Sistemas Informáticos
Curso 2006 - 2007

Evolución del consumo de la energía eléctrica

Simulación con agentes de fenómenos sociales

Gabriel Aranda López
Ainhoa Borobia Aldanondo
Laura Vázquez Rodríguez

Dirigido por:
Profesor: Juan Pavón Mestras
Departamento: Ingeniería del Software e Inteligencia Artificial

Facultad de Informática
Universidad Complutense de Madrid

Concesión de derechos

Los abajo firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria como la documentación y/o el prototipo desarrollado.

Gabriel Aranda López	Ainhoa Borobia Aldanondo	Laura Vázquez Rodríguez

En Madrid, a 6 de Julio de 2007.

1. AGRADECIMIENTOS 4

2. RESUMEN DEL PROYECTO 5

3. INTRODUCCIÓN 6

3.1 UN POCO DE HISTORIA 6
3.2 MAS 7
3.3 MOTIVACIÓN 9
3.4 OBJETIVOS 10
3.5 MÉTODO DE TRABAJO 11
3.6 ESTRUCTURA DEL DOCUMENTO 12

4. DESCRIPCIÓN DEL ESTADO DEL ARTE 13

4.1 INTRODUCCIÓN 13
4.2 TRABAJOS ANTERIORES 13
4.3 HERRAMIENTAS 19

5. REPAST 28

5.1 IMPLEMENTACIÓN DEL FRAMEWORK REPAST APLICADO A NUESTRO PROYECTO 28

6. ARQUITECTURA GENÉRICA DEL SISTEMA 33

6.1 MODELO VISTA CONTROLADOR 33
6.2 EXPLICACIÓN DEL MODELO GENÉRICO 34
6.3 UML 44
6.4 DETALLES DE IMPLEMENTACIÓN 54
6.5 ESCENARIO1: EVOLUCIÓN DEL CONSUMO AFECTADO POR CAMPAÑAS PUBLICITARIAS. 55
6.6 ESCENARIO2: FACTORES AMBIENTALES QUE MODIFICAN EL CONSUMO. 63
6.7 ESCENARIO3: FACTORES DEMOGRÁFICOS QUE AFECTAN AL CONSUMO. 67

7. CONCLUSIONES Y POSIBLES AMPLIACIONES 75

8. GLOSARIO DE TÉRMINOS 77

9. BIBLIOGRAFÍA 78

1. Agradecimientos

Queremos agradecer a Juan Pavón, nuestro director de proyecto, su amabilidad e inestimable ayuda en la realización de nuestro proyecto.

A nuestros familiares y amigos, por su apoyo y el cariño recibido.

Cada uno de nosotros queríamos agradecer a nuestros compañeros de proyecto el estar ahí en los momentos difíciles. También a Gabriel en particular le gustaría hacer mención a su madre, Araceli y a Rubén.

2. Resumen del proyecto

Castellano

Nuestro proyecto consiste en el desarrollo de un modelo de la evolución temporal del consumo eléctrico de una población utilizando un sistema multiagente. Se ha realizado un estudio desde distintos puntos de vista de varios aspectos del consumo eléctrico. Los aspectos que hemos estudiado en los distintos escenarios desarrollados han sido: la evolución del consumo debido a la influencia de campañas publicitarias y del comportamiento del vecindario; la evolución del consumo debido al cambio estacional y la evolución del consumo debido al movimiento de población.

Este proyecto ha sido implementado utilizando las librerías que ofrece la herramienta de simulación de sistemas multiagente Repast. El usuario puede configurar los datos de entrada a modelos de comportamiento de consumo eléctrico para agentes que representan familias. Como resultado, nuestro sistema muestra estadísticas sobre varios patrones de consumo dependientes de diferentes tipos de factores.

PALABRAS CLAVE: agentes software, sistemas multiagente, consumo eléctrico, simulación social, Repast.

English

This project is about of a model based on temporal evolution of a population's electrical consumption, using a multiagent system. We have studied differents points of view of the electrical consumption. These aspects, studied in the scenes developed, have been: the electrical consumption evolution caused by marketing, advertising influence and also manners or customs of neighbours in a community; the evolution of consumption due to seasons and finally the evolution of consumption caused by population migration.

This project is implemented using the libraries provided by the simulation of multiagent systems tool, Repast. Users may configure the input data to the models of electrical consumption patterns, where each agent represents a family. As a result, the system shows statistics about some consumption patterns, which deppend on different kinds of facts.

KEY WORDS : software agents, multiagent systems , electrical consumption, social simulation, Repast.

3. Introducción

Para comenzar este proyecto queríamos enmarcar en el área de la simulación la aparición de los sistemas multiagente como herramienta de modelado de sistemas complejos así como hacer una breve introducción general a los conceptos básicos relacionados con dichos sistemas. Después queríamos describir la motivación y objetivos que nos han hecho llevar a cabo este proyecto así como la línea de trabajo seguida en éste y su distribución por capítulos.

3.1 Un poco de historia

El primer impulso a la simulación de fenómenos reales sucedió en los años 70, con los primeros intentos de simular con un ordenador la evolución de ciertas variables en el tiempo. El ejemplo más conocido es el Informe Meadows realizado para el Club de Roma. Más tarde, en el campo de la sociología, Axelrod utilizó la simulación con ordenador para tratar de establecer la estrategia más racional en un contexto sin normas previas y sin una autoridad que pueda aplicar sanciones. Los resultados de su *'concurso computerizado'* de estrategias en torno al '*dilema del prisionero'* generaron un material inestimable para la reflexión teórica, continuado más tarde con una aplicación al sistema de relaciones internacionales.

Ya en los años noventa, la aplicación de sistemas multiagente vinculada al desarrollo de la Inteligencia Artificial, abre nuevas puertas a un vasto repertorio de herramientas para simular procesos complejos.

El propósito de la simulación social, como el de las ecuaciones estadísticas o el de los tipos ideales, es el de generar modelos de la realidad que permitan construir o validar teorías acerca de sus regularidades. Lo que hace de la simulación una estrategia analítica novedosa no son los propósitos, sino los medios: algoritmos de programación que contienen las reglas de actuación de unos agentes que interactúan entre ellos y con el entorno.

Las ventajas de la simulación social no son sólo experimentales sino también, y sobre todo, teóricas: sus modelos permiten encontrar los mecanismos de evolución del sistema, lo cual es impracticable con modelos de análisis estadístico.

Debido a la inexistencia de fundamentos teóricos para afirmar que un modelo con el mismo número de agentes que individuos observados en un sistema social de gran tamaño se comportará de la misma forma que un modelo con unos pocos agentes es importante la posibilidad de inclusión en la simulación de un gran número de agentes, lo que es posible con las tecnologías actualmente disponibles. De este modo podemos conseguir un modelo complejo en el que estén representados gran número de individuos así como un número elevado de variables representativas de las características de dichos individuos y de las interacciones entre sí.

Concretamente, la simulación basada en agentes inteligentes permite aprovechar eficientemente los avances descritos: si cada individuo es un agente distinto con sus características propias, por definición de agente perseguirá sus propios objetivos con decisiones

racionales (I); además, si el modelo trabaja con un gran número de agentes que interactúen entre sí, surgirán las dinámicas de grupo que se pretenden estudiar (II), precisamente la visión global que permite esta tecnología; por último, las características de los agentes (III) se adaptan muy bien a este problema, tanto por su independencia (como los individuos) y flexibilidad (con posibilidad de aprendizaje y evolución) como por su percepción y actuación en su entorno (colectivos) y otros agentes (individuos).

Así, es posible modelar una gran agrupación de individuos, lo suficientemente grande como para que pudieran surgir "espontáneamente" en su interior, colectivos con integrantes de determinadas afinidades comunes. Es decir, en simulación social basada en agentes describimos características micro para cada agente de manera que en la simulación emerjan las características macro de las poblaciones.

3.2 MAS

El dominio de los sistemas multiagente (MAS) o de la inteligencia artificial distribuida es una ciencia y una técnica que trata con los sistemas de inteligencia artificial en red.

El bloque fundamental de construcción de un sistema multiagente, como es de esperarse, son los agentes.

Aunque no existe una definición formal y precisa de lo que es un agente, éstos son por lo general vistos como entidades inteligentes, equivalentes en términos computacionales a un proceso del sistema operativo, que existen dentro de cierto contexto o ambiente, y que se pueden comunicar a través de un mecanismo de comunicación entre procesos, usualmente un sistema de red, utilizando protocolos de comunicación.

En cierto modo, un sistema multiagente es un sistema distribuido en el cual los nodos o elementos son sistemas de inteligencia artificial, o bien un sistema distribuido donde la conducta combinada de dichos elementos produce un resultado en conjunto inteligente.

Hay que notar que los agentes no son necesariamente inteligentes. Existen como en todo el resto del dominio de la inteligencia artificial, dos enfoques para construir sistemas multiagente:

- El enfoque formal o clásico, que consiste en dotar de los agentes de la mayor inteligencia posible utilizando descripciones formales del problema a resolver y de hacer reposar el funcionamiento del sistema en tales capacidades cognitivas. Usualmente la inteligencia es definida utilizando un sistema formal (por ejemplo, sistemas de inferencia lógica) para la descripción, raciocinio, inferencia de nuevo conocimiento y planificación de acciones a realizar en el medio ambiente.

- El enfoque constructivo, que persigue la idea de brindarle inteligencia al conjunto de todos los agentes, para que a través de mecanismos ingeniosamente elaborados de interacción, el sistema mismo genere comportamiento inteligente que no necesariamente estaba planeado desde un principio o definido dentro de los agentes mismos (que pueden ser realmente simples). Este tipo de conducta es habitualmente llamado comportamiento emergente. Normalmente nos

interesan aquellos MAS que poseen un comportamiento emergente que satisface ciertas propiedades predefinidas.

En general, los sistemas multiagente aplican una mezcla de técnicas: la orientación a objetos para implementar la estructura del sistema; los sistemas expertos para definir el comportamiento de cada agente y de la sociedad formada por todos ellos y para representar el conocimiento y la inteligencia artificial distribuida para la comunicación del conocimiento y la organización y distribución de los agentes.

Existen metodologías de ingeniería de agentes. Algunas de ellas intentan aplicar métodos de ingeniería del software convencionales a sistemas de agentes, extendiéndolos, como Agent UML o MESSAGE y otras intentan desarrollarse a partir de alguna metodología de inteligencia artificial (AI) o de los sistemas basados en el conocimiento (KBS), como MAS-CommonKADS o KAOS.

Se suelen utilizar agentes en situaciones con comunicaciones complejas y diversas; sistemas en los que no es práctico o posible especificar el comportamiento por reglas (el comportamiento de los agentes está guiado por sus objetivos); sistemas de negociación, cooperación y competencia entre distintas entidades y en sistemas autónomos.

De manera más concreta podemos definir un sistema multiagente, más conocido como MAS, como un conjunto de agentes autónomos actuando independientemente, es decir, usando sólo información local. De modo que, opcionalmente se pueden comunicar unos agentes con otros.

Los agentes perciben señales del entorno en donde están, cambios en el entorno debidos a acciones de otros agentes y comunicaciones entre agentes. Los agentes actúan tras un ciclo de razonamiento en el cual hay una fase de razonamiento local y otra de razonamiento global y un posible intercambio social.

La organización de la sociedad formada por los agentes del MAS tiene que ser modelada de manera descentralizada, de modo que podemos distinguir dos fases de modelado:

- Fase de modelado del individuo o agente: Los agentes realizan tareas o persiguen objetivos. Y tienen responsabilidades, control y un estado mental del agente. Hay que modelar el problema a partir del conocimiento específico en el ámbito en que trabajemos.

- Fase de modelado de la sociedad: consistente en modelar la estructura social de dependencias entre agentes (grupos potenciales); modelar los grupos dinámicos de acuerdo con relaciones sociales (agentes colaboradores, competidores, organizados); la coordinación de los modelos para llegar a los objetivos generales deseados y el método general para la interacción entre agentes.

Una vez hemos definido los posibles objetivos y las tareas globales que serán llevadas a cabo por la sociedad de agentes, queda por identificar los tipos de relaciones sociales que se pueden presentar en nuestro modelo:

- Qué tareas requerirán competitividad entre agentes.

- Qué tareas podrían interferir con otras tareas.
- Qué tareas requerirán cooperación entre agentes.

De manera que el modelo debe incluir para cada tipo de tarea:
- El conjunto de normas que guiarán el comportamiento del agente para dicho tipo de interacción.
- Las reglas que garanticen la consistencia general.

Por último hay que hacer un modelo del entorno en el que definamos las entidades y relaciones del entorno con los agentes del sistema.

3.3 Motivación

En esta memoria presentamos un sistema multiagente para simulación social con el objetivo del estudio del consumo eléctrico de una sociedad. En realidad, el resultado de dicho proyecto ha sido el estudio del consumo eléctrico debido a tres factores diferentes: ambientales, publicitarios y de inmigración/emigración. Todos estos modelos tienen la finalidad de predecir las futuras características macro del consumo eléctrico.

El suministro de energía eléctrica es esencial para el funcionamiento de nuestra sociedad. Su precio es un factor decisivo de la competitividad de buena parte de nuestra economía. El desarrollo tecnológico de la industria eléctrica y su estructura de aprovisionamiento de materias primas determinan la evolución de otros sectores de la industria.

Por otra parte, el transporte y la distribución de electricidad constituyen un monopolio natural: se trata de una actividad intensiva en capital, que requiere conexiones directas con los consumidores, cuya demanda de un producto no almacenable -como la energía eléctrica- varía en períodos relativamente cortos de tiempo.

Además, la imposibilidad de almacenar electricidad requiere que la oferta sea igual a la demanda en cada instante de tiempo, lo que supone necesariamente una coordinación de la producción de energía eléctrica, así como la coordinación entre las decisiones de inversión en generación y en transporte de energía eléctrica.

Por lo tanto, la predicción del consumo eléctrico es una pieza clave para la rentabilidad de la industria de energía eléctrica. Esto es, debido a la naturaleza de la energía eléctrica que es tal que no se puede almacenar; de modo que, toda aquella energía eléctrica que no se consume a continuación de ser producida, se pierde. Por ello, es importante un ajuste óptimo entre oferta y demanda eléctricas. De hecho Red Eléctrica Española, que es el organismo español encargado de la red de transporte de la electricidad desde productores hasta consumidores, dedica un gran esfuerzo a la optimización de dicho ajuste oferta/demanda para conseguir un mercado eléctrico competitivo.

La predicción del consumo de energía eléctrica según la simulación de modelos basados en agentes no se usa en los organismos encargados del ajuste oferta/demanda. Actualmente, Red Eléctrica Española utiliza su banco histórico de consumo así como sistemas expertos y otros

mecanismos más complejos para predecir el consumo. Sin embargo, en otros ámbitos como el consumo de agua se han realizado varios estudios utilizando agentes con los que se han conseguido buenos resultados para el estudio de características macro.

3.4 Objetivos

Con este trabajo nos hemos propuesto conseguir los siguientes objetivos:

- Conocer los recursos que proporciona Repast para la implementación de sistemas multiagente y familiarizarnos con el uso de dicha herramienta mediante la implementación de varios prototipos.

- Conocer el estado del arte de la simulación social basada en agentes a fin de conseguir ciertas directrices y prácticas aconsejables para la elaboración de un sistema multiagente para simulación social.

- Elección de un problema a modelar para el que se encuentre disponible suficiente información acerca del dominio que permita guiarnos para conseguir un buen modelo y posteriormente sacar conclusiones de los resultados obtenidos tras el modelado comparándolos con los datos disponibles.

- Conseguir un modelo general basado en agentes de modo que pueda ser utilizado en un futuro para desarrollar fácilmente sistemas multiagente de diversa índole y que sea fácilmente ampliable. En dicho modelo, ya están incluidos los elementos necesarios (funciones de transición, estados, tipos de agente, espacios) para el modelado del consumo eléctrico.
  Este aspecto es realmente importante, debido a que, en simulación social para conseguir un buen modelado son muy importantes conocimientos de sociología. De manera que lo ideal es un equipo interdisciplinar sociológico-informático, pero no siempre es posible. Así que sería importante para un mayor uso del modelado con agentes conseguir una aplicación que permitiera a sociólogos construir sus modelos abstrayéndose de los detalles de implementación, aunque en principio sólo permitiera implementar modelos relativamente sencillos.

- Implementación de tres escenarios concretos basados en el modelo general. Cada modelo pretende estudiar la tendencias que el consumo eléctrico va tomando según influyan en la población de agentes diferentes factores, debidos a la interacción agente-agente o a la interacción agente-espacio en el que vive.

- Análisis de los resultados conseguidos en los escenarios, a fin de evaluar la calidad de los modelos conseguidos.

- Obtención de conclusiones tanto acerca de los modelos conseguidos como acerca del modelado con agentes en general.

3.5 Método de trabajo

Para abordar los objetivos expuestos en el apartado anterior hemos seguido el siguiente plan de trabajo:

1. Estudio del estado del arte en cuanto a proyectos realizados acerca de simulación social basada en agentes a fin de ver cómo llevaron a cabo la resolución del problema que se plantearon y aprender de sus experiencias.

2. Estudio de modelos matemáticos existentes para algunas características sociales. Para observar los parámetros que se toman como relevantes en dichos modelos e incluirlos en nuestro sistema multiagente, entre otros.

3. Estudio de Repast que es la herramienta elegida para desarrollar nuestro sistema multiagente. Para ello desarrollamos en un principio un modelo muy sencillo que nos permitiera ver las posibilidades que nos ofrece Repast con respecto a gráficas y planificación. El sistema multiagente sencillo desarrollado modelaba la evolución de una sociedad desde el punto de vista demográfico.

4. Desarrollo de tres prototipos basados en el modelo sencillo que habíamos conseguido anteriormente. Los prototipos se desarrollaron independientemente y en paralelo. Tras su finalización se desarrolló un prototipo común uniendo los aspectos que cada uno de ellos tenía más desarrollado.

5. Elección y estudio de un problema concreto a resolver. Antes de tomar una decisión acerca de a qué problema queríamos aplicar nuestro modelo, tuvimos que comprobar que hubiese datos suficientes del ámbito que eligiésemos, de modo que tuviéramos la información necesaria para poder modelar cada agente. El problema elegido fue desarrollar un modelo del consumo de energía eléctrica de una sociedad.

6. Basándonos en el sistema multiagente conseguido de la fusión de los tres prototipos, desarrollamos el modelo general para el consumo energético. Poniendo especial atención a la arquitectura del sistema, de manera que fuese fácilmente ampliable y que separara claramente los gráficos del modelo en sí, y de utilizar los patrones adecuados para su diseño para su diseño: factoría abstracta, vista/controlador, etc.

7. A partir de ese modelo hemos desarrollado distintos escenarios para modelar el consumo eléctrico debido a distintos factores: el factor estacional, la inmigración/emigración y las campañas publicitarias para el uso responsable de energía eléctrica. De manera que para cada uno de ellos hemos desarrollado un modelo en el que emerge la influencia de dichos factores en el consumo eléctrico.

8. Por último, hemos analizado los resultados obtenidos en cada uno de los escenarios para obtener las conclusiones que exponemos en un apartado posterior de la memoria.

3.6 Estructura del documento

Los resultados de este trabajo y la experimentación que los apoya son recogidos en el resto del documento, que se compone de seis capítulos tal y como se indica a continuación.

En el segundo capítulo se expone el estado del arte, de modo que, se explican distintas herramientas utilizadas para el desarrollo de sistemas multiagente y a continuación realizamos una comparativa acerca de la herramienta elegida para la implementación de nuestro sistema multiagente, Repast, con alguna de estas herramientas para justificar la elección de ésta. Por otra parte aparece un repaso por algunos de los proyectos en los que se han desarrollado sistemas multiagente para simulaciones en el ámbito de la energía eléctrica: consumo, mercado, etc.

El tercer capítulo está dedicado al uso que hemos hecho, para la implementación de nuestro modelo, de la herramienta de modelado Repast.

En el capítulo cuarto se expone el diseño e implementación de los distintos modelos desarrollados a lo largo del proyecto. Primero, aparece el modelo general de agentes creado para usar de base para modificarlo en el que finalmente nos hemos basado del consumo energético. Segundo, se muestra el diseño del modelo general para el consumo energético. Aparecen sus diagramas UML de clase y de interacción; los patrones software utilizados y cómo los hemos aplicado y la explicación de cómo hemos modelado el consumo energético, conceptualmente.

A continuación, en este mismo capítulo, aparecen algunos detalles de implementación a los que hemos querido hacer una mención especial. Después explicamos el modelo y el diseño utilizados para la implementación de cada uno de los escenarios, así como los resultados obtenidos.

Finalmente, en el capítulo quinto se recogen las conclusiones obtenidas en el desarrollo de este trabajo y algunas futuras ampliaciones para éste que, consideramos, serían de interés.

Los dos últimos capítulos son el glosario de términos y la bibliografía utilizada que, como se puede comprobar, ha sido amplia y muy variada.

4. Descripción del estado del arte

4.1 Introducción

El desarrollo con Agentes Software es una de las ramas de la Inteligencia Artificial. Es difícil definir qué es un agente y nos hemos encontrado con muchas definiciones posibles. Simplificando mucho, se puede partir de la idea de objeto para definir un agente y añadirle una serie de características diferenciadoras.

Un agente es una unidad independiente que funciona y actúa por sí mismo y realiza una tarea concreta. Además otra característica importante de los agentes es su autonomía. Así como la capacidad de reacción ante otros agentes y antes eventos del entorno.

En la actualidad se están desarrollando muchas aplicaciones que utilizan sistemas multiagente. A la hora de desarrollar este proyecto nos hemos encontrado con aplicaciones de todo tipo, hemos encontrado aplicaciones industriales, de tráfico aéreo, comercio electrónico, medicina, telecomunicaciones, etc.

Otro elemento de estudio para el desarrollo de aplicaciones con sistemas multiagente son aplicaciones para usos concretos que abordan un problema común dentro de la sociedad. En este proyecto tratamos de abordar un problema de este tipo.

Un sistema multiagente nos permite modelar y simular una sociedad con unos datos, a priori conocidos y nos facilita establecer ideas sobre posibles futuros comportamientos de la sociedad, no solo a nivel de aumento de población. Nos da la posibilidad de intuir cómo cambiará la población y posibles respuestas a eventos que fijemos para nuestra sociedad artificial.

4.2 Trabajos anteriores

Entre los desarrollos de este tipo hemos encontrado trabajos para la predicción del consumo del agua, como por ejemplo la tesis doctoral de José Manuel Galán Ordaz que elabora un sistema multiagente para predecir el comportamiento de la ciudad de Valladolid estudiando la característica del consumo de agua.

Y relacionados con el problema de electricidad hemos encontrado también proyectos pero en menor cantidad que los que describen el problema del agua. A continuación vemos algunos.

4.2.1 Artículo sobre integración de mecanismos de razonamiento en agentes software inteligentes para la negociación de la energía eléctrica.

Introducimos este artículo como nuestra primera referencia por ser uno de los primeros que encontramos y por ser uno de los pocos que están en español. No es un artículo sobre consumo eléctrico si no sobre negociación de la energía eléctrica.

Se trata de un artículo escrito por F. J. Arias, J. Moreno, D. A. Ovalle de la universidad de la Rioja que trabajan con agentes inteligentes.

Una de las características principales de los agentes software inteligentes es que estos pueden tener un carácter deliberativo, el cual puede ser obtenido por medio de mecanismos de razonamiento que les permita desenvolverse de manera eficiente dentro de su entorno. La implementación de dichos mecanismos de razonamiento puede ser lograda mediante la integración de las plataformas JADE y JESS, con las que es posible implementar agentes de software híbridos (reactivos + deliberativos) mediante mecanismos de razonamiento basados en reglas.

El propósito de este artículo es ilustrar y resaltar las bondades en la integración de estas herramientas y su validación mediante una aplicación en concreto: Simulación de la negociación electrónica de contratos en el Mercado de Energía Eléctrica en Colombia.

4.2.2 Una aproximación con agentes software para los sistemas de distribución eléctrica autoconfigurables.

De la universidad de Puerto Rico en Mayagüez (EEUU) Janeth G. Gómez–Gualdrón y Miguel Vélez-Reyes, hemos encontrado un proyecto que relaciona la distribución eléctrica con los agentes software.

En este proyecto se habla de que podemos encontrar sistemas de distribución eléctrica en cualquier sitio hoy en día. Desde los que están en un barco hasta los localizados en centros de datos. En muchas aplicaciones críticas, es necesario mantener una mínima operatividad por debajo de las condiciones de fallo. Para conseguir esto, es necesario desarrollar herramientas de control de distribución, que permitan implementar un sistema autoconfigurable de control de energía eléctrica.

Este proyecto de investigación implementa un proyecto multiagente para desarrollar un sistema autoconfigurable de control de energía eléctrica.

Se propone un prototipo de sistema multiagente para reconfigurar el sistema y maximizar el número de servidores cargados con la mayor prioridad. La simulación se hace en MatlabTM, SimulinkTM y otro sistema multiagente implementado usando el lenguaje de programación Java y la plataforma JADE.

4.2.3 MASCEM un sistema multiagente que simula la competencia de mercados eléctricos

Realizado en el Instituto Politécnico de Oporto por Isabel Praca, Carlos Ramos y Zita Vale.

Se parte de la base de que mientras los sistemas de utilidad eléctrica en el mundo sigan camino de la mejora, con mercados competitivos, nuevas reglas, nuevos elementos y un nuevo comportamiento, continuará emergiendo. La competencia no sólo incrementa el número de empresas que se dedican al negocio eléctrico sino también afecta a su comportamiento. En este contexto la necesidad de nuevos modelos de comportamiento se hace obvia.

MASCEM, es un sistema multiagente que permite la combinación bilateral de mercados de comercio y fondo común. Eso también lleva a los compradores a tener un comportamiento estratégico, cosa común en aplicaciones similares.

Este sistema incluye entidades cambiantes como comerciantes de electricidad. El simulador ofrece estrategias y un escenario de análisis algorítmico para definir características y ofertas que incluye un modelo de nuevos agentes en el mercado, como comerciantes, que le dan mayor perspicacia a la evolución del mercado.

4.2.4 Soporte multiagente en tiempo real para la administración de energía eléctrica.

Publicado por la asociación IEEE (Washinton, EEUU), este artículo realizado en Dortmund (Alemania) por H.F. Wedde, S. Lehnhoff, E. Handschin y O. Krause nos habla de la administración en tiempo real usando un soporte multiagente.

En este artículo se trata el tema de cómo gestionar de manera adecuada la unión entre las distintas fuentes de energía. Partiendo de la base de que actualmente tenemos fuentes renovables y no renovables. Desde que los productores son a la vez consumidores y viceversa.

En primer lugar se describen las fases para la unión del proyecto R&D entre la Escuela de Computación y el College de Ingeniería Eléctrica de Dortmun. Esta unión se ha dedicado a descentralizar y adaptar la energía eléctrica a través de una arquitectura distribuida multiagente en tiempo real.

Teniendo en cuenta lo impredecible de las peticiones de los consumidores como los problemas de los productores, un control distribuido y con autonomía local son las principales novedades de este proyecto.

Este grupo presenta un algoritmo distribuido de negociación en tiempo real que da a los agentes distinto nivel de negociación.

A pesar de la falta de una visión global, este grupo es capaz de probar que en su modelo no habrá coaliciones de usuarios corruptos que pueden tomar ventaja de la extrema situación de subida por el exceso de consumo.

Su sistema multiagente muestra una gran robustez contra los problemas en comparación con las arquitecturas centralizadas. En largos experimentos se demuestra como en la estructura del suministro eléctrico alemán real los nuevos algoritmos pueden cubrir las necesidades y producciones específicas de una manera adaptable y flexible. Además demuestran que, bajo el enfoque descentralizado, los clientes pagan menos que bajo cualquier política o estructura de gestión convencional.

4.2.5 Un sistema basado en agentes para la protección de componentes eléctricos

Se trata de un artículo realizado en Italia por L. Hipólito y P. Siano sobre protección de componentes eléctricos.

Para gestionar un sistema eléctrico, una tecnología en expansión por el aumento en la capacidad de los grid's es necesario hacer uso de la electrónica, comunicaciones e información basada en innovación como comunicación de redes de datos combinadas con equipos inteligentes y microprocesadores, conocido como computación distribuida.

Este artículo trata de la implementación de una arquitectura multiagente para dispositivos eléctricos de control y protección. Se trata de un caso de estudio en el que la arquitectura se aplica a un transformador de potencia. En particular, para la realización del sistema que previene la sobrecarga de potencia.

Se desarrolla usando un sistema multiagente junto con un modelo fuzzy (de lógica difusa) de Takagi Sugeno Kang (TSK) que se identifica por un algoritmo genético para reproducir el comportamiento térmico de los transformadores de potencia. En el laboratorio se ha experimentado confirmando la efectividad del sistema de arquitectura multiagente.

4.2.6 Data Mining y visualización de datos en el mercado eléctrico español

Se trata de un artículo aparecido en la Revista Iberoamericana de Inteligencia Artificial. Realizado por Eugenio Francisco Sánchez Úbeda, Antonio Muñoz y José Villar.

Los datos sobre consumo eléctrico en general son muy extensos. En este artículo se propone un análisis de las curvas de consumo eléctrico y herramientas de minería de datos que son útiles para interpretar y predecir los comportamientos emergentes. Estos estudios están siendo usados por compañías eléctricas como Endesa.

Los gobiernos están actualmente liberalizando el mercado eléctrico para conseguir mayor eficiencia económica. El precio de la energía se obtiene, para cada hora, por la intersección de dos curvas, la de la oferta y la de la demanda. La curva de la oferta es una función escalón, mientras que la curva de la demanda es una función decreciente. En estos entornos los participantes tratan de maximizar el beneficio.

En distintos tipos de mercados las ofertas son enviadas por los participantes por bloques de energía con diferentes precios cada uno. Dentro del mercado de energía el volumen de datos es muy elevado. La interpretación de las curvas juega una baza muy importante para el mercado y su competencia.

El enfoque propuesto por patrones de aprendizaje no supervisado permiten descubrir patrones de comportamiento. Basándose en el pasado tratar de predecir el futuro.

El primer paso consiste en construir las curvas analizando la información publicada por los operadores. Se pueden agrupar las curvas de oferta y demanda para cada agente o agrupaciones de las mismas. Una vez disponibles comienza la fase análisis.

La caracterización del comportamiento de los agentes que se propone en el artículo se basa en la extracción del conocimiento usando técnicas de minería de datos. Como resultado obtenemos patrones de oferta y por otro lado curvas estadísticas de las curvas reales en torno a estos patrones. El análisis de la activación temporal de patrones y de su relación con otras variables son una fuente de conocimiento muy importante para el mercado.

Los modelos de aprendizaje supervisado que se usan son LHM y Sigmo, ayudan a obtener la esencia de las estrategias de ofertas de los distintos agentes, mientras que la aplicación de técnicas de agrupamiento facilita la visión conjunta de las mismas.

Se presentan también técnicas de visualización específica para extraer conocimiento relevante mediante análisis exploratorio.

La combinación de estas técnicas con la aplicación sistemática de métodos de aprendizaje supervisado para analizar la dependencia funcional entre variables ha permitido disponer de un conjunto de herramientas aptas para realizar análisis de distintos tipos.

Nos sirve para mantener actualizado el conocimiento disponible sobre el comportamiento estratégico de la competencia como para realizar estudios puntuales que puedan surgir.

4.2.7 Evaluación integradora de políticas de agua: Modelado con sociedades de agentes artificiales.

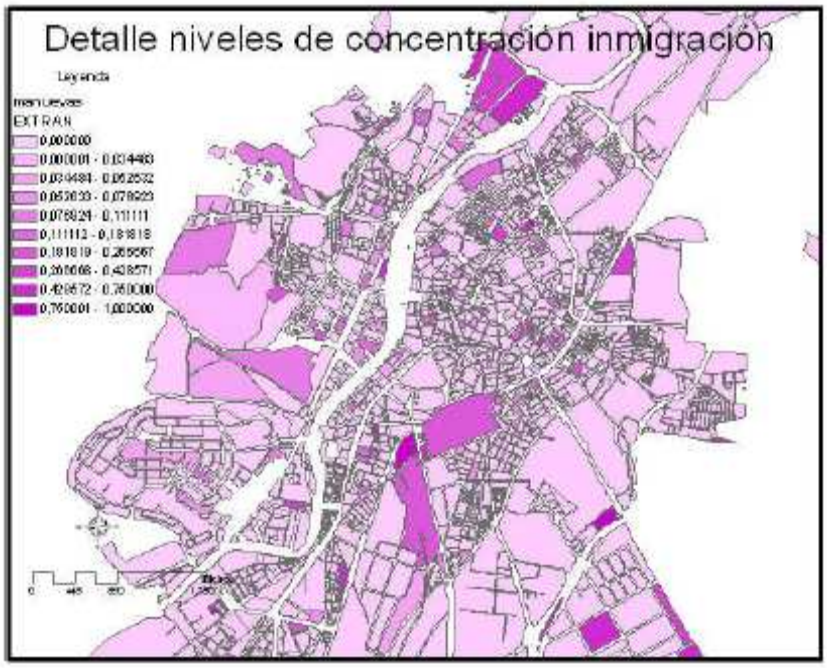
La tesis doctoral de José Manuel Galán Orgaz realizada en la Universidad de Burgos bajo la supervisión de Dr. Ricardo Olmo Martínez y el Dr. Adolfo López Paredes.

La tesis trata sobre gestión para consumo humano de agua. Muestra políticas del lado de la demanda y abastecimiento de agua aplicado a la ciudad de Valladolid. Utiliza métodos de estimación por demanda introduciendo como datos de entrada las características de las poblaciones.

Se centra también en demostrar la utilidad del modelado con agentes para resolver este tipo de problemas. Se adaptan distintos submodelos sociales para el estudio de características concretas de la población. Tiene en cuenta relaciones entre agentes, como afecta la vecindad,

características migratorias entre otras cosas. Todo esto con modelos matemáticos de fondo que explican comportamientos sociales.

Además en este proyecto se integran Sistemas de Información Geográfica (SIG) que hacen que la herramienta además sea aún más completa.



En un SIG, los datos que forman la base de datos espacial se representan mediante capas, cada una sobre una temática similar, que representan un determinado aspecto geográfico del entorno bajo estudio. Cada una de estas capas puede contener datos de atributos formando las bases de datos alfanuméricas.

Se ha parametrizado el modelo general en la Región Metropolitana de Valladolid obteniendo interesantes resultados. Las conclusiones obtenidas de esta forma son difícilmente alcanzables por otra vía, empleando por ejemplo las técnicas convencionales de previsión.

Se muestra de esta forma que la metodología descrita es capaz de complementar aquéllas ofreciendo visiones alternativas, realistas y rigurosas del problema.

Hay una primera parte de la tesis en la que se habla de cómo se ha hecho la estimación históricamente del agua y cómo factores ecológicos y de limitación de recursos han hecho que se tienda a nuevos métodos de estimación del consumo de agua. Habla de la relación entre la calidad de vida y el consumo de agua.

Un modelo en el sentido amplio es una abstracción de un sistema observado que permite establecer algún proceso de inferencia sobre cómo funciona el sistema o algunos aspectos de él. Los modelos basados en agentes hacen posible experimentar con escenarios completamente

repetibles y controlables, con la duración apropiada y con escalabilidad necesaria de acuerdo a la capacidad computacional necesaria disponible para el experimento, y sin riesgo para los individuos del sistema ni para un conjunto de recursos determinado.

El análisis de los resultados obtenidos mediante este conjunto de simulaciones puede servir para desarrollar teorías del comportamiento del sistema bajo las diferentes hipótesis utilizadas en la elaboración de los experimentos y entrar en las fases de validación de las explicaciones propuestas.

Finalmente en la tesis se desarrollan tres escenarios teniendo en cuenta diferentes características y demostrando la flexibilidad de su modelo.

4.3 Herramientas

Veremos a continuación herramientas y sistemas que se usan en la actualidad para el desarrollo de sistemas multiagente.

No se tiene, en la comunidad científica actual, cuál es la mejor herramienta para simulación con agentes. Esto trae el problema de que haya mucha gente trabajando en campos parecidos sin que el trabajo de uno pueda ser utilizado por otros.

Cuando una persona va a modelar con agentes tiene muchas posibilidades en cuanto a herramientas de desarrollo software. Hay diseñadores de software que utilizan el lenguaje C, el problema es que entonces tenemos que tener en cuenta otros factores (como son estructuras internas del sistema) que hacen que no se pueda aprovechar toda la potencia del modelado basado en agentes que dan otros lenguajes de más alto nivel.

Lo más difundido suele ser usar plataformas y lenguajes de propósito general generalmente con programación orientada a objetos como C++ o Java. Los desarrolladores tienen la ventaja de la modularidad y reutilización de código que aporta la programación orientada a objetos. Por otra parte esto también da mucha libertad al diseñador para diseñar todo lo que haya ideado. El problema suele venir a la hora de volver a implementar gráficas, librerías, etc.

En los últimos tiempos se vienen desarrollando un mayor número de plataformas asociadas a problemas concretos. Estos tipos de plataformas hacen que el diseñador esté totalmente restringido al dominio para el que fue implementado el modelo.

Dependiendo de los conocimientos y habilidades del modelador y del propósito específico del modelo, se elegirán unas herramientas u otras. Lo más habitual es utilizar librerías y herramientas basadas en características orientadas a objetos.

Dentro de estas opciones nos encontramos con librerías muy conocidas como *Swarm*, *Repast*, *SeSAm* o *Mason* basadas en lenguajes de propósito general como Java.

Una de las principales desventajas de estas aproximaciones es precisamente que el modelador necesita conocer con cierto nivel los lenguajes en los que se implementan las librerías. En la actualidad se busca hacer herramientas que sean muy flexibles, que permitan mucho movimiento, a la par que herramientas que sean sencillas de utilizar, pensando en desarrolladores que no tengan nada que ver con la informática. Si estamos abordando problemas relacionados con estudios sociológicos por ejemplo, es interesante buscar herramientas que sean de fácil acceso para sociólogos, médicos, dependiendo del campo sobre el que estemos trabajando. Se trata de acercar el mundo de la especificación al de la implementación lo máximo posible.

Existen por otro lado paquetes que permiten la construcción de modelos mediante *programación visual* como son StarLogo y AgentSheets. El problema de estos paquetes es que no tienen todas las funcionalidades que debieran para que sean sencillas para el usuario.

Veremos a continuación una selección de plataformas que hemos investigado a la hora de hacer este proyecto.

4.3.1 Swarm

Probablemente la herramienta de simulación más conocida, debido a que fue una de las pioneras. El proyecto Swarm se inició en 1994 dirigido por Chris Langton en el Santa Fe Institute de Nuevo México.

Su objetivo era el desarrollo tanto de un vocabulario, como de un conjunto estándar de herramientas para el desarrollo de modelos de simulación a través del modelado de colecciones de agentes que interactúan concurrentemente en un entorno dinámico. Inicialmente nació desde la biología computacional, motivo por el que esta especialmente diseñado para explorar el comportamiento de un gran número de agentes en interacción relativamente sencillos.

Swarm consiste en una colección de librerías portables y orientadas a objetos de componentes reutilizables que proporcionan un conjunto de herramientas flexible.

El componente fundamental que organiza los agentes en un modelo Swarm es el *swarm*. Los *swarms* son los componentes básicos de las simulaciones; un *swarm* no es más que una combinación de una colección de objetos que representan los agentes, un programa de actividades y eventos para los agentes (*scheduler*) y un conjunto de *inputs* y *outputs* que puede recibir o dar el *swarm*.

Cada *swarm* representa un modelo completo dentro de una simulación con su propia representación de tiempo. Los agentes en el modelo interactúan mediante series de pasos de tiempo, en realidad una simulación *Swarm* es modelo de simulación de eventos discretos. Las acciones se especifican mediante una colección de métodos únicos para cada clase de agente. El scheduler especifica el orden en el cual estos métodos se ejecutan a lo largo del tiempo.

En esta estructura se cumple uno de los pilares de la aproximación descentralizada que caracteriza al *ABM* (*agent-based model*), aquí nada en el modelo especifica el comportamiento global del sistema, sino que este comportamiento emerge como resultado de la interacción de los agentes individuales a través del tiempo.

Una característica fundamental y útil de la filosofía *Swarm* es que la estructura del programa posee dos niveles diferentes. En el primer nivel, se implementa el modelo (*ModelSwarm*) y en un segundo nivel, se implementa el observador (*Observer*) como un *swarm* encargado de "observar" la evolución de la simulación.

En esta filosofía se considera el modelo como un objeto con el que interactuar, recoger los datos que se necesitan y enviárselos a las herramientas de visualización (gráficos, diagramas e histogramas).

4.3.2 Ingenias

Dentro del grupo Grasia que pertenece al departamento de Inteligencia Artificial e Ingeniería del Software de la Universidad Complutense de Madrid se está desarrollando la metodología y herramienta Ingenias para el desarrollo con Agentes.

Ingenias se desarrolla a partir de MESSAGE centrándose en tres aspectos fundamentales:

- Integración de las vistas de diseño del sistema
- Integración de resultados de investigación
- Integración con el ciclo de vida de desarrollo de software

Uno de los objetivos de Ingenias es proporcionar un entorno que permita el modelado e implementación de sistemas multiagente para el estudio de características sociales en un entorno de computación en grid a través de la metodología Ingenias.

El segundo objetivo que aborda Ingenias es facilitar el modelado abstrayéndose en la medida de lo posible de todos los aspectos más relacionados con la programación, que el modelador modele pero no programe.

Para ello se dotado a la herramienta de la posibilidad de describir fenómenos de diferentes contextos sociales mediante lenguaje visual, lo que simplifica el modelado puesto la especificación de modelos mediante diagramas y de forma gráfica resulta más cercana al lenguaje natural, pero a la vez menos ambigua. Esta descripción visual además es independiente de la plataforma de simulación.

4.3.3 Mason

Mason es una plataforma para la simulación de eventos discretos en máquinas monoprocesador, dotada con herramientas de visualización muy potentes, gratuita y de código

abierto. Ha sido desarrollada conjuntamente por el Computer Science Department y el Center for Social Complexity de la George Mason University.

Inicialmente se originó como un conjunto de librerías muy condensadas destinadas a un muy amplio rango de aplicaciones, desde la robótica, a modelos físicos o de simulación social, auque es en esta última donde ha encontrado mayor aceptación.

Está especialmente orientada hacia la simulación intensiva, permitiendo recuperar simulaciones no finalizadas y considerando la velocidad de simulación como uno de sus objetivos. También pone mucho énfasis en la independencia de la plataforma y la replicabilidad de los resultados. En parte este último objetivo se fundamenta en que Mason está implementado en Java para aprovechar la portabilidad de la plataforma y la serialización de objetos.

Precisamente una de las ventajas que incluye Mason en la visualización de información es mediante la inclusión de las librerías Java 3D, aparte de como todas la herramientas basadas en Java permite la importación de todo tipo de librerías externas.

En muchos aspectos Mason recuerda a Swarm y sobre todo a Repast. De hecho comparte la filosofía de ambas y la separación modular y por capas entre el núcleo de la simulación y la capa de visualización. Su arquitectura se divide en un conjunto de utilidades de estructuras de datos de propósito general.

Una capa de modelado consistente en un conjunto de clases para la programación de eventos discretos, librerías de generación de números aleatorios y conjuntos de clases para la implementación de entornos. Con estas dos capas es suficiente para la generación intensiva de simulaciones en modo no gráfico. Y por último posee una capa de visualización para la monitorización completa de la simulación.

4.3.4 SeSAm

SeSAm (Shell for Simulated Agent Systems) es otro intento de facilitar el modelado y la experimentación de sistemas ABM (agent-based model). Está escrito en Java y es código abierto.

SeSAm proporciona un editor de comportamientos basado en máquinas de estado finito, similar a los diagramas de estado de UML por lo que aporta un mecanismo riguroso y extendido de modelado.

Sin embargo, y a pesar de que en sus requisitos de diseño se considera que está destinado a usuarios con conocimientos limitados de informática, resulta relativamente complejo puesto que hay que conocer un conjunto amplio de primitivas del entorno.

4.3.5 Repast

Repast corresponde al acrónimo de REcursive Porous Agent Simulation Toolkit. La versión Repast J es software libre y está disponible en la página www.sourceforge.net

En la actualidad quizá el paquete de modelado basado en agentes con mayor relevancia dentro de las herramientas disponibles en el campo.

Inicialmente fue concebido desde la Universidad de Chicago como un conjunto de librerías de clases de Java que funcionase conjuntamente con la estructura de simulación *Swarm*.

Hoy en día las principales diferencias entre Repast y Swarm son que Repast tiene múltiples implementaciones puras en diferentes lenguajes de programación orientados a objetos (Python y Java) y da soporte a herramientas para la realización de regresiones, programación de algoritmos genéticos o incluso la integración con Sistemas de Información Geográfica (SIG).

Ha sido mantenido por la Universidad de Chicago y el Argonne Nacional Laboratory. Repast está gestionado en la actualidad por Repast Organization for Architecture and Development.

En cuanto a la estructura, un modelo en Repast se puede considerar como una colección de agentes de cualquier tipo y un sistema que configura y controla la ejecución de los agentes de acuerdo a un programa.

Repast permite el desarrollo de diferentes modelos que luego se podrán simular paso a paso, mediante lo que se conoce como un step, o tiempo de reloj.

Para modelar con Repast es necesario tener claro qué será un agente dentro del sistema. Es decir definiremos cuales son las características, variables, comportamiento de nuestras unidades agente. Estos agentes pueden representar personas o como en nuestro proyecto representan familias enteras. También estableceremos las interacciones entre los agentes. La manera en que un elemento se relaciona con otro. Por otro lado definimos transición de un estado a otro del agente. Una gran complejidad de funciones de transición.

Otra de las grandes ventajas de Repast es que la librería te trae definido muchos elementos que son útiles para la simulación y para el desarrollo estadístico. En Repast es inmediato el hacer una gráfica de alguna característica, así como histogramas, etc.

La gran flexibilidad de Repast le permite también la inclusión de sistemas de información geográfica (GIS) que son datos vectoriales que nos permiten representar mapas y planos donde estarán nuestros agentes.

Cuando se define un modelo en Repast, es importante definir el Scheduler o planificador que nos ira diciendo paso a paso (con la función step) lo que ocurre en nuestro sistema.

4.3.6 Artículos de comparación de herramientas

Repast versión 3 puede pensarse como una especificación que ofrece servicios y funciones de modelado basado en agentes. Existen tres posibles implementaciones de esta especificación que podríamos haber utilizado. Todas ellas tienen los mismos servicios básicos que constituyen el sistema de Repast. Existe la versión para Java (Repast J), para Microsoft .NET (Repast .NET) y para Python Scripting (Repast Py). De éstas tres opciones hemos trabajado con Repast J puesto que Java es el lenguaje con el que estamos acostumbrados a programar y que conocemos, a diferencia de las otras dos opciones. Con la opción de Repast J pueden construirse modelos avanzados al igual que con Repast .NET. No ocurre así con Repast Py que sirve para realizar en general modelos básicos.

Repast 3 tiene una serie de características que comentamos a continuación:

- Incluye una variedad suficiente de plantillas de agentes y ejemplos. Sin embargo, la herramienta da a los usuarios flexibilidad al especificar las propiedades y comportamientos de los agentes.
- Repast es completamente orientado a objetos.
- Incluye una agenda de eventos discretos concurrente. Esta agenda soporta tanto operaciones de eventos discretos secuenciales como en paralelo.
- Repast ofrece por construcción herramientas para realizar registros de acceso y gráficas.
- Repast automatiza la estructura de simulación de Monte Carlo.
- Provee una gran cantidad de entornos y visualizaciones de agentes en dos dimensiones.
- Permite a los usuarios modificar y efectuar dinámicamente un acceso a las propiedades de los agentes, a las ecuaciones de comportamiento de los agentes y a las propiedades del modelo en tiempo de ejecución.
- Repast incluye librerías para algoritmos genéticos, redes neuronales, generación de números aleatorios y matemáticas especializadas.
- Incluye también modelado dinámico de sistemas empotrados.
- Tiene herramientas que soportan modelado de redes sociales.
- Integra el soporte para sistemas de información geográfica (SIG).
- Está implementado en varios lenguajes incluidos Java y C#.
- Los modelos pueden ser desarrollados en muchos lenguajes como por ejemplo Java, C#, Managed C++, Visual Basic .NET, Managed Lisp, Manager Prolog y Python Scripting.
- Repast está disponible en todas las plataformas modernas de computación incluyendo Windows, Mac OS y Linux. Para cada plataforma se incluye soporte tanto para computadoras personales como para clusters de computación científica a gran escala.

Otra de las características en las que destaca Repast es el hecho de que provee de una estructura conceptual para organizar y diseñar modelos basados en agentes (ABMs, agent-based models) y tiene también sus correspondientes librerías.

Repast comenzó como una implementación en Java de Swarm pero finalmente se diferenció significativamente de Swarm. Herramientas del tipo de Repast han tenido éxito en gran medida porque proveen de diseños de software estandarizados y herramientas sin las que se limitaría el tipo y complejidad de implementación de los modelos.

Para nuestra elección de la herramienta hemos sacado información de distintas evaluaciones de Repast frente a otras herramientas. De estos documentos intentamos sacar una medida que nos hiciera decidirnos por la herramienta más conveniente para desarrollar nuestro modelo.

Uno de los trabajos de comparativa consultados es el artículo "Agent-based Simulation Platforms: Review and Development Recommendations" escrito por Steven F. Railsback, Steven L. Lytinen y Stephen K. Jackson en el que hacen una comparación para un modelo que llaman *StupidModel* en el que realizan experimentos con cinco herramientas científicas distintas para desarrollar ABMs (agent-based models) incluyendo escenarios en los que varían las entradas como valores de los parámetros y replican el experimento en donde sólo varía la semilla del generador de números aleatorios. La comparación de cada plataforma la realizan basándose en algunas características como la velocidad de ejecución, cuestiones generales de la simulación, etc. NetLogo es la plataforma de mayor nivel, pues provee de un lenguaje de programación muy simple y útil, interfaces gráficas construidas y documentación comprensible.

Fue diseñado inicialmente para ABMs de individuos móviles con interacciones locales en un espacio de grid, pero no necesariamente torpe para otros ABMs. NetLogo es altamente recomendable, incluso para hacer prototipos de modelos complejos. Mason, Repast y Swarm con plataformas de "framework y librería", que poseen una estructura conceptual para organizar y diseñar con una velocidad de ejecución de alta prioridad. La versión de Swarm *Objective-C* es la plataforma de librería más madura y es estable y está bien organizada. *Objective-C* parece más natural que Java para ABMs pero tiene unos manejadores de error débil y la ausencia de herramientas de desarrollo es un inconveniente. Java Swarm permite llamar desde Java a librerías de Swarm en su versión de *Objective-C*; esto último no parece que combine las ventajas de ambos lenguajes bien. Repast provee funciones parecidas a las de Swarm en una librería de Java como una buena opción para muchos, pero partes de su organización y diseño podrían ser mejoradas. Una comparación improvisada de velocidad de ejecución encontrada fue que MASON y Repast son los más rápidos, que Swarm es el más rápido para modelos simples pero la más lenta para los modelos complejos y NetLogo uno intermedio.

Los dos objetivos de éste artículo son muy claros. El primero de ellos es revisar y comparar plataformas hoy en día usadas ampliamente para los ABMs: MASON, NetLogo, Repast y las versiones de Java y *Objective-C* de Swarm. El segundo es identificar las prioridades del desarrollo: ¿cuáles de las directrices generales en el futuro del desarrollo de plataformas para ABMs parece probable para hacerlas más productivas?

Unos de los resultados a los que llegan, desde el punto de vista de los objetivos, filosofías y terminología de las distintas plataformas, es que las diferencias de velocidad son dependientes mayormente del diseño de cada plataforma más que por sus agendas, espacios, etc. Con respecto a Repast llegan a la conclusión de que es una herramienta que busca facilitar su utilización por usuarios inexpertos en construir modelos pues ofrece varias maneras ya predeterminadas de crear modelos.

En resumen, las recomendaciones incluidas completando la documentación, para todas las plataformas menos para NetLogo, defiende las estructuras conceptuales, dando mejores herramientas para estadísticas de las salidas y automatización de los experimentos de simulación, simplificando las tareas comunes e investigando tecnologías para entender cómo surgen los resultados de la simulación.

Otro artículo de evaluación trata de librerías de Java libre para simulación científica-social basada en agentes (Evaluation of free Java-libraries for social-scientific agent based simulation) escrito en el 2004 por Robert Tobias y Carole Hofmann. En él se comparan cuatro librerías de programación que soportan simulación computacional social científica basada en agentes, a saber: Repast, Swarm, Quicksilver y VSEit (Versatile Simulation Environment for the Internet, Entorno de simulación versátil para Internet). Un primer paso fue evaluar las librerías según la documentación oficial y las opiniones de desarrolladores y usuarios y las experiencias e impresiones de los evaluadores.

Otro de los factores que evaluaron fue el esfuerzo/tiempo/energía ahorrado por parte de los sociólogos utilizando una librería ya programada con tenerla que programar ellos mismos. La evaluación demostró que el entorno de Repast es el mejor en estos aspectos con respecto a las herramientas mencionadas.

Una de las dificultades que mostró la simulación basada en agentes era el hecho de que es una rama de investigación que se está desarrollando rápidamente. El principio básico es que un sistema está construido por un número de subsistemas llamados agentes que eleva la complejidad del problema. Véase como ejemplo una ciudad de miles de individuos.

Los métodos han cobrado interés por la posibilidad de generar sistemas complejos basados en reglas simples, la solución del problema de transición micro-macro y la claridad de los modelos y otras ventajas. Algunas de las simulaciones sociales que ya han sido desarrolladas son “Mosler y Bruces 2003: Integrating commons dilemma findings in a general dynamic model of cooperative behavior in resource crises.” (Conclusiones de la integración de dilemas comunes en un modelo dinámico de comportamiento cooperativo en crisis de recursos) o “Mosler, Schwarz, Ammann y Gutscher 2001: Computer simulation as a method of further developing a theory: Simulating the Elaboration Likelihood Model (ELM)” (Simulación computacional como un método para el desarrollo de una teoría: Simulación del modelo de elaboración de la probabilidad o similitud).

Los criterios bajo los cuales determinaron la evaluación de dichas herramientas fueron las que definieron para poder evaluarlas y que se describen a continuación. Aunque el coste no es uno de los criterios puesto que las cuatro herramientas están disponibles gratis, existen diferencias como la disponibilidad del código fuente debido a sus licencias.

Las licencias incluyen otra cuestión: las propias modificaciones hechas en un framework de simulación que están bajo GLP o LGLP deben cumplir sus términos. No está prohibido hacer un uso comercial pero es obligatorio hacerlo código abierto. Adicionalmente a las licencias, la documentación y el soporte son características muy importantes, puesto que sin soporte para

resolver los problemas hasta el mejor programa es inútil. Finalmente, el framework debe tener viabilidad en el futuro, de manera que el soporte y la difusión de su uso estén asegurados.

Según éste artículo, los frameworks de software de código abierto para la creación de simulaciones basadas en agentes deben tener 3 ventajas comparadas con un lenguaje de programación de propósito general:

1. Reducción de la inversión necesaria para poder escribir programas que permitan al sociólogo centrarse más en la teoría y el trabajo del contenido del modelado.
2. Reducción del conocimiento sobre programación requerido para modelar y usar; incrementando la facilidad de modelado.
3. Incrementas la reusabilidad y la eficiencia de la simulación, como procedimientos complejos que hayan sido desarrollados por programadores expertos.

Para estos problemas son necesarias herramientas que cumplan lo aconsejado por Axelrod. Axelrod (1997) en el desarrollo de un modelo de simulación aconseja que el programa desarrollado cumpla tres requisitos: validez, usabilidad y extensibilidad. La validez a la que hace referencia Axelrod corresponde a que el programa ha de implementar correctamente el modelo conceptual. El segundo objetivo que marca Axelrod, el de la usabilidad, hace referencia a que el programa debe permitir tanto al investigador como a cualquier otro que desee ejecutar el software interpretar los resultados y entender como funciona fácilmente. Por último, la extensibilidad busca que al desarrollar un programa conviene tener en mente la posibilidad de adaptar el programa para nuevos usos.

Teniendo estos principios en cuenta, en el desarrollo de un modelo la primera decisión que se ha de tomar, incluso antes de elegir la plataforma o lenguaje de desarrollo, es el paradigma de modelado que resulta más interesante para los objetivos del modelo, posteriormente dentro del paradigma, declarativo o procedural, elegiremos la plataforma que más nos interese. Nuestra elección ha sido una estructura procedural pues al final nos hemos decantado por Repast.

Otra de los obstáculos que plantea realizar un modelo basado en agentes es que las aplicaciones orientadas a modelos de simulación social científica se caracterizan por tener agentes relativamente complejos, que contienen por regla varias teorías sociológicas y por el hecho de que, al mismo tiempo, la simulación trabaja con poblaciones de decenas de miles de agentes.

5. Repast

5.1 Implementación del framework Repast aplicado a nuestro proyecto

A la hora de implementar un modelo con agentes es imprescindible que haya tres elementos. La clase Agente, la clase Espacio y la clase principal que es la clase Modelo.

En la clase Agente se definen las características fundamentales que tiene cada una de las unidades que van a ser estudiadas. Esta clase será distinta en función de las características sobre las que se vaya a modelar.

En la clase Espacio se define el entorno que van a tener los agentes, los elementos con los que van a interactuar con ellos. En ocasiones, en modelos simples, la clase Espacio está integrada con el Modelo.

La clase principal de un modelo implementado con agentes es la clase Modelo. Está clase extiende de la clase SimModelImp.

La clase SimModelImp es una implementación parcial del interfaz SimModel. Muchos, si no todos, de los modelos reales heredarán de esta clase. Por defecto SimModelImp inicializa un generador de números aleatorios en la clase Random usando el tiempo del sistema como semilla.

Dentro del modelo habrá un objeto de tipo Schedule (planificador o agenda) que hereda de la clase ScheduleBase que controla la ejecución de acciones básicas (BasicAction, ActionGroups, SimActions y SimListActions) de acuerdo con un reloj de simulación. El reloj se incrementa al final de cada una de las BasicActions, una Schedule es en sí misma una BasicAction así que puede ser añadida a otro objeto de tipo Schedule.

Las acciones planificadas dentro de una Schedule harán iteraciones simulando concurrencia (en orden aleatorio). Si se quiere ejecutar acciones en un orden especifico, la acción debe ser añadida a un ActionGroup para un orden en secuencia. Este ActionGroup puede ser añadido a una Schedule para ejecución.

Especificando el orden en los métodos scheduleAtBeginning, scheduleActionAt y scheduleActionInterval se puede conseguir una acciones sucedan a otras.

Otro elemento que podemos encontrar es la lista de Agentes. Esta lista es la población de agentes que tendrá nuestro modelo. En modelos con poblaciones dinámicas se agregarán y eliminaran agentes de nuestra lista de agentes. En modelos con poblaciones estáticas esta lista tendrá siempre la misma longitud.

Cuando se inicializa el modelo se llama a la función begin. Esta función llama a su vez las funciones buildModel y buildSchedule, en caso de que sea necesario mostrar gráficas es en esta función donde se hace el display (se muestran).

En el buildModel es dónde se inicializa la lista de agentes que se usará luego en el modelo.
En el buildSchedule se define el ScheduleStep o el paso de planificiación. En esta función se define los elementos que ocurrirán en cada paso que de el modelo. Si estuviéramos trabajando con un modelo de estudios demográficos y cada paso (step) fuera un cuanto significativo de tiempo, en el ScheduleStep contaríamos los agentes vivos y eliminaríamos de nuestra lista los agentes muertos.

Así mismo dentro de la clase Agente existe un AgentStep que es un paso en la vida los agentes. En función de las características específicas cada agente tendrá su planificación concreta.

Otros elementos importantes que ofrece el framework es la posibilidad de mostrar estadísticas, histogramas y gráficos útiles para sacar conclusiones y poder comparar con modelos reales.

Un display es una presentación gráfica de agentes en su entorno. Se crean en Repast mediante el uso de tres tipos de clases: Espacios (la clase en el paquete uridiego.src.sim.space), los displays correspondientes a esos espacios y el DisplaySurface.

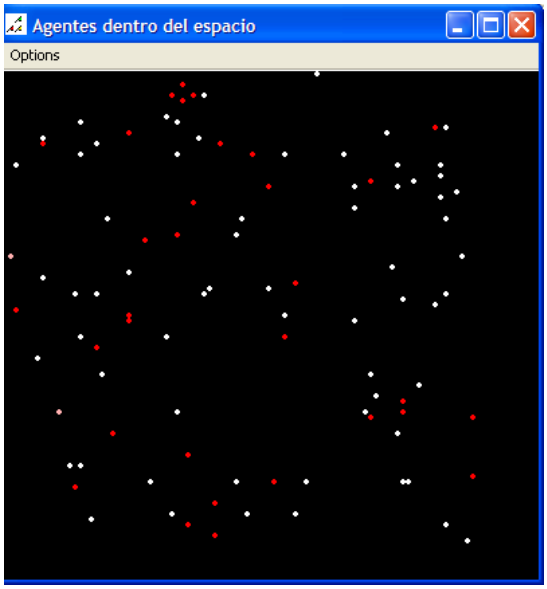


Ilustración 1. Ejemplo de DisplaySurface.

Los objetos como agentes, y entornos individuales deben implementar ciertos interfaces si queremos que se puedan mostrar.

Antes de mirar el método buildDisplay es útil comprender como Repast inicializa y dibuja los displays.

Para conseguir que Repast pinte, debemos seguir varios pasos. El primero es crear lo que vamos a pintar. En general los agentes y su entorno. Esto se hace en la función buildModel. También en buildModel los agentes son añadidos al espacio. Este espacio puede ser un Object2D por ejemplo. Es decir en esta función añadimos el agente al display y a la lista de agentes.

Después en la función buildDisplay se crea el display apropiado para el espacio (por ejemplo Object2DDisplay). Nuestra lista de agentes también se añade al display.

El display mismo también se añade al DisplaySurface. El DisplaySurface se crea en la función setup. En resumen, después de buildDisplay tienes un DisplaySurface que contiene displays que contiene espacios que a su vez contienen objetos que pueden ser mostrados.

Una vez hemos inicializado esta estructura la secuencia actual de dibujo es la siguiente: El método updateDisplay se llama desde el objeto DisplaySurface. Habitualmente esta llamada es planificada. Cuando se recibe la llamada el objeto DisplaySurface le dice a todos los displays que contienen que se muestren. Les hace una llamada a estos displays y ellos responden con información útil para el dibujo (coordenada, etc.) para cada objeto de la lista de objetos. Cada objeto se pinta a sí mismo. Estos métodos para pintar objetos varían ligeramente de acuerdo al display y están encapsulados en el interfaz de dibujo.

Para usar gráficos e histogramas sobre variables usamos las clases OpenSecuenceGraph, la clase Histogram, la clase OpenHistogram, etc.

Un OpenSequenceGraph muestra variables definidas en tiempo, dicho tiempo viene definido por el paso de reloj. En principio se muestra el objeto OpenSequenceGraph y luego se van añadiendo Sequences a él. Planificando y mostrando actualizaciones.

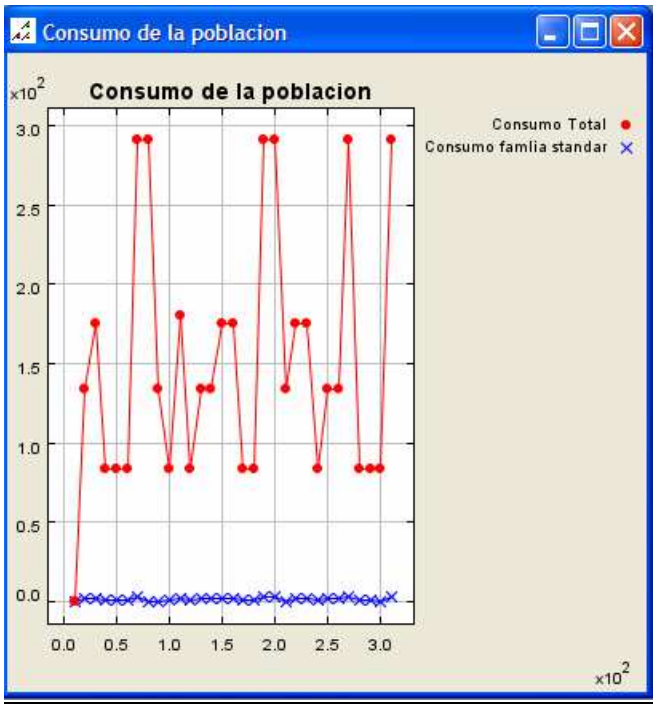


Ilustración 2. Ejemplo de OpenSequenceGraph.

Un Sequence es un interfaz que se implementa y sirve de fuente para mostrar los datos. Sólo tiene el método public double getSValue.

El OpenSequenceGraph trabaja llamando a getSValue de cada una de las secuencias que tiene. Un solo OpenSequenceGraph puede contener muchas secuencias.

Para actualizar el SequenceGraph, es necesario llamar al método step. Normalmente esta llamada planificará un tick o un intervalo. Cuando OpenSequenceGraph llama el método step, llamará al método getSValue y todas sus secuencias para mostrar los resultados.

Un OpenSequenceGraph se puede mostrar llamando a su método display. Esto se hace habitualmente en el método begin. También podemos mostrar los datos en un fichero mediante el método writeToFile de OpenSequenceGraph.

La clase OpenHistogram nos permite representar una variable dinámica, así como un dato que se genera de una colección de objetos. El rango del histograma es [lowerBound, maxValue] donde maxValue se calcula cada vez que se hace un step del gráfico.

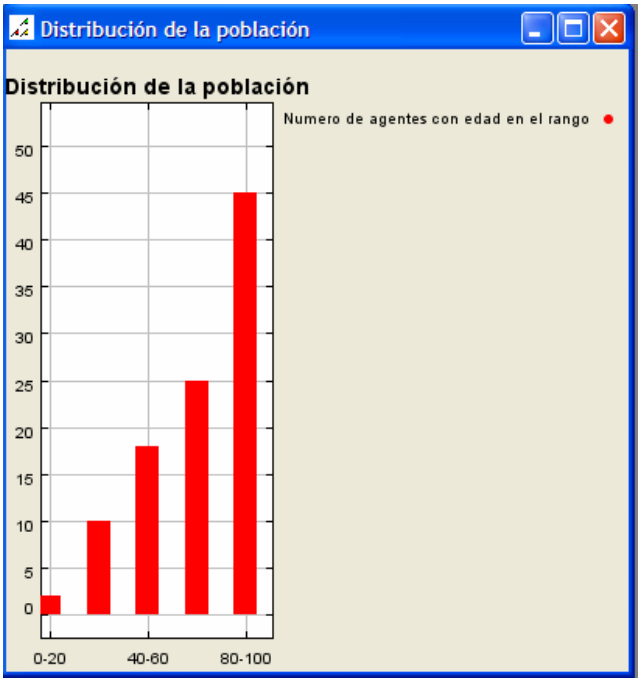


Ilustración 3. Ejemplo de OpenHistogram.

Usar OpenHistogram es muy parecido a usar OpenSequenceGraph. Se crea primero el OpenHistogram, los elementos del histograma. Estos elementos se van actualizando de manera planificada.

Cada HistogramItem es un elemento que muestra el histograma. Por ejemplo uno puede mostrar la edad de los agentes y otras características del entorno.

Para cada HistogramItem que contiene un OpenHistogram, ese HistogramItem itera sobre la lista específica cuando es creado pasando cada Objeto de la lista como argumento mediante getBinValue de BinDataSource. Este método, getBinValue, devuelve un valor de tipo double. Todos estos valores se distribuyen en el panel desde el minBound hasta el maxBound.

BinDataSource es un interfaz con sólo un método getBinValue(Object o) que se implementa en el modelo. Este objeto que se pasa por parámetro entra en el método getBinValue desde la lista de objetos junto con el método createHistogramItem de OpenHistogram. Cada objeto de la lista pasa por el método getBinValue durante la actualización.

OpenHistogram también tiene métodos para crear un BinDataSource mientras se crea un HistogramItem. En este caso se le pasa un objeto por referencia y el nombre del método para llamar al objeto. El método toma un objeto y devuelve un valor numérico.

Como OpenScheduleGraph, un OpenHistogram se actualiza usando la función step. Esta función puede ser planificada en un paso o en todo el intervalo. Cuando se llama al método step en OpenHistogram, el OpenHistogram se actualiza mediante los HistogramItems.

Un OpenHistogram se muestra llamando a la función display. Esto, de nuevo, se hace en la función begin dentro del modelo.

La clase Histogram representa una lista dinámica de barras y permite al usuario generar una gráfica de barras de datos usando una colección de objetos (además de la lista de agentes de tu modelo). Se diferencia de OpenHistogram en que el tamaño es estático. El rango del histograma va desde lowerBound hasta maxBound. Hay que tener en cuenta que a diferencia de OpenHistogram maxValue no se recalcula en cada paso, pero es un valor que nos da el modelo.

Usar un Histogram es muy parecido a usar un OpenHistogram. Se crea el histograma, se crea el HistogramItem mediante el histograma. Y se puede planificar las actualizaciones y su manera de ser mostradas. Sin embargo solo un HitogramItem se puede crear con un Histograma.

La clase Histogram1D nos da muchos métodos para trabajar con histogramas. Para ver lo que subyace se puede usar el método getHistogram de la clase Histogram1D.

6. Arquitectura genérica del sistema

6.1 Modelo Vista Controlador

Para la implementación de nuestro proyecto, hemos utilizado uno de los patrones más conocidos en el desarrollo: **el patrón MVC** (Modelo Vista Controlador). Este patrón nos permite/obliga a separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (cómo interactuar con el usuario, cómo se muestra la información).

Esta separación de los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos consigue que las modificaciones al componente de la vista puedan ser hechas con un mínimo impacto en el componente del modelo de datos. Esto es muy útil ya que los modelos suelen ser relativamente estables (dependiendo de la estabilidad en el dominio del problema que está siendo modelado). Además, utilizándolo conseguimos normalización y estandarización del desarrollo de Software.

En el patrón de diseño MVC, el flujo de la aplicación está dirigido por un *Controlador* central. El Controlador delega solicitudes -- en nuestro caso, el modelo en sí – a un manejador apropiado que sepa hacer la tarea encomendada por el modelo. Los manejadores están unidos al *Modelo*, y cada manejador actúa como un adaptador entre la solicitud y el Modelo. El Modelo representa, o encapsula, un estado o lógica de la aplicación. Finalmente, el control normalmente es devuelto a través del Controlador hacia la *Vista* apropiada.

En términos generales, construir una aplicación usando una arquitectura MVC implica definir tres clases de módulos.
Modelo: La representación específica del dominio de la información sobre la cual funciona la aplicación. La lógica de dominio añade significado a los datos.
Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
Controlador: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

El MVC desacopla vista y modelo estableciendo un protocolo de notificación entre ellos. Una vista debe asegurar que su apariencia refleja el estado del modelo. Cuando el modelo cambia, lo notifica a la vista que depende de él. Como respuesta, cada vista tiene la oportunidad de actualizarse. Este enfoque permite utilizar múltiples vistas de un modelo que proveen diferentes presentaciones. Se puede también crear nuevas vistas para un modelo sin reescribirlo.

Es común pensar que una aplicación tiene tres capas principales: presentación, dominio y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre V/C es menos clara.

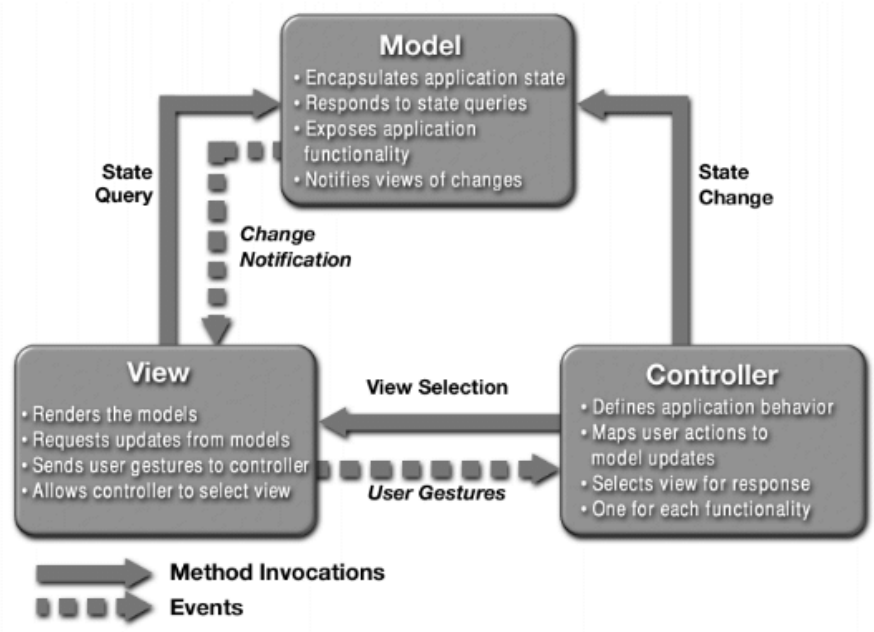


Ilustración 4. Modelo Vista Controlador (MVC).

6.2 Explicación del modelo genérico

6.2.1 Modelo general realizado antes para modificar al del consumo eléctrico.

Este modelo fue el resultado de la fusión de los tres prototipos desarrollados a partir del prototipo inicial.

En este modelo cada agente tiene un:

- Estado vital (recién nacido 0, niño 0-14, adolescente 14-19, adulto 19 66, anciano >66 o muerto)
- Forma física (bueno o fuera de forma)
- Salud (bueno, enfermo o enfermo crónico)
- Estado civil (soltero, casado, viudo o divorciado)
- Nivel cultural (alto, medio o bajo)
- Nivel económico (alto medio o bajo)

Estas son las características que tienen un estado y una función de transición asociadas. Pero hay otras que son inherentes a cada agente y que no cambian a lo largo de la simulación como el ID, el sexo, el atractivo o el espacio en el que está.

Cada agente dispone de una función de transición por cada una de sus características. Una función de transición hace que una característica pase de un estado a otro según ciertas reglas en las que pueden incluir factores pseudos-aleatorios.

En este modelo hay un único espacio geográfico donde se encuentran todos los agentes y dentro de este espacio geográfico hay dos espacios de salud (el bueno y el malo). De este modo, el estar en un espacio de salud bueno va a tener un cierto peso a la hora de que los agentes que están en él conserven una buena salud (aparece la interacción agente-entorno), al igual que va a influir su forma física o la etapa vital en la que se encuentra.

La interacción agente-agente está modelada en dos aspectos distintos. Todos los agentes tienen una probabilidad de tener hijos y si dentro del espacio se encuentran un agente-mujer de 13 a 55 años con un agente-hombre mayor de 13 años si la probabilidad de cada uno de ellos de tener hijos es mayor que 0,2 van a tener 2 hijos (este número lo hemos elegido con intención de seguir un poco la media de hijos por familia en nuestro país). No es requisito que dos agentes estén casados para que puedan procrear.

Por otra parte, también hay interacción agente-agente a la hora de elegir pareja para cambiar el estado civil a casado. De modo que en esta sociedad artificial aquellos posibles futuros casados son los que son atractivos, el estado del nivel económico y cultural es alto. De entre los posibles casados en cada step se van a casar algunos de ellos dependiendo de una cierta probabilidad.

La planificación del modelo se compone por:
- Al principio se crean todos los agentes y se inicializan aleatoriamente los estados de sus características que más adelante sólo podrán ser modificados por sus respectivas funciones de transición.
- Cada 10 quantos de tiempo se realizan la siguientes *BasicAction:*
    1. Actualizamos los estados de las características cada agente de la población.
    2. Se hace el ciclo de reproducción de agentes y el ciclo de cambio de estado civil de los agentes (que pueden cambiar de solteros a casados como de casados a divorciados).
    3. Se cuentan los agentes vivos.
    4. Se actualizan los histogramas de edad, salud y forma física de nuestra población. En los que se muestra la distribución de la población según estos distintos aspectos. Y también se muestra un *report* de todos los agentes.

A continuación parecen los resultados que obtenemos para una población inicial de 20 agentes y un espacio geográfico de 50x50:

1. Al ejecutar la aplicación nos aparece la interfaz que nos proporciona Repast para ver la simulación:

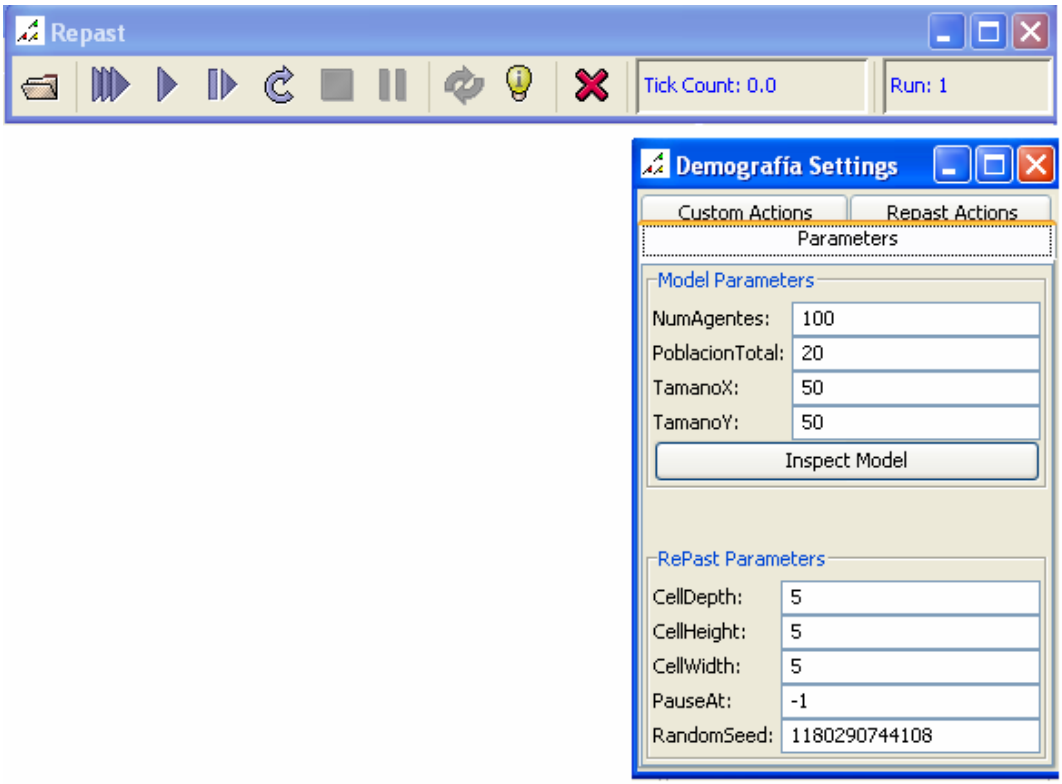


Ilustración 5. Interfaz de Repast para la simulación.

2. Al presionar  nos aparecen los gráficos que habíamos creado:

La población en el espacio geográfico creado

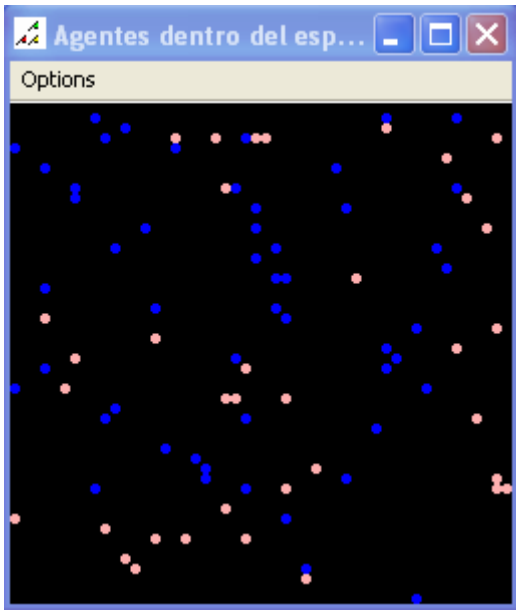


Ilustración 6. Población inicial.

Los agentes Niño aparecen en azul, los agentes Niña rosa, los agentes Adulto aparecen en verde si son hombres y si son mujeres en rojo y los agentes Anciano aparecen amarillo si son mujeres o en blanco si son hombres. Como podemos observar al principio sólo hay agentes niño pero en la siguiente captura ya se han hecho adultos o ancianos e incluso algunos de ellos ya han tenido descendencia.

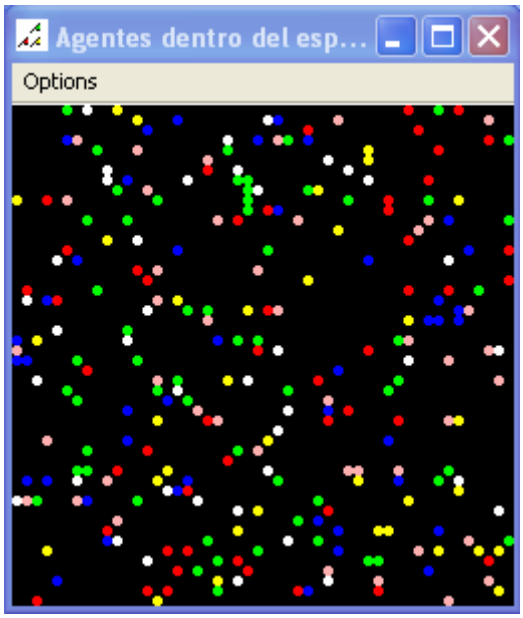


Ilustración 7. Evolución de la población.

Por otra parte, también podemos observar los histogramas que nos muestran como se distribuye la población según las distintas características:

Ilustración 8. Histograma de la forma física de la población.

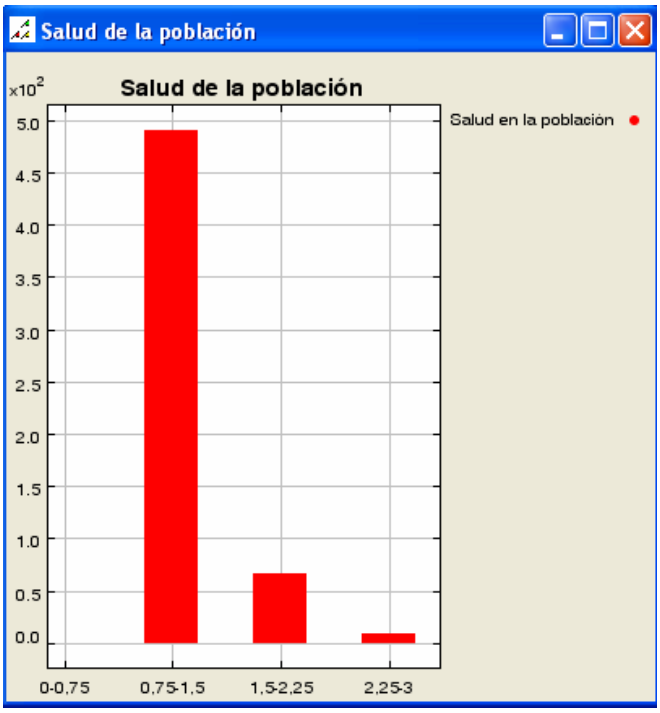


Ilustración 9. Histograma de la salud de la población.

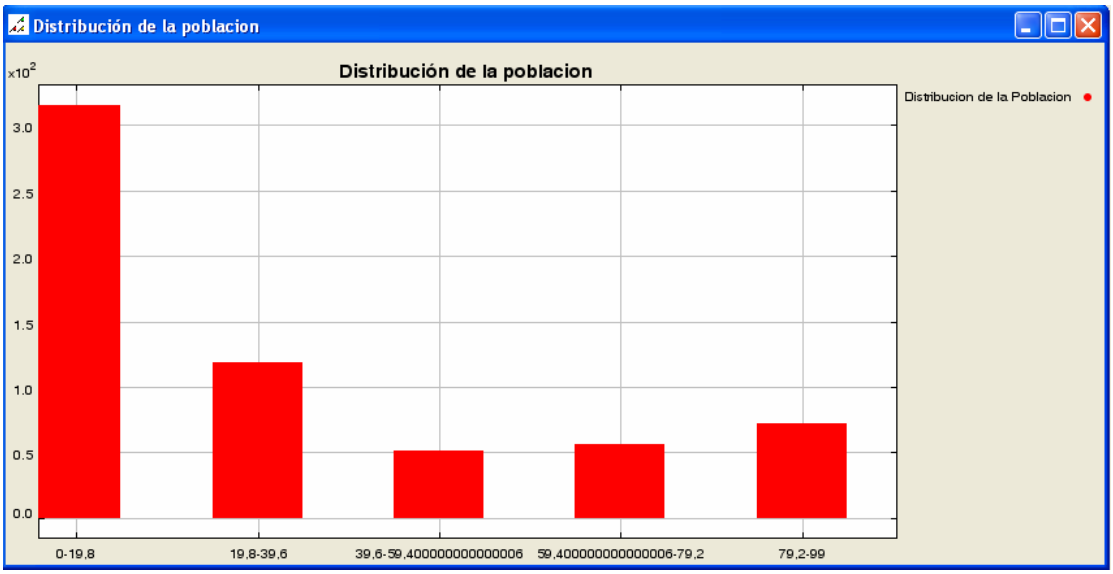


Ilustración 10. Histograma de la edad de la población.

Aparecen agrupados en intervalos de 0-20, 20-40, 40-60, 60-80 y de 80 a 99.

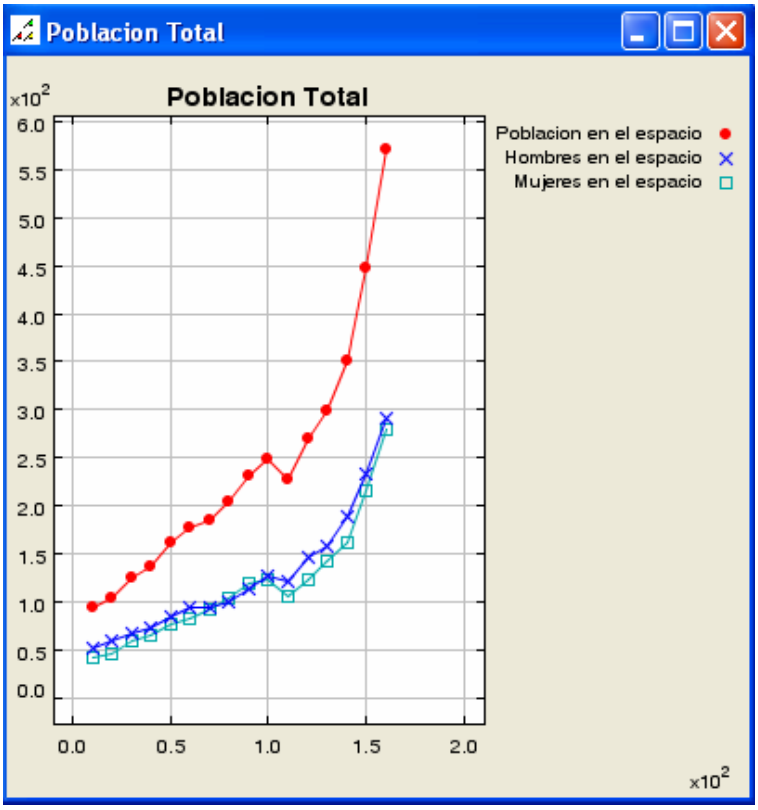


Ilustración 11. Distribución de la población por sexos.

En donde se puede observar que la población crece de manera exponencial.

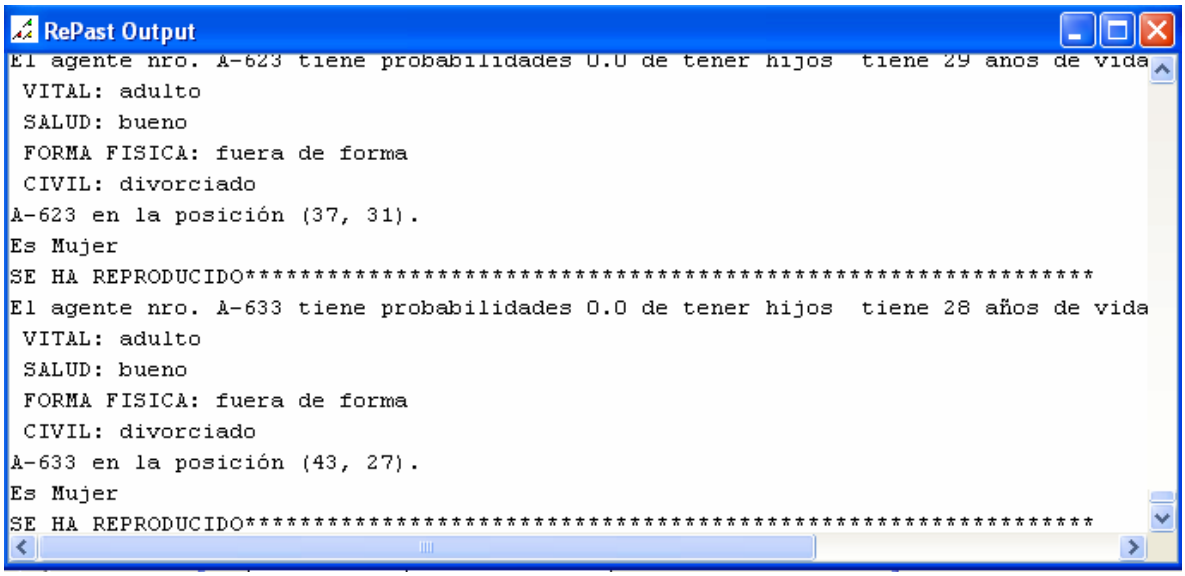


Ilustración 12. Report de los agentes.

6.2.2 Modelo de las energías.

El modelo general conseguido, es un modelo energético en el que se reflejan los roles de los agentes consumidores y productores que participan en el negocio energético. Todos los agentes están definidos por unas ciertas características que en cada momento se encuentran en un estado; dicho estado va evolucionando a lo largo del tiempo según indique la función de transición correspondiente a cada característica. En este primer modelo completo que logramos nos interesamos por los perfiles más extremos que pueden tomar los agentes del modelo energético. Nuestro momentum de simulación es una "fase" que se compone de 4 horas dentro de la franja diaria.

De este modo, nuestro modelo presenta agentes consumidores gastones y agentes consumidores ahorradores. En este primer modelo general nos referimos al uso que hacen estos agentes, que representan "unidades familiares", dentro del ámbito hogareño. Todos los agentes consumidores vienen caracterizados por la potencia consumida según el uso que hagan de los siguientes recursos:

- Agua caliente
- Frigorífico
- Horno Eléctrico
- Iluminación
- Lavadora
- Lavavajillas
- Microondas
- Ordenador
- Pequeños Electrodomésticos
- Secadora

- Televisor
- Vitro-cerámica o Cocina eléctrica

Dicho uso viene dado por las funciones de transición. Cada función de transición indica si en una cierta fase del día, cada día se compone de 6 fases de 4 horas cada una, el agente está utilizando dicho recurso o no. Las fases vienen dadas de la siguiente manera:

- Fase 1: de 12pm a 4am
- Fase 2: de 4am a 8am
- Fase 3: de 8am a 12am
- Fase 4: de 12am a 4pm
- Fase 5: de 4pm a 8pm
- Fase 6: de 8pm a 12pm

Hemos elegido estas fases ya que resultan bastante representativas, en cuanto a las actividades que se suelen acontecer en ellas relacionadas con el uso de la energía eléctrica:

Fase 1: Durante estas horas un consumidor tipo se encontraría descansando y por lo tanto no haría uso alguno de la energía eléctrica.

Fase 2: Esta franja horaria contiene las horas en que la gente se levanta y se prepara para abordar un nuevo día.

Fase 3: En la mayoría de los casos a lo largo de esta franja horaria no se hará consumo eléctrico alguno dentro del hogar debido a que la gente suele estar en el trabajo, colegio, universidad, etc.

Fase 4: Esta fase engloba las horas en las que se suele almorzar y por lo tanto habrá un consumo energético debido a las cocinas, microondas, etc.

Fase 5: Esta fase también registra un bajo consumo debido a que suele haber poca gente en sus hogares.

Fase 6: Durante estas horas se suele volver a casa y es cuando se registra un mayor uso de la iluminación y del ordenador así como también suele ser la hora de hacer la colada.

De esta manera, según estas suposiciones, hemos elaborado las funciones de transición de cada aparato que puede poseer o no un consumidor. Con lo que conseguimos una planificación del uso doméstico de estos aparatos como la que sigue:

Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Fase 6
	Agua Caliente				
Frigorífico	Frigorífico	Frigorífico	Frigorífico	Frigorífico	Frigorífico
			Horno Eléctrico		
	Iluminación				Iluminación
					Lavadora
		Lavavajillas		Lavavajillas	
					Microondas
					Ordenador
	Pequeño electrodoméstico				
			Vitrocerámica o cocina eléctrica		Vitrocerámica o cocina eléctrica
		Televisor	Televisor		Televisor
					Secadora

Tabla 1. Fases de los agentes consumidores.

En cada columna de la tabla aparecen los aparatos que estarán siendo utilizados por los agentes consumidores en el caso que los tengan. Como se puede observar, hay ciertos aparatos que consumen energía durante todo el día, por ejemplo un frigorífico necesita estar conectado a la red eléctrica durante todo el día para así poder mantener la temperatura de los alimentos.

En este modelo general la diferencia entre los consumidores gastadores y los ahorradores viene dada por la posesión o no de los aparatos; pero una aproximación más realista podría ser caracterizarlos también según el uso que hagan de un aparato en particular, es decir, tener distintas funciones de transición para una característica dada según el tipo de agente. Por ejemplo, dos agentes consumidores que tuvieran vitro-cerámica, podrían tener una potencia consumida muy diferente. Esto es, si uno de ellos fuese un amante de la buena cocina y aprovechase toda ocasión para hacer sus pinitos y otro, sin embargo, no confiase tanto en sus habilidades culinarias y se mantuviese alejado de los fogones. De este modo, la función de transición del cocinillas podría reflejar que en las fases correspondientes a los almuerzos su vitro-cerámica está a pleno funcionamiento, mientras el otro tipo de agentes no realiza uso alguno de ese aparato en dichas fases.

Así mismo, los agentes consumidores disponen de la característica “potencia consumida” que indica la potencia consumida en una cierta fase por el conjunto de aparatos que se encuentran funcionando en dicha fase.

El otro rol del mercado energético lo desempeñan los agentes productores. Cada productor posee una capacidad de producción que pueden llegar a cubrir o no dependiendo de la demanda energética de los consumidores, esta característica es estática, es decir, no hay una

función de transición que defina los estados por los que pasa a lo largo del tiempo; un productor, siempre que no esté averiado, va a tener dicha capacidad de producción. Así mismo, los productores vienen caracterizados por:

- Su potencia producida, que depende de la predicción de la necesidad energética que se basa en el consumo energético en la fase inmediatamente anterior a la actual.
- Su producción, que indica si el productor está "funcionando" o no. Un productor puede que no funcione debido a:
    1. La ausencia de demanda energética: dicha posibilidad no es realista debido a que las centrales se construyen en función de las necesidades que se van generando ya que no es rentable el alto coste del mantenimiento de un central de producción de energía eléctrica si no se compensa con el funcionamiento continuado de la instalación.
    2. La ocurrencia de alguna avería que imposibilite la producción.
    3. La realización de las actividades obligatorias de mantenimiento rutinario.

De este modo, en nuestro modelo, los productores están en funcionamiento en todas las fases con una producción que viene dada por la suma de todas las potencias consumidas de los agentes consumidores en la etapa inmediatamente anterior a la actual. En el modelo hemos incluido dos tipos de productores los productores primarios y los secundarios, de manera que, los primero poseen una capacidad de producción muy superior a los secundarios.

La interacción de agente-agente tan característica de un sistema multiagente viene reflejada en nuestro modelo, como ya hemos mencionado, de manera que los agentes productores ajustan su producción según la necesidad energética que los consumidores han mostrado anteriormente, en concreto, en la fase anterior. En realidad, el organismo encargado de esta tarea en España, llamado Red Eléctrica Española, basa la predicción del consumo energético en datos históricos de años anteriores aunque también tengan en cuenta otros factores más complejos que consiguen un ajuste mucho mejor. El fin de todo ello es hacer un ajuste óptimo de la oferta-demanda de energía en tiempo real, ya que la energía eléctrica no es un recurso que pueda almacenarse para su posterior uso, como podría ocurrir con el agua, sino que la energía eléctrica se genera, se transporta y se consume de manera totalmente continua y aquélla energía generada que no se consume queda en pérdida.

Todos nuestros agentes se encuentran localizados en espacios. El concepto espacio es muy general, no sólo puede referirse a un espacio geográfico sino también a un espacio climatológico (diferenciando así, por ejemplo, espacio con clima tropical de espacio con clima desértico) o un espacio político (diferenciando un espacio liberal de otro más conservador). De manera que, dichos tipos de espacios pueden solaparse teniendo zonas liberales y conservadoras dentro de un espacio con clima tropical, por ejemplo. En este modelo de energías genérico, en concreto tenemos un único espacio geográfico que suponemos de dos dimensiones y todo agente posee una situación dentro de él.

6.3 UML

6.3.1 Diagramas de paquetes.

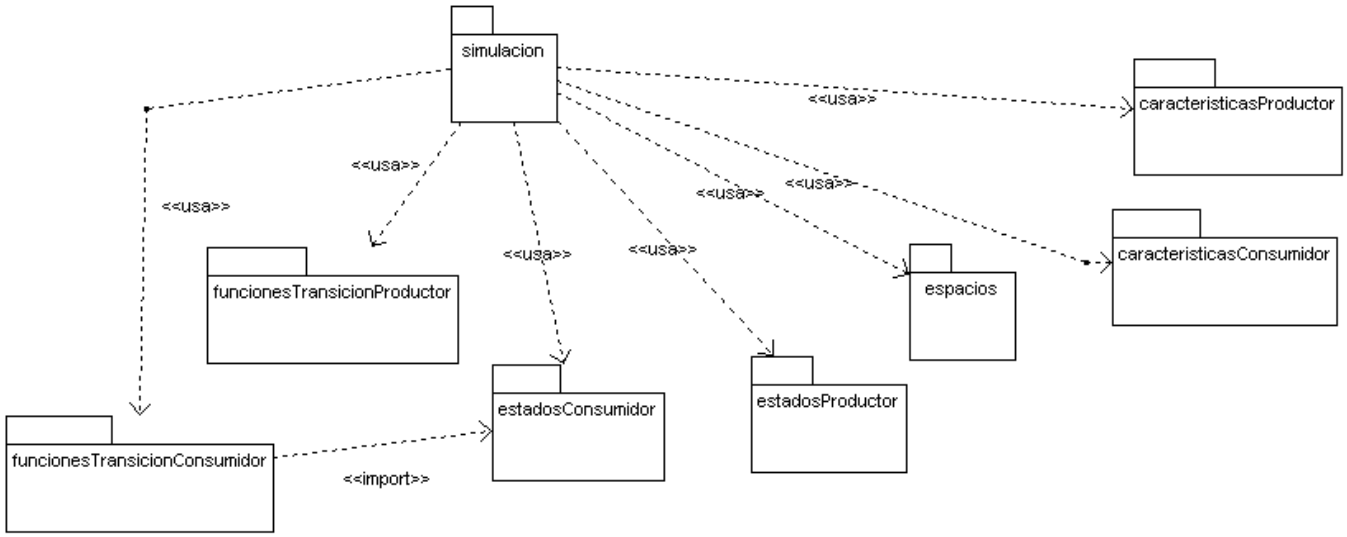


Ilustración 13. Diagramas de paquetes de la estructura del modelo.

6.3.2 Diagramas de clases.

El siguiente diagrama muestra las clases relacionadas con un agente. Como se puede observar todo nuestro modelo gira alrededor de la clase agente. Cada agente posee unas características que vienen determinadas por el estado en el que están y por la función de transición que permite cambiar dicho estado. Así como también un agente viene determinado por la situación en la que está. Dicha situación indica cada uno de los espacios en los que compatiblemente el agente puede estar. Por ejemplo, un agente puede estar en un "espacio de salud bueno" y a la vez en un "espacio educativo bueno"; si hablamos en términos del modelo del ciclo de vida.

En este diagrama también se puede apreciar el uso que hacemos del patrón "factoría abstracta" para la clase *FactoriaEspacio* así como el uso del patrón factoría para crear los objetos de todos los elementos que componen nuestro modelo. Por último, cabe destacar el patrón "composición" aplicado a la clase *FuncionTransicion*.

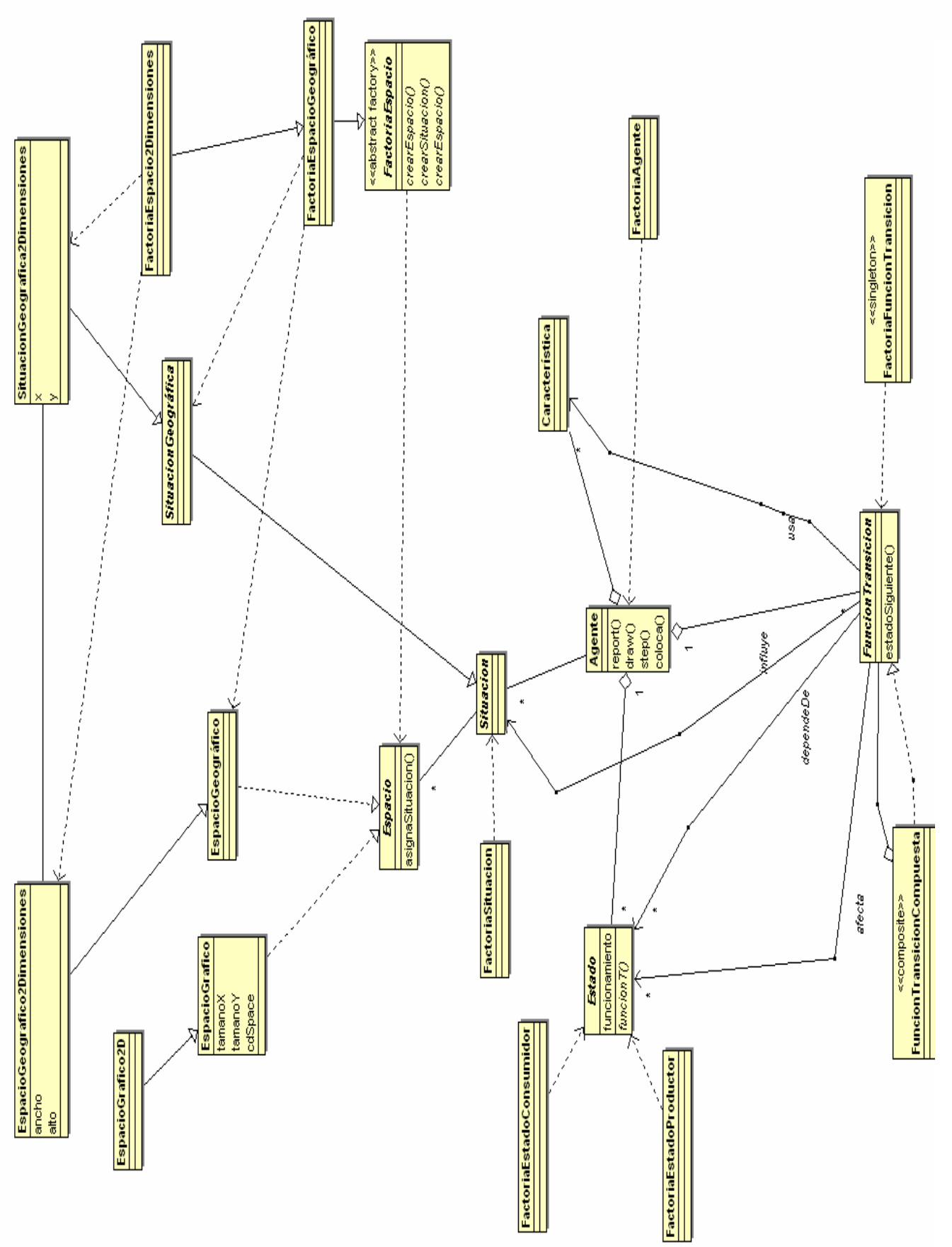


Ilustración 14. Diagrama de clases de las características relacionadas con el agente.

El diagrama que aparece abajo muestra la jerarquía de tipos de agentes que tenía la versión inicial de nuestro modelo de energías. De modo que los consumidores pueden ser gastadores o ahorradores, esto dependerá de los aparatos eléctricos de los que dispongan, y los productores pueden ser primarios si tienen una capacidad de producción eléctrica considerable o secundarios si su capacidad de suministro es más modesta.

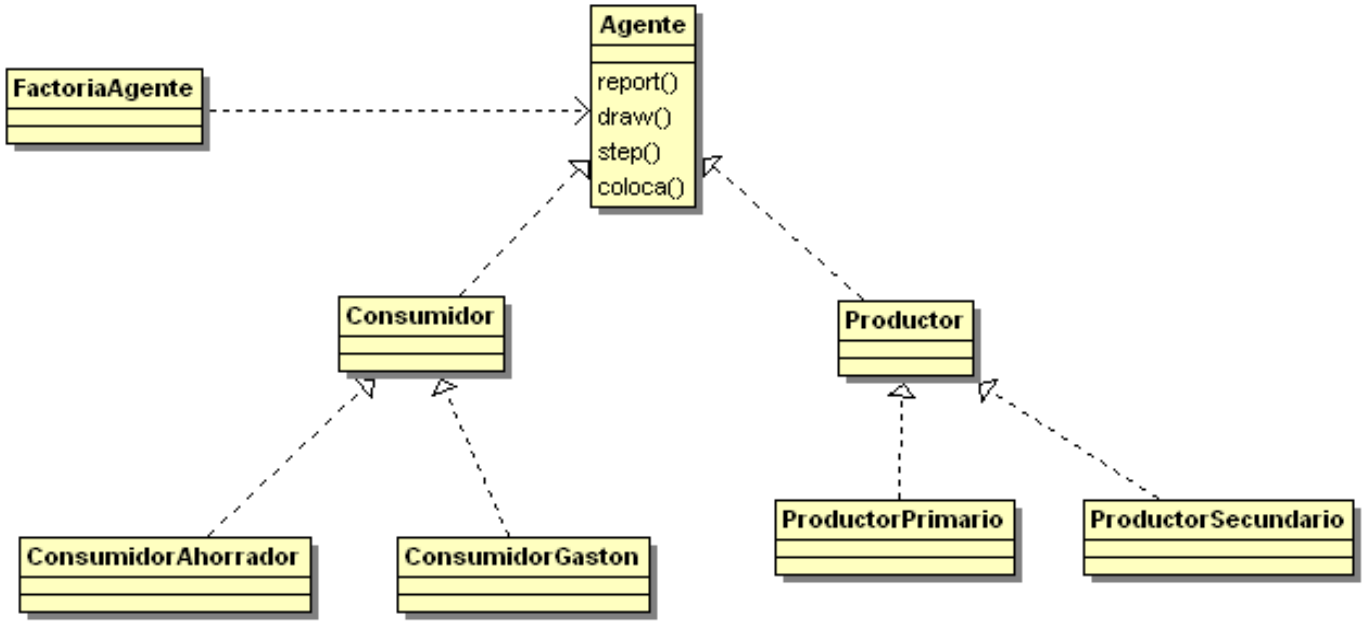


Ilustración 15. Diagrama de clases de los tipos de agente.

En el diagrama posterior mostramos lo relacionado con el estado de un agente y los tipos de estados que puede tener un agente consumidor. Un agente puede tener varios estados, de hecho, va a tener uno por cada una de sus características de manera que representan el valor que en un cierto momento toman dichas características. Para modificar dicho estado el agente tiene que hacer su step y ejecutar las funciones de transición para cada una de sus características.

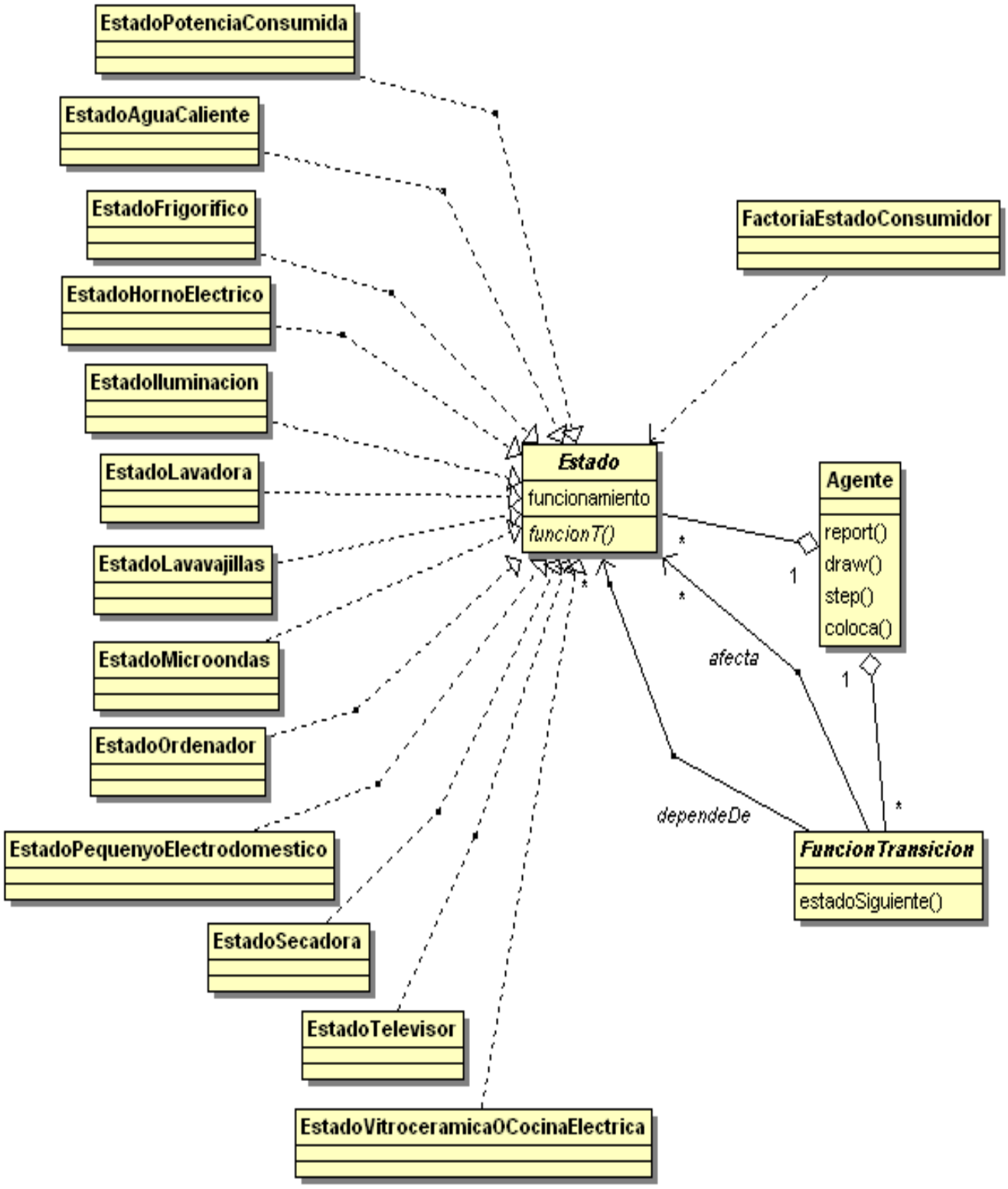


Ilustración 16. Diagrama de clases de los estados de un agente.

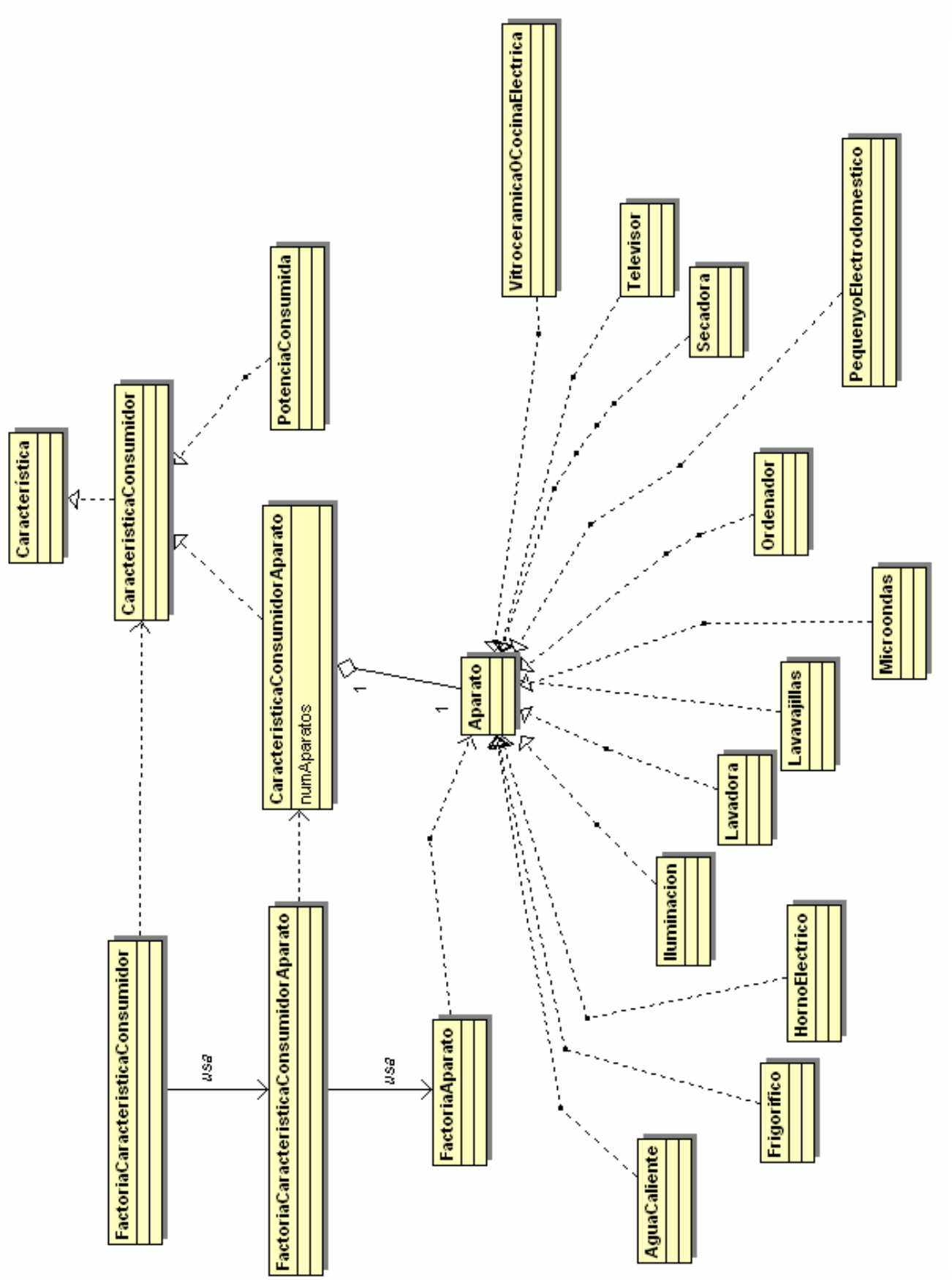


Ilustración 17. Diagrama de clases de las características que pueden determinar a un agente consumidor.

En el diagrama que aparece arriba se refleja todo lo relativo a las características que pueden determinar a un agente consumidor. Los agentes consumidores tienen dos clases de características: los aparatos que poseen y la potencia consumida por el uso de dichos aparatos.

En el siguiente diagrama viene reflejado todo lo relativo a las características que pueden determinar a un agente productor. Un agente productor viene determinado por su producción, es decir, si dicha central de generación de energía eléctrica está en funcionamiento o si, por el contrario, está en paro por revisión o por avería. Por otra parte, un agente productor también viene determinado por su potencia producida en una cierta fase que va a depender de la potencia consumida en la fase inmediatamente anterior.

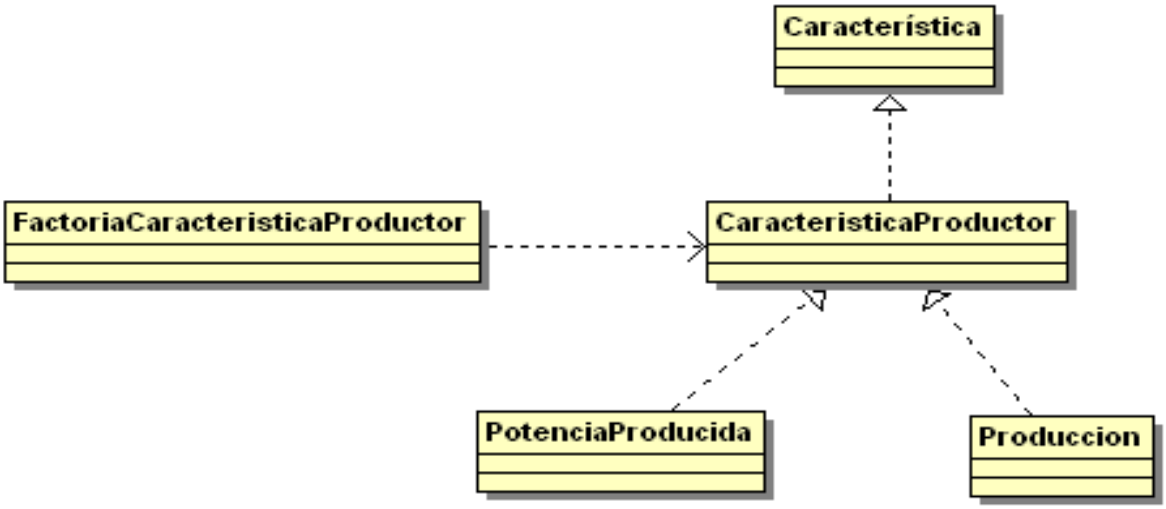


Ilustración 18. Diagrama de clases de las características de un agente productor.

Como podemos ver en el diagrama de clases que aparece a continuación, un agente consumidor tiene una función de transición para cada una de sus características, de forma que permite cambiar el estado de cada una de ellas.

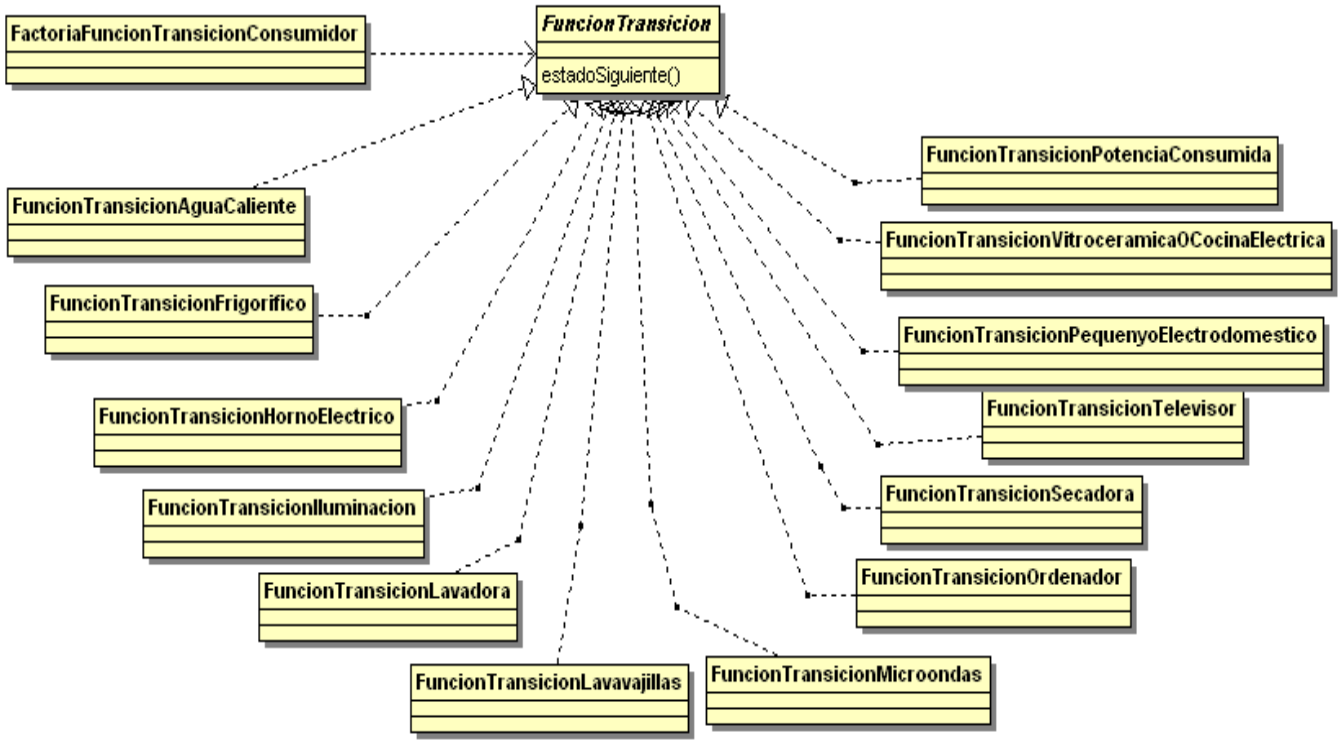


Ilustración 19. Diagrama de clases de las funciones de transición de un agente consumidor.

Al igual que los agentes consumidores, los agentes productores también poseen una función de transición para cada una de sus características.

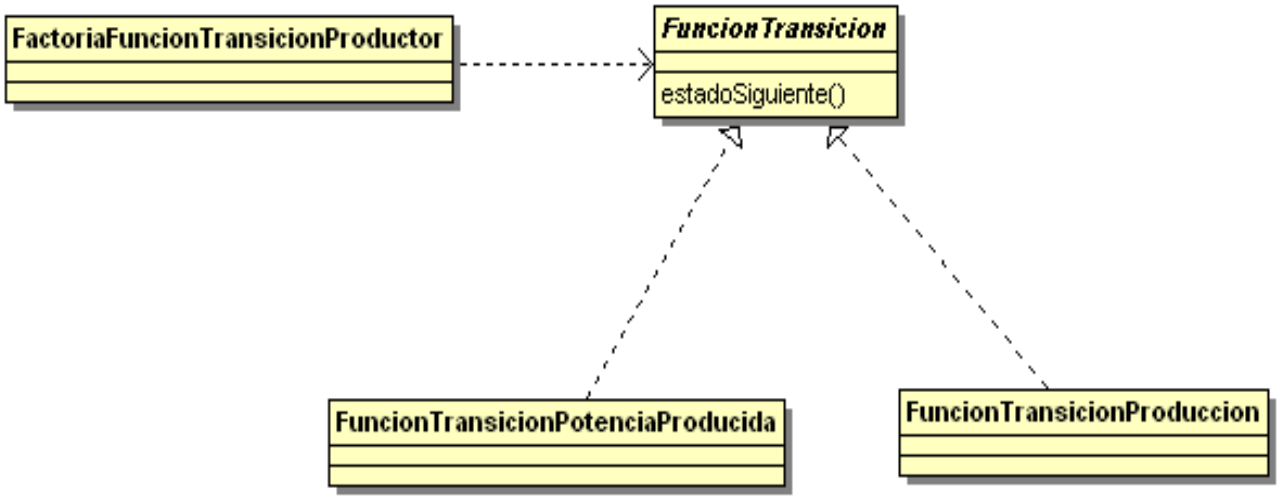


Ilustración 20. Diagrama de clases de las funciones de transición de un agente productor.

Por último, vemos los tipos de estados que puede tener un agente productor; que son los correspondientes a las características que éste tiene.

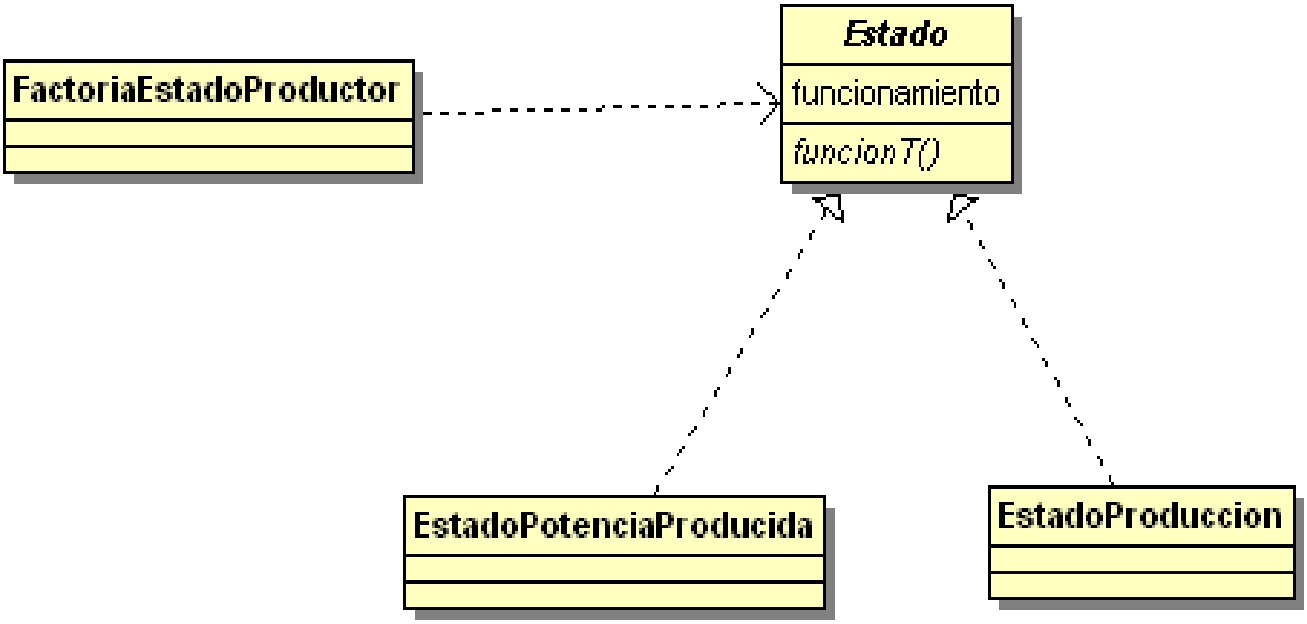


Ilustración 21. Diagrama de clases de los estados de un agente productor.

6.3.3 Diagramas de secuencia o interacción.

En el diagrama inferior mostramos la secuencia de comandos necesaria para la inicialización de todos los agentes que intervienen en nuestro modelo energético así como de los espacios en los que éstos se encuentran.

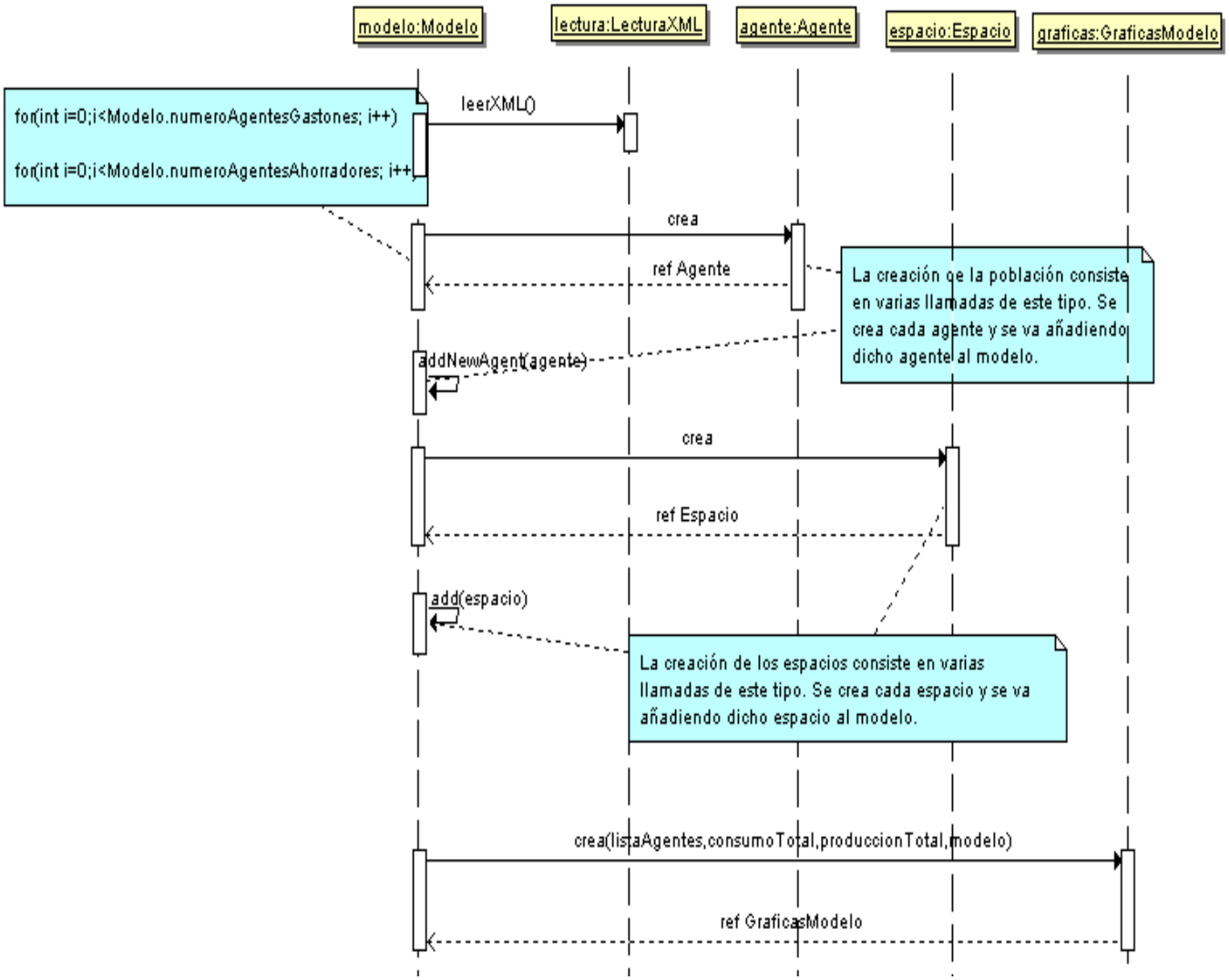


Ilustración 22. Diagrama de interacción de la inicialización del modelo.

El siguiente diagrama refleja la secuencia de acciones necesarias para realizar un *step* de un agente consumidor. Primero se actualizan los aparatos que el consumidor tiene en funcionamiento en ese instante y después se calcula la potencia consumida por todos ellos.

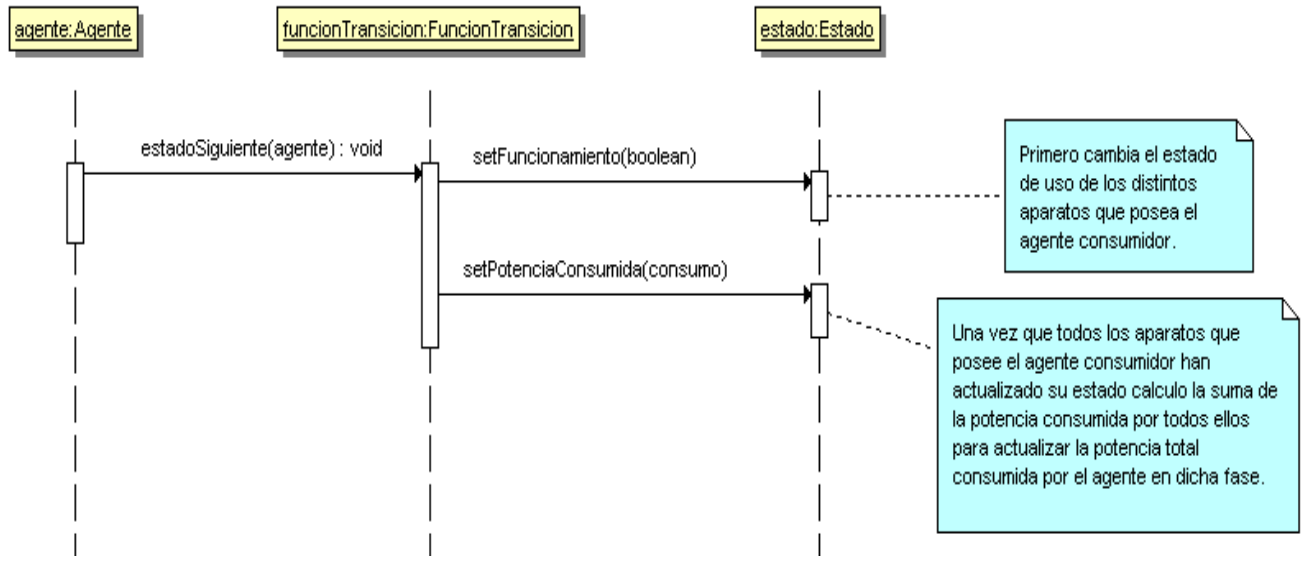


Ilustración 23. Diagrama de interacción del step de un agente consumidor.

En el diagrama que aparece a continuación aparece la secuencia de acciones necesarias para realizar un *step* de un agente productor. De modo que el *scheduler* del modelo le indica al productor que tiene que hacer su *step* y, a su vez, el agente productor llama a todas sus funciones de transición para que se actualice el estado de sus características.

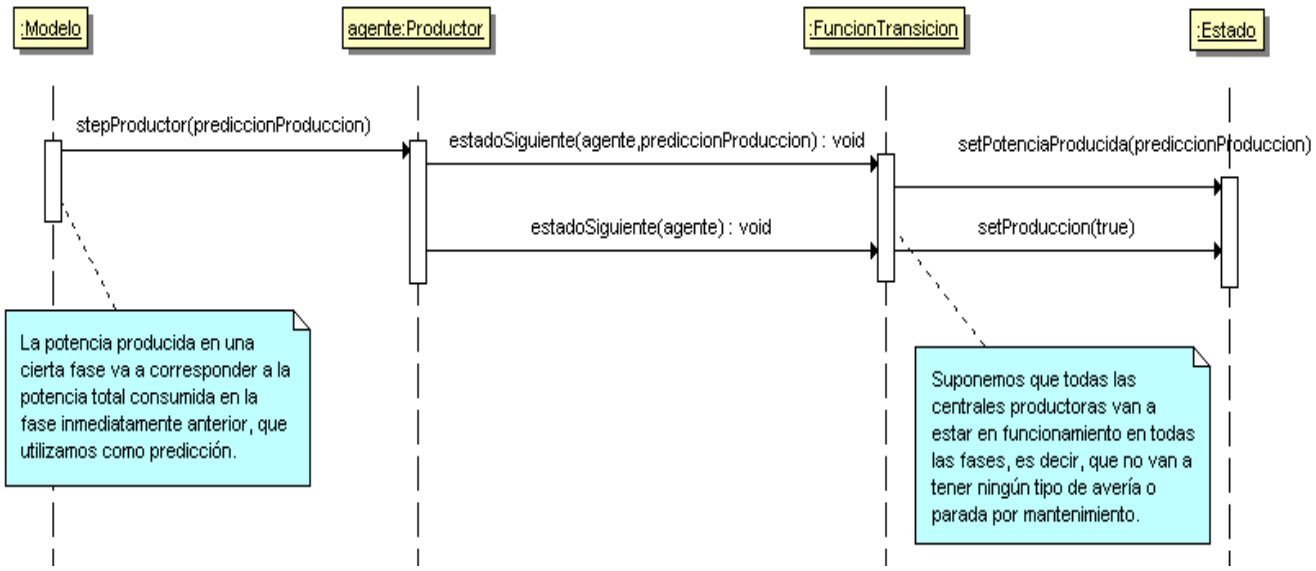


Ilustración 24. Diagrama de interacción del step de un agente productor.

Por último, en este diagrama mostramos la secuencia de acciones necesarias para inicializar la situación de los agentes en los distintos espacios del modelo. Es el propio *scheduler* del modelo el que le dice a cada agente que tiene que inicializar su situación y cómo inicializarla, de modo que, le pasa las coordenadas que va a ocupar dentro de un cierto espacio.

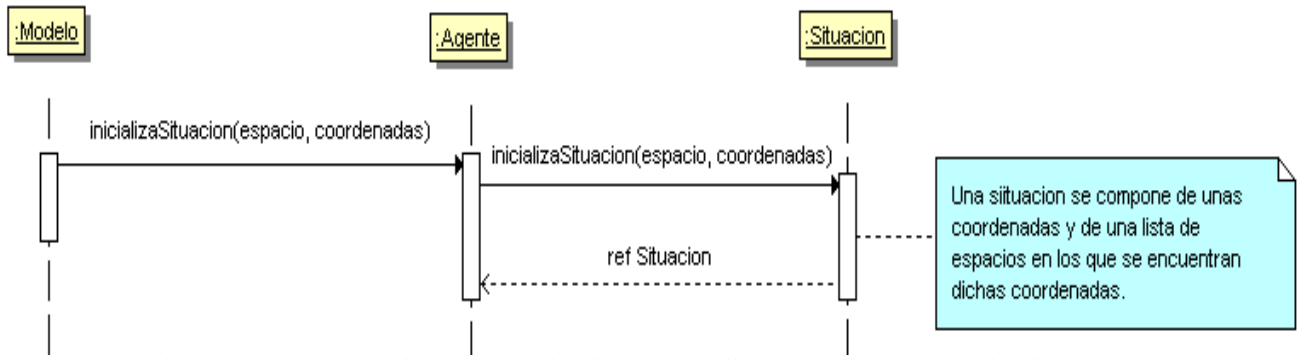


Ilustración 25. Diagrama de interacción de la inicialización de la situación de un agente.

6.4 Detalles de implementación

4.4.1. Archivo de configuración

Para pasarle al modelo general la población inicial utilizamos un archivo de configuración XML en el que indicamos las dimensiones X e Y del espacio en el que se van a encontrar los agentes así como el número de agentes de cada tipo. De modo, que dicho archivo tiene este aspecto:

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<configuration>
      <properties>
            <property name="numeroAgentes" value="12"/>
             <property name="WORLDXSIZE" value="88"/>
            <property name="WORLDYSIZE" value="88"/>
      </properties>
      <modelPopulations>
            <agents agentType="ConsumidorGaston" amount="6"/>
            <agents agentType="ConsumidorAhorrador" amount="6"/>
            <agents agentType="ProductorPrimario" amount="0"/>
            <agents agentType="ProductorSecundario" amount="0"/>
      </modelPopulations>
</configuration>
```

Ilustración 26. Archivo de configuración XMl.

El propósito del archivo de configuración es no generar la población inicial aleatoriamente sino elegirla y poder reutilizarla, de manera que nos facilite el análisis de resultado al elegir distintas poblaciones iniciales. La lectura de dicho archivo se ha realizado utilizando SAX.

4.4.2. Guía de instalación

Nuestra aplicación requiere tener instalada en su máquina la versión 1.5 de JDK y la versión 3 de RepastJ que puede descargarse gratuitamente en la página:

http://repast.sourceforge.net/download.html

Para ejecutar nuestro proyecto son necesarias las librerías repast.jar y aquéllas incluídas en la carpeta lib de Repast.

6.5 Escenario1: Evolución del consumo afectado por campañas publicitarias.

Tomando como punto de partida el diseño general del sistema de energía eléctrica, se ha estudiado este primer escenario centrado en la influencia de las campañas en el comportamiento de los agentes en cuanto a su consumo.

6.5.1 Explicación del modelo

En este escenario se estudian dos características fundamentalmente:
- En primer lugar la influencia de las campañas publicitarias.
- En segundo lugar la influencia de la vecindad.
- En tercer lugar la conjugación de los factores anteriores

En primer lugar lo común al estudio de todas las variables es la definición de cuatro tipos de consumidores:

- Consumidores sin actitud social no influenciables (ConsuminorNSNI) que representan a familias que no tienen actitud ecologista (de ahorro de engergía eléctrica) y que son poco proclives a cambiar esa actitud.
- ConsumidoresNSI, igual que los anteriores pero además son influenciables , es decir será más fácil que cambie su actitud de consumo. Por así decirlo este tipo de consumidores es más sensible a las campañas publicitarias.
- Finalmente los consumidores SI y SNI respectivamente influenciable y no influenciable. Estos dos consumidores a diferencia que los anteriores si tienen actitud social. Son, de alguna manera, más sensibles al problema del ahorro energético. La idea es que su actitud cambiará muy poco a lo largo de la simulación ya que de partida si tienen sensibilidad social.

En el dibujo podemos ver las cuatro clases de consumidores
-De verde los ConsumidoresSNI
-De amarillo los ConsumidoresSI
-De rojo los ConsumidoresNSNI
-De rosa los ConsumidoresNSI

Ilustración 27. Población inicial con los 4 tipos de agentes.

Para tener en cuenta la influencia de campañas publicitarias se ha definido la clase Campanya. Una campaña vendrá definida por dos características fundamentales: en primer lugar el momento de la campaña. En este modelo se ha introducido una variable time que cambia con cada ciclo de reloj del modelo de Repast. Esta variable controla el cambio de estación y el step de los agentes y de la agenda.

El otro elemento a tener en cuenta es la intensidad de la campaña. Veremos a continuación un ejemplo de simulación en el que ocurren dos campañas publicitarias. En este ejemplo hay una campaña en el momento 50 y en el momento 100. Vemos en los dos gráficos a continuación como les afecta en principio a los influenciables la campaña y a los no influenciables no.

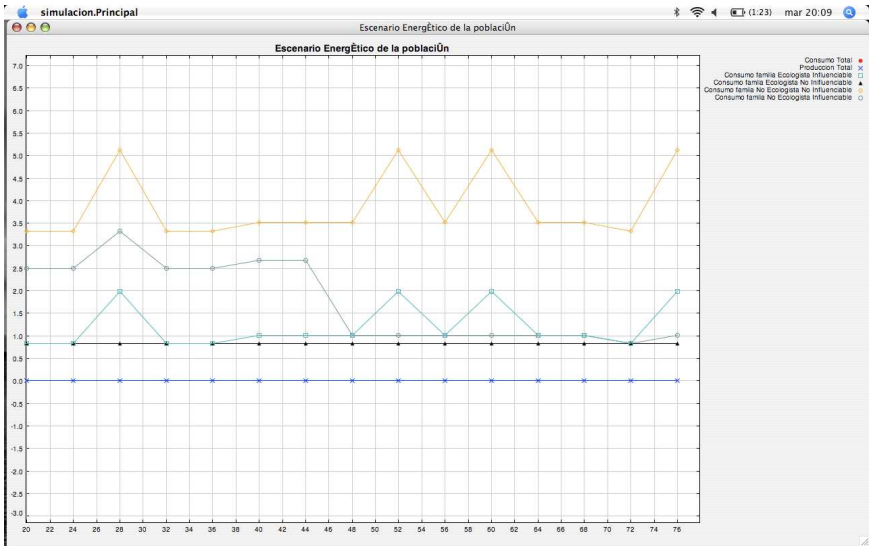


Ilustración 28. Consumo de los 4 tipos de familias tras una campaña publicitaria.

Después simulando comportamientos extremos la segunda campaña es de intensidad más fuerte y cambia el consumo de los no influenciables. Pero en mucha menor medida y tras la segunda campaña. Los Consumidores NSNI siguen siendo los que más consumen.

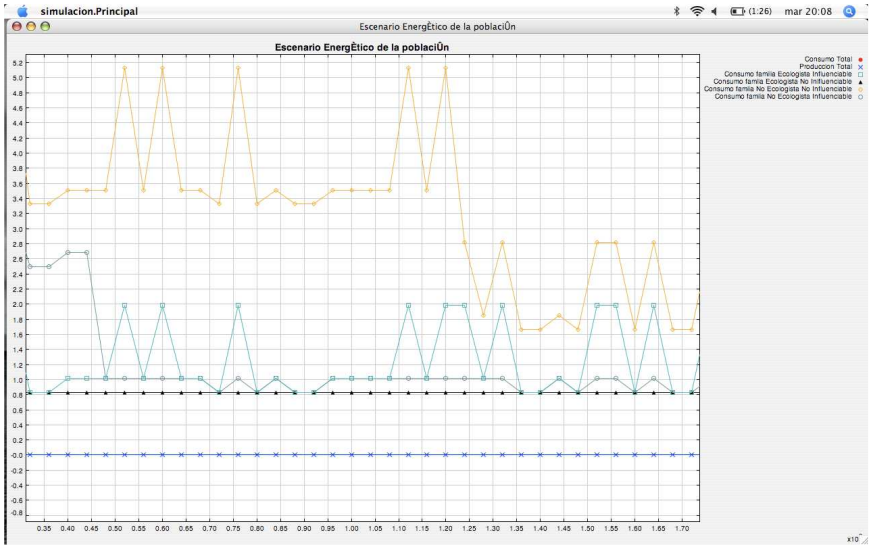


Ilustración 29. Consumo de los 4 tipos de familias tras una campaña publicitaria de mayor intensidad.

Vemos el consumo total de la primera y de la segunda campaña.

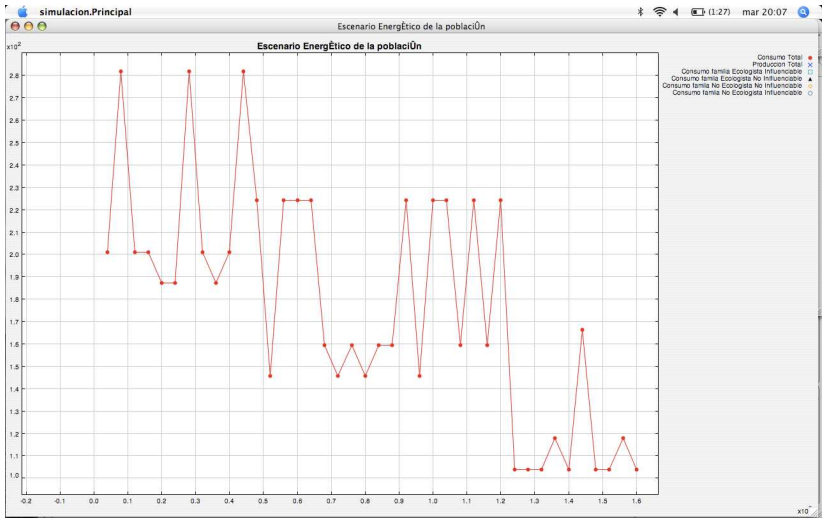


Ilustración 30. Consumo total de la población.

El siguiente factor que estudiaremos será la influencia del vecindario. Para ello partimos de una sociedades de agentes influenciables en el que se les introduce tres familias. Tal y como se ve en la imagen.

Ilustración 31. Población inicial para 3 agentes con actitud social en una comunidad influenciable.

Tras unos pasos en el sistema algunos agentes han cambiado de actitud pasan de NSI a SI.

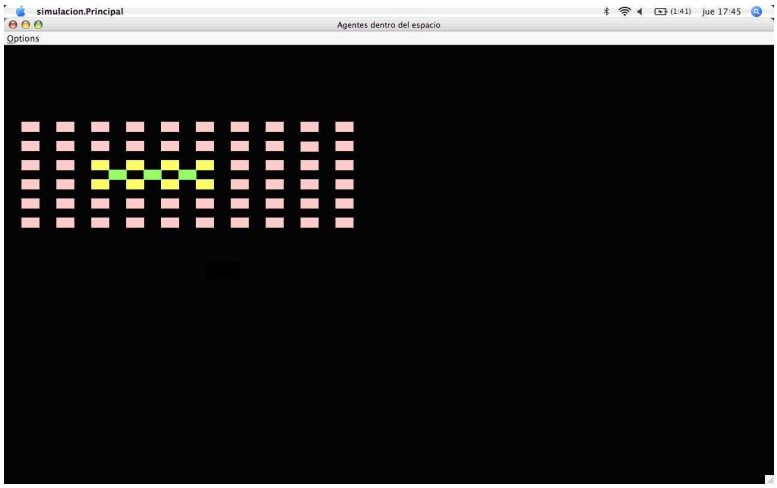


Ilustración 32. Población tras unos pasos de simulación cambia la actitud en algunas familias.

Finalmente vemos que si se deja en marcha el sistema todos los agentes cambian de actitud. Siguen siendo influenciables pero se han vuelto socialmente sensibles.

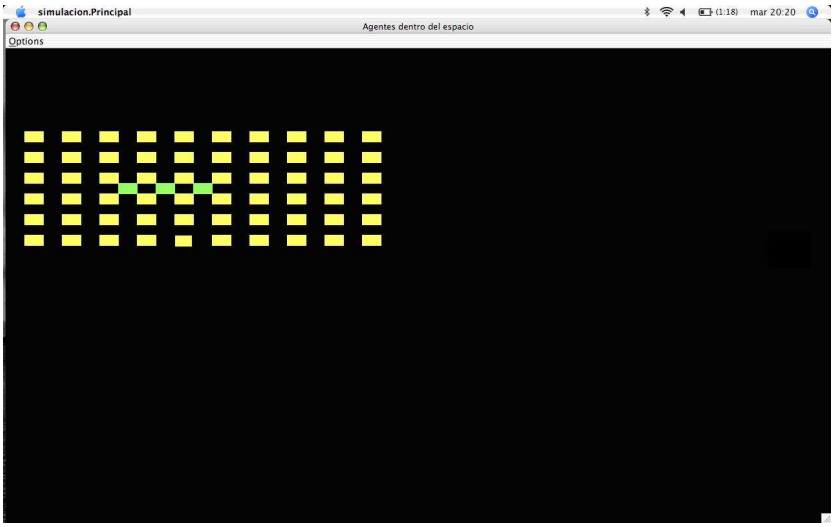


Ilustración 33. Al final de este modelo toda la comunidad influenciable cambia de actitud.

Y el consumo cambia en total según la siguiente gráfica. En primer lugar se nota una pequeña bajada y luego un cambio en todos los agentes.

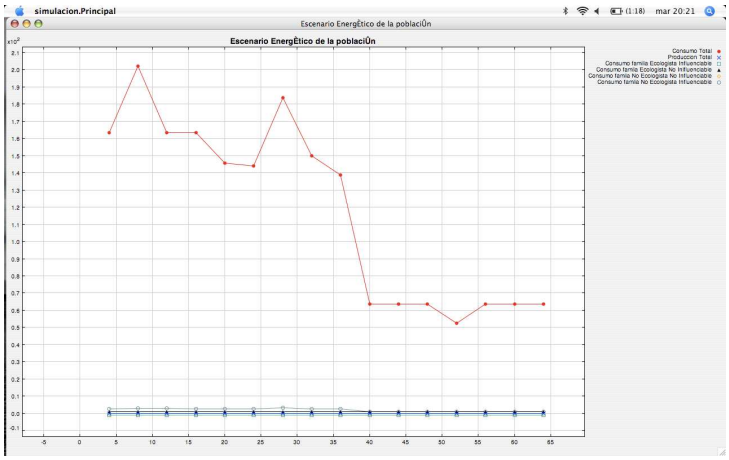


Ilustración 34. Consumo total de este submodelo.

Finalmente el estudio de las dos características a la vez, hay interacción entre agentes y campañas externas.

Partimos de una población como muestra el dibujo y hay una única campaña publicitaria.

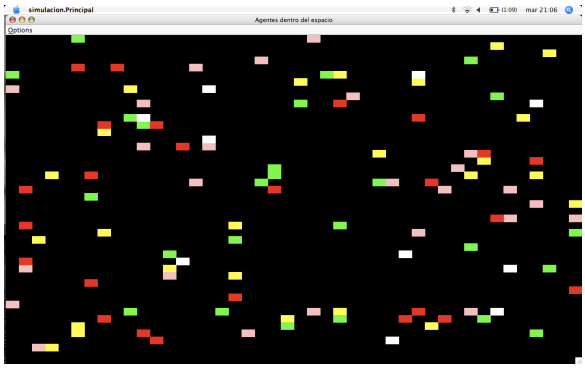


Ilustración 35. Población inicial del tercer submodelo.

Vemos como cambia los agentes y cómo es más importante en este modelo el efecto de la publicidad que la influencia entre agentes. Se observa en la gráfica más abajo como los agentes han cambiado.

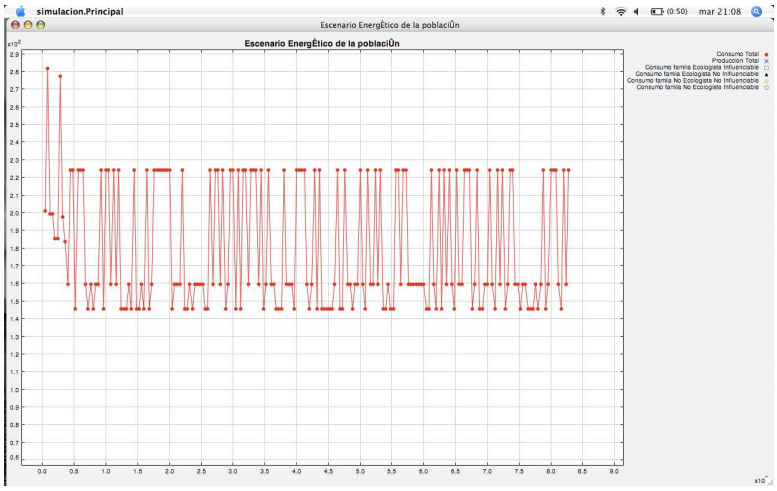


Ilustración 36. Consumo total del tercer submodelo.

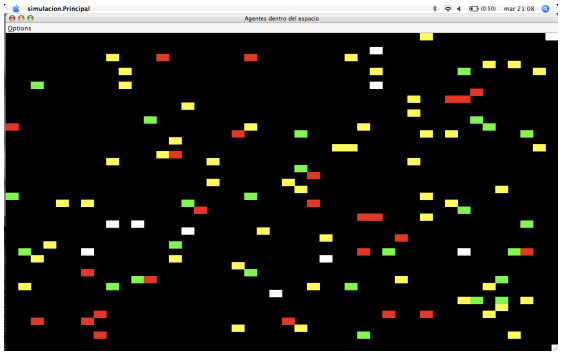


Ilustración 37. Población final, algunos agentes han cambiado de actitud.

6.5.2 UML

Este escenario consta de tres submodelos:

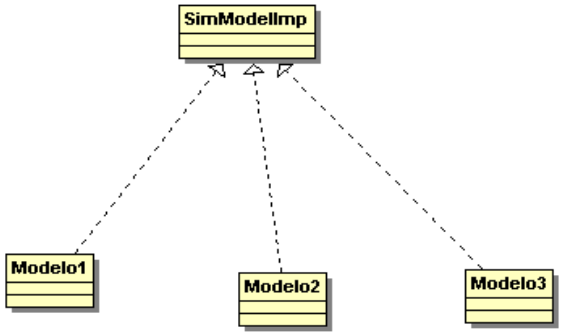


Ilustración 38. Los tres submodelos.

En el primer submodelo se tiene en cuenta el estudio de las campañas publicitarias. En el segundo submodelo se tiene en cuenta la relación Agente-Agente, sería un estudio de la influencia de la vecindad en el comportamiento de nuestras familias consumidoras.

El tercer submodelo tiene en cuenta los dos factores a la vez. En el tercer submodelo hay campañas publicitarias y a la vez unos agentes influyen en otros en el consumo eléctrico.

Por otro lado desarrollamos cuatro tipos de consumidores, que se muestran en el diagrama de clases posterior.

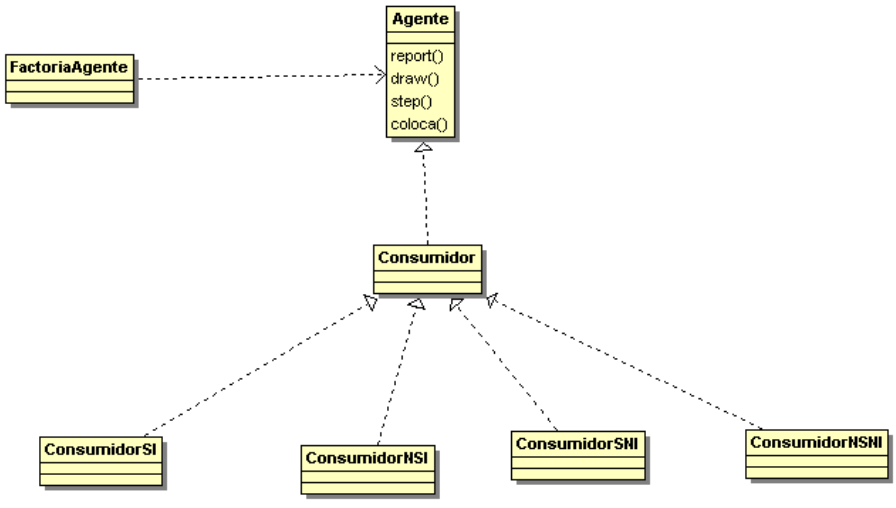


Ilustración 39. Diagrama de clases con los cuatro tipos de consumidores

Los consumidores tipo SI son consumidores con actitud ecologista, que son más influenciables. Los consumidores tipo SIN son consumidores con actitud social positiva, es decir que consumen poco y además son influenciables, será más fácil que cambien de actitud.

Los consumidores tipo NSI son aquellos que consumen mucho (no tienen actitud social positiva) y son influenciables. Los NSNI son aquellas familias que consumen mucho y tenderán a no cambiar su actitud.

6.5.3 Análisis de datos

Partiendo de una población en la que hay una única campaña publicitaria se refleja el efecto que tienen en los agentes influenciables que los que no. Se pone de manifiesto que el hecho de tener una característica que lo coloca como influenciable hace que reacciones con más facilidad a los cambios en su consumo mientras que los del tipo no influenciables no ocurre lo mimo.

6.5.4 Conclusiones y posibles ampliaciones

1) Esta arquitectura es útil para representar el cambio de una o varias variables a la vez.

2) En los ejemplos vistos anteriormente se da más importancia a la publicidad que al entorno. Es decir en el modelo influye más el entorno que los agentes entre sí.

Podemos definir modelos de consumo de manera sencilla, así como cambios sociales en la gente que sean útiles para situaciones reales.

6.6 Escenario2: Factores ambientales que modifican el consumo.

Realizamos un estudio del modelo genérico adaptándolo a otro que se centra en la evolución temporal del consumo partiendo del modelo genérico que tenemos como arquitectura del sistema salvo por cambios para adaptarlo al nuevo escenario.

6.6.1 Explicación del modelo

En el segundo escenario se analiza la influencia que tiene en la evolución del consumo de una población los cambios de estación: primavera, verano, otoño e invierno. Estos cambios de estación pueden influir en el consumo de manera significativa mediante la compra de más electrodomésticos que consuman mayor electricidad y utilizándolos en épocas del año que les hagan falta. Esto ocurre en países como España en donde las estaciones son los períodos del año en los que las condiciones climáticas imperantes se mantienen dentro de un cierto rango. Estos periodos duran aproximadamente tres aunque en las regiones de la tierra cercanas al ecuador

como por ejemplo los países que se localizan en el Caribe las estaciones son sólo dos, la estación seca y la lluviosa ya que en ellas varía drásticamente el régimen de lluvias, pero no así la temperatura.

En la figura siguiente se intenta modelar este cambio en el consumo debido a la compra de electrodomésticos que se utilizan o no dependiendo de la estación, afectando a la evolución temporal del consumo.

Se muestra como la conducta ahorradora de una persona refleja que aunque su consumo aumenta, lo hace en menos medida que una conducta gastadora y todo esto, en conjunto, refleja un aumento significativo en el consumo durante las estaciones de invierno (utilización de un mayor número de electrodomésticos para dar calor) y verano (utilización de un mayor número de electrodomésticos para dar frío).

Para mostrar lo que hemos añadido al diseño para modelar este escenario mostramos los diagramas UML que explican sólo las nuevas clases que se han introducido.

6.6.2 UML

Respecto de la arquitectura genérica del sistema se han añadido las siguientes clases:

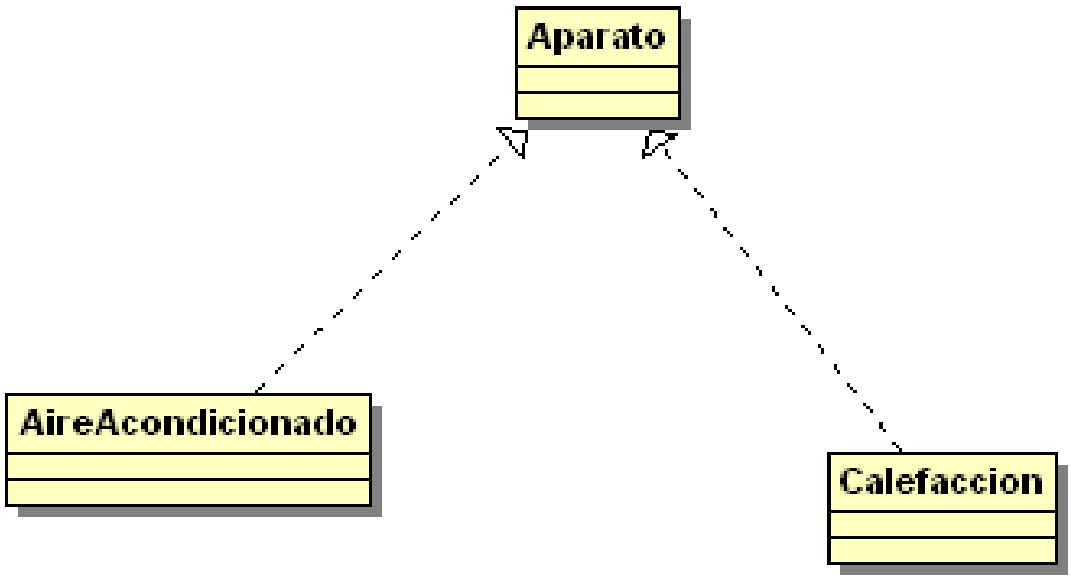


Ilustración 40. Diagrama de clases de los nuevos aparatos añadidos para el escenario 2.

Ya en la arquitectura genérica del sistema contábamos con distintos aparatos que contribuían al consumo de cada uno de los agentes. En este escenario se han añadido dos clases de aparatos más que influyen también en el consumo y que su utilización es dependiente en gran parte por la estación por la que el país se encuentre. Está claro que si una persona siente frío no

necesita para solucionar su problema a una tostadora o un televisor. Sin embargo, esta persona sí que buscará comprarse una estufa eléctrica para poder solventar su problema de frío. Con lo anterior se muestra la necesidad de incluir aparatos de consumo eléctrico que tengan una dependencia real de utilización según la estación que sea.

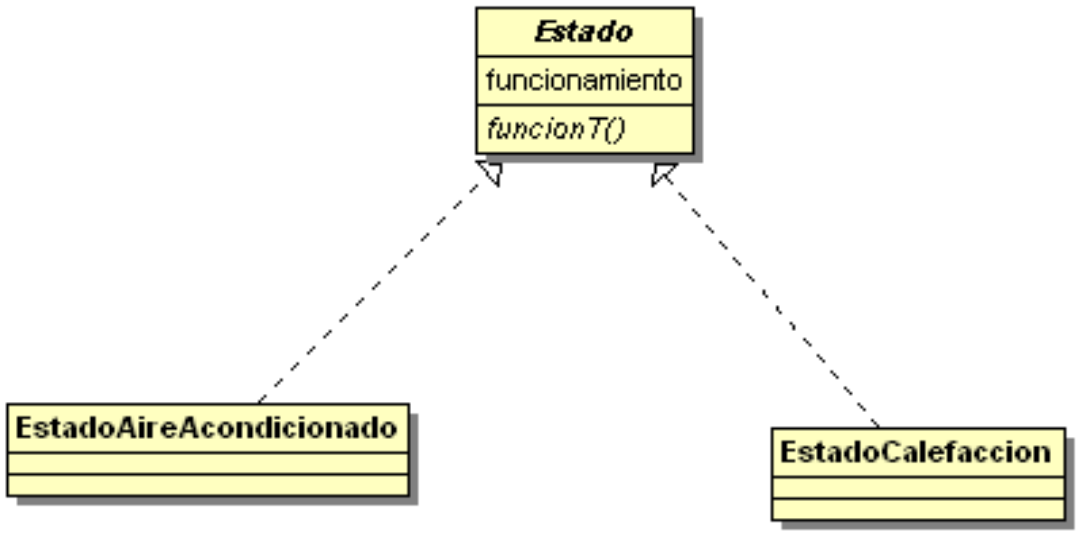


Ilustración 41. Diagrama de clases con los nuevos estados añadidos para el escenario 2.

En la figura anterior, análogamente a los aparatos añadidos, será necesaria también la inclusión de los estados del agente que modifican el consumo que hace cada agente de dichos aparatos mencionados anteriormente.

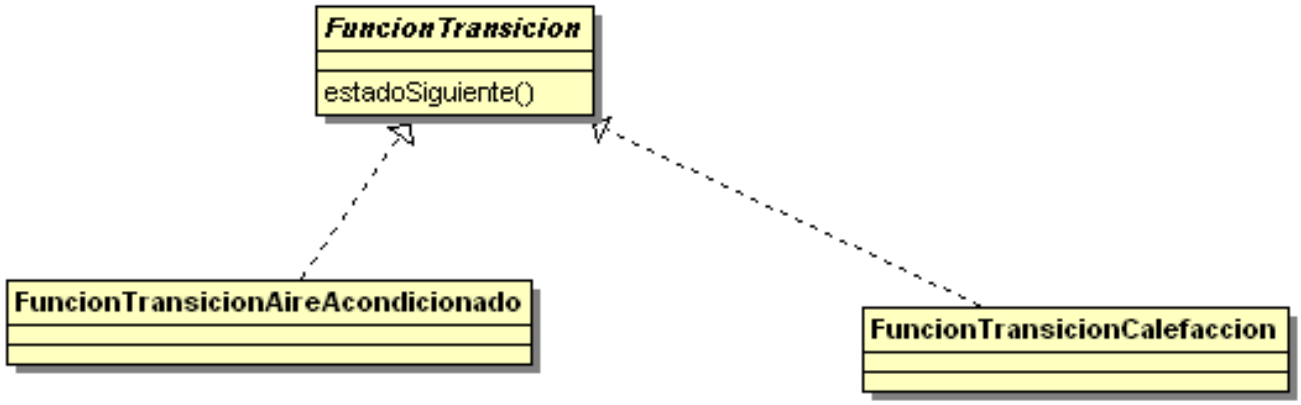


Ilustración 42. Diagrama de clases de las nuevas funciones de transición añadidas para el escenario 2.

En esta última figura se reflejan las funciones de transición de estado relacionadas con los estados que se han creado para gestionar los cambios de estado del agente con respecto a la utilización por parte de los agentes de sus aparatos.

6.6.3 Análisis de datos

Los resultados muestran que en los meses en los que más se consume es en los meses que corresponden a las estaciones de verano y invierno que cuando es necesario hacer un mayor uso de distintos aparatos debido al clima que se tiene en el que va los cambios son sustanciales y requieren una modificación significativa en el comportamiento del consumo de cada uno de los agentes.

Otro rasgo que se pone de manifiesto es que de los dos tipos de agentes que existen en este modelo, gastadores y ahorradores, ambos aumentan el consumo pero en los gastadores, dada su tendencia a consumir sin preocuparse de que están gastando en exceso, es más acusada que el aumento que ocurre en los agentes ahorradores que, aunque su consumo aumente también, lo hace en menor medida.

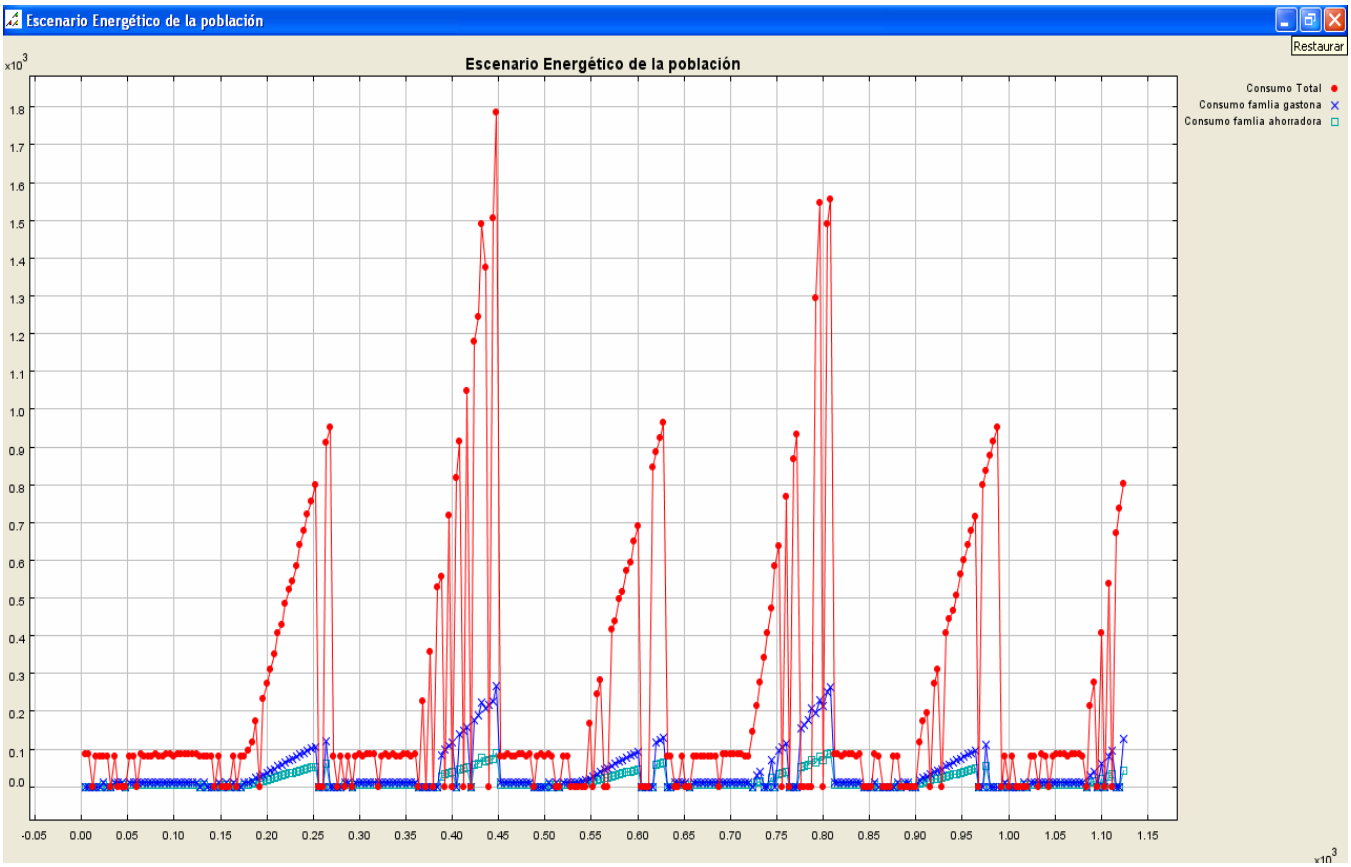


Ilustración 43. Evolución del consumo para el escenario 2.

En la figura anterior se muestra la evolución del consumo a medida que va pasando el tiempo de simulación. Se ha considerado la unidad de tiempo como el día (24 horas) de esta manera cada estación ocupa 90 *ticks* correspondería con 90 días que son los 3 meses que dura aproximadamente cada estación.

6.6.4 Conclusiones y posibles ampliaciones

Las conclusiones que podemos sacar del estudio de este escenario radican en que hay que tomar en cuenta la influencia de las estaciones puesto que la población sigue más o menos un patrón de comportamiento parecido en las estaciones. Esta influencia no depende de otros agentes puesto que si un agente es más caluroso que otro no le pregunta a otro agente si él tiene también calor para ir a comprar un aire acondicionado sino que únicamente puede influir en dónde le recomienda un precio mejor para comprar dicho aparato pero nunca determina firmemente la decisión de comprarlo o no.

6.7 Escenario3: Factores demográficos que afectan al consumo.

Este modelo del consumo eléctrico está obtenido a partir del modelo general que se acaba de explicar en la primera parte de este apartado. Este escenario trata de modelar los efectos producidos en el consumo eléctrico debidos al movimiento de individuos entre distintas poblaciones. Para ello, hemos tenido que añadir distintos aspectos al modelo que puedan motivar dicho movimiento de población.

4.7.1 Explicación del modelo

Lo primero que hemos tenido que ampliar con respecto al modelo general ha sido la distribución de los agentes en distintos espacios, de manera, que los agentes se pudieran mover de un espacio a otro dependiendo de diversos factores que aparecen detallados más adelante. De manera que, cada uno de los agentes está localizado en un barrio distinto. En concreto hay tres barrios: Barrio1, Barrio2 y Barrio3. El total de agentes inicialmente se distribuye equitativamente entre los tres barrios, de manera que todos ellos tengan la misma densidad de población.

Ahora bien, para motivar la “mudanza” de los agentes de un barrio a otro, hemos añadido características a los espacios:

- Actividad cultural
- Centros comerciales
- Equipamiento educativo
- Equipamiento sanitario
- Polígono Industrial
- Transporte
- Zonas verdes

De modo que cada espacio tiene un nivel de desarrollo en cada una de dichas características: AltoDesarrollo, NormalDesarrollo, BajoDesarrollo o SinDesarrollo. El estado inicial de desarrollo de dichas características para cada uno de los barrios es:

Inicial	Barrio1	Barrio2	Barrio3
Actividad cultural	Alto	Bajo	Alto
C. comerciales	Normal	Alto	Alto
Eq. educativo	Normal	Bajo	Alto
Eq. sanitario	Normal	Bajo	Alto
Pol. industrial	Sin	Alto	Sin
Transporte	Alto	Normal	Alto
Zonas verdes	Normal	Bajo	Alto

Tabla 2. Configuración inicial de las características de los espacios.

De esta forma los espacios se convierten en una especie de agentes que evolucionan (se desarrollan) en el tiempo ya que tiene unas características, un estado de dichas características y unas funciones de transición que permiten cambiar el estado de dichas características y que los espacios (en este caso barrios) evolucionen. En principio nuestros agentes están especialmente preocupados por el equipamiento sanitario y educativo del barrio donde viven, de modo que van a ir buscando establecerse en barrios con un AltoDesarrollo de estas dos características. Esta va a ser una de las motivaciones que provoquen la mudanza de los agentes.

Pero hay veces que nos pareciera que se hacen mudanzas sin razón aparente, no hay ningún problema en la vivienda actual, el barrio cumple con las expectativas de calidad de vida, etc. Pero, sin embargo, se produce la mudanza debido a que simplemente hay un afán por conocer nuevos horizontes. Esta característica tan humana viene reflejada en los agentes debido al "carácter nómada" que les hemos añadido. Este carácter nómada evoluciona según un algoritmo genético, de modo que, dicho carácter está formado por dos cromosomas X e Y, cada uno de ellos tiene 4 genes, que a su vez vienen determinados por un alelo de carácter booleano. Para calcular el valor de dicho carácter nómada pasamos a decimal el valor binario de cada cromosoma y calculamos la función de Bohachevsky de modo que el valor obtenido es un double que puede valer desde -1 hasta 1. Así consideramos que aquéllos agentes que lleguen al step con carácter nómada mayor que 0 se van a mudar aún estando en un barrio que tenga un AltoDesarrollo sanitario y educativo.

El carácter nómada de los agentes evoluciona como cualquier otra característica de éstos, es decir, tiene un estado y una función de transición que permite cambiar su estado. En esta ocasión, tratándose de algoritmos genéticos, cabe mencionar como funciona la función de transición del carácter nómada. En cada step se decide si los genes del agente referentes a este carácter van a mutar o no, de manera que si muta el agente se hará el C1 de todos los alelos correpondientes a los genes de cada uno de los cromosomas de dicho agente para este carácter. En principio, también está implementada la posibilidad de que un agente pueda cruzar sus genes con otros agentes de su mismo barrio de manera que así también se modifiquen sus respectivos caracteres nómadas. Esto implicaría cierta influencia de la vecindad en donde viven respecto a los hábitos de mudanza de un agente (interacción agente-agente) pero debido al alto coste en

tiempo que dicho cruce supone, la simulación se ralentiza de modo impracticable, así que hemos decidido no añadir esta dinámica de cruce en los agentes.

Por último hemos añadido a cada espacio una característica que indica si en dicho espacio se ha producido un corte de electricidad o no. Es decir, si el consumo eléctrico total de los agentes que habitan en dicho barrio ha sobrepasado los KW que ese espacio puede proporcionar a sus vecinos. Cuando esto ocurre, el sistema se colapsa y se corta el suministro eléctrico en todo el barrio, de manera que el consumo total se ve afectado también ya que los agentes de dicho barrio dejan de consumir. Los agentes que habitan en barrios que padecen cortes de electricidad entonces se mudan a otros barrios que sí que garantizan, de momento, el suministro. Esto es, independientemente de su carácter nómada o de la calidad sanitaria y educativa de los barrios en los que están o a los que van, sobreponen la necesidad de electricidad a estos otros factores.

4.7.2 UML

A continuación aparecen los cambios más significativos que hemos añadido al modelo general para realizar este escenario.

Con respecto a los espacios hemos añadido características y los estados y funciones de transición para dichas características de modo que los cambios que hay en este sentido en el diagrama general de clases se muestran a continuación:

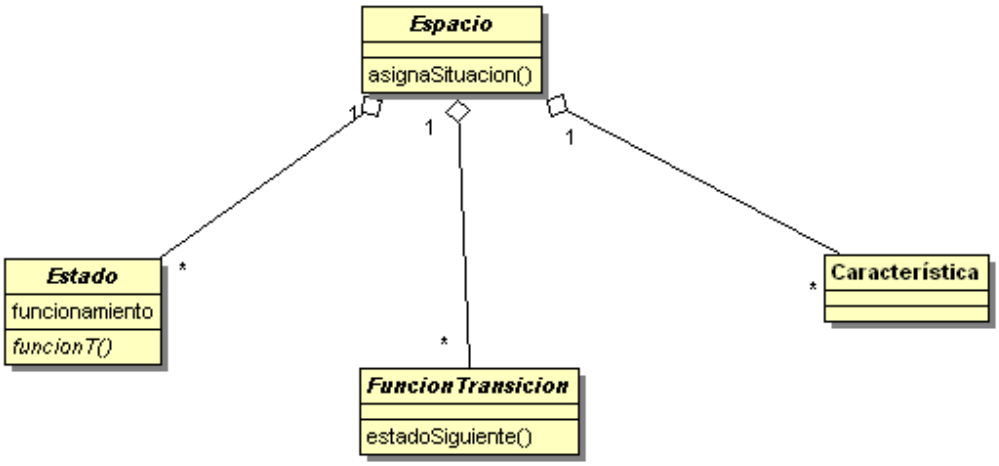


Ilustración 44. Diagrama de clases de las cualidades de los espacios.

Las características añadidas para los espacios son:

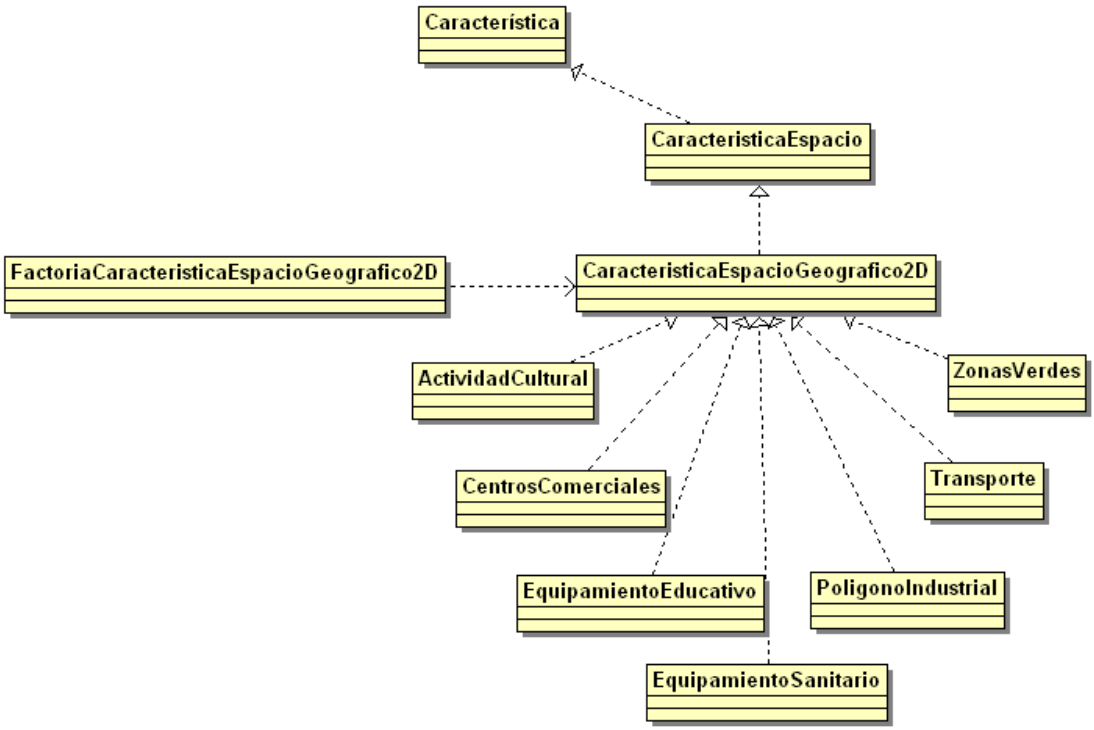


Ilustración 45. Diagrama de clases de las características de los espacios.

Los estados añadidos para los espacios son:

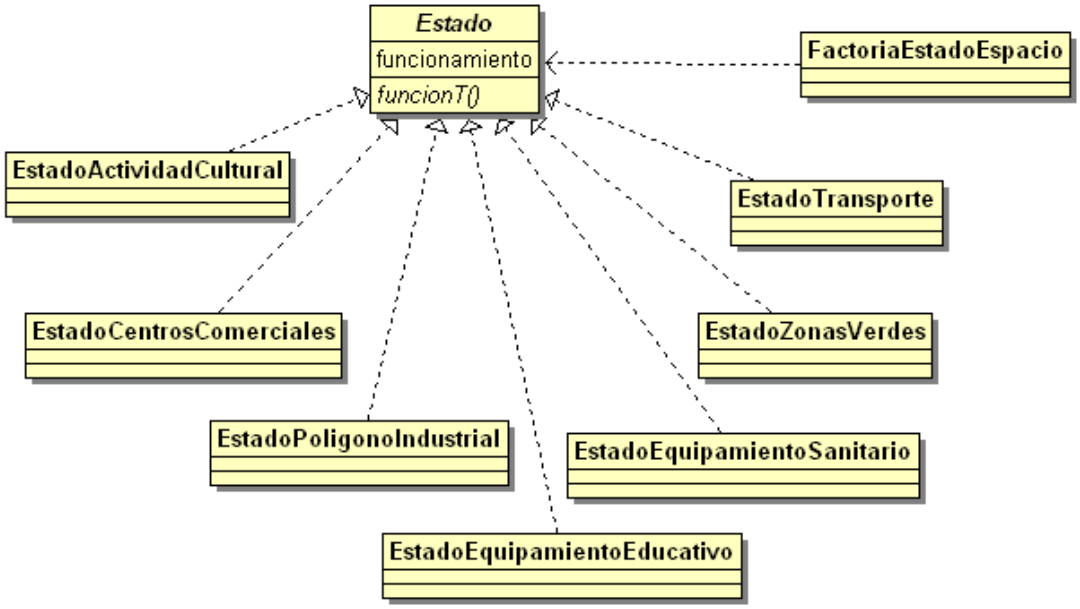


Ilustración 46. Diagrama de clases de los estados de los espacios.

Las funciones de transición añadidas para los espacios son:

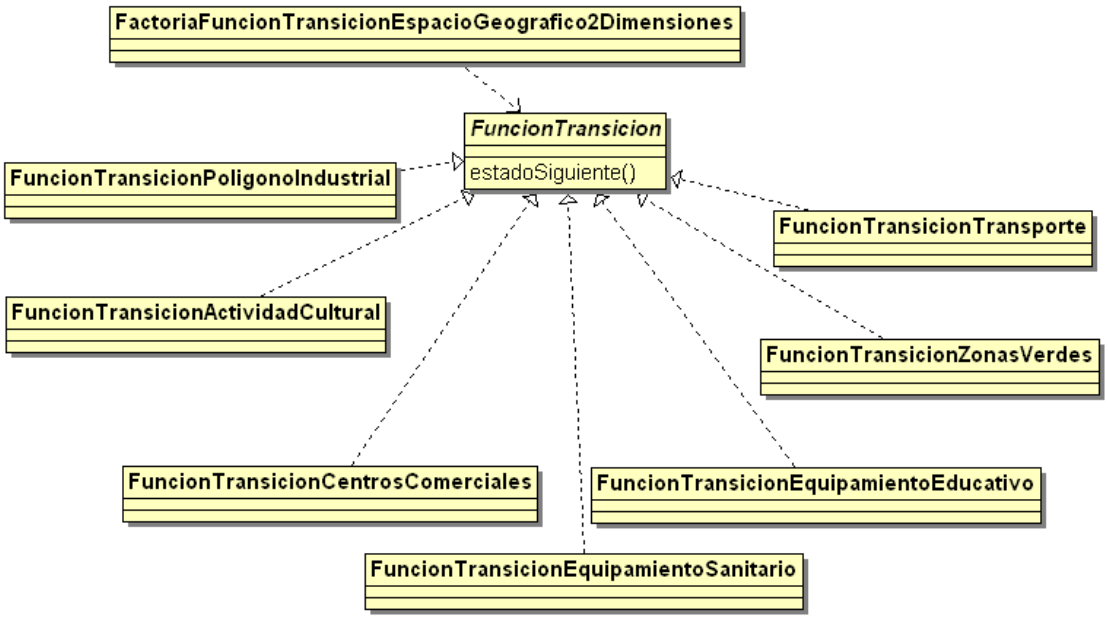


Ilustración 47. Diagrama de clases de las funciones de transición de los espacios.

También hemos añadido a las características de los agentes consumidores el carácter nómada así como su estado y su función de transición correspondiente. Por otra parte, hemos añadido las clases necesarias para la evolución del carácter nómada mediante un algoritmo genético, para estas clases auxiliares se ha creado un paquete nuevo llamado *algoritmosGeneticos*:

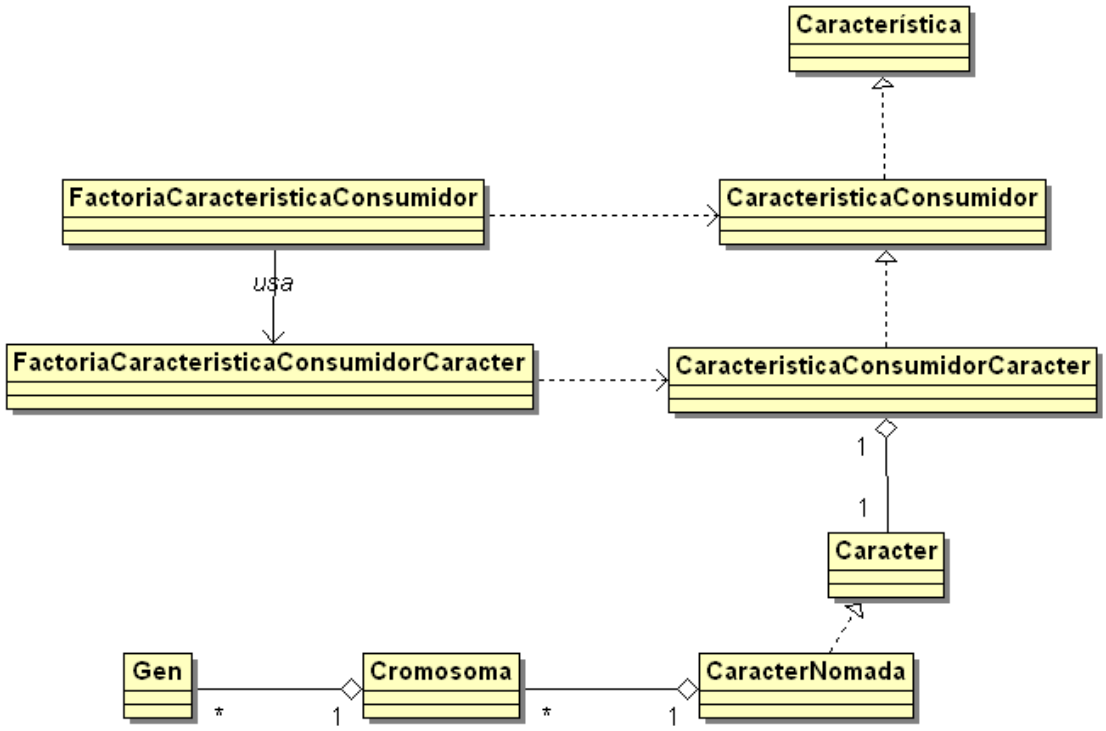


Ilustración 48. Diagrama de clases relacionadas con el carácter nómada añadido a los agentes consumidores.

4.7.3 Análisis de datos

En esta simulación hemos observado el comportamiento de los agentes cuando hacemos que los barrios evolucionen a la siguiente configuración desde la inicial:

Evolución	Barrio1	Barrio2	Barrio3
Actividad cultural	Alto	Bajo	Alto
C. comerciales	Normal	Alto	Alto
Eq. educativo	Alto	Bajo	Alto
Eq. sanitario	Alto	Bajo	Alto
Pol. industrial	Sin	Alto	Alto
Transporte	Alto	Normal	Alto
Zonas verdes	Normal	Bajo	Alto

Tabla 3. Configuración de las características de los espacios tras su desarrollo.

Es decir, vemos que en el Barrio 1 hay una mejora de los sistemas educativo y sanitario que van a tener en cuenta los agentes. La población inicial está formada por el mismo número de agentes consumidores ahorradores que de agentes consumidores gastadores.

Por otra parte, cada barrio tiene las siguientes capacidades de suministro eléctrico:

- Barrio 1: 12KW
- Barrio 2: 20KW
- Barrio 3: 14KW

Con estos datos, los resultados de la simulación se muestran en la siguiente gráfica; en rojo aparece el consumo total, en azul el consumo total que habría si no se produciesen cortes de electricidad, en verde el consumo eléctrico en el barrio1, en negro el consumo eléctrico en el barrio2 y en amarillo el consumo eléctrico en el barrio3:

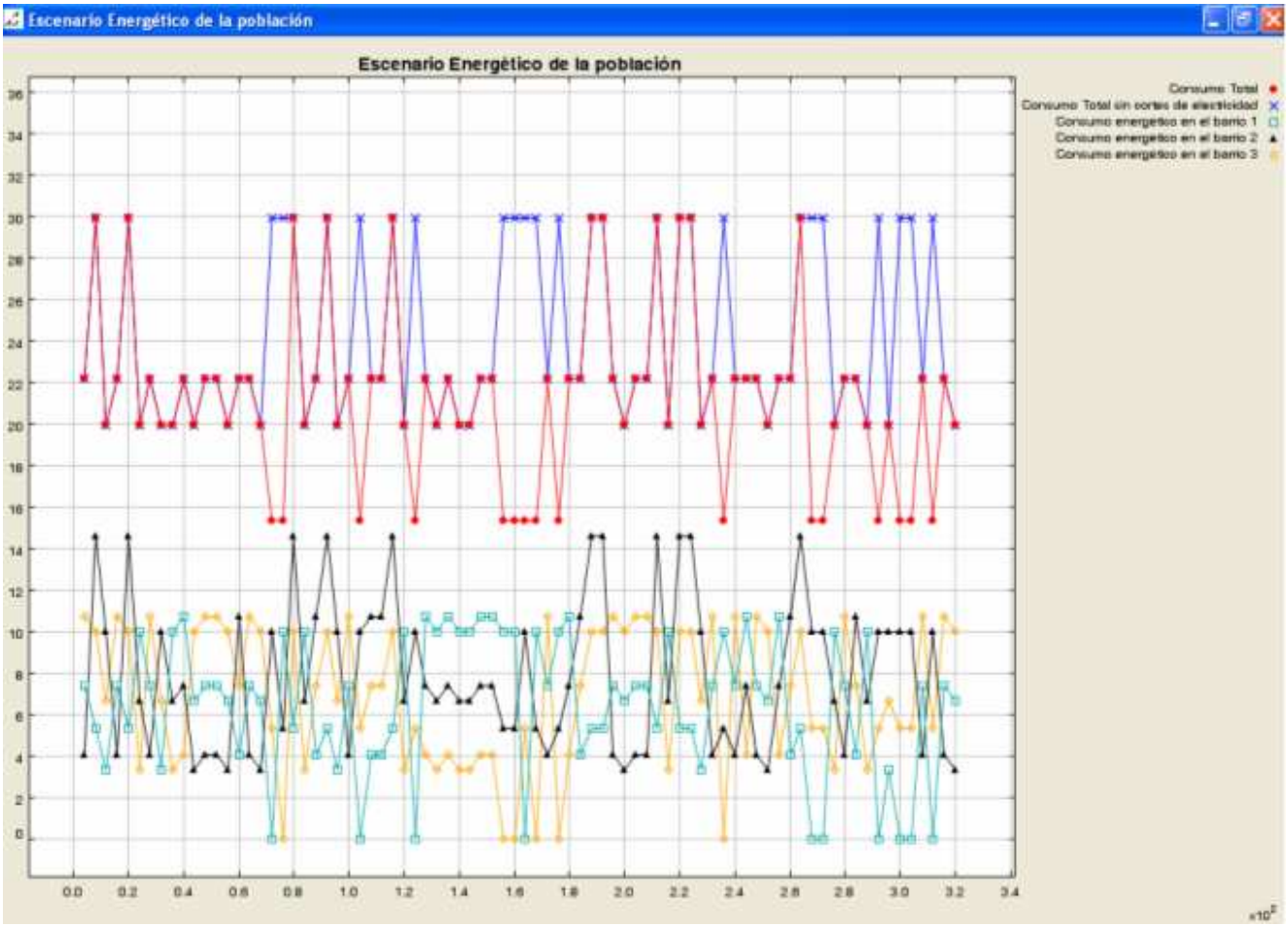


Ilustración 49. Evolución del consumo para el escenario 3.

Como se puede observar, como los agentes tienden a establecerse en barrios con nivel educativo y sanitarios altos, que en esta ocasión tras la 1ª evolución son el barrio1 y el barrio3, es en estos barrios donde se da un mayor número de cortes de suministro debido a la sobrecarga de la red producida por la alta densidad de población.

Al principio, los agentes se distribuyen bastante uniformemente pero enseguida empiezan a mudarse en búsqueda de un barrio mejor, lo que provoca que baje el consumo del barrio 2 que es el que peor condiciones sanitarias y de educación tiene, aunque su consumo nunca llega a 0, ya que como hemos explicado en el modelo la búsqueda de un buen barrio no es la única motivación de nuestros agentes, también influyen en ellos su carácter nómada. En esta situación vemos que debido a la superpoblación de los barrios 1 y 3 se producen cortes de electricidad, lo que hace que los agentes se muden y que normalmente sea el barrio 2 el que acoja estas mudanzas. Dichas mudanzas, hacen que la población y el consumo de los barrios 1 y 3 bajen y que ya no se produzcan cortes, lo que anima a algunos agentes del barrio 2 a volverse a mudar a estos barrios de alto nivel sanitario y educativo.

Como también se puede observar el barrio1 es mucho más propenso a los cortes de luz que el barrio3 ya que tiene una capacidad de suministro menor que éste.

4.7.4 Conclusiones y posibles ampliaciones

Tras el estudio y modelado de este problema podemos recoger las siguientes reflexiones:

- El haber conseguido anteriormente un modelo general que es fácilmente extensible, nos ha facilitado la implementación de este escenario.
- Una vez más queremos mencionar la necesidad de un buen nivel de conocimiento del dominio del problema, para así poder modelar los agentes con las características más relevantes, de tal forma que durante la simulación podamos observar como emerge un comportamiento en toda la comunidad de agentes.

Y finalmente, con respecto a este escenario serían interesantes las siguientes ampliaciones:

- Hacer el tratamiento del cruce entre los caracteres nómadas de los agentes más eficiente. Este aspecto conllevaría la interacción entre agentes directamente, que es muy interesante para el estudio.
- Implementar más caracteres que afecten la movilidad del agente y, del mismo modo, tener en cuenta más factores del espacio para decidir la mudanza de un agente o el establecimiento de éste en un cierto barrio.

El conseguir más espacios con características más dispares entre sí enriquecería este modelo de consumo energético debido a factores demográficos.

7. Conclusiones y posibles ampliaciones

A continuación enumeramos las conclusiones obtenidas a lo largo del trabajo desarrollado en nuestro sistema multiagente.

Con respecto a los sistemas multiagente en general:

- Los sistemas multiagente funcionan especialmente bien para modelar características macro (tendencias) de una población partiendo de las características micro de cada agente.

- Para poder modelar cada agente es necesario poseer cierto conocimiento del problema a tratar. De modo que, modelemos cada agente con las características que son más determinantes en el estudio del problema a tratar.

- Una de las características más importantes de un SMA es que el grupo de agentes que lo integran debe trabajar de manera cooperativa e individual. Los agentes trabajan de manera cooperativa para satisfacer las metas globales que se derivan de la búsqueda de las soluciones a los problemas globales y de manera individual, porque las metas globales son descompuestas en submetas, generando metas locales para los agentes que participarán en el desarrollo de las soluciones a los problemas.

- Cuando dos o más agentes trabajan en grupo, se presentan conflictos de intereses de manera natural. Es necesario que el ambiente en el que se desenvuelven provea procesos para la resolución de tales conflictos. Para ello se suelen establecer procesos de toma de decisiones. Algunas de estas tareas de decisión están caracterizadas por las relaciones cooperativas entre los miembros del grupo, en las cuales los individuos trabajan por el desempeño del grupo en general; o por relaciones competitivas en las cuales los miembros del grupo plantean posiciones definidas y discuten entre sí, defendiendo sus propios intereses.

Con respecto a la simulación basada en agentes para el estudio del consumo eléctrico:

- Los modelos de estudio del consumo tienen una alta complejidad debido a que en el mercado energético están involucrados agentes consumidores (particulares, empresas) con un comportamiento de consumo complejo.

- El cálculo de una buena predicción de consumo se hace complicada ya que una buena predicción implica un ajuste bastante exigente.

- Es importante mostrar de manera adecuada, con diferentes gráficas o mapas, los resultados obtenidos en nuestros modelos para así poder tener una buena comprensión de lo que ha sucedido en la simulación.

Sería interesante aplicar las siguientes ampliaciones al modelo para hacer un estudio del consumo eléctrico más realista:

- Añadir más tipos de agentes consumidores, sobretodo de grandes consumidores como fábricas que no aparecen reflejados en nuestro modelo.

- Permitir que dos agentes que tengan un cierto aparato puedan tener distintos consumos debidos a él, ya que, por ejemplo aunque dos agentes tengan televisor no generarán el mismo consumo debido a él, éste dependerá del uso que hagan del televisor.

- Considerar un concepto todavía más general de Agente que no tenga que tener situación, de modo que, podamos considerar los Espacios como Agentes que van evolucionando sus infraestructuras con el tiempo. Esto mejoraría la interacción espacio-agente.

- Implementar la interacción agente-agente. De manera que, el patrón de consumo eléctrico de un agente conocido pueda influir en el consumo de otro.

- Aumentar la interacción agente-entorno, de modo que la distribución energética pueda cesar debido a acciones de los agentes en el entorno por sobrecarga de la red o por averías en el sistema.

- Dotar al sistema multiagente de más objetivos comunes de manera que generara la necesidad más interacción entre ellos. Por ejemplo, unirse en grupos de queja para la bajada de tarifas.

- Mostrar mediante un mapa la situación de los agentes, ayudaría a sacar conclusiones de la simulación con más facilidad.

8. Glosario de términos

ABM Agent-based Model
AI Artificial Intelligence
GIS Geographic Information System
KBS Knowledge-based System
MAS Multiagent System
SAX Simple API for XML
UML Unified Modeling Language
XML Extensible Markup Language

9. Bibliografía

Agentes software y sistemas multiagente: conceptos, arquitecturas y aplicaciones, Ana Mas, Pearson Educación S.A., Madrid 2005

Tutorial de Repast
http://u.arizona.edu/~jtmurphy/H2R/HowTo01.htm

Red Eléctrica Española
www.ree.es

Evaluación Integradora de Políticas de Agua: modelado y simulación con sociedades artificiales de agentes. Tesis doctoral. José Manuel Galán Ordax.
Burgos, 2007

Artículo: Agent-based Simulation Platforms: Review and Development Recommendations.
Autores: Steven F. Railsback, Steven L. Lytinen y Stephen K. Jackson .

Artículo: Integración de mecanismos de razonamiento en agentes de software inteligentes para la negociación de energía eléctrica ***Autores****:* F.J.Arias, J.Moreno,D.A.Ovalle ***ISSN*** 0717-4195, Nº. 13, 2006
http://dialnet.unirioja.es/servlet/articulo?codigo=2208264

Artículo: Data Mining y visualización de datos en el mercado eléctrico español. Autores: Eugenio Francisco Sánchez Úbeda, Antonio Muñoz y José Villar.
Publicado en la Revista Íberoamericana de Inteligencia Artificial, 2006.

Artículo: A multi-agent approach for a self-reconfigurable electric power distribution system.
Autores: Janeth G. Gómez–Gualdrón y Miguel Vélez-Reyes. Universidad de Puerto Rico en Mayagüez (EEUU) , 2006
http://adsabs.harvard.edu/abs/2006SPIE.6229E..13G

Artículo: MASCEM un sistema multiagente que simula la competencia de mercados eléctricos.
Autores: Isabel Praca, Carlos Ramos y Zita Vale. Instituto Politécnico de Oporto, 2003.
http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/ex/&toc=comp/mags/ex/2003/06/x6toc.xml&DOI=10.1109/MIS.2003.1249170

Artículo: Soporte Multiagente en tiempo real para la administración de energía eléctrica.,
Autores: H.F. Wedde, S. Lehnhoff, E. Handschin y O. Krause. Dortmund (Alemania), 2006.Publicado por la asociación IEEE (Washington, EEUU).
http://portal.acm.org/citation.cfm?id=1153935

Artículo: "Evaluation of free Java-libraries for social-scientific agent based simulation" Robert Tobias y Carole Hotmann

Journal of Artificial Societies and Social Simulation vol. 7, no. 1, 2004
http://jasss.soc.surrey.ac.uk/7/1/6.html

Artículo: Un sistema basado en agentes para la protección de componentes eléctricos. Autores L. Hipólito y P. Siano. Italia, 2004.
http://www.actapress.com/PaperInfo.aspx?PaperID=17984