SIMULACIÓN DE MODELOS DEMOGRÁFICOS BASADOS EN AGENTES

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA, UNIVERSIDAD COMPLUTENSE DE MADRID
Curso 2011-12

Trabajo Fin de Máster en Sistemas Inteligentes
9 de Septiembre de 2012

Helio Alejandro Domínguez Bayo

Director:
Juan Pavón Mestras
Colaborador externo:
Samer Hassan

Convocatoria: Septiembre 2012
Calificación: Sobresaliente

AUTORIZACIÓN DE DIFUSIÓN

HELIO ALEJANDRO DOMÍNGUEZ BAYO

9 de Septiembre de 2012

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "SIMULACIÓN DE MODELOS DEMOGRÁFICOS BASADOS EN AGENTES", realizado durante el curso académico 2011-2012 bajo la dirección de Juan Pavón Mestras y con la colaboración externa de dirección de Samer Hassan en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

RESUMEN

El creciente interés en la simulación social basada en agentes, con modelos cada vez más complejos y ambiciosos, precisa de desarrollar modelos demográficos flexibles y realistas. En el presente trabajo se aborda la tarea del estudio de los requisitos elementales que dichos modelos demográficos basados en agentes deben cumplir. Se elabora una arquitectura general que se adapte y satisfaga estos requisitos, bajo los principios de modularidad, adaptabilidad y extensibilidad. Se ha dotado a dicha arquitectura de los procesos demográficos elementales: nacimientos, muertes y migraciones, además de la formación de parejas. Se ha tenido en cuenta también el amplio espectro de problemas al que se deberá adaptar la arquitectura y así estos procesos se han contemplado para poder modelarse desde un enfoque basado en reglas de comportamiento o desde una aproximación de arriba abajo. Se ha prestado especial atención a las causas sociales de la demografía, por lo que se ha contemplado la red social del modelo. Además, se aprovecha dicha arquitectura para replicar un conocido modelo demográfico basado en agentes, Artificial Anasazi, a la vez que se examinan y analizan los resultados obtenidos.

Palabras clave: simulación social basada en agentes, sistemas multiagente, demografía, modelo demográfico, replicación de modelos, Anasazi.

ABSTRACT

The ongoing interest in agent-based social simulation, with models increasingly complex and ambitious, creates the need to develop realistic and flexible demographic models. This work focuses on this need, as well as on the basic requirements that agent-based demographic models should fulfill. A software architecture is designed to fit and satisfy these requirements under the adaptability, modularity and extensibility principles. Fundamental demographic processes such as births, deaths and migrations have been considered, as well as marriage and coupling processes. The wide range of problems the architecture should meet is also considered, and so a two side approach to the modeling of these processes included: based on behavioral rules and a top-down approach as well. Social interaction between agents is given a special attention as a main demographic factor and social networks are included too. In addition, a well known agent-based demographic model, the Artificial Anasazi, is adapted to the new architecture and replicated, while the results are afterwards analyzed and discussed.

Keywords: agent-based social simulation, multi-agent systems, demography, demographic model, model replication, Anasazi.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE ABREVIATURAS

ABSS – Agent Based Social Simulation.

KISS – Keep It Simple, Stupid!

ABM – Agent Based Model o Agent Based Modelling

MSM – Microsimulation Model

1. INTRODUCCIÓN

En el estudio de los fenómenos sociales, las técnicas de simulación por ordenador representan cada vez más una manera de adquirir un mayor conocimiento, tanto cuantitativo como cualitativo, sobre dichos fenómenos. En los últimos años, los modelos basados en agentes han venido ganando cada vez más protagonismo a este respecto.

Básicamente, un modelo basado en agentes es un programa informático en el que hay unos entes, los agentes, a los que se dota de reglas de comportamiento, y se les permite interactuar. Es muy sencillo establecer el paralelismo entre los agentes y las personas, y a partir de ahí las posibilidades son infinitas.

Es por ello que hoy en día es muy frecuente que los investigadores de las diversas disciplinas de las ciencias sociales empleen esta técnica, gracias al potencial que tiene para modelar sistemas sociales y permitir comprender las causas y efectos de los mismos.

Dentro del contexto de la simulación social basada en agentes, la demografía representa un aspecto de vital importancia. Las ciencias sociales abarcan una gran cantidad de disciplinas: antropología, historia, geografía, economía, sociología, psicología, demografía, etc. Sin embargo, en muchas ocasiones, trazar las líneas donde unas acaban y otras empiezan resulta difícil, pues todas están interrelacionadas, y es complicado estudiar unas sin tener en cuenta a las otras.

Por lo tanto, el desarrollo de modelos demográficos resulta fundamental dentro del contexto más amplio de la simulación social basada en agentes, pues la demografía está presente, ya sea de forma implícita o explícita, en la práctica totalidad del resto de disciplinas de las ciencias sociales, y en particular dentro de aquellas a las que más atención se les da en el mundo de la simulación basada en agentes. Y, por supuesto, también está presente como una ciencia social más, y como objeto de estudio por sí misma.

Tradicionalmente la demografía ha sido considerada por los propios demógrafos una disciplina fundamentalmente empirista, que ha basado su estudio en el análisis de datos y observaciones, y que ha pecado muchas veces de quedarse en la mera descripción, dejando a un lado la teoría [1]. Sin embargo, desde hace un tiempo, algunos autores vienen reivindicando el papel que la teoría puede cumplir [2], y en concreto los modelos basados en agentes [3], en el enriquecimiento del conocimiento demográfico.

1.1. MOTIVACIÓN

Lo anterior pone de manifiesto la doble importancia que puede tener la demografía dentro del contexto de la simulación basada en agentes. Por un lado forma parte, en mayor o menor medida, de prácticamente todos los procesos sociales. Por otro, hay un creciente interés en el estudio de los propios procesos demográficos desde un punto de vista de la teoría, para lo cual se hace uso de técnicas de modelado basadas en agentes.

Sin embargo, al considerar la literatura, los modelos demográficos basados en agentes son bastante específicos: si son del primer tipo —como parte de otro proceso social—, suelen ser un complemento o módulo dentro de otro modelo basado en agentes más amplio y cuyo objetivo de estudio suele ser diferente del propiamente demográfico. Si son del segundo tipo —que se fijan en el problema demográfico en sí—, se suelen centrar en problemas demasiado concretos y específicos. En cualquier caso, son altamente dependientes de las condiciones particulares del modelo.

La idea de este trabajo surge para tratar de establecer y definir un modelo demográfico que sea capaz de abarcar los diferentes tipos de ABM que incluyen, de una u otra manera, la demografía. Es decir, se pretende construir una arquitectura que permita implementar la mayor cantidad posible de modelos demográficos basados en agentes, y que generalice y siente unas bases sobre las que todo modelo demográfico se deba sostener.

Evidentemente, lo anterior puede resultar no solo pretencioso sino inalcanzable, y más aun al nivel al que está el presente trabajo, pues la tarea puede ser de una dimensión y complejidad enorme. Es por ello que, aunque la motivación sea en principio tan ambiciosa, en este trabajo no se aspirará a más que a trazar unas líneas generales sobre cómo desarrollar un modelo demográfico basado en agentes, de manera que en un futuro puedan ser ampliadas y mejoradas.

1.2. OBJETIVOS DEL TRABAJO

Partiendo de la motivación expuesta anteriormente, se establece como objetivo principal diseñar una arquitectura que permita integrar modelos demográficos basados en agentes de diverso propósito, y que sea lo más general y extensible posible. Para ello, se establecen las siguientes metas:

- Realizar un estudio acerca de los modelos demográficos existentes, tanto basados en agentes como de otro tipo. Extraer las características principales que tienen y requisitos acerca de los datos que manejan. En el caso concreto de los basados en agentes, realizar un estudio también acerca de los atributos que puedan influir en sus decisiones o comportamientos demográficos para establecer un agente demográfico prototipo.

- Definir una arquitectura para modelos demográficos basados en agentes que sea lo más flexible y modular posible, para que se adapte al más amplio espectro posible de modelos.
- Probar la anterior arquitectura mediante un caso de estudio concreto, replicando un conocido modelo demográfico basado en agentes.
- Examinar los resultados obtenidos en la replicación anterior.

1.3. ESTRUCTURA DE LA MEMORIA

Para presentar el trabajo realizado, se ha escrito la presente memoria, organizándose como se explica a continuación:

- En primer lugar, el **Capítulo 2** introduce los principales conceptos demográficos, así como los indicadores sobre la población que más útiles pueden resultar a la hora de realizar modelos demográficos basados en agentes.
- El **Capítulo 3** hace un recorrido por el estado del arte, para tratar de comprender los conceptos clave a la hora de hacer modelos demográficos basados en agentes, además de tratar de definir los requisitos que ha de tener la arquitectura.
- En el **Capítulo 4** se expone la propuesta de arquitectura que se ha realizado, y que trata de cubrir tanto los objetivos planteados en el apartado anterior, como los requisitos extraídos del estudio del estado del arte y de los conceptos demográficos del Capítulo 2.
- Posteriormente, en el **Capítulo 5**, se presenta el caso de estudio realizado: El modelo Artificial Anasazi[1] es descrito en detalle siguiendo el protocolo ODD ([8], [9]) y se comenta el proceso de adaptación a la arquitectura propuesta.
- En el **Capítulo 6** se muestran y analizan las pruebas y resultados obtenidos con el modelo replicado anteriormente.
- Finalmente, el **Capítulo 7** expone las conclusiones del trabajo, poniendo de manifiesto hasta qué punto se han logrado los objetivos, y sugiriendo las líneas futuras por donde debería continuar el trabajo.

[1] Este modelo fue descrito originalmente en [4], [5] y [6]. Posteriormente Janssen lo replicó, y describe su trabajo en [7]. Es esta última descripción la que ha servido de base en la replicación realizada en el presente trabajo.

- Además, se han incluido dos apéndices al final del trabajo. El primero, **Apéndice A**, incluye trozos de código relevante que ha sido empleado a la hora de replicar el modelo del Capítulo 5.
- El **Apéndice B**, expone un modelo desarrollado e implementado en la arquitectura propuesta. El modelo no fue totalmente desarrollado. No obstante, se incluye como apéndice porque puede ilustrar bastantes de los conceptos de diseño de la arquitectura.

2. CONCEPTOS DEMOGRÁFICOS

2.1. ¿QUÉ ES LA DEMOGRAFÍA?

Según la definición de la R.A.E., la demografía es el "estudio estadístico de una colectividad humana, referido a un determinado momento o a su evolución".

Para Livi-Bacci [10], es necesario primero dar una definición de esta colectividad humana, es decir, de la población:

> *A partir de la definición de población puede deducirse una definición de la demografía, la cual estudiaría aquellos procesos que determinan la formación, la conservación y la desaparición de las poblaciones. Tales procesos, en su forma más agregada, son los de fecundidad, mortalidad y movilidad. La variedad de combinaciones de estos fenómenos, interdependientes entre sí, determina la velocidad de las modificaciones de la población, tanto en sus dimensiones numéricas como en su estructura.*

Y define la población como:

> *[…] conjunto de individuos, constituido de forma estable, ligado por vínculos de reproducción e identificado por características territoriales, políticas, jurídicas, étnicas o religiosas. […] continuidad en el tiempo que sólo puede asegurarse mediante la reproducción.*

Resumiendo, y como dice [11], se trata del estudio de la población, entendiendo ésta como "un conjunto de individuos que tienen dimensión temporal y que está asentado sobre un territorio." Añade, posteriormente, que "la dimensión temporal implica una dinámica propia, que es consecuencia de su capacidad de supervivencia y de reproducirse".

Por tanto, queda claro que la población no es un mero *stock* de personas, es un ente dinámico, un sistema reproductivo. Y, aunque el núcleo principal del estudio demográfico lo forman los nacimientos, muertes y entradas y salidas del sistema por las migraciones —componentes del "sistema reproductivo"—, es fundamental también el estudiar las causas que determinan estos tres fenómenos: la composición por edades y las pautas de cada fenómeno en cada una de tales edades, pero también otros comportamientos y características, como la nupcialidad, la salud, la anticoncepción, la regulación legal del aborto, las estructuras familiares y convivenciales, las políticas públicas sobre vivienda o el tratamiento fiscal a las familias [12].

2.2. INDICADORES Y CONCEPTOS MÁS IMPORTANTES EN DEMOGRAFÍA

La información que se presenta a continuación está obtenida de varios manuales básicos sobre demografía ([11], [13] y [14]) y de la página web del Instituto Nacional de Estadística [15]. Se presentan brevemente los conceptos fundamentales de demografía, así como las medidas o indicadores más frecuentes de los mismos. Se ha tratado de seleccionar aquellos que puedan resultar de utilidad a la hora de diseñar un modelo demográfico basado en agentes.

Finalmente, al final de cada apartado se enumeran brevemente otros factores socioeconómicos que bien afecten o se vean afectados por cada uno de los fenómenos demográficos estudiados, pues es información fundamental para saber de qué manera ha de diseñarse un modelo demográfico que integre algunos o todos de estos factores.

2.2.1. EL CRECIMIENTO Y LA ESTRUCTURA DE LA POBLACIÓN

Como sistema dinámico que es la población, su número cambia con el tiempo. Y los elementos que intervienen en este cambio son, únicamente, el número de nacimientos, el número de muertes, el número de personas que emigran y el número de personas que inmigran.

Se define así la ***ecuación fundamental*** o ***ecuación compensadora***, que determina el crecimiento de la población en un intervalo de tiempo *(t,t+1)*[2]:

$$P^{t+1} - P^t = N^{t,t+1} - D^{t,t+1} + I^{t,t+1} - E^{t,t+1}$$

Donde P^{t+1} es la población en el instante *t+1*, P^t la población en el instante *t*, y $N^{t,t+1}$, $D^{t,t+1}$, $I^{t,t+1}$ y $E^{t,t+1}$ son el número de nacimientos, número de defunciones, número de inmigrantes y número de emigrantes, respectivamente, en el intervalo *(t,t+1)*.

Se puede observar que la población crece en dos cantidades diferentes *(N-D)* e *(I-E)*. Se llama ***crecimiento natural*** a la primera, mientras que a la segunda se la denomina ***inmigración neta*** o ***saldo migratorio***. El primero es fundamental, en el sentido de que es el único que rige la dinámica de una población cerrada (una población en la que no hay intercambio migratorio externo, como podría ser una isla aislada del Pacífico, valga la redundancia).

Se define la tasa de crecimiento (r) como

$$r = \frac{P^{t+1} - P^t}{(P^{t+1} + P^t)/2}$$

[2] En demografía, se suele tomar como unidad de tiempo estándar el año, así pues el intervalo anterior sería un año. No obstante, si se especifica, cualquier intervalo es válido para las definiciones que se darán a lo largo de este trabajo.

Esta tasa indica y resume la dinámica de crecimiento de una población, aunque sin demasiado detalle, evidentemente. Se puede expresar también, al igual que la mayoría de tasas, en tanto por cien o tanto por mil, además de en una escala entre 0 y 1.

En cuanto a la estructura de la población, se puede tomar en consideración una gran cantidad de factores en base a los cuales clasificar y estructurar a la población (edad, sexo, estado civil, nivel económico, actividad profesional, etc.). Sin embargo, hay dos que, por su sencillez y la gran cantidad de información que aportan, suelen ser tenidos en cuenta más que ningún otro: la edad y el sexo.

Además, dividiendo la población en grupos de acuerdo a estos dos criterios, se puede obtener uno de los instrumentos más sencillos a la vez que útiles que hay en demografía: la ***pirámide de población***.

Las pirámides de población ofrecen de forma inmediata una imagen global de la población que se analiza. Pero permiten además, mediante un estudio más pormenorizado, poner de manifiesto cuáles han sido las vicisitudes de la evolución demográfica en un período de ocho o nueve decenios. Además pueden dar pistas sobre las principales tendencias que se observarán en el futuro [11].

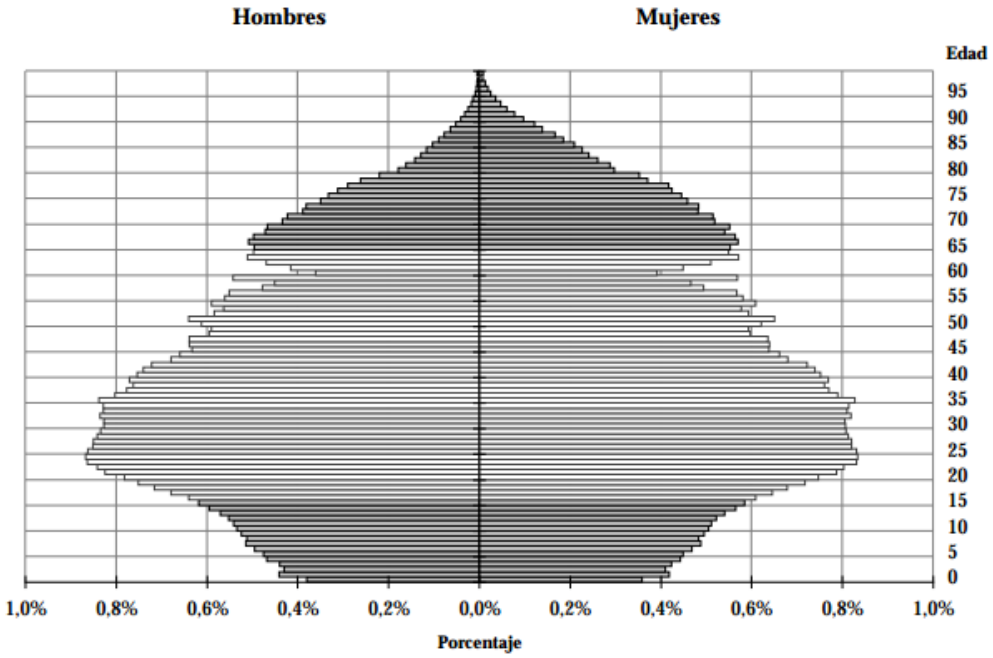

Figura 2.1*. Pirámide de la población española en 2000. Fuente: [16].*

2.2.2 LA MORTALIDAD

La mortalidad es el fenómeno demográfico que se interesa por el acontecimiento "defunción".

Se define la ***tasa bruta de mortalidad (m)*** como el cociente entre el número de defunciones entre la población media durante el periodo (en este caso, se realiza interpolación lineal y se toma el valor intermedio entre la población inicial en *t* y la población final en *t+1*).

$$m = \frac{D^{t,t+1}}{(P^{t+1} + P^{t})/2} * 1000$$

Sin embargo, dado que no distingue entre los distintos sectores por edades de la población, la tasa bruta de mortalidad no dice demasiado acerca de la verdadera distribución de la mortalidad en una población. Por eso se define la ***tasa específica de mortalidad (m_x^t)*** como el número de personas fallecidas a los *x* años durante el período dividido entre la población media de esa edad durante el período

$$m_x^t = \frac{D_x^t}{(P_x^{t+1} + P_x^t)/2} * 1000$$

Uno de los instrumentos más útiles a la hora de conocer el fenómeno de la mortalidad en una población son las **tablas de mortalidad**. Son tablas que resumen las condiciones de mortalidad de una población de manera transversal. El principio de construcción es ir eliminando los efectivos de una generación inicial a causa de la mortalidad. Esta generación, frecuentemente, no es real (pues conseguir datos de una generación entera a lo largo de toda su vida es bastante complicado), sino que es ficticia: se relacionan las defunciones de un año de calendario —o la media de diversos años, pues muchas veces las tablas muestran intervalos quinquenales— con la población registrada o estimada en el momento central de ese período. Esta población, que se compone de múltiples generaciones, se trata como si toda ella perteneciera a una única generación. Algunos de los datos que suelen incluir y que pueden resultar bastante útiles a la hora de hacer simulaciones con agentes son los siguientes:

- *Probabilidad de muerte* (${}_nq_x$) es la probabilidad que tiene una persona del grupo de edad *[x,x+n)* de morir durante ese intervalo de tiempo. Se calcula ${}_nq_x = (2n * {}_nm_x)/(2 + n * {}_nm_x)$
- *Probabilidad de supervivencia* (${}_np_x = 1 - {}_nq_x$) es el complementario del anterior, la probabilidad de que una persona de edad *x* alcance con vida la edad *x+n*.
- *Supervivientes a edades exactas* (l_x) representa el número de supervivientes de una población inicial hipotética (denominada raíz de la tabla y que suele ser por convenio una potencia de diez) a la que se le hubieran ido aplicando las probabilidades de muerte para ir descontando efectivos. Viene dada por $l_{x+n} = l_x$ -($l_x * {}_nq_x$)

- *Defunciones entre edades exactas* ($_nd_x$) el número de defunciones (otra vez, basándonos en la población hipotética inicial) acaecidas en el intervalo *[x,x+n)*, es decir, $_nd_x = l_x - l_{x+n}$
- *Población estacionaria* ($_nL_x$) representa la estructura por edades que adoptaría una población cuya mortalidad fuera la de la tabla y se mantuviera constante en el tiempo, con entradas por nacimiento también constantes y equivalentes a la raíz de la tabla. Se calcula $_nL_x = n(0.5 * (l_x + l_{x+n}))$
- *Tiempo vivido* (T_x) representa la serie acumulada de años-persona desde una edad determinada hasta el final de la tabla. T_x = $_nL_x + _nL_{x+n} + _nL_{x+2n} + \cdots$ hasta el final de la tabla.
- *Expansión de vida* (e_x) es la cantidad de años que, como media, puede esperar vivir una persona de edad *x*. $e_x = T_x/l_x$

Edad	l_x	$_nm_x$	$_nq_x$	$_nL_x$	T_x	e_x^0
0	1.000,00	4,05	4,03	996,77	82.503	82,50
1	995,967	0,37	1,46	3.978,78	81.506	81,84
5	994,513	0,09	0,46	4.971,30	77.527	77,95
10	994,054	0,08	0,38	4.969,35	72.556	72,99
15	993,671	0,20	1,01	4.965,95	67.587	68,02
20	992,668	0,29	1,44	4.959,90	62.621	63,08
25	991,235	0,37	1,84	4.951,80	57.661	58,17
30	989,410	0,44	2,19	4.941,84	52.709	53,27
35	987,240	0,61	3,07	4.928,94	47.767	48,38
40	984,213	0,96	4,79	4.909,76	42.838	43,53
45	979,502	1,50	7,46	4.879,97	37.928	38,72
50	972,193	2,38	11,85	4.833,31	33.048	33,99
55	960,670	3,43	17,00	4.764,15	28.215	26,37
60	944,337	5,34	26,35	4.661,95	23.451	24,83
65	919,449	8,42	41,26	4.506,20	18.789	20,44
70	881,515	15,26	73,62	4.251,82	14.283	16,20
75	816,618	30,55	141,93	3.793,33	10.031	12,28
80	700,715	61,15	264,34	3.029,17	6.238	8,90
85	515,487	114,14	437,51	1.975,91	3.208	6,22
90	289,956	198,72	635,45	927,18	1.233	4,25
95	105,703	322,71	808,66	264,87	305	2,89
100	20,225	499,20	1.000,00	40,51	41	2,00

Figura 2.2. *Tabla de mortalidad tipo. Obsérvese que tanto las tasas específicas de mortalidad como las probabilidades de muerte están expresadas en tantos por mil. (Tomada de [11].)*

En cuanto a los factores socioeconómicos relacionados con la mortalidad, se pueden mencionar el sexo, la edad, el nivel de renta, el nivel cultural o la actividad profesional. Todo esto, además, se puede ver afectado por otros factores ambientales o coyunturales como pueden ser la climatología, la situación económica global del país, guerras, etcétera.

2.2.3. NATALIDAD, FECUNDIDAD Y NUPCIALIDAD

La natalidad y la fecundidad[3] son confundidas con bastante frecuencia, pues se tiende a no distinguir la una de la otra. Sin embargo, mientras que la natalidad se refiere a la frecuencia de los nacimientos dentro de una población tomada en su conjunto, la fecundidad mide la frecuencia de nacimientos dentro del subconjunto de la población en edad de procrear.

Algunas medidas de fecundidad que pueden resultar útiles a la hora de modelar los fenómenos demográficos son las siguientes:

Una primera aproximación, aunque un tanto grosera, es la ***tasa bruta de natalidad (TBN)***. Representa la relación entre el número de nacimientos en el seno de la población en un año y la población media durante ese año (que, como antes, se puede tomar por interpolación lineal de las poblaciones inicial y final de ese año, que son los datos de los que se suele disponer, aunque a veces se toma la población a mitad de año, si se tiene el dato), y se suele expresar en tantos por mil

$$TBN^t = \frac{N^{t,t+1}}{(P_x^{t+1} + P_x^t)/2} * 1000$$

La ***tasa general de fecundidad (TGF)*** afina un poco más, y da la relación entre el número de nacimientos en un año, y la población que directamente tiene que ver con el fenómeno —o, al menos, la que establece un límite al número de nacimientos—, las mujeres en edad fértil , que suele considerarse entre 15 y 49 años (Pf_{15-49})

$$TGF^t = \frac{N^{t,t+1}}{(Pf_{15-49}^{t+1} + Pf_{15-49}^t)/2} * 1000$$

Y, al igual que sucedía con las tasas de mortalidad, se puede obtener una información más detallada si se restringe a franjas de edad más concretas, con la ***tasa específica de fecundidad (TEF)*** que son equivalentes a la anterior, pero para mujeres en edad *x*

$$TEF_x^t = \frac{N_x^{t,t+1}}{(Pf_x^{t+1} + Pf_x^t)/2} * 1000$$

La suma de las tasas específicas para cada edad resulta en el ***índice sintético de fecundidad***, también llamado ***número medio de hijos por mujer***. Este indicador refleja el número medio de hijos que tendría una mujer a lo largo de su vida, suponiendo que no hay mortalidad para las mujeres durante sus años fértiles y que durante ese período se registrarán las mismas TEF del año en cuestión.

Otro indicador bastante interesante es la ***edad media al ser madre (ā)*** y que se obtiene a partir de las tasas específicas de fecundidad de la siguiente manera

$$(\bar{a})^t = \frac{\sum(x + {}^n/_2) * TEF_x^t}{\sum TEF_x^t}$$

[3]También resulta frecuente la confusión con la *fertilidad*, sobre todo al trabajar con bibliografía en inglés, pues en este idioma los términos están "cruzados" con respecto al castellano. Lo que en castellano es fecundidad, se refiere a la *fertility* en inglés, mientras que la fertilidad se dice en este idioma *fecundity*, la capacidad biológica de procrear.

Indicadores como el **tiempo medio entre nacimientos**, que da el tiempo transcurrido, en media, entre dos nacimientos de una misma madre, o las **tasas de fecundidad por orden de nacimientos**, que indican el número de nacimientos según el nacido sea el primer, segundo, tercero o sucesivos hijos de la madre.

Finalmente, a la hora de modelar el fenómeno de los nacimientos, también puede ser de gran utilidad la **proporción de feminidad en los nacimientos**, que relaciona el número de nacimientos de mujeres entre el total de nacimientos. De igual manera se puede definir una **proporción de masculinidad en los nacimientos**.

El estudio de la nupcialidad, por otro lado, se encarga de analizar los fenómenos de constitución y disolución de matrimonios y, en los últimos tiempos, de otro tipo de uniones consensuales, cada vez más frecuentes en la sociedad.

En cuanto a los indicadores de la nupcialidad, éstos tendrán que hacerse diferenciando sexos, aunque las definiciones que se presentan no hacen referencia al sexo para no resultar reiterativo. En la misma línea que con la fecundidad, se definen:

La **tasa bruta de nupcialidadTBNup**, que relaciona el número de matrimonios producidos durante un año con la población media (de hombres o de mujeres) durante ese período

$$TBNup^t = \frac{M^t}{(P^{t+1} + P^t)/2} * 1000$$

Igual que antes, si se restringe a la población susceptible de casarse —mayores de 15 años— se obtiene la **tasa general de nupcialidad (TGNup)**

$$TGNup^t = \frac{M^t}{(P_{15+}^{t+1} + P_{15+}^t)/2} * 1000$$

Y siguiendo con la analogía con la fecundidad, se pueden especificar aun más, restringiéndose a edades concretas (o franjas de edad concretas) y obteniendo la **tasa específica de nupcialidad (TENup)**

$$TENup_x^t = \frac{M_x^t}{(P_x^{t+1} + P_x^t)/2} * 1000$$

Es interesante también estudiar el fenómeno de las primeras nupcias. Para ello bastaría con restringirse a la población que nunca se ha casado y calcular las mismas tasas.

Finalmente, y al igual que antes, se puede calcular un **índice sintético de nupcialidad** y la **edad media al matrimonio** a partir de las tasas anteriores, exactamente igual que se hacía con la fecundidad. Además, estas tasas pueden igualmente ser aplicadas al fenómeno del divorcio exactamente de la misma manera.

En cuanto a los factores que intervienen en la fecundidad, están en primer lugar los biológicos, entre los que destaca fundamentalmente la fertilidad, que depende de la salud, edad, etc. Por otro lado están los factores como nivel cultural, nivel de renta y otros elementos del medio social como pueden ser creencias religiosas y, por supuesto, la nupcialidad. A su vez, la nupcialidad está afectada por la edad y los niveles socioeconómicos y culturales, pues las parejas tienden a formarse endogámicamente dentro de grupos de

similar nivel social [17]. Por supuesto, también está condicionada y muy relacionada con la movilidad espacial, pues al casarse una pareja suele ir a vivir juntos a un nuevo hogar, así que las condiciones de disponibilidad de hogares también pueden afectar.

2.2.4. MOVIMIENTOS MIGRATORIOS

Los movimientos de la población, a diferencia de los fenómenos demográficos como mortalidad o fecundidad, son difíciles de definir conceptualmente ya que hay que tratar diferentes tipos de desplazamientos que realizan los humanos. Diferentes por duración y por alcance geográfico.

Entendiendo por residencia el *centro de gravedad* desde el que se originan los movimientos cíclicos y periódicos que una persona realiza en su vida cotidiana, y que determinan su *espacio de vida*. Se considerará migración al movimiento que modifica este centro de gravedad y que, por lo general, modificará el espacio de vida también.

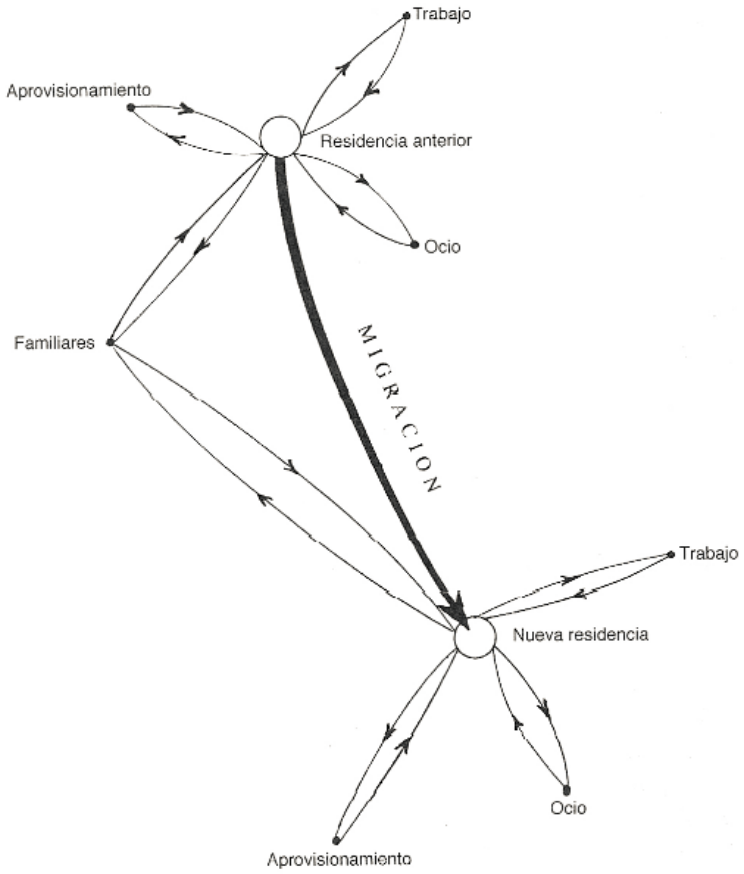

Figura 2.3. *Definición de espacio de vida. (Tomada de [11].)*

No se dará aquí indicadores sobre las migraciones, dado que este es un proceso mucho más flexible e indefinido que el resto de procesos demográficos y por tanto presenta una complejidad y una variabilidad mayor en su estudio. Aunque, en esencia, son similares a los estudiados, así que disponiendo de los datos adecuados —datos que pueden ser bastante específicos, eso sí— es sencillo hallar las tasas que interesen para el problema en cuestión y se pueden aplicar a la hora de hacer una simulación. Algún indicador sobre cantidades brutas o netas de migraciones también podría resultar útil a la hora de incluirlo en el modelo.

En cuanto a los factores relacionados con la migración, los más importantes, por frecuentes, son la edad ya que al alcanzar cierta edad las personas suelen abandonar el hogar paterno y la nupcialidad, como se ha comentado anteriormente muy ligada a los movimientos de la población. Por otra parte, es evidente que la coyuntura económica es un factor desencadenante de migraciones en busca de otras condiciones de vida, en la mayoría de los casos a otros países o bien del medio rural a las ciudades.

3. ESTADO DEL ARTE

En esta sección se presenta un breve repaso por los principales trabajos realizados hasta el momento sobre los campos relacionados con el presente trabajo.

En un primer apartado se repasan las principales líneas de investigación en simulación social basada en agentes, con un pequeño recorrido histórico de la simulación social, seguido de una relación de los trabajos más actuales sobre el tema.

A continuación se exponen las principales herramientas de software disponibles para hacer modelos basados en agentes.

En un tercer apartado se trata el tema de los modelos demográficos, tanto basados en agentes como en otras técnicas, y se hace también un breve repaso histórico de los mismos.

Finalmente se exponen las conclusiones y se justifican, en base al material presentado anteriormente, las elecciones realizadas en el proyecto.

3.1. SIMULACIÓN BASADA EN AGENTES

3.1.1. SIMULACIÓN BASADA EN AGENTES: USOS Y MODELOS

La simulación o modelado basada en agentes (ABM[4]) es un campo relativamente nuevo pero que en los últimos años está cobrando una gran importancia, sobre todo desde la década de 1990 [18].

Básicamente, un modelo basado en agentes es un programa formado por unos entes —los agentes— y un entorno en el que estos están situados. Los agentes tienen las siguientes propiedades [19]:

- Los agentes son **autónomos**, es decir, actúan sin que otros entes tengan control sobre ellos.
- Tienen **habilidad social**, es decir, capacidad de interactuar entre ellos.
- Tienen capacidad de **percibir** su entorno y actuar respondiendo a eventos que ocurran a su alrededor.

[4] *ABM*, *Agent Based Modelling*, modelado basado en agentes. Se utilizará indistintamente para referirse a simulaciones basadas en agentes o modelos basados en agentes.

- Finalmente, tienen capacidad de **influir** activamente en su entorno para cumplir sus objetivos.

Es una técnica que permite abordar y estudiar problemas de muy distinta índole. Entre los posibles usos de la simulación (ya sea con agentes o de otra índole) se encuentran la predicción, ejecución de tareas, entrenamiento, entretenimiento, demostración de hipótesis y descubrimiento [20]. No obstante, en este mismo artículo, se afirma que la función primordial de la simulación basada en agentes ha de ser la de enriquecer nuestro conocimiento de los procesos fundamentales que se modelan. Es decir, se trata más bien de una función explicativa.

De todas maneras, restringiéndose a la simulación basada en agentes en el ámbito académico, un estudio [21] de los principales artículos sobre el tema revela que los principales usos que presentan los modelos basados en agentes son la predicción, verificación de hipótesis o teorías, entrenamiento y análisis, entendiendo este último como adquirir mayor conocimiento y comprensión de cierto dominio en el que no hay una teoría o modelo específico que pueda ser verificado.

En cuanto a las aplicaciones concretas de los modelos basados en agentes, ya se ha comentado que son de lo más variadas. Esto viene dado, en gran medida, por la capacidad de los sistemas multi-agente de adaptarse perfectamente a problemas del mundo real, pues el mundo está formado por sistemas más o menos complejos, formados a su vez por agentes que interactúan entre sí. Y el paradigma de agentes permite una correspondencia prácticamente exacta entre las entidades del sistema a estudiar —y las relaciones entre ellos— y los componentes del modelo informático (los agentes) [22], [23].

A continuación, unos cuantos ejemplos de distintos modelos, en distintos dominios:

- **Comportamiento animal**: Hay multitud de ejemplos, desde el clásico [24] donde se consigue modelar el comportamiento de bandadas de pájaros, o bancos de peces en base a reglas simples, hasta modelos más complejos, como [25], donde se modelan los procesos migratorios de los alces en Yellowstone.
- **Modelado de procesos fisiológicos complejos** [26].
- **Organización**: Formación de grupos de trabajo en proyectos de ingeniería [27], [28] o planificación de actividades [29].
- **Tráfico y transportes**: En [30] se modela el comportamiento de conductores de automóviles.
- **Simulación de redes sociales** (tipo Facebook), como Krowdix [31], u otros ejemplos en que se estudia cómo se difunden rumores e información a través de las redes [32].
- **Economía**: estudio del comportamiento de mercados en [33] y en [34].

- **Gestión de recursos naturales**, como en [35], donde se estudia la demanda de agua para consumo doméstico en Valladolid, y una gran cantidad de estudios sobre usos del suelo [36].
- **Sociología**: estudios sobre la evolución de valores morales en sociedades [37], sobre la influencia de la presión social [38].

3.1.2. SIMULACIÓN SOCIAL BASADA EN AGENTES

Si hay un campo en donde la analogía entre el paradigma de modelado multi-agente y el sistema a modelar se cumple a la perfección, es el caso de los sistemas sociales. Un sistema social, formado por individuos heterogéneos que interactúan entre sí y con su entorno, movidos cada uno por sus propios objetivos, se presta a ser modelado con agentes de manera natural.

En el ámbito de las ciencias sociales la función explicativa de los fenómenos sociológicos cobra un papel fundamental [19] y predominante frente a la predictiva. Esto es debido, en primer lugar, a la propia complejidad inherente los sistemas sociales, pues estos son sistemas no lineales con un comportamiento altamente no predictivo (sistemas complejos). Además y tratándose de predicciones de carácter social o económico, está el riesgo de que la propia predicción afecte al resultado de la misma, y no se cumpla.

Así, frente a otras técnicas con una aproximación también al nivel de individuos como la microsimulación, que tiene un enfoque más dado a analizar las salidas macroscópicas al introducir una serie de datos microscópicos, dejando un tanto de lado los aspectos teóricos [18], la simulación social basada en agentes (ABSS[5]) tiene como misión fundamental comprender y explicar los fenómenos sociales.

Para ello, y por lo apuntado acerca de la complejidad de los procesos sociales, es conveniente adherirse al principio KISS (Keep It Simple, Stupid) [20], es decir, tratar de simplificar los modelos para que no resulten más complicados de lo necesario. Aunque, frente a un exceso de simplificación, han surgido otras opiniones que piden un compromiso entre la simplicidad y la expresividad del modelo, aunque con diferentes enfoques: [39] propone partir de un modelo lo suficientemente descriptivo, que será simplificado sólo si es necesario. Por otro lado, [40] parte de un caso de estudio más simple y propone ir haciendo más complejo y descriptivo poco a poco, por etapas.

El primer ejemplo destacado de simulación social, y uno de los más citados, se remonta a 1969, cuando Schelling propone su modelo de segregación [41], [42]. En él, los agentes están dispuestos en una cuadrícula, y los hay de dos clases. En cada ciclo, consideran si moverse a otra ubicación o permanecer en la que están, de acuerdo a si el porcentaje de

[5] *ABSS*, *Agent Based Social Simulation*, o Simulación Social Basada en Agentes.

vecinos de distinta clase que el suyo es mayor que un nivel de tolerancia dado. Los resultados muestran que para niveles de tolerancia bastante altos, es decir, agentes no demasiado "racistas", éstos terminan agrupándose con los de su misma clase, se produce un comportamiento global "racista".

Otro trabajo muy importante es el modelo Sugarscape [43]. En él, lejos de centrarse en un problema específico a estudiar, los autores se dedican a construir *sociedades artificiales*, construyen agentes a los que dotan de necesidades y comportamientos individuales, y observan los comportamientos globales emergentes. En principio, los agentes tan sólo necesitan comer, y para ello han de interactuar con el medio para obtener el alimento. Pero, poco a poco, los autores van introduciendo recolección de alimento, reproducción sexual, comercio, guerra, enfermedad, herencia, etc. hasta construir una sociedad artificial con un comportamiento global complejo que surge a partir de interacciones más o menos simples entre los agentes.

Hasta hace poco, los modelos de simulación social basados en agentes que tratan de estudiar problemas reales no han sido demasiado ambiciosos, sino que más bien se han centrado en problemas más concretos, en lugar de tratar de abarcar una gran cantidad de procesos sociales. No obstante, cada vez se construyen modelos más ambiciosos, normalmente interdisciplinares, y por lo tanto más complejos y que requieren de una metodología [44].

3.2. HERRAMIENTAS

En la actualidad existe una gran cantidad de herramientas para el desarrollo de ABM. Las hay de propósito general (lo que incluye desde realizar sencillas simulaciones de autómatas celulares hasta la posibilidad de programar videojuegos con ellas), otras que son para propósitos más concretos (por ejemplo, para estudio de mercados [45]).

Las hay basadas en lenguajes de programación de propósito general como Java, C++ o Python, y que vienen distribuidas como *frameworks* y librerías (como podrían ser MASON [46], RePast [47] o Swarm [48]) o algunas que emplean lenguajes específicamente creados para el modelado de ABM, como [49], o que vienen preparadas como un programa en el que se puede trabajar directamente, como puede ser NetLogo [50], o que incluso permiten un modelado visual (SeSAm [51] o AgentSheets [52]). Las primeras tienen la ventaja de que son más potentes, además de que los lenguajes orientados a objetos se adaptan bastante bien a la hora de modelar SMA (sistemas multi-agente) [43]. Las segundas no exigen tener que conocer, muchas veces bastante en profundidad, un lenguaje como Java, aunque también a costa de hacer algunos sacrificios en su funcionalidad (por ejemplo en AgentSheets resulta difícil hacer interactuar directamente a dos agentes [53]).

Esto último es la gran problemática de las herramientas potentes para hacer ABSS. Los sociólogos, que en última instancia son los que usarán y sacarán provecho de las simulaciones, no suelen tener conocimientos profundos de programación [22], así pues herramientas, sobre todo visuales (como por ejemplo INGENIAS [22]), que ayuden a la hora de realizar los modelos siempre serán de ayuda.

La elección de una herramienta u otra dependerá por tanto del dominio y necesidades concretas del problema a tratar [54]. En esta referencia se puede ver una extensa revisión de la mayoría de herramientas para ABM existentes. No obstante, siempre hay herramientas que son más populares que otras, bien porque lo merecen, o por modas, y que se pasan a detallar a continuación [55].

- **Swarm**: Es una de las primeras que se crearon. Es un *framework* y un conjunto de librerías. Originalmente desarrolladas en Objective-C, posteriormente surgió una versión Java. Es de propósito general y está pensada para abarcar un amplio abanico de modelos basados en agentes.

- **RePast**: También es un *framework* al igual que Swarm. Originalmente concebida como una reimplementación de Swarm en Java, su diseño debe mucho a Swarm. Actualmente hay versiones en otros lenguajes (C#, Python y más). Su ámbito es el de las ciencias sociales, e incluye por tanto herramientas propias de estos problemas. Es, junto con NetLogo, la más popular y usada entre la comunidad de ABSS. Incluye facilidades para tomar fotos y vídeos de las simulaciones, gráficas y permite manipular las simulaciones en tiempo de ejecución vía la interfaz gráfica. También incluye librerías para GIS[6], redes sociales, algoritmos genéticos, etc.

- **MASON**: *Framework* de propósito general implementado en Java, es similar a RePast en cuanto a funcionalidades, aunque su diseño pone un gran énfasis en la rapidez de ejecución y en la separación entre modelo y visualización [57]. Permite realizar simulaciones con un gran número de agentes, y es una plataforma muy modular y flexible.

- **NetLogo**: A diferencia de los anteriores, NetLogo es un entorno de modelado, desarrollado en Java, que no necesita de librerías externas, ni de compilación, ni de grandes conocimientos de programación. Usa el lenguaje Logo [58], que es de más alto nivel que los anteriores también. Su origen

[6] GIS (*Geographical Information System*) es un sistema de información diseñado para trabajar con datos referenciados por coordenadas espaciales o geográficas [56]. En otras palabras, es una manera de integrar descripciones e información de entornos geográficos en programas de ordenador.

como herramienta para la educación hace que sea sencilla de usar, aunque no por ello es incapaz de modelar sistemas bastante complejos [55].

3.3. SIMULACIÓN Y MODELOS DEMOGRÁFICOS

3.3.1. ALGUNOS MODELOS CLÁSICOS EN DEMOGRAFÍA

Tradicionalmente, la demografía se ha preocupado en gran medida por los datos y las observaciones macroscópicas. Además, al contrario que otras ciencias sociales, tiene un mayor potencial predictivo, pues mucha de la información necesaria para hacer estas predicciones se encuentra en la propia estructura de la población y se conocen y pueden delimitar sus factores principales (muertes, nacimientos, migraciones) [59]. Así, no es extraño que desde un principio haya habido un gran interés de los demógrafos en tratar de modelar y hacer proyecciones sobre la población. Los métodos que se describen a continuación responden a esta motivación.

LOS MODELOS EXPONENCIAL Y LOGÍSTICO

Los modelos más sencillos de crecimiento de la población son el exponencial y el logístico.

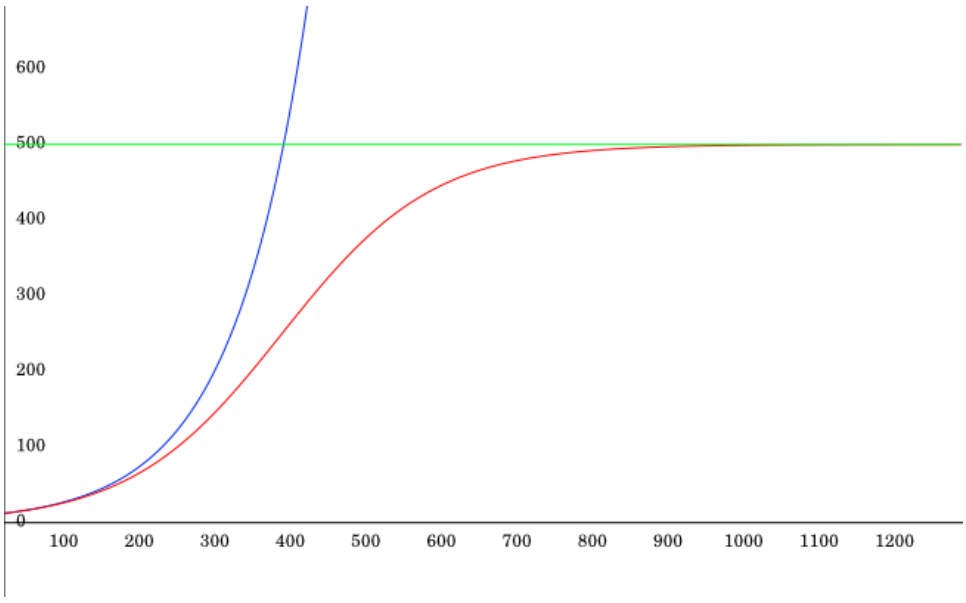

Figura 3.1. *Crecimientos logístico y exponencial para una población inicial de 10 individuos, capacidad de carga de 500 y tasa de crecimiento del 1%, es decir de 0.01.*

El primero se basa en el concepto de crecimiento geométrico. Si se parte de una población inicial P^0 con una tasa de crecimiento constante *r* —recuérdese que la tasa de crecimiento en un período determinado era el cociente entre el incremento en la población en ese período y la población media durante el período—. Entonces la población un año más tarde será

$$P^1 = P^0 + P^0 * r = P^0(1 + r)$$

Y a los dos años

$$P^2 = P^1 + P^1 * r = P^1(1 + r) = P^0(1 + r)^2$$

Y a los *t* años será

$$P^t = P^0(1 + r)^t$$

Sin embargo, con este enfoque, se está "actualizando" la población tan sólo una vez al año, cuando este proceso es más bien continuo, por lo que se podría actualizarla cada medio año, cada mes, cada día… e incluso de manera continua [14]. Es bien conocido que esto conduce a la función exponencial, por lo que la población en cualquier instante (medido en años, que se está tomando como unidad) vendrá dada por

$$P^t = P^0 e^{rt}$$

Es evidente que es un modelo muy simplista, que presupone una tasa de crecimiento constante e inmutable e ignora cualquier otro factor. Sin embargo en el pasado este método era empleado con frecuencia [11]. Pero tal vez lo más perturbador de este modelo es que asegura un crecimiento hasta el infinito.

El modelo logístico trata de solucionar este problema poniendo un límite al crecimiento, mediante una población máxima, y haciendo que la tasa de crecimiento vaya disminuyendo conforme la población se aproxima a ese máximo pero, aun así, ofreciendo un comportamiento muy similar al exponencial cuando la población está lejos del límite. Viene dado por esta ecuación

$$P^t = \frac{KP^0 e^{rt}}{K + P^0(e^{rt} - 1)}$$

Donde P^t es la población en el año t, K la población máxima admitida o capacidad de carga del sistema, *r* es la tasa de crecimiento anual y P^0 la población inicial.

No obstante, y pese a este límite, adolece de prácticamente los mismos defectos que el modelo exponencial.

EL MÉTODO DE LOS COMPONENTES

Este es un método general empleado para hacer **proyecciones** de la población y que, a diferencia de los anteriores, tiene en cuenta los diferentes factores demográficos que intervienen en la ecuación compensadora: fecundidad, mortalidad y migraciones, además de contemplar la población por edades y sexo. Además de tomar estos factores como *entrada* también produce una *salida* mucho más expresiva, que nos dice cosas acerca de la estructura por edades y el número de nacimientos, defunciones y migraciones.

Se podría decir que este método lo que hace es ir reconstruyendo una población paso a paso —generación a generación, aunque también se puede enfocar haciendo el seguimiento de una única generación a lo largo de su vida— en base a unas tasas de fecundidad, mortalidad y migraciones. La elección de estas tasas, no obstante, es lo que es realmente importante del modelo, aunque eso sí, como mínimo se les ha de exigir que sean tasas específicas por edad, o grupos de edad relativamente pequeños, para producir modelos más realistas. Pueden ser desde tasas constantes, lo cual acabará dando lugar a una población estable, si esta es cerrada[7], hasta proyecciones de estas tasas mediante submodelos específicos que las estiman [11].

Casi todas las proyecciones de población modernas se basan en este método. Examinando cómo se pasa de una población por sexo y edad del 1 de enero del año de partida a la del 1 de enero del año siguiente, bastará con repetir *n* veces la operación para llegar a la población por sexo y edad del 1 de enero del año *n*. En cada etapa, la proyección se desarrolla en dos tiempos. El primero consiste en trabajar sobre la población ya nacida y el segundo sobre la población que va a nacer. Se calcula en primer lugar, para cada sexo, los supervivientes a la edad *x+1* el 1 de enero siguiente, multiplicando el efectivo de edad *x* del 1 de enero de partida por la probabilidad de supervivencia entre *x y x+1*. Se suma la inmigración neta de la misma edad —caso de haberla—. Se tendrá entonces el resultado buscado para la población de un año y más. Queda por calcular el efectivo de los menores de un año, que se obtiene aplicando las tasas específicas de fecundidad por edad a los efectivos de mujeres de la misma edad, lo que produce los nacimientos del año, que se reparten por sexo gracias a las tasas de masculinidad al nacimiento, aplicando a dichos nacimientos las tasas de supervivencia (y eventualmente de inmigración neta) apropiadas. Lo que se hace para un año se vuelve a hacer para el siguiente y así hasta llegar al término de la proyección [13].

3.3.2. MICROSIMULACIÓN

La microsimulación o simulación microanalítica (MSM) es una técnica que, al igual que los modelos basados en agentes, y a diferencia de los modelos de dinámica de poblaciones, o el método de los componentes visto anteriormente, tiene una aproximación de abajo a arriba. La International Microsimulation Association [60] la define como una técnica de modelado que opera a nivel de unidades individuales. Éstas pueden ser personas, hogares u otro tipo de entes. En el modelo están representados mediante un conjunto de atributos (edad, sexo, etc.) que pueden ir cambiando en cada etapa de la simulación de acuerdo a una

[7] Una *población estable* es una población cerrada —en la que no hay lugar para migraciones, que está "aislada"— en la que la estructura por edades es constante. Si una población tiene tasas constantes de fecundidad y mortalidad, tenderá a estabilizarse con el tiempo, sin importar la estructura por edades inicial. Además, esta estructura por edades viene unívocamente determinada por las tasas de fecundidad y mortalidad concretas [13].

serie de reglas (probabilidades de transición entre un estado y otro) que pueden ser bien deterministas (con probabilidad 1), como por ejemplo aumentar la edad, o bien estocásticas (con probabilidad menor o igual a 1), como podría ser morir, tener hijos o casarse.

La microsimulación depende en gran medida de los datos sobre la población y normalmente esta es una gran cantidad de datos [59]. Además estos datos pueden llegar a ser, según el problema a tratar, bastante específicos o de "alta calidad" [61]. Sin embargo, esta que puede parecer su debilidad, es también lo que la hace una técnica muy descriptiva: como permite hacer estimaciones usando una gran cantidad de unidades [62], es sencillo, si se dispone de estos datos, lograr individuos heterogéneos que representen a la población de manera precisa.

Se pueden diferenciar —entre otras posibles clasificaciones en base a otros criterios— dos tipos de microsimulación. Por un lado se tienen los modelos estáticos, en los que el archivo de datos no cambia, y que suelen emplearse para simulaciones de procesos a corto plazo (el ejemplo típico es el efecto que produce en el consumo una subida de impuestos). Por otra parte, están los modelos dinámicos, en los que se actualiza el estado de cada unidad individual en cada etapa. Esto permite introducir nacimientos y muertes de forma explícita [19], y permite introducir influencia de unas personas a otras (relaciones) y del conjunto total de la población a las personas [62], aunque de forma un tanto limitada y sin la potencia que dan los ABM en este sentido. Estos modelos dinámicos permiten por tanto hacer simulaciones largo plazo, y son por tanto las que se emplean para modelar los procesos demográficos.

Aunque también se le ha dado otros usos, como modelado del tráfico [63], se ha empleado principalmente para hacer proyecciones de población, evaluación de políticas estatales (sobre impuestos, por ejemplo), diseño de reformas (también a nivel estatal, pensiones, programas de salud, etc.) [64]. Por tanto, en casi todos estos casos, sobre todo en los modelos dinámicos, el papel del modelo demográfico subyacente es vital.

Algunos modelos demográficos interesantes basados en la microsimulación pueden ser DYNASIM3 [65] empleado para analizar envejecimiento de la población , DYNACAN [66] para estudiar el impacto de políticas en el sistema de pensiones canadiense, MOSART [67]que realiza proyecciones de la población noruega, SESIM [68], también para estudiar temas de pensiones y envejecimiento de la población en Suecia o DYNAMOD [69] en Australia, que modela la población de ese país y realiza una proyección de su evolución en 50 años.

En los anteriores modelos, los principales eventos demográficos que se estudian se pueden agrupar en 3 categorías:

- Crecimiento de la población: migraciones, nacimientos, muertes.
- Hogares y familias: emancipación de los hijos, emparejamiento, matrimonio y divorcios.
- Otras características: nivel educativo, nivel de salud, status social.

Serán por tanto, éstos, factores a tener en consideración a lo largo del presente trabajo.

Finalmente el lector podría preguntarse qué diferencias existen realmente entre las dos maneras de modelar "desde abajo" que se han visto: la microsimulación y los ABM. En primer lugar la microsimulación no deja de ser una serie de datos en forma de matriz donde cada fila representa a un individuo y cada columna un atributo del mismo. A estos individuos, en función de los valores de sus atributos, se les van aplicando una serie de probabilidades que pueden hacer que sus atributos ("estados") cambien. Esto contrasta con el concepto, mucho más rico, de un agente, a la vez que mucho más flexible.

Los agentes se mueven por reglas de comportamiento, y no simplemente por probabilidades de transición entre estados. Además, los modelos de microsimulación generalmente evalúan, en base a unos datos de entrada en el nivel micro, las salidas producidas en el nivel macro, sin tener posibilidad de que haya *feedback* entre los diferentes niveles de entes (individuos, grupos, sociedad). Esto, en los ABM no sucede, y es quizás su mayor ventaja frente a otros métodos [18], [62].

Además de lo anterior, los modelos basados en agentes permiten modelar las interacciones entre individuos de forma mucho más rica.

Finalmente, esto permite que los modelos basados en agentes se puedan centrar en la teoría, mientras que los modelos de microsimulación se quedan muchas veces en el simple análisis de los datos [18].

No obstante, no son métodos antagónicos. Son frecuentes los modelos mixtos, que emplean conceptos de ambos métodos [61], como se verá en la siguiente sección.

3.3.3. MODELOS DEMOGRÁFICOS BASADOS EN AGENTES

La demografía, como ciencia social que es, también es susceptible de ser estudiada mediante modelos de agentes, aunque no ha sido hasta recientemente que los demógrafos han empezado a interesarse por esta técnica. Esto es debido, en parte, a que la demografía se ha preocupado más de los datos empíricos y de las técnicas para su análisis que de desarrollar una "teoría" sobre la demografía [1], aunque esta situación haya cambiado en los últimos tiempos, tal y como se comentaba en la Introducción.

Los modelos demográficos basados en agentes resultan por tanto de gran ayuda a la hora de estudiar los procesos demográficos desde el punto de vista teórico, pues ayudan a comprender sus mecanismos desde abajo, desde el nivel de los individuos [70]. Y, si no explicar estos procesos, sí al menos proponer explicaciones válidas para ellos. Lo que es seguro es que los modelos demográficos basados en agentes aportan un nuevo punto de vista en el estudio de la demografía.

Estos modelos, no obstante, suelen centrarse en aspectos concretos de los procesos demográficos, o en procesos concretos de la demografía. No hay, de momento, un modelo puro ABM que integre todos los procesos demográficos principales [62]. Esto, no obstante,

ha de ser matizado. No es que no haya ningún modelo que incluya muertes, nacimientos, emparejamientos y migraciones, que pueden ser considerados los procesos demográficos principales. En algunos de los modelos que se presentarán a continuación se incluyen todos estos aspectos. Sin embargo, algunos de ellos son simples probabilidades obtenidas de datos agregados, sin demasiada "esencia de agentes". Es decir, sin que se modelen desde un punto de vista de ABM, mediante interacciones y comportamientos de los agentes, sino con un enfoque dirigido desde arriba a abajo. Esto, por supuesto, encaja con lo que se ha comentado ya sobre los modelos de simulación social basados en agentes; suelen centrarse en un aspecto teórico a investigar, y se dota a los agentes de reglas sencillas — y plausibles— de comportamiento con las que tratar de obtener el comportamiento global deseado.

Por otra parte, se ha visto que una gran cantidad de procesos sociales son modelados usando agentes. Algunos son sencillos o a corto plazo, como pueden ser los modelos de tráfico, en los que obviamente no hay necesidad ni lugar para un modelo demográfico. Pero otros, que simulan procesos más largos en el tiempo, evidentemente necesitarán incorporar un modelo demográfico que dote de realismo a la simulación. La importancia del modelo demográfico subyacente puede ser, en estos casos, muy relevante a la hora de obtener resultados que se ajusten a los datos observados en la realidad [40], [71].

Por tanto, los modelos demográficos cumplen un doble papel dentro de la simulación basada en agentes; como objetos de estudio en sí mismos —de un proceso demográfico concreto— o como parte fundamental de otro modelo más amplio también. A continuación se verá un repaso de algunos modelos de cada tipo.

MODELOS SOBRE PROCESOS DEMOGRÁFICOS

A continuación se comentan ejemplos de modelos sobre algunos de los procesos demográficos más estudiados.

Demografía espacial

En estos modelos, se trata de estudiar los movimientos —entendidos como migraciones, o cambios de residencia— de los agentes, y las razones a los que atienden. Como modelos demográficos, al menos los casos estudiados, no tienen demasiado interés, pues se limitan a períodos de tiempo no demasiado extensos, y se centran más en modelar las reglas y comportamientos que hacen que los agentes decidan moverse.

En [72] se estudia la dinámica residencial del barrio de Yaffo, en Tel-Aviv, poblado por árabes y judíos, en el período 1955-1995. Existen una serie de restricciones a la hora de buscar o cambiar de vivienda. Por ejemplo, los árabes tienen unas preferencias arquitectónicas y los judíos otras, no se mudan a zonas en las que la población sea mayoritariamente del otro grupo, etc. Además, en cada etapa de la simulación, cada agente decide si se mudará o no, de acuerdo a una probabilidad que depende también de lo acorde

a sus preferencias que sea su situación actual. Si decide mudarse, busca una casa de entre las disponibles, elaborando una lista de las mismas por orden de atractivo, e intenta ocuparlas en ese orden con cierta probabilidad. Además, en el modelo se simulan migraciones fuera y dentro de la ciudad, pero esto sucede simplemente evaluando una probabilidad.

Formación de parejas

Otro campo muy estudiado son los mecanismos que hay tras la formación de parejas.

MADAM (Marriage and Divorce Annealing Model)[8] [73] es un modelo que intenta explicar los procesos de selección de pareja y la dinámica de divorcios, bajo la hipótesis de que la heterogeneidad de la población es un aspecto clave. La idea es que las personas, al buscar pareja, tratarán siempre de buscar a alguien que sea similar a ellos. Y una pareja será más "fuerte" cuanto más tenga en común.

Para ello, cada individuo cuenta con *k* características de entre *N* posibles, y un nivel de exigencia *j*, que inicialmente es alto, pero que se relaja con el paso del tiempo si la persona no encuentra pareja. Cuando los agentes alcanzan cierta edad, comienzan a buscar pareja. En cada ciclo, cada agente a cierto número de personas aleatoriamente, y se casan con la persona con la que tengan más en común. No obstante, aun casados siguen buscando pareja, y si encuentran una mejor que la que tienen, se divorciarán para casarse con la nueva.

El modelo no contempla ni nacimientos, ni muertes ni migraciones, tan sólo se centra en los emparejamientos. No obstante, obtiene buenos resultados, poniendo de manifiesto la importancia de la heterogeneidad en la población para obtener tasas de matrimonio y divorcio que se ajusten a datos reales.

Otros modelos que contemplan la importancia de la heterogeneidad a la hora de buscar parejas, pero con otros enfoques. En [74], la búsqueda de pareja se hace un poco más compleja, pues ya es una búsqueda de pareja en dos sentidos (el hombre debe encontrar a la mujer aceptable, y viceversa). Además, en este modelo, los agentes pasan un periodo de adolescencia en el cual experimentan con la búsqueda de pareja pero sin llegar a casarse, y ajustan así su nivel de exigencia, en función de la gente que han conocido y tratado a lo largo de este periodo.

También se tiene en cuenta en otros estudios la importancia del entorno social a la hora de emparejarse. Por ejemplo en [75] se modela la influencia de la presión social a la hora de emparejarse. Para ello, se introduce una red social para cada agente, que serán sus conocidos. Además, mantiene cada agente otra red social formada por las potenciales parejas. La probabilidad de casarse dependerá, en primer lugar, de la disponibilidad de parejas posibles y, en segundo lugar, del deseo que tenga la persona de casarse, que a su vez vendrá dado en gran medida por la cantidad de conocidos que ya estén casados. El modelo

[8] Es precisamente en este modelo en el que está basado el presentado en el Apéndice B, aunque ampliado.

incluye también nacimientos, pues una vez casados, las parejas tienen hijos con una probabilidad que es proporcional a la edad de la madre. Los hijos heredan las características de los padres y son situados aleatoriamente en el espacio, que no es sino un espacio unidimensional con topología de circunferencia.

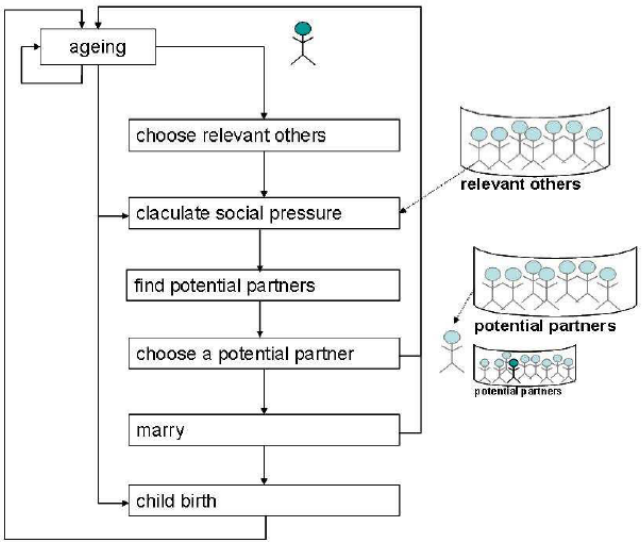

Figura 3.2. *Ciclo de búsqueda de pareja en [75].*

El ejemplo anterior pone de manifiesto la importancia de las redes sociales y su estructura a la hora de simular procesos demográficos, elemento que supone una "ganancia" añadida de los ABM sobre otras técnicas como microsimulación [18].

MODELOS DEMOGRÁFICOS

Los modelos que se han visto en el apartado anterior, a menudo no pueden considerarse modelos demográficos como tales, si acaso modelos basados en agentes sobre aspectos demográficos. Esto es así porque muchas veces no contemplan ni las muertes, ni los nacimientos ni las migraciones, o ninguno de los 3, elementos fundamentales de la ecuación compensadora (Población final = Población inicial + Nacimiento – Defunciones + Inmigraciones – Emigraciones), como se vio en el capítulo anterior.

Los modelos que se verán a continuación, por el contrario, son modelos demográficos "completos", en el sentido de que incluyen todas estas variables, fundamentales por otra parte si se quiere obtener un comportamiento realista de una población a largo plazo.

Sin embargo, y como se verá, muchas veces han de recurrir a aproximaciones de arriba abajo. Por ejemplo, la búsqueda de pareja puede estar guiada por el comportamiento de los

propios agentes, si la manera en que se forman las parejas tiene una influencia que se quiere estudiar; por otro lado, se pretende que la simulación se prolongue en el tiempo durante varias generaciones, y a la vez, se quiere que la natalidad se ajuste a unos valores dados porque son realistas y no son relevantes en el estudio. En este caso, se impondrá una tasa de natalidad determinada que se ajuste a un valor macro observado.

Se pasa a detallar algunos de los modelos estudiados representativos:

- **MameLuke settlements model** [76]: Es un modelo que simula la dinámica de asentamientos de varios grupos étnicos en la región de San Mariano, Filipinas, durante el periodo 1900-1992.

  Los agentes forman hogares, que son los que deciden en qué lugar establecerse y expandir las tierras de cultivo. Sin embargo, los procesos demográficos como muerte, nacimientos y matrimonios se producen a nivel de los individuos.

  Cada año en la simulación, se introducen nuevas familias de acuerdo a una serie de datos históricos, y éstas buscan asentamiento de acuerdo a los siguientes criterios: 1) Proximidad de miembros de su mismo grupo étnico. 2) Disponibilidad de un río cercano. 3) Una serie de datos acerca de la parcela, como inclinación del suelo, calidad del mismo, etc. Una vez que están asentados, intentarán ir expandiendo sus tierras de cultivo, que irán aumentando año a año.

  La mortalidad se modela mediante unas tablas que establecen la probabilidad de morir a una determinada edad.

  Los matrimonios se forman de la siguiente manera: con una cierta probabilidad establecida que depende de la edad, un hombre soltero de entre 15 y 45 años buscará una mujer de su etnia que sea entre 0 y 4 años menor que él. Si la mujer tiene 15 o más, le aceptará y formarán un nuevo hogar.

  Los nacimientos se producen de acuerdo a una probabilidad que depende de la edad de la mujer. Al nacer, el hijo pasa a formar parte del hogar.

  Finalmente, los hogares se disuelven cuando mueren el padre y la madre. En ese caso, las tierras de cultivo se reparten a partes iguales entre todos los hijos.

  Además de esto, el modelo incluye un módulo GIS que representa el entorno real de la zona.

- **IMSHED (Integrative Model for Simulating Household and Ecosystem Dynamics)** [77]: Es un modelo que simula y estudia el impacto de la creciente población rural en el hábitat de los pandas en el Parque Natural de Wolong. Este impacto se produce sobre todo porque la población

necesita madera como combustible, y esta madera es obtenida de los bosques del parque.

El modelo es bastante complejo, pues incluye una gran cantidad de factores que influyen en el consumo de madera. Además, está dividido en 3 módulos, el módulo socioeconómico, que es el que controla y calcula el consumo de madera de una familia y el consumo de electricidad, pues una familia puede o no cambiar su fuente de energía, dependiendo de diversos factores. Además, incluye un módulo del entorno, en el que se modela el crecimiento de los bosques, los caminos a través del bosque hasta el lugar de recolección de la madera, y la elección de estos lugares, además de la elección de asentamientos apropiados para los nuevos hogares. Finalmente, incluye un detallado modelo demográfico, que se pasa a explicar con más detalle.

Algunas decisiones que tienen que ver con el consumo de madera y otros aspectos de la simulación, se toman a nivel de hogar. Un hogar está formado por personas, y los procesos demográficos se producen al nivel de personas.

La probabilidad de que una persona muera, depende de su edad. En caso de que alguien muera, si estaba casado, su consorte pasará a formar parte de los solteros de nuevo. La otra manera en que la gente puede salir del sistema es emigrando, y puede producirse por dos razones: si la persona tiene entre 16 y 20 años, hay una probabilidad determinada —la tasa de población que tiene estudios superiores— de que emigre porque va a ir a la universidad; por otro lado, si la persona tiene más de 22 años, hay una probabilidad de que se case con un forastero y abandone la zona.

Los matrimonios se producen con una probabilidad que depende de la edad —siempre a partir de 22 años—. Los matrimonios pueden ser entre un local y un forastero —única manera en que se puede producir la inmigración—, o entre dos locales. El artículo no deja muy claro la mecánica en que se producen los emparejamientos, sólo habla de que con cierta probabilidad se producen. En cualquier caso, una vez se establece una nueva pareja, ésta permanecerá en la casa paterna de él o ella, dependiendo de ciertas condiciones como número y sexo de los hermanos, etc., que tampoco merece la pena detallar.

En cuanto a los nacimientos, se producen con una cierta probabilidad dada por una distribución binomial, cuyos parámetros dependen de características propias de cada individuo y que le son asignadas al nacer, como pueden ser número de hijos que tiene intención de tener a lo largo de su vida, o intervalo entre hijo e hijo.

Finalmente, se tiene la dinámica de los hogares. Éstos aumentan o disminuyen el número de miembros, o son creados o desaparecen de acuerdo a lo que pase al nivel de los individuos que los forman —si nacen, mueren, emigran, etc. —. Por otro lado, cuando un nuevo hogar es creado, se le asigna un lugar cercano al lugar del domicilio paterno, y se le asigna una porción del terreno cultivable de éstos.

- En el campo de la simulación histórica, diversos modelos tratan de reconstruir sociedades antiguas y conseguir que los patrones de asentamiento y población de los modelos se ajusten a los datos arqueológicos observados. Este es uno de los campos en los que más importancia puede tener el modelo demográfico, pues suelen simular períodos de tiempo largos, de varios siglos. Un ejemplo es [78], que trata de explicar la distribución y tamaño de los asentamientos de los indios Pueblo en la región de Mesa Verde, en Colorado, E.E.U.U. a través de los datos sobre la disponibilidad de agua. Este modelo es muy similar al que se ha implementado [5], por lo que los detalles se verán en capítulos ulteriores.
- **Modeling Ancient Settlement Systems (MASS)** [79]: En la línea de los anteriores, se trata de un proyecto de la Universidad de Chicago para estudiar asentamientos en la antigua Mesopotamia. Es un modelo bastante completo que incluye, agricultura, pastoreo, comercio, redes sociales de familiares, además de un entorno muy detallado con un módulo del clima, de la dinámica del crecimiento de la vegetación, erosión del suelo. También cuenta con una estructura social en la que los individuos forman familias, que a su vez forman una comunidad que influye en los individuos, aportando *feedback*, característica fundamental de los ABM. En [80] se detalla el modelo, que incluye un módulo demográfico que contempla tasa de natalidad, tasa de mortalidad que depende de la edad y el sexo, cambios en los roles de los individuos, de acuerdo a su edad y género, matrimonio, herencia, y reestructuración de los hogares —creación, destrucción y fusión de los mismos—.
- **Mentat** [71]: Enmarcado en un proyecto más amplio de tesis de doctorado, Mentat es un modelo basado en agentes y dirigido por datos[9] que estudia, con muy buenos resultados, la evolución de una serie de valores en la sociedad española en el período 1980-2000, basándose en datos de

[9] Esto quiere decir que, a diferencia de la mayoría de modelos de ABM —que toman un enfoque más *dirigido por la teoría*, en el que se centran en plantear hipótesis de comportamiento generalmente simples que tratan de validar resultados macroscópicos— se centra más en obtener un modelo que sea *descriptivo*, para lo cual se dota a los agentes de características obtenidas mediante datos reales e individuales, como puedan ser encuestas.

encuestas. Es un modelo complejo que incluye herramientas de Inteligencia Artificial.

El modelo demográfico que incluye, del que se destaca la gran importancia de que resulte realista, tiene un doble enfoque. Por un lado, las relaciones sociales se modelan de abajo a arriba, cada se relaciona con sus "vecinos", los otros agentes que tiene alrededor. Y cuanto más parecidos de acuerdo a una función de similaridad —que toma en cuenta atributos de los agentes— son, más probabilidades de convertirse en amigos tienen. Con el tiempo, además, cuanto más parecidos son dos amigos, más se refuerza su amistad. De estas amistades pueden surgir relaciones de pareja, pues los agentes buscan pareja filtrando a los posibles candidatos de entre su conjunto de amigos, para elegir luego a aquel con el que más tienen en común. A partir de aquí, el enfoque demográfico pasa a ser de arriba abajo, pues las probabilidades y número de hijos que tenga una familia responden a ecuaciones obtenidas a partir de datos estadísticos, así como la esperanza de vida que tengan, etc.

3.4. CONCLUSIONES

En las páginas anteriores, se ha visto que los modelos demográficos tienen una importancia fundamental en muchas de las aplicaciones de la simulación basada en agentes. Sobre todo en aquellas que simulan períodos largos de tiempo, pues la necesidad de aportar un comportamiento demográfico realista o, cuando menos, completo es básico en estos casos.

Se ha visto también que muchas de las aplicaciones de este tipo tienen que ver, bien con modelos de uso del suelo, bien con simulaciones históricas. En este tipo de modelos, cobra un papel fundamental el entorno, el realismo del mismo y su evolución e interacción con los agentes, pues este tipo de modelos son altamente específicos y dependientes de las condiciones geográficas. Por ello, es usual también que se integren con GIS [81].

También es muy frecuente en estos modelos que los individuos se organicen en varios niveles, sobre todo individuos y hogares, aunque a veces también estructuras sociales más amplias.

Además, estos modelos, que cada vez son más complejos y en un futuro lo serán aun más [44], debe contemplar una gran variedad de posibles agentes y por tanto la heterogeneidad de los mismos es clave. Sin embargo, esta heterogeneidad no ha de restringirse únicamente a características o atributos elementales de los mismos, como edad, situación laboral, etc., sino que debe incluir también a los comportamientos de los agentes, y permitir que estos cambien dinámicamente durante la ejecución de los modelos.

Por otro lado, el estudio de otras técnicas de modelado de la demografía, nos aporta técnicas que deberán de aplicarse casi con total seguridad a la hora de diseñar un modelo demográfico con agentes. Ya se ha comentado que muchas veces se tiene la obligación de modelar de arriba abajo, y para ello los modelos demográficos matemáticos son fundamentales. Algo similar sucede con las técnicas de microsimulación: muchas veces no se puede permitirse modelar todos los factores demográficos mediante complejas reglas de comportamiento, pero sí es posible tomar unos pocos factores —características individuales— en cuenta y, ayudados de la estadística, modelarlos de forma estocástica.

No obstante, el estudio de modelos basados en agentes en que se modelan con reglas los comportamientos demográficos ha enseñado también que, cuando se necesita adoptar este enfoque puro de ABM, ahí está disponible. En estos casos, sobre todo a la hora de modelar relaciones y emparejamientos, cobra una gran importancia la red social de cada agente.

En resumen, las características básicas que debería incluir un modelo demográfico basado en agentes son: módulos de natalidad (y/o de formación de nuevos hogares), mortalidad y migraciones lo bastante genéricos, flexibilidad de comportamientos de los agentes, red social que gestione relaciones que bien pueden ser de amistad, familiares u otro tipo (flexibilidad). En cuanto a los atributos que deberían contemplarse, aunque ya son algo más secundarios y por tanto deberían ser opcionales, estarían el status social, el nivel de educación y el estado de salud, que podrían variar y depender de condiciones ambientales. Como último punto, la interacción con el entorno ha de ser contemplada, y éste ha de poder cambiar, no ser estático.

En cuanto a los atributos que deben tener los agentes, al menos aquellos a los que se pretenda dotar de un carácter más social, han de contemplarse, por lo que se ha visto tanto aquí como en el Capítulo 2, los siguientes: sexo y edad, nivel económico o de renta, nivel cultural o sociocultural. Estos son fundamentales, y recomendable sería añadir también la actividad profesional y creencias religiosas.

Finalmente, para justificar la elección de MASON como plataforma elegida frente a otras, se ha tomado en cuenta, sobre todo, la rapidez de ejecución que tiene, la modularidad, su escalabilidad y la buena integración que tiene con otras aplicaciones, al ser un conjunto de librerías fácilmente invocables desde otros proyectos Java. Modelos demográficos es de esperar y deseable que puedan soportar un gran número de agentes, sobre todo si pretenden modelar aspectos realistas. Por otra parte, la modularidad es también uno de los objetivos que se pretenden alcanzar en el trabjo, y la posibilidad de ampliarlo en un futuro añadiendo nuevas funcionalidades también está facilitada gracias a la estructura de MASON. Por estas razones, parece adaptarse muy bien a los requisitos del proyecto.

4. DESCRIPCIÓN DE LA PROPUESTA DE ARQUITECTURA

4.1. INTRODUCCIÓN Y ESTRUCTURA DEL CAPÍTULO

En esta sección se aborda la descripción del diseño de una arquitectura general para hacer modelos demográficos basados en agentes. Ésta ha sido diseñada contemplando los objetivos que se comentaban en la Introducción, y por lo tanto se basa en gran parte en el estudio previo realizado, y que se ha expuesto en los dos capítulos anteriores.

En primer lugar, se ofrece una visión global, introduciendo la plataforma MASON en estructura y funcionamiento lo mínimo como para poder seguir el resto del diseño sin problemas. Una vez introducida, se comentan los componentes principales definidos en la arquitectura y su relación.

En el resto del capítulo se exponen los diferentes componentes de la arquitectura en detalle, justificando las decisiones tomadas. Éstos son: el modelo y el entorno (los factores demográficos), los agentes, los comportamientos y las redes sociales.

4.2. LA PLATAFORMA MASON

Se presenta a continuación una breve descripción de la estructura y el funcionamiento de MASON[10] para que la descripción posterior de la arquitectura resulte comprensible. Aunque se ha intentado abstraer en la medida de lo posible, no deja de estar construida sobre MASON y las únicas pruebas que se han realizado han sido sobre MASON también.

MASON es una plataforma de desarrollo de modelos de simulación organizado en forma de clases y librerías. Es flexible pues permite crear modelos basados en agentes, además de otros tipos, y puede usarse incluso para crear videojuegos. Está construida sobre Java, y su núcleo está diseñado con el objetivo de ser rápido y ligero —para poder soportar simulaciones con "hasta millones de agentes" [57], y permitir un gran número de simulaciones para realizar análisis de sensibilidad de los parámetros—, ser portable entre plataformas de *hardware* —permite parar una simulación, trasladarla a otro ordenador y

[10] MASON está disponible, junto a su documentación, en la web [46].

continuarla *exactamente igual* desde el punto en que se quedó— y separar muy claramente el modelo en sí de su visualización.

MASON está formada por tres capas: la capa del modelo, que incluye el "motor" necesario para realizar las simulaciones, la capa de visualización, que permite añadir interfaces gráficas para manejar y visualizar el modelo en 2 y 3 dimensiones, y una capa de utilidades. MASON también hace uso de otras librerías —tanto propias como de terceros— para ofrecer otras utilidades como redes sociales, generación de gráficos, vídeos e imágenes de la simulaciones, integración con GIS, etc.

En la Figura 4.1 se muestra la estructura que tienen los modelos en MASON. En primer lugar se tiene la clase ***SimState***. Esta clase no es sino un envoltorio de todo el modelo. Se ha comentado ya el hincapié que hace MASON en separar el modelo de la visualización. Pues bien, *SimState* es el modelo —de la visualización no se hablará aquí, pues carece de interés para los propósitos de esta memoria—. *SimState* encapsula los demás elementos que forman el modelo.

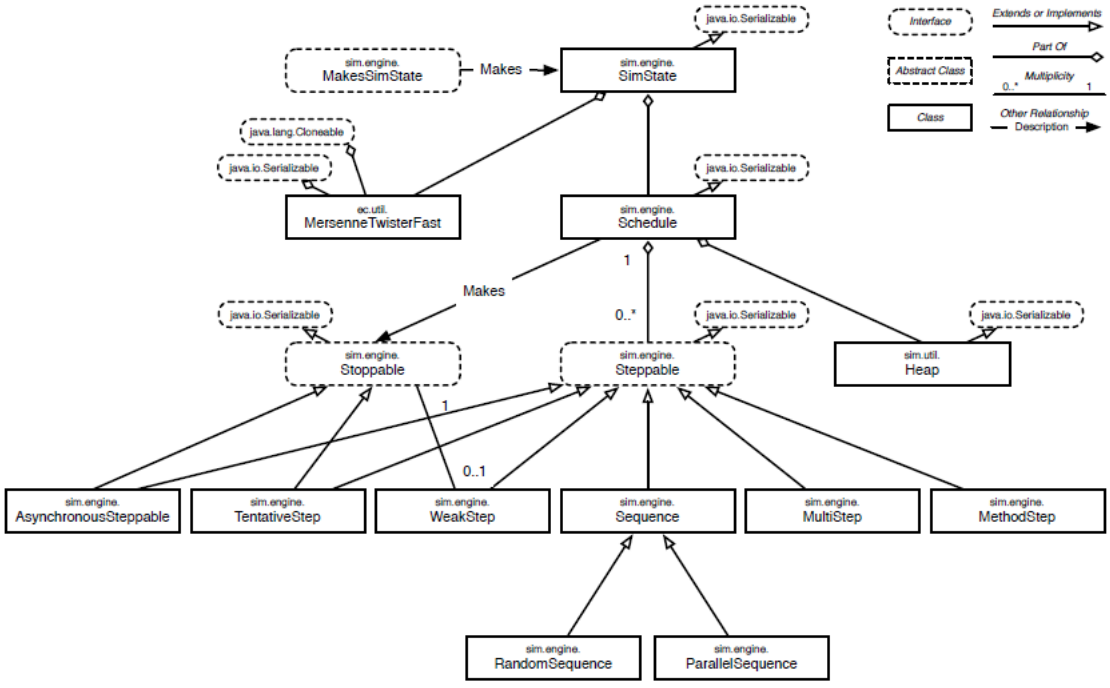

Figura 4.1. *Estructura de una simulación con MASON y el planificador de eventos discretos. (Tomado de [82]).*

El primero de estos elementos es la representación del tiempo de MASON, un planificador de eventos discretos[11] (*discrete-event schedule*), que viene dado por la clase ***Schedule***. Los actores de la simulación —generalmente los agentes u otros entes dinámicos que se quiera que cambien con el tiempo, como el entorno o un "observador" que se encargue de controlar las estadísticas, por ejemplo— se han de colocar aquí para que MASON los ejecute. Para ello, deberán implementar la *interface* ***Steppable***, que en esencia define un método *step()* que es el que es invocado desde el planificador de eventos discretos cuando es su turno.

Por otro lado, está la representación del espacio. En MASON se llaman ***fields*** y son desde *grids* de 2 o 3 dimensiones que simulan espacios discretos, hasta otras representaciones que simulan espacios continuos, además de grafos en las que los agentes son los nodos y las aristas las relaciones entre ellos. En cada modelo puede haber uno o varios de estos, y es donde se situará a los agentes "físicamente".

Finalmente, todo modelo de MASON incluye un generador de números aleatorios (clase *MersenneTwisterFast*) basado en el algoritmo *Mersenne Twister* [84].

Además de estos tres elementos que encapsula *SimState*, cuando un usuario quiere hacer su propio modelo ha de extender esta clase, y es en la subclase donde incluirá cualquier otra estructura de datos o parámetro que requiera su problema.

4.3 VISIÓN GLOBAL DE LA ARQUITECTURA

Una vez introducido MASON, se pasa a comentar la estructura general de la arquitectura propuesta. Aunque muy simplificado, la Figura 4.2 muestra un diagrama de clases del modelo demográfico propuesto.

En primer lugar, se puede observar la clase ***SimpleWorld***. Esta clase extiende a la Clase *SimState* de MASON. Sigue la misma filosofía que tiene en MASON, así pues será un contenedor o contexto para el resto del modelo. La diferencia es que aquí se le dota además de los elementos básicos que precisa un modelo demográfico: información acerca de fecundidad, mortalidad, nupcialidad y migraciones.

Además de los módulos demográficos, a la clase anterior se le añade un ***AbstractField2D***, que es la clase elemental que se ha empleado en este trabajo para el entorno "físico" de la simulación. En realidad, lo único que hace esta clase es encapsular uno de los *fields* de MASON. En concreto la *interface Grid2D*, que define estructura y operaciones básicas para *grids* en dos dimensiones. Estos son la representación más usual que se emplea en ABSS, sobre todo en los problemas "grandes" y realistas que se pretenden abarcar aquí —es cierto que no la única, se ha visto en [75] que se usa una estructura unidimensional—, y

[11] Esto convierte a MASON en un *simulador de eventos discretos*, que no es sino una manera de hacer simulación en la que el tiempo se representa como una secuencia cronológica de eventos que van siendo ejecutados uno detrás del otro en ese mismo orden [83].

desde luego los más apropiados, por su similitud topológica, para poder emplear GIS, aunque esto quede de momento fuera de los objetivos del trabajo. No obstante, y puesto que MASON provee de herramientas para ello, otras representaciones son fácilmente implementables.

Otro de los elementos que pueden verse en la Figura 4.2 es la clase ***DemographicItem***. Representa a los actores de la simulación, y se puede considerar como la unidad mínima con comportamiento demográfico, es decir, posibilidad de nacer, morir, reproducirse y migrar. Se ubica en el espacio de tipo *AbstractField2D* descrito anteriormente.

Finalmente, se observa la *interface* ***Behavior***, elemento de suma importancia en el modelo y que permite, como se verá, definir un amplio rango de comportamientos para los agentes.

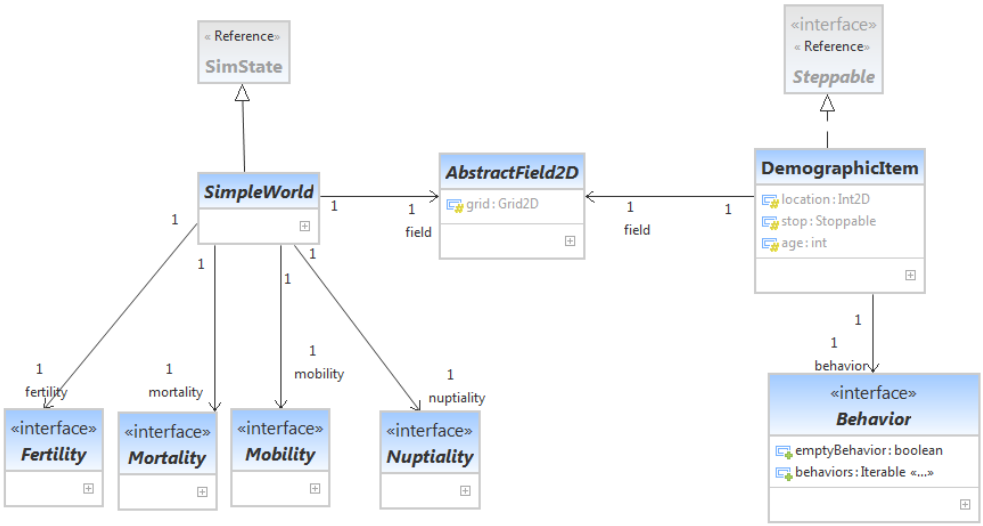

Figura 4.2. *Diagrama de clases simplificado.*

Con estos cuatro elementos que se han visto visto, se pueden realizar ya modelos demográficos. Por ejemplo, disponiendo de los datos adecuados, se podría implementar un modelo de proyección de la población mediante el método de componentes descrito en la sección 3.3.1. de manera prácticamente trivial. De hecho, estos cuatro elementos forman el núcleo de la arquitectura, y como se verá el resto deriva de ellos o los complementa.

4.4. EL MODELO Y EL ENTORNO

Siguiendo los objetivos de diseño, se ha decidido optar por dos posibilidades a la hora de realizar un modelo, tratando de ser lo más flexibles posible.

En primer lugar, se ha pensado en el tipo de simulaciones en **que no hay definida una estructura social** de interacciones, bien sean familiares u otro tipo de relaciones —generalmente amistad—. Por ejemplo, se adaptaría bien a modelos de poblaciones de corte más clásico, del tipo que usan los demógrafos para hacer proyecciones como el método de componentes, o para realizar modelos de microsimulación. Serviría además como base para la realización de modelos ecológicos y sobre poblaciones animales [85], [86]. Finalmente, se podría usar como modelo auxiliar en simulaciones que requieran de modelos demográficos simples pero realistas, y se centren en otros elementos.

Para ello, la clase *SimpleWorld* incorpora una serie de métodos básicos para añadir y quitar individuos de la simulación, además de para registrar muertes, nacimientos y migraciones.

Por otra parte, y puesto que se están tratando problemas de las ciencias sociales, se ha considerado que era necesario crear un marco que contemplara las **relaciones entre personas, como elemento fundamental en los procesos demográficos**, pero además como elemento fundamental en prácticamente cualquier estudio socioeconómico. En este tipo de simulaciones estarían englobadas la práctica totalidad de las que se han estudiado en la sección 3.3.3. Es decir, problemas en los que ha de haber un modelado rico de las relaciones entre familiares y conocidos —sobre todo ante la búsqueda de pareja—, problemas en los que hay relación social directa entre los agentes —como en [71]— y problemas en los que, simplemente, se contempla una estructura familiar y social, por simple que esta sea, como [77].

Esto ha dado lugar a la clase ***World*** y a la *interface* ***SocialWorld***, cuya relación y estructura se pueden ver en la Figura 4.3. A esta clase se le ha dotado además de una red social —clase ***SocialNetwork***—, cuya finalidad es representar el global de la población mediante un grafo en el que los nodos son las personas, y las aristas están valoradas con un tipo enumerado *Relation* —que se puede redefinir según las necesidades— e indica el tipo de relación entre dos personas.

En ambos casos ha de haber un espacio "físico", como se comentó en el apartado anterior. No se repetirá lo ya dicho, sin embargo sí comentar que, para permitir entornos que evolucionen conforme la simulación avanza, se ha incluido un *field* dinámico llamado ***MutableField2D***, cuya característica principal es que implementa la *interface Steppable* de MASON, lo cual permite añadirlo al planificador de eventos discretos y hacer que en cada iteración de la simulación pueda cambiar. Además, se ha incluido otra implementación de *AbstractField2D* llamada ***Field2D*** para tratar modelos en los que no haga falta que el entorno cambie, a no ser que sea por la influencia de los propios agentes, que siempre lo podrán hacer cambiar de una manera u otra.

El último elemento que conforma el entorno son los cuatro factores demográficos de los que se ha hablado ya en Capítulo 2. Están pensados para aportar la información demográfica necesaria para que el modelo se comporte de manera realista. Esta información demográfica puede estar en forma de diferentes indicadores (datos estadísticos de

organismos oficiales como el INE —Instituto Nacional de Estadística— [15] o la UNSD —United Nations Statistics Division— [87]), aunque será siempre información agregada, es decir, información sobre conjuntos de individuos y no sobre individuos concretos, como puedan ser tasas y probabilidades [11]. Esto quiere decir que es información de arriba a abajo, que *guiará* el comportamiento o decisiones de nuestros agentes, pero que no tomará decisiones por ellos si éstos están programados para tomar estas decisiones de abajo a arriba. En principio, estos elementos se especifican al arrancar la simulación y, aunque pueden ser lo complejos que se quiera, no se ha contemplado un mecanismo para cambiarlos a mitad de una simulación. Sin embargo, la manera de hacerlo sería, desde dentro de la instancia de *SimpleWorld* correspondiente, crear un elemento de tipo *Steppable* al que se le asignara la función de cambiar estos módulos. Más investigación en el futuro será necesaria para aclarar mejor este mecanismo, no obstante.

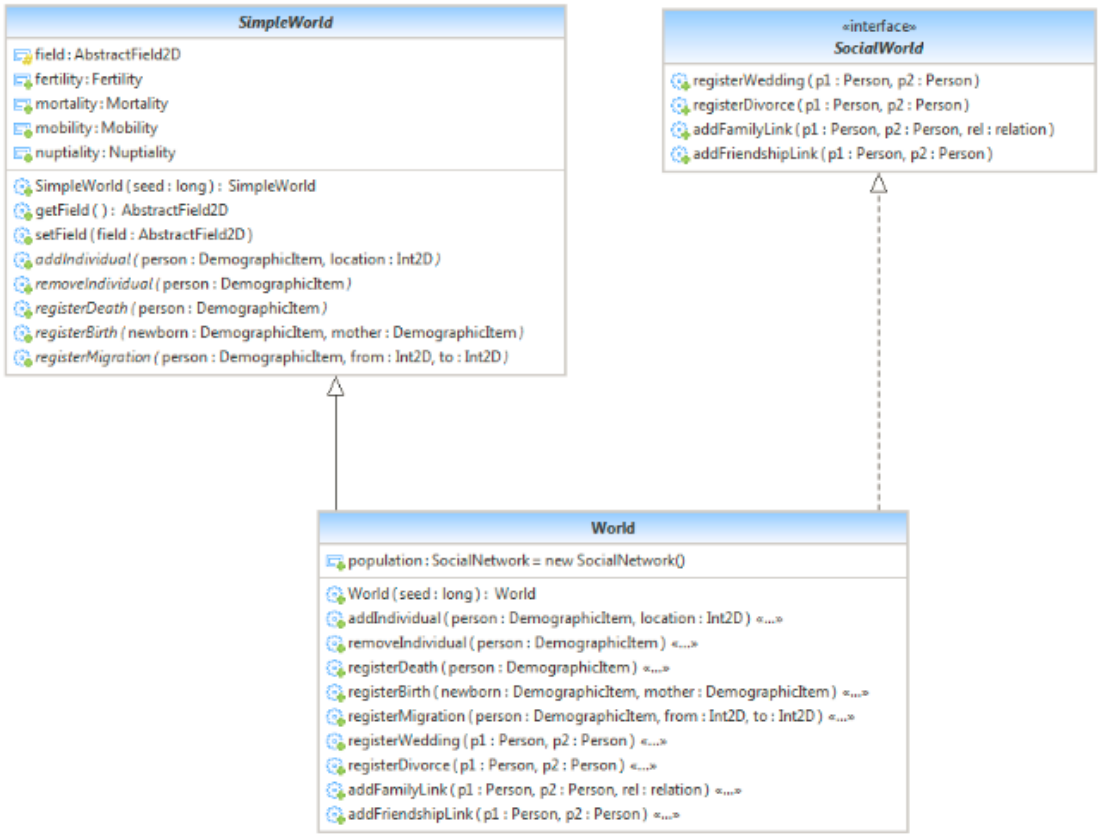

Figura 4.3. *Jerarquía de modelos.*

Es importante remarcar este punto: estos cuatro módulos demográficos no deberían incluir tomas de decisiones basadas en reglas que modelen comportamientos, pues estos han de ir en el módulo de comportamientos de cada agente —el *Behavior*, del que se hablará más adelante—. Sin embargo, si estas decisiones no son de interés en el problema, y lo único

que se quiere es que cierto comportamiento demográfico sea realista, entonces se recurrirá a ellos.

Se verá mejor lo que se quiere decir con esto mediante unos ejemplos que ilustren algunos casos posibles:

Supóngase que se está trabajando con un modelo que estudia factores que influyen en la toma de decisión de tener hijos. Estos factores pueden ser número de hijos que tiene ya la madre, cantidad de amigas con más hijas que ella, edad, situación laboral o económica. Y supóngase que la regla que dice si quiere tener otro hijo o no es la siguiente: si la mujer tiene menos de tres hijos, el 40% de sus amigas tienen más hijos que ella, tiene menos de 35 años, su situación económica es buena y tiene trabajo estable, entonces la mujer "decide" tener un hijo, independientemente de las tasas de fertilidad que existan. El módulo de fertilidad, como mucho, podrá decir si es niño o niña, de acuerdo a las probabilidades de que así sea que existan. El resto, al ser un comportamiento, irá en el *Behavior* de la mujer. La razón de esto es que, en general, los modelos que así se diseñan lo que pretenden es obtener patrones —en este caso de fecundidad— que se ajusten a la realidad en base a hipótesis de comportamientos como el que se ha detallado.

Sin embargo, sí podría ser muy útil en el siguiente ejemplo. Se quiere estudiar patrones de formación de parejas. Hay una serie de reglas que rigen el comportamiento de los agentes, cómo éstos conocen a posibles parejas, cómo se emparejan, etc. Sin embargo, se quiere dar una continuidad temporal al modelo. En este caso, a la hora de decidir cuándo y cuántos hijos tiene una mujer, será mejor dejarse guiar por los datos estadísticos sobre fertilidad, pues se asegurará así que el sistema tiene un comportamiento que se ajusta a la realidad, al menos en lo que al tener hijos se refiere. Bien es cierto que, si nuestras hipótesis sobre la formación de parejas no son correctas y se forman demasiadas, aun así el tener buenos datos sobre la fertilidad daría lugar a que la fertilidad de la población simulada fuera mayor que la observada. Sin embargo, esto sería un indicativo más de que nuestras hipótesis no están bien planteadas.

El caso extremo de uso de estos módulos será cuando no interesen lo más mínimo las razones que hacen actuar —demográficamente— a los agentes de una manera u otra, y lo único que se quiera es que actúen —demográficamente, de nuevo— de manera que se ajuste a los datos y estadísticas observados. En esta situación, los agentes recurrirán a estos módulos cada vez que tengan que evaluar si se casan, mueren, tienen descendencia o migran.

Dicho esto, hay que añadir que muchas veces no está tan claro donde terminan las reglas de comportamiento y dónde empiezan las estadísticas. Los mismos comportamientos basados en reglas necesitan muchas veces de variaciones estocásticas para dar mayor heterogeneidad al comportamiento de los agentes. O simplemente, se modelan unas partes de las reglas de manera "manual", mientras que otras se dejan al azar, o se confían a datos estadísticos. Y otras veces, tal y como ocurre con los modelos de microsimulación, se preferirá modelar estos fenómenos de manera que se produzcan mediante probabilidades

de transición entre diferentes estados (*transition rates*), que dependan de parámetros del agente en concreto.

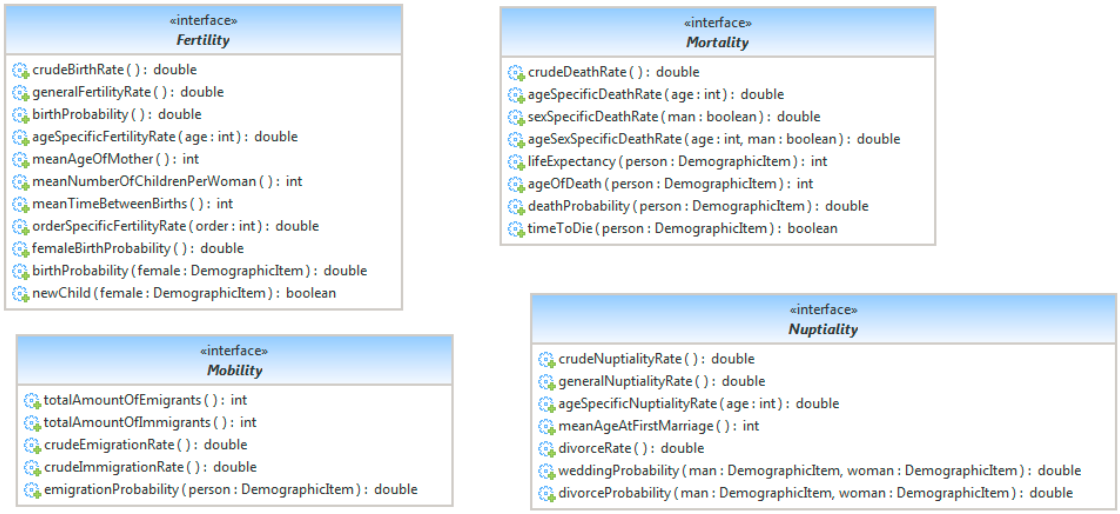

Figura 4.4. *Detalle de los módulos demográficos.*

No obstante, la distinción que aquí se hace es por claridad conceptual, y en la mayoría de los casos se puede hacer sin mayor problema.

En cualquier caso, la función de estos cuatro factores —fertilidad, mortalidad, nupcialidad y migraciones o movilidad, Figura 4.4— es dar un servicio a los agentes, a través de sus comportamientos. Se han definido mediante *interfaces* en los que se declaran una serie de métodos. Estos métodos pueden devolver *tasas*[12], independientes del agente en sí, aunque a veces tomen un entero que representa una edad como parámetro, o un booleano que representa el sexo, ya que muchas veces la información va desglosada en grupos de edad y sexo —ver el Capítulo 2—. Esta tasas lo que dan es información estadística, que generalmente se utilizará en forma de probabilidades, pero será ya el agente, de nuevo a través de sus comportamientos, el que use esta información como estime. Por otro lado hay una serie de métodos que dan información más "procesada". Estos métodos ya toman como parámetro al agente —lo cual permite, aun saliéndonos de la distinción conceptual que se proponía más arriba, que los módulos demográficos tomen decisiones por los agentes, lo que sigue siendo poco recomendable. Pero permite también las otras aproximaciones más intermedias de las que se ha hablado—, y su implementación será más dependiente del problema en cuestión.

Probabilidades de supervivencia, número de inmigrantes o emigrantes, o esperanza de vida son algunos ejemplos de estos métodos que se ha intentado que sean de utilidad en la

[12] Estas tasas, aunque en la mayoría de la literatura sobre demografía se expresan en tanto por mil, aquí se ha optado por expresarla como un valor real entre 0 y 1, pues generalmente se les dará un uso de "probabilidad".

creación de modelos con agentes. Se ha tratado de que estos métodos den la información más general posible. De los modelos estudiados en la literatura, en todos se podía recuperar la información que empleaban, si no directamente, indirectamente, a través de la información que dan estos métodos.

4.5. LOS AGENTES

El diseño de los agentes se ha realizado en paralelo al del entorno, como con *SimpleWorld* y *World*. En primer lugar se tiene la clase *DemographicItem*, que representa a un agente con comportamiento demográfico. Este agente, que bien podría ser un animal, una persona o un ente imaginario, dependiendo de la simulación que se esté realizando, tiene como características fundamentales que se puede ubicar en un espacio de tipo *grid* bidimensional y que tiene comportamiento, que generalmente será de tipo demográfico al menos, pero que puede complicarse tanto como se quiera —se hablará de los comportamientos en la sección siguiente—. Una simulación que emplee estos *DemographicItem* como agentes se situará generalmente en una instancia concreta de *SimpleWorld*.

Para continuar con el paralelismo, se tiene a continuación la clase ***Person***. Esta clase representa a un agente demográfico con un comportamiento social, para lo que implementa la *interface* ***Socializable***, que define los métodos básicos de interacción social entre personas (Figura 4.5).

Por tanto los agentes que sean de tipo *Person* y, por extensión, cualquiera que implemente la *interface Socializable* se ubicarán naturalmente en un entorno de tipo *SocialWorld*, pues este contiene los métodos y recursos necesarios para manejar este tipo de agentes. Por otra parte, a la clase *Person* se la ha dotado también de dos redes sociales para gestionar sus relaciones sociales: una para sus familiares y otra para sus amigos/conocidos.

Se ha completado la clase *Person* dotándola de tres atributos que permitan tomar en consideración sus características socioeconómicas más significativas a nivel de influencia demográfica [11]. Estos tres atributos vienen dados por sendas *interfaces* que los definen de manera muy somera, pero a la vez flexible para posibles implementaciones de las mismas. Si no se desean incluir porque el problema no contempla tales aspectos, se podrían ignorar.

Finalmente, se ha definido la *interface* ***Household***. Está pensada para ser un elemento demográfico de segundo orden: si se desea hacer una simulación en la que los hogares —*Households* en inglés— sean la unidad mínima, se debería emplear *DemographicItem* para modelar el agente *hogar*. Por otro lado, si se desea que la unidad mínima sean personas, pero que a su vez estas estén organizadas en hogares, y se quiere que estos hogares puedan

disponer a su vez de propiedades y comportamientos propios, se debería emplear una clase que herede de *DemographicItem* e implemente *Household*.

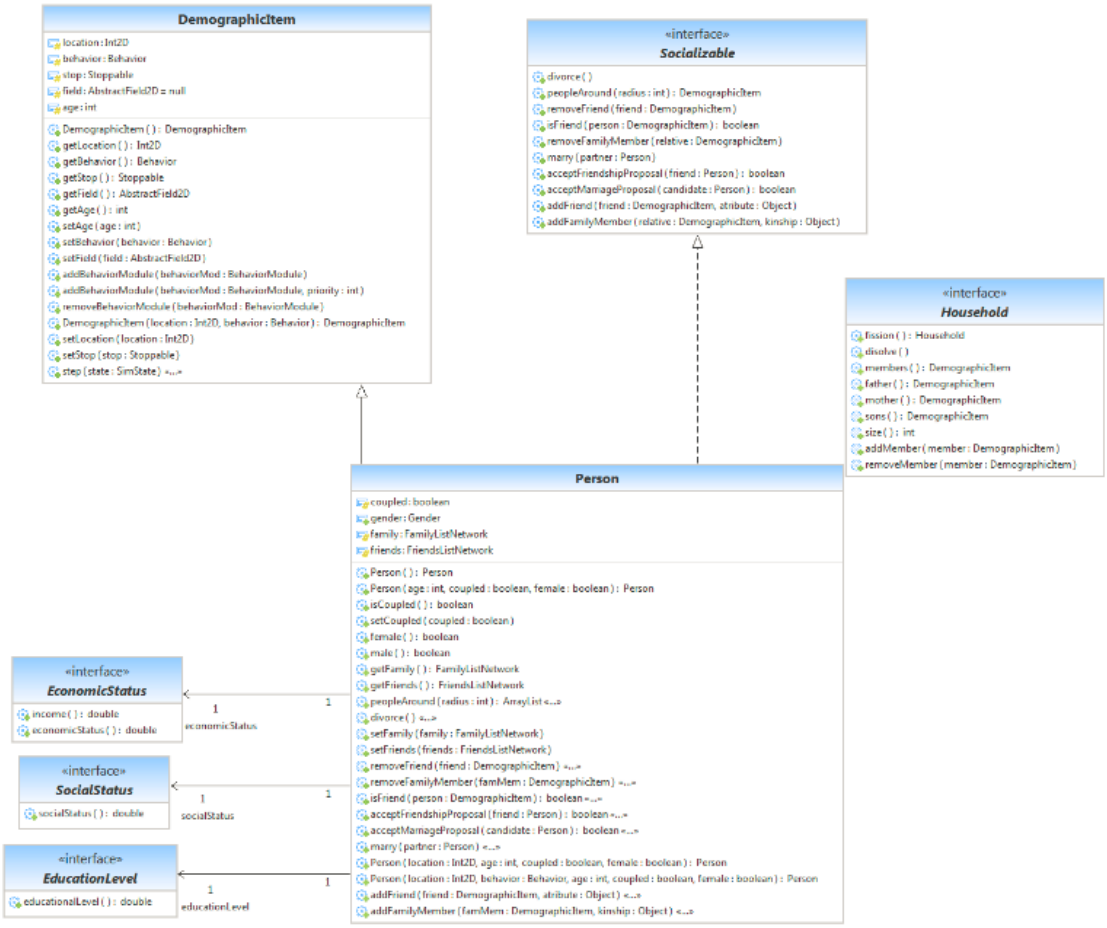

Figura 4.5. *Tipos de agentes.*

4.6. LOS COMPORTAMIENTOS

Una de las partes más importantes la simulación social basada en agentes son los propios comportamientos de los agentes, que permiten dar forma a las hipótesis que se quieren estudiar.

El objetivo aquí ha sido tratar de dotar a los agentes de flexibilidad a la hora de incorporar comportamientos. Con flexibilidad se quiere decir dar cabida a una amplia gama de comportamientos que, además, pueden ir separados en módulos. De esta manera se pueden combinar e intercambiar durante el ciclo de vida de un agente. Esto, además de dotar de heterogeneidad a los agentes, permite separar conceptualmente los

comportamientos por “tipos”, como comportamiento social, comportamiento demográfico, etc.

Algunos autores señalan que en la simulación social bien no se ha dado demasiada importancia a modelos cognitivos complejos y se ha optado por hacerlos “extremadamente simples” [88] o que, simplemente, no tiene por qué ser necesarios en todos los casos [89]. En cualquier caso, al diseñar los comportamientos de los agentes en este trabajo se ha dejado un tanto de lado qué modelo cognitivo y cómo de complejo o simple ha de ser.

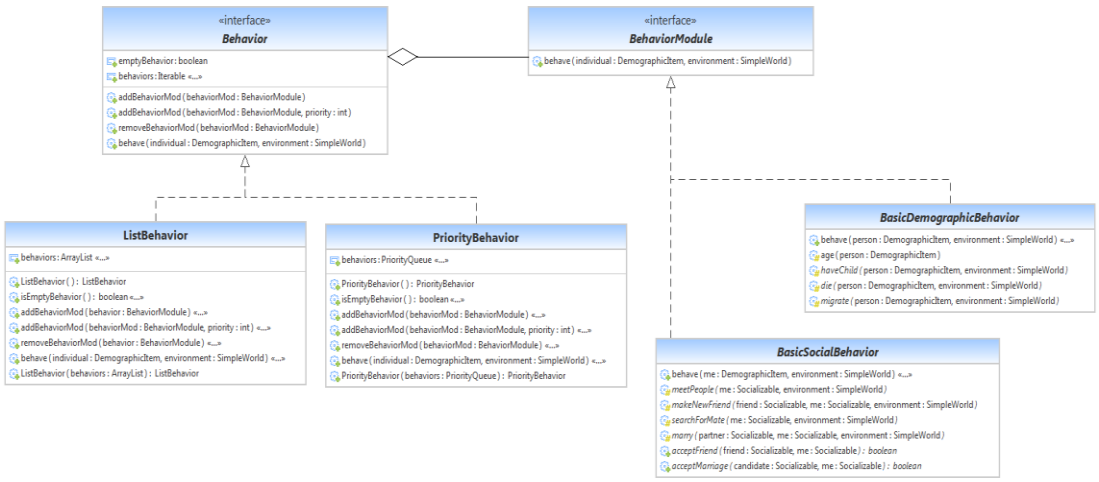

Figura 4.6. *Estructura de los comportamientos.*

La estructura básica que tienen se puede ver en la Figura 4.6. Por un lado se tiene la *interface* ***Behavior***. Como se muestra en la Figura 4.2, todo *DemographicItem* tenía como atributo un elemento de tipo *Behavior*. En realidad, no es sino un contenedor para los diferentes módulos de comportamiento —elementos de tipo ***BehaviorModule***— que define métodos para añadir y quitar módulos y para acceder a ellos. *ListBehavior* es una implementación de *Behavior* que emplea una lista para almacenar los módulos. En ocasiones será suficiente, sobre todo si no se le da importancia al orden en que deban de ejecutarse los diferentes *BehaviorModule*.

No obstante, y dada la influencia que pueda tener el orden de ejecución de los comportamientos en el resultado de la simulación, se ha incluido la clase *PriorityBehavior*, que permite definir un orden de ejecución mediante un entero que represente la prioridad, e implementado con una cola de prioridad.

Por otra parte se tiene a los comportamientos en sí, definidos mediante la *interface BehaviorModule*, que declara un único método: *behave()*. De aquí derivan las clases abstractas ***BasicDemographicBehavior*** y ***BasicSocialBehavior***. Como su nombre indica, estas clases sirven de base para comportamientos demográficos y sociales más complejos. Una vez más, la primera, que tiene que ver con aspectos demográficos, estará relacionada con

DemographicItem, mientras que la segunda será ya propia de agentes sociales como *Person*. En estas clases se ha definido el método *behave()* siguiendo el patrón *template method* [90], de manera que será en las implementaciones concretas de las mismas donde se defina el comportamiento, aunque ya definan ellas un cierto orden en la ejecución de los métodos propios de las mismas. Este orden está basado en el orden típico de ejecución que se ha encontrado en los modelos analizados, aunque es un aspecto que pude depender en gran medida del modelo en cuestión. En tal caso basta con reescribir *behave()* o instanciar directamente *BehaviorModule*, que es en realidad el elemento importante de este bloque.

Al igual que se ha puntualizado anteriormente la función de los componentes demográficos del modelo y su relación y posible conflicto de competencias con los comportamientos se hace a continuación la siguiente observación. Por claridad conceptual, han de ser los comportamientos —los elementos de tipo *BehaviorModule*— los que evalúen la situación y tomen las decisiones pertinentes, y no los agentes. Obviamente, los comportamientos tendrán que tener acceso a los atributos de los agentes y posiblemente al entorno que los rodea. Y es evidente que, en última instancia, tendrá que ser un método propio de la persona el que añada o quite a otra de entre sus amigos o familiares, pero esto debería ser porque el método, después de "decidir" que esa persona es buena o mala como amigo, invoca al método añadir o quitar amigo.

4.7. LAS REDES SOCIALES

Las redes sociales son elementos indispensables en un prácticamente cualquier modelo de simulación social [91]. Esto viene dado porque son la estructura más natural a la hora de representar relaciones entre personas, ya que *emergen* de manera natural como estructura a partir de las simples interacciones entre individuos ([92], [93]). Por ello son tanto objeto de estudio en sí mismas ([32], [94], [95]) como estructura "física" en la que situar a los agentes [96], o modelar las propias estructuras sociales.

En este trabajo se han incluido como herramienta para facilitar, por un lado la posibilidad de estudiar la composición de la población y su estructura, y por otro facilitar que los agentes sociales mantengan un registro de sus amistades o familiares. Por ello se ha dado un doble enfoque. En el primer caso, se ha incluido en la clase *World* —que si se recuerda, implementaba la *interface SocialWorld*— como una estructura de grafo. De esta manera se podrá estudiar la red de manera global.

Sin embargo, no es de manera global como actúan los agentes, así pues la representación que ellos usan de su red social, simplemente es una lista de amigos/familiares, en la que puede haber atributos para dotar de mayor riqueza y versatilidad a estas redes (Figura 4.7). Lo interesante de esto, como se ha dicho, es que a

pesar de la diferencia entre las dos estructuras, la primera surge de manera natural a partir de las segundas.

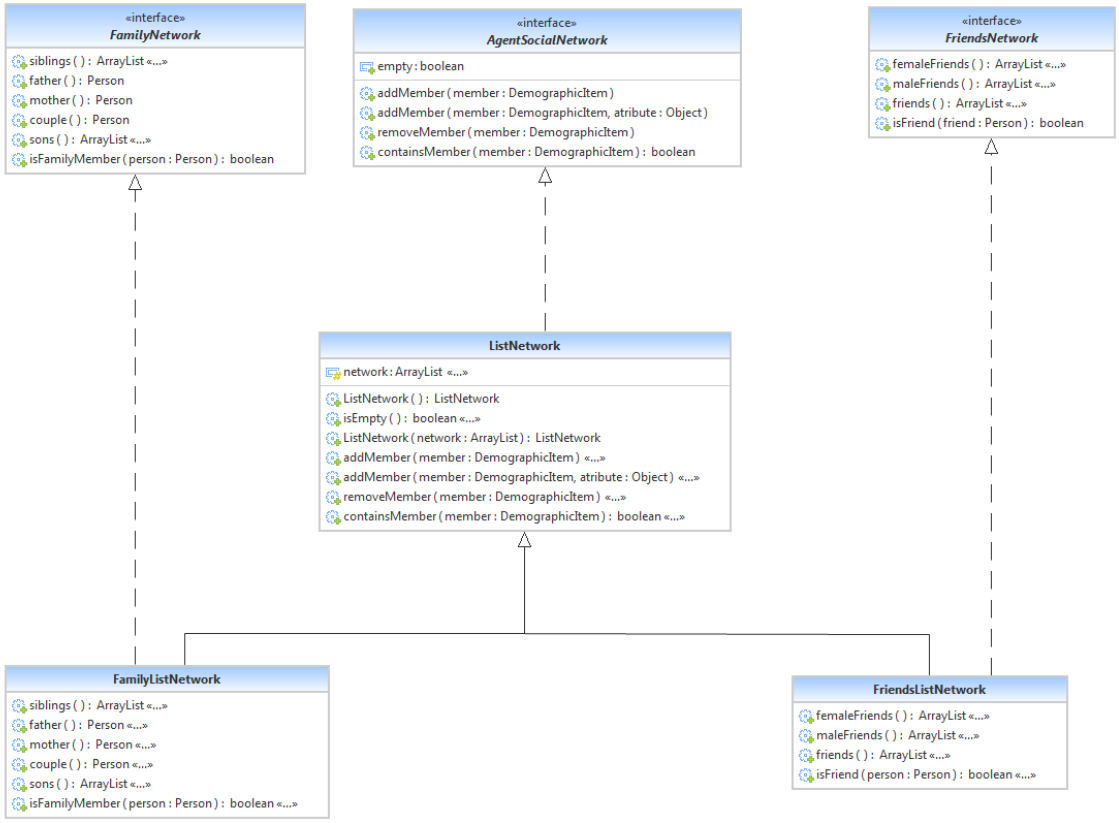

Figura 4.7. *Redes sociales a nivel de agente.*

5. CASO DE ESTUDIO: REPLICACIÓN DEL MODELO ARTIFICIAL ANASAZI

En el presente capítulo, se expone el modelo elegido para ser probado con la arquitectura que se ha definido previamente. Se introduce brevemente el tema de la replicación de modelos en ABM, y posteriormente se da una descripción del modelo Artificial Anasazi ([4], [5], [6]) siguiendo el protocolo ODD (descrito en [8] y [9]).

Una vez definido el modelo, se describirá su implementación y se comentará el resultado obtenido al adaptarlo a la arquitectura propuesta.

5.1. REPLICACIÓN DE MODELOS EN ABM

Replicar un modelo, como replicar un experimento, consiste en "repetir" el original. Su importancia es manifiesta en las ciencias experimentales donde la posibilidad de replicar un experimento y obtener los mismos resultados de manera repetida es necesario para asegurar que el resultado original no se trata de un caso aislado o una coincidencia ([97], citando a [98]). Y la simulación basada en agentes, como metodología, está muy próxima y tiene más en común con las ciencias experimentales que con las deductivas [19].

Es por ello que la replicación de los modelos basados en agentes se hace especialmente necesaria a la hora de confirmar y respaldar los resultados obtenidos por otros previamente —además de validar, verificar y profundizar en la comprensión del modelo [97]—, pues es posible que algunos resultados publicados no sean correctos debido a errores de programación, mala representación de lo que se estaba simulando o simplemente errores a la hora de analizar y transmitir los resultados [20]. Algunos autores han llegado a afirmar que no se debería de confiar en los resultados de un modelo hasta que no se haya replicado [99].

Y, pese a todo, la replicación de modelos es algo aun demasiado infrecuente ([97], [99]) y visto incluso como una tarea más propia de estudiantes que de científicos [100]. Sin embargo numerosos autores están reclamando la importancia y necesidad que merece este proceso dentro del mundo de ABM ([97], [99], [100] y [101]).

Pero para replicar un modelo, debe de conocerse cómo es el modelo. El problema es que, si no se da el suficiente nivel de detalle —y aun dándose muchos detalles—, múltiples factores pueden afectar a la implementación del modelo: interpretaciones por parte del

programador, el uso de diferentes estructuras de datos, implementaciones de algoritmos, lenguajes de programación, *hardware*, sistema operativo [97].

Por este motivo, para tratar de facilitar una definición clara y rigurosa de los modelos que permita que otros puedan comprenderlos correctamente y replicarlos con éxito, una serie de propuestas han surgido. De entre ellas, el protocolo ODD (propuesto en [8] y revisado y actualizado en [9]) está creciendo entre la comunidad de ABM y ha demostrado ya su utilidad en numerosos ejemplos [102], por lo que se está convirtiendo en un estándar. Por estos motivos la definición del modelo que se ha replicado seguirá este protocolo.

5.2. DESCRIPCIÓN DEL MODELO SEGÚN EL PROTOCOLO ODD

La siguiente descripción del modelo sigue el protocolo ODD (*Overview, Design concepts, Details,* [8] y [9]) para describir modelos basados en agentes y en individuos (*agent-based* e individual-*based models*) y consta de siete elementos. Los tres primeros ofrecen una perspectiva general, el cuarto expone y explica los conceptos generales que subyacen al modelo y los tres elementos restantes dan detalles del mismo.

El modelo que se describe es una replicación de *Artificial Anasazi*, descrito en [4], [6] y [5], y replicado previamente por Janssen[13]. En [7] se exponen los resultados obtenidos por Janssen, además de la descripción del modelo que replica, validando los resultados previamente obtenidos por Axtell et al. en [4]. En este trabajo el modelo que se ha replicado ha sido el descrito por Janssen. Los resultados obtenidos se detallarán en el siguiente capítulo.

La presente descripción se basa —y debe mucho— en la que Janssen da en [7], también siguiendo el protocolo ODD. No pretende sino describir la propia reimplementación que puede variar de la anterior en algún detalle.

A continuación se detallan los siete elementos que forman la descripción según el protocolo ODD.

5.2.1. PROPÓSITO

El propósito de esta simulación es explorar y analizar las casusas de la dinámica poblacional, observada indirectamente a través del registro arqueológico, de la sociedad Kayenta Anasazi en *Long House Valley*, Colorado (EE.UU.) en el período comprendido entre 800 d.C. y 1350 d.C.

Las condiciones medioambientales como factor clave en la productividad de los cultivos y el comportamiento de los agentes mediante una serie de reglas acerca del establecimiento

[13] Esta replicación está disponible —tanto el código fuente como la documentación— en [103].

de sus asentamientos y granjas son las condiciones cuya influencia se toma en cuenta para explorar dicha dinámica poblacional, y el abandono del Valle por la población al final de ese período.

5.2.2. ENTIDADES, VARIABLES DE ESTADO Y ESCALAS

Como variables globales del modelo, se definen las siguientes:

- ***harvestAdjustment***, es un parámetro entre 0 y 1 que representa porcentaje que se aprovecha del total de la cosecha.
- ***harvestVariance***, introduce variabilidad en la obtención de las cosechas. Valor entre 0 y 1.
- ***deathAge***, la edad máxima que puede alcanzar un *household*. Es la misma para todos.
- ***fertility***, la probabilidad de que un *household* de lugar a otro, si está en condiciones de hacerlo.
- ***fertilityEndsAge***, edad máxima a la que un *household* puede dar lugar a otro. Debe ser inferior o igual a *deathAge*.
- ***spatialHarvestVariance***[14], un parámetro entre 0 y 1 que sirve para dar variabilidad a la calidad del suelo.
- ***farmToResidenceDistance***, distancia máxima que puede haber entre un *household* y su lugar de cultivo, expresada en hectómetros.
- ***maizeGiveToChild***, un valor entre 0 y 1, representa la fracción del total de reservas de alimento que se dota al nuevo *household* cuando uno da lugar a otro.
- ***householdMinInitialCorn***, mínima cantidad inicial de alimento que puede tener un *household* en el momento de su creación.
- ***householdMaxInitialCorn***, máxima cantidad inicial de alimento que puede tener un *household* en el momento de su creación.
- ***householdMinInitialAge***, valor mínimo que puede tomar la edad de un *household*, en el momento en que es creado en la inicialización de la

[14] A diferencia de [7], que toma *harvestVariance = spatialHarvestVariance*, aquí se ha distinguido entre ambas variables. [4] resulta un tanto ambiguo: en la explicación verbal distingue entre ambas, pero al tratar el ajuste de parámetros sólo hace referencia a la primera.

simulación. Cuando un *household* es creado a partir de otro, tiene siempre 0 años.

- ***householdMaxInitialAge***, valor máximo que puede tomar la edad de un *household*, en el momento en que es creado en la inicialización de la simulación.
- ***householdMinNutritionNeed***, valor mínimo que puede tomar el requerimiento nutricional anual para un *household*.
- ***householdMaxNutritionNeed***, valor máximo que puede tomar el requerimiento nutricional anual para un *household*.
- ***minFertilityAge***, valor mínimo que puede tomar la edad a partir de la cual un *household* puede dar lugar a otro.
- ***maxFertilityAge***, valor mínimo que puede tomar la edad a partir de la cual un *household* puede dar lugar a otro.

Estos últimos ocho valores (*householdMinInitialCorn, householdMaxInitialCorn, householdMinInitialAge, householdMaxInitialAge, householdMinNutritionNeed, householdMaxNutritionNeed, minFertilityAge* y *maxFertilityAge*) permiten dar heterogeneidad a los agentes.

Cada agente representa un hogar (***household***) formado por cinco miembros, aunque el número de miembros no tiene relevancia a la hora de hacer el modelo. Cada *household* posee las siguientes variables de estado:

- ***age***, de tipo entero.
- ***location***, una dupla (x,y) que representa la ubicación en el valle.
- ***farmLocation***, una dupla (x,y) que representa la ubicación del terreno de cultivo asociado a este *household*.
- ***nutritionNeed***, de tipo entero, representa las necesidades de nutrición que ha de cubrir anualmente.
- ***fertilityAge,*** un entero que representa la edad a partir de la cual un hogar puede dar lugar a otro (representa el hecho de que una hija, miembro del hogar, alcanza la edad de casarse y formar su propio hogar).
- ***agedCornStocks***, un vector que representa las reservas de alimento y antigüedad del mismo.

El entorno representa *Long House Valley y* está formado por un *grid* de 80 de ancho por 120 de alto. Cada celda del *grid* representa una parcela cuadrada de una hectárea, es decir de 100m de lado. Además, cada parcela tiene unos valores asociados, que cambian en cada

año de la ejecución de la simulación, y que representan información sobre agua, productividad del suelo, tipo de suelo (Figura 5.1) y otros datos.

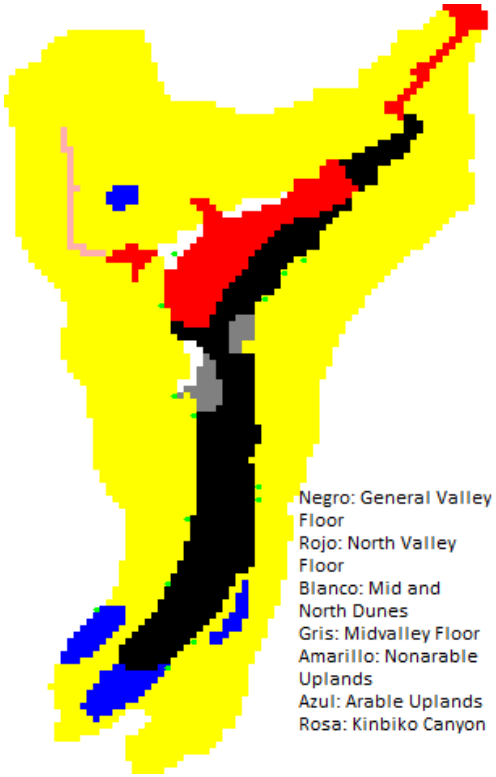

Figura 5.1. *Diferentes tipos de suelo en el valle.*

La simulación se extiende por un período que va desde el año inicial 800, hasta el año 1350, en que termina. Cada año está representado por una etapa o paso en la simulación.

5.2.3. VISIÓN GENERAL Y ORDEN DE LOS PROCESOS

El tiempo transcurre de forma discreta, por pasos o etapas. En cada uno de estos pasos, los siguientes eventos toman lugar en el orden en que aparecen, y la actualización de las variables se produce de forma asíncrona:

1. Se calcula el *baseYield* o rendimiento que produce cada una de las parcelas.
2. Se actualizan los datos relativos a las fuentes de agua de cada parcela.
3. Cada uno de los *households*, en un orden aleatorio, ejecuta la siguiente secuencia de acciones, *en el orden dado* a continuación:

a. Se obtiene la cosecha de ese año, y se actualizan las reservas de grano del hogar.
b. Se "consume" el alimento hasta cubrir las necesidades mínimas anuales (*nutritionNeed*).
c. Se estima el total de alimento que se obtendrá el año siguiente.
d. Si la anterior estimación no da para cubrir las necesidades mínimas anuales (*nutritionNeed*), entonces busca una nueva parcela para cultivar (*farmLocation*), y a continuación un nuevo asentamiento (*location*). En caso contrario, pasa al siguiente paso.
e. Se "reproduce", dando lugar, eventualmente, a un nuevo *household*.
f. Comprueba si ha de morir.
g. Aumenta la edad.

5.2.4. CONCEPTOS DE DISEÑO

Principios básicos.

El modelo sigue un diseño bastante simple en realidad, por lo que no se ha seguido ningún principio ni teoría, salvo el hecho de que, en esta implementación en concreto, trata de adaptarse a la arquitectura definida en este trabajo.

Emergencia.

Como se ha dicho anteriormente, el modelo es bastante simple en cuanto a sus reglas y a la "inteligencia" de los agentes. De hecho, en [7] se afirma que la precisión de los resultados obtenidos se debe, más que a los comportamientos de los agentes en sí, al hecho de que la capacidad de carga del sistema de por sí modela los resultados que se pueden esperar.

Adaptación.

Los agentes tratan, para sobrevivir, de cumplir con las necesidades de alimento anual que tienen. Para ello, si ven que no van a satisfacerlas, buscan nuevos lugares en los que establecer sus cultivos. Además, la productividad de estos lugares varía con el tiempo y condiciones ambientales, así pues los agentes responden a estos cambios. También han de adaptar su lugar de residencia en consecuencia, para que esté cerca del lugar donde cultivan y cerca de fuentes de agua, que también cambian a lo largo del tiempo.

Objetivos.

Conseguir, cada año, la cantidad de alimento necesaria para alcanzar el siguiente año. Esa cantidad se obtiene de la cosecha y las reservas de años anteriores. Si la suma de ambas alcanza la cantidad mínima anual necesaria, sobrevive.

Aprendizaje.

No hay ningún tipo de aprendizaje en el modelo.

Predicción.

Los agentes estiman la cosecha que obtendrán el año siguiente en base a la que han obtenido en el presente, suponiendo que será igual.

Sensibilidad.

Los agentes, a la hora de tomar decisiones sobre el lugar en el que establecerse tienen en cuenta, internamente, sus necesidades de alimento, su reserva de alimento, y su ubicación actual. Externamente, perciben a la hora de tomar estas decisiones factores como productividad del suelo, presencia de otros agentes, a través de asentamientos o granjas ubicadas en los lugares en que ellos buscan, y otra información acerca de las características de las parcelas como si son fuente de agua o si son susceptibles de sufrir inundaciones.

Interacción.

La única manera en que interactúan los agentes es, indirectamente, a través de su ocupación del suelo, bien sea para establecerse o para cultivar.

Aleatoriedad (Stochasticity).

En la inicialización, como se verá más adelante. Además, en la obtención de las cosechas anuales y en la ubicación inicial del lugar elegido para cultivar cuando un *household* se crea a partir de otro, aunque a continuación se le asigna una granja que cumpla los requerimientos, es tan sólo para tener con qué comparar en el algoritmo.

***Colectivos*.**

No hay organización de los agentes de más alto nivel que *household*.

***Observación*.**

Se recogen y se muestran datos acerca de la población, su ubicación y la de sus lugares de cultivo. Además, se muestran las poblaciones históricas, los rendimientos del suelo, las fuentes de agua y la capacidad de carga del sistema. Se muestran mediante la GUI.

5.2.5. INICIALIZACIÓN

Inicialmente, se crean 14 *households* (según estimaciones, son la población que habría en el año 800, inicio de la simulación). A cada una se le asigna un *stock* de alimento (maíz) aleatoriamente, según una distribución uniforme en el intervalo [*householdMinInitialCorn,*

householdMaxInitialCorn], para cada uno de los años que puede guardar alimento (los dos anteriores y el presente). Se le asigna también una edad en años aleatoriamente tomando un valor de la distribución uniforme en [*householdMinInitialAge, householdMaxInitialAge*], un valor aleatorio de la distribución uniforme en [*householdMinNutritionNeed, householdMaxNutritionNeed*] para el requerimiento anual de alimento, y un valor aleatorio de la distribución uniforme en [*minFertilityAge, maxFertilityAge*] para la variable *fertilityAge*.

A continuación, a cada uno de los *households* se le asigna una localización aleatoria y una localización para la granja aleatoria, y se le hace buscar una granja apropiada y un asentamiento próximo de manera similar a como se hace en el resto de la simulación.

El parámetro *quality* es un atributo que tiene cada parcela del terreno, se inicializa tomando un valor aleatorio de la distribución normal de media uno y desviación típica *spatialHarvestVariance*, N(1, *spatialHarvestVariance*). Obsérvese que *quality* debe ser un valor mayor o igual a cero, y la distribución de probabilidad anterior puede dar lugar a valores negativos (aunque con poca frecuencia). En ese caso, se le asignará a *quality* el valor 0.

Finalmente, se cargan los archivos de datos (ver sección siguiente) y se inicializan los atributos de las parcelas del *grid*.

Los valores[15] de las variables globales son

Variable	Valor inicial
harvestAdjustment	0.56
harvestVariance	0.1
deathAge	38
fertility	0.125
fertilityEndsAge	34
spatialHarvestVariance	0.4
farmToResidenceDistance	16
maizeGiveToChild	0.33
householdMinInitialCorn	2000
householdMaxInitialCorn	2400
householdMinInitialAge	0
householdMaxInitialAge	29
householdMinNutritionNeed	800
householdMaxNutritionNeed	800
minFertilityAge	16
maxFertilityAge	16

Tabla 5.1. *Valores iniciales para las variables globales.*

[15] Los cinco primero valores pueden modificarse de ejecución en ejecución, mediante la GUI. Los demás, en principio, no, aunque se puede acceder al código y modificarlos si se desea. Los valores son los obtenidos en la calibración y son los que dan un mejor ajuste (Ver Capítulo 6).

5.2.6. DATOS DE ENTRADA EXTERNOS (*INPUT DATA*)

Para obtener los datos sobre el entorno que son necesarios llevar a cabo la simulación, se han de cargar[16] una serie de archivos (que están disponibles en la replicación de Janssen en [103], y en la versión reimplementada para este trabajo en [104]). Algunos valores importantes que se han de obtener de esos datos son los siguientes:

- ***watersource***, de tipo *boolean* que representa si la parcela tiene una fuente de agua.
- ***zone***, que representa el tipo de terreno sobre el que está asentada la parcela, y que puede ser uno de los siguientes: *General*, *North*, *NorthDunes*, *Mid*, *MidDunes*, *Natural*, *Upland*, *Kinbiko*, *Empty*.
- ***apdsi***, que es una medida de variabilidad climática, que se usará para calcular el rendimiento de una parcela.
- ***hydro***, una medida que representa flujos de agua o inundaciones que puedan atravesar la parcela.

Los archivos de datos son los siguientes:

- *Map.txt* contiene información sobre a qué tipo de terreno pertenece cada parcela, el valor *zone* de la anterior descripción.
- *adjustedPDSI.txt* define, para cada categoría de *zone*, el valor del *adjusted Palmer Drought Severity* [105], un indicador de las condiciones de humedad y sequía para actividades agrícolas. Según el valor de este índice, se estima un rendimiento agrícola —la variable *yield* en el modelo— para el terreno en cuestión.
- *environment.txt*, información para cada parcela sobre en qué épocas es una zona inundada (valor *hydro* >= 0).
- *settlement.txt* tiene información acerca de los yacimientos históricos, y estimaciones de los asentamientos, su tamaño, dimensión espacial y temporal.
- *water.txt* es la información acerca de qué puntos son fuentes de agua, y cuándo, pues van variando con el tiempo.

[16] La manera en que se han de cargar estos archivos es más fácil de comprender si se consulta el código que los carga. Éste puede verse (y reutilizarse) en el Apéndice A.

5.2.7. SUBMODELOS

Los procesos que se han visto en el apartado 3, la visión general, estaban numerados como 1, 2, 3.a, 3.b, etc. Así serán denominados a continuación, para evitar confusiones.

1. Cálculo del baseYield de cada parcela

Cada parcela posee un atributo *yield*, que da una medida del rendimiento en cada momento, y que depende exclusivamente de los datos de entrada que se han visto en el apartado anterior. En concreto, dependen del tipo de terreno, de la época del año y del índice aPDSI. Para introducir algo de variabilidad local, se definía el atributo *quality*, que tomaba valor de manera aleatoria. Además, se tenía el atributo global *harvestAdjustment*. Con estos datos, se calcula

$$baseYield = yield * quality * harvestAdjustment$$

2. Actualizar fuentes de agua

En función del año y de otros parámetros (la zona y otros contenidos en *wáter.txt*) se define qué parcelas son *waterSource* ese año, y cuáles no. La manera concreta se puede consultar en el apéndice A.

3.a. Obtener la cosecha y actualizar las reservas de grano

La cosecha del año en curso se define como

$$lastHarvest = baseYield * \big(1 + N(0, harvestVariance)\big)$$

Donde *baseYield* corresponde al de la parcela en donde el *household* en cuestión tiene la granja. No obstante, lo anterior puede resultar en un valor negativo, sobre todo si *harvestVariance* es relativamente grande. En tal caso, se tomará como cero.

Las reservas de alimento se actualizan así

```
agedCornStocks[2] = agedCornStocks[1];
agedCornStocks[1] = agedCornStocks[0];
agedCornStocks[0] = lastHarvest;
```

donde *agedCornStocks[i]* representa el *stock* de alimentos de hace *i* años.

3.b. Consumir el alimento

Se empieza consumiendo el alimento almacenado con más antigüedad. Si con éste no es suficiente para cubrir las necesidades, se recurre al siguiente más antiguo. Si aún así no es suficiente, se recurre al de la última cosecha (en caso de que se almacene tan sólo el de dos años de antigüedad, si no se seguiría , consumiendo siempre primero el más antiguo). El código que lo implementa es:

```
nutritionNeedRemaining = nutritionNeed
ys = yearsOfStock
      while (ys > -1) {
            if (agedCornStocks[ys] >= nutritionNeedRemaining) {
                  agedCornStocks[ys] -= nutritionNeedRemaining;
                  nutritionNeedRemaining = 0;
            } else {
                  nutritionNeedRemaining -= agedCornStocks[ys];
                  agedCornStocks = 0;
            }
            ys--;
      }
}
```

3.c. Estimar el la cosecha del año siguiente

Simplemente, suponiendo que se obtendrá una igual a la que se ha obtenido este año:

$$estimateHarvest = agedCornStocks[1] + agedCornStocks[2] + lastHarvest$$

3.d. Buscar nuevo emplazamiento

Si la cosecha estimada para el año siguiente (*estimateHarvest*) es menor que las necesidades nutricionales (*nutritionNeed*) entonces busca una nueva parcela para cultivar siguiendo el siguiente algoritmo:

En primer lugar, se identifican todas parcelas que cumplen:

1. No hay ningún asentamiento.
2. No hay ninguna granja.
3. Tiene un *baseYield* >= *householdMinNutritionNeed.*
4. Está a una distancia máxima de 1.6km de una fuente de agua.

Si hay múltiples parcelas que cumplen las cuatro condiciones anteriores, seleccionar aquella más próxima al lugar de asentamiento actual[17]. En caso de que no haya ninguna granja disponible, muere.

[17] En [7] se escoge la que está más próxima al lugar actual en que se cultiva (granja), mientras que en [4] se elige la que está más próxima al lugar de asentamiento. Aquí se ha optado por esta opción porque parece más

En caso de que cambie de granja, ha de buscar un nuevo asentamiento que esté próximo a la granja. Para ello, se identifican todas las parcelas que cumplan:

1. No hay ninguna granja en ella (aunque puede haber otros *households* asentados, se permiten asentamientos de varios hogares). Además, no puede ser una parcela con valor de la variable *hydro* mayor que cero[18]. La variable *hydro* toma valor y se va ajustando en cada iteración, al igual que *yield*.
2. Debe estar a una distancia máxima de 1.6km de la nueva granja elegida.
3. Debe estar en una zona menos productiva que la nueva granja elegida.

Si hay múltiples sitios que satisfacen estos criterios, se selecciona el más cercano a una fuente de agua.

Si ningún sitio cumple los tres criterios, se tienen en cuenta los dos primeros.

En caso de que ningún sitio cumpla los dos primeros tampoco, se tiene en cuenta nada más que el primero.

Siempre habrá un lugar que cumpla esto al menos: su ubicación actual.

3.e. Reproducción: dar lugar a un nuevo hogar

La manera en que se perpetúa la población viene dada por la capacidad de un *household* de dar lugar a otro, representando que una hija se emancipa del hogar familiar y forma su propio hogar. Esto se producirá si se dan las siguientes condiciones, todas a la vez:

1. *age > fertilityAge*
2. *age < fertilityEndsAge*
3. *random < fertility*, siendo *random* un número aleatorio de la distribución uniforme en [0,1].
4. Hay granjas disponibles: una granja disponible es una que cumpla las cuatro condiciones que se veían en el apartado anterior para buscar una granja.

En este caso, se crea un nuevo *household*, al que se le da una proporción de las reservas de alimento del padre. Esta proporción viene determinada por el parámetro *maizeGiveToChild.* Se le ubica, en principio, en el mismo asentamiento que al padre. A continuación, se le asigna una granja aleatoria, y se procede a buscar una granja apropiada y un asentamiento próximo, de la misma manera en que se ha descrito en la sección anterior.

---

coherente, aunque se han probado las dos opciones, sin haber obtenido diferencias apreciables en los resultados de la simulación.

[18] En [7] no se nombra explícitamente la restricción de *hydro*, aunque en el código del modelo en NetLogo sí aparece reflejado, así como en [6].

3.f. Morir

Un *household* morirá si se dan una o ambas de las siguientes dos condiciones:

1. *nutritionNeedRemaining* > 0, es decir no cumple sus requisitos de alimentación mínimos.
2. *age* > *deathAge*, ha superado la edad máxima que puede vivir.

3.g. Aumentar la edad

El agente incrementa su edad (atributo *age*) en un año, que es la unidad de tiempo en el modelo.

5.3. IMPLEMENTACIÓN DEL MODELO DESCRITO

El modelo de Artificial Anasazi ha sido escogido por ser un modelo muy representativo y citado, además de ser considerado un ejemplo de éxito de ABM. Además, se tenía acceso tanto al código fuente del mismo, en la versión de Janssen implementada en NetLogo, como a cierta cantidad de documentación, más o menos precisa, que ha permitido el reimplementarlo, no sin algún que otro problema.

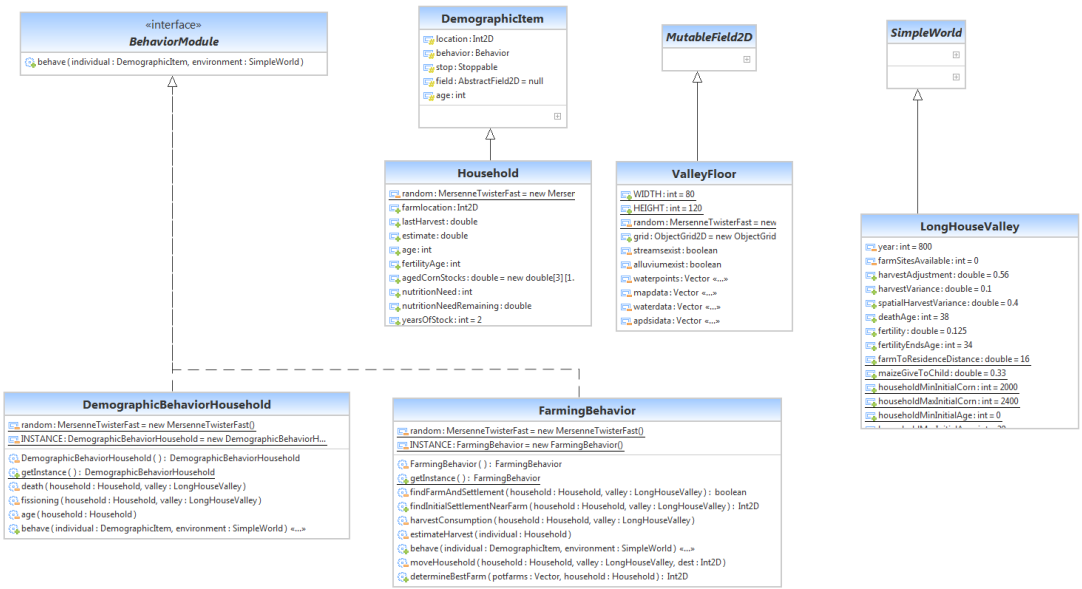

Figura 5.2. *Principales clases del modelo implementado y su relación con las de la arquitectura definida.*

El principal ha sido la gran diferencia conceptual entre un lenguaje como NetLogo y Java. El primero, diseñado específicamente para producir modelos de manera sencilla y directa, cuenta con una salida gráfica que se produce de manera prácticamente inmediata, además de un tratamiento del *grid* sobre el que se sitúan los agentes que da muchas facilidades para el tratamiento tanto las celdas del *grid*, que pueden ser personalizadas mediante atributos y comportamientos de manera sencilla, como de sus relaciones e interactuación con los agentes.

Sin embargo, una vez aclarados ciertos conceptos sobre NetLogo, el proceso de adaptarlo a la arquitectura propuesta en este trabajo no ha resultado demasiado complejo.

Se ha definido, en primer lugar, una clase *Household*, que representa a los agentes, los *households* descritos anteriormente y que extiende a la clase *DemographicItem*. No ha hecho falta emplear la clase *Person* ya que no existe interacción directa entre los agentes.

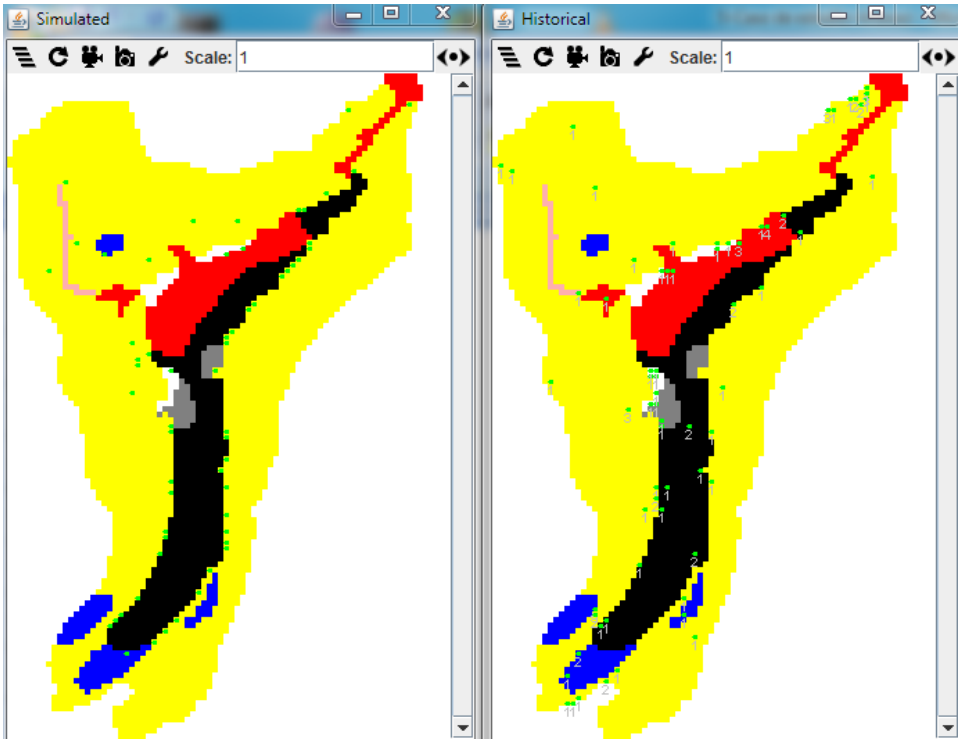

Figura 5.3. *Interfaz gráfica. A la izquierda, modelo simulado, con los households en verde. A la derecha, datos históricos.*

Por otra parte, como se describe en el modelo, el único medio de interacción entre los diferentes agentes es el propio entorno. Éste ha de ir actualizándose en cada iteración con los datos sobre el medio ambiente. Por ello se ha extendido la clase *MutableField2D*, construyendo la clase *ValleyFloor*. Otros entes necesarios para la simulación la información sobre los asentamientos históricos o las propias parcelas del valle, que cuentan con atributos

propios que van cambiando durante la simulación, han sido construidos como clases internas de *ValleyFloor*.

Este entorno ha sido situado, junto al resto de componentes, en la clase contenedor *LongHouseValley*, que extiende a *SimpleWorld*. Además, contiene otro *grid* en donde se han situado a los agentes, para tener un acceso más rápido y sencillo a ellos que a través de *ValleyFloor*.

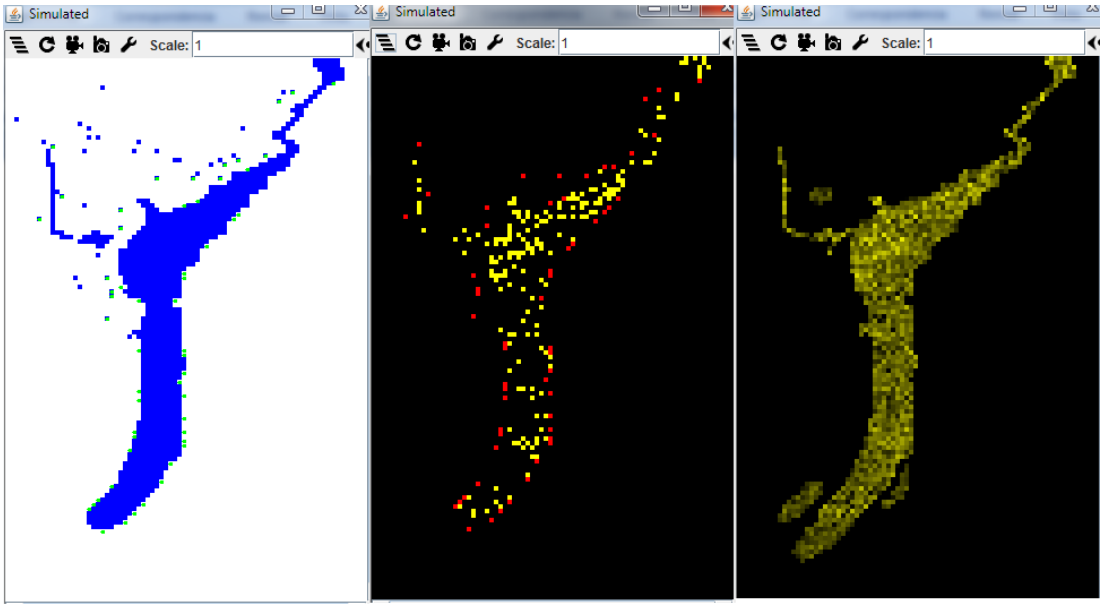

Figura 5.4. *Diferentes vistas sobre el entorno. De izquierda a derecha: fuentes de agua, ocupación del suelo (rojo asentamientos, amarillo granjas) y rendimiento del suelo (más claro, más rendimiento).*

Por último, se han creado dos tipos de *BehaviorModule* distintos. En uno, *FarmingBehavior*, se han incluido los comportamientos relativos a la búsqueda de asentamiento y de granja, junto a la recolección de la cosecha y el consumo de la misma. En el otro, *DemographicBehaviorHousehold*, se han incluido los comportamientos relativos a cuando se "emancipaba" una hija, es decir, cuando un *household* daba lugar a otro, como se ha venido diciendo en la descripción, además de la muerte.

No ha sido necesario incluir ninguno de los módulos demográficos, pues los únicos procesos demográficos que se modelaban externamente —o de arriba abajo, como se ha venido diciendo— eran la probabilidad de que se formase un nuevo *household* —lo que vendría a ser la fertilidad—, que era un valor fijo para cada ejecución del modelo, y la edad de muerte de y de fecundidad de los *households*, también fijos durante cada ejecución.

Un *applet* Java con el modelo está disponible para ser probado en https://sites.google.com/site/anasazimasonrep/.

6. PRUEBAS REALIZADAS

A lo largo de este capítulo, se consideran constantes todas las variables que no aparezcan nombradas explícitamente en esta sección, con los valores que se les dio en la descripción ODD del modelo. Por tanto, se tendrán en cuenta diferentes combinaciones de las siguientes seis variables: *harvestAdjustment*, *harvestVariance*[19], *deathAge*, *fertility*, *fertilityEndsAge*, *spatialHarvestVariance*.

6.1. RESULTADOS PREVIOS

El primer modelo realizado fue el de Axtell et al. [4]. En una primera versión, los autores prueban el sistema con los siguientes parámetros

harvestAdjustment	1
harvestVariance	0.1
deathAge	30
fertility	0.125
fertilityEndsAge	30
spatialHarvestVariance	0.1

Tabla 6.1. *Valores "base" empleados por Axtell antes de ajustar el sistema.*

Janssen, en [7], prueba su versión del modelo con los mismos valores, obteniendo el resultado que puede verse en la Figura 6.1, y que es el mismo resultado que obtenía Axtell.

Ambos reflejan, no obstante, cifras de población que llegan entre los años 1025-1275 a más de mil *households*, cuando los datos históricos reflejan que rondaban los doscientos.

[19] Al hablar de los resultados de estos autores, no se hará distinción entre *harvestVariance* y *spatialHarvestVariance*, pues como se comentó anteriormente las emplean como una única variable. En el contexto de este trabajo, no obstante, sí se ha distinguido entre ambas.

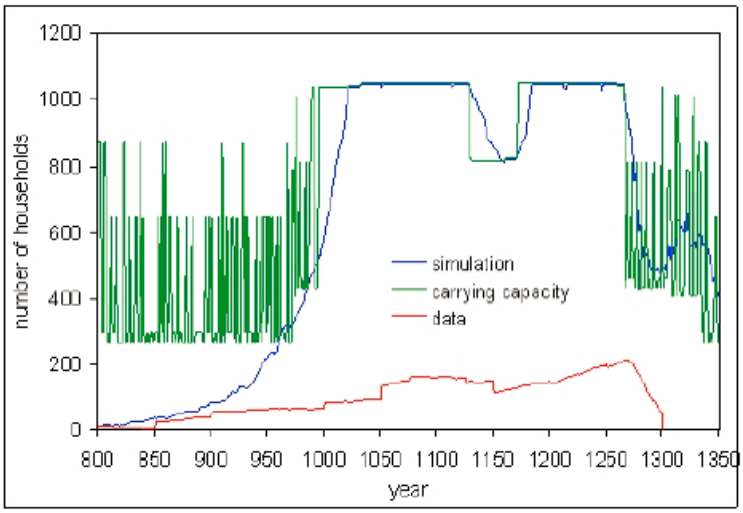

Figura 6.1. *Resultados obtenidos por [7], para los parámetros de la tabla uno.*

Axtell procede a calibrar los parámetros, llegando a los que se reflejan en la Tabla 6.2. El modelo de Axtell, no obstante, permite cierta heterogeneidad para los *households* en los atributos *fertility*, *deathAge* y *fertilityEndsAge*, por lo que los representa como intervalos, tomando los *households* valores según la distribución uniforme en los intervalos. La mejor ejecución de cien pruebas que obtiene Janssen con estos parámetros puede verse en la Figura 6.2

harvestAdjustment	0.6
harvestVariance	0.4
deathAge	(30-36)
fertility	0.125
fertilityEndsAge	(30-32)
spatialHarvestVariance	0.4

Tabla 6.2. *Datos calibrados por Axtell.*

Finalmente, Janssen procede a calibrar el modelo él mismo. Para ello, deja de lado la heterogeneidad en los parámetros *fertility*, *deathAge* y *fertilityEndsAge* por no obtener diferencias significativas y simplificar así el enorme espacio paramétrico. Considera pues valores de *deathAge* y *fertilityEndsAge* de 26, 28, 30, 32, 34, 36, 38 y 40, siempre siendo la edad de muerte mayor o igual a la edad de fin de la fertilidad. El parámetro *fertility* lo considera entre 0.095 y 0.185 a intervalos de 0.015. El parámetro *harvestAdjustment* lo explora a intervalos de 0.02 entre los valores 0.54 y 0.7. Por último, *harvestVariance* y *spatialHarvestVariance* los mueve a la par entre 0 y 0.7 a intervalos de 0.1.

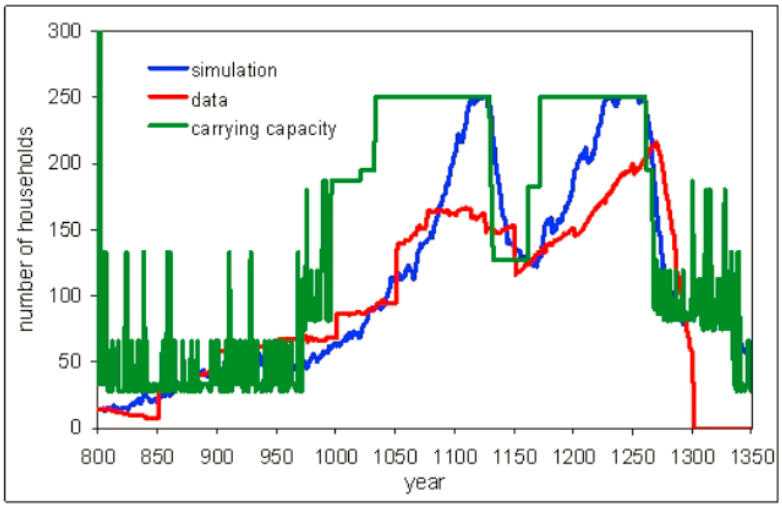

Figura 6.2. *Mejor ejecución de Janssen con los parámetros calibrados por Axtell.*

Con estas posibilidades, obtiene un ajuste óptimo que se puede ver en la Tabla 6.3. Para valorar la bondad de un ajuste, aplica p-normas vectoriales a los vectores diferencia de población histórica anual, y población simulada anual. En cada uno de estos vectores, la primera componente representa la población en el año inicial, la segunda la población del segundo año, y así sucesivamente.

harvestAdjustment	0.56
harvestVariance	0.4
deathAge	38
fertility	0.155
fertilityEndsAge	34
spatialHarvestVariance	0.4

Tabla 6.3. *Valores ajustados obtenidos por Janssen. Obtiene los mismos parámetros de ajuste óptimos para las dos normas.*

La 1-norma de un vector $x = (x_1, x_2, \dots, x_n)$ se define como

$$\|x\|_1 = |x_1| + |x_2| + \cdots |x_n|$$

La 2-norma, que no es sino la norma euclídea, se define como

$$\|x\|_2 = \sqrt{(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)}$$

En todos los casos vistos se observa que hay un valor —representado en verde en las gráficas— que limita el crecimiento de la población. Este es la capacidad de carga del sistema. En este modelo en particular, se define como el número de parcelas que tienen

baseYield >= nutritionNeed, es decir, el número de parcelas que, teóricamente, darían la cantidad de cosecha suficiente para mantener a un *household*. Se dice teóricamente porque luego en la práctica se aplica una variabilidad sobre *baseYield* que depende de *spatialHarvestVariance*.

La capacidad de carga es por lo tanto una cota superior de la población. La razón por la que se observa que en determinados puntos la población pueda estar por encima de esta capacidad de carga tiene que ver con que los *households* mantienen unas reservas de grano. Esto provoca que, cuando la capacidad de carga disminuye, la población tarde en responder a esta disminución, pues tiene el margen que le dan las reservas.

Además, la capacidad de carga, por la manera en que está definida, depende únicamente de los parámetros *spatialHarvestVariance* y *harvestAdjustment*, ya que *baseYield* se definía como

$$baseYield = yield * quality * harvestAdjustment$$

Y *quality* era un valor de la distribución N(1, *spatialHarvestVariance*).

Janssen calcula también los valores de estos parámetros que dan el ajuste óptimo de la capacidad de carga, obteniendo los mismos valores que para el ajuste de la población simulada. Además comprueba que *deathAge*, *fertilityEndsAge* y *fertility* —es decir, los parámetros puramente demográficos— no presentan demasiada influencia en el ajuste del modelo a los datos históricos, siempre y cuando aseguren la supervivencia de la población, lo que ocurre a partir de unos valores mínimos. Sin embargo tanto *spatialHarvestVariance* como *harvestAdjustment* son los parámetros que más peso tienen, y son precisamente de los que depende la capacidad de carga.

El valor óptimo de estos dos parámetros hace por tanto que ajuste bien tanto la simulación como la capacidad de carga. Es decir, el modelo de la población se ajusta a la capacidad de carga. Y por tanto siempre que la capacidad de carga se ajuste a los datos históricos, lo mismo hará la población simulada —siempre y cuando los valores del resto de variables sean "aceptables" como para permitir que la población crezca, como se comentaba anteriormente.

Concluye Janssen afirmando que el análisis realizado apoya y confirma las conclusiones dadas por [4]: que los factores ambientales por sí solos no bastan para explicar el repentino abandono de la zona por parte de la población a partir del año 1300. Ya que como se ve, la capacidad de carga a partir de esa fecha permite el sostenimiento de población.

El mérito del modelo pues, está más en este buen ajuste de la capacidad de carga que en el modelo de los agentes en sí, ya que estos ofrecen poca dinámica propia que contribuya a la dinámica poblacional observada. Actuando más como "suavizador" de la capacidad de carga que otra cosa.

6.2. RESULTADOS OBTENIDOS

6.2.1. PRUEBAS CON LOS DATOS CALIBRADOS POR LOS OTROS AUTORES

En primer lugar se procedió a realizar una serie de simulaciones con los datos obtenidos por los autores de [4] y [7].

Para los datos de la Tabla 6.1, datos iniciales previos a realizar el ajuste de los mismos, se obtuvo el resultado que muestra la Figura 6.3. En esta figura se muestra una ejecución tipo con estos parámetros, resultando la mayoría de ejecuciones muy similares, al menos gráficamente. Como se observa, el resultado es prácticamente el mismo que obtiene [7] (Figura 6.1): la población crece de forma exponencial hasta alcanzar la capacidad de carga —más de mil *households*—, y a partir de ahí se ajusta a lo que marca esta capacidad, aunque suavizada por el efecto de las reservas de grano. La razón de que sea tan alta la capacidad de carga es el valor también grande de *harvestAdjustment*, que vale 1.

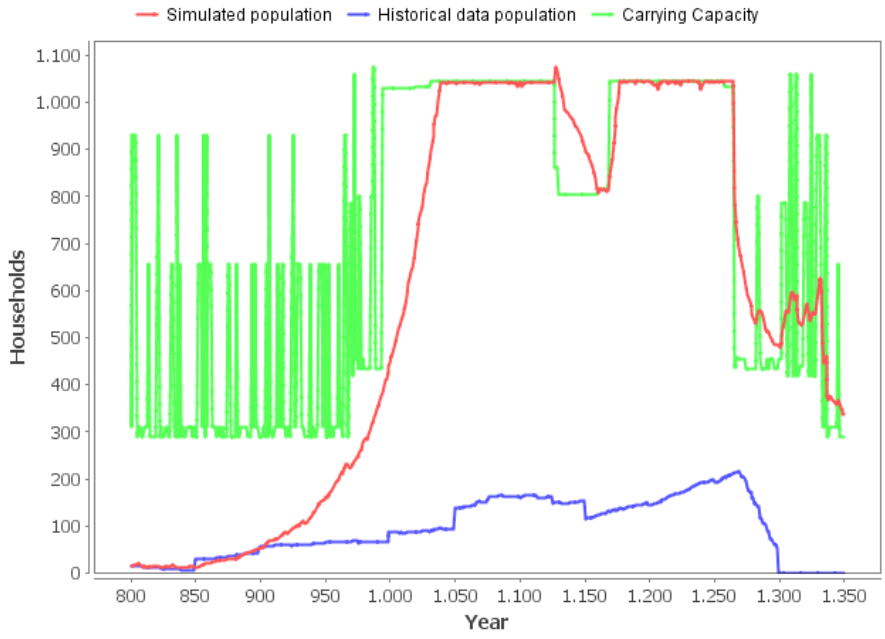

Figura 6.3. *Resultados obtenidos por el autor, para los parámetros de la tabla uno. En azul, datos históricos, en rojo los simulados, en verde capacidad de carga del sistema.*

Al realizar simulaciones con los datos ajustados por Axtell (Tabla 6.2) y los ajustados por Janssen (Tabla 6.3) la población se extinguía en torno al año 900 en la mayor parte de las ejecuciones y, en las que no lo hacía, mantenía un nivel muy por debajo de los datos históricos. Una ejecución típica (Figura 6.4) con los valores de la Tabla 6.2 y tomando *deathAge=33* y *fertilityEndsAge=31* —valores intermedios de los intervalos en que está definido—.

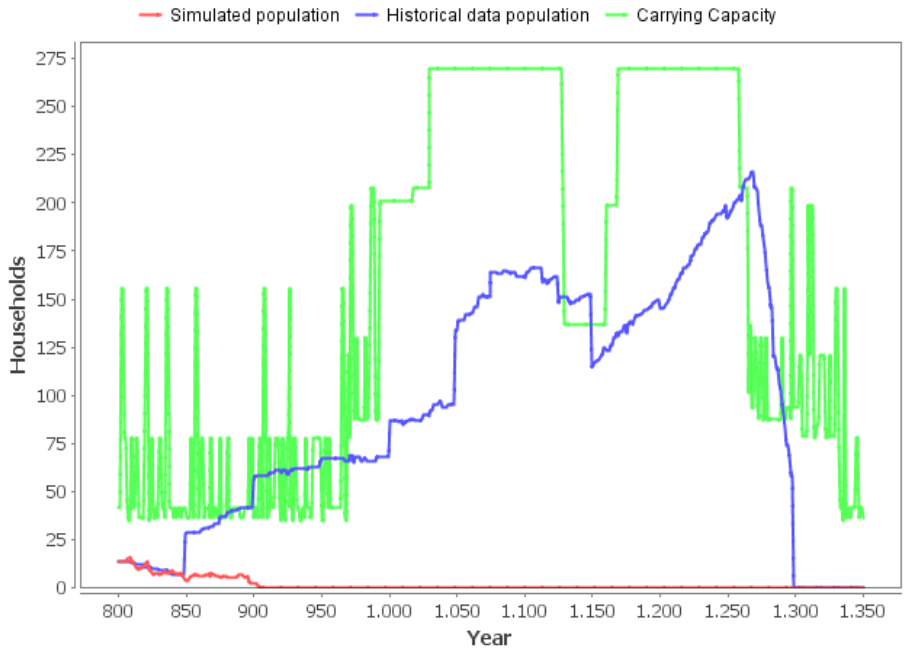

Figura 6.4. *Ejecución típica con los valores de la Tabla dos.*

Para detectar las causas de esta anomalía con respecto a los resultados esperados se procedió a variar los parámetros de uno en uno para investigar el efecto que esto producía.

- ***deathAge***: Aumentarlo por sí sólo no produjo ningún efecto en los resultados.
- ***fertilityEndsAge***: Al aumentar el valor de este parámetro hasta un valor de 50 conjuntamente con el anterior (ya que tiene que ser inferior, un *household* que ha muerto no puede dar lugar a otro) se observó cierta mejoría. A partir del año mil si había una población superior a 15 ó 20 la población se disparaba hasta alcanzar la capacidad de carga. No obstante, esto no producía aproximaciones buenas a los datos históricos. En la mayor parte de las ejecuciones, no obstante, la población se extinguía. Al probar con valores más altos (60) se conseguían resultados bastante aceptables en lo que respecta a la supervivencia de la población, aunque no a su ajuste a los datos. En cualquier caso, son valores demasiado altos que no resultan realistas.
- ***harvestAdjustment***: Al aumentar o disminuir este parámetro, se está modificando la capacidad de carga del sistema. Disminuirlo no resultó factible, pues da lugar a menos recursos alimenticios para los agentes, y la población se extinguía igualmente. Al aumentarlo a 0.7, se mejoraba mucho el resultado: aproximadamente en la mitad de los casos, la población

lograba sobrevivir. No obstante, el sistema presentaba poca similaridad estructural[20]: en los casos en que lograba sobrevivir pues unas veces se mantenía muy por debajo de los datos históricos, mientras que otras alcanzaba la capacidad de carga, llegando a poblaciones el doble de grandes que la histórica. Aumentar más este parámetro conseguía el mismo efecto pero más exagerado, y también mayor supervivencia de la población. Con un valor de 1, la población dejó de extinguirse.

- ***fertility***: Aumentando el valor hasta 0.2 se llegaron a obtener buenas aproximaciones a los datos históricos. No obstante, el comportamiento seguía siendo demasiado irregular: demasiadas ejecuciones terminaban con la extinción de la población. Con 0.3 se obtuvieron mayores tasas de supervivencia de la población. No obstante, el crecimiento ya resultaba demasiado rápido y poco realista. Valores mayores no pueden ser considerados factibles, pero aun así se exageraba este comportamiento: la gráfica de la población ajustaba a la forma de la de la capacidad de carga (Figura 6.5).

- ***spatialHarvestVariance***: Es el otro factor que determina la capacidad de carga. Con valores de 0.3 y 0.5 no se observó ninguna diferencia: la población seguía extinguiéndose. Valores más pequeños que 0.3 dieron lugar a una capacidad de carga ínfima. Con un valor de 0.2 la población se extinguía siempre antes de 40 años. Valores más grandes producían capacidades de carga demasiado altas. Con 0.6 el ajuste con respecto a la población histórica era muy bueno hasta el año 1050-1060. En ese momento la población se disparaba hasta niveles de la capacidad de carga, que se situaba en torno al doble que los datos. A partir de este valor la población dejó de extinguirse prácticamente, pero el ajuste con respecto a los datos históricos era muy malo, seguía observándose poca similaridad estructural entre ejecuciones. Valores más altos, incluso de 1, no presentaron comportamientos demasiado distintos. La capacidad de carga no aumentó más allá de 400-450, según ejecuciones.

- ***harvestVariance***: Con valores mayores se obtuvieron resultados peores. Para valores más pequeños se comenzó a observar mejoría. Con 0.3 la población se extinguía en pocas de las ejecuciones. El ajuste con respecto a los datos históricos también mejoró. Con 0.2 mejoró un poco más y ya prácticamente nunca se extinguía la población. La gráfica se ajustaba mucho a la de la capacidad de carga.

[20] *Similaridad estructural*, es un concepto que designa, en sistemas no deterministas, la diferencia que se produce entre diferentes ejecuciones del mismo. A mayor diferencia entre las mismas, menor similaridad estructural.

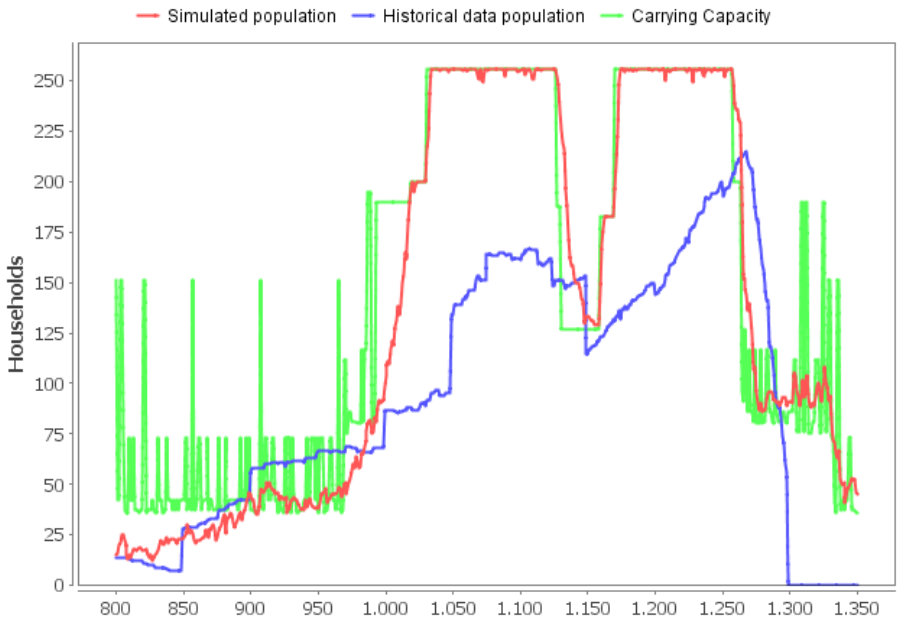

***Figura 6.5**. Ejecución con fertility=0.5.*

De los resultados que se obtuvieron se deduce que:

Aumentando parámetros que aumentan la natalidad, como *fertilityEndsAge* o *fertility* se consigue supervivencia, pero aun poca similaridad estructural. Esto indica que no es un problema de falta de nacimientos.

Por otro lado, aumentando la capacidad de carga mediante *harvestVariance* se conseguía que la población dejara de extinguirse para valores muy altos de esta. Esto indica que la causa por la que se extinguía la población era por no disponer de parcelas que dieran suficiente cosecha. Aumentando *spatialHarvestVariance* se conseguía el mismo efecto, pero con poblaciones más contenidas: esto se debía a que al aumentar *spatialHarvestVariance* se hacía que hubiera muchas parcelas con un rendimiento exageradamente alto, aunque también muchas con un rendimiento exageradamente malo. Mientras que aumentando *harvestVariance* se conseguía que todas las parcelas dieran un rendimiento mayor.

Si se disminuía *harvestVariance* se lograba que el sistema se comportara como se esperaba —como se comportaba el implementado en NetLogo por Janssen—. Población que se ajustaba bien a los datos históricos y que, aunque presentaba cierta variación de ejecución en ejecución, no eran tan acusadas, y siempre se conseguía que sobreviviera. La causa es clara: con este parámetro demasiado alto, se producía una gran diferencia entre cosechas, pudiendo obtenerse cosechas muy pequeñas, aunque también muy grandes. Esto producía que, a poblaciones pequeñas sobre todo, al obtener dos cosechas consecutivas bajas, lo cual es probable que suceda, no se consiguiera el suficiente alimento para sobrevivir.

Finalmente, al aumentar *harvestVariance* por encima de 0.5 en la implementación de Janssen se observó el mismo efecto: poblaciones que se extinguían. No obstante, aunque Janssen emplea el mismo valor para *spatialHarvestVariance* y *harvestVariance*, la causa no es esta, pues al aumentar el primero se conseguía aumentar la población y su supervivencia, así que la causa tiene que estar en un valor de *harvestVariance* demasiado alto. La razón de esta diferencia en el comportamiento entre el modelo de Janssen — y el de Axtell et al.— y el del presente trabajo parece estar pues en una diferente sensibilidad al parámetro *harvestVariance*.

Una posible causa de esto podría ser el emplear diferentes plataformas, cada una con su propio generador de números aleatorios, aunque no parece demasiado probable. No habría que descartar por tanto un error en la implementación, aunque no parece ser la causa: se ha revisado el código en profundidad y para el resto de casos ambos modelos se comportan de forma similar. En cualquier caso, es un problema a investigar más en profundidad en un futuro.

6.2.2. AJUSTE DE LOS PARÁMETROS

Debido al diferente comportamiento observado en el apartado anterior, se decidió proceder a buscar el mejor ajuste de parámetros para el modelo replicado en este trabajo.

La metodología empleada consistió en realizar análisis de sensibilidad del espacio paramétrico y comparar los resultados obtenidos en la simulación con los datos históricos mediante las dos normas empleadas en [7] y [4], definidas anteriormente.

El primer paso fue reducir el tamaño del espacio paramétrico a uno que fuera computacionalmente tratable. Para ello se tomaron las siguientes consideraciones sobre cada uno de los parámetros.

- ***deathAge*** y ***fertilityEndsAge***: Se consideraron a la vez, puesto que el segundo no puede tomar valores más altos que el primero. A las observaciones realizadas por Janssen sobre la limitada influencia de estos parámetros en el ajuste, se suman las realizadas aquí en el punto anterior. Por tanto, se decidió que estos parámetros pudieran tomar los valores 36 y 38 el primero, y 32 y 34 el segundo. Esto resultó en cuatro combinaciones posibles.
- ***fertility***: Vistos los resultados que obtenían los otros autores, y por las mismas razones que en el caso anterior, se decidió que pudieran tomar valores entre 0.125 y 0.155, a intervalos de 0.015. Esto resultó en tres combinaciones.

- ***harvestAdjustment***: Dada la influencia de este parámetro en la capacidad de carga, se decidió no situarlo en menos de 0.54, ni más allá de 0.6, ni que son los valores para los que Janssen y Axtell obtienen mejores resultados. Valores mayores o menores dan lugar a capacidades de carga demasiado altas o bajas que no se ajustan a lo observado. Se tomaron intervalos de 0.02, lo que resultó en un total de cuatro combinaciones.
- ***spatialHarvestVariance***: Valores menores que 0.3 daban capacidades de carga demasiado bajas, mientras que sucedía al contrario con valores mayores que 0.5, así pues se tomó este intervalo, evaluando también el valor 0.4. Esto resultó en tres combinaciones.
- ***harvestVariance***: Con las pruebas previas que se había realizado, se observaba que, cuanto más bajo el valor, tanto mejores parecían los resultados. Para comprobar esta impresión, se tomaron valores entre 0 y 0.2 a intervalos de 0.05. Valores mayores no se consideraron, pues a partir de 0.3 la población se extinguía en ocasiones. Esto dio en un total de cinco posibilidades.

Con las decisiones anteriores, se reduce el espacio paramétrico a un total de 4*3*4*3*5=720 combinaciones posibles. Para cada una de estas, se realizaron 15 simulaciones[21] y se calculó el valor medio de cada una de las normas para esas 15 ejecuciones. Finalmente, se escogieron aquellas combinaciones de parámetros que dieron menor valor medio para cada una de las dos normas empleadas. Los resultados obtenidos se muestran en la Tabla 6.4.

Parámetro	**1-norma**	**2-norma**
harvestAdjustment	0.56	0.56
harvestVariance	0.05	0.05
deathAge	36	36
fertility	0.125	0.125
fertilityEndsAge	34	34
spatialHarvestVariance	0.4	0.4
Valor medio de la norma	16118.87	866.90

Tabla 6.4. *Valores obtenidos en el ajuste de los parámetros del modelo. Se ha tomado la mejor de las medias de 15 ejecuciones para cada combinación de parámetros.*

Además, se buscó también el mejor ajuste de parámetros para la capacidad de carga, comparándola con los datos históricos de similar manera. En este caso, al haber sólo dos parámetros implicados, se exploró un mayor conjunto de valores para ambos:

[21] Se realizaron 15 simulaciones al ser el número de ejecuciones que realizan tanto Axtell et al. como Janssen.

harvestAdjustment se evaluó entre 0.2 y 0.7, a intervalos de 0.02, mientras que para *spatialHarvestVariance* se contemplaron valores entre 0.1 y 0.7 con intervalos de 0.1.

Una serie inicial de pruebas, con 15 ejecuciones para cada combinación de parámetros arrojó resultados un tanto chocantes: 0.44 como valor óptimo para *harvestAdjustment* y 0.7 para *spatialHarvestVariance* con ambas normas. Esto hizo que se repitieran, obteniendo (para *harvestAdjustment* y *spatialHarvestVariance*, respectivamente) en sucesivas pruebas los valores 0.5 y 0.5 para ambas normas, 0.54 y 0.4 para la 1-norma y 0.5 y 0.5 para la 2-norma, o 0.48 y 0.6 para la 1-norma y 0.44 y 0.7.

Esta disparidad en los valores indica que hay una variabilidad bastante grande en los valores que toma la capacidad de carga. Esta variabilidad es consecuencia de la inicialización del parámetro *quality* de las parcelas, según una distribución normal de media 1 y desviación típica *spatialHarvestVariance*. Esto parece apoyar la idea de que el sistema implementado aquí es más sensible a estos parámetros que actúan como varianza de la distribución normal, como se apuntaba al hablar del parámetro *harvestVariance*. Y parece apoyar el hecho de que no se tratara de un error del autor en la implementación, sino que se trate de la plataforma en que se ha implementado, ya que en [7] no sucede esto, realizando también 15 pruebas por cada combinación de parámetros.

Por tanto se decidió repetir este ajuste de los parámetros que afectan a la capacidad de carga, pero tomando 100 ejecuciones para cada par de posibles valores, número que debería ser suficiente para suavizar este efecto. En la Tabla 6.5 se muestran los resultados.

Parámetro	**1-norma**	**2-norma**
harvestAdjustment	0.44	0.44
spatialHarvestVariance	0.7	0.7
Valor medio de la norma	18100.43	1033.83

Tabla 6.5. *Valores obtenidos al realizar el ajuste de la capacidad de carga, empleando la media de 100 ejecuciones para cada par de valores.*

Además, este hecho tiene una consecuencia inmediata, y es que puede que los parámetros obtenidos en el ajuste del modelo no fueran los mejores, pues se realizaron también 15 ejecuciones. Por lo tanto, se decidió repetir el ajuste pero empleando 30 ejecuciones para cada combinación de valores en los parámetros. Esto ya supone el doble de tiempo de ejecución, y no se aumentó esta cifra más porque comenzaba a ser intratable. Los resultados se muestran en la Tabla 6.6.

Parámetro	1-norma	2-norma
harvestAdjustment	0.56	0.56
harvestVariance	0.1	0
deathAge	38	36
fertility	0.125	0.125
fertilityEndsAge	32	32
spatialHarvestVariance	0.4	0.4
Valor medio de la norma	16375.47	878.72

Tabla 6.6. *Valores obtenidos en el ajuste de los parámetros del modelo. Se ha tomado la mejor de las medias de 30 ejecuciones para cada combinación de parámetros.*

6.2.3. RESULTADOS SOBRE LA UBICACIÓN DE LOS AGENTES

El estudio expuesto anteriormente se refiere únicamente a las cifras de la población, pero el modelo contempla también la ubicación de los asentamientos de los agentes en lugares determinados, y trata de obtener los mismos patrones que los observados mediante el registro arqueológico [6].

En este caso se puede afirmar que los resultados obtenidos ajustan muy bien tanto lo observado en los datos como al modelo NetLogo de Janssen. En las figuras siguientes se muestra este resultado. Los parámetros empleados en las figuras correspondientes a este trabajo son los optimizados en el apartado anterior, mientras que en la versión de NetLogo se usan los de la Tabla 6.3.

En todas ellas se puede observar a la izquierda el modelo replicado en el presente trabajo, mientras que a la izquierda aparece el modelo de Janssen. En rojo aparecen los asentamientos y en amarillo las parcelas de cultivo o granjas. Obsérvese que en los asentamientos está permitido que haya múltiples *households*, por ello su número es menor que el de granjas.

Como se ve, tanto la cantidad de asentamientos y granjas como su distribución coinciden bastante. Evidentemente, están sujetas a cierto grado de aleatoriedad, tanto en el número de agentes como en su ubicación, pero los patrones de asentamiento tanto espacial como temporal coinciden.

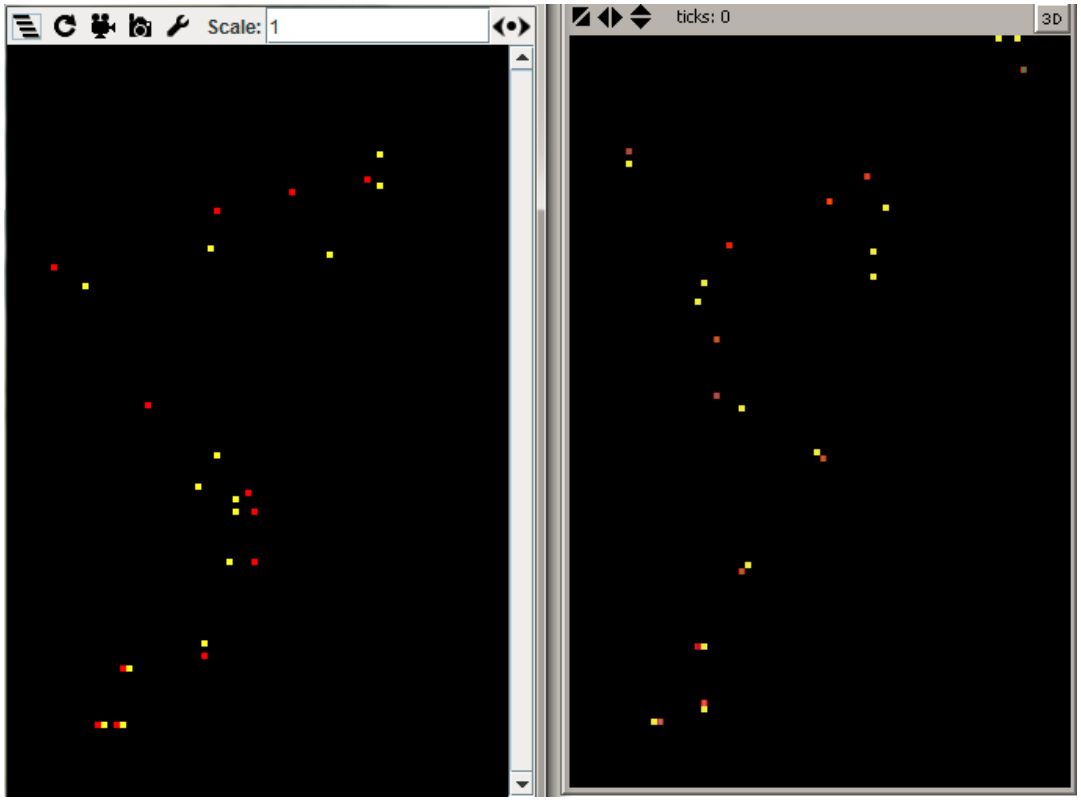

Figura 6.6. *Inicio de la simulación, año 800.*

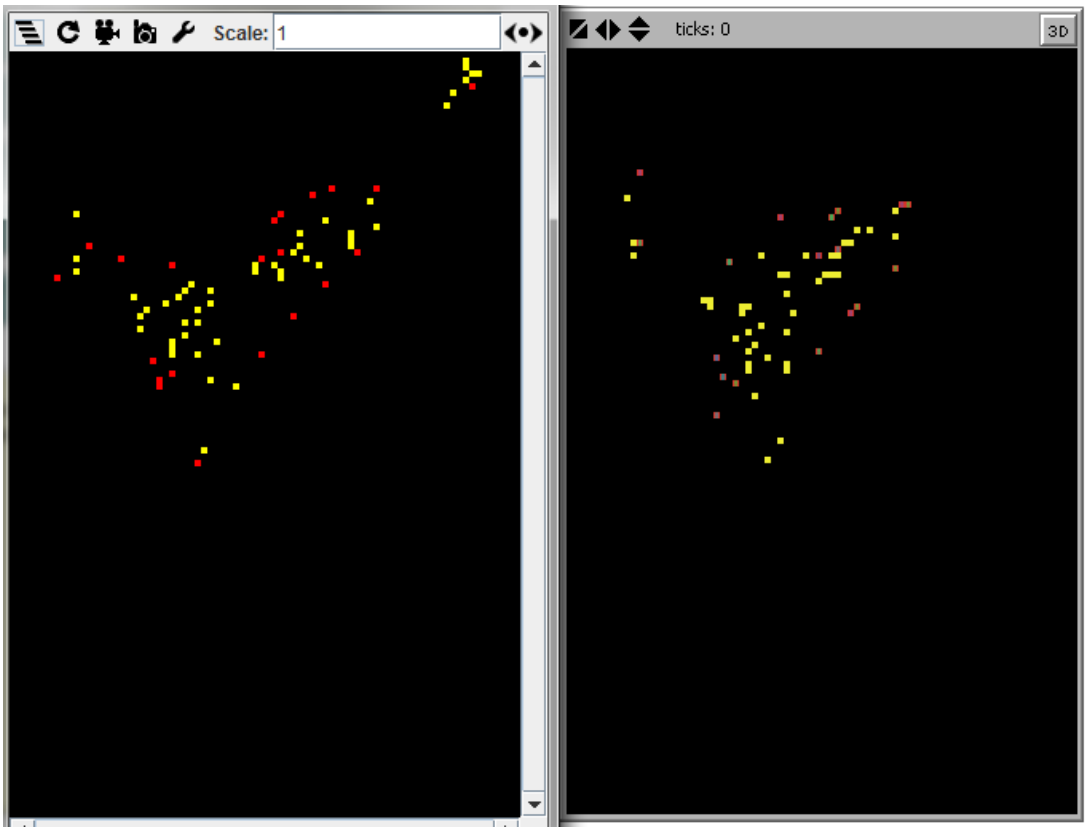

Figura 6.7. *Año 990.*

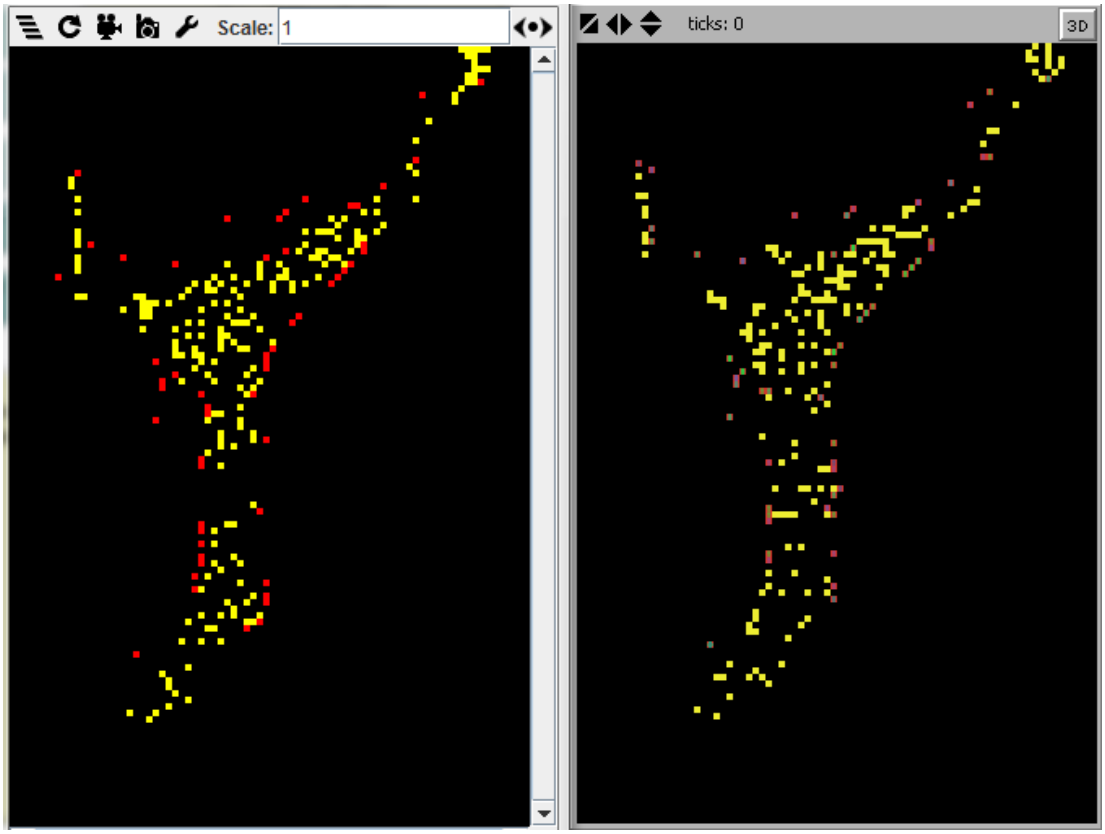

Figura 6.8. *Año 1098.*

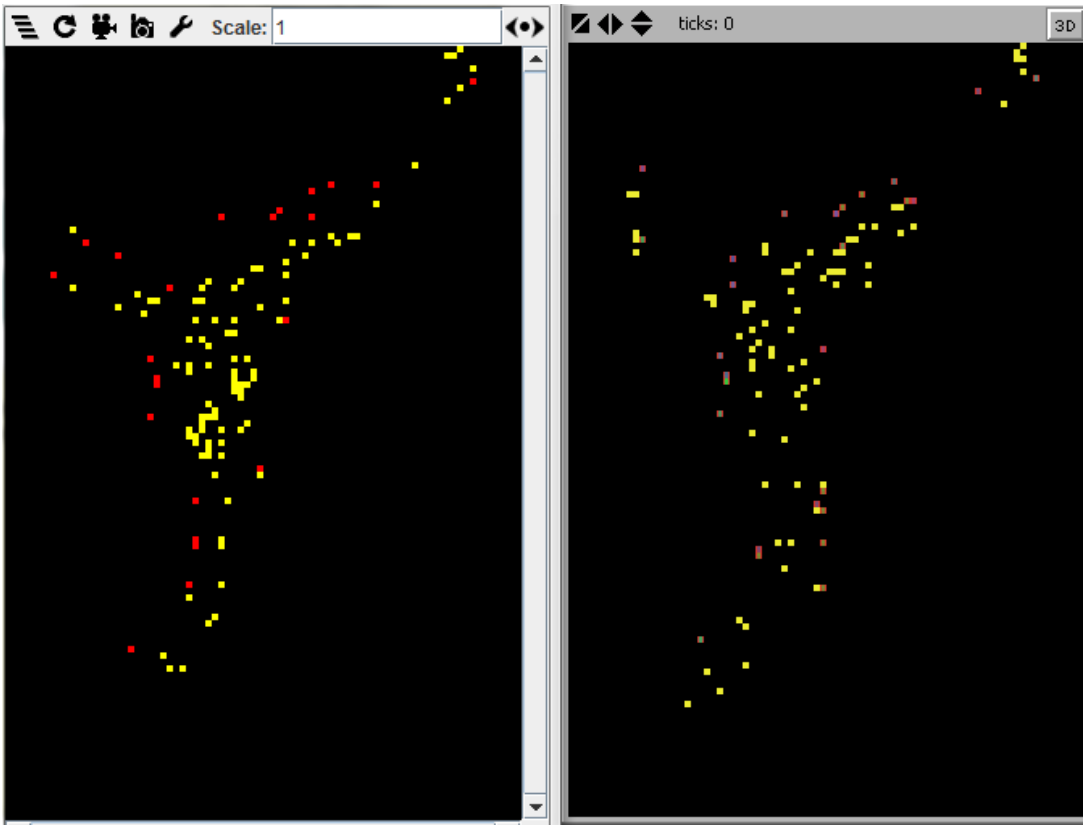

Figura 6.9. *Año 1150.*

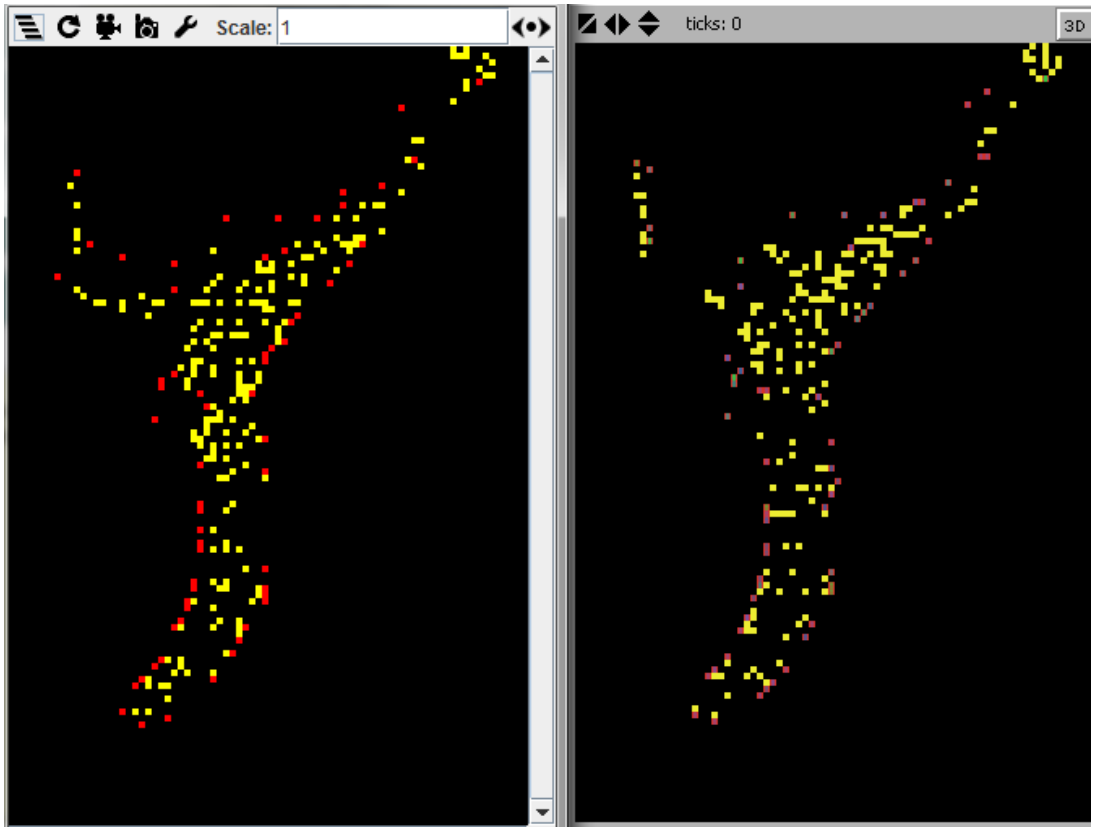

Figura 6.10. *Año 1200.*

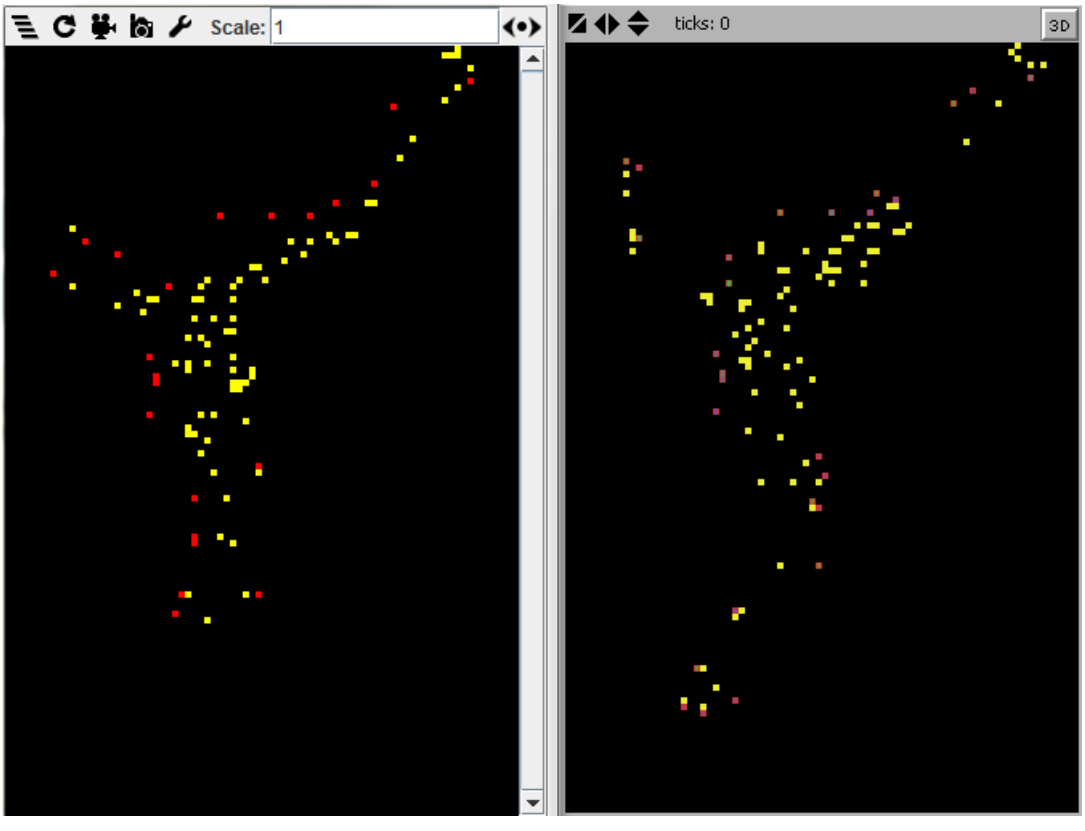

Figura 6.11. *Año 1276.*

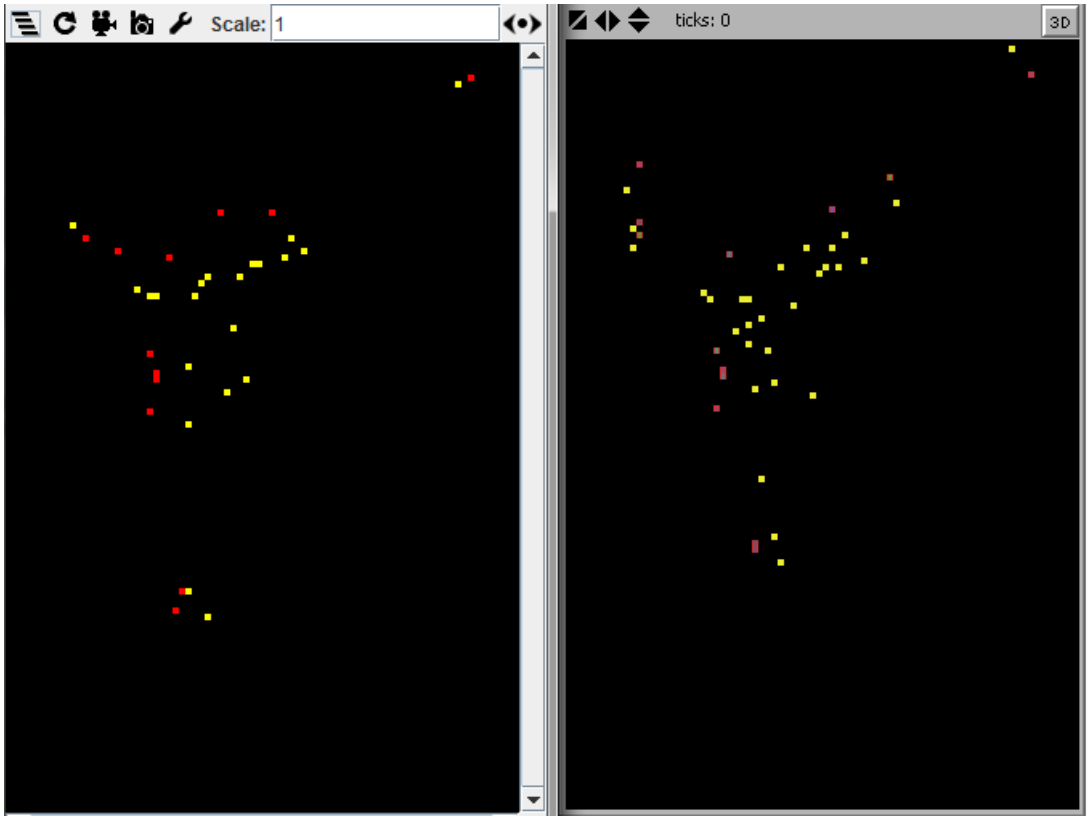

Figura 6.12. *Fin de la simulación, año 1350.*

6.3. ANÁLISIS DE LOS RESULTADOS

En primer lugar y si se observan los resultados de la Tabla 6.6, hay que notar que, pese a haber realizado el doble de simulaciones para cada combinación de parámetros, al final los resultados en cuanto a buen ajuste resultaron peores que los que se habían obtenido previamente, en el caso de ambas normas además. Esto vuelve a poner de manifiesto el alto componente aleatorio que tiene este modelo —o por lo menos esta implementación del mismo—, y lanza al aire la cuestión de si realmente existe un ajuste óptimo, o al menos uno lo suficientemente mejor que el resto. Además, parece que, de haberlo, conseguir calcularlo podría requerir hacer un examen mucho más exhaustivo del espacio de parámetros y, sobre todo, considerar una mayor muestra de ejecuciones para cada combinación de los parámetros. Por tanto, y para estos parámetros que varían, se puede hablar de un conjunto de valores "buenos" que dan prácticamente el mismo rendimiento.

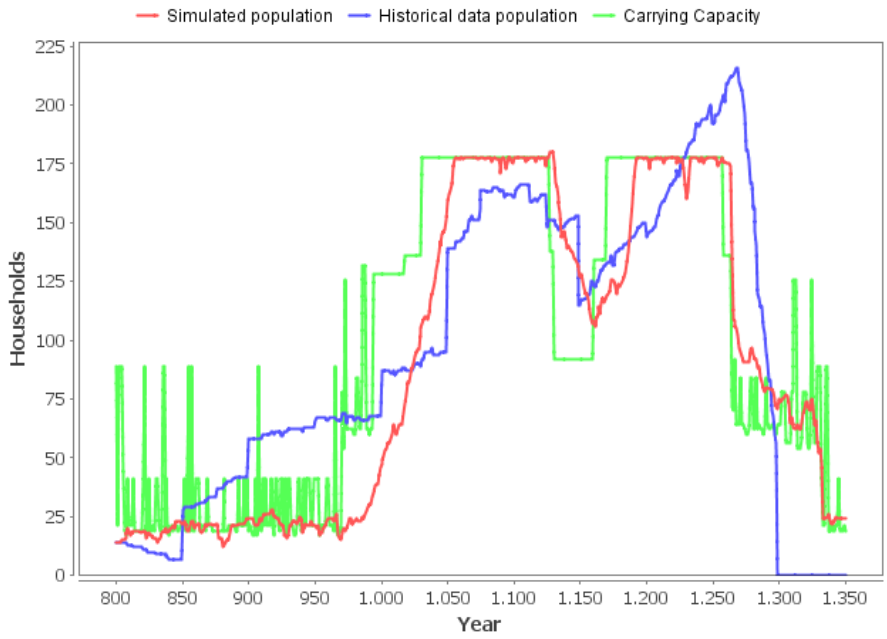

Figura 6.13. *Ejecución con los parámetros de la Tabla 6.6 para la 1-norma. En concreto en esta ejecución se obtuvieron los siguientes valores de ajuste: 14891 para la 1-norma, 803.17 para la 2-norma. Para la capacidad de carga se obtuvieron 19363 y 1061.57 para la 1 y la 2-norma, respectivamente.*

También se observa que, pese a todo, para los parámetros que Janssen da por fundamentales —en el sentido de que son los que determinan la capacidad de carga: *harvestAdjustment* y *spatialHarvestVariance* — se obtuvieron siempre los mismos valores. Esto viene a respaldar la idea que Janssen introducía acerca de la relativa importancia de los detalles modelo en sí —el modelado de las reglas de comportamiento de los agentes y demás consideraciones acerca de los mismos—, frente a la importancia de un buen ajuste de la capacidad de carga.

Similar comportamiento se produjo con *fertility*, obteniendo siempre los mismos valores óptimos. Solo que en este caso se obtuvo siempre el valor que dan Axtell et al., aunque en [7] Janssen comenta que los resultados son prácticamente los mismos para ambos valores de *fertility*.

El caso de *harvestVariance* ya es distinto. Aquí, como sucedía al probar y examinar los parámetros de los otros autores, para que haya un buen ajuste se necesita que tenga un valor bajo, que oscila, según prueba y norma considerada, entre 0 y 0.1. Sin embargo, puesto que en todo ese rango se obtienen buenos ajustes, se tomará siempre el valor 0.1, para dar algo de variabilidad y realismo a la recolección de las cosechas.

Por otra parte, los resultados obtenidos en el ajuste más fino (Tabla 6.5) de la capacidad de carga dieron valores óptimos un tanto fuera de lugar, si se tienen en cuenta los resultados de Janssen. Además de ser valores que no se corresponden con las hipótesis plausibles del modelo en su concepción original [6]. Esto, podría hacer pensar que los resultados que se

obtuvieron no respaldan la argumentación de Janssen acerca de la importancia primordial del ajuste de la capacidad de carga.

Sin embargo, un análisis más detallado de los resultados muestra que esto no tiene por qué ser así, aunque se matiza.

En primer lugar, se realizaron pruebas más exhaustivas de 500 ejecuciones para los valores que se habían obtenido al hacer el primer ajuste de la capacidad de carga, que dio resultados dispares. En la Tabla 6.7 pueden verse los resultados. En la última fila, además, aparece el resultado obtenido para los parámetros que se obtenían como óptimos en el ajuste del modelo completo.

Se observa que, salvo éste último, el resto ofrecen valores muy similares de ajuste para ambas normas. Por tanto, son todos buenos ajustes de la capacidad de carga a los datos históricos. Sin embargo, la última fila, que es precisamente la que mejor ajusta el modelo de la población en todos los casos como se ha visto, no ajusta tan bien.

Una posible explicación de esto es que, como apunta Janssen, el modelo de la población, con sus reglas acerca de los agentes y junto al resto de los parámetros, sí tiene un cierto efecto de suavizado del modelo de la capacidad de carga. En general, está un poco por debajo de esta, de ahí que el valor del ajuste de la capacidad de carga para estos parámetros sea un poco mayor. Lo extraño en tal caso sería lo que ocurre con el modelo de Janssen, pues obtiene exactamente los mismos valores para los dos factores que determinan la capacidad de carga (*harvestAdjustment* y *spatialHarvestVariance*) al ajustar la carga y al ajustar el modelo de población. No obstante, es un asunto sobre el que se podría profundizar en el futuro, cosa que ayudaría a descifrar y comprender más en profundidad el modelo.

En cualquier caso, y a pesar de la pequeña discrepancia con los resultados de Janssen, los resultados obtenidos aquí apoyan esta idea: lo primordial en un buen ajuste es la capacidad de carga.

Con la capacidad de carga bien ajustada —ya sea al mismo "nivel" de la población, como obtiene Janssen, o con un ligero desplazamiento con respecto a ésta, como se ha obtenido aquí— y unos valores para el resto de parámetros dentro del rango aceptable (que aseguren que la población puede crecer), la población tenderá a aproximarse al nivel de carga, aunque su evolución posterior vendrá suavizada por los factores que se comentaban.

harvestAdjustment	*spatialHarvestVariance*	1-norma	2-norma
0.5	0.5	18430.61	1059.52
0.54	0.4	18563.93	1076.59
0.48	0.6	18830.65	1062.06
0.44	0.7	18221.37	1038.96
0.56	0.4	21127.96	1138.78

Tabla 6.7. *Valores del ajuste de la capacidad de carga para diferentes valores de los parámetros.*

7. CONCLUSIONES Y TRABAJO FUTURO

7.1. CONCLUSIONES

En el presente trabajo se ha expuesto una arquitectura para modelos demográficos basados en agentes que ha sido diseñada tratado de adaptarse a los requisitos planteados inicialmente. En primer lugar que fuera flexible y capaz de abarcar el mayor rango de modelos posibles. Con el diseño realizado se ha contemplado que sea capaz de abarcar tanto modelos en los que la demografía es un elemento secundario (en el sentido de que no es el objeto a investigar, pero sí es fundamental para dar realismo a la simulación), como modelos que se centran en el estudio de las propias dinámicas demográficas, o de sus causas y procesos internos.

Por otro lado se ha tratado de que fuera modular y capaz de integrar los procesos demográficos y los factores implicados en ellos por separado, pudiéndose considerar o no dependiendo del problema a modelar.

Para ello, se ha diferenciado entre los actores más "sociales" y otros agentes demográficos, y se ha diferenciado en consecuencia el tipo de entorno que los puede albergar. Además, se ha tenido en cuenta la problemática actual con los modelos demográficos basados en agentes, que no dan "enfoque" de agentes —i.e. modelado mediante reglas de comportamiento— a todos los procesos demográficos que hay implicados y han de recurrir a orientaciones más de arriba abajo para modelar algunos de ellos.

Para tratar este problema, se ha propuesto un sistema basado en módulos de comportamiento y módulos de demografía por separado, que pueden usarse y combinarse para cubrir los diferentes enfoques que se le quiera dar en el modelo a los diferentes factores demográficos.

Esta separación que se ha establecido entre ambos enfoques, y que se plasma en la diferenciación entre módulos demográficos y comportamientos demográficos, permite dar una gran versatilidad a la arquitectura, pudiendo ser capaz de modelar problemas que van desde modelos con un enfoque —conceptual, que no de implementación— como el de microsimulación, modelos híbridos [61] o modelos más puros de ABM.

Sin embargo, hay que apuntar que si bien ha conseguido el objetivo de la flexibilidad, ha sido por dejar también una parte del diseño importante al programador que se encargue de

implementar cada modelo en cuestión. Además de no haber profundizado demasiado en otros aspectos importantes a tener en cuenta, como podría ser el añadir módulos sobre economía, medio ambiente, etc. que fueran intercambiables e interactuaran con el modelo y los agentes.

También se ha considerado el asunto de las redes sociales de los agentes, desde una doble perspectiva: local —al nivel de agentes—, y global, que permite estudiar la emergencia y evolución de las mismas, y cómo las segundas surgen a partir de las primeras.

Por otra parte, el estudio del caso de Artificial Anasazi que, si bien no es un modelo con un comportamiento demográfico muy complejo, ha reflejado adaptarse relativamente bien a la arquitectura.

Los resultados obtenidos en las pruebas realizadas con éste reflejan una gran similitud entre el modelo de Janssen [7], que es al que se ha tenido acceso para poder realizar pruebas y comparar, y el aquí replicado. Esta similitud de comportamiento se da tanto en niveles de población como en distribución espacio-temporal de la misma. Sin embargo, hay una diferencia en el comportamiento, en lo relativo a los parámetros que afectan a la aleatoriedad de ambos modelos, cuya causa no ha podido ser determinada, aunque exista la posibilidad tanto de un error del autor en la implementación del modelo, como un diferente comportamiento aleatorio de las plataformas empleadas.

No obstante, y pese a esa diferencia en la respuesta del modelo a los valores de los parámetros, los experimentos aquí realizados parecen apoyar las tesis de Janssen en su artículo.

7.2. TRABAJO FUTURO

El trabajo, como se ha comentado ya en la Introducción, no ha pretendido sino esbozar las líneas principales de una tarea tan compleja y enorme como sería hacer una arquitectura que permitiera implementar todo tipo de problemas y modelos sobre demografía, si acaso esto fuera posible. Por ello, muchas cosas han tenido que ser dejadas en el tintero. A continuación se expone una lista de algunas propuestas que hace el autor acerca de cómo y por dónde ampliar este trabajo:

- Profundizar más en especificaciones más concretas de los comportamientos, que pudieran tener en cuenta aspectos de Inteligencia Artificial que se usan habitualmente con agentes como aprendizaje automático, lógica *fuzzy*, generación de lenguaje o sistemas de reglas [19].
- Explorar posibilidades de ampliación del sistema mediante módulos para fenómenos que puedan afectar a la demografía, como economía, medio ambiente, etc.

- Integrar GIS como parte esencial de los modelos. Actualmente, es una técnica fundamental si se quiere modelar el medio geográfico con realismo. Además, hay una gran cantidad de ABM que emplean GIS [81].
- Definir un módulo que permita obtener estadísticas e indicadores demográficos como pirámides de población y que se adapte a las condiciones concretas del problema.
- Poner a prueba el diseño de la arquitectura. Una buena manera puede ser mediante la replicación de más modelos, ya que así se cubriría un doble objetivo. Por una parte, se podría refinar la arquitectura, mejorando aquellos aspectos que el autor haya podido dejar escapar. Por otro lado, se contribuiría a la comunidad de ABSS gracias a la replicación y análisis de modelos ya existentes, lo que permitiría validar y verificar dichos modelos.

APÉNDICE A. CARGA DE DATOS EXTERNOS.

Se presenta a continuación el código que maneja la carga, inicialización y actualización de los datos externos que modelan el entorno en el modelo Artificial Anasazi. Se presentan por no estar bien documentada la manera de manejarlos en la literatura disponible.

```
public class ValleyFloor extends MutableField2D {

	public static final int WIDTH = 80;
	public static final int HEIGHT = 120;
	private static MersenneTwisterFast random = new
MersenneTwisterFast();
	private ObjectGrid2D grid = new ObjectGrid2D(WIDTH,HEIGHT);
	private boolean streamsexist;
	private boolean alluviumexist;
	private Vector<Waterpoint> waterpoints;
	private Vector<Integer> mapdata;
	private Vector<Integer> waterdata;
	private Vector<Double> apdsidata;
	private Vector<Double> environmentdata;
	private Vector<Double> historicaldata;
	private Vector<HistoricalSettlement> historicalSettlements;

	//where the historical settlements will be represented
	public SparseGrid2D hisPopulation = new SparseGrid2D(WIDTH,HEIGHT);

	public enum Zone {
		General, North, NorthDunes, Mid, MidDunes, Natural, Upland,
Kinbiko, Empty
	}

	private enum MaizeZone {
		Yield1, Yield2, Yield3, SandDune, NoYield, Empty
	}

	//the next inner class represents a plot (100m x 100m size) in the
valley, the equivalent to
	//a patch in NetLogo
	public class Plot {
		private Color color;
		private boolean watersource;
		private Zone zone;
		private double apdsi;
		private double hydro;
		private double quality;
		private MaizeZone maizeZone;
		private double yield;
		private double BaseYield;
		private boolean ocfarm;
```

```
private int ochousehold;

public Color whatColor() {
	return color;
}
public void setColor(Color color) {
	this.color = color;
}
public boolean isWatersource() {
	return watersource;
}
public void ssetWatersource(boolean watersource) {
	this.watersource = watersource;
}
public Zone getZone() {
	return zone;
}
public void setZone(Zone zone) {
	this.zone = zone;
}
public double getApdsi() {
	return apdsi;
}
public void ssetApdsi(double apdsi) {
	this.apdsi = apdsi;
}
public double getHydro() {
	return hydro;
}
public void ssetHydro(double generalhydro) {
	this.hydro = generalhydro;
}
public double getQuality() {
	return quality;
}
public void ssetQuality(double quality) {
	this.quality = quality;
}
public MaizeZone getMaizeZone() {
	return maizeZone;
}
public void ssetMaizeZone(MaizeZone maizeZone) {
	this.maizeZone = maizeZone;
}
public double getYield() {
	return yield;
}
public void ssetYield(double yield) {
	this.yield = yield;
}
public double getBaseYield() {
	return BaseYield;
}
public void ssetBaseYield(double d) {
	BaseYield = d;
}
public boolean isOcfarm() {
	return ocfarm;
}
```

```
		public void ssetOcfarm(boolean ocfarm) {
			this.ocfarm = ocfarm;
		}
		public int getOchousehold() {
			return ochousehold;
		}
		public void ssetOchousehold(int ochousehold) {
			this.ochousehold = ochousehold;
		}

		public void calculateBaseYield(double harvestAdjustment) {
			ssetBaseYield(getYield() * getQuality() *
harvestAdjustment);
		}

		public void incHouseholdNum() {
			ssetOchousehold(getOchousehold() + 1);
		}

		public void decHouseholdNum() {
			ssetOchousehold(getOchousehold() - 1);
		}

		public Plot() {
			ocfarm = false;
			ochousehold = 0;
		}
	}

	public class Waterpoint {
		final int x;
		final int y;
		final int sarg;
		final int meterNorth;
		final int meterEast;
		final int typeWater;
		final int startDate;
		final int endDate;

		public Waterpoint(int x, int y, int sarg, int meterNorth, int
meterEast, int typeWater, int startDate, int endDate) {
			this.x = x;
			this.y = y;
			this.sarg = sarg;
			this.meterNorth = meterNorth;
			this.meterEast = meterEast;
			this.typeWater = typeWater;
			this.startDate = startDate;
			this.endDate = endDate;
		}
	}

	public class HistoricalSettlement {
		Int2D location;
		double SARG;
		double meterNorth;
		double meterEast;
		double startdate;
		double enddate;
```

```
		double mediandate;
		double typeset;
		double sizeset;
		double description;
		double roomcount;
		double elevation;
		double baselinehouseholds;
		int nrhouseholds;
		boolean visible;

		public HistoricalSettlement(Int2D location, double sARG,
				double meterNorth, double meterEast, double
startdate,
				double enddate, double mediandate, double typeset,
double sizeset,
				double description, double roomcount, double
elevation,
				double baselinehouseholds, int nrhouseholds,
boolean visible) {
			this.location = location;
			SARG = sARG;
			this.meterNorth = meterNorth;
			this.meterEast = meterEast;
			this.startdate = startdate;
			this.enddate = enddate;
			this.mediandate = mediandate;
			this.typeset = typeset;
			this.sizeset = sizeset;
			this.description = description;
			this.roomcount = roomcount;
			this.elevation = elevation;
			this.baselinehouseholds = baselinehouseholds;
			this.nrhouseholds = nrhouseholds;
			this.visible = visible;
		}

		public double getStartdate() {
			return startdate;
		}

		public void setStartdate(double startdate) {
			this.startdate = startdate;
		}

		public double getEnddate() {
			return enddate;
		}

		public void setEnddate(double enddate) {
			this.enddate = enddate;
		}

		public int getNrhouseholds() {
			return nrhouseholds;
		}

		public void setNrhouseholds(int nrhouseholds) {
			this.nrhouseholds = nrhouseholds;
		}
```

```
		public void checkVisibility(int year) {
			//if in the year "year" the settlement existed,
visibility will be set to true,
			//otherwise it will be set to false
			visible = ((year >= startdate) && (year < enddate) &&
((int) typeset == 1));
		}

		public void setnrhouseholds(int year) {
			if ((year > mediandate) && (year != enddate)) {
				nrhouseholds = (int) Math.ceil((baselinehouseholds
* (enddate - year) / (enddate - mediandate)));
				if (nrhouseholds < 1) nrhouseholds = 1;
			}
			if ((year <= mediandate) && (mediandate != startdate)) {
				nrhouseholds = (int) Math.ceil((baselinehouseholds
* (year - startdate) / (mediandate - startdate)));
				if (nrhouseholds < 1) nrhouseholds = 1;
			}
		}

		@Override
		public String toString() {
			return new String(""+nrhouseholds);
		}
	}

	public ValleyFloor(double spatialHarvestVariance) {
		super(WIDTH,HEIGHT);
		initMap(spatialHarvestVariance);
	}

	@Override
	public void step(SimState state) {
		int year = ((LongHouseValley) state).year();
		calculateYield(year);
		calculateBaseYield(((LongHouseValley)
state).harvestAdjustment);
		water(year);
		updateHistoricalData(year);
	}

	/**Updates the grid of historical settlements, acording to their
existence dates.
	 * Returns the population (number of historical households,
	 * i.e. sum of the households in each settlement*/
	public int updateHistoricalData(int year) {
		int population = 0;
		for (HistoricalSettlement hs : historicalSettlements) {
			hs.checkVisibility(year);
			hs.setnrhouseholds(year);
			if (hs.visible){
				hisPopulation.setObjectLocation(hs, hs.location);
				population+=hs.nrhouseholds;
			} else {
				hisPopulation.remove(hs);
			}
		}
```

```
		return population;
	}

	public void calculateBaseYield(double harvestAdjustment) {
		for (int x = 0;x< WIDTH;x++) {
			for (int y = 0;y<HEIGHT;y++) {
				plotAt(x,y).calculateBaseYield(harvestAdjustment);
			}
		}
	}

	public int calculateCarryingCapacity() {
		int cc = 0;
		for (int x = 0;x< WIDTH;x++) {
			for (int y = 0;y<HEIGHT;y++) {
				if (plotAt(x,y).getBaseYield() >=
LongHouseValley.householdMinNutritionNeed) {
					cc++;
				}
			}
		}
		return cc;
	}

	public Plot plotAt(int x, int y) {
		return (Plot) grid.get(x,y);
	}

	private void setPlotAt(int x, int y, Object plot) {
		grid.set(x, y, (Plot) plot);
	}

	private void initMap(double spatialHarvestVariance) {

		//load data and create the valley map
		for (int x = 0;x< WIDTH;x++) {
			for (int y = 0;y<HEIGHT;y++) {
				Plot plot = new Plot();
				plot.ssetWatersource(false);
				double quality = random.nextGaussian() *
spatialHarvestVariance + 1;
				if (quality >= 0) plot.ssetQuality(quality);
				else plot.ssetQuality(0);
				setPlotAt(x,y,plot);
			}
		}
		try {
			loadDataFiles();
		} catch (IOException e) {
			e.printStackTrace();
		}
		int xx = 0;
		int yy = 0;
		for (int val:mapdata) {
			switch (val) {
			case 0 : {
				plotAt(xx,yy).setColor(Color.black);
				plotAt(xx,yy).setZone(Zone.General);
				plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield2);
```

```
                        break;
                    }
                    case 10 : {
                        plotAt(xx,yy).setColor(Color.red);
                        plotAt(xx,yy).setZone(Zone.North);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield1);
                        break;
                    }
                    case 15 : {
                        plotAt(xx,yy).setColor(Color.white);
                        plotAt(xx,yy).setZone(Zone.NorthDunes);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.SandDune);
                        break;
                    }
                    case 20 : {
                        plotAt(xx,yy).setColor(Color.gray);
                        plotAt(xx,yy).setZone(Zone.Mid);
                        if (xx <= 74)
plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield1);
                        else
plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield2);
                        break;
                    }
                    case 25 : {
                        plotAt(xx,yy).setColor(Color.white);
                        plotAt(xx,yy).setZone(Zone.MidDunes);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.SandDune);
                        break;
                    }
                    case 30 : {
                        plotAt(xx,yy).setColor(Color.yellow);
                        plotAt(xx,yy).setZone(Zone.Natural);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.NoYield);
                        break;
                    }
                    case 40 : {
                        plotAt(xx,yy).setColor(Color.blue);
                        plotAt(xx,yy).setZone(Zone.Upland);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield3);
                        break;
                    }
                    case 50 : {
                        plotAt(xx,yy).setColor(Color.pink);
                        plotAt(xx,yy).setZone(Zone.Kinbiko);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.Yield1);
                        break;
                    }
                    case 60 : {
                        plotAt(xx,yy).setColor(Color.white);
                        plotAt(xx,yy).setZone(Zone.Empty);
                        plotAt(xx,yy).ssetMaizeZone(MaizeZone.Empty);
                        break;
                    }
                    }
                    if (yy <119) yy++;
                    else {xx++; yy = 0;}
                }

                //create the waterpoints
```

```
		waterpoints = new Vector<Waterpoint>();
		int i = 0;
		while (i+6 < waterdata.size()) {
			int sarg = waterdata.get(i); i++;
			int meterNorth = waterdata.get(i); i++;
			int meterEast = waterdata.get(i); i++;
			int typeWater = waterdata.get(i); i++;
			int startDate = waterdata.get(i); i++;
			int endDate = waterdata.get(i); i++;
			int x = (int) (25 + ((meterEast - 2392) / 93.5)); //this
is a translation from the input data in meters into location on the map.
			int y = (int) Math.floor(45 + (37.6 + ((meterNorth -
7954) / 93.5)));
			//there are 2 waterpoints which seem to be wrong,
because they both have meterNorth = 0 and meterEast = 0. ignore them
			if (x>0 && y>0) {
			waterpoints.add(new Waterpoint(x,119-
y,sarg,meterNorth,meterEast,typeWater,startDate,endDate));
			}
		}

		//create the historical settlements
		historicalSettlements = new Vector<HistoricalSettlement>();
		int k = 0;
		while (k+12 < historicaldata.size()) {
			double SARG = historicaldata.get(k);k++;
			double meterNorth = historicaldata.get(k);k++;
			double meterEast = historicaldata.get(k);k++;
			double startdate = historicaldata.get(k);k++;
			double enddate = historicaldata.get(k);k++;
			double mediandate = 1950 - historicaldata.get(k);k++;
			double typeset = historicaldata.get(k);k++;
			double sizeset = historicaldata.get(k);k++;
			double description = historicaldata.get(k);k++;
			double roomcount = historicaldata.get(k);k++;
			double elevation = historicaldata.get(k);k++;
			double baselinehouseholds = historicaldata.get(k);k++;

			int x = (int) (25 + ((meterEast - 2392) / 93.5)); //this
is a translation from the input data in meters into location on the map.
			int y = (int) Math.floor(45 + (37.6 + ((meterNorth -
7954) / 93.5)));
			Int2D location = new Int2D(x,119-y);
			int nrhouseholds = 0;
			boolean visible = false;
			historicalSettlements.add(new
HistoricalSettlement(location,
					SARG, meterNorth, meterEast, startdate,
enddate,
					mediandate, typeset, sizeset, description,
roomcount,
					elevation, baselinehouseholds,
nrhouseholds, visible));
		}
		//the next is done in order to have the settlements shown in
the GUI before the simulation starts
		int initialYear = 800;
		updateHistoricalData(initialYear);
	}
```

```
	private void loadDataFiles() throws IOException {
		Scanner s = null;
		InputStream is1 = getClass().getResourceAsStream("Map.txt");
		InputStream is2 =
getClass().getResourceAsStream("adjustedPDSI.txt");
		InputStream is3 = getClass().getResourceAsStream("water.txt");
		InputStream is4 =
getClass().getResourceAsStream("environment.txt");
		InputStream is5 =
getClass().getResourceAsStream("settlements.txt");
		InputStreamReader isr1 = new InputStreamReader(is1);
		InputStreamReader isr2 = new InputStreamReader(is2);
		InputStreamReader isr3 = new InputStreamReader(is3);
		InputStreamReader isr4 = new InputStreamReader(is4);
		InputStreamReader isr5 = new InputStreamReader(is5);

	    try {
	        s = new Scanner(new BufferedReader(isr1));
	        mapdata = new Vector<Integer>();

	        while (s.hasNext()) {
	            mapdata.add(s.nextInt());
	        }
	    } finally {
	        if (s != null) {
	            s.close();
	        }
	    }

	    try {
	         s = new Scanner(new BufferedReader(isr2));
	         apdsidata = new Vector<Double>();
	         while (s.hasNext()) {
	                apdsidata.add(Double.parseDouble(s.next()));
	         }
	    } finally {
	        if (s != null) {
	            s.close();
	        }
	    }

	    try {
	         s = new Scanner(new BufferedReader(isr3));
	         waterdata = new Vector<Integer>();
	         while (s.hasNext()) {
	                waterdata.add((s.nextInt()));
	         }
	    } finally {
	        if (s != null) {
	            s.close();
	        }
	    }

	    try {
	         s = new Scanner(new BufferedReader(isr4));
	         environmentdata = new Vector<Double>();
	         while (s.hasNext()) {
	                environmentdata.add(Double.parseDouble(s.next()));
```

```
            }
        } finally {
            if (s != null) {
                s.close();
            }
        }

        try {
            s = new Scanner(new BufferedReader(isr5));
            historicaldata = new Vector<Double>();
            while (s.hasNext()) {
                  historicaldata.add(Double.parseDouble(s.next()));
            }
        } finally {
            if (s != null) {
                s.close();
            }
        }
      }

      public void calculateYield(int year) {
            double generalapdsi = apdsidata.get(year - 200);
            double northapdsi = apdsidata.get(1100 + year);
            double midapdsi = apdsidata.get(2400 + year);
            double naturalapdsi = apdsidata.get(3700 + year);
            double uplandapdsi = apdsidata.get(3700 + year);
            double kinbikoapdsi = apdsidata.get(1100 + year);

            double generalhydro = environmentdata.get(((year -
382)*15)+1);
            double northhydro = environmentdata.get(((year - 382)*15)+4);
            double midhydro = environmentdata.get(((year - 382)*15)+7);
            double naturalhydro = environmentdata.get(((year -
382)*15)+10);
            double uplandhydro = environmentdata.get(((year -
382)*15)+10);
            double kinbikohydro = environmentdata.get(((year -
382)*15)+13);

            for (int x = 0;x< WIDTH;x++) {
                  for (int y = 0;y<HEIGHT;y++) {
                        switch (plotAt(x,y).getZone()) {
                        case General : {
                              plotAt(x,y).ssetApdsi(generalapdsi);
                              plotAt(x,y).ssetHydro(generalhydro);
                              break;
                        }
                        case North : {
                              plotAt(x,y).ssetApdsi(northapdsi);
                              plotAt(x,y).ssetHydro(northhydro);
                              break;
                        }
                        case Mid : {
                              plotAt(x,y).ssetApdsi(midapdsi);
                              plotAt(x,y).ssetHydro(midhydro);
                              break;
                        }
                        case Natural : {
                              plotAt(x,y).ssetApdsi(naturalapdsi);
```

```
                              plotAt(x,y).ssetHydro(naturalhydro);
                              break;
                        }
                        case Upland : {
                              plotAt(x,y).ssetApdsi(uplandapdsi);
                              plotAt(x,y).ssetHydro(uplandhydro);
                              break;
                        }
                        case Kinbiko : {
                              plotAt(x,y).ssetApdsi(kinbikoapdsi);
                              plotAt(x,y).ssetHydro(kinbikohydro);
                              break;
                        }
                        }

                        switch (plotAt(x,y).getMaizeZone()) {
                        case NoYield : {
                              plotAt(x,y).ssetYield(0);
                              break;
                        }
                        case Empty : {
                              plotAt(x,y).ssetYield(0);
                              break;
                        }
                        case Yield1 : {
                              if (plotAt(x,y).getApdsi() >= 3.0)
plotAt(x,y).ssetYield(1153);
                              else if ((plotAt(x,y).getApdsi() >= 1.0) &&
(plotAt(x,y).getApdsi() < 3.0)) plotAt(x,y).ssetYield(988);
                              else if ((plotAt(x,y).getApdsi() > -1.0) &&
(plotAt(x,y).getApdsi() < 1.0)) plotAt(x,y).ssetYield(821);
                              else if ((plotAt(x,y).getApdsi() > -3.0) &&
(plotAt(x,y).getApdsi() <= -1.0)) plotAt(x,y).ssetYield(719);
                              else if (plotAt(x,y).getApdsi() <= -3.0)
plotAt(x,y).ssetYield(617);
                              break;
                              }
                        case Yield2 : {
                              if (plotAt(x,y).getApdsi() >= 3.0)
plotAt(x,y).ssetYield(961);
                              else if ((plotAt(x,y).getApdsi() >= 1.0) &&
(plotAt(x,y).getApdsi() < 3.0)) plotAt(x,y).ssetYield(824);
                              else if ((plotAt(x,y).getApdsi() > -1.0) &&
(plotAt(x,y).getApdsi() < 1.0)) plotAt(x,y).ssetYield(684);
                              else if ((plotAt(x,y).getApdsi() > -3.0) &&
(plotAt(x,y).getApdsi() <= -1.0)) plotAt(x,y).ssetYield(599);
                              else if (plotAt(x,y).getApdsi() <= -3.0)
plotAt(x,y).ssetYield(514);
                              break;
                        }
                        case Yield3 : {
                              if (plotAt(x,y).getApdsi() >= 3.0)
plotAt(x,y).ssetYield(769);
                              else if ((plotAt(x,y).getApdsi() >= 1.0) &&
(plotAt(x,y).getApdsi() < 3.0)) plotAt(x,y).ssetYield(659);
                              else if ((plotAt(x,y).getApdsi() > -1.0) &&
(plotAt(x,y).getApdsi() < 1.0)) plotAt(x,y).ssetYield(547);
                              else if ((plotAt(x,y).getApdsi() > -3.0) &&
(plotAt(x,y).getApdsi() <= -1.0)) plotAt(x,y).ssetYield(479);
```

```
                                    else if (plotAt(x,y).getApdsi() <= -3.0)
plotAt(x,y).ssetYield(411);
                                    break;
                              }
                              case SandDune : {
                                    if (plotAt(x,y).getApdsi() >= 3.0)
plotAt(x,y).ssetYield(1201);
                                    else if ((plotAt(x,y).getApdsi() >= 1.0) &&
(plotAt(x,y).getApdsi() < 3.0)) plotAt(x,y).ssetYield(1030);
                                    else if ((plotAt(x,y).getApdsi() > -1.0) &&
(plotAt(x,y).getApdsi() < 1.0)) plotAt(x,y).ssetYield(855);
                                    else if ((plotAt(x,y).getApdsi() > -3.0) &&
(plotAt(x,y).getApdsi() <= -1.0)) plotAt(x,y).ssetYield(749);
                                    else if (plotAt(x,y).getApdsi() <= -3.0)
plotAt(x,y).ssetYield(642);
                                    break;
                              }
                              }
                        }
                  }
            }

            public void water(int year) {
                  streamsexist = ((year >= 280 && year < 360) || (year >= 800 &&
year < 930) || (year >= 1300 && year < 1450));
                  alluviumexist = (((year >= 420) && (year < 560)) || ((year >=
630) && (year < 680)) || ((year >= 980) && (year < 1120)) || ((year >=
1180) && (year < 1230)));

                  for (int x = 0;x< WIDTH;x++) {
                        for (int y = 0;y<HEIGHT;y++) {
                              boolean ws = ((alluviumexist &&
(plotAt(x,y).getZone() == Zone.General || plotAt(x,y).getZone() ==
Zone.North || plotAt(x,y).getZone() == Zone.Mid ||plotAt(x,y).getZone() ==
Zone.Kinbiko))
                              || (streamsexist && plotAt(x,y).getZone() ==
Zone.Kinbiko));
                                    plotAt(x,y).ssetWatersource(ws);
                        }
                  }

                  plotAt(72,119-114).ssetWatersource(true);
                  plotAt(70,119-113).ssetWatersource(true);
                  plotAt(69,119-112).ssetWatersource(true);
                  plotAt(68,119-111).ssetWatersource(true);
                  plotAt(67,119-110).ssetWatersource(true);
                  plotAt(66,119-109).ssetWatersource(true);
                  plotAt(65,119-108).ssetWatersource(true);
                  plotAt(65,119-107).ssetWatersource(true);

                  for (Waterpoint wp : waterpoints) {
                        boolean ws = (wp.typeWater == 2) || (wp.typeWater == 3
&& (year >= wp.startDate && year <= wp.endDate));
                              plotAt(wp.x,wp.y).ssetWatersource(ws);
                  }
            }

      }
```

APÉNDICE B. APLICACIÓN DE LA ARQUITECTURA PROPUESTA EN UN MODELO DE SIMULACIÓN DE LA DINÁMICA DE MATRIMONIOS Y DIVORCIOS.

En este Apéndice se presenta un modelo demográfico que se desarrolló como parte del trabajo de una asignatura. Posteriormente, durante la elaboración del presente trabajo, se decidió adaptarlo a la arquitectura que se ha descrito. Aunque el modelo de simulación no se llegó a implantar totalmente por falta de tiempo, sí que se creó su estructura de clases, de acuerdo a la arquitectura.

El modelo es una versión ampliada de MADAM [73], del que ya se habló en el Capítulo sobre el Estado del Arte. El modelo original no consideraba un "círculo social", ni tenía en cuenta nacimientos o muertes de los agentes, ni tampoco consideraba una estructura física como entorno para los agentes, ya que los candidatos a pareja eran simplemente escogidos al azar de entre toda la población. En esta versión, en cambio, se decidió dotar de todos estos aspectos al modelo, para así hacerlo más rico.

En primer lugar, se presentará una descripción del modelo siguiendo un poco la orientación del protocolo ODD, aunque dejando de lado múltiples aspectos del mismo, pues no es el propósito aquí el describir el modelo con la finalidad de que sea replicado ni estudiado, sino tan sólo de ilustrar un poco más el resto del trabajo realizado.

En segundo lugar, se mostrará cómo se ha adaptado dicho problema a la arquitectura.

Descripción del modelo

1. Propósito

El propósito de este modelo es probar cómo se adapta a la arquitectura del presente trabajo. El modelo toma un modelo existente anterior, MADAM, y lo amplía tomando en consideración una mayor dinámica social y demográfica que el anterior.

El próposito del modelo original MADAM es investigar los procesos demográficos del matrimonio y el divorcio, e intentar explicarlos a partir de la hipótesis de que las parejas se

formarán de acuerdo al principio de homofilia. Dos personas se emparejarán cuanto más tengan en común.

Para ello, se modelan los emparejamientos entre los agentes de manera que buscan una pareja que sea similar en características, de acuerdo a una exigencia mínima que se va relajando con el tiempo. Además, se permiten los divorcios. Si una persona casada encuentra a otra persona con la que tiene más en común. El objetivo es, posteriormente, comparar las tasas de divorcios y matrimonios que se obtienen con datos reales.

2. Entidades, variables de estado y escalas

En primer lugar se tienen los agentes del modelo, que representan a personas en busca de pareja. Los agentes poseen las siguientes variables de estado:

- *Ubicación*: la situación espacial en el entorno, formada por una dupla (*x,y*).
- *Etapas de simulación*: representada mediante un entero, es la edad del agente, pero expresada en etapas de simulación, no en años (1 año = 50 etapas de simulación).
- *Edad muerte*: un entero que representa la edad a la que la persona morirá, expresada en años, aunque puede cambiar durante la ejecución si logra sobrevivir a cierto tiempo.
- *Características*: un vector de *booleanos* que representa una serie de características abstractas sobre la persona en concreto, como podrían ser sus gustos, su color de pelo, etc.
- *Nivel de exigencia*: es un entero. Representa el número mínimo de características comunes que una potencial pareja ha de compartir con el agente. Va disminuyendo conforme la persona va avanzando en edad, siempre que siga soltera.
- *Género*: puede ser masculino o femenino.
- *Emparejado*: una variable *booleana* que representa si una persona está casada o permanece soltera.
- *Familia*: una lista con referencias a los miembros de su familia cercana (padres, madres, hijos/as o hermanos/as).
- *Amigos*: una lista con referencia a los amigos del agente, las personas con las que se relaciona y de entre las que buscará pareja.

Por otra parte se tiene el entorno físico en donde están situados los agentes. Está formado por un *grid* rectangular de tamaño arbitrario, alto x ancho, que son variables de estado del modelo. Cada celdilla del *grid* no representa ninguna superficie en particular. Sin embargo, sí que hay un radio sobre el que los agentes buscarán amigos con los que relacionarse a su alrededor, representando una vecindad típica de estas representaciones del espacio, como puede ser la distancia de Moore.

Como variables de entorno se tiene el número de características totales de la población *N* y el número de características por persona, *k*. El primero representa el conjunto de todas las posibles características existentes en el “mundo” de la simulación. El segundo representa,

de entre todas las características existentes, las que un determinado agente posee como propias. Por tanto, se cumple que *k* es menor que *N* (*k*<*N*). Además, hay una *población inicial*. La relación entre ésta y el tamaño del *grid*, determinará la densidad de la población, y por tanto el que haya mayor o menor número de amistades entre los agentes.

Finalmente, cada etapa de la simulación representa una fracción de año. En concreto, 50 etapas forman un año. Por tanto, el tiempo está representado de forma discreta.

3. Process overview and scheduling

En cada etapa de la simulación, cada agente ejecuta la siguiente secuencia de acciones, en el orden en que aparecen:

1. Conocer gente.
2. Buscar pareja.
3. Aumentar la edad.
4. Tener hijos.
5. Morir.

Así mismo, en cada etapa se toman los agentes al azar de forma secuencial, y para cada uno de ellos se ejecutan las acciones descritas. Todo esto tiene lugar de forma asíncrona.

3. Inicialización

Inicialmente, se crean una serie de agentes, de acuerdo a la *población inicial*. Cada vez que se crea un agente nuevo, ya sea en la inicialización o por nacimiento, se le asignan *k* características al azar, de entre las *N* totales. Además, se le asigna una ubicación aleatoria en el *grid*. Además, al *nivel de exigencia* se le asigna el valor *k*. El género de la persona estará determinado por una probabilidad dada de antemano, la *tasa de feminidad al nacimiento*. En último lugar, cada vez que un nuevo agente es creado, un valor entero aleatorio entre 0 y 49 con probabilidad uniforme es sumado a *etapas de simulación*, a parte de la edad en etapas que el agente tenga (que será la edad en años que se le quiera dar, o 0 si es un recién nacido, multiplicada por 50, el número de etapas por año). Esto es para evitar que todos los agentes se "actualicen" a la vez, pues ciertos procesos, como buscar amigos, tienen lugar en las etapas número 0 y 25 de cada año, es decir, si $etapas\ de\ simulación \equiv 0\ (mod\ 25)$.

Posteriormente, y esto ya sólo en la inicialización, se crean una serie de relaciones de amistad entre los agentes recién creados, de acuerdo al siguiente algoritmo:

```
Bag gente = population.getAllNodes();
            for (int i=0; i<gente.size(); i++){

                  MadamPerson persona = (MadamPerson) gente.get(i);
```

```
                  //le buscamos un amigo que esté más o menos cerca
de él
                  MadamPerson personaB = null;
                  int radioMedioAmistad = 40;
                  int factorAleatorio = (int)
(random.nextGaussian()*9); //algunos buscarán más cerca, otros más
lejos
                  int radioEfectivo = radioMedioAmistad +
factorAleatorio;
                  if (radioEfectivo <= 0) radioEfectivo =
radioMedioAmistad;
                  ArrayList<DemographicItem> genteCerca =
persona.peopleAround(radioEfectivo);
                  int cont = 0;
                  do {
                        //si con este radio no le alcanza para
conocer a nadie, busca un poco más lejos
                        int aux = radioEfectivo;
                        while (genteCerca.size() == 0){
                              aux++;
                              genteCerca = persona.peopleAround(aux);
                        }

                        personaB = (MadamPerson)
genteCerca.get(random.nextInt(genteCerca.size()));
                        cont++;
                  }
                  while ((persona == personaB) && (cont <=
genteCerca.size()));
                  addFriendshipLink(persona, personaB);
            }
            stats.reset(this);
            schedule.scheduleRepeating(stats);
      }
```

4. Datos de entrada externos

El proceso demográfico de la mortalidad viene determinado por la *Tabla de Mortalidad de la Población Española en los años 1990-1991*. Dicha tabla está elaborada por el INE, y recoge las tasas específicas de mortalidad por grupos de edad de 5 años de extensión, salvo para neonatos hasta un año de edad, que están en su propio grupo. Cada tasa específica se ha interpretado como la probabilidad que tiene una persona que esté en dicho grupo de edad de morir, mientras siga estando en ese grupo de edad.

5. Submodelos

1. Conocer gente

Para cada agente, dos veces al año, —cada 25 etapas de simulación— esta acción tiene lugar. La manera en que se produce es la siguiente:

Si el agente tiene más de 7 años, buscará un amigo. El proceso de búsqueda puede ser de dos maneras:

- Con probabilidad 0.3 buscará de esta manera: de entre la gente que está a un radio de 40 unidades, más un factor aleatorio de distancia (modelado con una normal de media cero y desviación típica 9), escogerá aleatoriamente a un individuo que **no** forme parte aun de su lista de amigos, en caso de que los haya.
- Con probabilidad complementaria 0.7, se buscará un amigo de entre los amigos de sus amigos: Se escoge al azar un amigo de su lista de amigos, y posteriormente se escoge al azar a uno de éstos.

2. Buscar pareja

Todos los agentes considerados adultos (con 16 años o más) y con no más de 60 años ejecutan la siguiente búsqueda de pareja. Al igual que antes, se produce 2 veces al año, o cada 25 etapas de simulación.

El agente A escoge, de entre sus amigos, a los de sexo contrario. A continuación, toma al azar a uno de ellos, póngase el agente B. Si cumple sus requisitos, A le propone matrimonio a B. El agente B responderá afirmativamente a la petición si el agente A cumple a su vez los requisitos del agente B.

Una persona cumplirá los requisitos de otra si el número de características que comparten en común es mayor o igual que su *nivel de exigencia*. Además, no puede haber una diferencia de edad de más de 8 años, y ningún menor de 16 puede casarse.

En caso de que una persona casada acepte una petición, o su petición sea aceptada, se divorciará de su actual pareja y formará un nuevo matrimonio.

Finalmente, se actualiza el *nivel de exigencia*. Esto tiene lugar para todos los agentes adultos (16 en adelante) y se produce de la siguiente manera:

- Si el agente está casado, el *nivel de exigencia* se mantendrá constante e igual al número de características en común que tiene con su actual pareja. Aun así, seguirá buscando parejas que puedan mejorar dicho nivel.
- Si el agente no está casado, reducirá su *nivel de exigencia* conforme a la siguiente regla: El nuevo nivel de exigencia será $nivel\ exigencia = ke^{-0.02 edadEnAños}$. De esta manera, el nivel se actualizará una vez cada año, pues depende únicamente de este valor entero.

3. Aumentar edad

El agente aumenta en una unidad su variable de estado *etapas de simulación*.

4. Tener hijos

Una vez cada dos años (o 100 etapas de simulación), los agentes de *género* mujer, tienen la posibilidad de tener un hijo. Esto se producirá si se dan las siguientes condiciones:

- La mujer está casada.
- La mujer está en edad fértil, que se considera entre 16 y 50 años.
- El número de hijos que tiene actualmente la mujer es menor que 3.

Si se dan las condiciones anteriores, se evaluará una probabilidad, que depende de la edad de la mujer, y en función de esa probabilidad se producirá el nacimiento o no. La probabilidad es $probabilidadNacimiento = e^{-0.0875 * edadEnAños}$.

5. Morir

En cada etapa de la simulación, se comprueba si una persona ha alcanzado ya la edad de morir. En tal caso, muere. Como se ha comentado, cada persona posee un atributo *edad de muerte* que determina la edad a la que una persona ha de morir. Sin embargo, ésta edad va cambiando de la siguiente manera.

En principio, a cada persona se le asigna una edad de morir lo suficientemente alta como para que no se alcance (100 años, o más).

Sin embargo, en cada etapa se comprueba también si una persona "entra" en un nuevo grupo de edad. En caso de que así sea, se evalúa la probabilidad que tendrá de morir en el intervalo de vida que abarca a dicho grupo. Si ha de morir en este intervalo de edad, se le asigna una edad aleatoria dentro del intervalo como edad de muerte. En caso contrario, sigue con su "edad alta" de muerte.

Adaptación del modelo a la arquitectura propuesta

El modelo se ha adaptado a la arquitectura propuesta de manera bastante natural, sobre todo el modelado de los aspectos demográficos y sociales, tanto los que vienen determinados como comportamientos de los agentes, como los que se determinan mediante datos, desde arriba. En cuanto al entorno —al contenedor de la simulación (subclase de *SimpleWorld*)— no se entrará en detalles ya que no presenta excesivo interés. No obstante, sí que se ha hecho pleno uso casi todas las características que contempla la arquitectura, como redes sociales locales y globales.

Cada agente está representado por un ejemplar de la clase *MadamPerson*, que es una instancia de *Person*. Por tanto, posee los atributos y métodos de ésta: tiene métodos para interactuar con otras personas al nivel de hacer amistades, establecer matrimonios y disolverlos.

Lo realmente interesante ha sido la manera en que se han definido y separado los elementos que determinan el comportamiento demográfico. Este modelo entra dentro de lo que se ha venido llamando "modelos mixtos", los nacimientos de nuevos agentes están modelados de manera un tanto intermedia: dependen de ciertos atributos de las mujeres, como el número de hijos que ya tenga o la edad de ésta. Pero al final dependen de una probabilidad sobre la fecundidad de las mujeres, que en este caso ha sido tomada de manera un tanto arbitraria para no complicar las cosas. Por otra parte, la dinámica social y de formación y disolución de parejas es plenamente ABM, basada en reglas de comportamiento de los agentes y en interacción entre ellos. Finalmente, el modelado de la mortalidad viene plenamente determinado desde arriba, ya que se ha modelado mediante datos estadísticos, como se ha explicado.

En primer lugar, para modelar las muertes se ha recurrido a unos de los módulos demográficos de los que se hablaba al describir la arquitectura: en concreto se ha instanciado la *interface Mortality*, de la manera que se puede ver en el siguiente código:

```
public class MortalityMadam implements Mortality {

	//SINGLETON

	private static MersenneTwisterFast random = new
MersenneTwisterFast();
	private static MortalityMadam INSTANCE = new MortalityMadam ();

	private MortalityMadam () {}

	public static MortalityMadam getInstance() {
		return INSTANCE;
	}

	@Override
	public boolean timeToDie(DemographicItem persona) {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public int ageOfDeath(DemographicItem persona) {
		int age = ((Person) persona).getAge();
		if (age == 1){
			if (random.nextBoolean(probabilidadMuerte(age))){
				return (age + random.nextInt(4));
			}
		}
		else
			if (random.nextBoolean(probabilidadMuerte(age))){
				return (age + random.nextInt(5));
```

```
                }
            return 100;
        }

        @Override
        public double crudeDeathRate() {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public int lifeExpectancy(DemographicItem person) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public double deathProbability(DemographicItem person) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public double ageSpecificDeathRate(int age) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public double sexSpecificDeathRate(boolean man) {
            // TODO Auto-generated method stub
            return 0;
        }

        @Override
        public double ageSexSpecificDeathRate(int age, boolean man) {
            // TODO Auto-generated method stub
            return 0;
        }


        // Tabla de mortalidad de la población española, años 1990-
1991. Fuente:
        // INE.
        public double probabilidadMuerte(int edad) {
            double pm;

            if (edad < 1)
                  pm = 0.00783;
            else if ((1 <= edad) && (edad < 5))
                  pm = 0.00182;
            else if ((5 <= edad) && (edad < 10))
                  pm = 0.00118;
            else if ((10 <= edad) && (edad < 15))
                  pm = 0.00128;
            else if ((15 <= edad) && (edad < 20))
                  pm = 0.00323;
            else if ((20 <= edad) && (edad < 25))
                  pm = 0.00511;
            else if ((25 <= edad) && (edad < 30))
                  pm = 0.00596;
```

```
            else if ((30 <= edad) && (edad < 35))
                  pm = 0.00628;
            else if ((35 <= edad) && (edad < 40))
                  pm = 0.00696;
            else if ((40 <= edad) && (edad < 45))
                  pm = 0.00963;
            else if ((45 <= edad) && (edad < 50))
                  pm = 0.01384;
            else if ((50 <= edad) && (edad < 55))
                  pm = 0.02316;
            else if ((55 <= edad) && (edad < 60))
                  pm = 0.03411;
            else if ((60 <= edad) && (edad < 65))
                  pm = 0.05197;
            else if ((65 <= edad) && (edad < 70))
                  pm = 0.08113;
            else if ((70 <= edad) && (edad < 75))
                  pm = 0.12843;
            else if ((75 <= edad) && (edad < 80))
                  pm = 0.21538;
            else if ((80 <= edad) && (edad < 85))
                  pm = 0.34711;
            else if ((85 <= edad) && (edad < 90))
                  pm = 0.51997;
            else if ((90 <= edad) && (edad < 95))
                  pm = 0.71682;
            else if ((95 <= edad) && (edad < 100))
                  pm = 0.89388;
            else
                  pm = 1.0;

            return pm;
      }
```

En cuanto a la fecundidad, se ha recurrido también a instanciar la *interface* correspondiente. En este caso, se ha encargado este módulo de dar la probabilidad de nacimiento, dada la edad de la madre, tal y como se explicaba al describir la arquitectura. Será pues el correspondiente comportamiento demográfico el que evalúe otros atributos de la madre como si está casada o tiene ya demasiados hijos para tomar la decisión de tener o no un hijo.

```
public class FertilityMadam implements Fertility {

      private static MersenneTwisterFast random = new
MersenneTwisterFast();


      public FertilityMadam() {}



      @Override
      public double birthProbability(DemographicItem female) {

            return Math.exp(-0.0875*female.getAge());
      }
```

```
	@Override
	public boolean newChild(DemographicItem female) {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public double birthProbability() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public double crudeBirthRate() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public double generalFertilityRate() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public int meanAgeOfMother() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public int meanNumberOfChildrenPerWoman() {
		return 3;
	}

	@Override
	public int meanTimeBetweenBirths() {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public double ageSpecificFertilityRate(int age) {
		return 0;
	}

	@Override
	public double orderSpecificFertilityRate(int order) {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public double femaleBirthProbability() {
		//suppose there are 106 male births every 100 female
births
		return 100.0/206.0;
	}
    }
```

Los comportamientos de los agentes, por otra parte, se han separado en los dos tipos elementales que propone la arquitectura: *BasicDemographicBehavior* y *BasicSocialBehavior*.

Además, aprovechando las diferentes maneras de actuar que se tienen en el problema, según se trate de adultos, niños, solteros o mujeres, se han definido una serie de variaciones de estos comportamientos. En realidad, y debido a la simplicidad del modelo, esto no resulta necesario, y podría parecer que es un simple "forzar las cosas". Sin embargo, se ha hecho para demostrar lo que se podría llegar a hacer en modelos más complejos, según las necesidades. Esto convierte al modelo en mucho más flexible y modular: Se pueden introducir nuevas variaciones de estos mismos comportamientos según las necesidades y se puede cambiar de comportamiento durante la ejecución de la simulación. De esta manera, se han definido dos tipos de comportamientos demográficos, que son instancias de *BasicBehaviorModule*:

- ***DemographicBehavior***: Es el comportamiento demográfico por defecto, para niños y niñas y hombres. Especifica los submodelos ***Aumentar edad, Tener hijos*** y ***Morir***. Evidentemente, los agentes a los que se dota de este comportamiento no pueden tener hijos, así que el método correspondiente a tener hijos se encarga de *checkear* en cada etapa de la simulación si el agente es mujer y alcanza los 16 años. En ese momento, cambia el módulo de comportamiento demográfico del actuar a uno de tipo *FemaleDemographicBehavior*.
- ***FemaleDemographicBehavior***: Es igual que el anterior, pero propio de mujeres en edad fértil. En este caso, el método que corresponde a tener hijos se encarga de lo que se comentó al describir el proceso ***Tener hijos***.

Por otra parte, se han definido tres tipos de comportamiento social, que extienden a *BasicSocialBehavior*:

- ***SingleSocialBehavior***: Este módulo de comportamiento es el que tienen adultos solteros. En él se definen los dos submodelos ***Conocer gente*** y ***Buscar pareja*** especificados anteriormente. Una vez que el método correspondiente a buscar pareja tiene éxito y el agente se casa, cambia este módulo por *MarriedSocialBehavior*.
- ***MarriedSocialBehavior***: Es similar al anterior, pero para agentes que estén casados. La única diferencia está en que no actualiza el *nivel de exigencia*, pues los agentes casados lo mantienen constante en el número de características que comparten con su actual pareja.
- ***ChildSocialBehavior***: Es el que se encarga de definir los submodelos propios del comportamiento social en los niños. Al igual que sucedía con el

comportamiento demográfico, los niños no buscan pareja. En lugar de eso, el método correspondiente se encarga de comprobar la edad que tiene el niño y, al alcanzar los 16 años, cambia de módulo de comportamiento.

Como se puede observar, resulta bastante natural definir así los comportamientos, además de resultar sencillo intercambiarlos según las necesidades, como se comentaba.

REFERENCIAS

[1] T. K. Burch, «Data, Models, Theory and Reality: The Structure of Demographic Knowledge», in *Agent-Based Computational Demography*, F. C. Billari y A. Prskawetz, Eds. Physica-Verlag HD, 2003, pp. 19–40.

[2] T. K. Burch, «Demography in a new key», *Demographic Research*, vol. 9, pp. 263–284, dic. 2003.

[3] F. C. Billari y A. Prskawetz, *Agent-Based Computational Demography: Using Simulation to Improve Our Understanding of Demographic Behaviour*. Springer, 2003.

[4] R. Axtell, J. Epstein, J. Dean, G. Gumerman, A. Swedlund, J. Harburger, S. Chakravarty, R. Hammond, J. Parker, y M. Parker, «Population growth and collapse in a multiagent model of the Kayenta Anasazi in Long House Valley», *Proceedings of the National Academy of Sciences*, vol. 99, n°. 90003, pp. 7275–7279, may 2002.

[5] J. S. Dean, G. J. Gumerman, J. M. Epstein, R. Axtell, A. C. Swedlund, M. T. Parker, y S. McCarroll, «Understanding Anasazi Culture Change Through Agent-Based Modeling», Santa Fe Institute, Working Paper 98-10-094, 1998.

[6] G. J. Gumerman, A. C. Swedlund, J. S. Dean, y J. M. Epstein, «The evolution of social behavior in the prehistoric American southwest», *Artif. Life*, vol. 9, n°. 4, pp. 435–444, 2003.

[7] M. A. Janssen, «Understanding Artificial Anasazi», 31-oct-2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/4/13.html. [Accessed: 30-ago-2012].

[8] V. Grimm, U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, A. Huth, J. U. Jepsen, C. Jørgensen, W. M. Mooij, B. Müller, G. Pe'er, C. Piou, S. F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R. A. Stillman, R. Vabø, U. Visser, y D. L. DeAngelis, «A standard protocol for describing individual-based and agent-based models», *Ecological Modelling*, vol. 198, n°. 1–2, pp. 115–126, sep. 2006.

[9] V. Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, y S. F. Railsback, «The ODD protocol: A review and first update», *Ecological Modelling*, vol. 221, n°. 23, pp. 2760–2768, nov. 2010.

[10] M. Livi-Bacci, *Introducción a la Demografía*. Editorial Ariel, 1993.

[11] J. Vinuesa, J. V. Angulo, y F. Z. López, *Demografía: Análisis y Proyecciones*. Síntesis, 1994.

[12] «Qué es la demografía», *Apuntes de demografía*. [Online]. Available: http://apuntesdedemografia.wordpress.com/curso-de-demografia/que-es-la-demografia/. [Accessed: 21-ago-2012].

[13] J. Vallin, *La Demografía*. Alianza Editorial, 1995.

[14] H. S. Shryock, J. S. Siegel, y E. A. Larmon, *The methods and materials of demography*, vol. 2. U.S. Dept. of Commerce, Bureau of the Census : for sale by the Supt. of Docs. U.S. Govt. Print. Off., 1980.

[15] «Instituto Nacional de Estadística. (National Statistics Institute)». [Online]. Available: http://www.ine.es/. [Accessed: 21-ago-2012].

[16] «Profes.net». [Online]. Available: http://www.gh.profes.net/. [Accessed: 04-sep-2012].

[17] M. Kalmijn, «Intermarriage and Homogamy: Causes, Patterns, Trends», *Annual Review of Sociology*, vol. 24, nº. 1, pp. 395–421, 1998.

[18] F. C. Billari, T. Fent, A. Prskawetz, y J. Scheffran, «Agent-Based Computational Modelling: An Introduction», in *Agent-Based Computational Modelling*, F. C. Billari, T. Fent, A. Prskawetz, y J. Scheffran, Eds. Physica-Verlag HD, 2006, pp. 1–16.

[19] N. Gilbert y K. G. Troitzsch, *Simulation for the Social Scientist*. McGraw-Hill International, 2005.

[20] R. Axelrod, «Advancing the art of simulation in the social sciences», *Complexity*, vol. 3, nº. 2, pp. 16–22, 1997.

[21] P. Davidsson, J. Holmgren, H. Kyhlbäck, D. Mengistu, y M. Persson, «Applications of agent based simulation», in *Proceedings of the 2006 international conference on Multi-agent-based simulation VII*, Berlin, Heidelberg, 2007, pp. 15–27.

[22] A. López-Paredes y J. Pavón, «Agent Based Simulation for the Study of Complex Social Systems», *Economy Informatics*, vol. 9, nº. 1, pp. 22–28, sep. 2009.

[23] M. W. Macy y R. Willer, «FROM FACTORS TO ACTORS: Computational Sociology and Agent-Based Modeling», *Annual Review of Sociology*, vol. 28, nº. 1, pp. 143–166, 2002.

[24] C. W. Reynolds, «Flocks, herds and schools: A distributed behavioral model», in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1987, pp. 25–34.

[25] D. A. Bennett y W. Tang, «Modelling adaptive, spatially aware, and mobile agents: Elk migration in Yellowstone», *International Journal of Geographical Information Science*, vol. 20, nº. 9, pp. 1039–1066, 2006.

[26] F. Amigoni, M. Dini, N. Gatti, y M. Somalvico, «Anthropic agency: a multiagent system for physiological processes», *Artificial Intelligence in Medicine*, vol. 27, nº. 3, pp. 305–334, mar. 2003.

[27] J. Martínez-Miranda y J. Pavón, «An Agent-Based Simulation Tool to Support Work Teams Formation», in *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, vol. 50, J. M. Corchado, S. Rodríguez, J. Llinas, y J. M. Molina, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 80–89.

[28] R. M. Crowder, M. A. Robinson, H. P. N. Hughes, y Y.-W. Sim, «The Development of an Agent-Based Modeling Framework for Simulating Engineering Team Work», *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. PP, nº. 99, pp. 1 –15, 2012.

[29] K. K. E. Kafeza, «Speeding up electronic commerce activities using CapBasED-AMS», pp. 192 – 199, 2000.

[30] S. El hadouaj, A. Drogoul, y S. Espié, «How to Combine Reactivity and Anticipation: The Case of Conflicts Resolution in a Simulated Road Traffic», in *Multi-Agent-Based Simulation*, vol. 1979, S. Moss y P. Davidsson, Eds. Springer Berlin / Heidelberg, 2001, pp. 157–167.

[31] D. Blanco-Moreno, R. Fuentes-Fernandez, y J. Pavon, «Simulation of Online Social Networks with Krowdix», in *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, 2011, pp. 13 –18.

[32] D. Liu y X. Chen, «Rumor Propagation in Online Social Networks Like Twitter – A Simulation Study», in *2011 Third International Conference on Multimedia Information Networking and Security (MINES)*, 2011, pp. 278 –282.

[33] J. Pascual, J. Pajares, y A. López-Paredes, «Explaining the Statistical Features of the Spanish Stock Market from the Bottom-Up», in *Advances in Artificial Economics*, vol. 584, C. Bruun, Ed. Springer Berlin Heidelberg, 2006, pp. 283–294.

[34] M. Franke, A. Geyer-Schulz, y B. Hoser, «On the Analysis of Asymmetric Directed Communication Structures in Electronic Election Markets», in *Agent-Based Computational Modelling*, F. C. Billari, T. Fent, A. Prskawetz, y J. Scheffran, Eds. Physica-Verlag HD, 2006, pp. 37–59.

[35] J. M. Galán, A. López-Paredes, y R. del Olmo, «An agent-based model for domestic water management in Valladolid metropolitan area», *Water Resources Research*, vol. 45, n^{o}. 5, may 2009.

[36] R. B. Matthews, N. G. Gilbert, A. Roach, J. G. Polhill, y N. M. Gotts, «Agent-based land-use models: a review of applications», *Landscape Ecology*, 2007. [Online]. Available: http://epubs.surrey.ac.uk/1598/. [Accessed: 15-ago-2012].

[37] S. Hassan, L. Antunes, y J. Pavón, «Mentat: A Data-Driven Agent-Based Simulation of Social Values Evolution», in *Multi-Agent-Based Simulation X*, vol. 5683, G. Di Tosto y H. Van Dyke Parunak, Eds. Springer Berlin / Heidelberg, 2010, pp. 135–146.

[38] H. Kim y P. S. Bearman, «The structure and dynamics of movement participation», *American sociological review*, vol. 62, n^{o}. 1, pp. 70–93.

[39] B. Edmonds y S. Moss, «From KISS to KIDS – An 'Anti-simplistic' Modelling Approach», in *Multi-Agent and Multi-Agent-Based Simulation*, vol. 3415, P. Davidsson, B. Logan, y K. Takadama, Eds. Springer Berlin / Heidelberg, 2005, pp. 130–144.

[40] S. Hassan, L. Antunes, y M. Arroyo, «Deepening the Demographic Mechanisms in a Data-Driven Social Simulation of Moral Values Evolution», in *Multi-Agent-Based Simulation IX*, vol. 5269, N. David y J. S. Sichman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 167–182.

[41] T. Schelling, «Models of Segregation», 1969. [Online]. Available: http://www.rand.org/pubs/research_memoranda/RM6014.html. [Accessed: 15-ago-2012].

[42] T. C. Schelling, «Dynamic models of segregation†», *The Journal of Mathematical Sociology*, vol. 1, n^{o}. 2, pp. 143–186, 1971.

[43] J. M. Epstein y R. L. Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press, 1996.

[44] C. Cioffi-Revilla, «A Methodology for Complex Social Simulations», 31-ene-2010. [Online]. Available: http://jasss.soc.surrey.ac.uk/13/1/7.html. [Accessed: 16-ago-2012].

[45] «Altreva - Agent-based financial market modeling and trading software for forecasting stocks and other securities». [Online]. Available: http://www.altreva.com/. [Accessed: 17-ago-2012].

[46] «MASON Multiagent Simulation Toolkit». [Online]. Available: http://cs.gmu.edu/~eclab/projects/mason/. [Accessed: 17-ago-2012].

[47] «Repast Suite». [Online]. Available: http://repast.sourceforge.net/. [Accessed: 17-ago-2012].

[48] «SwarmWiki». [Online]. Available: http://www.swarm.org/index.php/Main_Page. [Accessed: 17-ago-2012].

[49] «Agent iSolutions». [Online]. Available: http://www.agentisolutions.com/index.htm. [Accessed: 17-ago-2012].

[50] «NetLogo Home Page». [Online]. Available: http://ccl.northwestern.edu/netlogo/. [Accessed: 17-ago-2012].

[51] «SeSAm - Integrated Environment for Multi-Agent Simulation». [Online]. Available: http://www.simsesam.de/. [Accessed: 17-ago-2012].

[52] «AgentSheets». [Online]. Available: http://www.agentsheets.com/. [Accessed: 17-ago-2012].

[53] N. Gilbert, «Platforms and methods for agent-based modeling», *Proceedings of the National Academy of Sciences*, vol. 99, n°. 90003, pp. 7197–7198, may 2002.

[54] C. N. and G. Madey, «Tools of the Trade: A Survey of Various Agent Based Modeling Platforms», 31-mar-2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/2/2.html. [Accessed: 17-ago-2012].

[55] S. F. Railsback, S. L. Lytinen, y S. K. Jackson, «Agent-based Simulation Platforms: Review and Development Recommendations», *SIMULATION*, vol. 82, n°. 9, pp. 609–623, ene. 2006.

[56] J. Star y J. E. Estes, *Geographic information systems: an introduction*. Prentice Hall, 1990.

[57] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, y G. Balan, «MASON: A Multiagent Simulation Environment», *SIMULATION*, vol. 81, n°. 7, pp. 517–527, ene. 2005.

[58] Wikipedia contributors, «Logo (programming language)», *Wikipedia, the free encyclopedia*. Wikimedia Foundation, Inc., 16-ago-2012.

[59] E. Silverman, J. Bijak, y J. Noble, «Feeding the beast: Can computational demographic models free us from the tyranny of data?», 2011. [Online]. Available: http://eprints.soton.ac.uk/272839/. [Accessed: 03-sep-2012].

[60] «IMA - International Microsimulation Association». [Online]. Available: http://ima.natsem.canberra.edu.au/. [Accessed: 17-ago-2012].

[61] M. Birkin y B. Wu, «A Review of Microsimulation and Hybrid Agent-Based Approaches», in *Agent-Based Models of Geographical Systems*, A. J. Heppenstall, A. T. Crooks, L. M. See, y M. Batty, Eds. Springer Netherlands, 2012, pp. 51–68.

[62] E. Morand, L. Toulemon, S. Pennec, R. Baggio, y F. Billari, «Demographic modelling: the state of the art», *SustainCity Working Paper, 2.1a, Ined, Paris.*, 2010.

[63] «Cube Dynasim 4». [Online]. Available: http://www.citilabs.com/dynasim/. [Accessed: 17-ago-2012].

[64] C. O'Donoghue, «Dynamic Microsimulation: A Methodological Survey», *Brazilian Electronic Journal of Economics*, vol. 4, n°. 2, 2001.

[65] M. Favreault, «A Primer on the Dynamic Simulation of Income Model (DYNASIM3)», 01-feb-2004. [Online]. Available: http://www.urban.org/publications/410961.html. [Accessed: 20-ago-2012].

[66] R. J. Morrison, «DYNACAN (Longitudinal Dynamic Microsimulation Model)», vol. International Symposia in Economic Theory and Econometrics, n°. 16, pp. 461–465, feb. 2007.

[67] N. M. Stolen, «MOSART (Dynamic Cross-Sectional Microsimulation Model)», vol. International Symposia in Economic Theory and Econometrics, n°. 16, pp. 433–437, feb. 2007.

[68] O. Sundberg, «SESIM (Longitudinal Dynamic Microsimulation Model)», vol. International Symposia in Economic Theory and Econometrics, n°. 16, pp. 453–460, feb. 2007.

[69] S. Kelly, «DYNAMOD», vol. International Symposia in Economic Theory and Econometrics, n°. 16, pp. 439–442, feb. 2007.

[70] F. C. Billari, F. Ongaro, y A. Prskawetz, «Introduction: Agent-Based Computational Demography», in *Agent-Based Computational Demography*, F. C. Billari y A. Prskawetz, Eds. Physica-Verlag HD, 2003, pp. 1–17.

[71] S. Hassan Collado, «Towards a data-driven approach for Agent-Based Modelling: simulating Spanish postmodernisation», Universidad Complutense de Madrid, Madrid, 2009.

[72] I. Benenson, I. Omer, y E. Hatna, «Entity-based modeling of urban residential dynamics: the case of Yaffo, Tel Aviv», *Environment and Planning B: Planning and Design*, vol. 29, n°. 4, pp. 491 – 512, 2002.

[73] T. H. and P. Todd, «Population Heterogeneity and Individual Differences in an Assortative Agent-Based Marriage and Divorce Model (MADAM) Using Search with Relaxing Expectations», 31-oct-2008. [Online]. Available: http://jasss.soc.surrey.ac.uk/11/4/5.html. [Accessed: 19-ago-2012].

[74] P. M. Todd y F. C. Billari, «Population-Wide Marriage Patterns Produced by Individual Mate-Search Heuristics», in *Agent-Based Computational Demography*, F. C. Billari y A. Prskawetz, Eds. Physica-Verlag HD, 2003, pp. 117–137.

[75] F. Billari, T. Fent, A. Prskawetz, y B. Aparicio Diaz, «The "Wedding-Ring". An agent-based marriage model based on social interaction», *Demographic Research*, vol. 17, pp. 59–82, ago. 2007.

[76] M. G. A. Huigen, K. P. Overmars, y W. T. de Groot, «Ecology and Society: Multiactor Modeling of Settling Decisions and Behavior in the San Mariano Watershed, the Philippines: a First Application with the MameLuke Framework», *Ecology and Society*, vol. 11, n°. 2, 2006.

[77] L. An, M. Linderman, J. Qi, A. Shortridge, y J. Liu, «Exploring Complexity in a Human–Environment System: An Agent-Based Spatial Model for Multidisciplinary and Multiscale Integration», *Annals of the Association of American Geographers*, vol. 95, n°. 1, pp. 54–79, 2005.

[78] T. A. Kohler, J. Kresl, C. van West, E. Carr, y R. H. Wilshusen, «Be there then: a modeling approach to settlement determinants and spatial efficiency among late

ancestral pueblo populations of the Mesa Verde region, U.S. southwest», in *Dynamics in human and primate societies*, Oxford University Press, 2000, pp. 145–178.

[79] «MASS - Project Overview». [Online]. Available: http://oi.uchicago.edu/OI/PROJ/MASS/introduction.htm. [Accessed: 20-ago-2012].

[80] T. J. Wilkinson, J. H. Christiansen, J. Ur, M. Widell, y M. Altaweel, «Urbanization within a Dynamic Environment: Modeling Bronze Age Communities in Upper Mesopotamia», *American Anthropologist*, vol. 109, nº. 1, pp. 52–68, 2007.

[81] A. J. Heppenstall, A. T. Crooks, L. M. See, y M. Batty, Eds., *Agent-Based Models of Geographical Systems*, 2012.ª ed. Springer, 2011.

[82] S. Luke, *Multiagent Simulation and the MASON Library*, First Edition. Online version 1.0. Department of Computer Science. George Mason University, 2011.

[83] S. Robinson, *Simulation: The Practice of Model Development and Use*. John Wiley & Sons, 2004.

[84] M. Matsumoto y T. Nishimura, «Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator», *ACM Trans. on Modeling and Computer Simulation*, vol. 8, nº. 1, pp. 3–30, 1998.

[85] V. Grimm y S. F. Railsback, *Individual-based Modeling And Ecology*. Princeton University Press, 2005.

[86] B. Breckling, «Individual-based modelling: potentials and limitations», *ScientificWorldJournal*, vol. 2, pp. 1044–1062, abr. 2002.

[87] «UNSD // United Nations Statistics Division - Demographic and Social Statistics». [Online]. Available: http://unstats.un.org/unsd/demographic/. [Accessed: 27-ago-2012].

[88] R. Sun, «Cognitive Architectures and Multi-agent Social Simulation», in *Multi-Agent Systems for Society*, vol. 4078, D. Lukose y Z. Shi, Eds. Springer Berlin / Heidelberg, 2009, pp. 7–21.

[89] N. Gilbert, «When does social simulation need cognitive models?», 2006. [Online]. Available: http://www.cambridge.org. [Accessed: 28-ago-2012].

[90] *Patrones de Diseño: Elementos de software orientado a objetos reutilizable*, Reimp. Madrid: Addison-Wesley, 2006.

[91] S. J. Alam y A. Geller, «Networks in Agent-Based Social Simulation», in *Agent-Based Models of Geographical Systems*, A. J. Heppenstall, A. T. Crooks, L. M. See, y M. Batty, Eds. Springer Netherlands, 2012, pp. 199–216.

[92] A. F. Josep M. Pujol, «How Can Social Networks Ever Become Complex? Modelling the Emergence of Complex Networks from Local Social Exchanges», 31-oct-2005. [Online]. Available: http://jasss.soc.surrey.ac.uk/8/4/12.html. [Accessed: 28-ago-2012].

[93] B. Edmonds, «How Are Physical and Social Spaces Related? — Cognitive Agents as the Necessary "Glue"», in *Agent-Based Computational Modelling*, F. C. Billari, T. Fent, A. Prskawetz, y J. Scheffran, Eds. Physica-Verlag HD, 2006, pp. 195–214.

[94] L. H. and N. Gilbert, «Social Circles: A Simple Structure for Agent-Based Social Network Models», 31-mar-2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/2/3.html#Thiriot%202008. [Accessed: 28-ago-2012].

[95] J. Sabater y C. Sierra, «Reputation and social network analysis in multi-agent systems», in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, Bologna, Italy, 2002, pp. 475–482.

[96] N. Gilbert, «Agent-based social simulation: dealing with complexity», 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.4004&rep=rep1&type=pdf. [Accessed: 28-ago-2012].

[97] U. Wilensky y W. Rand, «Making Models Match: Replicating an Agent-Based Model», *Journal of Artificial Societies and Social Simulation*, vol. 10, n°. 4, p. 2, oct. 2007.

[98] K. R. Popper, *The logic of scientific discovery*. Hutchinson, 1959.

[99] B. H. Edmonds, «Replication, Replication and Replication: Some Hard Lessons from Model Alignment», 31-oct-2003. [Online]. Available: http://jasss.soc.surrey.ac.uk/6/4/11.html. [Accessed: 07-sep-2012].

[100] J. Rouchier, C. Cioffi-Revilla, J. G. Polhill, y K. Takadama, «Progress in Model-To-Model Analysis», 31-mar-2008. [Online]. Available: http://jasss.soc.surrey.ac.uk/11/2/8.html. [Accessed: 07-sep-2012].

[101] A. C. R. Martins, «Replication in the Deception and Convergence of Opinions Problem», 31-oct-2008. [Online]. Available: http://jasss.soc.surrey.ac.uk/11/4/8.html. [Accessed: 07-sep-2012].

[102] J. G. Polhill, «ODD Updated», *JASSS*, vol. 13, n°. 4, p. 9, 2010.

[103] «Artificial Anasazi | Open Agent Based Modeling Consortium». [Online]. Available: http://www.openabm.org/model/2222/version/1/view. [Accessed: 30-ago-2012].

[104] «ArtificialAnasaziDataFiles». [Online]. Available: http://code.google.com/p/tfm-sim-social-agentes/downloads/detail?name=dataFiles.rar&can=2&q=#makechanges. [Accessed: 30-ago-2012].

[105] «NOAA's Palmer Drought Severity Index». [Online]. Available: http://www.drought.noaa.gov/palmer.html. [Accessed: 30-ago-2012].