Diseño y Simulación de Sistemas de Colas con *e-Motions*

José M. Bautista, Javier Troya, and Antonio Vallecillo

GISUM/Atenea Research Group. Universidad de Málaga (Spain).
{jmbautista,javiertc,av}@lcc.uma.es

Resumen Este trabajo propone el uso de un lenguaje de dominio específico de alto nivel y ejecutable para analizar el rendimiento de un sistema de colas implementado mediante diferentes estrategias. En concreto se utiliza un enfoque basado en el Desarrollo del Software Dirigido por Modelos (DSDM) para modelar el comportamiento del sistema de facturación de una aerolínea y obtener datos relativos a su rendimiento en cuanto a costes y tiempos de espera, dependiendo de la estrategia utilizada. Para lograr ese propósito se utiliza *e-Motions*, una herramienta que permite al usuario modelar y analizar sistemas en tiempo real de forma gráfica.

1. Introducción

Los sistemas de colas son ampliamente usados, tanto en nuestra vida cotidiana, como en el ámbito de la Ingeniería Software. Existe una gran variedad de situaciones reales que se pueden modelar utilizando la Teoría de Colas, desde comprar entradas en las taquillas de un cine a acceder a los servidores de una página web. Por este motivo es interesante analizar la calidad de servicio que proporciona este tipo de sistemas. Estudiando los resultados obtenidos del análisis se pueden modificar o implementar mejores sistemas de control de congestión u optimizar la planificación de procesos que gestionan las colas.

En este trabajo mostramos un enfoque DSDM para diseñar y modelar un sistema de colas de modo preciso y eficaz con un alto nivel de abstracción. Para llevar a cabo nuestro propósito hemos empleado *e-Motions*, un lenguaje que permite especificar, analizar y simular sistemas en tiempo real. Mediante su interfaz gráfica es posible definir el comportamiento de un lenguaje de dominio específico de manera intuitiva.

Para ilustrar nuestro enfoque hemos utilizado como ejemplo el sistema de colas de facturación de una aerolínea. Hemos estudiado diferentes estrategias con el fin de optimizar el tiempo de espera de los pasajeros en las colas manteniendo un coste razonable para la aerolínea.

La estructura de este artículo es la siguiente. La Sección 2 introduce el caso de estudio en el que se basa nuestro análisis y los parámetros de calidad estudiados. A continuación, la Sección 3 describe el comportamiento dinámico de dicho sistema para su simulación y análisis, como posteriormente comentamos en la Sección 4. Por último, la sección 5 presenta las conclusiones.

2. Caso de estudio

Un ejemplo típico de los sistemas de colas que pretendemos analizar se encuentra en los mostradores de facturación de una aerolínea. Los parámetros de calidad en este tipo de sistemas dependen fundamentalmente de dos factores: el tiempo medio de espera de los pasajeros, y el coste asociado al número de mostradores abiertos (cada mostrador tiene un coste a pagar al aeropuerto). Encontrar el óptimo en cada caso es importante para las compañias aéreas.

Una de las primeras decisiones a tomar en el diseño de este tipo de sistemas es la relativa al modelo a escoger. Podemos por ejemplo seguir un modelo español, en donde cada mostrador tiene asociada una cola, o bien un modelo anglosajón, en donde hay una cola común para todos los mostradores. Los españoles siempre nos hemos quejado que nuestro sistema es peor. En este trabajo analizamos ambos sistemas y comparamos los resultados obtenidos.

2.1. Planteamiento del problema

Se supone que el vuelo lo componen $T = 240$ viajeros. El periodo de facturación es de dos horas y todos los pasajeros llegan a la zona de facturación en ese periodo. La frecuencia con que lo hacen sigue una campana de Gauss, llegando el mayor número de pasajeros justo una hora antes del cierre.

Supongamos que la aerolínea dispone de un máximo de $N = 5$ mostradores de facturación. Cada mostrador tiene asociada una cola de pasajeros a los que atender. El tamaño máximo de la cola viene determinado por una variable $\mathsf{maxSize}_i, i = 1..N$ (cada mostrador puede tener un tamaño de cola distinto). Asimismo, el tiempo de servicio de cada mostrador viene determinado por una distribución exponencial de media $\mathsf{serviceTime}_i$ $(i = 1..N)$ puesto que cada mostrador puede tardar más o menos en servir a los clientes (esto modela, por ejemplo, que la persona atendiendo el mostrador tenga más o menos experiencia, o que sea más o menos espabilado). Inicialmente supondremos que todos los mostradores siguen la misma distribución, de media $\mathsf{serviceTime}_i = 60$ segundos.

En el sistema anglosajón, cuando un pasajero llega un encargado de la aerolínea (el gestor del embarque, o *dispatcher*) le indica a cuál de los mostradores ha de dirigirse para facturar. Los pasajeros esperan en una cola a que el *dispatcher* los redirija a uno de los mostradores abiertos. En este caso, el tamaño máximo de la cola asociada a cada mostrador es 1, y la cola del *dispatcher* se supone no acotada. En el sistema español, por contra, el tamaño máximo de la cola asociada a cada mostrador es no acotada.

El número de mostradores abiertos puede ser fijo a lo largo de todo el embarque, o bien irse abriendo o cerrando dependiendo de las decisiones del *dispatcher* (que puede seguir diferentes algoritmos). Por ejemplo, en el sistema español el *dispatcher* siempre remite a los pasajeros a la cola menos poblada, y ordena abrir un mostrador cuando el tamaño de alguna de las colas de los mostradores abiertos sobrepasa un cierto umbral. Igualmente, cuando alguna de las colas baja de otro umbral determinado el *dispatcher* ordena cerrarla. El mostrador cerrado sigue operando hasta servir a los viajeros que tenía en la cola, aunque no admite

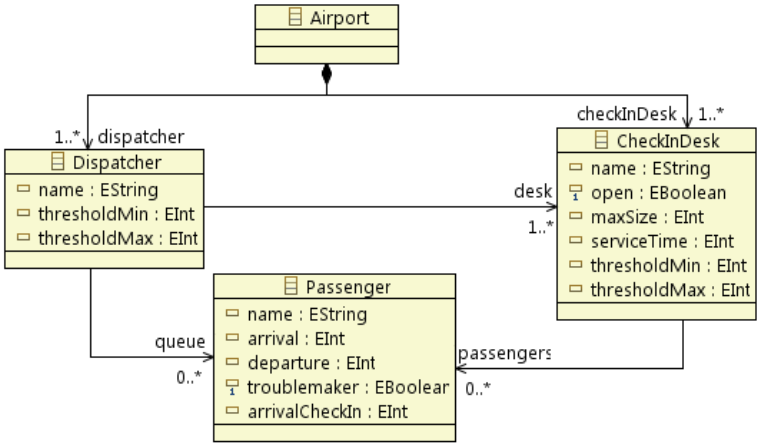


Figura 1. Metamodelo Airport.

a nuevos viajeros (a menos que lo abran otra vez mientras servía a los viajeros que estaban en su cola, algo que puede suceder). El tiempo que trascurre entre que se ordena abrir un mostrador y éste se abre es constante (supongamos que es de 5 minutos).

También vamos a suponer que de los 240 pasajeros hay algunos de ellos que pueden dar problemas, añadiendo algunos minutos a su tiempo de servicio (por ejemplo porque han olvidado el pasaporte, no encuentran el billete o llevan equipaje de más). Supongamos que este tiempo añadido se distribuye según una distribución exponencial de parámetro 5 (es decir, que en media va a relentizar 5 minutos el procesado de sus billetes). Esto le puede pasar a cualquier pasajero, independientemente de cuando llegue.

2.2. Modelado del sistema

El metamodelo del sistema se muestra en la figura 1. En un aeropuerto hay personal de gestión de embarque, representado aquí por la clase Dispatcher, y mostradores de facturación ó CheckInDesk. Para tomar decisiones el dispatcher cuenta con unos umbrales prefijados, uno mínimo y uno máximo, de tamaño de la cola (thresholdMin y thresholdMax).

Los mostradores cuentan con un atributo, open, para indicar si están abiertos o cerrados al público. En caso de estar abiertos tendrán una cola de pasajeros cuyo tamaño máximo vendrá determinado por el atributo maxSize. Cada mostrador empleará por defecto un tiempo, definido a partir del atributo serviceTime, en atender a cada pasajero. Para facilitar la labor de control de congestión de las colas, se establece para cada mostrador unos umbrales mínimo y máximo de tamaño de cola (thresholdMin y thresholdMax).

Los pasajeros de la aerolínea almacenan en el atributo arrival el instante de llegada al aeropuerto, momento en el cual se dirigen al gestor de embarque,

que les informa del mostrador al que deben acudir. Una vez que el pasajero se presenta en la cola del mostrador que le ha sido asignado se registra tanto el instante en que llega a dicha cola, mediante el atributo arrivalCheckIn, como el momento en que abandona el mostrador una vez realizada la facturación, en el atributo departure. Para identificar a los pasajeros problemáticos se emplea el atributo troublemaker.

2.3. Parámetros a analizar. Observadores

Para analizar el "mejor" sistema es preciso determinar un criterio de funcionamiento óptimo. En nuestro caso hemos tratado de buscar un equilibrio entre el tiempo medio de servicio de los clientes y el coste que supone a la compañía el embarque. Para ello nos hemos centrado en los siguientes parámetros:

- **Tiempo medio de servicio**. Tiempo medio transcurrido desde que un pasajero entra en la zona de facturación hasta que abandona el mostrador con los billetes.
- **Coste**. Coste total del proceso de embarque. Es el producto de la suma de las unidades de tiempo que cada mostrador ha estado abierto durante la facturación y el coste por unidad de tiempo que supone a la aerolínea un mostrador abierto. Hemos considerado que cada minuto que un mostrador está abierto cuesta 5 euros (variable cpu en la fig. 8).
- **Tiempo máximo de espera**. Máximo de los tiempos de espera de los pasajeros.

Para medir estos parámetros utilizamos los *observadores*, elementos que permiten monitorizar sistemas modelados con lenguages específicos de dominio [1]. Para ello basta con definir un metamodelo con los observadores que queremos incluir, y añadirlo al del propio sistema. En *e-Motions*, esto es posible hacerlo de forma no-intrusiva, es decir, sin modificar el metamodelo de nuestro sistema. La figura 2 muestra el metamodelo de los observadores definidos en este caso para poder medir las propiedades de calidad de servicio mencionadas anteriormente. A continuación se describen brevemente los tres observadores utililizados:

- GeneralOb. Observador general para contar los pasajeros que llegan, los que ya han facturado, su tiempo medio de espera, el tiempo máximo de espera, el coste total del sistema y el tiempo total que han tardado todos en facturar.
- DispacherOb. Observador asociado al *dispatcher* del sistema. Cuenta el número de pasajeros que ha atendido, el tiempo medio que los pasajeros han estado esperando a ser atendidos y la longitud media de la cola del dispatcher.
- DeskOb. Hay un observador de este tipo asociado a cada mostrador, para contar el número de pasajeros atentidos en el mostrador, el tiempo de apertura, la longitud media de la cola del mostrador y el tiempo medio que los pasajeros han estado esperando en la cola a ser atendidos.

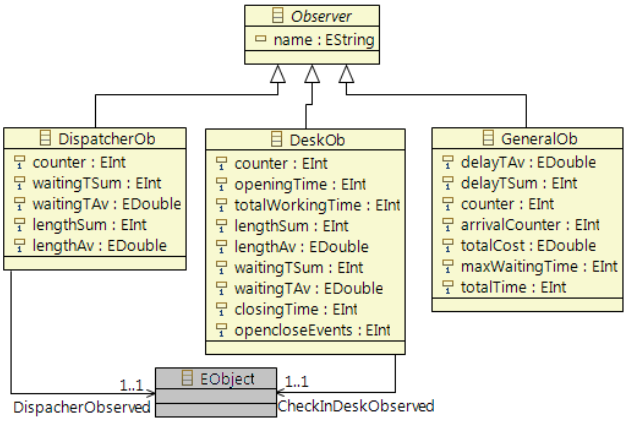


Figura 2. Metamodelo de observadores.

3. Modelado del comportamiento del sistema

3.1. *e-Motions.*

e-Motions [2] es una herramienta que permite especificar, simular y analizar sistemas en tiempo real de forma precisa e intuitiva. *e-Motions* ha sido desarrollado para la plataforma Eclipse.

El comportamiento dinámico de un Lenguaje de Modelado de Dominio Específico se especifica en *e-Motions* mediante transformaciones *in-place*. Este tipo de transformaciones están formadas por un conjunto de reglas, cada una de las cuales representa una posible acción del sistema. Estas reglas son de la forma $l : [\text{NAC}]^* \times \text{LHS} \rightarrow \text{RHS}$, donde l es el nombre de la regla y LHS (left-hand side), RHS (right-hand side) y NAC (negative application conditions) son patrones de modelos que representan ciertos estados del sistema. Una regla se aplica si en el modelo existe una coincidencia con el patrón LHS y no aparece ninguna con los posibles NAC que tenga dicha regla. En el caso de que aparezcan varios patrones coincidentes con LHS se selecciona uno aleatoriamente y se aplica la regla sobre él, de modo que dicho patrón es sustituido por una instancia del modelo representado en RHS. Las reglas se aplican en un orden no determinístico hasta que ninguna es aplicable –aunque este comportamiento puede ser modificado mediante algunos mecanismos de control de ejecución como por ejemplo estrategias [3].

Es importante resaltar que *e-Motions* soporta el uso de expresiones OCL a través de mOdCL [4], el cual implementa y da semántica a OCL en Maude [3]. *e-Motions* permite generar de forma automatizada (mediante transformaciones implementadas en ATL [5]) las correspondientes especificaciones en Maude del modelo diseñado. Maude se utiliza como una notación formal para dar una semántica precisa a las especificaciones de *e-Motions* [6]. De este modo, se podrán realizar tareas de análisis formal y simulación a partir de las especificaciones.

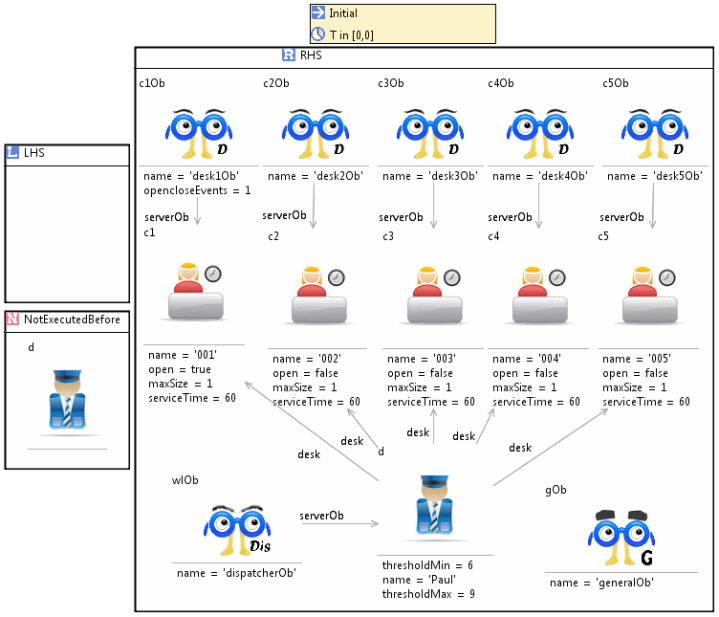


Figura 3. Configuración inicial.

e-Motions facilita la representación explícita de las acciones que modelan las reglas. Incluyendo un objeto denominado Action Execution podemos especificar una acción concreta y el conjunto de objetos involucrados en ella. Este mecanismo es útil para comprobar si un objeto participa en una acción o si una acción ya ha sido ejecutada.

Un tipo especial de objeto, Clock, representa el paso del tiempo. Dicho objeto permite conocer la cantidad de tiempo transcurrido en cualquier instante.

3.2. Comportamiento del sistema de facturación

El comportamiento del sistema se ha especificado mediante un conjunto de reglas que describen cada una de las posibles acciones que pueden llevarse a cabo. Distinguiremos dos estrategias de gestión, la primera basada en una cola (que modela el sistema anglosajón) y la segunda basada en múltiples colas (que se corresponde con el modelo español).

Estrategia de una sola cola (modelo *anglosajón*). Esta estrategia establece que en cada mostrador sólo podrá haber un pasajero, el cual será atendido de inmediato. Los pasajeros que van llegando al aeropuerto forman una cola delante del gestor de embarque que redirige a cada cliente a un mostrador vacío.

En primer lugar, para implementar este funcionamiento, es necesario especificar una configuración inicial del sistema. La regla Initial, mostrada en la figura 3, se encarga de generar el modelo inicial sobre el que se simulará el sistema. La

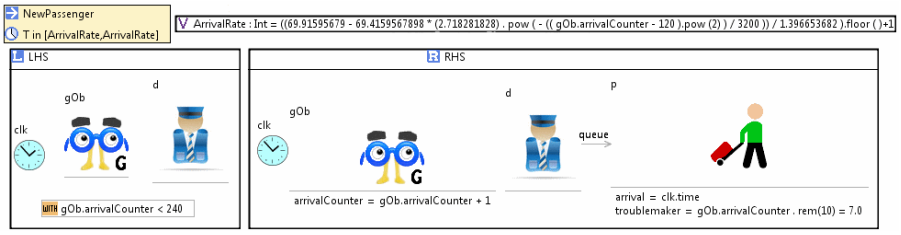


Figura 4. Regla NewPassenger.

inclusión del NAC evita que la regla se lance si ya se ha ejecutado con anterioridad. El LHS no contiene ningún elemento puesto que no es necesaria ninguna condición para que esta regla se ejecute. Como se observa en la imagen, el modelo de inicio está compuesto por un gestor de embarque y cinco mostradores de facturación. Los atributos thresholdMin y thresholdMax del gestor de embarque definen unos umbrales mínimo y máximo que se utilizan como referencia del tamaño de la cola a la hora de tomar decisiones para organizar a los pasajeros. Los mostradores se inicializan con un valor máximo de cola (maxSize) de un pasajero y con un tiempo medio de atención para cada pasajero (serviceTime) de 60 segundos (las unidades de tiempo consideradas en este sistema son los segundos).

La regla NewPassenger (figura 4) simula la llegada de pasajeros al aeropuerto. El observador general llevará la cuenta del número de pasajeros que han llegado a la terminal de salida en su atributo arrivalCounter. Con ello controlamos que el embarque se realice para el número de pasajeros fijado, en este caso 240. El nuevo pasajero, que automáticamente se coloca en la cola frente al gestor de embarque, almacena el instante de llegada en el atributo arrival. El atributo troublemaker determina si el pasajero será problemático, retrasando en tal caso el tiempo total en que se realiza su facturación.

Como mencionamos anteriormente, hemos considerado que la llegada de pasajeros al aeropuerto seguirá una distribución de Gauss, donde el mayor número de pasajeros llega a la mitad del tiempo de embarque. Es decir, los pasajeros comenzarán llegando poco a poco dos horas antes del vuelo, la frecuencia de llegada irá incrementando hasta alcanzar su pico una hora antes del vuelo, para de nuevo empezar a decrementar hasta poco antes de la hora del vuelo, momento en que llega el último pasajero. Mediante esta distribución intentamos modelar la forma en que, normalmente, los pasajeros llegan para hacer su facturación en la vida real. La duración de la regla la determina la variable ArrivalRate. El cálculo que se observa sobre dicha variable es una simplificación de la función de distribución de la campana de Gauss.

El siguiente paso es indicar a cada pasajero el mostrador que va a atenderle, para ello hemos diseñado la regla AssignCheckInDesk (figura 5). Para que la regla se dispare deben darse dos condiciones. Por un lado, el pasajero que va a ser asignado a un mostrador debe ser el primero de la cola, y por otro lado

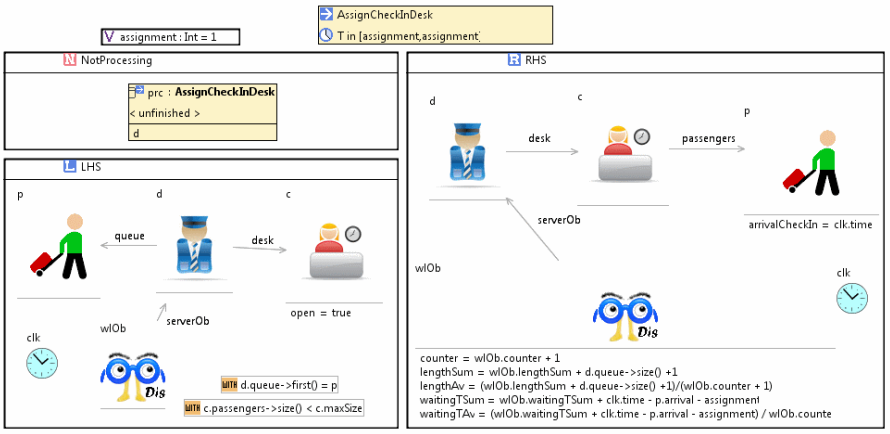


Figura 5. Regla AssignCheckInDesk.

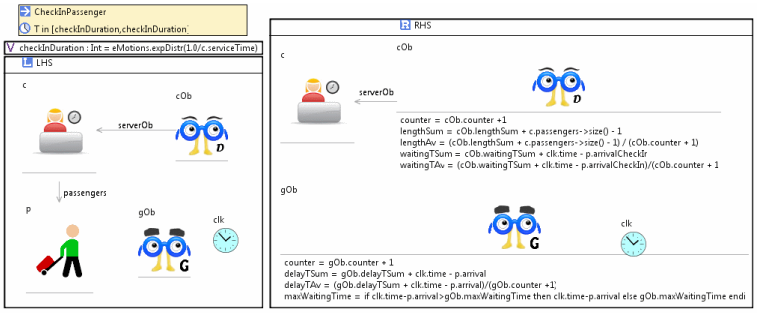


Figura 6. Regla CheckInPassenger.

el mostrador debe tener un número de pasajeros en espera menor al tamaño máximo de cola definido para ese mostrador (para esta estrategia hemos fijado dicho tamaño máximo a uno). El que el atributo open haya de valer *true* hace necesario que el mostrador esté abierto para poder recibir al pasajero. Dado que el *dispatcher* sólo puede atender a un pasajero a la vez, hemos añadido un NAC que evitará que esta regla se active simultáneamente para el mismo *dispatcher* y diferentes pasajeros. Cuando la acción se lleva a cabo, el pasajero pasará a colocarse en el mostrador que le ha sido asignado. Su atributo arrivalCheckIn almacena el instante en que esto se produce. Por último, incluimos un observador que almacena y calcula una serie de datos para obtener el tiempo medio de espera en la cola del *dispatcher*, así como la longitud media de la cola. La duración de esta regla es constante puesto que la acción que representa es la de indicar al pasajero el mostrador que le atenderá. La variable assignment almacena dicho valor que se ha fijado a un segundo.

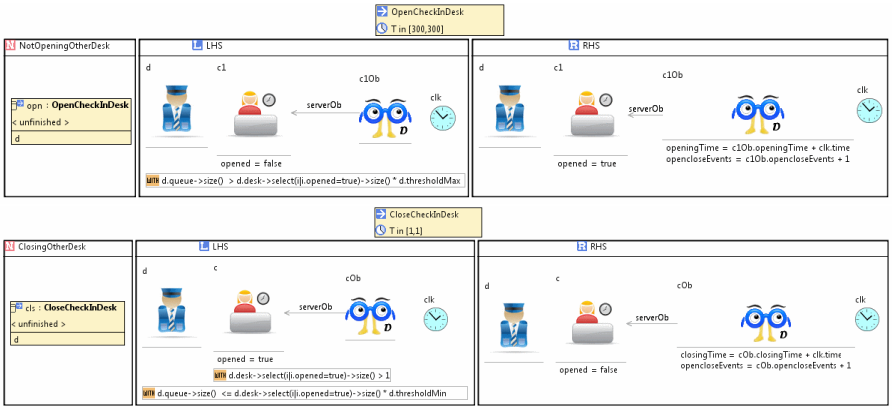


Figura 7. Reglas OpenCheckInDesk y CloseCheckInDesk.

Una vez que un pasajero llega a alguno de los mostradores, podrá obtener su tarjeta de embarque. Esta acción se representa mediante la regla CheckInPassenger, mostrada en la figura 6. Una vez que el pasajero ha facturado se dirigirá a su puerta de embarque, desapareciendo de nuestro sistema. En la parte RHS de la regla se actualizan los valores de los observadores. El tiempo de servicio de cada mostrador es definido por una distribución exponencial cuyo valor medio es determinado mediante el atributo serviceTime de cada mostrador. En nuestro ejemplo, el tiempo de servicio de los pasajeros en los mostradores se distribuye con una función exponencial de media un minuto.

En la configuración inicial de nuestro sistema sólo había abierto un mostrador. Durante la facturación se habilitarán el resto según las necesidades. Para ello se siguen diferentes criterios según la estrategia. Las reglas OpenCheckInDesk y CloseCheckInDesk (figura 7) especifican estas acciones.

En la estrategia de una sola cola se abre un nuevo mostrador cuando el tamaño de la cola formada frente al gestor de embarque sea mayor que el producto entre el número de mostradores abiertos y el umbral máximo definido en el atributo thresholdMax. En nuestro modelo inicial el umbral se ha fijado a 12. Además, el atributo open toma el valor *false* para que esta regla sólo se ejecute sobre mostradores que no estén abiertos. En el RHS el mostrador pasará a cambiará el estado del atributo open, permaneciendo abierto a partir de ese instante para atender pasajeros. Además, el correspondiente observador almacena el instante de apertura y el número de eventos de apertura y cierre que se han producido sobre el actual mostrador para posteriormente poder calcular el tiempo total que ha estado abierto. La duración de esta acción tiene un valor constante de 5 minutos. Con ello se contempla desde que se da la orden de apertura hasta que el empleado activa el mostrador.

El cierre de los mostradores viene especificado por la regla CloseCheckInDesk. Esta acción se lleva a cabo cuando el tamaño de la cola sea menor o igual al

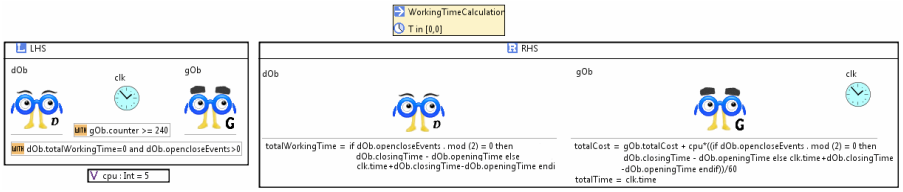


Figura 8. Regla WorkingTimeCalculation.

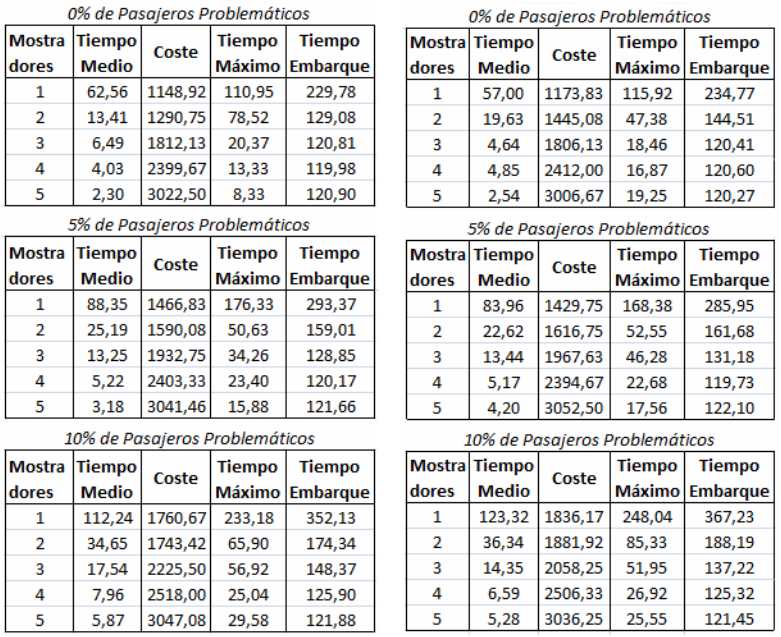
producto entre el número de mostradores abiertos y el umbral definido en el atributo thresholdMin. Su duración es de un segundo.

Una vez que todos los pasajeros han realizado la facturación se ejecuta la regla WorkingTimeCalculation, mostrada en la figura 8. Esta regla realiza un cálculo del tiempo total que ha estado abierto cada mostrador y del coste final que supone a la aerolínea el proceso de facturación. La variable cpu establece el coste por minuto de mantener abierto un mostrador.

Estrategia de múltiples colas (modelo *español*). Esta estrategia admite que se forme una cola frente a cada uno de los mostradores abiertos. El gestor de embarque asigna a cada pasajero el mostrador de facturación con la cola de menor tamaño. En este caso no hemos fijado un tamaño máximo de cola en los mostradores. En el instante en el que una de las colas supere un tamaño establecido se abrirá un nuevo mostrador. Del mismo modo, cuando todas las colas estén por debajo de un valor mínimo predefinido, se procederá a cerrar el mostrador con menos pasajeros en cola.

Para implementar esta estrategia hemos utilizado un conjunto de reglas similares a las vistas para la estrategia de una cola. A pesar de ello, ha sido necesario introducir algunas modificaciones para obtener un comportamiento totalmente diferente del sistema. Así, en la regla Initial se ha definido un valor umbral mínimo (thresholdMin) y otro máximo (thresholdMax) para cada mostrador. Al cambiar la política de colas ha sido necesario introducir una nueva condición en la regla AssignCheckInDesk. Dicha condición viene expresada por la siguiente expresión OCL: d.desk ->select (j | j.open = true) ->forAll (i | c.passengers ->size() <= i.passengers ->size()). El propósito de la misma es seleccionar el mostrador abierto con la cola de menor tamaño. En la regla CheckInPassenger, hemos restringido la ejecución de la misma de modo que sólo pueda llevarse a cabo la acción sobre el pasajero que se encuentra en la primera posición de la cola, puesto que ahora puede haber más de un pasajero en las colas de los mostradores. Esto se ha conseguido mediante la condición: c.passengers ->first() = p.

Por último las condiciones de las reglas OpenCheckInDesk y CloseCheckInDesk han variado para dotar al sistema de la funcionalidad deseada. A la hora de abrir un nuevo mostrador se aplica la condición d.desk ->select(j | j.open = true) ->exists(i | i.passengers ->size() >= i.thresholdMax). Ésta permite que se lleve a cabo la acción cuando la cola de alguno de los mostradores abiertos supera el tamaño máximo definido en su atributo thresholdMax. La condición d.desk ->select(j | j.open=true) ->forAll(i | i.passengers ->size() <= i.thresholdMin) ha-



0% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	62,56	1148,92	110,95	229,78
2	13,41	1290,75	78,52	129,08
3	6,49	1812,13	20,37	120,81
4	4,03	2399,67	13,33	119,98
5	2,30	3022,50	8,33	120,90

5% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	88,35	1466,83	176,33	293,37
2	25,19	1590,08	50,63	159,01
3	13,25	1932,75	34,26	128,85
4	5,22	2403,33	23,40	120,17
5	3,18	3041,46	15,88	121,66

10% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	112,24	1760,67	233,18	352,13
2	34,65	1743,42	65,90	174,34
3	17,54	2225,50	56,92	148,37
4	7,96	2518,00	25,04	125,90
5	5,87	3047,08	29,58	121,88

(a) Estrategia de una cola.

0% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	57,00	1173,83	115,92	234,77
2	19,63	1445,08	47,38	144,51
3	4,64	1806,13	18,46	120,41
4	4,85	2412,00	16,87	120,60
5	2,54	3006,67	19,25	120,27

5% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	83,96	1429,75	168,38	285,95
2	22,62	1616,75	52,55	161,68
3	13,44	1967,63	46,28	131,18
4	5,17	2394,67	22,68	119,73
5	4,20	3052,50	17,56	122,10

10% de Pasajeros Problemáticos

Mostradores	Tiempo Medio	Coste	Tiempo Máximo	Tiempo Embarque
1	123,32	1836,17	248,04	367,23
2	36,34	1881,92	85,33	188,19
3	14,35	2058,25	51,95	137,22
4	6,59	2506,33	26,92	125,32
5	5,28	3036,25	25,55	121,45

(b) Estrategia de varias colas.

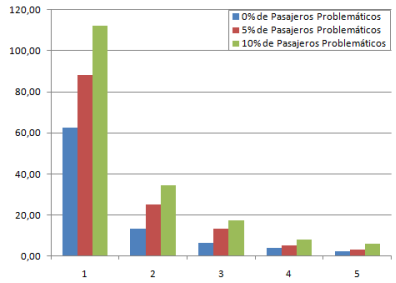
Figura 9. Resultados de las simulaciones (tiempo expresado en minutos).

bilita el cierre de uno de los mostradores cuando el tamaño de la cola de todos los mostradores abiertos sea igual o inferior al valor definido en thresholdMin.

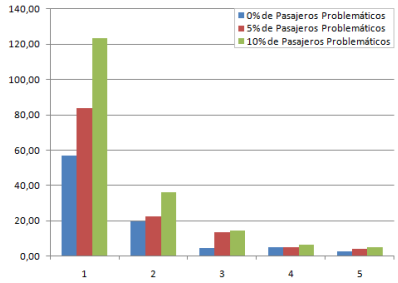
4. Simulación del sistema

En este apartado se muestran resultados obtenidos por los observadores para ambas estrategias. Para cada estrategia se han definido diferentes escenarios de simulación y se ha simulado el sistema varias veces.

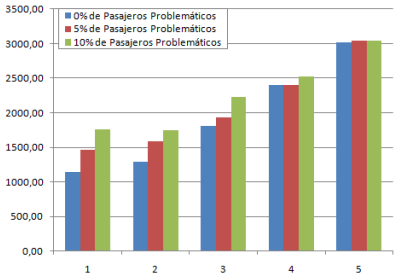
Gestión estática de mostradores. En primer lugar, y para facilitar inicialmente las comparaciones entre los dos modelos, hemos supuesto que el número de mostradores abiertos no se modifica de forma dinámica durante el proceso de facturación, y hemos simulado ambos sistemas para diferentes valores de N (entre 1 y 5). Además, para cada uno de esos casos se han obtenido resultados para diferentes porcentajes de pasajeros problemáticos (0, 5 y 10 por ciento). En cada uno de los diferentes escenarios se han realizado varias simulaciones y se ha calculado la media de los valores obtenidos. Las tablas de las figuras 9(a) y 9(b) muestran los resultados obtenidos de estas simulaciones para los dos modelos.



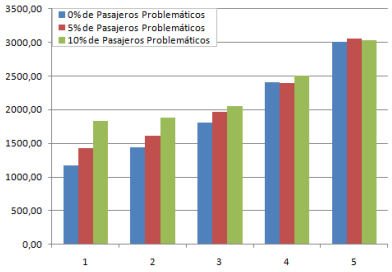
(a) Tiempo medio facturación: una cola.



(b) Tiempo medio facturación: varias colas.



(c) Coste facturación: una cola.



(d) Coste facturación: varias colas.

Figura 10. Tiempo medio y coste de la facturación.

Para facilitar el análisis de los resultados hemos elaborado una serie de gráficos, algunos de los cuales se muestran en las figuras 10(a), 10(b), 10(c) y 10(d). Para todos ellos, el eje X indica el número de mostradores abiertos. Como era previsible, el tiempo medio de espera disminuye y el coste total aumenta conforme el número de mostradores abiertos aumenta. En líneas generales, los resultados son similares para ambas estrategias, lo que indica que *no hay diferencia de rendimiento entre los dos modelos*.

Un dato destacable es la ganancia que se obtiene entre tener uno y dos mostradores abiertos. El tiempo medio de espera disminuye notablemente mientras que el coste se mantiene en valores muy similares. Además el tiempo máximo que un pasajero tiene que esperar en cola también disminuye considerablemente. En un principio se puede pensar que tener un solo mostrador va a ser bastante más barato que tener dos. Sin embargo, con un mostrador, el tiempo total de embarque es mucho mayor que cuando se tienen dos mostradores, lo que hace que aumente el precio. Cuando hay dos mostradores, hay que pagar por ambos, pero el hecho de tener dos hace que los pasajeros embarquen más rápidamente.

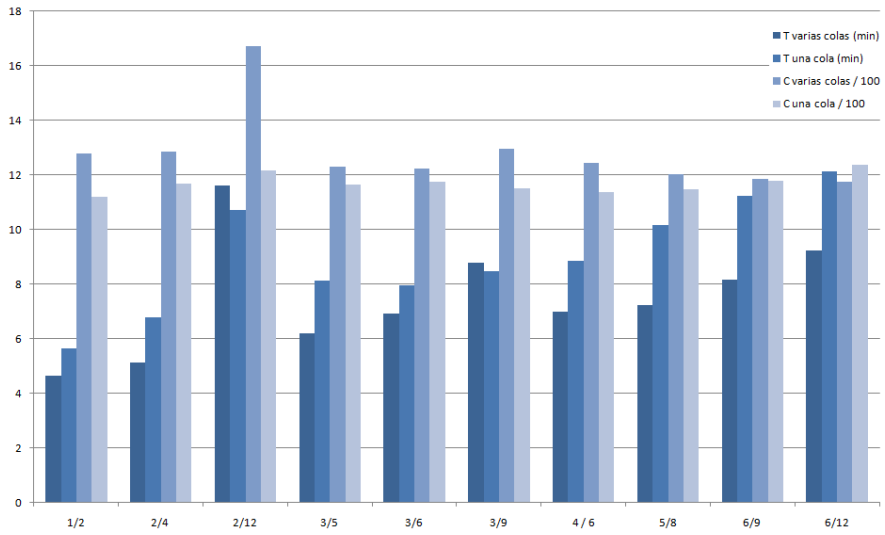


Figura 11. Tiempo y coste de ambas estrategias.

Una situación opuesta ocurre cuando se pasa de tener 4 mostradores activos a 5. En tal caso, la mejora en cuanto a tiempo de espera es muy pequeña frente al incremento de coste que supone abrir un nuevo mostrador. De nuevo la explicación tiene que ver con el tiempo que se tarda en atender a todos los pasajeros, siendo muy similar con 4 y 5 mostradores abiertos. Si analizamos el coste podemos observar que conforme aumenta el número de mostradores abiertos el porcentaje de pasajeros problemáticos afecta en menor medida. Es decir, los costes se asemejan más mientras más mostradores se abren.

Otro punto a destacar es la necesidad de operar con al menos dos mostradores cuando existan pasajeros que reclamen o presenten alguna queja. Nótese, por ejemplo, que si un 10 % de los pasajeros dan problemas la diferencia de abrir uno o dos mostradores es tener un tiempo de espera de 112 minutos o de 34 minutos, respectivamente.

Gestión dinámica de mostradores. Una vez obtenidos estos resultados hemos simulado el sistema aplicando las estrategias de apertura y cierre de mostradores explicadas en la sección 3.2. La figura 11 muestra una gráfica con el tiempo y coste (eje Y) para ambas estrategias dependiendo de varios rangos de umbrales mínimos y máximos (eje X) que determinan la apertura y cierre. El tiempo se muestra en minutos y el coste se muestra divido entre 100. Los valores corresponden a la media obtenida tras ejecutar varias simulaciones.

En la gráfica se observa que para la mayor parte de los umbrales elegidos, el tiempo medio de espera de los pasajeros es menor en el sistema de varias colas. De hecho, esta diferencia es bastante significante para algunos umbrales. Así por ejemplo, para los umbrales en los que los pasajeros esperan más en el sistema con varias colas ($[2, 12]$ y $[3, 9]$), la diferencia de tiempo es muy pequeña.

5. Conclusiones

En este trabajo hemos presentado un ejemplo del uso de los lenguajes ejecutables de dominio específico para analizar sistemas no triviales, de una forma flexible, de alto nivel, y utilizando conceptos y notaciones cercanos a los expertos del dominio. El ejemplo elegido ha sido el de los sistemas de colas usados en algunos países, tratando de determinar la mejor estrategia en cada caso. Hasta donde hemos averiguado, no hemos encontrado estudios que muestren análisis similares a los realizados aquí.

Tras analizar dos sistemas con algunas variantes, los datos obtenidos en cuanto a prestaciones y coste son similares, por lo que a priori ninguna es mejor que otra. Al menos, en cuanto a los parámetros analizados (coste del embarque, tiempo media de espera de los pasajeros, y tiempo máximo de espera).

Aunque hemos visto que desde un punto de vista objetivo (por parte de la aerolínea) no hay diferencia, también es cierto que en España no consideramos igual de justo un sistema u otro. Un factor que puede tener mucha influencia en este juicio es una propiedad que distingue ambos modelos: en el sistema de una cola todo pasajero factura antes que los pasajeros que llegaron después que él, mientras que en el modelo de varias colas puede que un pasajero facture antes que otros que llevaban en el sistema más tiempo que él, pero esperando en otras colas que han ido más lentas. Teniendo en cuenta este aspecto, la estrategia de una cola puede ser considerada más justa (*fair*) que la de varias colas, al menos desde el punto de vista de los pasajeros.

Como trabajo futuro nos planteamos estudiar algunos parámetros más de calidad para complementar el análisis comparativo. En este sentido, un estudio más detallado de la variable aleatoria Υ formada por los tiempos de espera de los pasajeros (además de la media y el máximo) puede dar indicaciones sobre algún aspecto más, como por ejemplo la diferencia entre ellos, o cómo se distribuyen con respecto al tiempo. La ventaja de nuestro enfoque es que es muy fácil diseñar y simular nuevos experimentos para analizar el sistema y comparar estrategias.

Referencias

1. Troya, J., Rivera, J.E., Vallecillo, A.: Simulating domain specific visual models by observation. In: Proc. of DEVS'10. (2010)
2. Rivera, J.E., Durán, F., Vallecillo, A.: A graphical approach for modeling time-dependent behavior of DSLs. In: Proc. VL/HCC'09. (September 2009) 51–55
3. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework. Volume 4350 of LNCS. Springer (2007)
4. Roldán, M., Durán, F.: Representing UML models in mOdCL. Available at `http://maude.lcc.uma.es/mOdCL` (2008)
5. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming **72**(1-2) (2008) 31–39
6. Rivera, J.E., Durán, F., Vallecillo, A.: On the behavioral semantics of real-time domain specific visual languages. In: Proc. WRLA'10. Volume 6381 of LNCS., Springer (2010) 174–190