}	$/u	(4.6)
Coste fijo unitario	C_{fu}	$/u	(4.9)
Coste variable unitario	C_{vu}	$/u	(4.8)
Coste unitario	C_u	$/u	(4.10)
Capacidad utilizada	λ	Indef	1
Margen de beneficio deseado	μ_d	Indef	0,2

Tabla 4.5: Variables submodelo coste de la tecnología

A continuación se muestran las ecuaciones que rigen el submodelo de coste de la tecnología:

$$C_{vi}=\left(\frac{P_i}{1+\mu_d}\right)+\left(\frac{1}{1+\gamma}\right) \tag{4.6}$$

La proporción de coste fijo a variable sirve para adaptar el modelo a diferentes escenarios de coste.

$$C_{fi}=\left(\frac{P_i}{1+\mu_d}\right)+\left(\frac{\gamma}{1+\gamma}\right) \tag{4.7}$$

Los dos costes iniciales, dependen del margen de beneficio esperado.

$$C_{vu}=C_{vi}\times L_c \tag{4.8}$$

El coeficiente de aprendizaje junto al coste variable inicial determina los costes variables unitarios.

$$C_{fu}=C_{fi}\times L_c \tag{4.9}$$

$$C_u=C_{vu}+\frac{C_{fu}}{\lambda} \tag{4.10}$$

A mayor utilización de la capacidad de la empresa, el coste unitario baja.

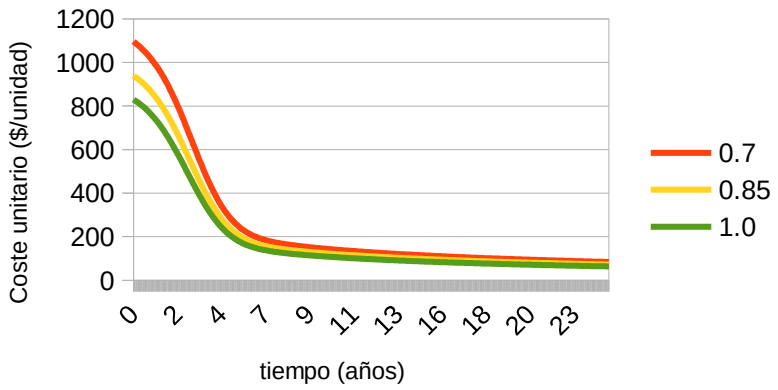


Figura 4.10: Costes unitarios para diferentes valores de capacidad utilizada

Precio de la tecnología

Una vez que tenemos calculado el coste y teniendo en cuenta el margen de beneficio deseado, podemos calcular el precio. A este precio lo llamaremos Precio de coste, ya que se calcula a partir del coste unitario y el margen de beneficio deseado. Sin embargo, el precio objetivo, el precio al cual el producto saldrá al mercado, depende de otros factores. Boulding y Staelin (1990) analizaron los siguientes factores: poder sobre los suministradores, poder sobre los compradores, escasez de competencia, ausencia de amenaza de entrada de competidores, posición de mercado y factor de empresa (calidad, posicionamiento).

Los anteriores factores se condensan en un ratio llamado “Proporción entre precio objetivo y precio coste” que afectan el precio del producto. Esta tasa, se estima como un porcentaje por arriba o por debajo del precio de coste.

Por otro lado, el lapso de tiempo comprendido entre la toma de decisión de ajustar el precio al precio objetivo, desempeña un papel importante en la toma de decisiones de la empresa. Este tiempo se introduce en el modelo con el nombre de "tiempo de ajuste".

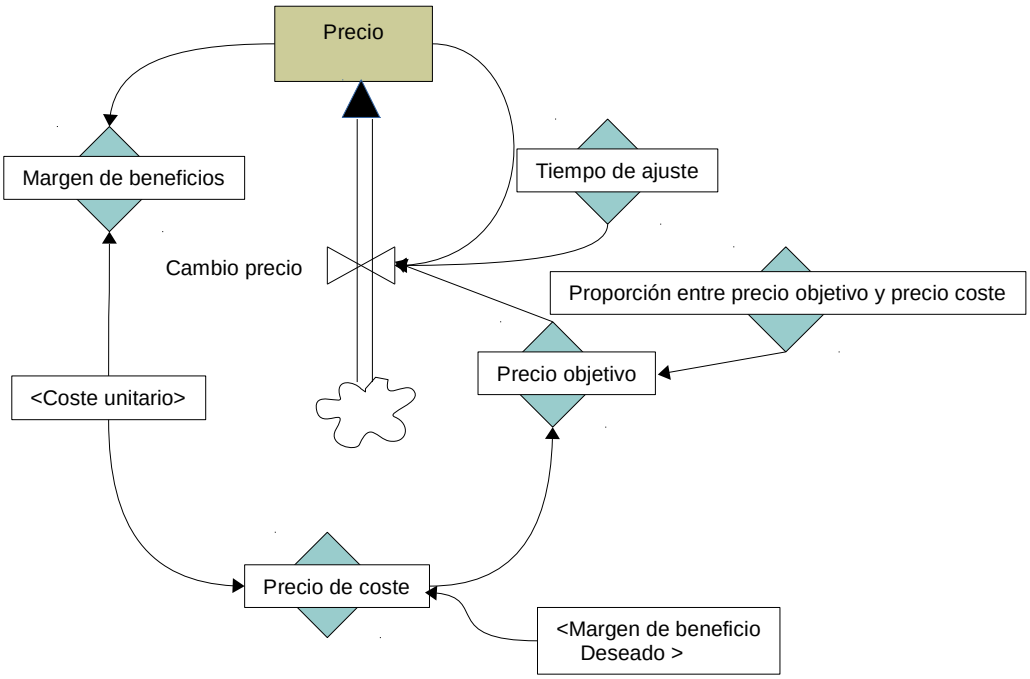


Figura 4.11: Submodelo cálculo del precio

Variable	**Notación**	**Unidad**	**Valor**
Proporción entre precio objetivo y precio coste	Θ	Indef	1,01
Precio objetivo	P_t	$/u	(4.12)
Precio de coste	P_c	$/u	(4.11)
Tiempo de ajuste	T_a	año	0,5
Cambio precio	δ_p	$/(u*año)	(4.13)
Precio	P	$/u	(4.14)
Margen de beneficios	μ	Indef	(4.15)

Tabla 4.6: Variables submodelo precio

A continuación se muestran las ecuaciones que rigen el submodelo de establecimiento del precio:

$$P_c = (1 \times \mu_d) \times C_u \tag{4.11}$$

$$P_t = (P_c) \times \Theta \tag{4.12}$$

$$\delta_p = \frac{dP}{dt} = \frac{P_t - P_c}{T_a} \tag{4.13}$$

$$P = P_i + \int \delta_p \, dt \tag{4.14}$$

$$\mu = \left(\frac{P}{C_u}\right) - 1 \tag{4.15}$$

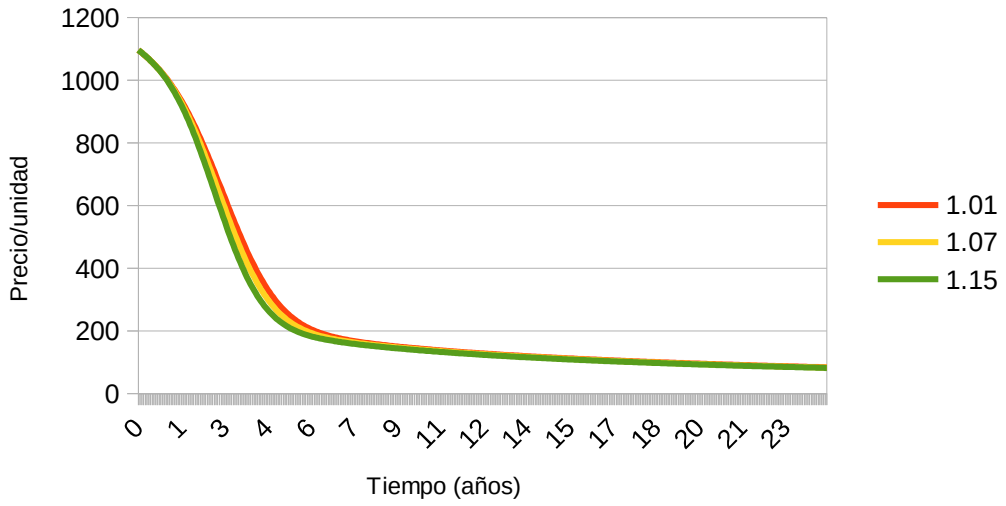


Figura 4.12: Precio para diferentes tasas entre precio objetivo

Ingresos e inversiones

En este modelo, se considera que los beneficios en la venta se pueden dedicar a: Investigación y Desarrollo, mejorando la calidad del producto o el proceso de fabricación; o en *marketing*, haciendo que el producto se difunda por el mercado.

Una parte del beneficio se destina a la inversión, de esta parte se asume que el 50% se dedicará a *marketing* e Investigación y Desarrollo.

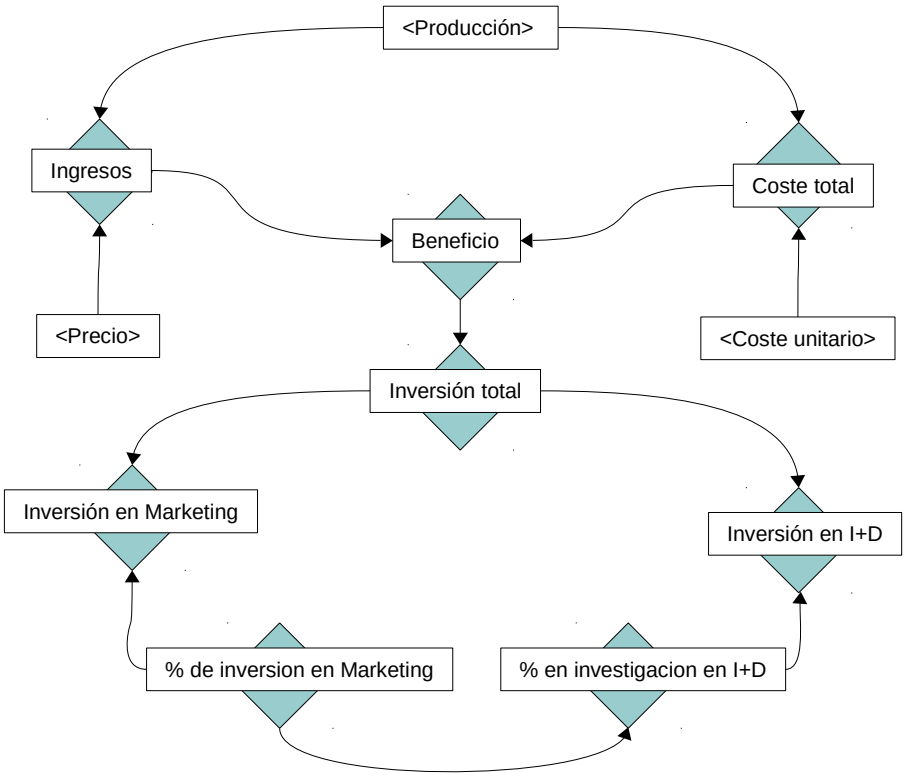


Figura 4.13: Submodelo Ingresos e inversiones

Variable	**Notación**	**Unidad**	**Valor**
Ingresos	ρ	$/año	(4.17)
Coste total	K	$/año	(4.18)
Beneficio	π	$/año	(4.16)
Inversión total	I	$/año	(4.19)
Inversión en marketing	I_m	$/año	(4.24)
Inversión en I+D	I_r	$/año	(4.20)
Porcentaje de beneficio en inversión	β	Indef	0,4
Porcentaje de inversión en marketing	β_m	Indef	0,2
Porcentaje de inversión en I+D	β_r	Indef	(4.21)

A continuación se muestran las ecuaciones que rigen el submodelo de ingresos e inversiones:

$$\pi = \rho - K \tag{4.16}$$

$$\rho = \delta_p \times P \tag{4.17}$$

$$K = \delta_p \times C_u \quad (4.18)$$

$$I = \pi \times \beta \quad (4.19)$$

$$I_r = I \times \beta \quad (4.20)$$

$$\beta_r = 0{,}5 - \beta_m \quad (4.21)$$

$$0 \leq \beta \leq 1 \quad (4.22)$$

$$0 \leq \beta_m \leq 0{,}5 \quad (4.23)$$

$$I_m = I \times \beta_r \quad (4.24)$$

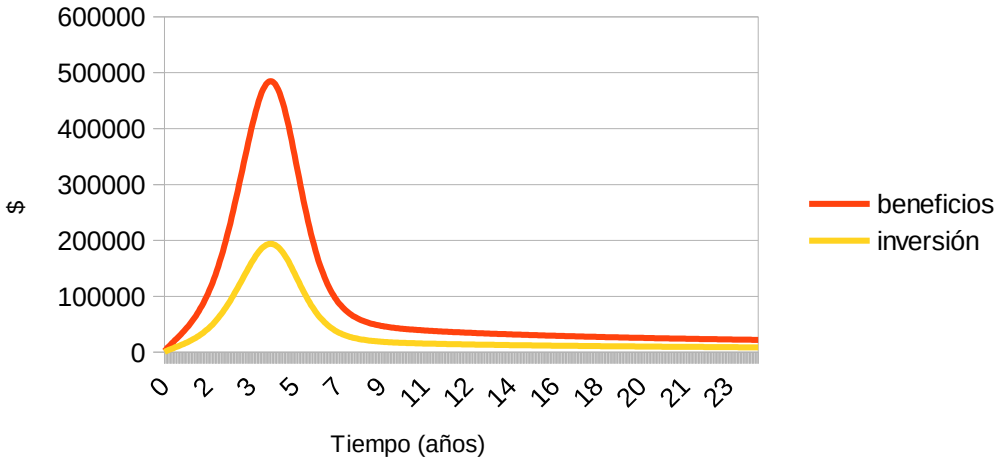


Figura 4.14: Beneficios e Inversión total

Rendimiento del *marketing*

En este modelo la eficacia del *marketing* está directamente asociada al total invertido en *marketing*,:a más *marketing*, más clientes estarán interesados en adquirir el producto y habrá más ventas, con lo que aumentará el gasto en *marketing* y se podrá invertir en más *marketing*.

Se añade al modelo una variable exógena denominada "mejora del rendimiento del *marketing* por inversión"*, que, como el nombre señala, sirve para medir el efecto en las ventas que produce el aumento en inversión en *marketing*.

* La variable mejora del rendimiento del marketing por inversión, a resultado muy influyente en el modelo.

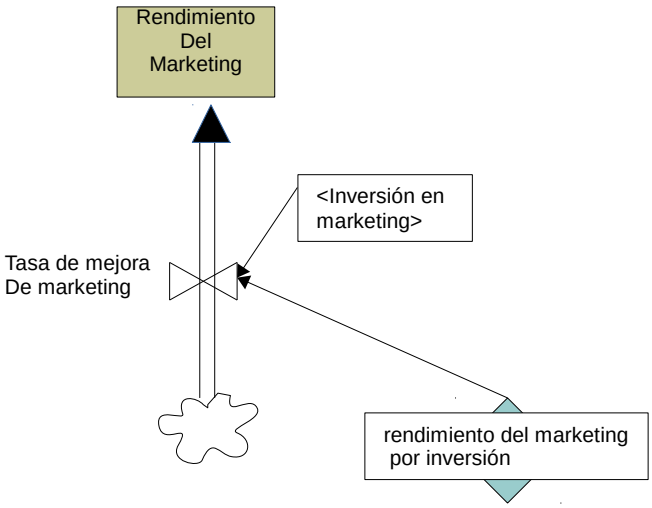


Figura 4.15: Submodelo rendimiento del marketing

Variable	Notación	Unidad	Valor
Mejora del rendimiento del marketing por inversión	η_m	1/$	0,00001
Tasa de mejora de rendimiento del marketing	δ_m	1/$	(4.25)
Rendimiento del marketing	M	Indef	(4.26)

Tabla 4.7: Variables del submodelo rendimiento del marketing

$$\delta_M = \eta_m \times I_m \tag{4.25}$$

$$M = \int \delta_M \, dt \tag{4.26}$$

Mejora del rendimiento de la tecnología

El submodelo de esta sección es simple y plantea que la inversión en I+D mejora el rendimiento de la tecnología, gracias al conocimiento adquirido (Leenders 2002). El modelo comienza

considerando el hecho de que la acumulación de conocimiento, puede determinarse por la cantidad de inversión en I+D realizada. El impacto de la investigación en I+D en el modelo es equivalente al efecto del aprendizaje por fabricación que hemos analizado anteriormente.

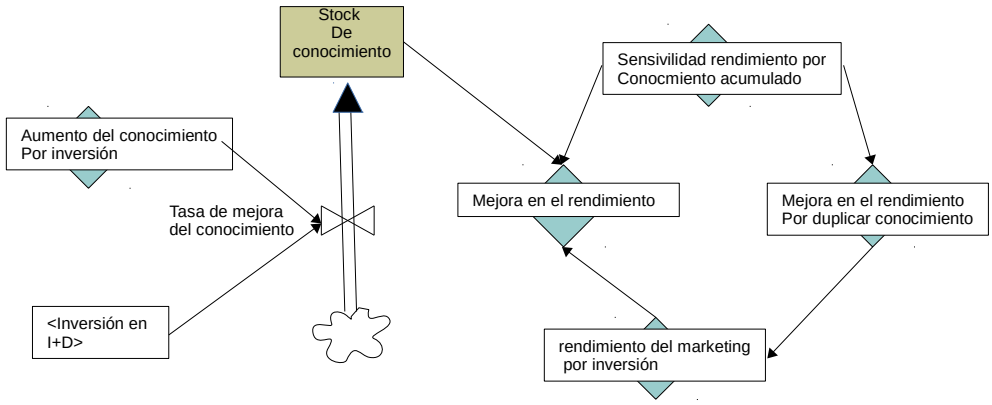


Figura 4.16: Submodelo mejora del rendimiento de la tecnología

Variable	**Notación**	**Unidad**	**Valor**
Mejora en el rendimiento por duplicar conocimiento	ε_k	Indef	0,5
Tasa de contribución a la mejora del rendimiento por cocimiento acumulado	ω	Indef (0,1)	0,5
Sensibilidad del rendimiento por conocimiento acumulado	S_k	Indef	(4.29)
Mejora en el rendimiento	R	Indef	(4.30)
Stock de conocimiento	K	Indef	1-(4.28)
Aumento del conocimiento por inversión	η_r	1/$	0,000001
Tasa de incremento del conocimiento	δ_k	1/año	(4.27)

Tabla 4.8: Variables del submodelo mejora del rendimiento de la tecnología

Ecuaciones del submodelo de mejora del rendimiento de la tecnología:

$$\delta_k = I_r \times \eta_k \tag{4.27}$$

$$K = \int \delta_k \, dt \tag{4.28}$$

$$S_k = \ln \frac{(1+\varepsilon_k)}{\ln(2)} \tag{4.29}$$

$$R = K^{S_k} \times \omega \tag{4.30}$$

Atractivo de la tecnología

El atractivo de la tecnología influencia positivamente el incremento de ventas de la tecnología en el mercado. En el modelo, el nivel de atractivo tecnológico se basa en dos variables: la reducción de precio y la mejora del rendimiento. El atractivo de la tecnología se estima comparando el precio inicial con el precio de cada año. Por otra parte, el atractivo por la mejora del rendimiento/calidad es asumido como una función lineal sobre la mejora en el rendimiento de la tecnología.

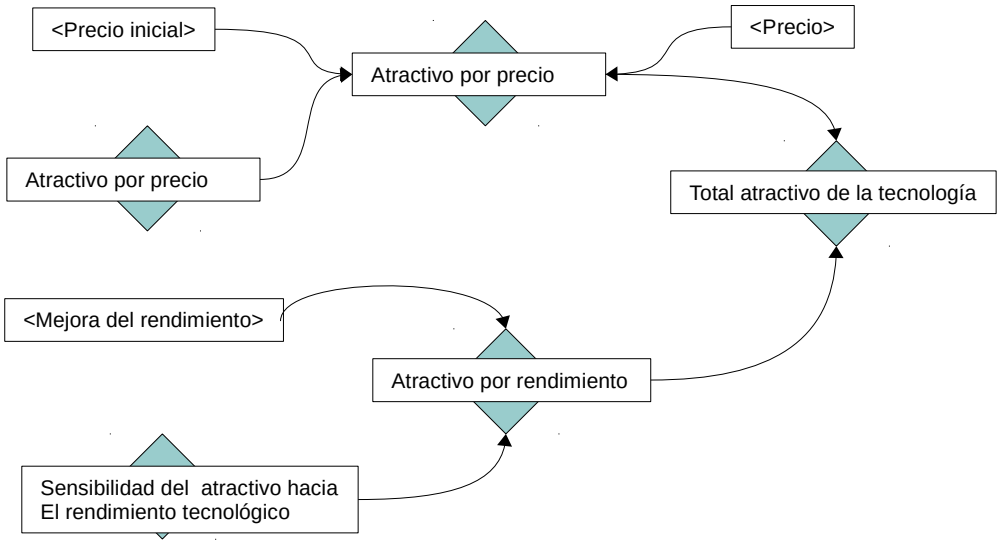


Figura 4.17: Submodelo atractivo de la tecnología

Variable	Notación	Unidad	Valor
Sensibilidad al atractivo del precio	S_p	Indef	-2
Atractivo del precio	α_p	Indef	(4.31)
Sensibilidad al atractivo tecnológico	S_r	Indef	0,5
Atractivo hacia el rendimiento tecnológico	α_r	Indef	(4.32)
Atractivo total de la tecnología	α	Indef	(4.33)

Tabla 4.9: Variables del submodelo atractivo de la tecnología

$$\alpha_p = \exp\left(\frac{S_p \times P}{P_i}\right) \tag{4.31}$$

$$\alpha_r = S_r \times R \tag{4.32}$$

$$\alpha = \alpha_p + \alpha_r \tag{4.33}$$

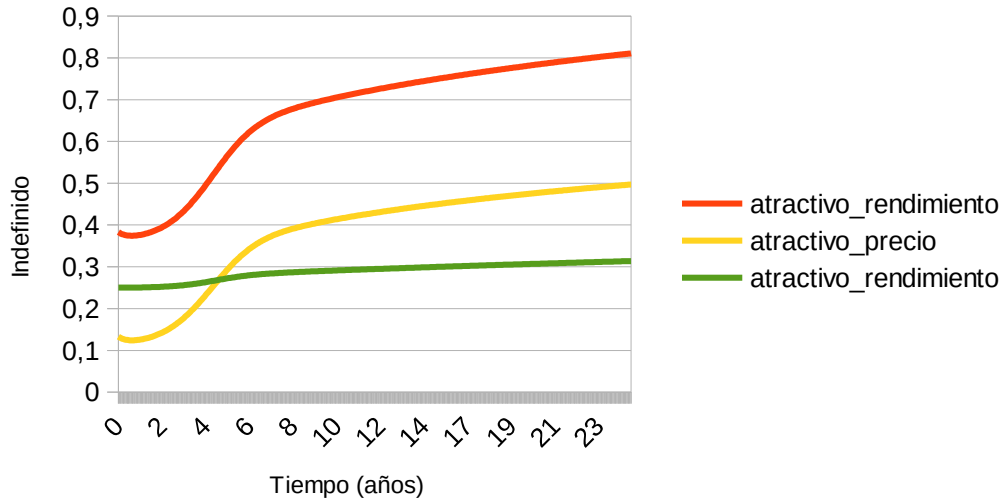


Figura 4.18: Submodelo atractivo de la tecnología

Actitud del consumidor y legitimidad de la tecnología

La legitimidad de la tecnología hace referencia a la popularidad y aceptación por parte de los usuarios de la nueva tecnología. Una tecnología legitimada será más fácilmente adoptada, ya que los usuarios la reconocerán como más válida importante y fiable, de la cual depender.

Los productores, pueden aumentar la legitimidad y la actitud de los consumidores hacia la nueva tecnología, aumentando la inversión en I+D con el objeto de mejorar la calidad o bajar el precio.

En este modelo, el crecimiento de la atracción hacia la tecnología influencia positivamente el crecimiento de variables que hacen que la tecnología gane legitimidad entre los consumidores. El mecanismo por el cual se difunde la legitimidad de la tecnología es el efecto boca a boca y el *marketing*. Cuanto mayor sea el número de seguidores, mayor será la probabilidad de que alguno convenza a un cliente potencial para que adquiera la nueva tecnología. La inversión en *marketing*, al contrario que el boca a boca, depende de los beneficios que obtiene la empresa, aunque, en último factor, también depende del atractivo de la tecnología que crece exógenamente por el *learn-by-doing* y el I+D.

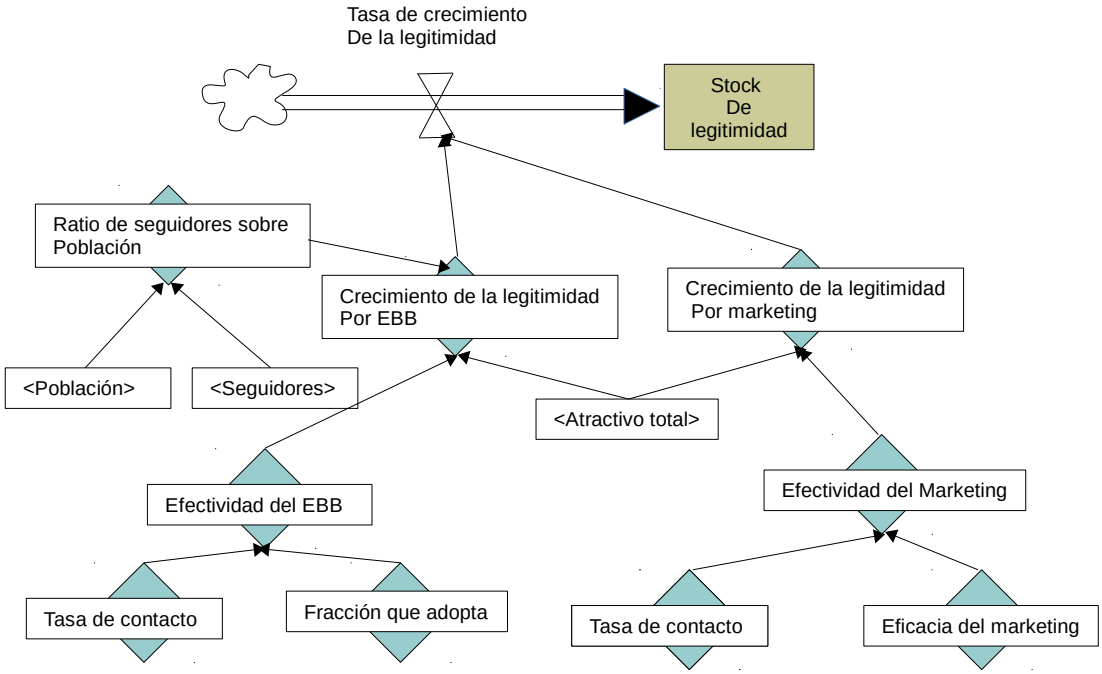


Figura 4.19: Submodelo legitimidad de la tecnología

Variable	Notación	Unidad	Valor
Tasa de contacto entre clientes y medios de comunicación	R_m	1/año	0.12
Eficacia del marketing	E_m	1/año	(4.34)
Crecimiento de la legitimidad por marketing	L_m	1/año	(4.38)
Tasa de contacto	R_c	1/año	50
Fracción seguidora	A_f	Indef	0.01
Efectividad EBB	E_w	1/año	(4.35)
Crecimiento de la legitimidad por EBB	L_w	1/año	(4.37)
Tasa de seguidores sobre población	σ	Indef	(4.36)
Tasa de crecimiento de la legitimidad	δ_L	1/año	(4.39)
Stock de legitimidad	L	Indef	(4.40)

Tabla 4.10: Variables del submodelo legitimidad de la tecnología

$$E_m = R_m \times M \tag{4.34}$$

$$E_w = R_c \times A_f \tag{4.35}$$

$$\sigma = \frac{N_a}{N} \tag{4.36}$$

$$L_w = E_w \times \sigma \times (1+\alpha) \tag{4.37}$$

$$L_m = E_m \times (1+\alpha) \tag{4.38}$$

$$\delta_L = L_m + L_w \tag{4.39}$$

$$L = \int \delta_L \, dt \tag{4.40}$$

Adopción de la tecnología

La introducción de una nueva tecnología en un mercado es un proceso dividido en diferentes fases, en las que, en cada una, intervienen diferentes tipos de individuos, tal y como vimos al analizar el modelo de Rogers. Al comienzo, la tecnología solo es adquirida por innovadores, lo cuales asumen el riesgo de adquirir una innovación no contrastada. Los innovadores mediante el EBB (efecto boca a boca) y el *marketing* son las claves para comenzar la difusión, como vimos al analizar el modelo de Bass.

En este modelo, la tasa de adopción se determina por dos variables: por un lado, la tasa de crecimiento de la legitimidad y, por otro, la fracción de seguidores tempranos (innovadores). Los seguidores tempranos adoptan la tecnología independientemente de su legitimidad, podemos decir que son inmunes al efecto boca a boca y al *marketing*.

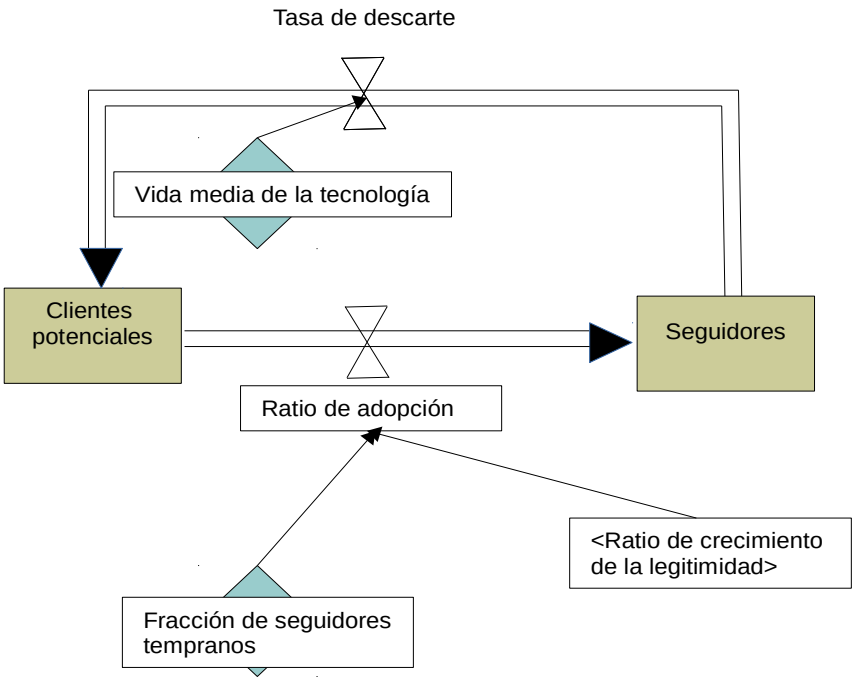


Figura 4.20: Adopción de la tecnología

Variable	**Notación**	**Unidad**	**Valor**
Clientes potenciales	N_p	casa	(4.44)
Seguidores	N_a	casa	(4.41)
Tasa de adopción	R_a	casa/año	(4.42)
Fracción de seguidores tempranos	δ_E	casa/año	0.002
Tasa de descarte	R_d	casa/año	(4.43)
Vida media de la tecnología	T_f	Año	8

Tabla 4.11: Submodelo adopción de la tecnología

$$N_a = N_{a0} + \int R_a \, dt \tag{4.41}$$

$$R_a = (\delta_E + \delta_L) \times N_p \tag{4.42}$$

$$R_d = \frac{N_a}{T_f} \tag{4.43}$$

$$N_p = N - N_a \tag{4.44}$$

Este modelo se centra en la primera adquisición[1]. Para poder simular la readquisición del producto, se asume que los usuarios que han adoptado la tecnología, una vez la hayan descartado, pasan a ser parte del *stock* de clientes potenciales. En este caso la tasa a la que el producto es descartado depende del número de seguidores y de la vida útil de la innovación.

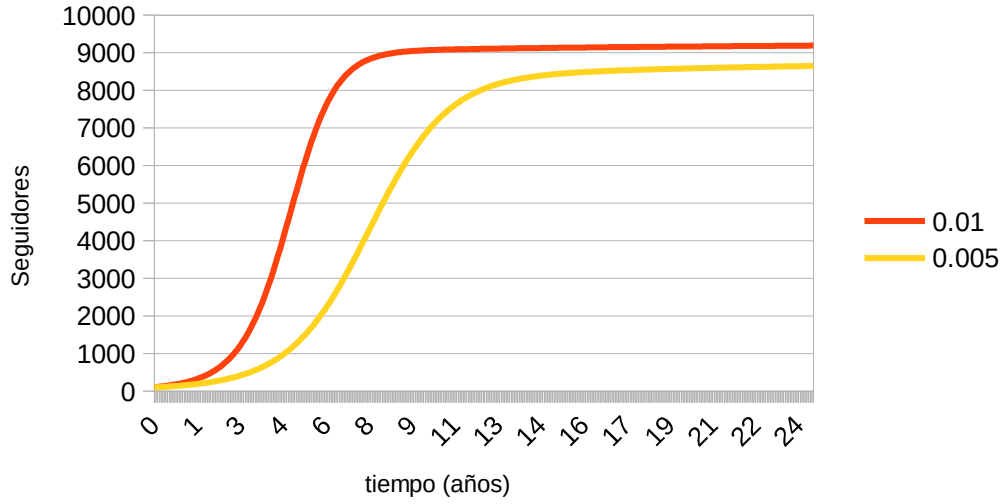


Figura 4.21: Adopción

Como indicábamos al inicio de este capítulo, los nombres de las variables en esta memoria han sido traducidas del inglés al castellano; no obstante, las variables utilizadas en VensimPLE mantienen el nombre original en inglés.

Al contrario que NetLogo, VensimPLE soporta espacios en blanco en el nombre de las variables, con lo que en NetLogo los espacios en blanco en las variables, han sido sustituidos por el carácter "_".

1 Por eso en la Figura 58 no aparecen ciclos

Castellano	Inglés (NetLogo)	Variable	Unidad
Aprendizaje	Learning	L_c	Indef
Atractivo del precio	Attractiveness from price	α_p	Indef
Atractivo hacia el rendimiento tecnológico	Attractiveness from technology performance	α_r	Indef
Atractivo total de la tecnología	Total attractiveness of technology	α	Indef
Aumento del conocimiento por inversión	Knowledge increment per investment	η_r	1/$
Beneficio	Profit	π	$/año
Cambio de precio	Change in price	δ_p	$/(u*año)
Capacidad utilizada	Capacity utilization	λ	Indef
Coste fijo inicial	Initial fixed cost	C_{fi}	$/u
Coste fijo unitario	Unit fixed cost	C_{vu}	$/u
Coste total	Total cost	K	$/año
Coste unitario	Unit cost	C_u	$/u
Coste variable inicial	Initial variable cost	C_{vi}	$/u
Crecimiento de la legitimidad por EBB	Growth of legitimacy through WoM	L_w	1/año
Crecimiento de la legitimidad por marketing	Growth of legitimacy through marketing	L_m	1/año
De coste fijo a variable	Fixed to variable cost	γ	Indef
Efectividad EBB	WoM effectiveness	E_w	1/año
Eficacia del marketing	Marketing effectiveness	E_m	1/año
Experiencia inicial en producción	Initial production experience	Q_i	Indef
Exponente curva de aprendizaje	Learning curve exponent	S_L	Indef
Fracción de seguidores tempranos	Early adopters fraction	δ_E	casa/año
Fracción seguidora	Adoption fraction	A_f	Indef (0, 0.1)
Ingresos	Revenue	ρ	$/año
Inversión en I+D	Investment in R&D	I_r	$/año
Inversión en marketing	Investment in marketing	I_m	$/año
Inversión total	Total investment	I	$/año
Margen de beneficio deseado	Desired profit margin	μ_d	Indef
Margen de beneficios	Profit margin	μ	Indef

Mejora del rendimiento del marketing por inversión	Marketing performance improvement per investment	η_m	1/$
Mejora en el rendimiento	Performance improvement	R	Indef
Mejora en el rendimiento por duplicar conocimiento	Performance improvement per doubling of knowledge	ε_k	Indef
Número de productos por usuario	Number of product per adopter	q	u/casa
Porcentaje de beneficio en inversión	Share of profit in investment	β	Indef (0,1)
Porcentaje de inversión en I+D	Share of investment in R&D	β_r	Indef
Porcentaje de inversión en marketing	Share of investment in marketing	β_m	Indef (0,0.5)
Precio	Price	P	$/u
Precio de coste	Cost price	P_c	$/u
Precio inicial	Initial price	P_i	$/u
Precio objetivo	Target price	P_t	$/u
Producción	Production	δ_Q	u/año
Producción acumulada	Cumulative production	Q	u
Proporción entre precio objetivo y precio coste	Ratio of target price to cost price	Θ	Indef
Rendimiento del marketing	Marketing performance	M	Indef
Seguidores	Adopters	N_a	casa
Seguidores potenciales	Potential adopters	N_p	casa
Sensibilidad al atractivo del precio	Sensitivity of attractiveness to the price	S_p	Indef
Sensibilidad del atractivo al rendimiento tecnológico	Sensitivity of attractiveness to the technology performance	S_r	Indef
Sensibilidad del rendimiento por conocimiento acumulado	Sensitivity of performance per cumulative knowledge	S_k	Indef
Stock de conocimiento	Stock of knowledge	K	Indef
Stock de legitimidad	Stock of legitimacy	L	Indef
Tasa de adopción	Adoption rate	R_a	casa/año
Tasa de contacto	Contact rate	R_c	1/año
Tasa de contacto entre cliente y medios de comunicación	Rate of contact between customers and means of marketing	R_m	1/año

Tasa de contribución a la mejora del rendimiento por conocimiento acumulado	Contribution share of knowledge accumulation in performance improvement	ω	Indef (0,1)
Tasa de crecimiento de la legitimidad	Rate of growth of legitimacy	δ_L	1/año
Tasa de descarte	Discard rate	R_d	casa/año
Tasa de mejora de rendimiento del marketing	Marketing performance improvement rate	δ_m	1/$
Tasa de mejora del rendimiento	Rate of increase in knowledge	δ_k	1/año
Tasa de progreso	Progress ratio	ε_C	Indef
Tasa de seguidores sobre población	Ratio of adopters to population	σ	Indef
Tiempo de ajuste	Adjustment time	T_a	año
Vida media de la tecnología	Technology life time	T_f	año

Tabla 4.12: Nombre de variables castellano-inglés

4.5 Conclusiones

En este capítulo hemos presentado el concepto de difusión de la innovación. También hemos presentado el trabajo más relevante realizado por (Rogers 1962), que, como decíamos, es una de las teorías sociales más respetadas, así como aplicada empírica y conceptualmente. En él nos hemos basado para realizar el modelo de difusión de la innovación híbrido que describiremos en los siguientes capítulos.

Hemos analizado el modelo matemático de la difusión propuesto por Frank Bass, modelo ampliamente utilizado, y, por último, hemos visto una implementación de un modelo de difusión realizada en Dinámica de Sistemas por Ali Ahmadian en el entorno Vensim, que engloba y amplia el modelo de F. Bass.

Como indicábamos al principio del Apartado 4.4 solo hemos tenido acceso a la tesis y no a la implementación del modelo; el trabajo de reimplementación estaba contemplado en el anteproyecto pero, como veremos en el Capítulo 8, ha requerido un esfuerzo mucho mayor del estimado inicialmente. Tenemos que indicar que este mayor esfuerzo ha sido debido sobre todo al diferente tratamiento que realizan VensimPLE y NetLogo de las variables de nivel, cuestión que nos llevó varios días descubrir y corregir.

5 Análisis y descomposición del modelo de difusión

5.1 Introducción

En este capítulo vamos a descomponer el modelo de difusión de Ahmadian en dos niveles: un nivel micro, que simulará el comportamiento individual de los seguidores de la innovación y que se implementará utilizando la metodología multiagente, y un nivel empresarial que se simulará mediante la Dinámica de Sistemas.

A la hora de descomponer el modelo de difusión, haremos uso de las teorías y principios de los modelos de Everett y de Bass, estudiados en el Capítulo 4.

5.2 Justificación de la descomposición del modelo

Tradicionalmente se ha considerado que el mercado se comporta como el mundo real, regido por leyes físicas universales. Desgraciadamente, estas consideraciones no funcionan, y esto es porque el mercado se rige por las decisiones y acciones que adoptan los individuos que forman el mercado. Las leyes del mercado, al contrario que las leyes físicas, emergen de las decisiones de las personas que lo conforman. La fuerza de la simulación social basada en agentes reside en que se apoya en casos de uso y modelos matemáticos.

Según Rand y Rust (2011) las razones por las que resulta adecuado simular el mercado mediante sistemas multiagentes son:

1. El mercado está compuesto por más de un par de agentes; según Rand y Tust, en el caso de un número menor, la teoría de juegos suele resultar más eficiente.
2. Cuando las decisiones que toman los agentes se adoptan mediante interacciones locales complejas.
3. Cuando existe una heterogeneidad de individuos, puede haber personas con diferentes niveles de renta, sociales y demográficos.
4. Permiten la representación de entornos dinámicos desde simples espacios bidimensionales hasta datos provenientes de sistemas GIS.
5. Cuando el proceso que se va a modelar es dinámico, cambia en el tiempo.

6. Porque los agentes pueden ser adaptativos y variar sus acciones si producen un resultado negativo.

5.3 Descomposición del modelo

Como veíamos en la Figura 4.3 del modelo de difusión que hemos presentado, existen dos niveles que justifican intentar realizar una aproximación multimétodo: el primero, el nivel del productor, que desde una perspectiva *top-down* se modelará utilizando la Dinámica de Sistemas y, por otro lado, el mercado compuesto de consumidores, cuyas reacciones de adopción de la tecnología, transmisión de impresiones, es claramente *botom-up* y que se modelará mediante sistemas multiagentes. Ambos modelos se combinarán de una manera bidireccional en NetLogo.

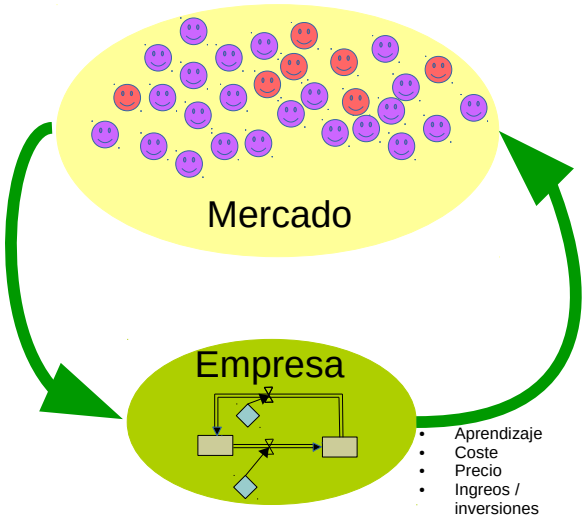


Figura 5.1: Descomposición en niveles del modelo de difusión

El modelo de Dinámica de Sistemas proveerá *inputs* al modelo multiagente y éste, a su vez, suministrará los niveles de seguidores y clientes potenciales al modelo de Dinámica de Sistemas. Para poder entender la transformación que vamos a realizar para dar paso a los agentes, nos tenemos que centrar en la Figura 4.20: Adopción de la tecnología .

Al descomponerlo de esta manera, no solo podemos obtener emergencias del modelo multiagente, sino que, al individualizar los compradores, podemos introducir consideraciones sobre la distribución geográfica de los agentes que de otra forma serían imposibles de modelar utilizando la metodología de Dinámica de Sistemas.

Esta división implica que el nivel o naturaleza de algunas variables del modelo cambie por completo, pudiéndose dar estos tipos de cambios:

1. Algunas variables de nivel desaparecerán del modelo, y su estimación se realizará como un agregado del nivel multiagente.
2. Algunas variables auxiliares también desaparecen al convertirse en atributos propios del agente. En este caso, esta individualización afectará al comportamiento individual y no al colectivo.
3. Algunas constantes pasarán a ser variables estocásticas, como, por ejemplo, la variable ratio de contacto**.**

Los submodelos afectados en la descomposición son el 8. Actitud del consumidor y legitimidad de la tecnología, y 9. Adopción de la tecnología del capítulo anterior. La Actitud del consumidor no se modelará de manera colectiva (agregada), sino que ahora se individualizará al agente, con lo que las variables afectadas pasarán a ser individuales del agente.

Variable	Notación	Transformación
Clientes potenciales	N_p	Agregado SMA
Seguidores	N_a	Agregado SMA
Tasa de adopción	R_a	Agregado SMA
Fracción de seguidores tempranos	δ_E	Sin cambios
Tasa de descarte	R_d	Sin cambios
Vida media de la tecnología	T_f	Sin cambios
Tasa de contacto entre clientes y medios de comunicación	R_m	Estocástica
Eficacia del marketing	E_m	Sin cambios
Crecimiento de la legitimidad por marketing	L_m	Atributo agente
Tasa de contacto	R_c	Estocástica
Fracción seguidora	A_f	Agregado SMA
Efectividad EBB	e_w	Sin cambios
Crecimiento de la legitimidad por EBB	L_w	Atributo agente
Tasa de seguidores sobre población	σ	Sin cambios
Tasa de crecimiento de la legitimidad	δ_L	Atributo agente
Stock de legitimidad	L	Atributo agente

Tabla 5.1: Variables afectadas por la división

5.4 Descripción del modelo de Dinámica de Sistemas

Como ya hemos mencionado en el aparado anterior, el modelo de dinámica de sistemas sufre unos cambios mínimos: los submodelos afectados son el 8. Actitud del consumidor y legitimidad de la tecnología, y 9. Adopción de la tecnología del capítulo anterior.

1. Los niveles del modelo de Dinámica de Sistemas Clientes potenciales y Seguidores ahora se reducen a sendos sumatorios de tipos de agentes del sistema multiagente.

$$\begin{aligned} Seguidores &= \sum tadopters \\ Clientes\ Potenciales &= \sum tpotentials \end{aligned} \tag{5.1}$$

2. Parte del submodelo de legitimidad de la tecnología pasará a ser parte del sistema basado en agentes en concreto la variable ***Stock de legitimidad*** pasa a ser un atributo de cada agente, que será la suma de dos variables ***legitimacy-wom*** y ***legitimacy-marketing,*** que irán acumulando el nivel de convencimiento para adoptar la tecnología que tendrá cada individuo.

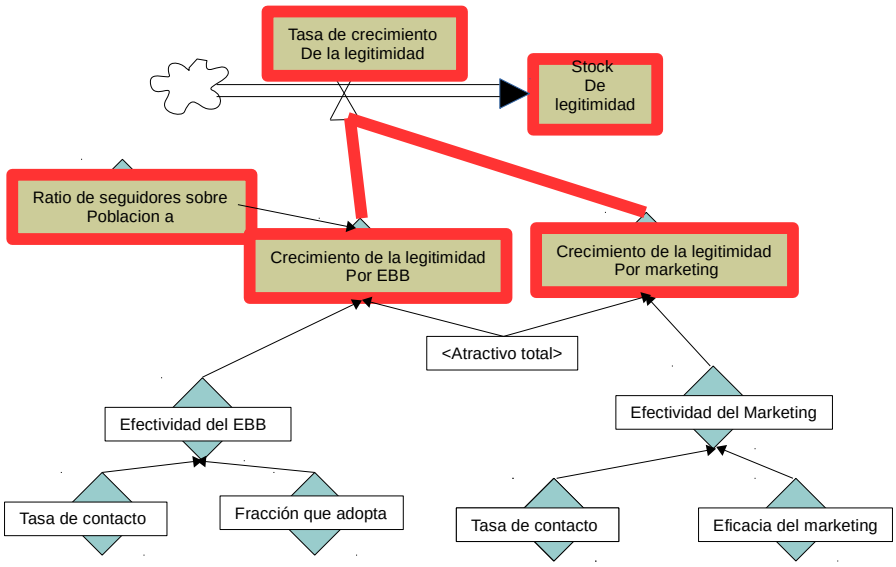


Figura 5.2: Variables omitidas en el modelo multimétodo

El cálculo de estas variables se realizará siguiendo las mismas ecuaciones que en el modelo de Dinámica de Sistemas, pero de manera individual en el modelo multiagente.

```
set legitimacy-marketing ( legitimacy-marketing +
pat-intensity * Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology ) * dt )

set legitimacy-wom (legitimacy-wom
+ Wom_effectiveness * ( 1 + Total_attractiveness_of_technology ) * dt)
```

Código 5.1: Cálculo de variables en MA

5.5 Descripción del sistema basado en agentes

5.5.1 Tipos de agentes

El mercado estará compuesto por tres tipos de agentes, que han sido diseñados siguiendo

varios de los principios expuestos en el Apartado 2.3.5: son agentes reactivos, extremadamente simples, con un mínimo de estados internos y que interaccionan con el entorno y el resto de agentes.

Si analizamos los principios del Apartado 2.3.5 y las fases de adopción del modelo de Rogers 4.2, vemos que lo expuesto en el párrafo anterior es contradictorio o da pie a otra interpretación. Podríamos haber creado unos agentes adaptativos con cinco estados (Conocimiento, Persuasión, Decisión, Implantación , Confirmación), que indicarán su grado de propensión a adoptar la innovación.

Teniendo en cuenta que pretendemos ejecutar una simulación con un gran número de agentes y que además pretendemos enlazar este modelo con herramientas GIS, limitando el ámbito de actuación de los agentes a posiciones geográficas delimitadas, hemos seguido un principio básico en la simulación social basada en agentes: el principio citado por Axelrod en su libro (1997): KISS (Keep It Simple, Stupid!), que viene a decir: que cuanto más sencillo mejor. Este principio es equivalente a la navaja de Ockham[1] y se ha popularizado en el mundo de la programación. Por eso ,a la hora de diseñar el proceso de adopción individual de los agentes, en vez, de optar por un modelo de estados que se ajusta a la teoría de Rogers y también a lo estudiado en el Apartado 2.3, hemos optado por adaptar el modelo de adopción colectivo del modelo de Ahmadian a cada uno de los agentes. Esta simplificación nos va a permitir un ahorro computacional significativo, no carente de una base teórica solida por detrás, ya que como veremos cuando analicemos cómo se han implementado los clientes potenciales, se ha intentado individualizar la q y p del modelo de Bass en cada agente.

Tortugas

Existen dos tipos de agentes: los clientes potenciales y los seguidores. Cada grupo tiene sus propios atributos, pero todo agente tiene en común dos atributos, ***homex*** y ***homey,*** que sirven para almacenar su hogar bas:; la posición inicial, de donde nunca se alejan más de una distancia establecida por el parámetro **homed**[2]. Como veremos más adelante, estas variables, junto a la distancia máxima, nos permitirá simular una red social para cada agente siguiendo la teoría de Hamill & Gilbert (2009) .

Clientes potenciales

1 Principio de economía o también conocido como principio de parsimonia.
2 Configurable mediante deslizable en la ventana principal.

Los clientes potenciales ***tpotentials*** representan a personas que no han adaptado la innovación. Los clientes potenciales dispondrán de dos variables, ***legitimacy-wom*** y ***legitimacy-marketing,*** que se utilizarán para acumular la legitimidad de la innovación, como sustitutos de la variable de nivel ***stock de legitimidad*** del modelo de Dinámica de Sistemas, estas variables son el equivalente individual de las variables p y q del modelo de Bass. Cuando la suma de ambas variables alcance el valor de la constante ***adoption-limit,*** el cliente adquirirá la innovación y pasará a ser un ***tadopter***.

Siguiendo la teoría de la difusión de Rogers, expuesta en el Apartado 4.2, los clientes potenciales podrán pertenecer a una de las siguientes categorías: innovadores, primeros seguidores, mayoría precoz, mayoría tardía y rezagados. Cada una de estas categorías representa un porcentaje de la población inicial de clientes potenciales y tendrán diferentes valores en los atributos ***legitimidad-bab*** y ***legitimidad-marketing*** para plasmar las diferentes categorías de propensión al cambio descrito por la teoría.

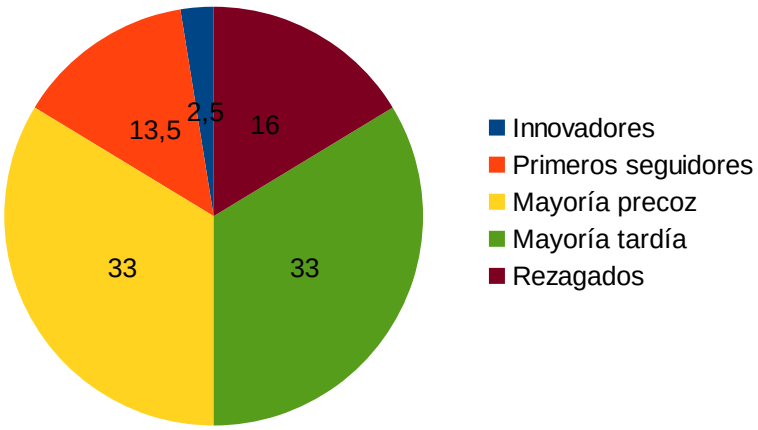


Figura 5.3: Tipos de seguidores y peso en el modelo

En el modelo que hemos estudiado, podemos apreciar que la ecuación de adopción contiene dos variables que configuran la adopción en cada paso de simulación:

$$R_a = (\delta_E + \delta_L) \times N_p \tag{5.2}$$

δ_L *Tasa de crecimiento de la legitimidad*

δ_E *Fracción de seguidores tempranos*

En el modelo basado en agentes, la decisión de adquirir la innovación se adopta cuando se alcanza el valor ***adoptionlevel;*** sin embargo, para no alterar demasiado el modelo original, y ya que en cierta medida, la fracción de seguidores tempranos puede considerarse una categoría de la teoría de la difusión de Rogers, mantendremos en el modelo esta asunción. En cada paso existirá una parte de la población que independientemente adoptará la innovación.

```
let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
```

Código 5.2: Fracción de seguidores tempranos

Seguidores

Los seguidores ***tadopters*** representan a personas que han adaptado la innovación. Solo tendrán una variable llamada ***duration,***que indica el tiempo que se lleva usando la innovación y que servirá para hacer que el seguidor vuelva a adoptar el producto cuando acabe la vida útil de éste.

Cuando la variable ***duration*** sea igual *a* ***Technology_life_time***, el seguidor volverá a ser un *tadopter,* pero en este caso, al haber sido usuario de la innovación, le costará menos adoptar la innovación y tendrá unos valores elevados en las variables ***legitimacy-wom*** y ***legitimacy-marketing***.

Las baldosas

Las baldosas por su parte, podrán disponer de soportes publicitarios, los agentes que se acerquen recibirán un impacto publicitario que aumentará la legitimidad de la innovación. Las baldosas

dispondrán de una variable ***advertising*** que permitirá indicar si disponen de capacidad publicitaria; inicialmente se establece que el 10% de las baldosas dispongan de soporte publicitario.

Interacciones

En este modelo existen dos tipos de interacciones:

- Cuando un agente se encuentra en el vecindario a un agente que ha adoptado la innovación, su legitimidad por efecto boca a boca se incrementará.
- Cuando un agente se encuentre próximo a una baldosa publicitaria su atributo de ***legitimacy-marketing*** se incrementará.

5.5.2 Comportamiento de los agentes

Con cada *tick* los agentes realizarán las siguientes acciones:

1. Se girarán aleatoriamente y se desplazarán una baldosa hacia adelante, excepto si la distancia a su hogar base es mayor que el parámetro ***homed;*** en este caso, el agente se girará hacia su hogar base y se desplazará una baldosa.
2. En caso de encontrarse a una distancia de uno de una baldosa con soporte publicitario, los agentes, recibirán un contacto publicitario aumentando el nivel de su variable ***legitimacy-marketing***.
3. En caso de encontrar en su perímetro a un agente que ya haya adoptado la tecnología ***tadopter,*** los agentes "intercambiarán opiniones" y aumentará el nivel de su variable ***legitimacy-wom***.
4. Cuando la suma de las variables ***legitimacy-wom*** y ***legitimacy-marketing*** alcance un determinado nivel, el agente pasará a ser un seguidor de la tecnología ***tadopter***.
5. Los agentes se moverán por un espacio toroidal que contenga una concentración de agentes adecuada para que el modelo se comporte como el modelo de Dinámica de Sistemas, se ha estimado que un tamaño de mundo 111x111 es adecuado.
6. Los seguidores ***tadopters*** aumentarán la variable ***duration*** un ***dt*** valor del paso de integración.

Al iniciar la simulación, todos los agentes se colocan aleatoriamente y cada agente considera esta ubicación inicial su hogar base. El propósito de la regla número 1 es que los agentes puedan moverse en círculos, pero delimitándose a un área cercana a su hogar base. Este movimiento en círculos pretende simular la difusión en redes sociales generadas en función de la vecindad.

Existen numerosas teorías a la hora de representar de una manera realista una red social, pero para simular de una manera simple la difusión en un espacio acotado, hemos seguido un modelo de relaciones equivalente a los "círculos sociales" de Hamill y Gilbert (2009). Con esta regla no se tienen en cuenta las posibles relaciones a larga distancia que pueden existir y que cada vez, gracias a Internet, tienen más peso. Algunos autores proponen crear enlaces entre los agentes para dar cabida a este tipo de relaciones, en nuestro caso hemos preferido la sencillez (KISS).

5.6 Descripción detallada del modelo en NetLogo

Como todo programa de NetLogo, en el procedimiento ***setup*** inicializamos el entorno

```
to setup
 ca
 setup-globals
 system-dynamics-setup
 setup-agents
 setup-sliders
end
```

Código 5.3: Procedimiento de inicialización

1. Inicializamos las variables globales.
2. Inicializamos el modelo de Dinámica de Sistemas.
3. Inicializamos los *sliders*.
4. Inicializamos los agentes.

El proceso de inicialización de agentes ***setup-agents*** realiza las siguientes tareas:

1. Crea las baldosas que van a tener soporte publicitario.

```
ask patches [ set advertising false]
ask n-of ( Population * 0.10 ) patches [
  set pcolor green
  set advertising true
  set intensity random-float 2
]
```

Código 5.4: Creación de soportes publicitarios

2. Creamos los ***tadopters*** iniciales.

```
ask n-of Adopters patches [
  sprout-tadopters 1
  [
    set color red
    set homex xcor
    set homey ycor
  ]
]
```

Código 5.5: Creación se seguidores iniciales

3. Creamos los clientes potenciales siguiendo las proporciones e la teoría de Rogers, mostrado en el Apartado 5.5.1

```
create-tpotentials ( Population - Adopters )
        [
           set color sky
             setxy random-xcor random-ycor
             set homex xcor
             set homey ycor
             set class "none"
         ]
         ask n-of ( Potential_adopters * 0.135 ) tpotentials
         [
          set legitimacy-wom 0.54
          set legitimacy-marketing 0.5
          set class "primeros"
         ]
         ask n-of ( Potential_adopters * 0.33) tpotentials with [class = "none" ]
         [
          set legitimacy-wom random adoptionlevel * 57 / 100
          set legitimacy-marketing 0.3
          set class "precoces"
         ]
         ask n-of ( Potential_adopters * 0.33 ) tpotentials with [class = "none" ]
         [
           set legitimacy-wom -0.3
          set legitimacy-marketing -0.2
          set class "tardios"
         ]
         ask n-of ( Potential_adopters * 0.16 ) tpotentials with [class = "none" ]
         [
          set legitimacy-wom -1.1
          set legitimacy-marketing -1.1
          set class "rezagados"
         ]
```

Código 5.6: Creación de clientes potenciales

Una vez inicializado el modelo, comienza el ciclo de iteraciones, en cada paso se realizan la siguientes tareas recogidas en el procedimiento ***go***:

```
        to go
         if ticks >= 25 [ stop ]
         agent-go
         set-current-plot "Adoption"
         system-dynamics-go
         system-dynamics-do-plot
         graph
        end
```

Código 5.7: Paso del ciclo de iteraciones

1 Ejecutamos un paso del modelo de Dinámica de Aistemas, llamando al procedimiento ***system-dynamics-go***.

2 Actualizamos las gráficas llamando a ***system-dynamics-do-plot*** y ***graph***.

```
to graph
 set-current-plot "Rate" plot Adoption_rate * ( 1 / dt )
 set-current-plot "Cumulative_production" plot Cumulative_production
 set-current-plot "Total_cost" plot Total_cost
 set-current-plot "Cost_price" plot Cost_price
 set-current-plot "Price" plot Price
 set-current-plot "Marketing_performance" plot Marketing_performance
 set-current-plot "Profit" plot Profit
 set-current-plot "Investment_on_marketing" plot Investment_on_marketing
 set-current-plot "Revenue" plot Revenue
 set-current-plot "Performance_improvement" plot Performance_improvement
 set-current-plot "Atractiveness_from_price " plot Atractiveness_from_price
       set-current-plot            "Total_attractiveness_of_technology"            plot
Total_attractiveness_of_technology
 set-current-plot "Rate_of_growth_of_legitimacy" plot Rate_of_growth_of_legitimacy
           set-current-plot           "Growth_of_legitimacy_through_wom"           plot
Growth_of_legitimacy_through_wom
          set-current-plot         "Ratio_of_adopters_to_population_alpha"         plot
Ratio_of_adopters_to_population_alpha
 set-current-plot "Wom_effectiveness" plot Wom_effectiveness
 escribe-csv
end
```

Código 5.8: Procedimiento de actualización de gráficas

3 Ejecutamos un paso en el modelo multiagente, todas las funciones y tareas se han recogido en un procedimiento llamando ***agent-go***.

```
       to agent-go
         move-turtles
         marketing
         contact
         amortizate
         discard
       end
```

Código 5.9: Paso del modelo multiagente

4 Como podemos ver en el código del procedimiento ***agent-go,*** cada paso en el modelo multiagente se realiza:

4.1 Las reglas de movimiento de las tortugas recogidos en el procedimiento

move-turtles pretenden crear un "círculo social".

```
to move-turtles
 ask turtles [
        ifelse abs(xcor - homex) < homed or abs(ycor - homey) < homed
        [
            right random 360
        ]
        [
            facexy homex homey
        ]
        forward 1
 ]
end
```

Código 5.10: Reglas de movimiento de los agentes

4.2 Los agentes que se encuentran cerca de una baldosa "soporte" publicitario reciben un impacto, el procedimiento encargado de esta acción es ***marketing***.

```
to marketing
 ask patches with [ advertising = true ]
 [
         let pat-intensity [ intensity ] of self
         ask tpotentials in-radius 1
         [
      set   legitimacy-marketing   (   legitimacy-marketing   +   pat-intensity   *
       Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology ) * dt )
         ]
 ]
end
```

Código 5.11: Procedimiento de impacto publicitario

4.3 Los agentes que han adaptado la tecnología ***tadopters***, y que tengan en su vecindario a algún agente que no la ha adoptado (tpotential), provocarán un efecto boca a boca aumentando el valor de la variable ***wom-legitimacy***.

```
        let contactos tpotentials with [ any? neighbors with [ any? tadopters-here ] ]
         ask contactos [
          set legitimacy-wom (legitimacy-wom + Wom_effectiveness * ( 1 +
         Total_attractiveness_of_technology ) * dt)
         ]
        let convertidos count contactos  with [ legitimacy-wom + legitimacy-
        marketing > adoptionlevel ]
    ask contactos with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
        [
           set breed tadopters set color red
        ]
        ;Enviamos valores al modelo SD
        set Adoption_rate convertidos + tempranos
        set Adopters count tadopters
```

Código 5.12: Efecto boca a boca

4.4 Como el modelo original recoge, una parte de los seguidores ***tadopters*** descarta la tecnología y pasa a ser ***tpotential*** a una velocidad marcada por la vida útil de la tecnología.

```
to discard
  let cuantos n-of ( ( count tadopters) / ( Technology_life_time * ( 1 / dt ) ))
tadopters
 let discartes cuantos
 ask discartes
 [
  set breed tpotentials
  set color sky
  set legitimacy-wom random adoptionlevel * 99 / 100
  set class "repetidores"
 ]
 set Discar_rate count discartes
 set Potential_Adopters count tpotentials
end
```

Código 5.13: Procedimiento de fin de vida útil

5.7 Validación del modelo híbrido frente al de Dinámica de sistemas

5.7.1 Metodología

La validación mide el grado en el que el modelo de simulación y sus datos asociados reproducen el fenómeno que se va a simular, desde la perspectiva del uso para el que fue destinado el

modelo. Al contrario de la verificación del modelo, la validación responde a la pregunta ¿hemos construido el modelo correcto?

Al tratarse de un modelo multimétodo, no podemos aplicar las metodologías de validación aplicadas en la Dinámica de Sistemas, por este motivo vamos a utilizar un método de validación de propósito general ideado por Naylor y Finger (1967) que es ampliamente utilizado en el mundo de la simulación.

Los tres pasos para validar el modelo son los que ilustramos en la siguiente figura junto a las técnicas utilizadas :

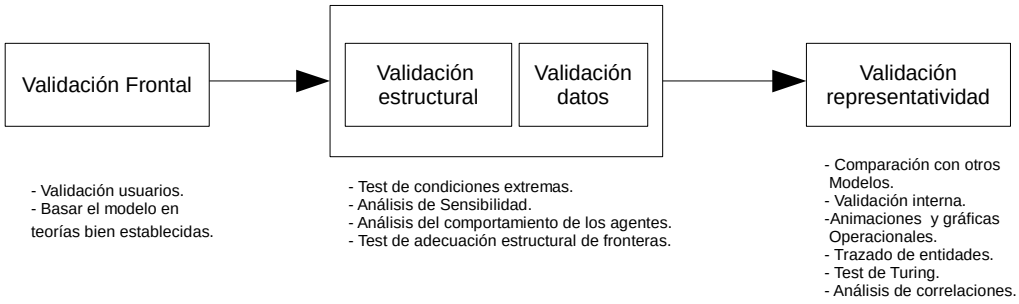


Figura 5.4: Pasos de validación y técnicas utilizadas

1. **Validación frontal.**

   Consiste en preguntar a conocedores del sistema si el modelo o su comportamiento son razonables para el propósito para el que fue definido. Para conseguir este objetivo, conviene que desde el primer momento las teorías estén bien establecidas.

2. **Validar las Asunciones del modelo.**

   Las asunciones hechas sobre el modelo pueden ser de dos categorías: asunciones estructurales y asunciones sobre los datos.

   La validación estructural es un proceso multidimensional donde entran en juego problemas de representación, relaciones causales y matemáticas, y que consiste en identificar las estructuras

responsables del correcto comportamiento.

La validación de los datos se refiere, a si, por ejemplo, de los datos empíricos inferimos un determinado tipo de comportamiento para un proceso y ajustamos una distribución teórica a dichos datos, verificar si el ajuste es correcto o no.

Un paso complementario de este proceso es realizar un análisis de sensibilidad de los parámetros del modelo.

3. **Determinar cuán representativos son los resultados de la simulación**.

Consiste en determinar cuán representativos son los resultados de la simulación con los datos del sistema simulado. En el caso de sistemas existentes, este paso se puede considerar como la validación empírica.

5.7.2 Determinación del alcance de la validación

En toda validación, el primer paso consiste en definir cuál es el propósito del modelo propuesto, pues esta cuestión es determinante a la hora de establecer los mecanismos de validación.

En nuestro caso, el propósito del modelo es: ilustrar cómo implementar un modelo de la difusión de la innovación integrándolo con un sistema multiagente y sistemas de información geográfica.

Partimos de un modelo teórico, realizado con la metodología de Dinámica de Sistemas que asumimos como válido. En función del ámbito de aplicación de las técnicas de validación, se aplicarán al modelo en su conjunto o a las partes modificadas.

5.7.3 Validación Frontal

Al haber realizado un modelo con un propósito ilustrativo y no simular un sistema real, no disponemos de usuarios que lo vayan a utilizar con los que discutir sobre la validez del modelo.

Por el contrario, sí podemos establecer la validez acerca de las teorías que hemos utilizado en la modificación realizada al modelo original.

La composición de los agentes sigue las categorías definidas por la Teoría de la Innovación de (Rogers 1962) analizadas en el Apartado 4.2. Consideramos que esta teoría es ampliamente aceptada.

En cuanto al efecto boca a boca es una técnica de *marketing* totalmente válida y que, gracias al auge de Internet, empieza a utilizarse cada vez más dando pie al *marketing* viral. Creemos que la teoría es válida ver (Kotler & Keller 2009), si bien en la siguiente sección discutiremos la validez de las asunciones que hemos realizado siguiendo esta teoría.

Las variables ***legitimacy-wom*** y ***legitimacy-marketing*** que determinan la adopción de la innovación, se inspiran en la p (coeficiente de innovación) y la q (coeficiente de imitación) del modelo de Bass analizadas en el Apartado Error: no se encuentra la fuente de referencia.

5.7.4 Validación de las asunciones

Una de las principales asunciones que hemos hecho a la hora de realizar la descomposición del modelo es que las variables auxiliares del modelo de Dinámica de Sistemas **crecimiento de la legitimidad por EBB** y **de la legitimidad por marketing** han pasado a ser individuales para cada agente.

La ecuación de la adopción deja de ser un producto sobre la población, y pasa a ser una suma de atributos de cada agente (***legitimacy-wom*** y ***legitimacy-marketing****),* que cuando alcanza el valor del parámetro ***adoptionlevel*** convierte al agente en seguidor de la innovación.

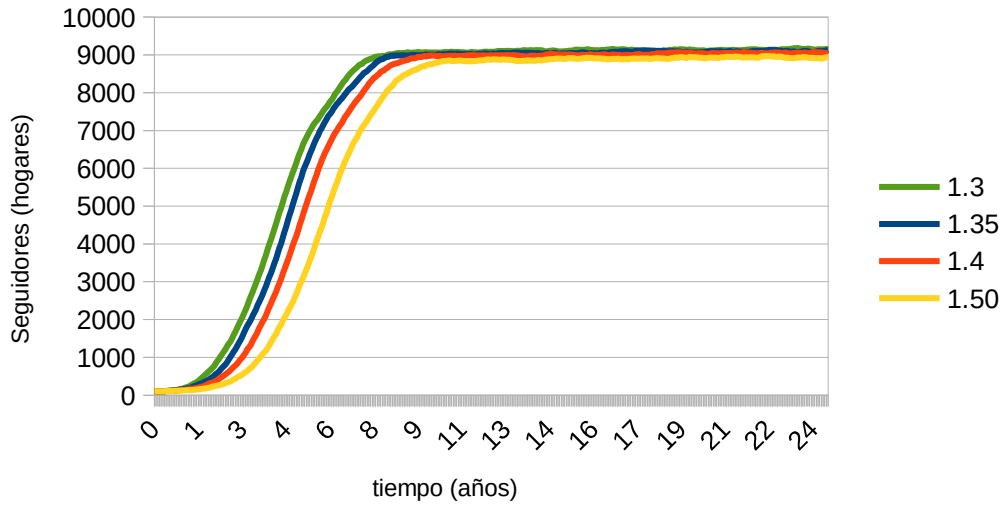


Figura 5.5: Análisis de sensibilidad del parámetro adoptionlevel.

En cuanto al espacio de agentes NetLogo ofrece un espacio Discreto 2D hemos establecido una densidad de 10.000 agentes por 12.321 celdas. Al inicio se producen 630 contactos por paso y en la parte final tenemos una tasa de contactos de 980.

Debido a las reglas de movimiento e interacción impuestas a los agentes, la variable adopción es una variable estocástica. Para determinar su validez, ver que no hay una gran variabilidad, hemos realizado un test de validez interna.

5.7.5 Validación de la representatividad de los resultados del modelo.

Al contrario que la Dinámica de Sistemas, los modelos multiagentes suelen utilizarse para analizar escenarios no observables, como, por ejemplo, la implementación de políticas hipotéticas o nuevas tecnologías. “La naturaleza de los modelos multiagentes provoca que no existen datos reales para validarlos” (Bert et al. 2014).

Nuestro modelo, al ser una reimplementación de un modelo teórico de difusión de la innovación, no puede ser comparado con datos reales. Pero sí podemos compararlo con el modelo teórico de Dinámica de Sistemas del que partimos. A continuación mostramos una comparativa entre los dos modelos parametrizados de la siguiente manera:

Parámetro	Valor
S_progress_ratio	0,8
S_capacity_utilization	1
S_Ratio_if_target_price_to_cost_price	1,07
adoptionlevel	1,35
Homed	3

Tabla 5.2: Parámetros de simulación

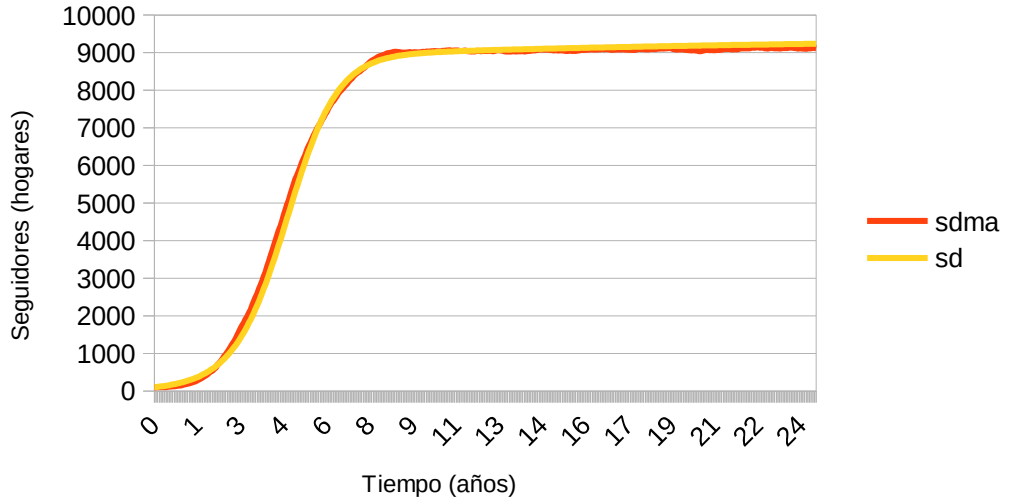


Figura 5.6: Adopción modelo SD y SDMA

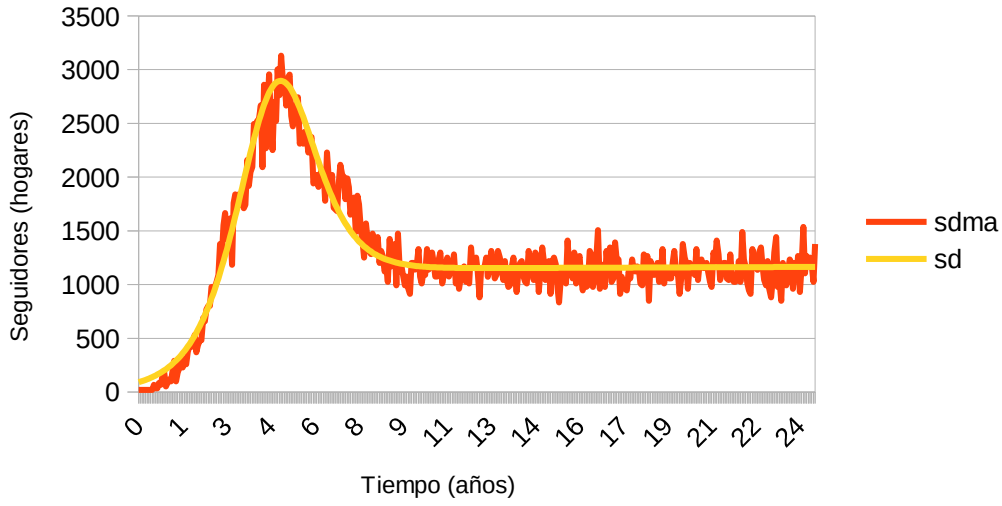


Figura 5.7: Tasa de adopción modelo SD y SDMA

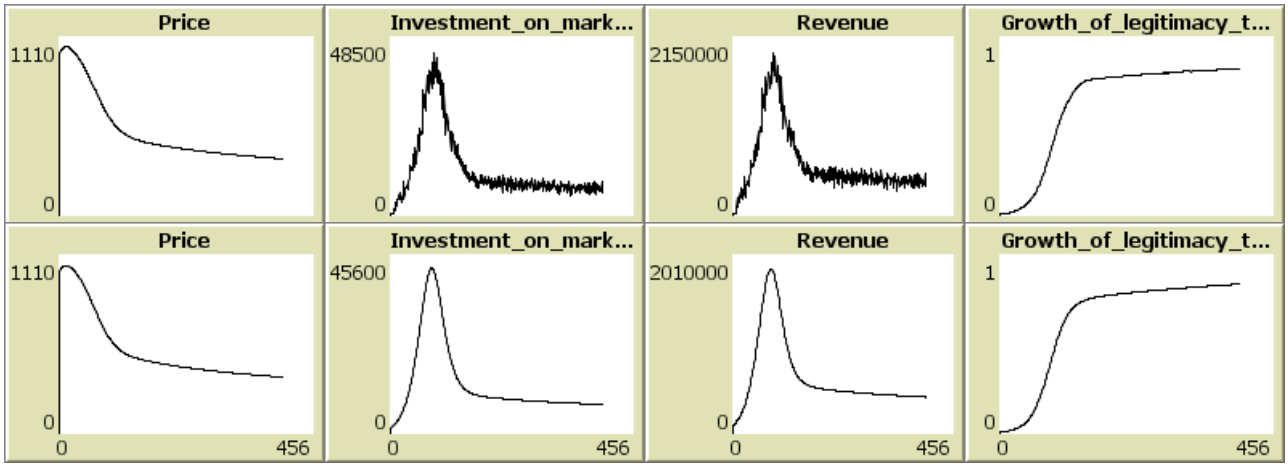


Figura 5.8: Comparación variables modelo DS/MA

Si realizamos un análisis de la correlación de ambos modelos, obtenemos que el coeficiente de correlación de Pearson nos da un valor de 0,996 y Spearman 0,95. Como vemos, ambos modelos están fuertemente correlacionados y confirman la validez del modelo híbrido.

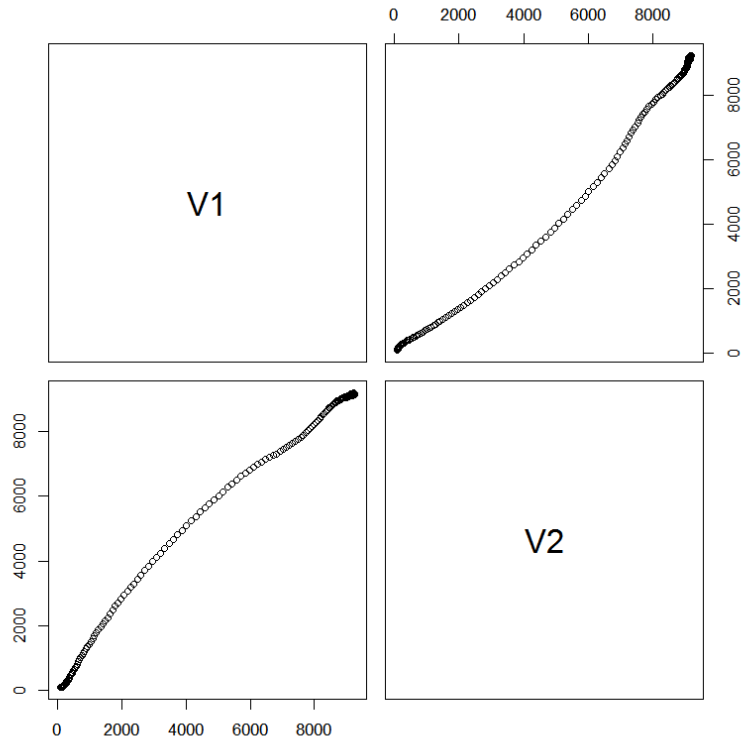


Figura 5.9: Gráfico de dispersión para modelos SD /SDMA

5.8 Conclusiones

Guiados por la teoría de la difusión de Everett, hemos propuesto una descomposición del modelo del capítulo anterior en dos niveles. Está descomposición, ha supuesto tener que proponer varias asunciones e hipótesis referentes al comportamiento de los agentes y tener que programar desde cero el nivel multiagente. También hemos propuesto una forma de bucle de realimentación multinivel al obtener *feed-backs* desde ambos niveles.

La descomposición y programación del modelo en NetLogo no ha resultado muy costosa y podemos concluir que las características de entorno multinivel del NetLogo son directas y adecuadas.

El modelo resultante se comporta de una manera similar al modelo de Dinámica de Sistemas como hemos podido validar empíricamente, y ofrece la posibilidad de añadir a través del modelo multiagente una dimensión espacial al proceso de difusión de la innovación que analizaremos en el siguiente capítulo.

6 Inclusión de información SIG

6.1 Introducción

Uno de los objetivos de este proyecto es explicar el proceso de introducción de datos provenientes de sistemas de información geográfica en NetLogo. Vamos a introducir unos “shapes” procedentes del CNIG Centro Nacional de Información Geográfica (www.cnig.es) utilizando la extensión GIS de NetLogo, que analizamos en el punto 3.8.3, y que han sido retocados usando la herramienta que presentamos en el punto 3.8 QGIS Valmiera, para adaptarla a los propósitos de nuestro modelo de difusión de la innovación, que no son otros que intentar reproducir el modelo de difusión previamente analizado acotado en un ámbito geográfico.

Es posible que las hipótesis y escalas utilizadas en el modelo híbrido no se ajusten a la cartografía seleccionada (España) y es que hemos puesto el modelo anterior con una población de 10.000 seguidores potenciales dispersos en un país de 45 millones, no pretendíamos validar este apartado ya que dicha validación quedaba fuera del alcance del proyecto. El objetivo es explicar cómo se puede integrar información geográfica en un modelo SDMA elaborado con NetLogo.

Aun así, hemos intentado realizar un modelo lo más real posible y nos hemos encontrado con dificultades *hardware*, al no disponer de datos de difusión de una innovación reales que se ajusten a nuestra escala. Aunque sí existen datos reales para explicar procesos de difusión de la innovación a otras escalas, no disponemos de los recursos *hardware* para poder trabajar con ellos, y es que en el desarrollo de esta parte del modelo, nos hemos encontrado con limitaciones de memoria. Al intentar incrementar la escala de nuestro modelo, la máquina virtual de Java ha llegado a su límite (estándar) más de 1,3 GB en varias ocasiones. El modelo con 10.000 agentes que presentamos puede llegar a consumir 820 MB.

También es cierto que no hemos realizado ninguna optimización y hemos trabajado con un exceso de datos innecesarios para nuestro propósito.

6.2 Características del Shape utilizado

El *shape* que vamos a utilizar contiene los límites municipales del Reino de España, a los cuales les hemos asociado siguiendo los procedimientos explicados en el Apartado Trabajando con Shapes, otros campos provenientes de bases de datos externas.

El *shape* original en formato ESRI, *polígonos_Municipios,* obtenido de la web del Centro Nacional de Información Geográfica, contiene una capa con 8.111 polígonos que acotan los límites municipales, usando el sistema de referencia geodésico ED50 (European Datum 1950).

Además de los polígonos el fichero posee los siguientes atributos:

1. id: Identificador único.
2. nombre: Nombre oficial del municipio o de la comunidad conjurisdiccional.
3. CodigoINE: Código del Instituto Nacional de Estadística.
4. Jurisdicci: códigos INE de los municipios que tienen jurisdicción sobre este territorio. Coincide con el CodigoINE excepto en el caso de las comunidades con jurisdiccionales, en donde aparecen separados por punto y coma.
5. provincia: código INE de la provincia a la que pertenece el municipio.

Partiendo de este *shape* y siguiendo las técnicas mostradas en el Apartado 3.8, se han añadido atributos mediante el cruce con bases de datos externas y cálculos de áreas hasta disponer de un fichero ESRI, al que le hemos añadido los siguientes atributos[1] .

1. Área. Superficie del polígono en hectáreas, en el atributo MUNICIPI_6.
2. Población. Población del polígono, en el atributo MUNICIPI_5.

Con estos atributos podremos obtener la densidad de población de los municipios y posicionar agentes en función de la densidad de cada polígono.

6.3 Importación del Shape

Vamos a ver paso a paso, como vimos en la sección 3.8.3, cómo integrar la información del

1 Solo se describen los atributos utilizados en NetLogo

shape en nuestro modelo de NetLogo.

El primer paso es cargar la extensión GIS en NetLogo;

```
extensions [ gis ]
```

Después definimos la transformación

```
 gis:load-coordinate-system "resultado.prj"
```

Cargamos los *datasets*

```
set distritos gis:load-dataset "resultado.shp"
```

Una vez cargado el *dataset* lo visualizamos;

```
gis:set-drawing-color white
gis:draw distritos 1
```

Realizamos la intersección

```
gis:apply-coverage distritos "MUNICIPI_5" poblacion
gis:apply-coverage distritos "MUNICIPI_6" area
```

Código 6.1: Código importación información GIS

En el ejemplo anterior, asociamos el atributo MUNICIPI_5 del *shape* (la población) al atributo de la baldosa población.

6.4 Modificación del modelo híbrido

Ahora que disponemos de información sobre densidad poblacional, lindes y fronteras. Tenemos que modificar el comportamiento de los agentes para que:

- Los agentes no salgan de las fronteras al desplazarse.
- Las zonas con mayor densidad tengan una mayor presencia de agentes que las zonas desérticas.
- Los soportes publicitarios se coloquen dentro de las fronteras y en zonas pobladas.
- El número de celdas dentro de las fronteras sean similares al número de celdas en el modelo híbrido.

Modificación de la reglas de Desplazamiento

Como hemos señalado, los agentes tienen que respetar las fronteras y tienen que tener una tendencia a mantenerse en las zonas más pobladas.

Para restringir el movimiento de los agentes a los límites del *shape* importado, utilizaremos el atributo población de las baldosas. En el caso de que la población sea igual a cero, sabremos que la baldosa es externa al *shape*, con lo que evitaremos desplazarnos a estas baldosas.

La regla de desplazamiento se adapta para tener en cuenta esta nueva limitación, quedando de la siguiente manera:

```
to move-turtles
 ask turtles[
  ifelse abs(xcor - homex) < homed or abs(ycor - homey) < homed
  [
    right random 360
  ]
  [
    facexy homex homey
  ]
  let poblacionactual [ poblacion ] of patch-here
  let enfrente [ poblacion ] of patch-ahead 1
  if (is-number? poblacion) and (is-number? enfrente) [
   if enfrente > 0
   [
      forward 1

   ]
  ]
 ]
end
```

Código 6.2: Modificación reglas de movimiento

Posicionamiento de Agentes y de Soportes publicitarios

En el modelo híbrido, posicionábamos los agentes de un modo aleatorio y uniforme. En este supuesto, lo que nos interesa es que los agentes se distribuyan por las zonas más densamente pobladas, manteniendo las proporciones respecto a la población. Para ello modificamos el procedimiento **setup-agents** para permitir una distribución en función de la densidad añadiendo el siguiente fragmento:

```
ask patches[
  if (is-number? poblacion ) [
   if ( poblacion > 0 and area > 0)[
   let numagentes ( densidad * area-paches / Population )
   sprout-tpotentials (floor numagentes ) [ customiza-tpotentials]
   ;print numagentes
   let decimal ( numagentes - (floor numagentes))
   if (decimal > random-float 1) [sprout-tpotentials 1 [ customiza-tpotentials ]]
   ]
  ]
 ]
```

Código 6.3: Procedimiento para distribuir agentes en celdas en función de la densidad de población

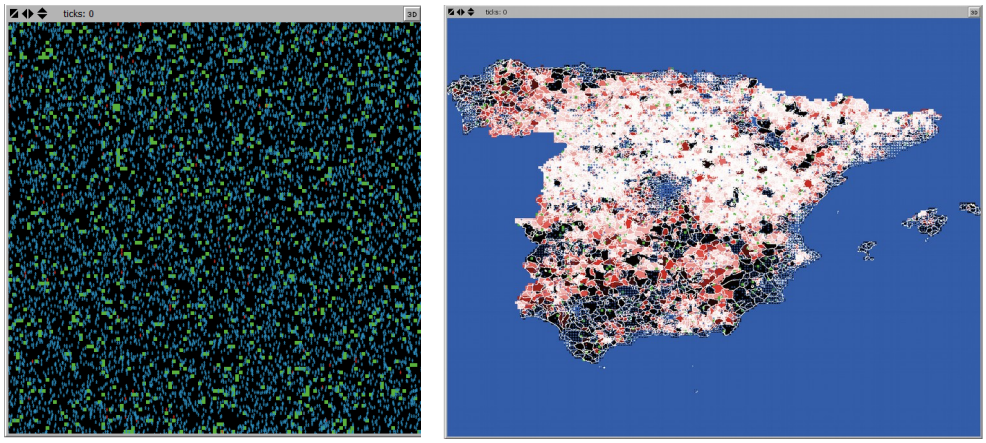


Figura 6.1: Comparación de posicionamiento inicial en modelo GIS

En la Figura 6.1 observamos como los agentes se han posicionado en las zonas más pobladas; destacan Madrid, Barcelona y la zona de Levante.

Esta manera de posicionar los agentes implica una forma diferente de crear los agentes a la vista en el capítulo anterior. En este caso la creación se realiza no de una manera global, como veíamos

en el código 5.6 para todo el mundo, si no celda a celda.

La creación se realiza en la línea siguiente

```
sprout-tpotentials 1 [ customiza-tpotentials ]
```

Se crea un agente del tipo tpotential y sus propiedades se establecen ejecutando el procedimiento ***customiza-tpotentials*.** Las propiedades que tenemos que inicializar por cada agente, no son otras más que el tipo de cliente potencial, sus variables ***legitimacy-wom*** y /***egitimacy-marketing*** y las coordenadas de su lugar de vivienda habitual ***homex, homey,*** respetando los porcentajes entre los diferentes tipos de seguidores establecidos en el anterior capítulo (según la teoría de Rogers).

```
to customiza-tpotentials
 set size 1
 set color blue
 set homex xcor
 set homey ycor
 let class-prob (random 1000)
 ifelse (class-prob < 136)
 [
  set legitimacy-wom 0.54
  set legitimacy-marketing 0.5
  set class "primeros"
 ]
 [
  ifelse (class-prob >= 135 and class-prob < 465 )
  [
   set legitimacy-wom random adoptionlevel * 57 / 100
   set legitimacy-marketing 0.3
   set class "precoces"
  ]
  [
   ifelse (class-prob >= 465 and class-prob <= 795 )
   [
    set legitimacy-wom -0.3
    set legitimacy-marketing -0.2
    set class "tardios"
   ]
   [
    if (class-prob >= 840 )
    [
      set legitimacy-wom -1.1
      set legitimacy-marketing -1.1
      set class "rezagados"
    ]


   ]
  ]

 ]
end
```

Código 6.4: Procedimiento de inicialización de agentes modelo GIS

Para maximizar la eficacia publicitaria de las baldosas sobre la población de clientes potenciales, las baldosas se han distribuido de un forma aleatoria en aquellas baldosas con una densidad poblacional mayor al parámetro ***publicity_density***.

Densidad de Celdas

Teniendo en cuenta que el modelo anterior de dimensión 111x111 contenía 12.321 baldosas, se ha aumentado el tamaño a 205x161 para que las baldosas contenidas dentro del *shape*, la península quitando el mar, fueran 12.305, el número de baldosas más aproximado al modelo anterior, es decir, una densidad de agentes por celda de 0,81.

Patches altamente densificados

Aunque el número de celdas es similar en el modelo híbrido como en el modelo con GIS, en el segundo modelo las poblaciones de agentes se concentran en determinadas baldosas, capitales de provincia y grandes metrópolis. Esta concentración provoca que en las citadas áreas el número de contactos por el efecto boca a boca se dispare. Para corregir este problema, establecemos una nueva regla que impide que un agente tenga más de un número determinado de contactos por paso, y queda de la siguiente manera:

Se han impuesto dos límites:

1. Un cliente potencial ***tpotential*** se contactará a lo sumo una vez por *tick*.
2. Cada seguidor ***tadopter*** solo contactará como máximo con 10 clientes potenciales que se encuentren en su vecindario.

```
to contact
 let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
 let contactos tadopters with [ any? neighbors with [ any? tpotentials-here ] ]
 ask contactos [

  ifelse count [tpotentials-on neighbors] of self > 10 [
   ask [n-of 10 tpotentials-on neighbors ] of self [
    if ( condt < 2 )[
              set   legitimacy-wom   (legitimacy-wom   +   Wom_effectiveness   *   (   1   +
Total_attractiveness_of_technology ) * dt)
     set condt condt + 1
    ]
   ]
  ]
  [
   ask [ tpotentials-on neighbors ] of self [
    if ( condt < 1 )[
              set   legitimacy-wom   (legitimacy-wom   +   Wom_effectiveness   *   (   1   +
Total_attractiveness_of_technology ) * dt)
     set condt condt + 1
    ]
   ]

  ]
  ]
 ;Calculo un valor de una variable SD suma de agentes
  let convertidos count tpotentials   with [ legitimacy-wom + legitimacy-marketing >
adoptionlevel ]
 ask tpotentials with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
 [
   set breed tadopters set color red
 ]
 ;Enviamos valores al modelo SD
 set Adoption_rate convertidos ;+ tempranos
 set Adopters count tadopters
 ;Reseteo condt
 ask tpotentials
 [
  set condt 0
 ]
 end
```

Código 6.5 Regla de contacto

6.5 Validación

Como hemos detallado en el aparado anterior, las principales modificaciones que hemos realizado al modelo del Capítulo 5 son la aparición de fronteras y la distribución de población en función de la densidad.

La validación de estas modificaciones es relativamente sencilla y la podemos realizar de una forma visual.

En el caso de la distribución de la población así como el de los soportes publicitarios, solamente hace falta comprobar el resultado de la carga inicial de agentes y soportes (botón setup), que como mostrábamos en la Figura 6.1, corresponde a la distribución esperada; los agentes no se posicionan inicialmente en el mar

En cuanto a la modificación de las reglas de movimiento para que los agentes no sobrepasen las fronteras, las reglas son claras y, mediante la simulación del modelo, damos fe de que los agentes nunca sobrepasan las fronteras tal, y como era de esperar.

Figura 6.2: Validación perímetros de fronteras

Para validar el modelo multiagente con restricciones geográficas, vamos a comparar los resultado de la simulación de este modelo con los resultados obtenidos con el modelo teórico de

Dinámica de Sistemas del que partimos.

La simulación se ha realizado con los siguientes parámetros:

Parámetro	**Valor**
S_progress_ratio	0,8
S_capacity_utilization	1
S_Ratio_if_target_price_to_cost_price	1,07
Adoptionlevel	1,35
Homed	3
Publicity_density	0,05

Tabla 6.1: Parámetros de simulación

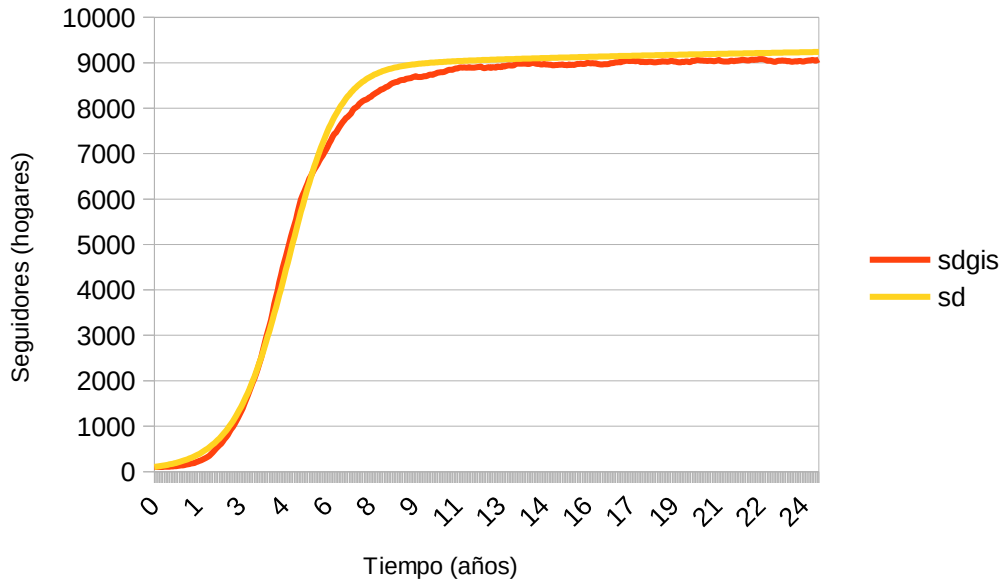


Figura 6.3: Adopción modelo SD y SDGIS

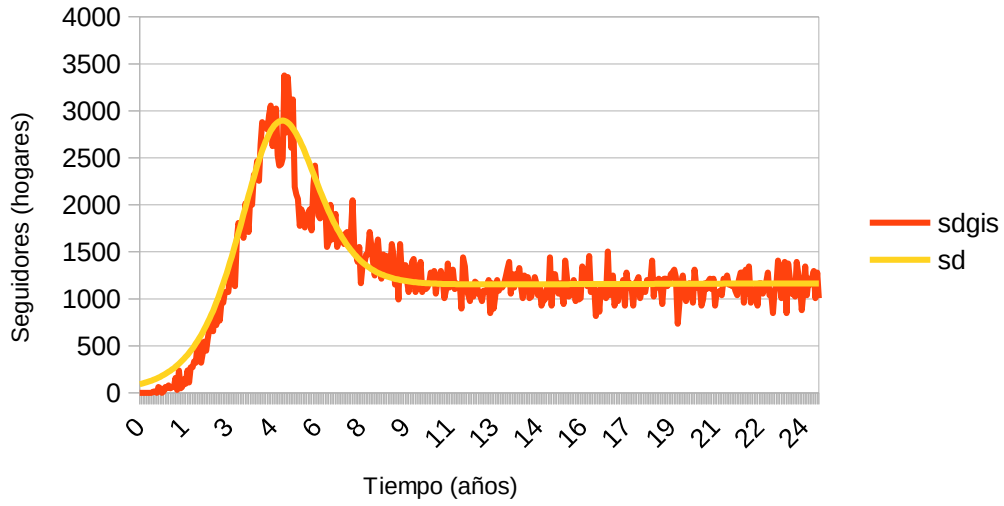


Figura 6.4: Tasa de adopción modelo SD y SDGIS

Como observamos, el modelo es más oscilante y aleatorio que el modelo sdma; queda más claro en la siguiente figura donde representamos los resultados de varias simulaciones realizadas con los mismos parámetros.

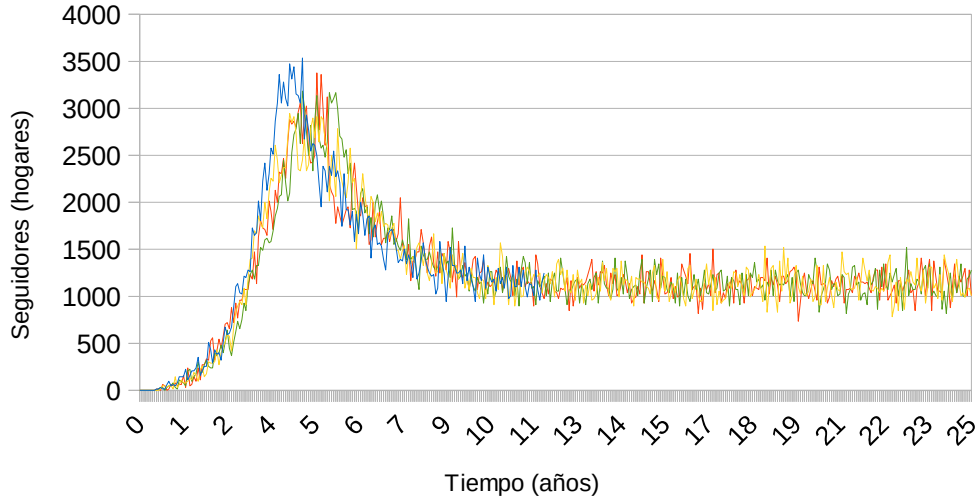


Figura 6.5: Resultado de varias simulaciones con idénticos parámetros

Esta variabilidad depende de la ubicación inicial de los seguidores tempranos cuya dispersión o concentración determinan el comportamiento de la adopción. Para contrastar esta observación, hemos realizado una serie de experimentos en casos frontera del modelo y hemos comparado su

comportamiento.

En la siguiente figura mostramos el resultado de la misma simulación siempre con los mismos parámetros, pero con la diferencia de que en cada serie los seguidores tempranos pertenecen a la provincia del nombre de la serie. Hemos elegido Mallorca, por su insularidad y Cataluña por su alta densidad poblacional.

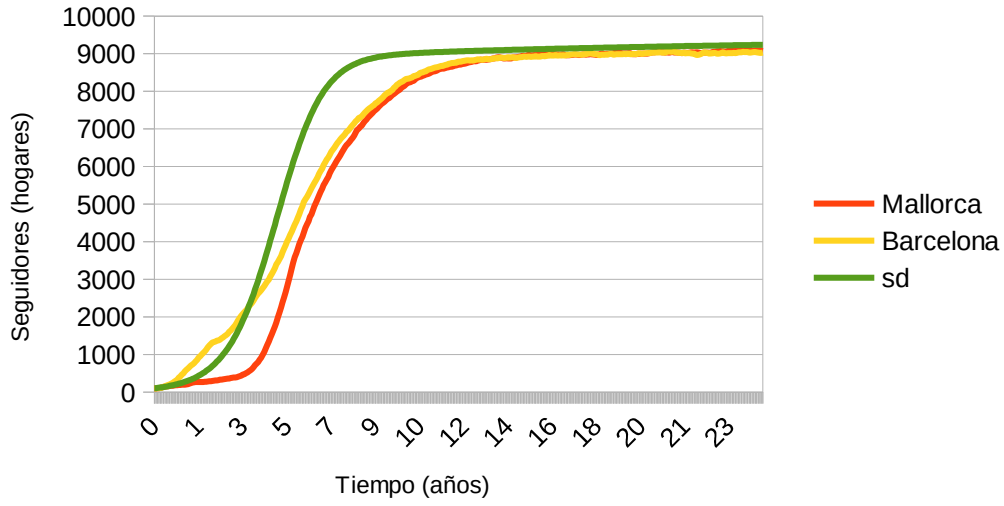


Figura 6.6: Inicio de difusión concentrado en provincia

Como observamos en la Figura 6.6, el proceso de difusión es dependiente de la ubicación inicial de los seguidores iniciales. Otra forma de apreciar esta variabilidad es mostrando resultados locales de la adopción. Para las siguientes tres simulaciones idénticamente parametrizadas, exceptuando la distribución inicial de seguidores tempranos, podemos ver que el inicio del proceso de adopción es completamente diferente cada vez.

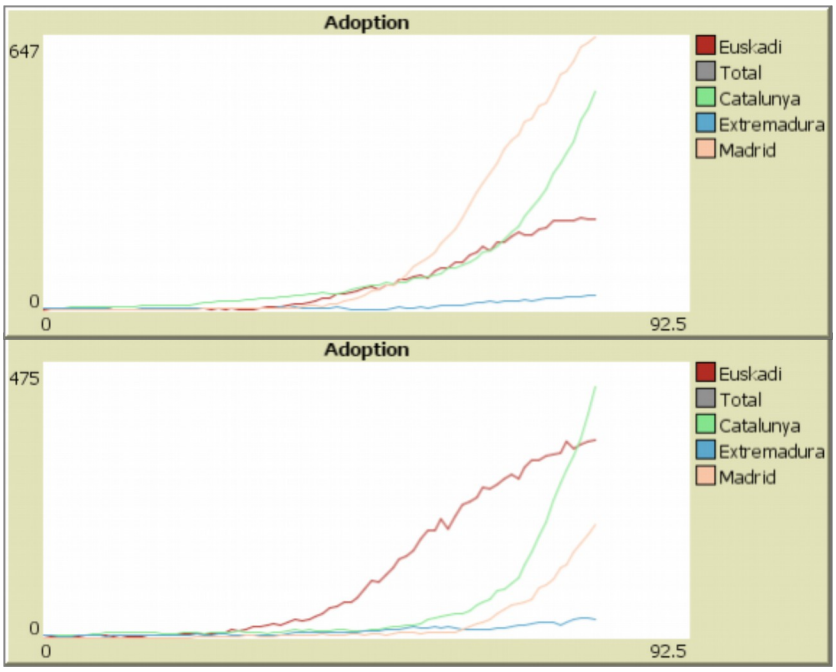


Figura 6.7: Variabilidad local del modelo

6.6 Conclusiones

Hemos intentado añadir una dimensión espacial al modelo de difusión gracias a la posibilidad de ubicar agentes geográficamente mediante la extensión GIS de NetLogo, lo cual nos ha requerido reprogramar el modelo del capítulo anterior, para adaptar el comportamiento de los agentes a la aparición de una nueva dimensión geoespacial.

Al incluir información GIS, hemos teniendo que variar los procedimientos de distribución inicial de baldosas y agentes, así como la forma de crear agentes para respetar la densidad poblacional de la cartografía seleccionada y las reglas de movimiento de los agentes para impedir imposibles físicos.

Al intentar comparar el modelo GIS con el del Capítulo anterior hemos visto cómo la concentración de agentes iniciales puede afectar al modelo de difusión acelerando o postergando el ritmo de adopción.

7 Aplicación práctica del modelo, Som Energia

7.1 Introducción

Nos gustaría hacer incapié en que este proyecto no ha sido desarrollado a medida para este caso, es más resulta difícil encontrar ejemplos en el mundo real de difusión de innovación en mercados cerrados (restringidos). Nuestro objetivo ha sido explicar como se puede realizar la combinación de la Dinámica de Sistemas con sistemas multiagentes introduciendo restricciones espaciales.

En este capítulo vamos a poner a prueba la fexibilidad del modelo, para ello vamos a aplicar el modelo a la difusión de un servicio en vez de a un producto. Vamos a aplicar el modelo, a un producto innovador que ha surgido en Cataluña y que se está extendiendo al resto del estado; se trata de la cooperativa de producción y consumo de energías renovables Som Energía.

La cooperativa Som Energia se formó en 2010 impulsada por un grupo de profesores y ex-alumnos de la Universidad de Girona, con la intención de reproducir experiencias exitosas en otros países europeos como Ecopower (Flandes), Enercoop (Francia) o Greenpeace Energy (Alemania).

Los principales objetivos de esta cooperativa son ofrecer a sus socios la posibilidad de consumir energía 100% renovable a un precio similar al de la electricidad convencional.

La cooperativa ofrece abiertamente sus datos en su propia web con lo que hemos dispuesto de datos reales sobre la evolución de socios en los últimos 3-4 años, aunque disponemos de datos a nivel nacional vamos a acotar los resultados a la CCAA de Cataluña ya que es un proyecto local.

7.2 Cambios realizados

Producción

El modelo que hemos contemplado hasta ahora, ha sido un modelo productivo, en el que se producía un producto, cuya vida útil, se prolongaba durante varios ejercicios. En el caso de la

producción de un servicio, consideramos que la entrega del producto se realiza cuando se genera la factura. Por lo tanto, el cambio mas importante es, que la adopción no se realiza a razón de X productos por seguidor (X en los anteriores ejemplos ha sido 1 producto por seguidor), establecemos que la adopción se realiza mes a mes, dado que Som Energia tiene un modelo de facturación mensual.

Por lo tanto la producción acumulada depende de los seguidores y no del ratio de adopción. La Ecuación 4.3 queda de la siguiente manera:

$$Q = Q_i + \int R_a \, dt \tag{7.1}$$

Paso de integración

Al tener una factura mensual hemos cambiado el paso de integración a **dt** 1/12 en vez de **dt** 1/16.

Cambios en la cartografiá

Utilizando la cartografía del anterior capítulo, hemos confeccionado un *shape* correspondiente a la CCAA de Cataluña mediante segmentación utilizando QGIS.

Redimensión de la matriz mundo

Hemos modificado el tamaño de la matriz a 221x201 para que las celdas resultantes, al aplicar la cartografía, se ajusten a una densidad similar a la del modelo híbrido del capítulo 5 de 0,83 tortugas por celda.

Población

Consideramos que los potenciales clientes de Som energia Cataluña son 13.500 y la población inicial.

Precio inicial

El precio del servicio se ha establecido en 110€/mes ya que es el consume medio por hogar en Catalunya según la OCU (Organización de Consumidores y Usuarios).

Población inicial

Som Energía comienza su andadura en Girona en diciembre del año 2010 cuando 150 socios deciden crear la cooperativa. Por lo que los seguidores tempranos de este servicio innovador se establecen en 350[1] personas en nuestro modelo y los hemos ubicado en la provincia de Girona y limítrofes aunque la mayoría de los seguidores tempranos se ubican en Girona.

7.3 Resultados

Disponemos a través de la página web de Som Energia de los datos de evolución de número de socios de los ejercicios 2013 y 2014.

Ene	feb	mar	abr	may	jun	jul	ago	set	oct	nov	dic	Ene 14	Feb 14	Mar 14	Abr 14	May 14	Jun 14	Jul 14 Ago	Set	Oct	
2660	3157	3459	3690	3919	4054	4246	4519	4736	5013	5199	5571	5805	6094	6324	6547	6729	6978	7151	7302	7446	7688
593	648	694	719	751	773	802	830	857	911	930	976	998	1035	1063	1087	1113	1145	1164	1189	1213	1240
154	191	223	254	273	283	304	315	327	333	351	379	400	422	435	456	465	483	499	513	523	542
133	169	187	200	215	218	231	252	261	273	279	303	316	330	344	360	368	386	396	407	422	435
3540	**4165**	**4563**	**4863**	**5158**	**5328**	**5583**	**5916**	**6181**	**6530**	**6759**	**7229**	**7519**	**7881**	**8166**	**8450**	**8675**	**8992**	**9210**	**9411**	**9604**	**9905**

Tabla 7.1: Evolución clientes Som Energia 13/14

Hemos realizado una simulación con los mismos parámetros que en el capítulo anterior $\varepsilon_C = 0{,}8$, $\lambda = 1$, $\Theta = 1{,}07$ (ver Figura 6.1) . Y con una distribución geográfica inicial de seguidores que se asemeje a la histórica.

Provincia	**Clientes iniciales**
Girona	150
Barcelona	155
LLeida	25

Tabla 7.2: Clientes iniciales por provincia

1 El 31 de Diciembre de 2010 Som energía tenía 350 socios, fuente https://www.somenergia.coop/quienes-somos/

En la Figura 7.1, reproducimos el resultado de la simulación (en rojo) y los datos reales del total de seguidores en Cataluñaa en el periodo 2013-1014 (en amarillo). Si nos fijamos, la evolución del número declientes según la simulación se acrecentaría a mediados finales del ejercicio 2012.

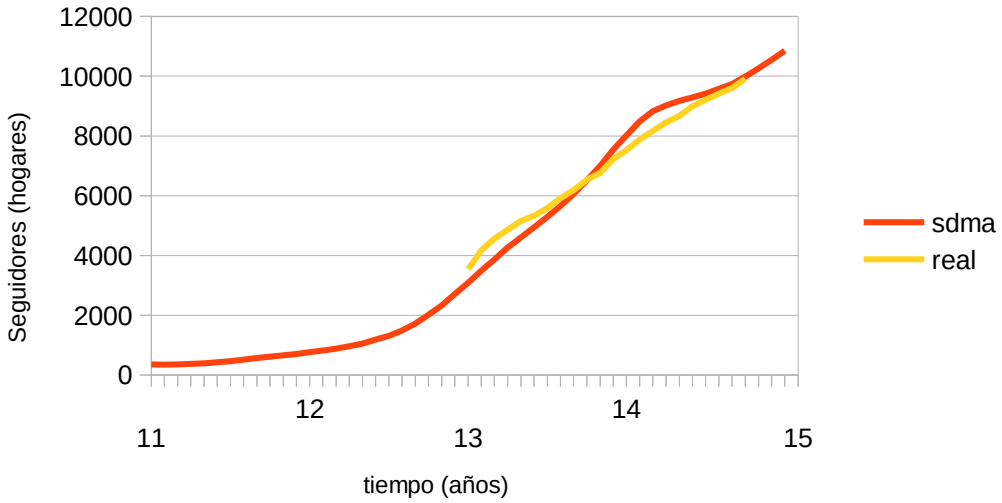


Figura 7.1: Evolución clientes Som Energia Cataluña 2011-214

Lamentablemente, no disponemos de los datos correspondientes al ejercicio 2012[1] para Cataluña, pero sí disponemos de los datos Nacionales del ejercicio 2012 (en los cuales el peso de Catalunya es preponderante). En la Figura 7.2, donde se muestran los datos históricos de seguidores totales, podemos ver que en el ejercicio 2012 se produce un cambio de tendencia tal como se recoge en la simulación.

1 En la página web de Som Energía aparce una entrada referente a estos datos pero no hay enlace

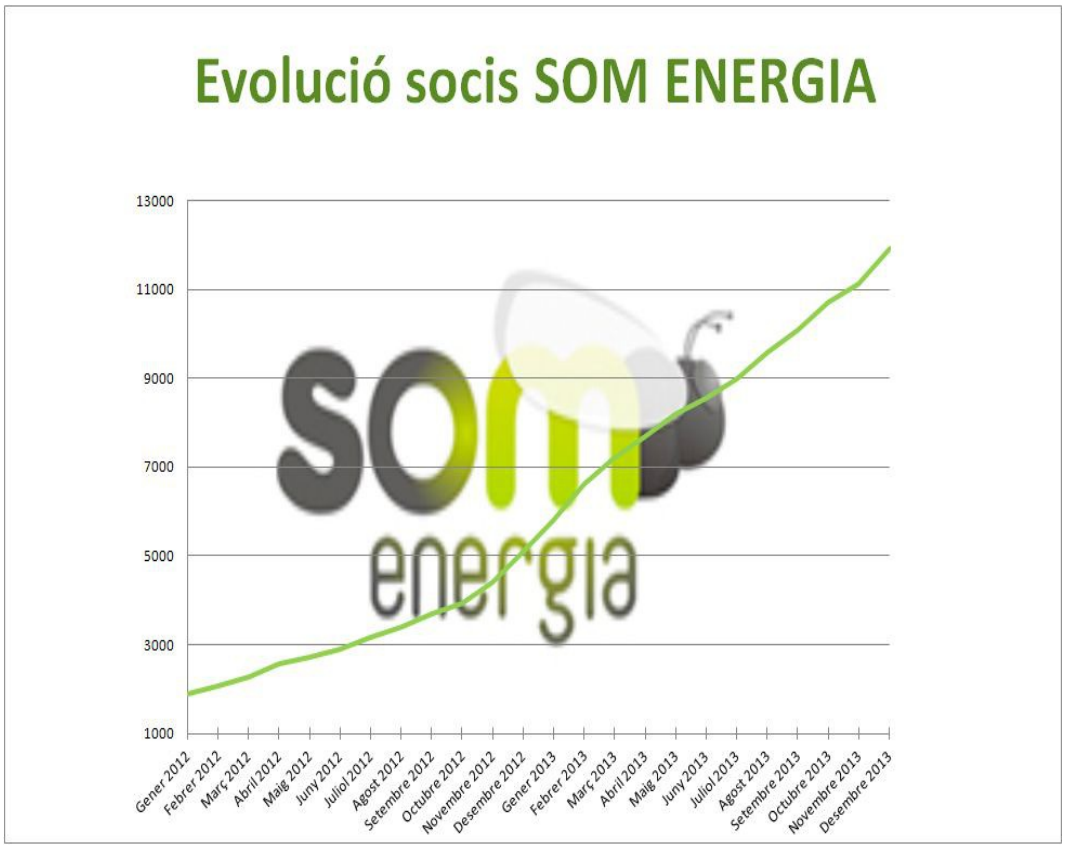


Figura 7.2: Evolución socios Som Energia Totales 2012-2013

A continuación, comparamos el resultado real frente al obtenido mediante simulación de los socios en octubre de 2014, último mes de la simulación, tanto en valores absolutos Figura 7.4, como en valores relativos Figura 7.3.

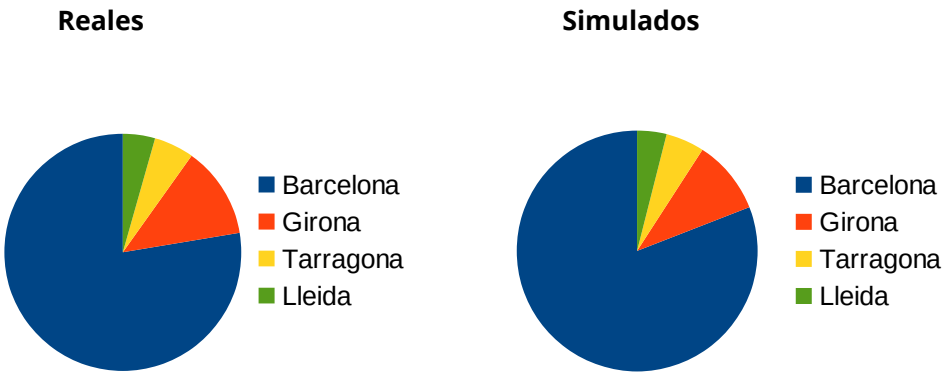


Figura 7.3: Clientes Som Energia Octubre 2014 reales / simulación

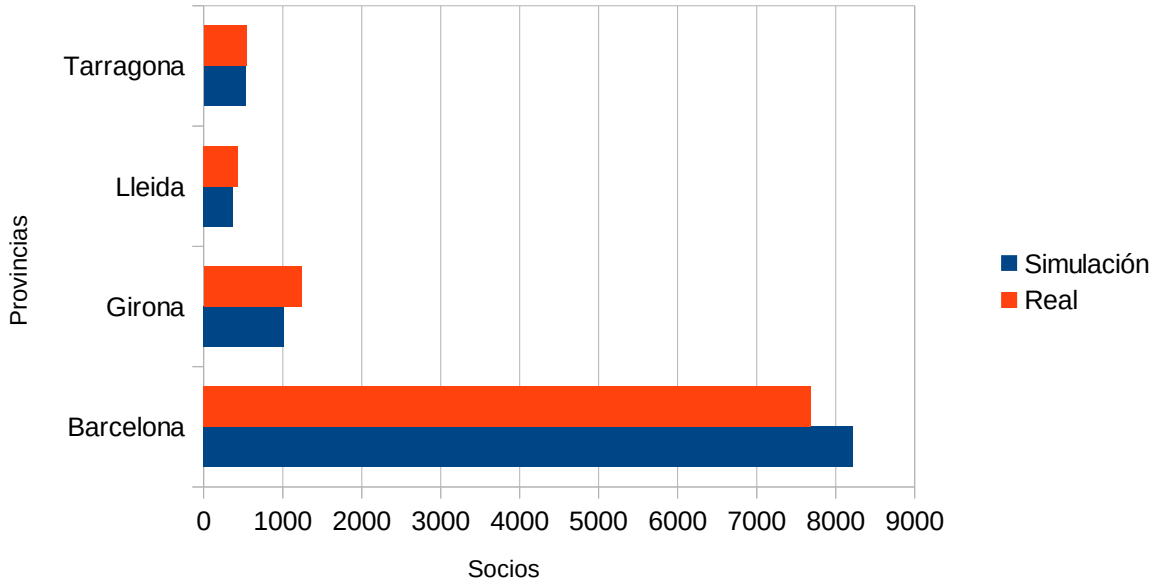


Figura 7.4: Socios octubre 2014

7.4 Conclusiones

Hemos aplicado el modelo a un proceso de difusión de un servicio de suministro, electricidad verde, cuyas características distan mucho del objetivo teórico con el que se creo el modelo original, distribución de productos industriales. Realizando unos pequeños ajustes, hemos conseguido un resultado que se aproxima al real, lo cual demuestra la flexibilidad del modelo.

Cabe destacar que la distribución inicial de seguidores influye en la evolución del proceso de difusión y que si probamos a realizar la misma simulación con una distribución diferente de seguidores iniciales el resultado puede variar.

La provincia de Girona presenta la desviación más grande entre los datos reales y los simulados, aunque le hemos dado un peso importante a la hora de distribuir seguidores iniciales a esta provincia, hay un déficit de socios según el modelo. Quizás, en el caso real, el efecto del *marketing* o de la publicity[1] ha podido ser mayor en esta provincia ya que el modelo a considerado una difusión homogénea de la difusión del *marketing* en Cataluña.

1 Publicity, dícese de la publicidad obtenida gratuitamente, reportajes, documentales, chralas ...

8 Planificación y costes del proyecto

8.1 Introducción

El proyecto fue planificado desde su inscripción como anteproyecto, con el objetivo definido de estar finalizado en octubre de 2014. Vamos a repasar los hitos y entregables para después analizar en cuáles y por qué, se han producido desviaciones.

Los entregables del proyecto son los siguientes:

1. Estudio comparativo de las herramientas de simulación AnyLogic, NetLogo, Repast.
2. Tutorial de NetLogo.
3. Implementación del modelo de difusión de la innovación de Ahmadian en VensimPLE
4. Reimplementación del modelo de difusión de la innovación VensimPLE en NetLogo.
5. Modelo híbrido de la difusión de la innovación.
6. Modelo híbrido integrado con GIS.
7. Memoria.

8.2 Planificación y desarrollo del proyecto

La planificación inicial fue la siguiente:

Estudio e implementación del modelo clásico	01/03/14	20/03/14
Estudio	01/03/14	10/03/14
Implementación Vensim	01/03/14	10/03/14
Redacción entregable	11/03/14	20/03/14
Estudio comparativo de herramientas de simulación	22/03/14	06/05/14
AnyConnect	22/03/14	20/04/14
NetLogo	22/03/14	20/04/14
Repast	22/03/14	20/04/14
Redacción entregable	21/04/14	06/05/14

Análisis y descomposición del sistema	07/05/14	16/05/14
Implementación	14/05/14	15/07/14
Implementación SD en NetLogo	14/05/14	27/05/14
Implementación SD+MA en NetLogo	28/05/14	15/07/14
Tutorial de NetLogo	14/05/14	15/07/14
Validación	17/06/14	16/07/14
Redacción Entregable	17/07/14	26/07/14
Implementación GIS	17/07/14	05/08/14
Validación de la Implementación GIS	06/08/14	31/08/14
Redacción memoria	01/09/14	30/09/14

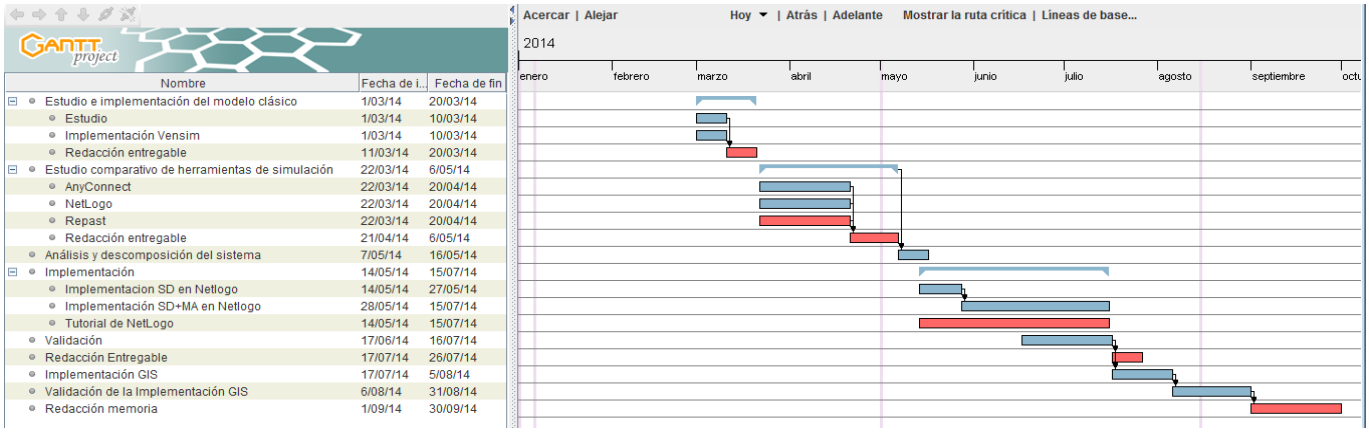


Figura 8.1: Planificación inicial del proyecto

La gran mayoría de la planificación se ha cumplido, pero nos hemos encontrado varios hitos retrasados; en concreto:

1. Implementación del modelo de difusión de la innovación de Ahmadian en VensimPLE.
2. Reimplementación del modelo de difusión de la innovación VensimPLE en NetLogo.
3. Estudio comparativo de las herramientas de simulación AnyLogic, NetLogo, Repast.
4. Memoria.

En los dos primeros casos la estimación inicial realizada fue incorrecta, solo se puede atribuir a la inexperiencia en el manejo de las herramientas VensimPLE y NetLogo. El modelo inicial es

enormemente complejo, con multitud de ecuaciones, y cualquier mínimo fallo en la transcripción provocaba errores muy difícilmente localizables. En cuanto a la reimplementación en NetLogo, hubo también un retardo ya que NetLogo realizaba un tratamiento de las variables de nivel diferente a VensimPLE y hasta que nos dimos cuenta pasaron un par de jornadas.

En cuanto al estudio de las herramientas comparativas, realizamos un apartado con un coste excesivamente alto en horas. Consistía en realizar un *Benchmark* sobre la misma máquina, que medía el rendimiento de un problema clásico de la IA, "el juego de la vida", en los diferentes entornos de simulación. Al final, este apartado no se ha publicado ya que nos dimos cuenta de que el problema que se va a simular influía notablemente en el comportamiento de las plataformas, y los resultados no eran extrapolables.

Para finalizar, la última y mayor desviación se ha producido en la estimación del tiempo dedicado a redactar la memoria. Aunque el tiempo de programación de este proyecto no ha sido muy elevado, el número de metodologías, teorías y, herramientas tratadas; junto a la necesidad de presentarlas y justificar su idoneidad han incrementado enormemente el esfuerzo en la redacción de la memoria, esfuerzo que no se había contemplado en la planificación inicial.

8.3 Costes del proyecto

Vamos a realizar un análisis de coste desde un punto de vista empresarial.

Costes Fijos

Tratándose de un proyecto para el que hemos utilizado un PC, hemos estimado que resultaría más rentable comprar un PC para el proyecto que el coste que supondría alquilar (leasing) un PC para este periodo.

Portátil Acer ..... 449 (IVA incluido)

Costes Variables

El coste más importante es el de personal, que hemos establecido de la siguiente manera: salario bruto de Ingeniero Informático 39.102[1] €/anuales, suponiendo unas cargas sociales del 29.9% el coste anual asciende a 50.793 €/año. El coste por hora, teniendo en cuenta que de media en España se trabajan 1.665[2] horas (prorrateado) , sería de 30,5 € por hora. Suponemos que el personal dedicado a este proyecto tele-trabaja, con lo que no incluimos costes de alquileres de locales/oficinas al coste del proyecto.

El coste total en horas del proyecto se detalla en la siguiente tabla:

Estudio e implementación del modelo clásico	
Estudio	38h
Implementación Vensim	25h
Redacción entregable	16h
Estudio comparativo de herramientas de simulación	20h
Redacción entregable	4h
Análisis y descomposición del sistema	19h
Implementación	
Implementación SD en NetLogo	30h
Implementación SD+MA en NetLogo	50h
Tutorial NetLogo	15h
Validación	20h
Redacción entregable	6h
Implementación GIS	50h
Redacción memoria	80h
TOTAL	373h

Tabla 8.1: Desglose de horas por hito

El proyecto suma 373 horas, con lo que el coste variable del proyecto asciende a:

Coste Personal ..... 11.376,5 €

Coste Total ..... 11.825 €

1 Fuente Infojobs http://orientacion-laboral.infojobs.net/salarios-infografia
2 Según estadísticas de la OCDE

8.4 Conclusiones

El trabajo se ha realizado con retrasos respecto a su planificación inicial. Retrasos asumibles que tienen su origen en dos causas: la dificultad de estimar el esfuerzo y las numerosas herramientas y tecnologías involucradas en el proyecto.

Desde un punto de vista académico el coste en créditos ECTS del proyecto asciende a 14 créditos (373 / 25[1]) lo que equivale a 28 créditos antiguos que son los aplicables a esta titulación. Teniendo en cuenta que se exigen 6 créditos, podemos concluir que se ha cumplido sobradamente el requisito exigido para la obtención del título.

1 ECTS=25h fuente http://www.uned.es/gabinete/espacio/texto.asp-id_texto=5.htm

9 Conclusiones y trabajos futuros

9.1 Introducción

Autores como (Schieritz & Milling 2003) , (Lorenz & Jost 2006) , (Pourdehnad et al. 2002) plantean el uso combinado de diferentes métodos de modelados, ya que cada método puede resultar mejor a la hora de simular ámbitos diferentes. Los sistemas complejos pueden descomponerse en varios niveles y cada nivel, dependiendo de sus características, puede ser convenientemente modelado utilizando un paradigma que no tiene por qué corresponder con el resto de niveles. En el caso que hemos estudiado, la difusión de la innovación, hemos visto que su descomposición en niveles, resultaba en que, cada nivel se adecuaba mejor a una forma de modelado en concreto.

En general, fenómenos como la toma de decisiones políticas, establecimiento de estrategias etc. tienen lugar en un nivel macro, que a su vez afectan a otro nivel, el micro, donde sus decisiones afectan a individuos que a su vez conforman un comportamiento colectivo que afecta al nivel macro. Este bucle de realimentación multinivel no puede ser modelado mediante el uso de la Dinámica de Sistemas. En estos casos uno puede tener la tentación de modelar el sistema utilizando una metodología multiagente donde varios tipos de agentes pueden ocuparse de cada nivel. No debemos olvidar que computacionalmente los modelos de dinámica de sistemas, son infinitamente más eficientes que los sistemas multiagente y por tanto lo más adecuado es utilizar un modelo utilizando ambas metodologías.

La irrupción de herramientas de modelado multinivel como las que hemos comparado en este trabajo permiten realizar modelos multimétodo fácilmente, si bien es cierto que requieren una adaptación por parte de los modeladores acostumbrados a trabajar con una única metodología. Para el simulador social proveniente de ciencias sociales, la combinación de varios paradigmas de simulación junto a tecnologías GIS, la barrera de entrada muy importante. Tras el análisis de herramientas de simulación que hemos realizado en el Apartado 2.5 concluimos que NetLogo resulta un entorno adecuado para iniciarse en la simulación multimétodo con soporte GIS.

Los entornos modernos de simulación multiagente, además comienzan a integrarse con sistemas de información geográficos, abriendo un nuevo campo de modelado quizás una nueva

disciplina de modelado geográfico. Esta integración resulta totalmente natural con una metodología multiagente, pero no casa demasiado bien con una metodología como la Dinámica de Sistemas acostumbrada a trabajar con variables a nivel macro.

Concluimos que ante este dilema existencial la Dinámica de Sistemas puede responder integrándose con sistemas multiagentes como hemos mostrado en este trabajo.

9.2 Conclusiones

Hemos visto que con el lenguaje NetLogo resulta sencillo realizar modelos multimétodo de Dinámica de Sistemas y sistemas multiagentes, un enfoque híbrido que permite modelar sistemas compuestos de diferentes niveles.

Hemos comparado NetLogo con otros entornos de simulación híbridos (AnyLogic y Repast) y hemos visto que no hay una plataforma claramente más ventajosa que las otras, cada una tiene su nicho y es más adecuada para modelar ciertos tipos de sistemas. Concluimos que las características diferenciadoras de NetLogo respecto a los otros entornos son la baja curva de aprendizaje, debido a la sencillez y potencia de su lenguaje, y su validez como herramienta de prototipado rápido.

La extensión GIS de NetLogo, al igual que el Lenguaje NetLogo, es muy sencilla de utilizar y da soporte a diferentes tipos de *rasters* y *shapes*. Sin embargo, desde un punto de vista computacional no es nada eficiente; la carga inicial del *raster* es lenta y consume mucha memoria; este consumo que depende de la información importada del *shape* y del tamaño de la matriz definida en NetLogo.

Hemos visto como gracias a las nuevas herramientas de simulación multimétodo con soporte GIS como MAGISmo, NetLogo podemos extender modelos clásicos en Dinámica de Sistemas para que tengan en cuenta la información geográfica.

Hemos visto que gracias a la librería GDAL y a herramientas como QGIS, que utilizan la citada librería, la creación o adaptación de *shapes* existentes para su posterior uso en NetLogo es relativamente sencilla.

Por último hemos elaborado un modelo de difusión en el que la distribución inicial de los agentes afecta al proceso de difusión. Un comportamiento que observamos en el día a día, que gracias a las herramientas GIS y sistemas multiagentes puede empezar a estudiarse.

9.3 Trabajos futuros

El marco teórico que hemos llevado a la práctica mediante un ejemplo puede ser fácilmente transportable a problemas de crecimiento urbano, recogida selectiva de basuras, problemas ecológicos.

Como hemos indicado a lo largo del proyecto, existen varias cuestiones que no han sido recogidas en este modelo pero que podrían ser recogidas fácilmente y podrían ser posibles trabajos a realizar en el futuro:

1. Como señalábamos en el apartado 5.5.2 el comportamiento de los agentes sigue un modelo de relaciones equivalente a los “circulos sociales” de Hamil y Gilbert, modelo, que no tiene en cuenta las relaciones a largas distancia, y que se podrían simular mediante enlaces entre agentes.

2. Como señalabamos en el apartado 5.5.1 los agentes que hemos desarrollado son simples y no han recogido los cinco estados (Conocimiento, Persuasión, Decisión y Confirmación) de la teoria de Rogers. Estados que fácilmente podrían haber sido incorporados al modelo, si hubiéramos construidos agentes siguiendo alguna arquitectura de las estudiadas en el apartado 2.3.4 lógica,BDI, etc.

3. La competencia, este modelo no ha contemplado el efecto de la irrupción de la competencia. Si bien se podría realizar fácilmente, teniendo en cuenta que en la literatura DOI se pueden encontrar innumerables teorias y ejemplos.

El modelo que hemos extendido no deja de ser un marco teórico. Habría que poner a prueba el modelo resultante en otros casos diferentes ya que el ejemplo práctico que hemos analizado, Som Energia , no es un caso que se adecue al modelo de difusión del que hemos partido. Creemos que esté es el principal trabajo a realizar en el futuro, aplicar el modelo a un proceso de difusión global del que se dispongan datos locales mas precisos.

10 Listado de Referencias y Bibliografía

Ahmadian, A., 2008. System dynamics and technological innovation system.

Anón, 2008. How to Build a Combined Agent Based / System Dynamics Model in AnyLogic. , p.23. Disponible en: http://www.xjtek.com/upload/iblock/c8d/c8d0c764d2026b92a18dc53c3fef44f3.pdf.

Aracil, J., 1995. *Dinámica de Sistemas* Isdefe, ed., Madrid.

Aracil, J. & Gordillo, F., 2005. *Dinámica de Sistemas*, Madrid: ALIANZA EDITORIAL.

Arrow, K.J., 1962. The economic implications of learning by doing. *The Review of Economic Studies*, 29, pp.155-173.

Axelrod, R., 1997. *The complexity of cooperation*, Press, Princeton University. Disponible en: http://faculty.fuqua.duke.edu/~willm/bio/cv/papers/ComplexityCooperation1995_CoalitionFormation.pdf.

Bass, F.M., 1969. A New Product Growth for Model Consumer Durables. *Management Science*, 15, pp.215-227. Disponible en: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.15.5.215.

Benenson, I. & Torrens, P.M., 2004. Geosimulation: object-based modeling of urban phenomena. *Computers, Environment and Urban Systems*, 28, pp.1-8.

Bert, F.E. et al., 2014. Lessons from a comprehensive validation of an agent based-model: The experience of the Pampas Model of Argentinean agricultural systems. *Ecological Modelling*, 273, pp.284-298.

Boulding, W. & Staelin, R., 1990. Environment, Market Share, and Market Power. *Management Science*, 36, pp.1160-1177.

Brooks, R., 1986. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2.

Covrigaru, A., L.R., 1991. Deterministic Autonomus Systems. *IA magazine*, 12(3), pp.110-117.

Epstein, J.M. & Axtel, R., 1996. Growing Artificial Societies: Social Science from the Bottom Up. *Foreign Affairs*, 76.

Forrester, J.W., 1962. *Industrial Dynamics*, Cambridge, Machassusetts: MIT Press.

Forrester, J.W., 1969. *Urban Dynamics*, Cambridge, Massachusetts: MIT Press.

Forrester, J.W., 1971. *World Dynamics*, Cambridge, Massachusetts: MIT Press.

Fourt, L.A. & Woodlock, J.W., 1960. Early Prediction of Market Success for New Grocery Products. *Journal of Marketing*, 25, pp.31-39.

Gebetsroither, E., 2010. Combining Multi Agent System modeling and System Dynamics modeling. En *Proceedings of the 20th European Meeting on Cybernetics and Systems Research (EMCSR)*.

Gebetsroither, E., 2013. Multimethod Modeling and Simulation Supporting Urban Planning Decisions. En C. Walloth, J. Martin Gurr, & J. (Alexander Schmidt, eds. *Understanding Complex Urban Systems: Multidisciplinary Approaches to Modeling*. Springer International Publishing, pp. 13-27. Disponible en: http://download.springer.com/static/pdf/992/chp%3A10.1007%2F978-3-319-02996-2_2.pdf?auth66=1412849794_ebdce494ebadf3f7383aa157aebabdfc&ext=.pdf.

GMU (George Mason University), 2014. MASON. Disponible en: http://cs.gmu.edu/~eclab/projects/mason/.

Hamill, L. & Gilbert, N., 2009. Social Circles: A Simple Structure for Agent-Based Social Network Models. *Journal of Artificial Societies and Social Simulation*, 12.

Helbing, D. ed., 2012. *Agent-Based Simulations and Experiments to Study Emergent Social Behavior*,

Huhns, M. N. & Singh, M.P., 94d. C. Distributed Artificial Intelligence for Information Systems. *CKBS-94 Tutorial, June 15, University of Keele, UK*.

J. Poza García, D., Manual de Netlogo en español. Disponible en: https://sites.google.com/site/manualnetlogo/ [Accedido octubre 20, 2014].

Kahouli-Brahmi, S., 2008. Technological learning in energy-environment-economy modelling: A survey. *Energy Policy*, 36, pp.138-162.

Kotler, P. & Keller, K.L., 2009. *Marketing Management*, Disponible en: http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=44033223&site=ehost-live.

Leenders, M., 2002. The effectiveness of different mechanisms for integrating marketing and R&D. *Journal of Product Innovation Management*, 19, pp.305-317. Disponible en: http://www.sciencedirect.com/science/article/pii/S0737678202001479.

Lorenz, T. & Jost, A.P., 2006. Towards an orientation framework in multi-modeling paradigms. *Proceedings of the 24th International Conference of the System Dynamics Society*, p.86.

Macal, C. & North, M., 2010. Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4, pp.151-162.

Mansfield, E., 1961. Technical Change and the Rate of Imitation Author. *Econometrica*, 29, pp.741-766.

Meadows D.H., Meadows D.L., Randers J., B.I.W.W., 1972. *The Limits to Growth: A Report for the Club of Rome's Project on the Predicament of Mankind*, New York: Universe Books.

Meadows, Dennis L, D.H. y J.M.R., 1985. *The Electronic Oracle: Computer Models and Social Decisions* Chichester, ed.,

Michael E. Bratman, 1999. *Intention, Plans, and Practical Reason*,

Morlán Santa Catalina, I., 2010. *Modelo de Dinámica de Sistemas para la implantación de Tecnologías de la Información en la Gestión Estratégica Universitaria*. Universidad del País Vasco / Euskal Herriko Unibertsitatea.

Naylor, T.H. & Finger, J.M., 1967. Verification of Computer Simulation Models. *Management Science*, 14, p.B-92-B-101.

Nikolai, C. & Madey, G., 2009. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. *Journal of Artificial Societies and Social Simulation*, 12.

Nwana, H.S., 1996. Software agents: an overview. *The Knowledge Engineering Review*, 11, p.205.

Pourdehnad, J., Maani, K. & Sedehi, H., 2002. System Dynamics and Intelligent Agent-Based Simulation: Where is the Synergy? En *System Dynamics*. Disponible en: http://systemdynamics.org/conferences/2002/proceed/papers/Pourdeh1.pdf.

Qgis, Quantum GIS Project. *qgisorg*. Disponible en: http://www.qgis.org/.

Rand, W. & Rust, R.T., 2011. Agent-based modeling in marketing: Guidelines for rigor. *International Journal*

of Research in Marketing, 28, pp.181-193.

ROAD (Repast Organization for Architecture and Design), 2014. No Title. Disponible en: http://repast.sourceforge.net.

Rodrigez, A., 2012. Sistemas multiagente: Conceptos básicos.

Rodríguez Anaya, A., 2012. Sistemas multiagente: Conceptos básicos. , p.56.

Rogers, E.M., 1962. *Diffusion of Innovations*, Disponible en: http://books.google.com.sg/books?id=zw0-AAAAIAAJ&hl=en.

Russell, E. & Edelson, D., 2012. GIS Extension. Disponible en: http://ccl.northwestern.edu/netlogo/docs/gis.html.

Russell, S. & Norvig, P., 2009. *Artificial Intelligence: A Modern Approach*, Disponible en: http://portal.acm.org/citation.cfm?id=1671238&coll=DL&dl=GUIDE&CFID=190864501&CFTOKEN=29051579\npapers2://publication/uuid/4B787E16-89F6-4FF7-A5E5-E59F3CFEFE88.

Schelling, T.C., 1971. Dynamic models of segregation†. *The Journal of Mathematical Sociology*, 1, pp.143-186.

Schieritz, N. & Milling, P., 2003. Modeling the forest or modeling the trees. En *Proceedings of the 21st International System Dynamics Society*. pp. 1-15. Disponible en: http://www.systemdynamics.org/conferences/2003/proceed/PAPERS/140.pdf.

SGD, 2014. Swarm. Disponible en: http://www.swarm.org.

Sterman, J.D., 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Disponible en: http://www.amazon.com/dp/0071179895.

System Dynamics Society, System Dynamics Society. Disponible en: http://tools.systemdynamics.org/core-sd-software/ [Accedido noviembre 12, 2014].

Wilensky, U., 2014a. Modeling Commons. Disponible en: http://modelingcommons.org/account/login [Accedido octubre 20, 2014].

Wilensky, U., 1999. NetLogo. Disponible en: http://ccl.northwestern.edu/netlogo/.

Wilensky, U., 2014b. NetLogo faqs. Disponible en: http://ccl.northwestern.edu/netlogo/faq.html.

Wilensky, U., 2014c. NetLogo Models Library. Disponible en: http://ccl.northwestern.edu/netlogo/models/ [Accedido octubre 20, 2014].

Wilensky, U., 2006a. NetLogo Tabonuco Yagrumo Hybrid model. Disponible en: http://ccl.northwestern.edu/netlogo/models/TabonucoYagrumoHybrid.

Wilensky, U., 2006b. NetLogo User Manual. Disponible en: http://ccl.northwestern.edu/netlogo/docs/.

Wooldridge, M., Jennings, N.R. & Wooldbridge, M., 1995. Intelligent agents: Theory and practice. *Knowledge engineering …*, 10, pp.1-62. Disponible en: http://journals.cambridge.org/production/action/cjoGetFulltext?fulltextid=4100132.

XT Technologies, 2014. AnyLogic. Disponible en: http://www.anylogic.com/.

11 Siglas, Abreviaturas y Acrónimos

COMMOD Community modelling

DOI Difussion of innovation

DS Dinámica de sistemas

EBB Efecto boca a boca

I+D Investigación y desarrollo

KISS Keep It Simple, Stupid!

MA Multiagente

SED Sistema de Eventos Discretos

SMA Sistemas multiagentes

12 ANEXO I – Modelo de difusión de la Innovación mediante Dinámica de Sistemas

```
globals [ fichero ]

to setup
 ca
 system-dynamics-setup
 set fichero "bassskетch-7.csv"
 if (file-exists? fichero) [file-delete fichero]
 set Progress_ratio Slider_progress_ratio
end


to go
 if ticks >= 25 [ stop ]
 set-current-plot "Adoption"
 system-dynamics-go
 system-dynamics-do-plot
 set-current-plot "Rate" plot Adoption_rate * ( 1 / dt )
 set-current-plot "Cumulative_production" plot Cumulative_production
 set-current-plot "Total_cost" plot Total_cost
 set-current-plot "Unit_cost" plot Unit_cost
 set-current-plot "Price" plot Price
 set-current-plot "Marketing_performance" plot Marketing_performance
 set-current-plot "Profit" plot Profit
 set-current-plot "Investment_on_marketing" plot Investment_on_marketing
 set-current-plot "Revenue" plot Revenue
 set-current-plot "Performance_improvement" plot Performance_improvement
 set-current-plot "Atractiveness_from_price " plot Atractiveness_from_price
          set-current-plot          "Total_attractiveness_of_technology"          plot
Total_attractiveness_of_technology
 set-current-plot "Rate_of_growth_of_legitimacy" plot Rate_of_growth_of_legitimacy * (
1 / dt )
           set-current-plot           "Growth_of_legitimacy_through_wom"           plot
Growth_of_legitimacy_through_wom
         set-current-plot         "Ratio_of_adopters_to_population_alpha"         plot
Ratio_of_adopters_to_population_alpha
 set-current-plot "Wom_effectiveness" plot Wom_effectiveness
         set-current-plot         "Growth_of_legitimacy_thorugh_marketing"         plot
Growth_of_legitimacy_thorugh_marketing
 set-current-plot "Learning" plot Learning
 escribe-csv


end

to escribe-csv


 file-open fichero
 file-print  (word ticks "," Learning)
```

```
 file-close

end

;; System dynamics model globals
globals [
 ;; constants
 Population
 Adoption_fraction
 Contact_rate
 Technology_life_time
 Initial_cumulative_adopters
 Progress_ratio
 Number_of_product_per_adopter
 Initial_production_experience
 Fixed_variable_cost
 Initial_price
 Desired_profit_margin
 Capacity_utilization
 Ratio_of_target_price_to_cost_price
 Adjustment_time
 Marketing_performance_improvement_per_investment
 Share_of_investment_on_marketing
 Share_of_profit_on_investment
 Knowledge_increment_per_investment
 Performance_improvement_per_doubling_of_knowledge
 Contribution_share_of_knowledge_Acumulario_in_performance_improvement
 Sensivity_of_atractiveness_to_the_price
 Sensivity_of_attractivenes_to_the_technology_performance
 Rate_of_contact_between_customers_and_means_of_marketing
 Early_adopters_fraction
 ;; stock values
 Marketing_performance
 Stock_of_knowledge
 Stock_of_legitimacy
 Adopters
 Cumulative_production
 Potential_adopters
 Price
 ;; size of each step, see SYSTEM-DYNAMICS-GO
 dt
]

;; Initializes the system dynamics model.
;; Call this in your model's SETUP procedure.
to system-dynamics-setup
 reset-ticks
 set dt 0.5
 ;; initialize constant values
 set Population 10000
 set Adoption_fraction 0.01
 set Contact_rate 50
 set Technology_life_time 8
 set Initial_cumulative_adopters 100
 set Progress_ratio 0.8
 set Number_of_product_per_adopter 1
 set Initial_production_experience 1000
```

```
 set Fixed_variable_cost 3
 set Initial_price 1000
 set Desired_profit_margin 0.2
 set Capacity_utilization 1
 set Ratio_of_target_price_to_cost_price 1.01
 set Adjustment_time 0.5
 set Marketing_performance_improvement_per_investment 0.00001
 set Share_of_investment_on_marketing 0.2
 set Share_of_profit_on_investment 0.4
 set Knowledge_increment_per_investment 0.000001
 set Performance_improvement_per_doubling_of_knowledge 0.5
 set Contribution_share_of_knowledge_Acumulario_in_performance_improvement 0.5
 set Sensivity_of_atractiveness_to_the_price -2
 set Sensivity_of_attractivenes_to_the_technology_performance 0.5
 set Rate_of_contact_between_customers_and_means_of_marketing 0.12
 set Early_adopters_fraction 0.002
 ;; initialize stock values
 set Marketing_performance 0
 set Stock_of_knowledge 1
 set Stock_of_legitimacy 0
 set Adopters Initial_cumulative_adopters
 set Cumulative_production Initial_production_experience
 set Potential_adopters Population - Adopters
 set Price Initial_price
end

;; Step through the system dynamics model by performing next iteration of Euler's
method.
;; Call this in your model's GO procedure.
to system-dynamics-go

 ;; compute variable and flow values once per step
 let local-Learning Learning
 let local-Learning_curve_exponent Learning_curve_exponent
 let local-Unit_variable_cost Unit_variable_cost
 let local-Unit_fixed_cost Unit_fixed_cost
 let local-Initial_unit_fixed_cost Initial_unit_fixed_cost
 let local-Initial_unit_variable_cost Initial_unit_variable_cost
 let local-Unit_cost Unit_cost
 let local-Cost_price Cost_price
 let local-Target_price Target_price
 let local-Profit_margin Profit_margin
 let local-Investment_on_marketing Investment_on_marketing
 let local-Total_investment Total_investment
 let local-Investment_on_rd Investment_on_rd
 let local-Share_of_investment_on_rd Share_of_investment_on_rd
 let local-Profit Profit
 let local-Revenue Revenue
 let local-Total_cost Total_cost
 let local-Performance_improvement Performance_improvement
              let              local-Sensivity_of_performance_per_cumulative_knowledge
Sensivity_of_performance_per_cumulative_knowledge
 let local-Atractiveness_from_price Atractiveness_from_price
 let local-Atractiveness_from_performance Atractiveness_from_performance
 let local-Total_attractiveness_of_technology Total_attractiveness_of_technology
 let local-Growth_of_legitimacy_through_wom Growth_of_legitimacy_through_wom
 let local-Ratio_of_adopters_to_population_alpha Ratio_of_adopters_to_population_alpha
```

```
                    let                    local-Growth_of_legitimacy_thorugh_marketing
Growth_of_legitimacy_thorugh_marketing
 let local-Marketing_efectiveness Marketing_efectiveness
 let local-Wom_effectiveness Wom_effectiveness
 let local-Adoption_rate Adoption_rate
 let local-Discar_rate Discar_rate
 let local-Production Production
 let local-Change_in_price Change_in_price
                    let                    local-Marketing_performance_improvement_rate
Marketing_performance_improvement_rate
 let local-Rate_of_increasing_in_knowledge Rate_of_increasing_in_knowledge
 let local-Rate_of_growth_of_legitimacy Rate_of_growth_of_legitimacy

 ;; update stock values
 ;; use temporary variables so order of computation doesn't affect result.
     let     new-Marketing_performance     (     Marketing_performance     +     local-
Marketing_performance_improvement_rate )
      let      new-Stock_of_knowledge      (      Stock_of_knowledge      +      local-
Rate_of_increasing_in_knowledge )
 let new-Stock_of_legitimacy ( Stock_of_legitimacy + local-Rate_of_growth_of_legitimacy
)
 let new-Adopters ( Adopters + local-Adoption_rate - local-Discar_rate )
 let new-Cumulative_production ( Cumulative_production + local-Production )
  let  new-Potential_adopters  (  Potential_adopters  -  local-Adoption_rate  +  local-
Discar_rate )
 let new-Price ( Price + local-Change_in_price )
 set Marketing_performance new-Marketing_performance
 set Stock_of_knowledge new-Stock_of_knowledge
 set Stock_of_legitimacy new-Stock_of_legitimacy
 set Adopters new-Adopters
 set Cumulative_production new-Cumulative_production
 set Potential_adopters new-Potential_adopters
 set Price new-Price

 tick-advance dt
end

;; Report value of flow
to-report Adoption_rate
  report  (  ;;Adopters  *  Adoption_fraction  *  Contact_rate  *  Potential_adopters  /
Population
  (  Early_adopters_fraction  +  Rate_of_growth_of_legitimacy  *  (  1  /  dt  )  )  *
Potential_Adopters
 ) * dt
end

;; Report value of flow
to-report Discar_rate
 report ( Adopters / Technology_life_time
 ) * dt
end

;; Report value of flow
to-report Production
 report ( Number_of_product_per_adopter * Adoption_rate * ( 1 / dt )
 ) * dt
end
```

```
;; Report value of flow
to-report Change_in_price
 report ( ( Target_price - Price ) / Adjustment_time
 ) * dt
end

;; Report value of flow
to-report Marketing_performance_improvement_rate
 report ( Investment_on_marketing * Marketing_performance_improvement_per_investment
 ) * dt
end

;; Report value of flow
to-report Rate_of_increasing_in_knowledge
 report ( Knowledge_increment_per_investment * Investment_on_RD
 ) * dt
end

;; Report value of flow
to-report Rate_of_growth_of_legitimacy
 report ( Growth_of_legitimacy_thorugh_marketing + Growth_of_legitimacy_through_wom
 ) * dt
end

;; Report value of variable
to-report Learning
    report    (    Cumulative_production    /    Initial_production_experience    )    ^
Learning_curve_exponent
end

;; Report value of variable
to-report Learning_curve_exponent
 report ( ln Progress_ratio ) / ( ln 2 )
end

;; Report value of variable
to-report Unit_variable_cost
 report Initial_unit_variable_cost * Learning
end

;; Report value of variable
to-report Unit_fixed_cost
 report Initial_unit_fixed_cost * Learning
end

;; Report value of variable
to-report Initial_unit_fixed_cost
 report ( Initial_price / ( 1 + Desired_profit_margin ) ) * ( Fixed_variable_cost / ( 1
+ Fixed_variable_cost ) )
end

;; Report value of variable
to-report Initial_unit_variable_cost
  report  (  Initial_price  /  (  1  +  Desired_profit_margin  )  )  *  (  1  /  (  1  +
Fixed_variable_cost ) )
end
```

```
;; Report value of variable
to-report Unit_cost
 report Unit_variable_cost + ( Unit_fixed_cost / Capacity_utilization )
end

;; Report value of variable
to-report Cost_price
 report ( 1 + Desired_profit_margin ) * Unit_cost
end

;; Report value of variable
to-report Target_price
 report Cost_price * Ratio_of_target_price_to_cost_price
end

;; Report value of variable
to-report Profit_margin
 report ( Price / Unit_cost ) - 1
end

;; Report value of variable
to-report Investment_on_marketing
 report Total_investment * Share_of_investment_on_marketing
end

;; Report value of variable
to-report Total_investment
 report Profit * Share_of_profit_on_investment
end

;; Report value of variable
to-report Investment_on_rd
 report Total_investment * Share_of_investment_on_rd
end

;; Report value of variable
to-report Share_of_investment_on_rd
 report 0.5 - Share_of_investment_on_marketing
end

;; Report value of variable
to-report Profit
 report Revenue - Total_cost
end

;; Report value of variable
to-report Revenue
 report Price * Production * ( 1 / dt )
end

;; Report value of variable
to-report Total_cost
 report (1 / dt ) * Production * Unit_cost
end

;; Report value of variable
```

```
to-report Performance_improvement
 report ( Stock_of_knowledge ^ Sensivity_of_performance_per_cumulative_knowledge ) *
Contribution_share_of_knowledge_Acumulario_in_performance_improvement
end

;; Report value of variable
to-report Sensivity_of_performance_per_cumulative_knowledge
 report ln ( 1 + Performance_improvement_per_doubling_of_knowledge ) / ln ( 2 )
end

;; Report value of variable
to-report Atractiveness_from_price
 report exp ( ( Sensivity_of_atractiveness_to_the_price * Price ) / Initial_price )
end

;; Report value of variable
to-report Atractiveness_from_performance
        report        Sensivity_of_attractivenes_to_the_technology_performance        *
Performance_improvement
end

;; Report value of variable
to-report Total_attractiveness_of_technology
 report Atractiveness_from_price + Atractiveness_from_performance
end

;; Report value of variable
to-report Growth_of_legitimacy_through_wom
   report   Wom_effectiveness   *   Ratio_of_adopters_to_population_alpha   *   (   1   +
Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Ratio_of_adopters_to_population_alpha
 report Adopters / Population
end

;; Report value of variable
to-report Growth_of_legitimacy_thorugh_marketing
 report Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Marketing_efectiveness
                    report                    Marketing_performance                    *
Rate_of_contact_between_customers_and_means_of_marketing
end

;; Report value of variable
to-report Wom_effectiveness
 report contact_rate * Adoption_fraction
end

;; Plot the current state of the system dynamics model's stocks
;; Call this procedure in your plot's update commands.
to system-dynamics-do-plot
 if plot-pen-exists? "Marketing_performance" [
```

```
  set-current-plot-pen "Marketing_performance"
  plotxy ticks Marketing_performance
 ]
 if plot-pen-exists? "Stock_of_knowledge" [
  set-current-plot-pen "Stock_of_knowledge"
  plotxy ticks Stock_of_knowledge
 ]
 if plot-pen-exists? "Stock_of_legitimacy" [
  set-current-plot-pen "Stock_of_legitimacy"
  plotxy ticks Stock_of_legitimacy
 ]
 if plot-pen-exists? "Adopters" [
  set-current-plot-pen "Adopters"
  plotxy ticks Adopters
 ]
 if plot-pen-exists? "Cumulative_production" [
  set-current-plot-pen "Cumulative_production"
  plotxy ticks Cumulative_production
 ]
 if plot-pen-exists? "Potential_adopters" [
  set-current-plot-pen "Potential_adopters"
  plotxy ticks Potential_adopters
 ]
 if plot-pen-exists? "Price" [
  set-current-plot-pen "Price"
  plotxy ticks Price
 ]
end
```

13 ANEXO II – Modelo híbrido de difusión de la Innovación

```
;; System dynamics model globals
globals [
 ;; constants
 Population
 Adoption_fraction
 Contact_rate
 Technology_life_time
 Initial_cumulative_adopters
 Progress_ratio
 Number_of_product_per_adopter
 Initial_production_experience
 Fixed_variable_cost
 Initial_price
 Desired_profit_margin
 Capacity_utilization
 Ratio_of_target_price_to_cost_price
 Adjustment_time
 Marketing_performance_improvement_per_investment
 Share_of_investment_on_marketing
 Share_of_profit_on_investment
 Knowledge_increment_per_investment
 Performance_improvement_per_doubling_of_knowledge
 Contribution_share_of_knowledge_Acumulario_in_performance_improvement
 Sensivity_of_atractiveness_to_the_price
 Sensivity_of_attractivenes_to_the_technology_performance
 Rate_of_contact_between_customers_and_means_of_marketing
 Adoption_rate
 Discar_rate
 Early_adopters_fraction
 ;; stock values
 Marketing_performance
 Stock_of_knowledge
 Stock_of_legitimacy
 Adopters
 Cumulative_production
 Potential_adopters
 Price
 ;; size of each step, see SYSTEM-DYNAMICS-GO
 dt
]

;; Initializes the system dynamics model.
;; Call this in your model's SETUP procedure.
to system-dynamics-setup
 reset-ticks
 set dt 0.0625
 ;; initialize constant values
 set Population 10000
 set Adoption_fraction 0.01
 set Contact_rate 50
 set Technology_life_time 8
 set Initial_cumulative_adopters 100
```

```
 set Progress_ratio 0.8
 set Number_of_product_per_adopter 1
 set Initial_production_experience 1000
 set Fixed_variable_cost 3
 set Initial_price 1000
 set Desired_profit_margin 0.2
 set Capacity_utilization 1
 set Ratio_of_target_price_to_cost_price 1.01
 set Adjustment_time 0.5
 set Marketing_performance_improvement_per_investment 0.00001
 set Share_of_investment_on_marketing 0.2
 set Share_of_profit_on_investment 0.4
 set Knowledge_increment_per_investment 0.000001
 set Performance_improvement_per_doubling_of_knowledge 0.5
 set Contribution_share_of_knowledge_Acumulario_in_performance_improvement 0.5
 set Sensivity_of_atractiveness_to_the_price -2
 set Sensivity_of_attractivenes_to_the_technology_performance 0.5
 set Rate_of_contact_between_customers_and_means_of_marketing 0.12
 set Adoption_rate 0
 set Discar_rate 0
 set Early_adopters_fraction 0.002
 ;; initialize stock values
 set Marketing_performance 0
 set Stock_of_knowledge 1
 set Stock_of_legitimacy 0
 set Adopters Initial_cumulative_adopters
 set Cumulative_production Initial_production_experience
 set Potential_adopters Population - Adopters
 set Price Initial_price
end

;; Step through the system dynamics model by performing next iteration of Euler's
method.
;; Call this in your model's GO procedure.
to system-dynamics-go

 ;; compute variable and flow values once per step
 let local-Learning Learning
 let local-Learning_curve_exponent Learning_curve_exponent
 let local-Unit_variable_cost Unit_variable_cost
 let local-Unit_fixed_cost Unit_fixed_cost
 let local-Initial_unit_fixed_cost Initial_unit_fixed_cost
 let local-Initial_unit_variable_cost Initial_unit_variable_cost
 let local-Unit_cost Unit_cost
 let local-Cost_price Cost_price
 let local-Target_price Target_price
 let local-Profit_margin Profit_margin
 let local-Investment_on_marketing Investment_on_marketing
 let local-Total_investment Total_investment
 let local-Investment_on_rd Investment_on_rd
 let local-Share_of_investment_on_rd Share_of_investment_on_rd
 let local-Profit Profit
 let local-Revenue Revenue
 let local-Total_cost Total_cost
 let local-Performance_improvement Performance_improvement
              let              local-Sensivity_of_performance_per_cumulative_knowledge
Sensivity_of_performance_per_cumulative_knowledge
```

```
 let local-Atractiveness_from_price Atractiveness_from_price
 let local-Atractiveness_from_performance Atractiveness_from_performance
 let local-Total_attractiveness_of_technology Total_attractiveness_of_technology
 let local-Growth_of_legitimacy_through_wom Growth_of_legitimacy_through_wom
 let local-Ratio_of_adopters_to_population_alpha Ratio_of_adopters_to_population_alpha
                    let                    local-Growth_of_legitimacy_thorugh_marketing
Growth_of_legitimacy_thorugh_marketing
 let local-Marketing_efectiveness Marketing_efectiveness
 let local-Wom_effectiveness Wom_effectiveness
 let local-Production Production
 let local-Change_in_price Change_in_price
                    let                    local-Marketing_performance_improvement_rate
Marketing_performance_improvement_rate
 let local-Rate_of_increasing_in_knowledge Rate_of_increasing_in_knowledge
 let local-Rate_of_growth_of_legitimacy Rate_of_growth_of_legitimacy

 ;; update stock values
 ;; use temporary variables so order of computation doesn't affect result.
     let     new-Marketing_performance     (     Marketing_performance     +     local-
Marketing_performance_improvement_rate )
      let      new-Stock_of_knowledge      (      Stock_of_knowledge      +      local-
Rate_of_increasing_in_knowledge )
 let new-Stock_of_legitimacy ( Stock_of_legitimacy + local-Rate_of_growth_of_legitimacy
)
 let new-Adopters ( Adopters )
 let new-Cumulative_production ( Cumulative_production + local-Production )
 let new-Potential_adopters ( Potential_adopters )
 let new-Price ( Price + local-Change_in_price )
 set Marketing_performance new-Marketing_performance
 set Stock_of_knowledge new-Stock_of_knowledge
 set Stock_of_legitimacy new-Stock_of_legitimacy
 set Adopters new-Adopters
 set Cumulative_production new-Cumulative_production
 set Potential_adopters new-Potential_adopters
 set Price new-Price

 tick-advance dt
end

;; Report value of flow
to-report Production
 report ( Number_of_product_per_adopter * Adoption_rate * ( 1 / dt )
 ) * dt
end

;; Report value of flow
to-report Change_in_price
 report ( ( Target_price - Price ) / Adjustment_time
 ) * dt
end

;; Report value of flow
to-report Marketing_performance_improvement_rate
 report ( Investment_on_marketing * Marketing_performance_improvement_per_investment
 ) * dt
end
```

```
;; Report value of flow
to-report Rate_of_increasing_in_knowledge
 report ( Knowledge_increment_per_investment * Investment_on_RD
 ) * dt
end

;; Report value of flow
to-report Rate_of_growth_of_legitimacy
 report ( Growth_of_legitimacy_thorugh_marketing + Growth_of_legitimacy_through_wom
 ) * dt
end

;; Report value of variable
to-report Learning
    report    (    Cumulative_production    /    Initial_production_experience    )    ^
Learning_curve_exponent
end

;; Report value of variable
to-report Learning_curve_exponent
 report ( ln Progress_ratio ) / ( ln 2 )
end

;; Report value of variable
to-report Unit_variable_cost
 report Initial_unit_variable_cost * Learning
end

;; Report value of variable
to-report Unit_fixed_cost
 report Initial_unit_fixed_cost * Learning
end

;; Report value of variable
to-report Initial_unit_fixed_cost
 report ( Initial_price / ( 1 + Desired_profit_margin ) ) * ( Fixed_variable_cost / ( 1
+ Fixed_variable_cost ) )
end

;; Report value of variable
to-report Initial_unit_variable_cost
  report  (  Initial_price  /  (  1  +  Desired_profit_margin  )  )  *  (  1  /  (  1  +
Fixed_variable_cost ) )
end

;; Report value of variable
to-report Unit_cost
 report Unit_variable_cost + Unit_fixed_cost / Capacity_utilization
end

;; Report value of variable
to-report Cost_price
 report ( 1 + Desired_profit_margin ) * Unit_cost
end

;; Report value of variable
to-report Target_price
```

```
 report Cost_price * Ratio_of_target_price_to_cost_price
end

;; Report value of variable
to-report Profit_margin
 report ( Price / Unit_cost ) - 1
end

;; Report value of variable
to-report Investment_on_marketing
 report Total_investment * Share_of_investment_on_marketing
end

;; Report value of variable
to-report Total_investment
 report Profit * Share_of_profit_on_investment
end

;; Report value of variable
to-report Investment_on_rd
 report Total_investment * Share_of_investment_on_rd
end

;; Report value of variable
to-report Share_of_investment_on_rd
 report 0.5 - Share_of_investment_on_marketing
end

;; Report value of variable
to-report Profit
 report Revenue - Total_cost
end

;; Report value of variable
to-report Revenue
 report Price * Production * ( 1 / dt )
end

;; Report value of variable
to-report Total_cost
 report Production * Unit_cost * ( 1 / dt )
end

;; Report value of variable
to-report Performance_improvement
 report ( Stock_of_knowledge ^ Sensivity_of_performance_per_cumulative_knowledge )   *
Contribution_share_of_knowledge_Acumulario_in_performance_improvement
end

;; Report value of variable
to-report Sensivity_of_performance_per_cumulative_knowledge
 report ln ( 1 + Performance_improvement_per_doubling_of_knowledge ) / ln ( 2 )
end

;; Report value of variable
to-report Atractiveness_from_price
 report exp ( ( Sensivity_of_atractiveness_to_the_price * Price ) / Initial_price )
```

```
end

;; Report value of variable
to-report Atractiveness_from_performance
        report        Sensivity_of_attractivenes_to_the_technology_performance        *
Performance_improvement
end

;; Report value of variable
to-report Total_attractiveness_of_technology
 report Atractiveness_from_price + Atractiveness_from_performance
end

;; Report value of variable
to-report Growth_of_legitimacy_through_wom
   report   Wom_effectiveness   *   Ratio_of_adopters_to_population_alpha   *   (   1   +
Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Ratio_of_adopters_to_population_alpha
 report Adopters / Population
end

;; Report value of variable
to-report Growth_of_legitimacy_thorugh_marketing
 report Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Marketing_efectiveness
                    report                    Marketing_performance                    *
Rate_of_contact_between_customers_and_means_of_marketing
end

;; Report value of variable
to-report Wom_effectiveness
 report contact_rate * Adoption_fraction
end

;; Plot the current state of the system dynamics model's stocks
;; Call this procedure in your plot's update commands.
to system-dynamics-do-plot
 if plot-pen-exists? "Marketing_performance" [
  set-current-plot-pen "Marketing_performance"
  plotxy ticks Marketing_performance
 ]
 if plot-pen-exists? "Stock_of_knowledge" [
  set-current-plot-pen "Stock_of_knowledge"
  plotxy ticks Stock_of_knowledge
 ]
 if plot-pen-exists? "Stock_of_legitimacy" [
  set-current-plot-pen "Stock_of_legitimacy"
  plotxy ticks Stock_of_legitimacy
 ]
 if plot-pen-exists? "Adopters" [
  set-current-plot-pen "Adopters"
```

```
  plotxy ticks Adopters
 ]
 if plot-pen-exists? "Cumulative_production" [
  set-current-plot-pen "Cumulative_production"
  plotxy ticks Cumulative_production
 ]
 if plot-pen-exists? "Potential_adopters" [
  set-current-plot-pen "Potential_adopters"
  plotxy ticks Potential_adopters
 ]
 if plot-pen-exists? "Price" [
  set-current-plot-pen "Price"
  plotxy ticks Price
 ]
end

globals [ file ]

breed [ tadopters tadopter ] ;; tortugas clientes
breed [ tpotentials tpotential ] ;; tortugas clientes potenciales

patches-own [ advertising intensity ]

tadopters-own [
 duration
 homex
 homey
]

tpotentials-own[
 class ; Tipo de agente seg�n la teoria de E.Rogers
 homex
 homey
 legitimacy-wom
 legitimacy-marketing
]


;;;
```

```
to setupsliders
 set Progress_ratio S_progress_ratio
 set Capacity_utilization S_capacity_utilization
 set Ratio_of_target_price_to_cost_price S_ratio_of_target_price_to_cost_price
end


;;procedimiento que crea los agentes y baldosas
to setup-agents
 set-default-shape turtles "person"
 if (file-exists? file) [file-delete file]
 ask n-of Adopters patches [
  sprout-tadopters 1
  [
    set color red
    set homex xcor
    set homey ycor
  ]
 ]
 ask patches [ set advertising false]
 ask n-of ( Population * 0.10 ) patches [
  set pcolor green
  set advertising true
  set intensity random-float 2
 ]
 create-tpotentials ( Population - Adopters )
 [
   set color sky
   setxy random-xcor random-ycor
   set homex xcor
   set homey ycor
   set class "none"
 ]
 ask n-of ( Potential_adopters * 0.135 ) tpotentials
 [
  set legitimacy-wom 0.54
  set legitimacy-marketing 0.5
  set class "primeros"
 ]
 ask n-of ( Potential_adopters * 0.33) tpotentials with [class = "none" ]
 [
  set legitimacy-wom random adoptionlevel * 57 / 100
  set legitimacy-marketing 0.3
  set class "precoces"
 ]
 ask n-of ( Potential_adopters * 0.33 ) tpotentials with [class = "none" ]
 [
  set legitimacy-wom -0.3
  set legitimacy-marketing -0.2
  set class "tardios"
 ]
 ask n-of ( Potential_adopters * 0.16 ) tpotentials with [class = "none" ]
 [
  set legitimacy-wom -1.1
  set legitimacy-marketing -1.1
  set class "rezagados"
 ]
```

```
end
;;;
```

```
 set-current-plot "Cumulative_production" plot Cumulative_production
 set-current-plot "Total_cost" plot Total_cost
 set-current-plot "Cost_price" plot Cost_price
 set-current-plot "Price" plot Price
 set-current-plot "Marketing_performance" plot Marketing_performance
 set-current-plot "Profit" plot Profit
 set-current-plot "Investment_on_marketing" plot Investment_on_marketing
 set-current-plot "Revenue" plot Revenue
 set-current-plot "Performance_improvement" plot Performance_improvement
 set-current-plot "Atractiveness_from_price " plot Atractiveness_from_price
          set-current-plot          "Total_attractiveness_of_technology"          plot
Total_attractiveness_of_technology
 set-current-plot "Rate_of_growth_of_legitimacy" plot Rate_of_growth_of_legitimacy * (
1 / dt )
           set-current-plot           "Growth_of_legitimacy_through_wom"           plot
Growth_of_legitimacy_through_wom
          set-current-plot         "Ratio_of_adopters_to_population_alpha"         plot
Ratio_of_adopters_to_population_alpha
 set-current-plot "Wom_effectiveness" plot Wom_effectiveness
 escribe-csv
end




to contact
 let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
 let contactos tpotentials with [ any? neighbors with [ any? tadopters-here ] ]
 ask contactos [
       set    legitimacy-wom    (legitimacy-wom    +    Wom_effectiveness    *    (    1    +
Total_attractiveness_of_technology ) * dt)
 ]
 ;Calculo un valor de una variable SD suma de agentes
  let  convertidos  count  contactos   with  [  legitimacy-wom  +  legitimacy-marketing  >
adoptionlevel ]
 ask contactos with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
 [
   set breed tadopters set color red
 ]
 ;Enviamos valores al modelo SD
 set Adoption_rate convertidos + tempranos
 set Adopters count tadopters
end

to amortizate
 ask tadopters [
  set duration ( duration + 1 * dt  )
 ]
end


to marketing

 ask patches with [ advertising = true ]
```

```
 [

  let pat-intensity [ intensity ] of self
  ask tpotentials in-radius 1
  [
         set   legitimacy-marketing   (   legitimacy-marketing   +   pat-intensity   *
Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology ) * dt )
  ]
 ]
end

to discard

 let  cuantos  n-of  (  (  count  tadopters)  /  (  Technology_life_time  *  (  1  /  dt  )  ))
tadopters
 let discartes cuantos
 ask discartes[
  set breed tpotentials
  set color sky
  set legitimacy-wom random adoptionlevel * 99 / 100 ;set legitimacy-wom adoptionlevel
  set class "repetidores"
 ]
 set Discar_rate count discartes
 set Potential_Adopters count tpotentials

end


to escribe-csv


 file-open file
 file-print (word ticks ","  Adopters)
 ;file-print  (word ticks ","  ( Adoption_rate * ( 1 / dt ) ))
 file-close
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; END ITERACCIONES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

14 ANEXO III – Modelo híbrido de difusión de la innovación con soporte GIS

```
;; System dynamics model globals
globals [
 ;; constants
 Population
 Adoption_fraction
 Contact_rate
 Technology_life_time
 Initial_cumulative_adopters
 Progress_ratio
 Number_of_product_per_adopter
 Initial_production_experience
 Fixed_variable_cost
 Initial_price
 Desired_profit_margin
 Capacity_utilization
 Ratio_of_target_price_to_cost_price
 Adjustment_time
 Marketing_performance_improvement_per_investment
 Share_of_investment_on_marketing
 Share_of_profit_on_investment
 Knowledge_increment_per_investment
 Performance_improvement_per_doubling_of_knowledge
 Contribution_share_of_knowledge_Acumulario_in_performance_improvement
 Sensivity_of_atractiveness_to_the_price
 Sensivity_of_attractivenes_to_the_technology_performance
 Rate_of_contact_between_customers_and_means_of_marketing
 Adoption_rate
 Discar_rate
 Early_adopters_fraction
 ;; stock values
 Marketing_performance
 Stock_of_knowledge
 Stock_of_legitimacy
 Adopters
 Cumulative_production
 Potential_adopters
 Price
 ;; size of each step, see SYSTEM-DYNAMICS-GO
 dt
]

;; Initializes the system dynamics model.
;; Call this in your model's SETUP procedure.
to system-dynamics-setup
 reset-ticks
 set dt 0.0625
```

```
 ;; initialize constant values
 set Population 10000
 set Adoption_fraction 0.01
 set Contact_rate 50
 set Technology_life_time 8
 set Initial_cumulative_adopters 100
 set Progress_ratio 0.8
 set Number_of_product_per_adopter 1
 set Initial_production_experience 1000
 set Fixed_variable_cost 3
 set Initial_price 1000
 set Desired_profit_margin 0.2
 set Capacity_utilization 1
 set Ratio_of_target_price_to_cost_price 1.01
 set Adjustment_time 0.5
 set Marketing_performance_improvement_per_investment 0.00001
 set Share_of_investment_on_marketing 0.2
 set Share_of_profit_on_investment 0.4
 set Knowledge_increment_per_investment 0.000001
 set Performance_improvement_per_doubling_of_knowledge 0.5
 set Contribution_share_of_knowledge_Acumulario_in_performance_improvement 0.5
 set Sensivity_of_atractiveness_to_the_price -2
 set Sensivity_of_attractivenes_to_the_technology_performance 0.5
 set Rate_of_contact_between_customers_and_means_of_marketing 0.12
 set Adoption_rate 0
 set Discar_rate 0
 set Early_adopters_fraction 0.002
 ;; initialize stock values
 set Marketing_performance 0
 set Stock_of_knowledge 1
 set Stock_of_legitimacy 0
 set Adopters Initial_cumulative_adopters
 set Cumulative_production Initial_production_experience
 set Potential_adopters Population - Adopters
 set Price Initial_price
end

;; Step through the system dynamics model by performing next iteration of Euler's
method.
;; Call this in your model's GO procedure.
to system-dynamics-go

 ;; compute variable and flow values once per step
 let local-Learning Learning
 let local-Learning_curve_exponent Learning_curve_exponent
 let local-Unit_variable_cost Unit_variable_cost
 let local-Unit_fixed_cost Unit_fixed_cost
 let local-Initial_unit_fixed_cost Initial_unit_fixed_cost
 let local-Initial_unit_variable_cost Initial_unit_variable_cost
 let local-Unit_cost Unit_cost
 let local-Cost_price Cost_price
 let local-Target_price Target_price
 let local-Profit_margin Profit_margin
 let local-Investment_on_marketing Investment_on_marketing
 let local-Total_investment Total_investment
 let local-Investment_on_rd Investment_on_rd
 let local-Share_of_investment_on_rd Share_of_investment_on_rd
```

```
 let local-Profit Profit
 let local-Revenue Revenue
 let local-Total_cost Total_cost
 let local-Performance_improvement Performance_improvement
              let              local-Sensivity_of_performance_per_cumulative_knowledge
Sensivity_of_performance_per_cumulative_knowledge
 let local-Atractiveness_from_price Atractiveness_from_price
 let local-Atractiveness_from_performance Atractiveness_from_performance
 let local-Total_attractiveness_of_technology Total_attractiveness_of_technology
 let local-Growth_of_legitimacy_through_wom Growth_of_legitimacy_through_wom
 let local-Ratio_of_adopters_to_population_alpha Ratio_of_adopters_to_population_alpha
                    let                    local-Growth_of_legitimacy_thorugh_marketing
Growth_of_legitimacy_thorugh_marketing
 let local-Marketing_efectiveness Marketing_efectiveness
 let local-Wom_effectiveness Wom_effectiveness
 let local-Production Production
 let local-Change_in_price Change_in_price
                    let                    local-Marketing_performance_improvement_rate
Marketing_performance_improvement_rate
 let local-Rate_of_increasing_in_knowledge Rate_of_increasing_in_knowledge
 let local-Rate_of_growth_of_legitimacy Rate_of_growth_of_legitimacy

 ;; update stock values
 ;; use temporary variables so order of computation doesn't affect result.
     let     new-Marketing_performance     (     Marketing_performance     +     local-
Marketing_performance_improvement_rate )
      let      new-Stock_of_knowledge      (      Stock_of_knowledge      +      local-
Rate_of_increasing_in_knowledge )
 let new-Stock_of_legitimacy ( Stock_of_legitimacy + local-Rate_of_growth_of_legitimacy
)
 let new-Adopters ( Adopters )
 let new-Cumulative_production ( Cumulative_production + local-Production )
 let new-Potential_adopters ( Potential_adopters )
 let new-Price ( Price + local-Change_in_price )
 set Marketing_performance new-Marketing_performance
 set Stock_of_knowledge new-Stock_of_knowledge
 set Stock_of_legitimacy new-Stock_of_legitimacy
 set Adopters new-Adopters
 set Cumulative_production new-Cumulative_production
 set Potential_adopters new-Potential_adopters
 set Price new-Price

 tick-advance dt
end

;; Report value of flow
to-report Production
 report ( Number_of_product_per_adopter * Adoption_rate * ( 1 / dt )
 ) * dt
end

;; Report value of flow
to-report Change_in_price
 report ( ( Target_price - Price ) / Adjustment_time
 ) * dt
end
```

```
;; Report value of flow
to-report Marketing_performance_improvement_rate
 report ( Investment_on_marketing * Marketing_performance_improvement_per_investment
 ) * dt
end

;; Report value of flow
to-report Rate_of_increasing_in_knowledge
 report ( Knowledge_increment_per_investment * Investment_on_RD
 ) * dt
end

;; Report value of flow
to-report Rate_of_growth_of_legitimacy
 report ( Growth_of_legitimacy_thorugh_marketing + Growth_of_legitimacy_through_wom
 ) * dt
end

;; Report value of variable
to-report Learning
    report    (    Cumulative_production    /    Initial_production_experience    )    ^
Learning_curve_exponent
end

;; Report value of variable
to-report Learning_curve_exponent
 report ( ln Progress_ratio ) / ( ln 2 )
end

;; Report value of variable
to-report Unit_variable_cost
 report Initial_unit_variable_cost * Learning
end

;; Report value of variable
to-report Unit_fixed_cost
 report Initial_unit_fixed_cost * Learning
end

;; Report value of variable
to-report Initial_unit_fixed_cost
 report ( Initial_price / ( 1 + Desired_profit_margin ) ) * ( Fixed_variable_cost / ( 1
+ Fixed_variable_cost ) )
end

;; Report value of variable
to-report Initial_unit_variable_cost
  report  (  Initial_price  /  (  1  +  Desired_profit_margin  )  )  *  (  1  /  (  1  +
Fixed_variable_cost ) )
end

;; Report value of variable
to-report Unit_cost
 report Unit_variable_cost + Unit_fixed_cost / Capacity_utilization
end

;; Report value of variable
```

```
to-report Cost_price
 report ( 1 + Desired_profit_margin ) * Unit_cost
end

;; Report value of variable
to-report Target_price
 report Cost_price * Ratio_of_target_price_to_cost_price
end

;; Report value of variable
to-report Profit_margin
 report ( Price / Unit_cost ) - 1
end

;; Report value of variable
to-report Investment_on_marketing
 report Total_investment * Share_of_investment_on_marketing
end

;; Report value of variable
to-report Total_investment
 report Profit * Share_of_profit_on_investment
end

;; Report value of variable
to-report Investment_on_rd
 report Total_investment * Share_of_investment_on_rd
end

;; Report value of variable
to-report Share_of_investment_on_rd
 report 0.5 - Share_of_investment_on_marketing
end

;; Report value of variable
to-report Profit
 report Revenue - Total_cost
end

;; Report value of variable
to-report Revenue
 report Price * Production * ( 1 / dt )
end

;; Report value of variable
to-report Total_cost
 report Production * Unit_cost * ( 1 / dt )
end

;; Report value of variable
to-report Performance_improvement
 report ( Stock_of_knowledge ^ Sensivity_of_performance_per_cumulative_knowledge )   *
Contribution_share_of_knowledge_Acumulario_in_performance_improvement
end

;; Report value of variable
to-report Sensivity_of_performance_per_cumulative_knowledge
```

```
 report ln ( 1 + Performance_improvement_per_doubling_of_knowledge ) / ln ( 2 )
end

;; Report value of variable
to-report Atractiveness_from_price
 report exp ( ( Sensivity_of_atractiveness_to_the_price * Price ) / Initial_price )
end

;; Report value of variable
to-report Atractiveness_from_performance
        report        Sensivity_of_attractivenes_to_the_technology_performance        *
Performance_improvement
end

;; Report value of variable
to-report Total_attractiveness_of_technology
 report Atractiveness_from_price + Atractiveness_from_performance
end

;; Report value of variable
to-report Growth_of_legitimacy_through_wom
   report   Wom_effectiveness   *   Ratio_of_adopters_to_population_alpha   *   (   1   +
Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Ratio_of_adopters_to_population_alpha
 report Adopters / Population
end

;; Report value of variable
to-report Growth_of_legitimacy_thorugh_marketing
 report Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology )
end

;; Report value of variable
to-report Marketing_efectiveness
                    report                    Marketing_performance                    *
Rate_of_contact_between_customers_and_means_of_marketing
end

;; Report value of variable
to-report Wom_effectiveness
 report contact_rate * Adoption_fraction
end

;; Plot the current state of the system dynamics model's stocks
;; Call this procedure in your plot's update commands.
to system-dynamics-do-plot
 if plot-pen-exists? "Marketing_performance" [
  set-current-plot-pen "Marketing_performance"
  plotxy ticks Marketing_performance
 ]
 if plot-pen-exists? "Stock_of_knowledge" [
  set-current-plot-pen "Stock_of_knowledge"
  plotxy ticks Stock_of_knowledge
 ]
```

```
 if plot-pen-exists? "Stock_of_legitimacy" [
  set-current-plot-pen "Stock_of_legitimacy"
  plotxy ticks Stock_of_legitimacy
 ]
 if plot-pen-exists? "Adopters" [
  set-current-plot-pen "Adopters"
  plotxy ticks Adopters
 ]
 if plot-pen-exists? "Cumulative_production" [
  set-current-plot-pen "Cumulative_production"
  plotxy ticks Cumulative_production
 ]
 if plot-pen-exists? "Potential_adopters" [
  set-current-plot-pen "Potential_adopters"
  plotxy ticks Potential_adopters
 ]
 if plot-pen-exists? "Price" [
  set-current-plot-pen "Price"
  plotxy ticks Price
 ]
end





extensions [ gis ]

globals [ file distritos contador area-paches]

breed [ tadopters tadopter ] ;; tortugas clientes
breed [ tpotentials tpotential ] ;; tortugas clientes potenciales

patches-own [ advertising intensity poblacion area densidad provincia ]

tadopters-own [
 duration
 homex
 homey

]

tpotentials-own[
 class ; Tipo de agente seg?n la teoria de E.Rogers
 homex
 homey
 legitimacy-wom
 legitimacy-marketing
 condt
]


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  SETUP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
to lit
 clear-ticks
 clear-turtles
 clear-all-plots
 ask patches with [ advertising = true ]
 [
  set pcolor 0
  set advertising false
 ]
end

to setup
 if ( area-paches = 0 ) [ loadgis ]
 lit
 setup-globals
 system-dynamics-setup
 setup-agents
 setupsliders
end

to setup-globals
 set file "basssketch-7.csv"

end

to setupsliders
 set Progress_ratio S_progress_ratio
 set Capacity_utilization S_capacity_utilization
 set Ratio_of_target_price_to_cost_price S_ratio_of_target_price_to_cost_price
end


;;procedimiento que crea los agentes y baldosas
to setup-agents
 let current-agent 0
 ;set-default-shape turtles "person"
 if (file-exists? file) [file-delete file]
 ; Coloco a los seguidores iniciales en Madrid
 ;Madrid 3207247
 ;Mallorca provincia 7
 ;Mallorca 10144
 ;Barcelona 1611822 provincia 8
 ask n-of Adopters patches with [ area > 0 ] [
  sprout-tadopters 1 [ customiza-tadopters ]
 ]



 ask patches [ set advertising false]
  ;Colocamos las baldosas publicitarias de una forma pseudoaleatoria en función de la
densidad de la población
 ask n-of ( Population * 0.10 ) patches with [ densidad > publicity_denstity ] [
  if (is-number? poblacion)[
   if (poblacion > 0 and area > 0)[
    set pcolor green
    set advertising true
```

```
    set intensity random-float 2
   ]
  ]
 ]
 ;Colocamos los agentes por el mapa aleatoriamente en función de la densidad de la
población
  ask patches[
  ;al contratio que en load gis población tenía nulos (después los quite con qgis) en
este caso los he quitado con qgis no necesito ifs
  if (is-number? poblacion ) [
   if ( poblacion > 0 and area > 0)[
   let numagentes ( densidad * area-paches / ( Population ) )
   sprout-tpotentials (floor numagentes ) [ customiza-tpotentials]
   let decimal ( numagentes - (floor numagentes))
   if (decimal > random-float 1) [sprout-tpotentials 1 [ customiza-tpotentials ]]
   ]
  ]
 ]
 ask n-of ( population - count tadopters - count tpotentials) patches with [ area > 0 ]
[
  sprout-tpotentials 1 [ customiza-tpotentials ]
 ]


end

to customiza-tadopters
 set size 1
 set color red
 set homex xcor
 set homey ycor
end

to customiza-tpotentials
 set size 1
 set color blue
 set homex xcor
 set homey ycor
 set condt 0
 let class-prob (random 1000)
 ifelse (class-prob < 136)
 [
  set legitimacy-wom 0.54
  set legitimacy-marketing 0.5
  set class "primeros"
 ]
 [
  ifelse (class-prob >= 135 and class-prob < 465 )
  [
   set legitimacy-wom random adoptionlevel * 57 / 100
   set legitimacy-marketing 0.3
   set class "precoces"
  ]
  [
   ifelse (class-prob >= 465 and class-prob <= 795 )
   [
    set legitimacy-wom -0.3
```

```
    set legitimacy-marketing -0.2
    set class "tardios"
   ]
   [
    if (class-prob >= 840 )
    [
      set legitimacy-wom -1.1
      set legitimacy-marketing -1.1
      set class "rezagados"
    ]


   ]
  ]

 ]
end

to loadgis
 gis:load-coordinate-system "resultado.prj"
 set distritos gis:load-dataset "resultado.shp"
 gis:set-drawing-color white
 gis:draw distritos 1
 gis:apply-coverage distritos "MUNICIPI_5" poblacion
 gis:apply-coverage distritos "MUNICIPI_6" area
 gis:apply-coverage distritos "PROVINCIA" provincia
 ask patches
 [if (is-number? poblacion )
  [ ifelse (poblacion > 0 and area > 0)
   [
    set pcolor scale-color red poblacion 5000 10
    set densidad ( poblacion / area )
    set contador contador + poblacion

   ]
   [ set pcolor blue
    set densidad 0
   ]

  ]

 ]
 ;metros 49720870
 ;poblacion
 set area-paches ( 248972156 / count patches)
 print  "Total area="
 print  contador
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  END SETUP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;  ITERACCIONES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to go
 if ticks >= 25 [ stop ]
 agent-go
 set-current-plot "Adoption"
 system-dynamics-go
 system-dynamics-do-plot
 graph

end


to agent-go
  move-turtles
  marketing
  contact
  amortizate
  discard

end

to move-turtles
 ask turtles[
  ifelse abs(xcor - homex) < homed or abs(ycor - homey) < homed
  [
    right random 360
  ]
  [
    facexy homex homey
  ]
  let poblacionactual [ poblacion ] of patch-here
  let enfrente [ poblacion ] of patch-ahead 1
  if (is-number? poblacion) and (is-number? enfrente) [
   if enfrente > 0
   [

      forward 1

   ]
  ]
 ]
end



;;procedimiento para actualizar las gr?ficas
to graph
 set-current-plot "Rate" plot Adoption_rate * (1 / dt )
 set-current-plot "Cumulative_production" plot Cumulative_production
 set-current-plot "Total_cost" plot Total_cost
 set-current-plot "Cost_price" plot Cost_price
 set-current-plot "Price" plot Price
 set-current-plot "Marketing_performance" plot Marketing_performance
```

```
 set-current-plot "Profit" plot Profit
 set-current-plot "Investment_on_marketing" plot Investment_on_marketing
 set-current-plot "Revenue" plot Revenue
 set-current-plot "Performance_improvement" plot Performance_improvement
 set-current-plot "Atractiveness_from_price " plot Atractiveness_from_price
          set-current-plot          "Total_attractiveness_of_technology"          plot
Total_attractiveness_of_technology
 set-current-plot "Rate_of_growth_of_legitimacy" plot Rate_of_growth_of_legitimacy
           set-current-plot           "Growth_of_legitimacy_through_wom"           plot
Growth_of_legitimacy_through_wom
          set-current-plot         "Ratio_of_adopters_to_population_alpha"         plot
Ratio_of_adopters_to_population_alpha
 set-current-plot "Wom_effectiveness" plot Wom_effectiveness
 set-current-plot "Adoption"
 set-current-plot-pen "Euskadi"
 plot count tadopters-on patches with [ provincia = 1 or provincia = 20 or provincia =
41 ]
 set-current-plot-pen "Total"
 plot count tadopters
 set-current-plot-pen "Madrid"
 plot count tadopters-on patches with [ provincia = 28 ]
 escribe-csv
end



to contact
 let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
 let contactos tadopters with [ any? neighbors with [ any? tpotentials-here ] ]
 ask contactos [

  ifelse count [tpotentials-on neighbors] of self > 10 [
   ask [n-of 10 tpotentials-on neighbors ] of self [
    if ( condt < 2 )[
             set   legitimacy-wom   (legitimacy-wom   +   Wom_effectiveness   *   (   1   +
Total_attractiveness_of_technology ) * dt)
     set condt condt + 1
    ]
   ]
  ]
  [
   ask [ tpotentials-on neighbors ] of self [
    if ( condt < 1 )[
             set   legitimacy-wom   (legitimacy-wom   +   Wom_effectiveness   *   (   1   +
Total_attractiveness_of_technology ) * dt)
     set condt condt + 1
    ]
   ]


  ]
  ]
 ;Calculo un valor de una variable SD suma de agentes
 let convertidos count tpotentials   with [ legitimacy-wom + legitimacy-marketing >
```

```
adoptionlevel ]
 ask tpotentials with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
 [
   set breed tadopters set color red
 ]
 ;Enviamos valores al modelo SD
 set Adoption_rate convertidos + tempranos
 set Adopters count tadopters
 ;Reseteo condt
 ask tpotentials
 [
  set condt 0
 ]

end



to contact1
 let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
 let listadoptadores [self] of tadopters with [ any? neighbors with [ any? tpotentials-
here ]]
 foreach listadoptadores
 [
  ifelse count ([ tpotentials-on neighbors ] of ? ) > 8
  [
    ask n-of 7 [ tpotentials-on neighbors ] of ?
    [
              set  legitimacy-wom  (legitimacy-wom  +  Wom_effectiveness  *  (  1  +
Total_attractiveness_of_technology ) * dt)
    ]
  ]
  [
    ask [ tpotentials-on neighbors ] of ?
    [
              set  legitimacy-wom  (legitimacy-wom  +  Wom_effectiveness  *  (  1  +
Total_attractiveness_of_technology ) * dt)
    ]

  ]
 ]
 ;Calculo un valor de una variable SD suma de agentes
  let convertidos count tpotentials  with [ legitimacy-wom + legitimacy-marketing >
adoptionlevel ]
 ask tpotentials with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
 [
   set breed tadopters set color red
 ]
 ;Enviamos valores al modelo SD

 set Adoption_rate convertidos + tempranos
 set Adopters count tadopters
end
```

```
to contact0
 let tempranos Early_adopters_fraction * count tpotentials * dt
 ask n-of tempranos tpotentials [
   set breed tadopters set color red
 ]
 let contactos tpotentials with [ any? neighbors with [ any? tadopters-here ] ]
 ask contactos [
       set   legitimacy-wom   (legitimacy-wom   +   Wom_effectiveness   *   (   1   +
Total_attractiveness_of_technology ) * dt)
 ]
 ;Calculo un valor de una variable SD suma de agentes
  let  convertidos  count  contactos   with  [  legitimacy-wom  +  legitimacy-marketing  >
adoptionlevel ]
 ask contactos with [ legitimacy-wom + legitimacy-marketing > adoptionlevel ]
 [
   set breed tadopters set color red
 ]
 ;Enviamos valores al modelo SD
 set Adoption_rate convertidos + tempranos
 set Adopters count tadopters
end


to amortizate
 ask tadopters [
  set duration ( duration + 1 * dt  )
 ]
end


to marketing

 ask patches with [ advertising = true ]
 [

  let pat-intensity [ intensity ] of self
  ask tpotentials in-radius 1
  [
         set   legitimacy-marketing   (   legitimacy-marketing   +   pat-intensity   *
Marketing_efectiveness * ( 1 + Total_attractiveness_of_technology ) * dt )
  ]
 ]
end

to discard

 let  cuantos  n-of  (  (  count  tadopters)  /  (  Technology_life_time  *  (  1  /  dt  )  ))
tadopters
 let discartes cuantos
 ask discartes[
  set breed tpotentials
  set color sky
  set legitimacy-wom random adoptionlevel * 99 / 100 ;set legitimacy-wom adoptionlevel
  set class "repetidores"
```

```
 ]
 set Discar_rate count discartes
 set Potential_Adopters count tpotentials

end


to escribe-csv


 file-open file
 file-print (word ticks ","  Adopters)
 ;file-print  (word ticks ","  ( Adoption_rate * ( 1 / dt ) ))
 file-close
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; END ITERACCIONES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```