La Simulación en Ingeniería del Software

Tema 24. La simulación en Ingeniería del Software

1. Introducción

La simulación es la imitación de la operación de un proceso en el mundo real que evoluciona a lo largo delo tiempo. Simular algo requiere en primer lugar la elaboración de un modelo que represente las características y comportamientos del sistema o proceso modelado. El modelo representa una cierta vista de un sistema y la simulación el funcionamiento de es esa vista del sistema en el tiempo.

La simulación se utiliza en muchos contextos. Aquí estamos interesados en la simulación por ordenador. Una simulación por ordenador es una simulación que se ejecuta en un solo

ordenador o una red de ordenadores, para reproducir el comportamiento de un sistema. La simulación utiliza un modelo del sistema.

Simular requiere, en primer lugar, construir un modelo del sistema cuyo comportamiento queremos reproducir. Los modelos pueden tener diferentes finalidades. Aquí estamos interesados en modelos dinámicos que representen un conjunto de hipótesis y de leyes adecuadas para explicar un determinado aspecto de la realidad. La primera cuestión importante es que un modelo se construye para explicar un determinado aspecto. Un modelo no pretende explicar toda la realidad sólo algunas características de la misma. Es importante, para construir un buen modelo, indicar los objetivos buscados, las escalas de tiempo relevantes y los resultados esperados.

La simulación se ha convertido en una metodología para explorar la comprensión de la realidad. Construimos un modelo con hipótesis sobre la realidad, contrastamos los resultados y validamos o no las hipótesis. Por otra parte, con la simulación, podemos obtener lo que ocurría en un sistema si actuamos sobre él y es muy difícil, o muy caro, llevar a cabo una experimentación real.

Un modelo dinámico consta de la descripción de la situación del sistema en un instante dado del tiempo, que llamaremos el **estado del sistema**, y un conjunto de reglas que definen como ese estado cambia con el tiempo. Simular es el proceso de definir un estado inicial del sistema y ejecutar las reglas definidas para obtener el estado del sistema en el futuro.

Un modelo puede construirse para considerar una realidad a diferentes niveles de abstracción. A un nivel de abstracción bajo tenemos que tener en cuenta objetos individuales y su comportamiento para representar entidades del mundo real. Aparecen en el modelo elementos individuales tales como personas, productos, vehículos, animales, casas, etc. A un nivel de abstracción alto tenemos en cuenta valores agregados y tendencias. Los elementos individuales nunca se consideran. Entre los dos extremos hay muchas posibilidades de abstracción. Igualmente relevante es la escala de tiempo. Un modelo puede construirse para representar una realidad a escalas de días, de meses, de años o de siglos. El modelo resultante incluirá hipótesis y dados completamente diferentes.

Los modelos pueden ser continuos o discretos según que la variable que representa el tiempo tome valores continuos o en instantes discretos. Estocásticos o deterministas según que los cambios del estado vengan dadas por leyes deterministas o por eventos aleatorios. Un tipo especial de modelos discretos son los basados en agentes. Aquí las entidades están representadas directamente y poseen un estado interno y un conjunto de comportamientos o normas que determinan cómo será el estado siguiente del agente. Los agentes son objetos activos. En la simulación de eventos discretos las entidades del mundo se representan mediante objetos pasivos. Es decir con estado internos pero sin reglas propias para determinar la evolución del mismo.

Según el tipo de modelos podemos considerar tres grandes paradigmas de simulación.

- Dinámica de Sistemas (SD)
- Simulación de Eventos Discretos (DE)

- Simulación Basada en Agentes (AB).

SD y DE son tradicionales, AB es más nuevo. SD es adecuada para modelos continuos a un nivel de abstracción alto, mientras que DE y AB son en tiempo discreto y nivel de abstracción medio o bajo. Los modelos basados en agentes se están utilizando en todos los niveles de abstracción. Los agentes pueden modelar objetos de muy diversa naturaleza y escala. Los Sistemas Dinámicos (SD) en general podemos considerarlos con un caso particular de la Dinámica de Sistemas orientado al modelar sistemas físicos.

Actualmente hay una evolución creciente hacia la integración y la cooperación eficaz entre los diferentes los tres anteriores paradigmas de modelado.

2. Dinámica de Sistemas

La dinámica de sistemas (SD) es una metodología de modelado adecuada para la comprensión y discusión de temas y problemas complejos. Originalmente desarrollada en la década de 1950 para ayudar a los directores de empresas a mejorar su comprensión de los procesos industriales, la dinámica del sistema está siendo utilizado actualmente en todo el sector público y privado para el análisis y diseño de políticas.

La dinámica de sistemas es un aspecto de la teoría de sistemas como un método para entender el comportamiento dinámico de sistemas complejos. La base del método es el reconocimiento de que la estructura un sistema - entendiendo por estructura del sistema un conjunto de elementos y un conjuntos de relaciones de influencia entre las partes, a veces, con retardos de tiempo incluidas en las mismas - es a menudo tanto o más importante, para determinar el comportamiento del sistema, que los propios componentes individuales. La Teoría de Sistemas en, general, también afirma que a menudo hay propiedades que están asociadas al sistema como un todo pero no a cada una de las partes. O dicho de otra forma, en algunos casos, el comportamiento del todo no puede ser explicado en términos de la conducta de las partes.

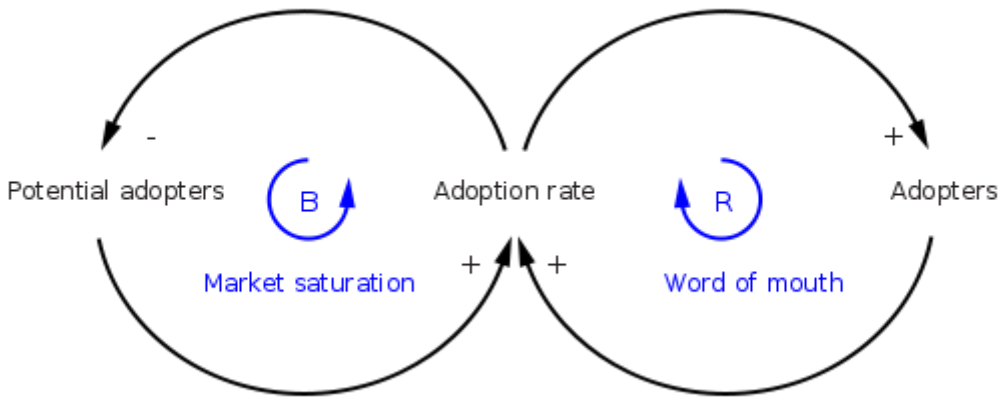
La dinámica de sistemas fue creada a mediados de los años 1950 por el profesor *Jay Forrester*, del Instituto de Tecnología de Massachusetts. En esos momentos, los directivos de *General Electric* estaban perplejos porque el empleo en sus plantas de electrodomésticos en Kentucky exhibía un ciclo de tres años. El ciclo económico se estimó que era una explicación insuficiente para la inestabilidad laboral. A partir de simulaciones *Forrester* fue capaz de mostrar cómo la inestabilidad en el empleo se debía a la estructura interna de la empresa y no a una fuerza externa, como el ciclo de negocios.

Elementos de Dinámica de Sistemas: Bucles de Realimentación

En Dinámica de Sistemas, un sistema (por ejemplo, un ecosistema, un sistema político o un sistema mecánico) se representa inicialmente como un diagrama causal. Un diagrama causal

es una representación gráfica de los constituyentes de un sistema y las interacciones entre los mismos. Al capturar las interacciones y por consiguiente, los circuitos de realimentación, un diagrama causal revela la estructura de un sistema. Mediante la comprensión de la estructura, se hace posible determinar el comportamiento de un sistema a lo largo de un cierto período de tiempo.

Como un ejemplo veamos un diagrama causal para modelar el mecanismo de difusión de un nuevo producto o idea. Partimos de dos ideas que modelan el estado del sistema: el número de individuos que han adoptado el producto y el potencial de individuos que los podrían adoptar. Ambas consideradas como variables continuas y no discretas. Junto a las dos variables que definen el estado del sistema tenemos la tasa de adopción que explica el ritmo de cambio del estado. El conjunto de las tres variables tiene unas influencias entre sí tal como se muestra en el diagrama causal siguiente. Estas influencias pueden ser positivas o negativas. De una forma muy general podemos decir que una variable es influida por otra de forma positiva cuando depende de ella a través de una función que es monótona creciente y si la influencia es negativa la relación funcional es monótona decreciente.



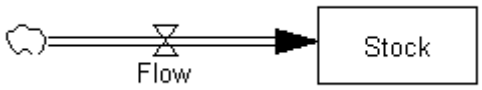
Las relaciones de influencia forman bucles. Unos son bucles de realimentación positiva, si el productos de los signos de las influencias es positivo, y otros bucles de realimentación negativa, si el producto de los signos de las influencias es negativo. La cuestión clave, conocida de la Teoría de Sistemas Dinámicos, es que los bucles de realimentación positiva modelan mecanismos de crecimiento exponencial. Los bucles de realimentación negativa modelan mecanismos de crecimiento de estabilización similares a los que mantienen constante la temperatura del cuerpo humano.

En el ejemplo anterior hay dos bucles de realimentación: uno positivo y otro negativo que compiten entre sí para hacer evolucionar e sistema.

Niveles y Flujos

Los diagramas causales ayudan a la visualización de la estructura de un sistema, y la comprensión de las causas del comportamiento del mismo. Pero el análisis que podemos hacer es solo cualitativo. Para realizar un análisis cuantitativo más detallado, hay que refinar el

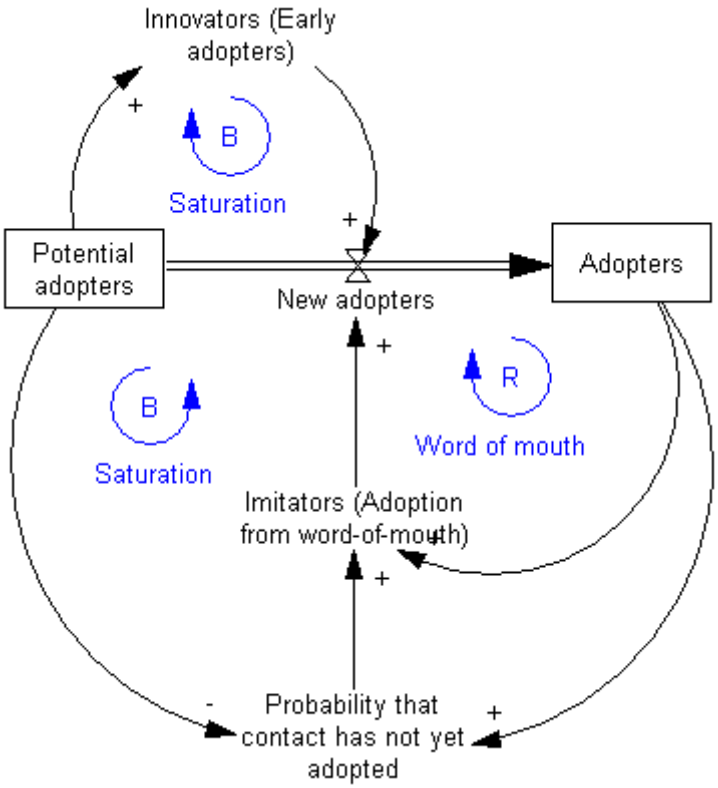
diagrama causal se transforma en otros de niveles y flujos. Los niveles son las variables del modelo causal que representan el estado del sistema en un tiempo dado. Los flujos modelan las tasas de cambio de cada almacén. Gráficamente los niveles (stocks) y sus flujos asociados se representan como rectángulos y flujos que los alimentan.



Junto a los niveles y los flujos aparecen otras variables que llamaremos auxiliares. Estas son variables que dependen funcionalmente de los niveles o de otras variables auxiliares pero no de los flujos. Por ejemplo innovadores, individuos que adoptan el nuevo producto de forma temprana debido a las cualidades del mismo, imitadores (individuos que adoptan el producto por imitación de otros que ya los tienen) o probabilidad de contacto entre un individuo que tiene el producto y otro que no lo tiene.

Además existen variables exógenas. Son variables que influyen sobre las demás pero no son influidas. Si estas variables son constantes se denominan parámetros. Ejemplos de parámetros son la tasa de innovadores o la tasa de imitadores surgidos de contactos entre individuos con y sin el producto.

Con estos elementos tenemos el diagrama de niveles y flujos.



Este diagrama puede convertirse en un sistema de ecuaciones diferenciales que integrado nos proporcionará la evolución de sistema modelado.

Para hacerlos de forma más compacta escojamos un símbolo para representar cada uno de los niveles, flujos, variables intermedias y parámetros en la forma:

- *Clientes Potenciales*: x
- *Individuos que ya tienen el producto*: y
- *Nuevos individuos con el producto*: f
- *Probabilidad de contacto*: c
- *Imitadores*: f1
- *Innovadores*: f2
- *Tasa de imitación*: q
- *Tasa de innovación*: p

Con esas variables el sistema de ecuaciones diferenciales resultante es:

$$c = \frac{x}{x+y}$$
$$f1 = q\ y\ c$$
$$f2 = p\ x$$
$$f = f1 + f2$$
$$\frac{dx}{dt} = -f$$
$$\frac{dy}{dt} = f$$

Que eliminando las variables intermedias y los flujos resulta en

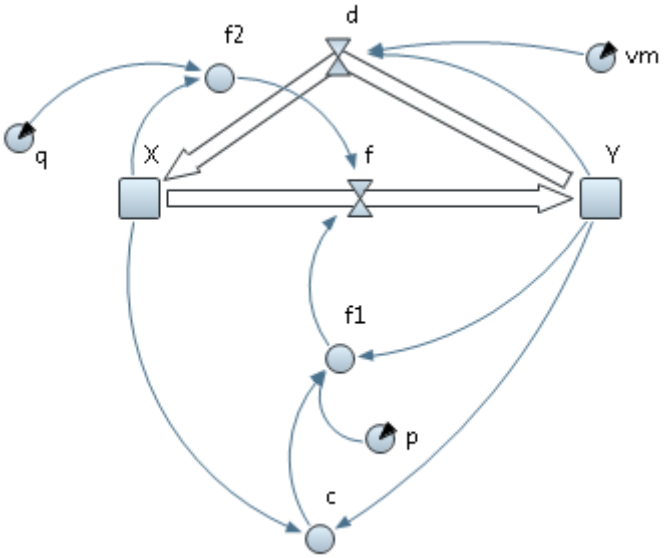
$$\frac{dx}{dt} = -x(\frac{q\ y}{x+y} - p)$$
$$\frac{dy}{dt} = x(\frac{q\ y}{x+y} - p)$$

Para poder integrar ese sistema de ecuaciones diferenciales hacen falta las condiciones iniciales (el valor inicial de los niveles) y el valor de los parámetros.

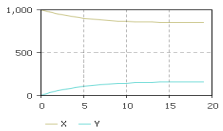
El modelo puede mejorarse considerando que el producto adoptado por los clientes tiene una vida media tras la cual caduca y el cliente pasa ser un cliente potencial de otro nuevo producto. Si la vida media del producto la representamos por el parámetro m es necesario incluir un nuevo flujo desde el almacén y hasta el x. Este flujo es igual a y/m y las ecuaciones quedarían:

$$\frac{dx}{dt} = -x\left(\frac{q\ y}{x+y} - p\right) + \frac{y}{m}$$
$$\frac{dy}{dt} = x\left(\frac{q\ y}{x+y} - p\right) - \frac{y}{m}$$

El modelo, usando la herramienta de simulación *AnyLogic*, queda de la forma:



La evolución del sistema es de la forma:



Metodología para construir un modelo

La primera etapa en la construcción de un modelo se suele denominar conceptualización. Esta etapa se compone de una serie de pasos. El primer paso, la decisión sobre el propósito modelo, significa indicar el objetivo del modelo. El segundo paso es definir los límites del modelo y las variables clave. Esto implica seleccionar los elementos necesarios para generar el comportamiento de interés según por el objetivo del modelo. Después de definir el límite del modelo y la identificación de variables clave, algunas de se representan gráficamente en el tiempo. A esto lo llamaremos un modo de referencia. Los modos de referencia más importantes son las hipótesis y los modos históricos de referencia. El paso final en la conceptualización es decidir sobre los mecanismos básicos, los bucles de retroalimentación del sistema.

Un modelo de dinámica del sistema se construye para entender el sistema de fuerzas que han creado un problema en un sistema. Para tener un modelo significativo, debe haber algún

problema subyacente en un sistema que crea la necesidad de comprenderlo para poder transformarlo. Para comprender el sistema hay que reunir los datos relevantes sobre él. Datos relevantes son no sólo los datos estadísticos, también el conocimiento de expertos en el dominio. Es importante considerar también el público interesado en el modelo. Un modelo es muy diferente si se construye para alumnos de primaria, que si se construye por la Agencia de Protección Ambiental de los Estados Unidos con el objetivo de tomar decisiones políticas. Si la estructura y el comportamiento del modelo no pueden ser entendidas por el público objetivo, o si no responde a cuestiones interesantes para ese público, entonces el modelo se vuelve inútil.

El propósito de un modelo suele caer en una de las siguientes categorías:

- Aclarar el conocimiento y comprensión del sistema
- Descubrir las políticas que mejoren el comportamiento del sistema
- Capturar modelos mentales y servir como medio de comunicación y unificador.

Al diseñar un modelo de dinámica de sistemas hay que definir claramente los límites del modelo e indicar los elementos endógenos y los exógenos. Los primeros participan en los bucles de realimentación del sistema y los segundos influyen sobre el sistema pero no son influidos por éste.

Los Modos de Referencia son representaciones gráficas de la evolución de las variables clave de un sistema a través del tiempo. El modo de referencia capta modelos mentales y datos históricos sobre el papel, da pistas de la estructura del modelo apropiado, y sirven para comprobar la verosimilitud del modelo una vez construido. Se explicitan modos de referencia para mostrar la existencia de algún fenómeno o comportamiento digno de ser modelado.

El paso final en la conceptualización es decidir sobre los mecanismos básicos del sistema. Específicamente, los mecanismos básicos son los bucles de realimentación en el modelo. Los mecanismos básicos representan el conjunto más pequeño de relaciones capaces de generar el modo de referencia. Los mecanismos básicos también pueden ser considerados como la historia más simple que explica el comportamiento dinámico del sistema.

Una manera de buscar los mecanismos básicos es escoger la variables que van a representar el estado del sistema, los niveles, y posteriormente añadir las causas que los hacen evolucionar, los flujos.

Algunas herramientas para modelar: retardos, promedios, predicciones

Existen algunas herramientas que ha demostrado su utilidad a la hora de modelar determinados tipos de sistemas. Son los retardos, los promedios y los pronósticos.

La función de retardo se necesita con frecuencia en la dinámica de sistemas para el modelado efectos retardados, es decir, cuando se necesita algún tiempo para la toma de decisiones, o que se produzcan algunos procesos antes de que se adopte la medida.

La función tiene la forma

delay(flow, delayTime, initialValue)

Los retardos pueden ser de información, cuando se quiere modelar un retardo en la percepción de algo, o de flujo, si se quiere modelar un retardo en un flujo de materia. Por otra parte los retardos pueden ser de diversos órdenes. A mayor orden del retardo éste se aproxima a un retardo puro. A menor orden el retardo se aproxima a un promedio de los valores del pasado dando más importancia a los más cercanos.

La función de pronóstico devuelve una previsión del valor que una variable tendrá en un tiempo del futuro. Tiene la forma:

forecast(input, averageTime, horizon)

La función devuelve el valor que tomará *input* en el tempo at *averageTime + horizon*.

3. Simulación de Eventos Discretos

La simulación de eventos discretos (DES), ve un sistema como una secuencia discreta de eventos en el tiempo. Cada evento tiene lugar en un momento determinado y puede producir un cambio de estado en el sistema. Entre dos eventos consecutivos, se supone que ningún cambio se produce en el estado del sistema. Por tanto, la simulación puede saltar directamente en el tiempo de un evento al siguiente. Una alternativa a la visión basada en eventos es la basada en procesos. En este enfoque, cada actividad en un sistema corresponde a un proceso separado y se simula típicamente por un hilo en el programa de simulación. En este enfoque, los eventos discretos, que son generados por hilos, causarían que otros hilos actualizaran su estado, se despertaran, o pasaran a un estado inactivo. Ambas alternativas comparte la idea de la existencia en un conjunto de eventos discretos que al ocurrir hacen cambiar el estado del sistema.

Esto contraste con la simulación continua en la que el estado del sistema va evolucionando de forma continua en el tiempo debido a las acciones que se están llevando a cabo.

Un ejemplo de este tipo de modelos es la cola formada por los clientes que llegan a un banco para ser atendido por un cajero. En este ejemplo, las entidades del sistema son cliente-cola y Escrutadores. Los eventos del sistema son la llegada de un cliente, cliente empieza a ser atendido y salida de un cliente. Los estados del sistema, que se cambian por estos eventos, son de número de clientes en la cola y estado (ocupado o libre) del cajero o de los cajeros del banco. Los tiempos entre llegados de dos clientes y el tiempo de servicio de un cliente por un cajero son variables aleatorias que deben ser caracterizadas.

La simulación con eventos discretos se adecua muy bien al modelado centrado en procesos. Un proceso modelar objetos del mundo real (operaciones, clientes, productos, componentes, vehículos, etc.), procesos (secuencias de operaciones que implican típicamente colas, retrasos,

utilización de recursos), y recursos. Los procesos suelen especificarse en forma de diagramas de flujo y para construir estos diagramas es necesario disponer de una biblioteca de componentes.

Elementos de un modelo de eventos discretos

- **Sistema**: Una colección de entidades (personas, máquinas, etc.) que interactúan entre sí para llevar a cabo uno o más objetivos.
- **Modelo**: Una representación abstracta de un sistema, que normalmente contiene estructural, lógica o matemática relaciones que describen un sistema en términos de Estado, entidades y sus atributos, conjuntos, procesos, eventos, actividades y retrasos.
- **Estado del Sistema**: Una colección de las variables que contienen toda la información necesaria para describir el sistema en un momento dado. El estado del sistema, como en el caso de la simulación continua, es la situación de un sistema en un momento dado del tiempo. La evolución en el tiempo del estado del sistema, S (t), se puede representar matemáticamente por una función cuyos valores cambian en los momentos en que ocurren eventos.
- **Entidad**: Cualquier objeto o componente en el sistema que requiere la representación explícita en el modelo (un servidor, un cliente, una máquina, etc.).
- **Atributos**: Propiedades de una entidad determinada (prioridad de un cliente, etc.).
- **Colas**: Una colección de entidades ordenadas de alguna manera. Por ejemplo entidades que esperan usar un recurso cuando esté disponible.
- **Evento**: Hecho instantáneo que puede cambia el estado de un sistema (como la llegada de un nuevo cliente). Pueden ser endógenos, si se producen internamente debido a que el estado del sistema alcanza una condición, o exógenos si la causa es externa.
- **Lista de Eventos**: La simulación mantiene al menos una lista de eventos. Un evento se describe por el momento en que se produce y una información que describe su tipo. Junto con los eventos se tienen en cuenta normalmente actividades.
- **Actividad**: Puede ser considerada como un intervalo de tiempo, delimitado por dos eventos, dónde se lleva a cabo algún tipo de acción.
- **Recurso**: Son un tipo especial de entidades que son usadas por otras entidades para llevar a cabo una acción. Una entidad puede solicitar un recurso, espera mientras se le concede, lo usa y posteriormente lo libera.
- **Reloj:** La simulación debe llevar la cuenta del tiempo de simulación. A diferencia de la simulación continua, en la simulación discreta el tiempo da saltos y en cada ocurrencia de un evento el reloj salta al próximo evento disponible.
- **Generadores de números aleatorios:** La simulación necesita variables aleatorias de varios tipos. Los valores de esta variable se logran mediante uno o más generadores de números aleatorios.
- **Estadísticas:** La simulación realiza normalmente un seguimiento de las estadísticas del sistema, que cuantifican los aspectos de interés. En el ejemplo de banco anterior, pueden ser de interés el tiempo medio de espera en la cola en función del número de cajeros disponibles.

- **Condición de parada:** Teóricamente una simulación de eventos discretos podría funcionar para siempre. Es el diseñador de la simulación el que debe decidir cuando la simulación terminará. Las opciones típicas son: en el tiempo t o después de un número n de eventos o, más en general, cuando la medida X alcance el valor de x.

Librería de componentes para modelar procesos en la herramienta AnyLogic

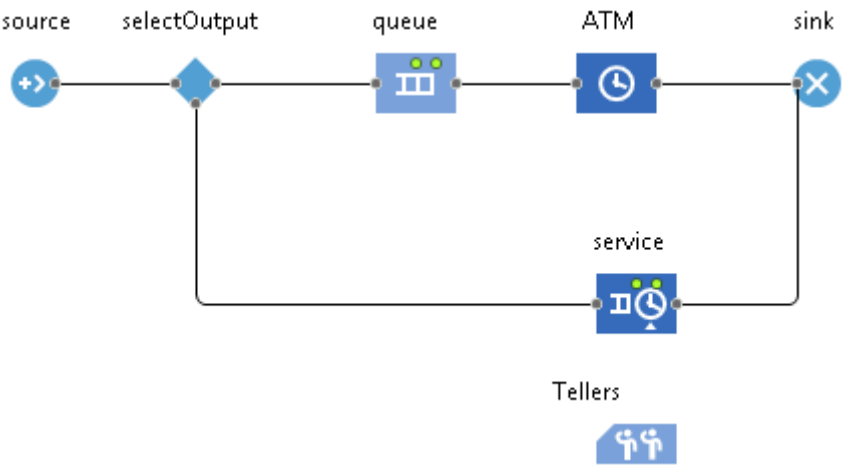
Algunos de los componentes son los siguientes:

1. *Fuente (Source):* Genera objetos (activos o pasivos) según determinados parámetros. Es el punto adecuado para el comienzo de un proceso.
2. *Sumidero (Sink):* Elimina los objetos que le van llegando. Es un punto adecuado para modelar el fin de un proceso.
3. *Retardo (Delay):* Retarda los objetos durante una cantidad de tiempo.
4. *Cola (Queue):* Una cola donde los objetos esperan a su siguiente objetivo en el flujo del proceso, o un dispositivo de almacenamiento de propósito general para los objetos. Es posible asociar un tiempo máximo de espera con un objeto.
5. *Selecciona (SelectOutput):* Encamina a los objetos que le llegan a uno de los dos caminos de salida dependiendo de una condición (probabilista o determinista). La condición puede depender del objeto o de factores externos.
6. *Recursos (ResourcePool):* Define un conjunto de unidades de recursos que pueden ser apropiadas y liberadas por los objetos que fluyen por el proceso.
7. *Servicio (Service):* Es un mecanismo para gestionar el uso de recursos por parte de los objetos del proceso. De manera general reserva un número determinado de unidades de recursos para un objeto, lo retrasa, y libera posteriormente las unidades reservadas. Es útil si el objeto no tiene que hacer nada más que ejecutar un retraso entre tomar los recursos y liberarlos.

Ejemplo

Usando los elementos anteriores podemos modelar el proceso que ocurre en un banco donde hay cajeros automáticos (ATM) y gestores para atender a los clientes que van llegando. Un modelo simple puede ser el siguiente. Usamos una fuente para ir generando los clientes que van llegando al banco, una selección para modelar la decisión para modelar la elección por parte de un cliente de ir al cajero automático o a un gestor. El modelo cuenta con un elemento de retardo (ATM) que modela el tiempo en el cajero y una cola de individuos esperando para usar el cajero. Además un servicio que internamente tiene un conjunto de recursos, en este caso los agentes, una cola de espera y un mecanismo para reservar agentes, otro para liberarlos y un retardo para simular el tiempo del individuo con el agente. Finalmente un sumidero para eliminar los individuos del sistema.

Una vez simulado podemos recolectar estadísticas sobre la longitud de las colas, el uso de los gestores, etc.



4. Simulación Basada en Agentes

La simulación basada en agentes modela un sistema como un conjunto de agentes que, interactuando entre sí, tienen definido un comportamiento individual y posiblemente unos objetivos. Los modelos basados en agentes son esencialmente descentralizados, con un enfoque individual (en oposición al nivel de sistema de los modelos de Dinámica de Sistemas). En un modelo basado en agentes identificamos los agentes (personas, empresas, proyectos, activos, vehículos, ciudades, animales, barcos, productos, etc.), definimos su comportamiento, los ubicamos en un determinado medio ambiente y establecemos conexiones con otros agentes. El comportamiento global, macroscópico, surge entonces como resultado de las interacciones de los comportamientos individuales de los agentes.

Antes de decidirse por un modelo basado en agentes debemos comprobar si los enfoques tradicionales pueden ser utilizados. Por ejemplo, si el sistema que se está modelando encaja bien con el paradigma de procesos (es decir, puede ser descrito como una secuencia de acciones sobre los agentes) puede ser beneficioso considerar a los agentes como objetos pasivos y escoger un enfoque de procesos, usando una librería de componentes adecuada, en lugar de especificar el comportamiento individual cada agente. Del mismo modo, si los agentes en el sistema no tienen individualidad y estamos interesados en las propiedades macroscópicas, agregadas, puede ser mejor el enfoque de Dinámica de Sistemas.

Como hemos comentado los agentes pueden representar cosas muy diversas: vehículos, proyectos, productos, ideas, organizaciones, inversiones, parcelas de tierra, personas en diferentes roles, etc. Los agentes son los principales bloques constructivos de este tipo de modelos, tienen asociado un estado interno y pueden tener asociado un comportamiento, memoria (historia), contactos, etc. Dentro de un agente se pueden definir variables, eventos, diagramas de transición de estados, modelos de Dinámica de Sistemas. También se puede

incrustar otros agentes o añadir diagramas de flujo de proceso. Se pueden definir tantos tipos de agentes como sea necesario.

El diseño de un agente comienza con la identificación de sus propiedades, su comportamiento y la interfaz con el mundo exterior. En el caso de un gran número de agentes con conexiones dinámicas, como las redes sociales, los agentes pueden comunicarse mediante llamadas a funciones.

El estado interno del agente y su comportamiento se pueden implementar en un número de maneras. El estado del agente puede ser representado por una serie de variables, por un diagrama de estados y transiciones, etc. El comportamiento puede ser pasivo (sólo reaccionan a las llegadas de mensajes o llamadas de función y no tienen su propio tiempo), o activo, cuando la dinámica interna (tiempos de espera, dinámica de procesos internos o valores de variables) del agente hace que éste actúe.

El modelado con agentes es esencialmente jerárquico. Un agente puede contener en su interior otros agentes o poblaciones de agentes. En agente contenedor lo denominaremos entorno de los agentes contenidos. Un agente puede contener además un diagrama de niveles y flujos (es decir una ecuación diferencial), un proceso como hemos visto anteriormente en los modelos de eventos discretos, constantes, variables, statecharts, etc.

Definiendo y creando agentes

Un agente es similar a un objeto activo, es decir un objeto que tiene actividad interna. Esta actividad interna le permite reaccionar ante eventos ejecutando acciones o cuando se cumple alguna condición sobre el valor de sus propiedades. Como los objetos, los agentes son de un tipo que tiene propiedades y un estado interno. Un tipo puede heredar de otro. La actividad interna de un agente se diseña mediante un diagrama de estados y transiciones que denominaremos un *statechart*.

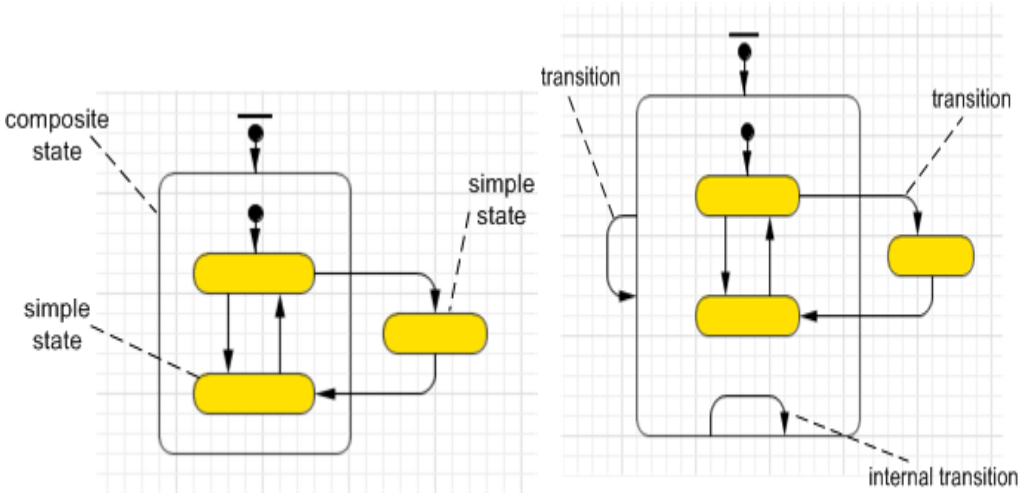
Diagrama de estados y transiciones: Statechart

Un diagrama de estados y transiciones, un statechart, es una herramienta adecuada para especificar el comportamiento de un agente. Un *statechart* tiene estados y transiciones. Las transiciones pueden ser desencadenadas por las condiciones definidas por el usuario (tiempos de espera, mensajes recibidos o condiciones booleanas). La ejecución de una transición puede conducir a un nuevo estado.

Los elementos de un *statechart* son:

- Punto de entrada: estado inicial de un *statechart*
- Estado: Un punto de control dónde el agente tiene capacidades de reacción a eventos, o a condiciones sobre los valores de sus variables. Puede ser simple o compuesto
- Estado inicial: Estado inicial de un estado compuesto
- Estado final: Punto final del *statechart*

- Transición: Una transición indica un cambio de un estado a otro. Tras dispararse una transición se cambia de un estado a otro y tiene asociada una acción.
- Punto de unión o ramificación: Representa punto de ramificación o de conexión entre transiciones. Es adecuado para crear una transición que tiene más de un estado destino y también para indicar que varias transiciones se fusionan entre sí para realizar una acción común. Puede tener asociada una condición para decidir la rama de la bifurcación a tomar.



La transición puede ser disparada como resultado de varios tipos de eventos:

- Después de que transcurra el tiempo especificado (*timeout*)
- En un número de veces por unidad de tiempo (*rate*)
- Cuando la condición indicada se convierte en verdadera
- Cuando el mensaje que coincida con el descriptor especificado es recibida por el *statechart*
- Cuando un agente llega a su destino.

El statatechart como tipo de datos

Una de la formas de disparar una transición es mediante la recepción de un mensaje. Hay dos formas de enviar un mensaje a un *statechart:*

- Llamar al método *receiveMessage ()*
- Llamar al método *fireEvent ()*

Llamando a *receiveMessage()* no se asume ningún tipo de cola para los mensajes entrantes. Si el mensaje recibido no puede disparar inmediatamente una transición, se descarta. Por lo tanto, si, por ejemplo, hay dos transiciones: una (de estado S0 a S1) provocada por el mensaje A, y otra (de S1 a S2) activada por el mensaje B, y la *statechart* recibe los mensajes A y B al mismo tiempo estando en el estado S0, sólo se disparará la primera transición, y el mensaje B será descartado.

Cuando se llama a *fireEvent ()* los mensajes se añaden a una cola. El mensaje añadido a la cola se puede consumir inmediatamente o después de una serie pasos de longitud cero de

medidas, o de lo contrario será descartado. En el ejemplo anterior ambas transiciones se tomará si los mensajes A y B se reciben a través de método *fireEvent().*

Utilizar *fireEvent ()* es menos eficiente que usar *receiveMessage()* .

Por otra parte el *statechart* es un tipo de datos que ofrece los siguientes métodos:

- *void fireEvent(Object msg):* Añade el mensaje a la cola de mensajes
- *Agent getAgent():* Devuelve el agente que posee el statechart
- *boolean isStateActive(short state):* Verdadero si el statechart se encuentra en ese estado
- *void onChange():* Debería ser llamado si el *statechar*t tiene al menos una transición de tipo *Condición* o Tasa cuando algo cambia en el agente y, probablemente, la tasa cambia o condición sea hace verdadera.
- *boolean receiveMessage(int msg):* Similar al siguiente método
- *boolean receiveMessage(Object msg):* Se enviará un mensaje a la *statechart* sin hacer cola: el mensaje o bien se consume inmediatamente (si hay una transición coincidente activa) o se descarta. Si dos o más mensajes llegan simultáneamente y ambos el disparo de una transición, se ignorará el segundo.

Propiedades y métodos

Para cada tipo de agente podemos definir, como en Java, variables, parámetros y funciones. Como en la programación orientada a objetos todos los elementos de un modelo (eventos, gráficos de estado, los objetos de la biblioteca, etc.) se representan mediante objetos de un tipo determinado.

La mayor parte del código que se escribe es el código de un tipo de agente, más precisamente es el código de uno de los métodos de ese tipo de agente. No importa si se está definiendo una acción asociada a un evento o escribir el código de inicio de un agente.

Para acceder a una propiedad o método de un elemento se escribe punto "." después del nombre del elemento y luego escribir el nombre de la propiedad o método. Por ejemplo, para obtener el tamaño de una cola se escribe *queue.size ().* Para acceder a un elemento de una población se indica el nombre de la colección y un índice. Por ejemplo *employees.get (246) .performance ().*

Cada agente tiene un elemento que representa el agente contenedor y para acceder a sus propiedades se sigue la misma regla. Así si el objeto contenedor es *main* entonces se puede escribir *main.announceSale ()* para llamar a un método del mismo.

Para llamar a una función de un elemento del agente se debe poner, de forma similar, el nombre del elemento seguido de punto "." antes de la llamada a la función: <objecto> .<Método>.

Por ejemplo:

- *statechart.receiveMessage("Go!")* : Manda mensaje al objeto s*tatechart*
- *statechart.inState(going). Comprueba si el objeto está en un estado*

- *ob.getOwner():* Devuelve el objeto propietario
- *ob.getIndex():* Devuelve el índice del objeto en la población donde se encuentra

Población de agentes

Como en el caso de los objetos llamaremos población de un tipo de agentes al conjunto de agentes creados y no eliminados hasta ese momento. Los modelos de agentes están dotados de mecanismos para crear agentes, eliminarlos o crear poblaciones completas de los mismos. Una población de agentes estará contenida dentro de un entorno que es un agente contenedor.

Sobre los agentes de una población se pueden calcular los estadísticos usuales: contar los agentes que cumplen una condición, suma, media, mínimo, máximo de los valores de una expresión evaluada sobre un agente.

Las propiedades estadísticas definen una función disponible.

Entorno

Un agente puede desempeñar el papel de entorno para las poblaciones de otros agentes o para otros agentes individuales. Así los entornos pueden ser organizados jerárquicamente (agentes de tipo empresa pueden vivir en un medio ambiente y agentes-empleados pueden vivir en entornos locales de una empresa).

Interacción entre agentes

Agentes que viven en un determina do entorno pueden comunicarse a través de envío de mensajes entre sí y por otros mecanismos como variables o parámetros compartidos.

Variables (o parámetros) compartidas de dos agentes son variables conectadas entre sí que tienen el mismo valor en cualquier momento del tiempo. Es decir, los cambios de una se propagan inmediatamente a la otra.

Los mensajes y su envío puede ser implementada de muchas maneras diferentes. La primera forma es que cada agente tenga *enlaces a vecinos* en su entorno y usar esos enlaces para enviar y recibir mensajes. El conjunto de enlaces puede ser visto como una red que puede ser definida por el entorno y complementada con enlaces específicos.

Los enlaces entre agentes pueden definirse utilizando el elemento enlace a agentes (*Link to agents*). Por ejemplo, si una persona tiene padres, cónyuge e hijos, entonces el agente correspondiente puede tener las conexiones:

- *Person padre*
- *Person madre*
- *Person children (múltiple)*

Los enlaces pueden ser simples o múltiples, unidireccionales o bidireccionales. Una vez definidas las conexiones hay inicializarlas añadiendo agentes (o eliminando) mediante los métodos:

- *connectTo(Agent a)*
- *boolean disconnect()*

A través de las conexiones definidas e inicializadas podemos enviar mensajes con los métodos:

- *send(Object msg):* Envía un mensaje al agente conectado mediante un enlace simple.
- *send(Object msg, Agent dest):* Envía un mensaje al agente concreto conectado mediante un enlace múltiple
- *sendToAllConnected(Object msg):* Envía un mensaje a todos los agentes conectados mediante un enlace múltiple.
- *sendToRandomConnected(Object msg):* Envía un mensaje a un agente concreto elegido al azar de entre los conectado mediante un enlace múltiple.

Las conexiones definidas por el entorno se mantienen, en *AnyLogic*, en la propiedad *connnections* que está definida por defecto y pueden formar distintos tipos de redes. Algunos de estos tipos de redes son:

- *Random*: Los agentes están conectados al azar con un número medio dado de conexiones por agente
- *Basada en una distancia*: Dos agentes están conectados si la distancia entre ellos es de menos de un valor determinado
- *Anillo*: Las conexiones de los agentes forman un anillo en donde cada agente está conectado a un número dado de agentes más cercanos
- *Pequeño mundo*: Puede considerarse como anillo donde algunos enlaces han sido reconectados a los agentes que están a larga distancia
- *Libre:* Algunos agentes son centros con una gran cantidad de conexiones y otros son ermitaños con pocas

Los tipos de red y los parámetros de la misma (como el número de vecinos) se fijan en el entorno donde resida el agente.

La red anterior se puede complementar con otros enlaces individuales.

Las conexiones tienen asociados mecanismos para tratar con los mensajes recibidos. Estos mecanismos son:

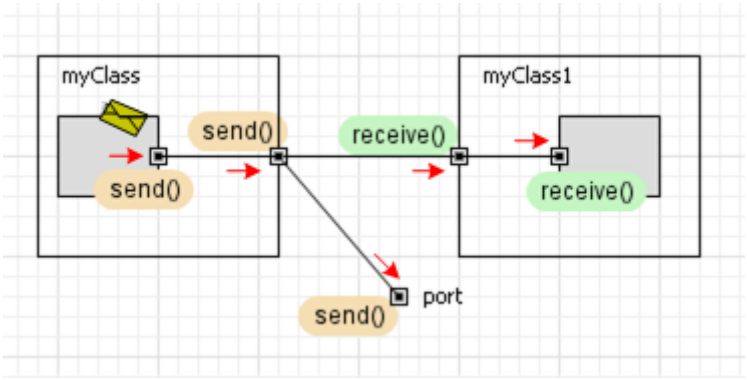
- Indicar el tipo del mensaje que puede ser *int, double, boolean, String* o de una clase Java indicando otros.
- Código a ejecutar cuando se recibe el mensaje.
- Reenviar mensaje a. Aquí se puede elegir el *statechart* u otro elemento del agente.

La siguiente forma de envío de mensajes es mediante *puertos y conectores*. Los mensajes pueden enviarse y recibirse a través de puertos que son elementos bidireccionales que pueden

servir para entrada y salida. Dos puertos en diferentes agentes se conectan mediante conectores que son caminos a través de los cuales fluyen mensajes.

Cuando se envía un mensaje a través de un puerto, éste se reenvía a lo largo de todos los conectores que parten del mismo. Cuando se recibe un mensaje en un puerto, se reenvía a todos los *statechart* del agente y a través de los conectores que parten de ese puerto.

Cuando llega un mensaje a un *statechart*, es recibido y se llama al método *receiveMessage()* del mismo.



Para enviar un mensaje, simplemente debe llamar al método *send()* de un puerto. Por ejemplo:

- *portA.send (nuevo mensaje ());*

Para recibir mensajes a través de un puerto es necesario definir una forma de gestionarlos asociando un código a la propiedad *On Receive* del puerto. Este código se ejecuta cada vez que se recibe un mensaje y puede usar el mensaje entrante. El mensaje recibido se puede reenviar a un *statechart*. En ese caso los mensajes aceptados por el puerto se colocan en la cola asociada al *statechart*.

De Dinámica de Sistemas a Modelado con Agentes

Podemos considerar un modelo de Dinámica de Sistemas como una cadena de niveles y flujos por dónde va pasando una entidad. Determinadas reglas de decisión controlan los flujos a partir del valor de los niveles y otras variables intermedias. En cada nivel vemos, de forma agregada, un número de entidades. El punto de partida clave, para pasar de un modelo de Dinámica de Sistema a otro de Agentes, es desagregar los flujos y los niveles. Es decir ver los niveles como cajas que contienen elementos discretos en vez de un valor representado por una variable agregada. Estos elementos se convertirán en agentes. Posteriormente se trata de ver los que está sucediendo a cada agente, cuantos estados tiene y como transita de un estado a otro. Las transiciones entre estados y el momento en que ocurren vienen determinados por la velocidad de los flujos. La transición entre los estados se puede implementar de varias maneras: síncrona, cuando la decisión se toma cada paso el tiempo *dt*, y asíncrona cuando la

transición produce un nuevo evento en el futuro. En esta última opción el tiempo evoluciona a saltos de un evento al siguiente y la simulación suele ser mucho más eficiente.

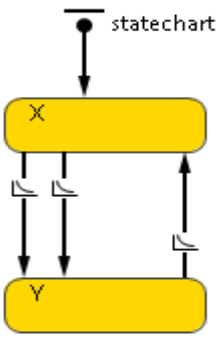
Para que la simulación basada en agentes tenga interés frente a los modelos de Dinámica de Sistemas los agentes deben tener algún tipo de actividad por sí mismos.

Es posible pasar de un modelo de sistemas a otro basado en agentes siguiendo las ideas:

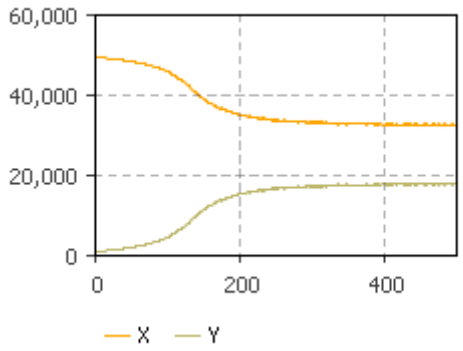
- Determinar los distintos tipos de agentes que podrían considerarse en el modelo de Dinámica de Sistemas
- Asignar a cada uno de los agentes una vida consistente en el paso por una secuencia de niveles y flujos.
- Convertir la secuencia de niveles y flujos en un *statechart* dónde cada nivel va a dar lugar a un estado y los flujos nos permiten determinar las transiciones a otros estados y el tiempo de estancia en ese estado.
- Añadir al agente capacidades de reacción a eventos difíciles de modelar en Dinámica de Sistemas
- Calcular el número de agentes necesarios a usar

Como ejemplo consideremos un modelo de agentes similar al de Dinámica de Sistemas que modelaba la difusión de un nuevo producto. Podemos considerar un único agente que sería una persona interesada en comprar un nuevo producto. La persona se encontraría en dos estados: X e Y (individuos que todavía no han comprado el producto pero son potenciales compradores y los que ya lo tienen). Un individuo pasa del estado X a Y por dos razones: por innovación o por imitación. De los individuos en estado X un porcentaje de los mismos compra el producto porque lo considera innovador, otro porque imita a otros con los que se encuentra y ya lo tienen. La primera razón podemos modelarla como una tasa de innovación constante de los individuos en X, la proporción para modelar la segunda razón depende del número de individuos con los que se encuentra y una tasa de imitación. El paso del estado Y a X lo hace el individuo después de que el producto ha llegado a ser obsoleto. Es decir después de su vida media.

Los agentes anteriores los ubicamos dentro de otro agente, su entorno. Este dispone de una población de personas inicialmente en el estado X. El *statachart* de la Persona es de la forma:



Hay dos estados que corresponden a los existentes en el modelo de Dinámica de Sistemas. Los flujos del modelo dan lugar a transiciones. Estas disparadas con mecanismos de tipo *rate*. Es decir disparando la transición un número de veces promedio por unidad de tiempo. Con un número de agentes inicial igual a 50000 todos en el estado X la evolución obtenida es de la forma:



Partiendo del número de agentes en el estado *X*, que representaremos por x, el número de agentes en *Y*, que representaremos por *y*, la tasa de innovación (*tin*), la tasa de contacto (*tc*), la tasa de imitación (*tim*) y la vida media del producto (*vm*), las tasas de disparo de las transiciones son: *tin, tim*tc*y,1/vm.*

Alternativamente cuando conocemos el tiempo medio de estancia en un estado, vm, podemos usar un disparo de tipo *timeout(vm).*

Tasas de disparo de transiciones y flujos en Dinámica de Sistemas

Supongamos un estado de un Modelo de Dinámica de Sistemas denominado *X* cuyo valor en un instante del tiempo sea *x*. Sea un flujo calculado con una tasa constante α y proporcional al valor del estado. Es decir $f = \alpha\, x$. Esto quiere decir que el número de individuos que salen del estado por unidad de tiempo es $\alpha\, x$. En el modelado basado en agentes tenemos que considerar cada agente individualmente. Concluimos que si en una población de x individuos ocurren $\alpha\, x$ eventos de salida por unidad de tiempo por cada agente individual ocurren α. La transición correspondiente hay que programarla para que ocurra α veces por unidad de tiempo (*rate*(α)). Esta idea se generaliza al caso de que α no sea constante y dependa del valor de otras variables.

De forma similar se calcula la tasa de disparo cuando los individuos que entran a un estado permanecen un tiempo medio en el mismo de *v*. En este caso el número de individuos que salen del estado será $\frac{n}{v}$, y la tasa individual adecuada para el disparo de las transiciones $\alpha = \frac{1}{v}$.

Con las anteriores ideas podemos construir un modelo de agentes a partir de otro de Dinámica de Sistemas y posteriormente enriquecerlo con mecanismos que podamos captar a nivel individual pero difícil de modelar a nivel macroscópico.

De Simulación de eventos discretos a Modelado con Agentes

En un modelo de simulación de eventos discretos tenemos entidades que son creadas y fluyen a lo largo de un proceso. Piden recursos, esperan en colas para conseguirlos, los usan, los liberan, y al final salen del sistema. Las entidades se convertirán en agentes. Las unidades de recursos también pueden ser modeladas como agentes si es necesario.

Tras su creación, el agente solicitará un servicio (pero puede que no lo consiga de inmediato), espera en estado el recurso (corresponde a la entidad espera en la cola de un Servicio). Cuando se concede el recurso, el agente pasa al estado que usa el recuso y está en él tiempo de uso y, cuando haya terminado, decida si solicitar otro servicio o espera. Cuando ha terminado, el agente sale del diagrama de flujo.

5. Modelos de Procesos de Negocio

Un proceso de negocio es una secuencia de actividades o tareas interconectadas que se llevan a cabo, en una empresa o institución, para crear un producto o proporcionar un servicio a un cliente. El modelado de procesos de negocio (BPM) es la actividad de representación de los procesos de una empresa buscando analizar o mejorar el proceso. El objetivo empresarial es, a menudo, aumentar la velocidad de proceso o reducir el tiempo de ciclo; aumentar la calidad; o reducir costes, como la mano de obra, materiales o los costes de capital. A menudo la decisión de invertir en el modelado de procesos de negocio está motivada por la necesidad de documentar los requisitos de un proyecto de tecnologías de la información.

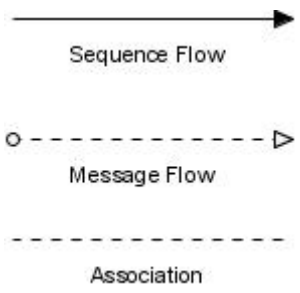
Business Process Model Notation (BPMN) es una representación gráfica para especificar los procesos de negocio en un modelo de proceso de negocio. Los modelos BPMN consisten en diagramas construidos a partir de un conjunto limitado de elementos gráficos. Los elementos básicos de BPMN son:

- *Elementos del modelo*: Eventos, Actividades y Puertas de Control
    - Un evento, representado por un círculo, denota algo que sucede (en comparación con una actividad, que es algo que se hace). Hay varios tipos de eventos:

    - Una actividad: se representa con un rectángulo de esquinas redondeadas, describe un trabajo que se debe hacer

- Punto de decisión: se representa con un rombo con alguna anotación adicional, representa si se bifurca o se fusionan caminos del proceso de negocio. Los tipos de puertas de control son:

- *Conexiones*. Los elementos del modelo están conectados entre sí utilizando conectores de tipo secuencia, mensaje o asociación. Se representan por líneas.



- *Artefactos*: Elementos que permiten añadir información al modelo. Algunos son: datos, grupos y anotaciones.

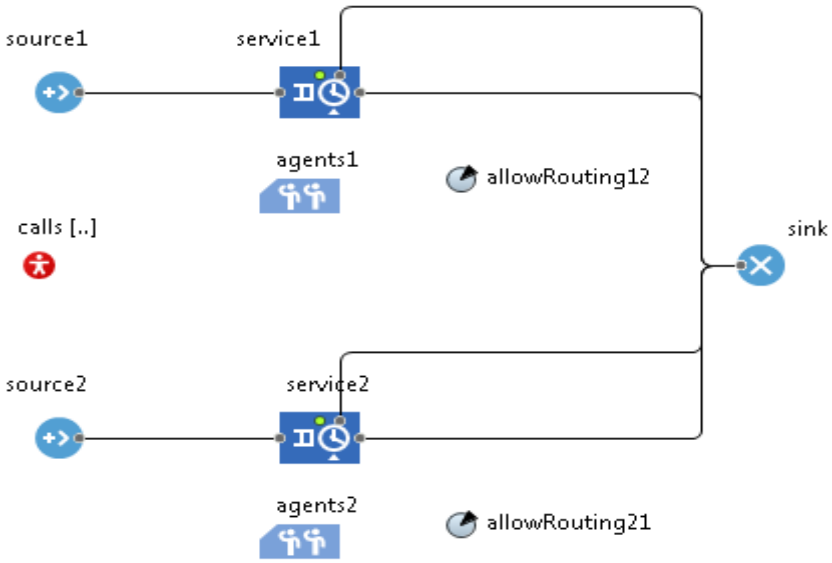
- *Contenedores*: Elementos para agregar un conjunto de actividades que se pueden utilizar para organizar y categorizar diferentes las actividades que se llevan a cabo por diferentes departamentos o para distinguir entre participantes humanos y del sistema, etc.

Un modelo de negocio descrito mediante BPMN puede ser completado con informaciones sobre la duración media de las tareas, los retardos en la transmisión de la información etc. Con esa información es posible construir un modelo de simulación con alguna de las técnicas vistas anteriormente o una combinación de ellas.

La simulación nos permite analizar y evaluar la eficacia de los procesos de negocio ya existentes en la empresa, para elaborar un plan de mejora que aborde los problemas descubiertos, para mejorar los procesos de negocio existentes, y desarrollar nuevos procesos de negocio basados en los experimentos de simulación.

Como ejemplo podemos considerar un centro de llamadas con dos tipos de expertos con diferentes habilidades y dos tipos de llamadas entrantes. El modelo se centra en enrutamiento de llamadas. Las llamadas llegan con determinadas frecuencias y se colocan en las colas correspondientes (una cola para cada tipo de llamada). Algunas personas que llaman abandonan la cola después de un cierto tiempo de espera. Cada tipo de experto está

capacitado para gestionar un tipo de llamada en particular. Junto con las llamadas en las que están especializados los expertos también puedes atender llamadas de otro tipo, pero menos eficiente. La lógica de enrutamiento de llamadas es el siguiente: una llamada se dirige al agente especializado en ella, si hay uno disponible; de lo contrario, la llamada se dirige a otro experto cualquiera, si está disponible. Las métricas en este modelo son las longitudes de las colas de espera y los niveles de servicio para ambos tipos de llamadas.



El modelo anterior podría ser una forma de modelar, de forma simplificada, el centro de llamadas anterior. Las llamadas se modelan como agentes que pueden decidir abandonar la cola de espera. Los expertos se agrupan en servicios con un número de expertos cada uno, colas de espera, mecanismos para abandonar la espera, mecanismos de enrutamiento de un servicio al otro, etc.

6. Modelos en sistemas sociales

Las interacciones de las personas en la sociedad son de la naturaleza sumamente compleja. Satisfacer necesidades cambiantes de los ciudadanos, obtener las mejores respuestas a problemas complejos no es tarea fácil. Por otra parte los estudios de marketing para cada nuevo anuncio de producto y servicio necesitan modelar la difusión de las ideas. En muchos casos el trabajo se orienta a l búsqueda de un grupo de referencia que representa la media de la población a la que está destinado el producto o servicio puesto que los estudios a gran escala son costosos.

Una metodología con cada vez más apoyos es construir modelos de simulación para explorar las hipótesis y comprobar sus consecuencias. Para ello es posible modelar una población de agentes con comportamientos conocidos o que se proponen como hipótesis para pronosticar el comportamiento global de la sociedad.

La simulación basada en agentes se utiliza a menudo en la investigación de nuevas hipótesis en sistemas sociales. Un beneficio del modelado y la simulación posterior es el hecho de las hipótesis formuladas, al tener que ser encajadas en un modelo, tienen que ser formuladas más coherentemente. La simulación puede ser útil cuando no hay otra manera de observar los agentes durante sus interacciones y cuando se trata de obtener leyes macroscópicas a partir de un comportamiento individual formulado como hipótesis más unas interacciones entre los individuos asumidas.

Muchos modelos se han construido de una pequeña población, sus negocios, sus viviendas y sus infraestructuras de. Los tipos de objetos activos (agentes) en este modelo son las personas y las empresas. La ciudad puede contar zonas con diferentes condiciones para las empresas y barrios de diferentes calidades. Se puede modelar mecanismos por lo que las personas se cambian de trabajo o se cambian de barrio. Se pueden incluir en el modelo los tiempos de viaje, los gastos del mismo y el precio de alquiler. El modelo podría calcular las emisiones de CO2 resultante según el tipo de transporte usado.

Un modelo de un mercado de telecomunicaciones diseñado desde el punto de vista de un operador en competencia con los demás. La elección de los usuarios móviles se basa en el precio de comercialización y la agresividad de los operadores. Los usuarios también mantienen una cierta lealtad a la marca. Además de las llamadas de voz cada operador ofrece servicios de valor añadido (VAS) que también generan ingresos. El uso de voz y VAS, así como la probabilidad de adopción y dejar el mercado dependen de la edad del usuario. Mientras que algunos usuarios dejan el mercado, los nuevos usuarios llegan. Aquí los agentes son los clientes que van tomando decisiones de cambiarse de operador, continuar con el mismo o salir del sistema dependiendo de los precios, la lealtad a la marca, etc. También son agentes podemos considerar agentes a los operadores si queremos modelar el mercado completo.

7. Simulación y Optimización

A veces es necesario observar el comportamiento del sistema en determinadas condiciones, o mejorar el rendimiento del mismo, por ejemplo, mediante el ajuste de algunos parámetros y/o la estructura del sistema. En estos casos hablamos de optimización de un modelo. La optimización es el proceso de encontrar la combinación óptima de condiciones que producen la mejor solución posible. La optimización puede ayudarle a encontrar, por ejemplo, el rendimiento óptimo de un servidor o el mejor método para procesamiento de facturas.

Para concretar el tipo de optimización que queremos debemos concretar la siguiente información:

- *Parámetros de optimización*. Son los parámetros que queremos mover, dentro de un rango, para encontrar el mejor valor posible. Puede haber varios parámetros de optimización. El espacio de búsqueda depende del número de parámetros de simulación

- *Función objetivo*: El objetivo del proceso de optimización es encontrar los valores de los parámetros que producen un máximo o mínimo de la función objetivo. La función objetivo es una expresión matemática construida con los parámetros de optimización o los resultados de la simulación.
- *Restricciones*: Una restricción es una condición definida sobre los parámetros de optimización. Cada vez que el motor de optimización genera un nuevo conjunto de valores para los parámetros de optimización, se comprueba si estos valores satisfacen las restricciones definidas. Si lo hacen, el motor funciona el modelo con estos valores. Si no, el motor genera otro conjunto de valores, y así sucesivamente. Así, el espacio de búsqueda se reduce, y la optimización se lleva a cabo más rápido.
- *Requisitos*: Un requisito es una restricción adicional impuesta sobre la solución encontrada por el motor de optimización. Los requisitos se comprueban al final de cada simulación, y si no se cumplen, los parámetros utilizados son rechazados. De lo contrario se aceptan los parámetros.
- *Condición de parada de la simulación*: Especifica las condiciones para que una simulación termine. Una condición habitual es acabar cuando haya transcurrido un determinado tiempo.
- *Condición de parada de la optimización*: La optimización puede detener bajo dos circunstancias: el número máximo de simulaciones se ha excedido o el valor de la función objetivo deja mejorar de manera significativa.

En la optimización de modelos estocásticos es posible usar replicaciones de una simulación. Esta característica permite tener en cuenta resultados de múltiples repeticiones de una simulación. Es posible ejecutar un número fijo de replicaciones por simulación o un número variable de replicaciones por simulación. Esta característica permite de analizar la significación estadística de los objetivos a optimizar.

8. Validación de modelos de simulación

En los modelos de simulación se habla de: validación del modelo, verificación de la implementación modelo, y validación operacional. En general, podemos clasificar los métodos de validación de los modelos de simulación en dos grupos principales: a) métodos subjetivos, y b) métodos cuantitativos. Los métodos subjetivos dependen en gran medida del juicio de los expertos del dominio. Un criterio particularmente útil para la validación de un modelo es la respuesta a la pregunta: ¿La estructura del modelo tiene sentido y es coherente? Los métodos cuantitativos, a menudo llamados métodos estadísticos, incluyen una variedad de técnicas estadísticas. La comparación de modelo a modelo se utiliza cuando existe otro modelo, que modela el mismo fenómeno. Comparar varios modelos desarrollados independientemente es una técnica adecuada de validación.

La validación de modelos y en particular la validación de los que representan sistemas sociales no es una tarea trivial. Un sistema puede proporcionar múltiples niveles de la correspondencia con un modelo, tanto a niveles agregados como a un nivel micro. Escoger estas

correspondencias y elegir adecuadamente los datos del sistema real para comparar es una tarea que puede ser costosa pero que debemos abordar a para convencernos y convencer a los demás de la validez del modelo.