Universitat Autònoma
de Barcelona

Evaluación de una Metodología para la Resolución de Problemas de Optimización

Memòria del Projecte d'Enginyeria Tècnica en Informàtica de Sistemes
realitzat per
Airam Luis Gutiérrez González
i dirigit per
Mercedes E. Narciso Farias
Escola d'Enginyeria
Sabadell, *Setembre* de *2010*

La sotasignat, ***Mercedes E. Narciso Farias***, professora de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció
per en ***Airam Luis Gutiérrez González***

I per a que consti firma la present.
Sabadell, ***Setembre*** de ***2010***

Signat: ***Mercedes E. Narciso Farias***

Resumen

El presente proyecto tiene como objetivo evaluar una metodología para la resolución de problemas de optimización. Para ello se utiliza el formalismo de modelado de Redes de Petri Coloreadas para representar un Sistema de Eventos Discretos, que describirá el problema de optimización a resolver.

La importancia de seguir una metodología como la que se evalúa en este proyecto es que permite realizar cambios en cualquiera de las fases a fin de obtener los mejores resultados posibles para un problema de optimización.

El caso de estudio a optimizar en este proyecto se conoce como Tiempo de Tránsito de Pasajeros, y se define como el tiempo que tarda un pasajero con escalas en recorrer el trayecto que va desde la puerta de embarque de llegada hasta la puerta de embarque de salida asignada. Este problema está teniendo mucha importancia en la actualidad, dado que muchos aeropuertos están cambiando su modelo de negocio hacia los aeropuertos *hub* o aeropuertos de conexión, lo que implica que deben reducir este tiempo de tránsito al mínimo posible puesto que el objetivo final es aumentar el perfil del pasajero con escala.

Agradecimientos:
A mi futura mujer, Maria, sin su perseverancia para que me centrara en mis estudios no hubiera sido posible finalizar el proyecto.
A mi primo Alejandro y a mi tío Juan Ramón, grandes influencias en mis decisiones académicas.
A mi directora de proyecto, Mercedes, por guiarme durante todo este tiempo, aguantando mis errores.
Y en especial a mis padres, artífices de que me encuentre aquí presentando este proyecto.

Tabla de Contenidos

CAPÍTULO I Introducción

Este proyecto consiste en la evaluación de una metodología para la resolución de problemas de optimización [8]. Dicha metodología consta de seis fases que permiten modelar y simular un sistema para obtener soluciones a problemas de optimización de sistemas reales que se puedan representar como un Sistema de Eventos Discretos (SED en adelante). Se entiende como SED, un sistema en el cual la evolución de su estado depende completamente de la ocurrencia en el tiempo de eventos discretos asíncronos. Los SED son útiles para representar o modelar entornos industriales, logísticos o de planificación de la producción y en la actualidad su uso en estas áreas ha crecido enormemente.

La gran ventaja del modelado y la simulación es que permite la experimentación controlada de representaciones de los sistemas reales sin tener que correr los riesgos que representarían en el mundo real. También permite la reducción del tiempo de pruebas, ya que una simulación se realiza en mucho menos tiempo que el sistema real al que representa [6]. Como herramienta de modelado de esta metodología se utilizará el formalismo de Redes de Petri Coloreadas (RPC en adelante) [7] para poder modelar y simular el SED representado.

1.1 Objetivo del proyecto

El objetivo de este proyecto es evaluar una metodología para resolver problemas de optimización. Para ello, se ha escogido un caso de estudio sobre el que se aplicará esta metodología, cuyos puntos más relevantes son:

- Describir el sistema a optimizar definiendo las entidades, los recursos y los eventos del mismo.
- Desarrollar el modelo RPC del sistema descrito.
- Simular dicho modelo con la ayuda de un simulador de RPCs.
- Aplicar heurísticas[1] para analizar resultados y reducir tiempos de cálculo.

[1] La capacidad heurística es un rasgo característico de los humanos, desde cuyo punto de vista puede describirse como *el arte y la ciencia del descubrimiento y de la invención* o de resolver problemas mediante la creatividad y el pensamiento lateral o pensamiento divergente. (Wikipedia.org) // En algunas ciencias, se entiende como la manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc. (Diccionario de la RAE)

1.2 Descripción del Caso de Estudio

Se optimizará un concepto aeroportuario de gran importancia para muchos aeropuertos internacionales, se trata del *Passengers Transfer* o Tránsito de Pasajeros. Se define como el trayecto que realiza un pasajero con escala desde que aterriza en el aeropuerto de tránsito hasta que despega de él. La minimización de este tiempo cobra mucha importancia en los llamados aeropuertos *hub* o aeropuertos de conexión. Para entender el concepto de aeropuerto *hub*, se puede recurrir al campo de la Telemática (ver *Figura 1*). Un *hub* es un concentrador y como tal, recoge señales de datos de un punto para distribuirlas a cualquier otro punto, es decir, todo lo contrario a una conexión punto a punto.

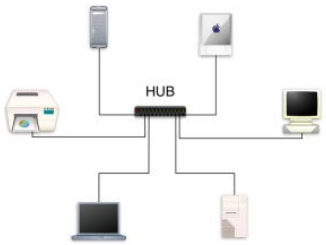


Figura 1: Ejemplo de *hub* telemático

En el ámbito de la aviación, un aeropuerto *hub* se define como un aeropuerto que absorbe el tráfico de otros aeropuertos con destinos muy concretos con el objetivo de descargarlos de tráfico aéreo. Este sistema se conoce como *"hub and spoke"*, es decir centro y distribución. Así se le puede ofrecer a un pasajero la posibilidad de acceder a muchos más destinos sin que tengan que ser ofrecidos por cada aeropuerto y, por lo tanto, minimizando los recursos. Así cada aeropuerto se especializa en una cantidad de destinos concretos, maximizando el tráfico de pasajeros hacia esos destinos [3].

Normalmente estos aeropuertos son sede de una operadora de vuelo que opera la gran mayoría de vuelos desde allí. Un claro ejemplo es el Aeropuerto de Palma de Mallorca, que hace de *hub* para la operadora de vuelo Air Berlín y que sirve de nexo entre distintos puntos de Alemania y España siempre a través de Palma. Ahí radica la importancia de minimizar el tiempo de espera de los pasajeros en el aeropuerto, ya que diariamente reciben una gran cantidad de tránsitos. Por lo que con un solo destino, Ámsterdam-Palma por ejemplo, pueden ofrecer a sus clientes centenares de destinos entre Ámsterdam y cualquier punto de la península, ya sea a través de Air Berlín o de

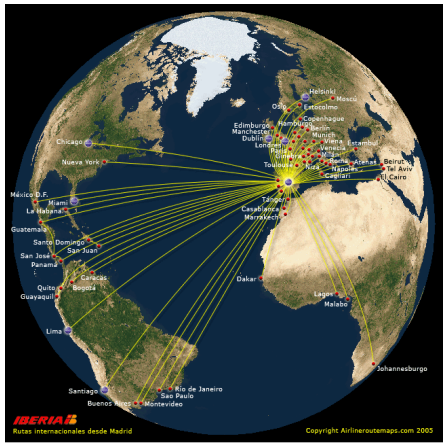


Figura 2: Ejemplo de *hub* en aviación

convenios con otras operadoras. De esta manera el servicio ofrecido al cliente crece de manera significante y sin embargo los recursos utilizados son los mismos.

Para este caso de estudio se tomará la siguiente situación: el pasajero aterriza en una terminal del aeropuerto procedente de otro país y debe dirigirse a otra puerta de embarque en la misma terminal, siendo éste un caso altamente habitual. Durante el trayecto dentro de la terminal el pasajero abandonará la puerta de origen, recorrerá dicha terminal, pasará por el control de aduanas, puesto que es un pasajero procedente de otro país, atravesará los controles de seguridad y finalmente llegará a su puerta de embarque de destino donde esperará hasta su hora de embarque. En ese trayecto los tiempos que son sensibles de disminuir, puesto que vienen determinados directamente por el uso de los recursos humanos de los que dispone el aeropuerto, son el tiempo de paso por el control de aduanas y el tiempo de paso por el arco de seguridad. El resto de tiempos dependen de la estructura del aeropuerto en el momento de su construcción, por lo tanto, tomaremos como hipótesis que no afectan a este modelo como variables.

1.3 Estructura de la memoria

Esta memoria está dividida en 5 capítulos. A continuación se resume muy brevemente el contenido de cada uno.

Capítulo 1 - Introducción: Se describe el objetivo principal del proyecto, así como también el ámbito de estudio general y posteriormente se concreta el caso de estudio.
Capítulo 2 - Estudio de viabilidad: Se analiza la viabilidad del proyecto considerando los recursos de los que se dispone.
Capítulo 3 - Fundamentos teóricos: Se expone en qué consisten las Redes de Petri (RP en adelante) y en concreto las RPC. Posteriormente se define en qué consiste la metodología a desarrollar.
Capítulo 4 - Caso de Estudio: En este capítulo se describe el caso de estudio escogido y se aplican las diferentes fases de la metodología a evaluar.
Capítulo 5 - Conclusiones: Se describen las conclusiones a las que se ha llegado y se proponen líneas futuras de investigación para conseguir posibles mejoras.

CAPÍTULO II Estudio de Viabilidad

En este capítulo se va a analizar la viabilidad económica, organizativa y técnica del proyecto descrito en el capítulo I, teniendo en cuenta las restricciones de tiempo impuestas por la asignatura y los recursos de los que dispone el proyectista [9].

2.1 Objeto de Estudio

2.1.1 Descripción de la situación a tratar

El objetivo principal es la evaluación de una metodología para la resolución de problemas de optimización. Por lo que es necesario establecer si es viable llevar a cabo las seis fases de la metodología teniendo en cuenta los recursos económicos, temporales y humanos de los que se dispone. Dicha metodología se aplicará a un caso de estudio en concreto, como es el del tránsito de pasajeros en aeropuertos de conexión *(hub)*.

2.1.2 Perfil de usuario / cliente

Este proyecto de investigación va dirigido a personas con conocimientos técnicos, como puede ser un ingeniero o un matemático. Debe tener conocimientos sobre problemas complejos de optimización de recursos. Y también debe conocer el formalismo de RPC. No es necesario conocimiento previo en actividades aeroportuarias.

Como perfil objetivo del caso de estudio, este proyecto va orientado a las personas encargadas de gestionar la planificación de tiempos en un aeropuerto, por lo que sólo les haría falta conocer los resultados de la investigación sin entrar a aclarar conceptos técnicos de la realización.

También está orientado a personas que se dediquen a resolver problemas de optimización de recursos en cualquier ámbito. Se les ofrece una herramienta que marca unas pautas (fases) a seguir para lograr el objetivo final, que es resolver un problema concreto de optimización.

2.2 Descripción del sistema a realizar

En este apartado se presenta la metodología a evaluar, los recursos necesarios, los puntos críticos y los procedimientos de solución para los diferentes riesgos que se pueden presentar.

2.2.1 Descripción

Al tratarse de la evaluación de una metodología específica, el desarrollo del proyecto consistirá en seguir paso a paso las fases de la misma, evaluando su conveniencia y los resultados en cada momento. Para ello, a continuación se describen las seis fases de las que consta la metodología en cuestión [8]:

- Fase 1: Descripción del problema a tratar definiendo datos y aspectos relevantes
- Fase 2: Modelado del Sistema a través del formalismo RPC
- Fase 3: Representación del Conocimiento y Heurísticas
- Fase 4: Construcción del Modelo de Decisión
- Fase 5: Simulación del Modelo de Decisión: Espacio de estados y soluciones
- Fase 6: Análisis de Resultados: Mejoras de Optimización

2.2.2 Recursos

Antes de describir los recursos de los que se dispone, es necesario especificar los requerimientos tecnológicos mínimos pensando en una correcta evaluación de la metodología. Los recursos de hardware necesarios son un PC con Windows XP y un procesador de textos básico.

Para la realización de este proyecto se dispone de los siguientes recursos divididos en hardware, software y recursos humanos:

- Hardware:
  - PC Intel Celeron 1.6GHz, 3GB RAM y 250 HDD
  - Impresora Láser B/N Samsung ML-2571N
- Software:
  - Windows XP Professional
  - Open Office 3.2.1
  - Interfaz gráfica para el modelado de RPC

- Simulador de RPC
- Interfaz para el análisis de resultados

➢ Recursos Humanos:

- 1 Diseñador para representar y modelar el sistema.
- 1 Analista para escribir el código fuente, simular y analizar resultados.

2.2.3 Evaluación de Riesgos

Los problemas de mayor envergadura se pueden encontrar a la hora de simular el caso de estudio, por lo que se enumeran a continuación:

- Tiempos de simulación elevados: Debido a que la ejecución de una simulación realiza todas las combinaciones posibles de los valores de las variables, con sólo aumentar uno de éstos el tiempo de simulación puede aumentar enormemente. Para reducir estos tiempos se aplicarán heurísticas. De la misma manera, también se observará el proceso en el mundo real y se utilizarán los datos observados para modificar el modelo de manera que se eliminen soluciones que no se corresponden con la realidad, reduciendo la necesidad de recursos de simulación.

- Bloqueo de procesos en el análisis: Al generar archivos de solución o análisis con gran cantidad de datos, cuando se editan para analizarlos, el PC puede dejar de responder o no abrir el archivo debido a las limitaciones de las interfaces o del procesador de textos. La solución pasa por evitar la generación de archivos de análisis demasiado grandes. En última instancia se podría analizar manualmente, sin ayuda de la aplicación de análisis, pero entonces el tiempo de análisis de los resultados crecería desorbitadamente.

- Pérdida de buenas soluciones: Este riesgo es común a todos los procesos de simulación con heurística, ya que al reducir el espacio de búsqueda[2] para disminuir el tiempo de simulación, se puede cometer el error de eliminar un camino que llegaría a una solución mejor que la mejor solución obtenida hasta el momento. En este caso, nunca se podrá averiguar si se ha obtenido la

[2] El espacio de búsqueda está formado por todos los posibles subconjuntos de caminos generados en la simulación del sistema modelado sin tener en cuenta si llegarán a una buena solución o no. *[es.wikipedia.org]*

mejor solución posible, sólo en el caso de no obtener ninguna solución se deduce que la heurística aplicada no está dando resultado y entonces se procedería a modificar la heurística.

2.2.4 Análisis de Costes Humanos y Materiales

Como en cualquier proyecto, ya sea de investigación o no, la dedicación en horas de los integrantes tiene un coste, y como en este caso se trata de un proyecto final de carrera, se hará un estudio del coste estimado estableciendo 15€/h tanto para el diseñador como para el analista. En la *Tabla 1* se definen todas las tareas que se van a llevar a cabo junto al orden de ejecución de las mismas.

Tabla 1: Costes Humanos del proyecto

Orden	Concepto de la Tarea	Coste en Horas
1	Estudio de Viabilidad	15
2	Estudio de los fundamentos teóricos	35
3	Descripción y Comprensión del problema a tratar	25
4	Modelado del Sistema a través del formalismo RPC	25
5	Representación del Conocimiento	30
6	Construcción del Modelo de Decisión	20
7	Simulación del Modelo de Decisión	40
8	Análisis de Resultados y Mejoras de Optimización	30
9	Documentación	30
	Total Horas:	250 horas
	Total Coste (15€/h)	3750€

En el caso del material a utilizar, a parte del valor del hardware en sí, se define un coste de uso, generado por el gasto lógico del mismo, el uso de consumibles y las posibles averías que pueda tener a lo largo del desarrollo del proyecto. Teniendo en cuenta una vida útil generalizada de cada recurso de 6 años y un tiempo de uso aproximado de 6 meses durante el desarrollo del proyecto, el coste por uso equivaldrá a la doceava parte del valor de ese recurso: 6 años = 72 meses, 72 meses / 6 meses = 12.

Tabla 2: Costes Materiales del Proyecto

Concepto	Coste UD.	Coste por uso
Hp Compaq 2230s	500€	41.70€
Samsung ML-2751N	200€	16.70€
Licencia de Windows XP Professional	140€	11.70€
Software OpenSource (Open Office, PrimoPDF)	0€	0€
Total	840€	70€

Según estos cálculos, sumando los costes de los recursos humanos, más los costes por material, más los costes por el uso de dicho material, se estima un coste total del proyecto de: **3750€+840€+70€=4660€**

2.3 Planificación del Proyecto

2.3.1 Planificación de Tareas

La planificación inicial se ha realizado teniendo en cuenta que el tiempo total disponible ha tenido que ser repartido entre el desarrollo del proyecto y la jornada laboral del estudiante. En la *Figura 3,* se muestra la representación de la planificación de tareas mediante un diagrama de GANTT en el que aparecen las dependencias entre tareas y su consecución en el tiempo.

2.4 Conclusiones

Después de analizar todos los riesgos, recursos y costes del sistema se ha llegado a la conclusión de que el proyecto es viable dedicando las horas de las que se dispone. Este estudio de viabilidad demuestra que en este proyecto de investigación basado en la simulación, el coste de los materiales es mínimo puesto que no es necesario invertir en materiales para experimentar ni para la construcción de ninguna herramienta de pruebas, y por lo tanto económicamente es viable.

Se debe aclarar que la viabilidad de un proyecto sólo analiza costes económicos, humanos y riesgos controlables. Es independiente si los resultados obtenidos al final son mejores o no de los ya existentes, los resultados del proyecto no definen su viabilidad.

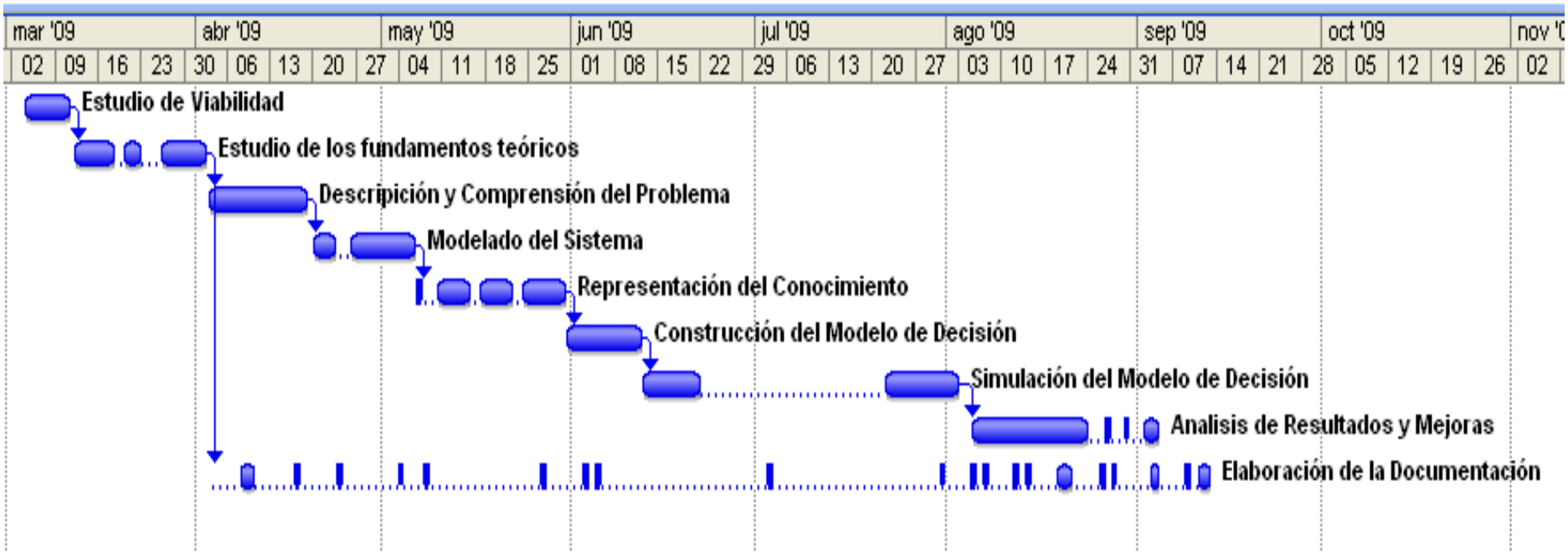


Figura 3: Planificación de las Tareas del Proyecto

CAPÍTULO III Fundamentos Teóricos

En este capítulo se describen los conceptos necesarios para la comprensión del proyecto y la interpretación de sus resultados. Se presenta aquí el concepto de Redes de Petri y una extensión de las mismas denominadas Redes de Petri Coloreadas utilizada por la metodología a evaluar.

3.1 Introducción a las RP

Una RP es un formalismo matemático basado en elementos gráficos, relaciones y reglas que permite representar un SED de manera gráfica [9]. Las RP fueron formuladas en 1962 por el matemático y científico de la computación alemán Carl Adam Petri.

Las Redes de Petri son realmente útiles para modelar y analizar SED, ya que permiten representar el paralelismo y la sincronización, elementos característicos de estos sistemas.

Un SED es un sistema que cambia de estado en instantes irregulares de tiempo, propiciado por la ocurrencia de un evento. Se considera que durante el resto del tiempo, el estado del sistema permanece constante. Los elementos más significativos de un SED son los siguientes [11]:

- **Entidades**: Conjunto de elementos del sistema. Se pueden identificar entidades de dos tipos:
  - Permanentes: Son elementos estáticos en cuanto a que su número no aumenta o disminuye en el sistema. Suelen ser las entidades que ejecutan las actividades, y también se suelen utilizar para definir los recursos del sistema.
  - Temporales: Se utilizan para representar los elementos procesados del sistema, pueden aumentar, disminuir o transformarse. Son las entidades que pueden no permanecer en el sistema permanentemente. En general se utiliza el término recurso para las entidades permanentes y así diferenciarlas de las temporales.

- **Atributos:** Sirven para definir las características de las entidades. El estado de una entidad viene definido por sus atributos y son susceptibles de cambio después de la ocurrencia de cada evento del sistema.

- **Actividades:** Son todas las tareas o procesos que se llevan a cabo en el sistema y por lo general suelen estar encapsuladas entre eventos. Las actividades suelen afectar a más de una entidad al mismo tiempo, por lo que es necesario satisfacer el estado de todas en el mismo instante de tiempo. Es esencial conocer la duración de las actividades con el fin de calcular en qué momento finalizará dicha actividad.

- **Eventos:** Son todas las acciones instantáneas que suelen provocar que el sistema cambie de estado. Que un sistema cambie de estado quiere decir que las entidades y recursos afectados por ese evento han cambiado. De manera general, un evento indica el inicio final de una actividad o tarea del sistema. También pueden representar la llegada de entidades al sistema. Se pueden distinguir dos tipos de eventos:
    - Condicionados: Son aquellos en que deben satisfacerse una serie de condiciones para que sucedan.
    - No condicionados: Son aquellos que tienen planificada su ocurrencia en el tiempo independientemente del estado del sistema.

- **Colas:** También se denominan unidades de almacenamiento, y representan la acumulación de entidades, normalmente temporales.

Tras esta clasificación de elementos, todos los datos y aspectos relevantes del sistema deben quedar representados en alguno de estos grupos. A continuación se mostrará cómo se representan estos elementos en una RP.

3.2 Especificación y Estructura de las RP

Una RP se define como un grafo dirigido, ponderado y bipartito en el que se observan los siguientes elementos [5]:

- **Nodos Lugar:** Gráficamente se representan como círculos o elipses y se pueden usar para modelar colas, recursos y entidades del sistema.
- **Nodos Transición:** Gráficamente se representan como rectángulos y permiten modelar eventos y actividades del sistema. Se dice que una transición está habilitada cuando se cumplen todas las condiciones para la ocurrencia del evento o actividad modelados, a partir de entonces puede producirse el disparo de dicha transición en cualquier instante.
- **Arcos dirigidos:** Gráficamente se representan como una flecha. Conectan un nodo lugar con un nodo transición o viceversa, nunca nodos del mismo tipo.
- **Pesos asociados a los arcos:** Gráficamente se representan como un número junto a un arco al que están asociados. Describen tanto las condiciones necesarias para la ocurrencia de un evento como los efectos sobre el estado del sistema después de la ocurrencia de un evento.
- **Tokens o marcas:** Gráficamente se representan como puntos dentro de un nodo lugar. Se usan para modelar el número de entidades de cada tipo disponibles en el sistema en cada instante.
- **Marcado:** Representa cualquier distribución de tokens en los nodos lugar. El marcado inicial representa el estado inicial del sistema antes de que ocurra ningún evento.

Formalmente una RP es una tupla de cinco elementos,

$$RdP = (P, T, A, W, M_0)$$

$P = (p_1, p_2, p_3, \ldots, p_m)$ Conjunto finito de *m* nodos lugar.

$T = (t_1, t_2, t_3, \ldots, t_n)$ Conjunto finito de *n* nodos transición.

$A = (a_1, a_2, a_3, \ldots, a_i)$ Conjunto finito de *i* arcos.

$A \subseteq (P \times T) \cup (T \times P)$ Subconjunto del producto cartesiano de P y T.

$W : A \rightarrow \{0,1,2,3,\ldots\}$ Peso asociado a cada arco.

$M_0 : P \rightarrow \{0,1,2,3,\ldots, p_m\}$ Marcado inicial del sistema.

Para definir el estado del sistema en un instante dado, se utiliza un vector que está compuesto por el número de tokens de cada nodo lugar en ese mismo instante. Sería,

$$M_i = \left[p_1, p_2, p_3, \ldots, p_m \right]$$

3.3 Ejemplo de una RP

A continuación se muestra, con un ejemplo, el comportamiento de una RP.

> *"En una planta de reciclaje se dispone de máquinas que reciclan metal. En el proceso se convierten 2kg de basura en 1 cubo de metal listo para reutilizar o almacenar y un bulto de material no reciclable listo para descartar. Durante el proceso de reciclaje se gasta 1kg de producto químico por cada 2kg de basura."*

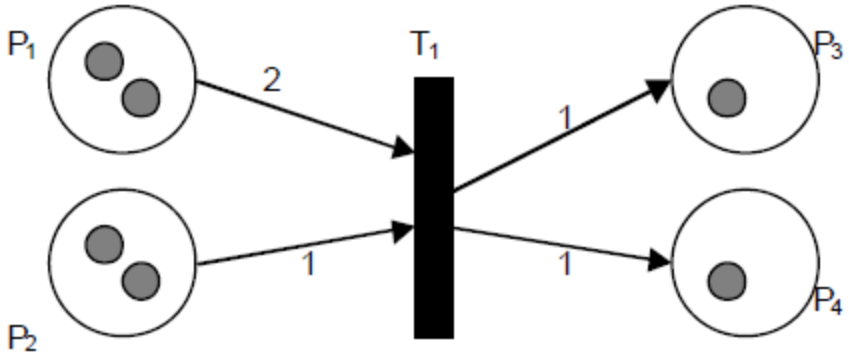


Figura 4: Estado Inicial (M_0) de la RP

En este sistema se definen los siguientes elementos:

Nodos Lugar:

P_1: Material pendiente de reciclar (cada token representa 1kg).

P_2: Compuestos químicos para el reciclaje (cada token representa 1kg).

P_3: Material ya reciclado (cada token representa 1 cubo).

P_4: Material para descartar, (cada token representa 1 bulto).

Nodos Transición:

T_1: Proceso de reciclaje de cada máquina.

Marcado inicial del sistema:

M_0= [(2),(2),(1),(1)]

Arcos del sistema:

P_1-T_1: Peso 2, define 2kg de basura, cantidad necesaria para que se active T_1.

P_2-T_1: Peso 1, representa 1kg de producto químico para que se active T_1.

T_1-P_3: Peso 1, representa 1 cubo de material reutilizable tras activarse T_1.

T_1-P_4: Peso 1, representa la masa de material desechable tras activarse T_1.

Cabe destacar que cada transición modela el funcionamiento de una máquina, por lo que habrá tantas transiciones como máquinas queramos simular.

Cuando el sistema se inicializa, se dispone de 2kg de basura, 2 Kg. de producto químico, un bloque de metal reciclado listo para ser utilizado y un bulto de material desechable. Para que T_1 esté habilitada es necesario disponer de al menos 2kg de metal y al menos 1kg de producto químico, situación que satisface el estado inicial (M_0), por esa razón, T_1 se habilita y el sistema puede cambiar de estado (pasar a M_1).

M_1= [(0),(1),(2),(2)]

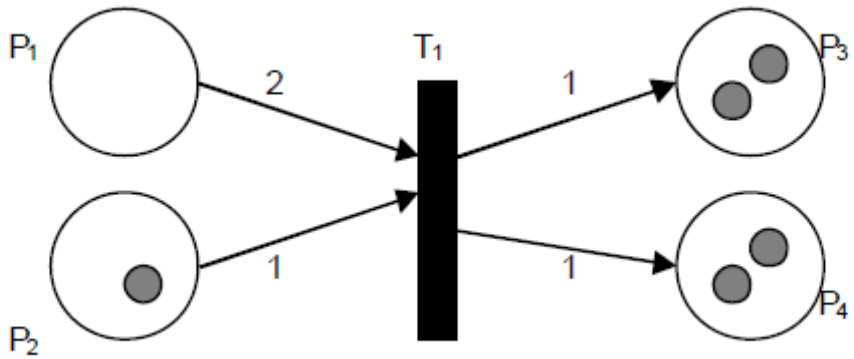


Figura 5: Estado para M_1 de la RP

En este nuevo estado, el sistema tiene 0kg de metal por reciclar, 1kg de producto químico para la descomposición y, como resultado, se genera 1 cubo de metal listo para reutilizar y un bulto de material desechable. En este momento, la transición T_1 ya no se puede volver a disparar, porque aunque se cumple una de las dos condiciones (se

dispone de 1kg de producto químico), no existe basura pendiente de reciclar (son necesarios 2kg de basura) y por lo tanto el sistema se queda en *standby* esperando a que se cumplan las condiciones para que T_1 se vuelva a disparar.

3.4 Redes de Petri Coloreadas

Una vez se han descrito la especificación de las RP, se muestra una extensión de las mismas, las Redes de Petri Coloreadas o RPC. Se trata de un tipo de redes que dispone de mucha más capacidad para modelar sistemas complejos de manera más compacta.

Una de las características principales de las RPC es que, a diferencia de las RP, permiten modelar las propiedades de las entidades que se representan. A continuación se definirán los elementos que diferencian a las RPC de las RP[7 y 8].

- **Conjunto Color:** También se define como colores simplemente a los atributos o características que tiene una entidad. Una característica importante es que todos los tokens que formen parte de un mismo nodo lugar deberán estar definidos por el mismo Conjunto Color.
- **Expresiones de inicialización:** Similar al marcado inicial en las RP, en este caso consiste en el número inicial de tokens dentro de cada nodo lugar pero especificando el valor inicial del color de cada uno de ellos (atributos, características). Se representa subrayando toda la expresión y colocándola al lado del nodo lugar al que corresponde. En caso de que haya tokens dentro de un mismo nodo lugar con distinto valor del color, se debe expresar como una suma de tokens donde cada sumando corresponde al número de tokens que tienen los mismos valores de color.

$$\underline{n_1'(c_{11}, c_{12}, c_{13}, ..., c_{1x}) + n_2'(c_{21}, c_{22}, c_{23}, ..., c_x) + ... + n_a'(c_{a1}, c_{a2}, c_{a3}, ..., c_{ax})}$$

donde,

n_a: número de tokens del nodo lugar con los mismos valores para los colores descritos entre paréntesis.

c_{ax}: valor de un componente del color del token de tipo a.

- **Estado Inicial (M_0):** Corresponde a la configuración inicial del sistema modelado, evaluando el número de tokens de cada nodo lugar y los valores de color de sus atributos.
- **Expresiones de Arco:** Consiste en la formalización de restricciones entre los colores de los distintos tokens de los nodos lugar conectados a la entrada de la transición y de la salida de la transición hacia el nodo lugar de destino.
- **Guardas:** Expresiones lógicas (cierto o falso) que se usan para definir los valores que deben tener algunos de los atributos o componentes de color de los tokens de los nodos lugar conectados a la entrada de la transición para que la transición esté habilitada.
- **Marcado:** representa la información mínima y necesaria para poder predecir cuales son los posibles eventos que pueden producirse. Se representa gráficamente con un número que indica el número de tokens seguido de un apóstrofe y entre paréntesis el valor de color de los atributos del nodo lugar.

Formalmente una RPC es una tupla de nueve elementos,

$$RPC = (\Sigma, P, T, A, N, C, G, E, I)$$

$\Sigma = \{C_1, C_2, C_3, ..., C_n\}$	Conjuntos finitos y no vacíos de colores, donde *n* es el número de conjuntos color.
$P = \{P_1, P_2, P_3, ..., P_m\}$	Conjunto finito de *m* nodos lugar.
$T = \{T_1, T_2, T_3, ..., T_w\}$	Conjunto finito de *w* nodos transición.
$A = \{A_1, A_2, A_3, ..., A_i\}$	Conjunto finito de *i* arcos que conectan P y T y viceversa
$N : A \to (P \times T) \cup (T \times P)$	Función nodo, permite asociar a cada arco un par ordenado de nodos T y P.
$C : P \to \Sigma$	Función color, permite especificar para cada nodo lugar el tipo de entidades que puede almacenar.
$G : T \to bool$	Función guarda, permite asociar a cada nodo transición una expresión lógica.

$E: A \rightarrow C(P_j)$	Expresión de arco, especifica el tipo de token del nodo lugar de entrada/salida asociado.
$I: P \rightarrow C(P_j)$	Función de inicialización, especifica el valor de los colores de los tokens iniciales.

3.4 Ejemplo de una RPC

Para entender mejor los conceptos descritos en el apartado anterior, se expone un ejemplo que muestra todos los elementos que contiene una RPC.

> *"En una empresa disponen de un control de acceso por voz y DNI, cada trabajador debe introducir su tarjeta ID y decir en voz alta su número de DNI para que la puerta les permita la entrada y la salida."*

A continuación se definen todos los elementos necesarios para modelar este sistema.

Nodos Lugar:

Empleados: Nodo lugar que representa a todos los trabajadores de la empresa.

Grabación: Nodo lugar que representa el registro del reconocimiento de voz de cada empleado.

Planta: Nodo lugar que contiene los trabajadores que están trabajando en cada instante.

Nodos Transición:

T1: Control de Acceso a la empresa, en el ejemplo sólo se modela la entrada.

Conjuntos color:

ID: Identificador de cada empleado, corresponde al DNI.

CARD: Tarjeta de identificación de cada empleado.

REC: Grabación de referencia de cada empleado registrada.

REC_ACT: Grabación del empleado en el momento de acceder al control.

Cuando el sistema se inicializa hay dos empleados esperando en la puerta para poder acceder a su puesto de trabajo. En ese momento se acerca el primero, introduce su tarjeta y dice su número de DNI. El sistema lo compara con el registro correspondiente

y si coincide abre la puerta, de lo contrario el sistema se queda esperando otro reconocimiento del empleado o la introducción de la tarjeta del siguiente empleado.

Se guarda registro de todas las grabaciones del día con el objetivo de intentar mejorar el sistema de reconocimiento o actualizar registros de voz si algún empleado accediera mal muchas veces (estadísticas).

Tras establecer todas las características del sistema, se representa en la *Figura 6* el modelado del sistema en su estado inicial.

Partiendo de este estado inicial, se evalúa por primera vez la función guarda de T1, [ID=CARD & REC=REC_ACT)], que necesita que haya un token en el nodo lugar *Grabación* que también exista en el nodo lugar *Empleados*, ese token existe y es *1'(12345678, "12345678")*. Por lo tanto, T1 está activada y puede dispararse en cualquier instante. En la *Figura 7* se observa cómo quedaría el sistema después de que T1 se dispare.

Una vez se ha disparado T1, el nodo lugar *Planta* pasa a tener ese token y ya no hay ningún token en los nodos lugar *Grabación* y *Empleados* que cumpla las condiciones de la función guarda de T1, por lo que el sistema se quedará a la espera de que lleguen nuevos tokens al nodo lugar *Grabación* (ya que *Empleados* es un recurso del sistema y no varía) para volver a evaluar la condición y activarse de nuevo.

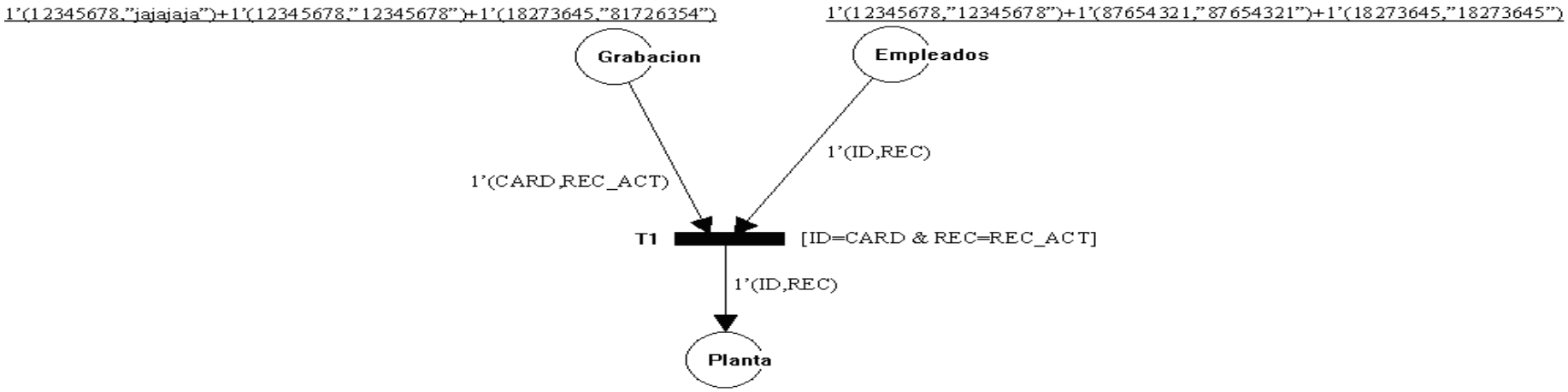


Figura 6: Estado inicial de la RPC.

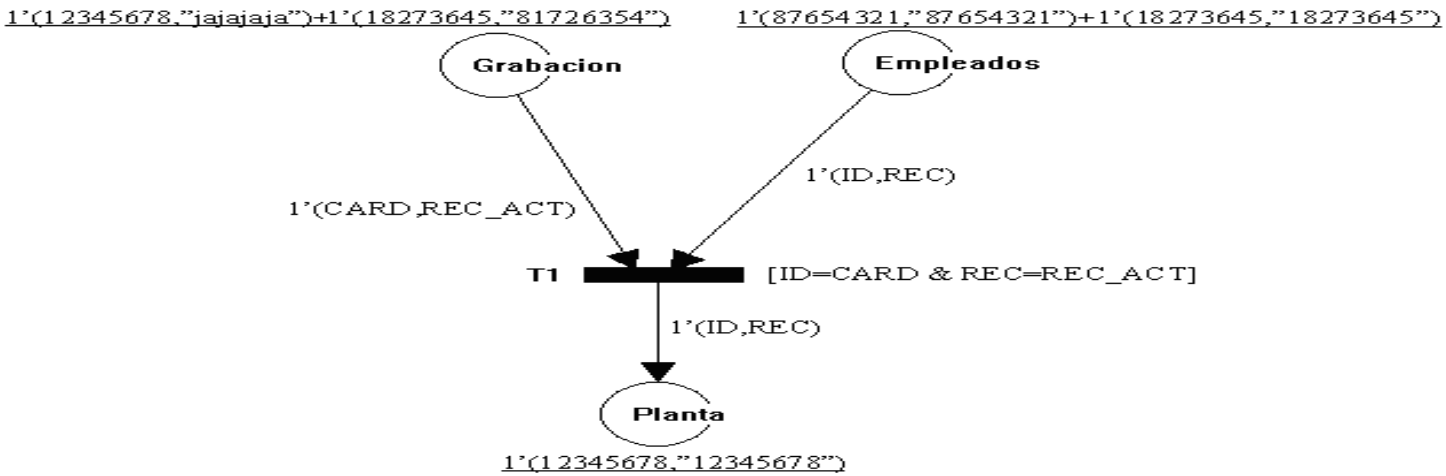


Figura 7: Estado del sistema después de que T1 salte por primera vez.

3.5 Metodología para resolver problemas de optimización

En este apartado se presenta la metodología a evaluar. Esta metodología se define en seis fases claramente diferenciadas. Cada una de estas fases constituye un paso hacia la resolución de un problema de optimización. Esta metodología es aplicable a cualquier problema de optimización que pueda ser representado como un SED, aunque su aplicación es especialmente interesante en problemas de planificación y *scheduling* como el que comprende este proyecto. A continuación se muestra el Diagrama de Fases de dicha metodología[8].

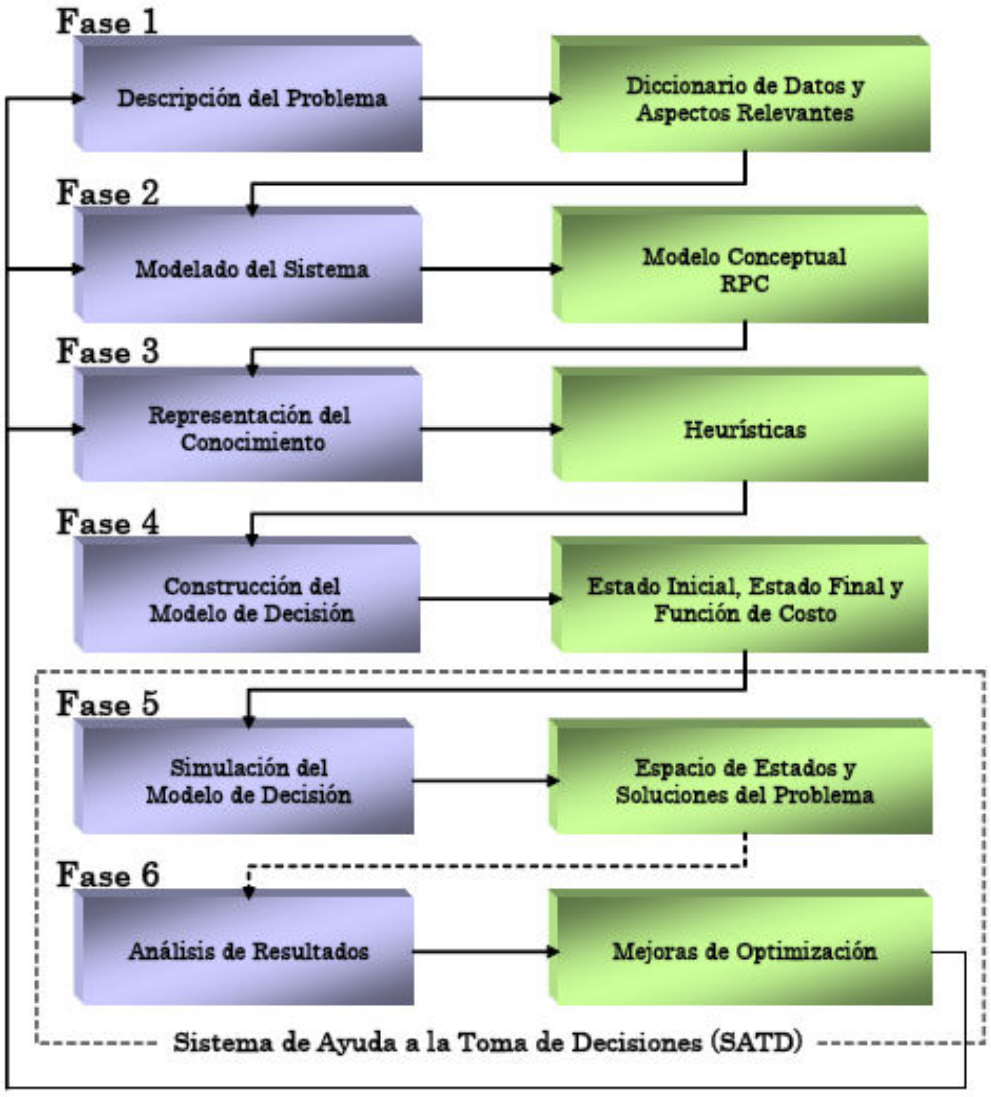


Figura 8: Fases de la Metodología

En la *Figura 8*, los rectángulos de la izquierda identifican la actividad llevada a cabo en cada fase y los rectángulos de la derecha indican el resultado de dicha fase y a su vez, la información de entrada para la siguiente fase. A continuación se describen cada una de estas fases.

3.5.1 Fase 1: Descripción del Problema

Para llevar a cabo esta fase es necesario comprender bien el dominio del problema con el objetivo de abstraer sólo los aspectos más importantes para el desarrollo del modelo. Dentro de esta fase se observan 3 pasos:

3.5.1.1 Abstracción del Problema como un SED

Este paso consiste en enunciar todos los datos y demás aspectos relevantes del problema de tal forma que quede descrito como un SED.

3.5.1.2 Determinación de los Objetivos del Problema

Este paso consiste en la descripción de los objetivos que se desean alcanzar mediante la solución del problema de optimización. Los objetivos deben ser observables, precisos, razonables, comprensibles, medibles y deben poder expresarse en términos de alguno de los estados finales descritos en el apartado *3.1. Introducción a las Redes de Petri.*

3.5.1.3 Determinación de las Dinámicas de Interés del SED

La dinámica de un SED queda determinada por los posibles eventos de inicio o fin de una actividad que comportarán distintos cambios de estado. Los cambios de estados del sistema tienen lugar en el instante de tiempo en el que ocurre un evento. En este paso se deben realizar las siguientes tareas:

- **Lista de actividades,** se debe incluir tiempo y duración de cada una, y **eventos** de interés que pueden generar un cambio de estado en el sistema.
- **Lista de todos los recursos** del sistema, entidades temporales, colas o unidades de almacenamiento.
- **Lista de condiciones** a cumplir para que cada evento pueda ocurrir. Dichas condiciones se denominarán *precondiciones*.
- **Lista para especificar los cambios de estado** ante la ocurrencia de eventos. Estos resultados se denominarán *poscondiciones*.

3.5.2 Fase 2: Modelado del Sistema

En esta fase se lleva a cabo el modelo conceptual del sistema descrito en la fase anterior. Dicho modelo permite plasmar el conocimiento que se tiene del sistema con el objetivo de desarrollar un modelo que represente el sistema real. Por lo que en esta fase se transformarán todos los elementos de un SED al formalismo de las RPC.

3.5.2.1. Especificación de las Relaciones del SED

En este paso se determinan las relaciones estructurales y dinámicas más importantes del sistema, así como las relaciones causa-efecto que describen cómo afecta al sistema la toma de diferentes decisiones.

3.5.2.2. Formalización Gráfica de las Relaciones entre Eventos

Después de especificar todos los elementos del SED, se construye el modelo de RPC que describirá la estructura y el comportamiento de éste. Para ello, es necesario establecer una representación de cada elemento en la RPC.

Se debe tener en cuenta que no existen unas normas que indiquen cómo representar cada elemento del SED en una RPC, ya que esto dependerá mucho de la capacidad de abstracción y creatividad del modelador, pero sí que se pueden definir unas pautas básicas a seguir para facilitar el trabajo inicial.

- Las colas de espera, unidades de almacenamiento, recursos y entidades temporales se pueden representar como nodos lugar en la RPC.
- El estado de los recursos y de las entidades temporales se pueden representar mediante tokens coloreados en los nodos lugar.
- Las variables de estado se pueden asociar a los atributos de las entidades representados como colores de los tokens.
- Los eventos y/o actividades se pueden representar como nodos transición.
- Las precondiciones y poscondiciones se pueden representar mediante las expresiones de los arcos de entrada y de salida respectivamente de la RPC.

3.5.2.3. Análisis de los Efectos de las Posibles Decisiones

Durante este paso de la metodología se analizan aquellas propiedades del sistema que no se encuentran directamente relacionadas con su evolución en el dominio temporal pero que determinan su dinámica, como podrían ser:

- La existencia de bloqueos mutuos en el sistema o *deadlock* que se define como el bloqueo permanente de un conjunto de procesos concurrentes que compiten por los recursos del sistema o bien se comunican entre ellos. Un conjunto de procesos está en *deadlock* cuando todos los procesos en ese conjunto están esperando la ocurrencia de un evento que sólo puede ser causado por otro proceso del conjunto.
- El conjunto de posibles estados que puede alcanzar el sistema a partir de un cierto estado.

Mediante el análisis de la RPC es posible detectar estados no deseados que pueden ser evitados mediante el uso de heurísticas tal y como se explicará en la siguiente fase. Además, durante este paso es posible modificar la RPC a fin de evitar bloqueos mutuos en la dinámica del sistema. Al finalizar esta fase se obtiene un modelo conceptual formalizado mediante una RPC que representa la estructura y comportamiento o dinámica del SED.

3.5.3. Fase 3: Representación del Conocimiento

El uso de una RPC para formalizar la estructura y comportamiento del SED puede ser considerado como una técnica de representación del conocimiento, ya que permite representar entidades y estado de las entidades (nodos lugar y tokens coloreados) y conocimiento (guardas). Al finalizar esta fase se obtiene un conjunto de condiciones en el disparo de uno o más eventos (restricciones), representadas mediante guardas en las transiciones que representan los eventos.

3.5.4. Fase 4: Construcción del Modelo de Decisión

En esta fase, el modelo conceptual definido en la fase 2 y las condiciones obtenidas en la fase 3, se transforman en un modelo de problema de decisión que pueda ser resuelto como un problema de búsqueda, para lo cual es necesario seguir los siguientes pasos:

3.5.4.1 Especificación del Estado Inicial

Este estado representa las condiciones iniciales del problema. En el caso de un SED, representa la situación inicial de los recursos, entidades temporales, colas de espera y unidades de almacenamiento. En un modelo de RPC el Estado Inicial corresponde a la configuración inicial de los tokens que viene dada, a su vez, por las expresiones de inicialización en cada uno de los nodos lugar que componen dicha RPC.

3.5.4.2 Especificación del Objetivo de Optimización

El objetivo de optimización permite determinar si un estado concreto del sistema se puede considerar estado final o estado objetivo. En un SED, este objetivo de optimización representa el valor deseado para cada uno de los elementos del mismo. En el modelo de RPC, corresponde a los marcados de cada nodo lugar que son considerados soluciones al problema de optimización. La calidad de esta solución estará determinada por la creación de una función de costo que evaluará las soluciones a fin de determinar la solución o soluciones de menor costo.

3.5.4.3 Especificación de una Función de Costo

La función de costo permitirá asignar un costo numérico a cada estado del SED, que refleje una medida de rendimiento (tiempo, uso de recursos…). Para determinar la función de costo es necesario especificar cuáles de las variables de estado son consideradas variables de decisión. A fin de asignar valores a las variables de decisión para minimizar la función de costo, es necesario especificar en el modelo las penalizaciones de alcanzar ciertos estados. En una RPC esta información puede formularse mediante una función de costo que pondere el valor de las variables de decisión en los diferentes nodos lugar, de tal manera que la función asigne altos costos a aquellos marcados que correspondan a estados no deseados.

Una vez se han llevado a cabo los tres pasos anteriores, se obtiene un modelo de decisión donde:

- Las variables de decisión son ciertas variables de estado del modelo conceptual, seleccionadas en función de la medida de rendimiento que se desea optimizar.
- La función objetivo es la función de costo que se desea minimizar.

- Las restricciones vienen especificadas por los diferentes valores de los atributos (colores) que pueden tomar los tokens en los diferentes estados finales especificados.
- Las relaciones funcionales son los guardas y las expresiones de arcos de entrada y salida de las transiciones.

La solución de este modelo consistirá entonces en encontrar los valores de las variables de decisión que:

- Cumplan o satisfagan simultáneamente todas las restricciones.
- Alcancen el objetivo establecido, es decir, den el valor mínimo a la función de costo respecto a todas las otras soluciones factibles.

Al finalizar esta fase, se obtiene un modelo de decisión que puede ser resuelto mediante la simulación del modelo de RPC, ya que representa todos sus componentes.

3.5.5. Fase 5: Simulación del Modelo de Decisión

En esta fase se evalúa de modo automático los diferentes escenarios que pueden presentarse en el sistema modelado. Para cada estado alcanzable se deben evaluar y disparar los posibles eventos activados, o un subconjunto de éstos en caso de usar la función de costo como heurística, a fin de generar el espacio de estados del problema.

Se entiende como espacio de estados el conjunto de todos los estados generados tras la simulación del modelo. Dicho espacio de estados permite describir formalmente un problema como un conjunto de transformaciones, desde una situación dada hasta unas situaciones deseadas, a través de un conjunto de operaciones permitidas. El espacio de estados forma un grafo en el cual los nodos son estados y los arcos entre los nodos son acciones o eventos. Un camino en el espacio de estados será un conjunto de estados conectados por una secuencia de acciones.

Así, para resolver el modelo de decisión obtenido en la fase anterior, es necesario generar el espacio de estados del problema, lo cual se puede conseguir mediante la construcción del árbol de alcance de la RPC. Se entiende como árbol de alcance el conjunto de todos los marcados a los que se puede llegar partiendo del estado inicial del sistema, M_0. Se llamarán hijos de M_0 aquellos marcados consecuencia de habilitarse una

transición desde M_0, que a su vez serán padres de los siguientes marcados que se creen. Por tanto, se llamarán hojas aquellos marcados que no generen hijos (sin transiciones habilitadas).

Al finalizar esta fase, se tienen soluciones factibles representadas como un camino, formado por una secuencia de eventos, desde el estado inicial hasta un estado final que satisface todas las restricciones impuestas en el modelo de decisión.

3.5.6. Fase 6: Análisis de Resultados

En la fase anterior se puede generar el conjunto de todos los estados alcanzables del sistema, sin embargo, en algunos casos, dicho conjunto puede ser tan amplio que puede convertir el problema en un problema de optimización combinatoria NP-Completo, lo que quiere decir que la solución a este problema de optimización sería imposible de determinar en un tiempo de cálculo razonable debido al crecimiento exponencial del espacio de estados.

Durante esta fase de la metodología se deben llevar a cabo análisis que permitan decidir qué estados no son necesarios o útiles para la búsqueda de buenas soluciones al problema, ya sea modificando las heurísticas ya propuestas o aplicando nuevas, de tal forma que el espacio de estados quede reducido y se obtengan soluciones en un tiempo de cálculo razonable.

CAPÍTULO IV El Caso de Estudio

En este capítulo se aplicará paso a paso las fases de la metodología descrita en el capítulo anterior aplicando el formalismo matemático de RPC.

4.1 Fase 1: Descripción del Problema

El caso a estudiar en este proyecto es la gestión del tiempo de tránsito en un aeropuerto *hub*, en concreto el proceso comprendido entre el aterrizaje del avión de origen hasta el despegue del avión de destino. Durante dicho proceso el pasajero atravesará 2 puntos de control, en primer lugar el Control de Aduanas (o control de pasaportes) y posteriormente el Control de Seguridad, y se trasladará por las diferentes zonas del aeropuerto hasta llegar a la puerta de embarque de destino.

Para establecer unos tiempos de referencia durante todo el proceso a simular, se utilizará el ADRM *(Airport Development Reference Manual)*[1] como referencia. Se trata de un manual que establece unas recomendaciones para el diseño de aeropuertos basadas en la gran experiencia acumulada por la industria aeronáutica a lo largo de muchos años y proyectos. Este manual ha sido publicado por la IATA *(International Air Transport Asociation)*[1] y se edita periódicamente con el objetivo de adaptarse a las nuevas necesidades del sector. En este manual se definen aspectos económicos, de seguridad, de tiempos, de establecimiento de las distintas zonas, etc., que ayudan al correcto funcionamiento de un aeropuerto.

Con el objetivo de establecer unos parámetros cuantificables se define un "Nivel de Servicio", que puede entenderse como una serie de rangos (A-F, donde A es el nivel más alto y F el nivel más bajo) con los que valorar el funcionamiento y desarrollo de un aeropuerto. En este proyecto se usará Nivel de Servicio C, que es el mínimo exigido por la IATA, ya que determina un buen servicio por un coste razonable[4].

A la hora de determinar los objetivos del problema hay que fijarse en el estado actual del sistema en el mundo real y en función de esos resultados, establecer unos objetivos cuantificables. Actualmente la gestión de colas abarca muchos recursos en los

aeropuertos, debido a que intervienen una gran cantidad de elementos no predecibles, como por ejemplo las personas. El objetivo principal de este problema será disminuir el tiempo de transito de un pasajero en un aeropuerto y para ello, el concepto más influyente en el resultado es el tiempo de espera en colas, por lo que el objetivo particular será disminuir este tiempo de espera. Se puede observar en la *Figura 7* los tiempos de espera estipulados según *ADRM*[2].

Tabla 3: Tiempos de espera en cola para Nivel C según el *ADRM*

	De corto a aceptable	De aceptable a largo
Control de seguridad	0 a 3 min	3 a 7 min
Control de pasaportes	0 a 7 min	7 a 15 min

Tomando estos tiempos como referencia a la hora de valorar qué calidad debe alcanzar el sistema y teniendo en cuenta la dificultad de establecer unos tiempos de atención medios debido a todos los parámetros implicados (vuelo conflictivo, alarma en arco de seguridad, pérdida de pasaporte…), se ha decido establecer los siguientes tiempos, tomando como base los datos facilitados por el ADRM ayudados por la propia observación en vuelos:

- Tiempo de inspección de pasaportes: 30 segundos.
- Tiempo total en el control de seguridad: 20+5+12=37 segundos
    - Tiempo de preparación en el arco de seguridad: 20 segundos.
    - Tiempo de paso por el arco de seguridad: 5 segundos.
    - Tiempo de salida del arco de seguridad: 12 segundos.
- Tiempo total de paseo por la terminal: 120+300+240=660 segundos.
    - Tiempo desde la llegada hasta el control de aduanas: 120 segundos.
    - Tiempo desde el control de aduanas hasta el de seguridad: 300 segundos.
    - Tiempo desde el control de seguridad hasta el embarque: 240 segundos.

Con el fin de representar el problema como un SED, a continuación se muestran los diferentes componentes del problema a modelar.

4.1.1. Entidades:

- Aviones de origen: Se trata de los aviones que aterrizan en el aeropuerto *hub* y descargan los pasajeros. No todos los pasajeros que provienen de ese avión utilizarán el aeropuerto como escala, pero sí todos los que se han representado dentro de cada avión en el modelado. Es decir que un avión puede aterrizar con 300 pasajeros pero sólo 80 harán escala en ese aeropuerto, el resto estarán ya en su destino. Este elemento del sistema se define como una *Entidad Temporal*.
- Pasajeros en transito: Son todos aquellos pasajeros que vienen en cada avión y que utilizan el aeropuerto como enlace con otro avión hacia otro destino. También quedarán definidos como *Entidades Temporales*.
- Panel de Información de Vuelos: Contiene toda la información necesaria para que el pasajero se dirija hacia su nuevo destino, con puerta de embarque, hora de salida y código interno de identificación del avión. En base a esa información se le creará la tarjeta de embarque al pasajero, puesto que no la trae de origen. En este caso, el panel se define como un *Recurso o Entidad Permanente*.
- Controladores de Aduanas: Se trata de los empleados del aeropuerto que inspeccionan el pasaporte. Se asumirá que todos los pasajeros son inspeccionados, aunque en un sistema real puede pasar que a ciertas horas o con cierto tráfico de pasajeros la inspección no sea del 100%. Estos controladores se representarán como *Recursos*.
- Controladores de Seguridad: En este caso se trata de los empleados del aeropuerto que inspeccionan a los pasajeros en los arcos de seguridad con las cintas transportadoras de equipaje de mano. Al igual que los controladores de aduanas, éstos también se representan como *Recursos.*
- Colas de Espera: Se agrupa en este concepto todas las colas que los pasajeros tendrán que seguir durante su tránsito en el aeropuerto. Las dos colas que se van a gestionar en este caso son las referentes a Aduanas y Seguridad. Y se representan como *Recursos*.
- Puertas de embarque: Este elemento agrupa tanto a las puertas de llegada como a las de salida, sin estar diferenciadas como tal. Puesto que son elementos que no se transforman ni desaparecen, se considerarán como *Entidades Permanentes.*

4.1.2. Eventos y Actividades:

- Llegada de los pasajeros al aeropuerto: Se trata del primer evento que se debe modelar, no debe cumplir ninguna condición especial, simplemente que haya aviones pendientes de aterrizar.
- Asignación de la tarjeta de embarque: Se trata de una actividad un poco especial, ya que en un sistema real el pasajero ya trae la tarjeta de embarque desde el aeropuerto de origen, pero se decidió crear este evento "no real" con el fin de facilitar la obtención de estos datos en el modelo. Más adelante se especificarán las comprobaciones a realizar.
- Control de Aduanas: Se trata del primer evento influyente en el objetivo del problema. Consiste, como se ha definido anteriormente, en la inspección del pasaporte del pasajero. También más adelante se definirán sus condiciones.
- Control de Seguridad: El más conflictivo de los eventos a analizar, la gestión de colas en este caso es crucial para la minimización del tiempo de tránsito. Se trata del paso por los arcos de seguridad incluyendo descalzo, cacheo, repetición del proceso, casos que pueden penalizar mucho los tiempos medios de espera.
- Embarque de pasajeros: En este evento el pasajero ya ha llegado a su puerta de embarque de destino y se dispone a embarcar. El tiempo de tránsito acaba en el momento que el pasajero llega a la puerta de embarque, no se tiene en cuenta el tiempo de espera para embarcar, puesto que dicho tiempo no forma parte del tiempo de tránsito.

4.1.3. Atributos:

Dada la gran cantidad de atributos que existen en este sistema, se van a agrupar por concepto a representar y más adelante se profundizará en cada atributo en concreto.

- Identificadores: Se trata de un índice que se utilizará para diferenciar a cada entidad (tokens dentro de un mismo nodo lugar). Dicho identificador podría ser un DNI, un número de puerta, un número de vuelo, un código interno de empleado o simplemente un número natural secuencial.
- Estado del pasajero: Se define con el objetivo de conocer en todo momento en qué zona del aeropuerto se encuentra el pasajero y poder controlar su tiempo de tránsito en cada instante.

- Horas: Con este atributo se hace referencia a las horas de llegada y de salida de los aviones. El objetivo de su definición es poder comparar el tiempo de tránsito con el tiempo disponible entre la llegada al aeropuerto y la salida del mismo.
- Número de Pasajeros: Este es un atributo que se usa varias veces en distintos recursos y que puede representar el número de pasajeros que ha atendido un controlador, el número de pasajeros que hay en una cola esperando a ser atendidos, la cantidad de pasajeros de cada avión que harán escala, el número de pasajeros que ya han aterrizado en el aeropuerto o los que hay a la espera del embarque en la puerta correspondiente.
- Tiempos: El atributo más influyente en la obtención de una buena solución. Como en el caso anterior, este atributo tiene un valor y un significado diferente para cada recurso o entidad. Puede representar el tiempo de trabajo de los controladores, el tiempo de ocio de los mismos o, el más importante, el tiempo de tránsito de los pasajeros.

4.1.4. Precondiciones y Poscondiciones:

A continuación se describen las condiciones que deben cumplir las variables de estado para que los eventos puedan ocurrir.

Precondiciones a la llegada de aviones: Debe haber aviones pendientes de aterrizar y la información sobre el vuelo debe estar en el panel de información de ese aeropuerto.

Precondiciones al Control de Aduanas: El pasajero debe tener tarjeta de embarque asignada y encontrarse en la cola de aduanas esperando su turno. Debe haber algún controlador de aduanas libre para poder atenderle.

Precondiciones al Control de Seguridad: El pasajero debe haber finalizado el Control de Aduanas y encontrarse en la cola de seguridad esperando su turno. Debe haber algún controlador de seguridad libre para poder atenderle.

Precondiciones al Embarque del pasajero: El pasajero debe haber finalizado ambos controles y estar situado en la cola de la puerta de embarque correspondiente a su vuelo.

Poscondiciones a los Controles de Aduanas y Seguridad: Como el tiempo de servicio de los controles de aduanas y seguridad es distinto, en función del tipo de control que haya pasado el pasajero, el valor acumulado de tiempos variará.

4.2 Fase 2: Modelado del Sistema

Una vez definidos los elementos del sistema a modelar y realizado el análisis de los objetivos del problema a resolver, se va a modelar el sistema a simular. Se podrá observar la evolución del diseño desde la primera plasmación conceptual hasta la versión final, mucho más avanzada y mejorada. Para ello, se mostrarán las tres versiones más importantes del mismo, obviando modificaciones menores hechas con el único fin de corregir errores del modelo.

4.2.1. Transfer versión_1:

A continuación se muestran 3 tablas con la descripción de los colores, los nodos lugar y los nodos transición de la primera versión. Es necesario destacar que, aunque esta versión se llegó a representar gráficamente, nunca se obtuvieron resultados debido a errores de diseño que fueron corregidos en posteriores versiones. Pero sí que se considera interesante la exposición de la misma como primer diseño conceptual del sistema.

Tabla 4: Definición de Colores para la RPC versión_1

COLOR	DEFINICIÓN	DESCRIPCIÓN
Vuelo	int 0..1000	Identificador del vuelo.
p_emb	int 1..30	Número de puerta de embarque.
n_pas	int 0..1000	Número de pasajeros que viajan en un avión.
id_o	int 1..10	Identificador interno de los controladores.
est_p	int 1..3	Identificador de estado del pasajero.
busy_o	bool	Boolean para indicar el estado del controlador.
Avion	product vuelo*n_pas	Representa los aviones que deben aterrizar. Si n_pas es 0, significa que ya ha aterrizado.
Controlador	product id_o*busy_o	Representa los controladores del sistema. De 1 a 3 aduanas y de 4 a 10 seguridad.
Info	product vuelo*p_emb	Representa el panel de información de vuelos con número de vuelo y puerta de embarque.
Pasajero	product est_p*p_emb*vuelo	Representa a los pasajeros del aeropuerto y su estado.

Tabla 5: Nodos Lugar para la RPC versión_1

LUGAR	COLOR	DESCRIPCIÓN
Aviones	Avion	Representa aviones del sistema. En él se puede comprobar el identificador único del vuelo y el número de pasajeros de los que van a bordo que deben hacer escala en ese aeropuerto.

Info_vuelo	Info	Contiene la información necesaria para generar la tarjeta de embarque del pasajero. Incluye el número de vuelo de destino y puerta de embarque.
Pasajeros	Pasajero	Representan a los pasajeros que deben hacer escala, el estado a lo largo de toda su estancia en el aeropuerto y la información proporcionada por Info.
Control	Controlador	Representa a los controladores de aduanas y de seguridad. Contiene el identificador de tipo de controlador que sirve a su vez de identificador interno y su estado en cada momento.
Puertas	P_emb	Representa con un solo color el número de puerta de embarque del aeropuerto. No diferencia entre llegada y salida.

Tabla 6: Nodos Transición para la RPC versión_1

TRANS.	**DESCRIPCIÓN**
Lleg	Representa la llegada de aviones al aeropuerto.
Temb	Representa la asignación de tarjeta de embarque al pasajero.
Ini_Ctrl	Representa el inicio de la actividad de los Controladores.
Fin_Ctrl	Representa el fin de la actividad de los Controladores.
Emb	Representa el proceso de embarque en el avión de destino.

En cuanto a las expresiones de arco y funciones guarda, debido a la falta de espacio para mostrarlas en el gráfico de la RPC, se han etiquetado y se definen en la *Tabla 7* y *8*.

Tabla 7: Expresiones de Arco para la RPC versión_1

ID	Definición	Descripción
1	Av_Any → 1'(vuelo_a, n_pas)	La transición correspondiente hace la lectura de los datos del nodo lugar Aviones. Se utiliza tanto para *Lleg* como para *Emb*
2	Ll_Av → 1'(vuelo_a, n_pas-1)	Representa el desembarque de un pasajero, por eso, el contador de pasajeros del avión disminuye en 1.
3	Ll_Pa → 1'(0,0,0)	Al mismo tiempo que se representa el desembarque del pasajero, se crea un token nuevo en el nodo Pasajeros con los 3 colores inicializados a 0.
4	In_Any → 1'(vuelo_info, pta)	Esta expresión de arco se utilizará para representar la lectura y devolución de datos del nodo *Info_Vuelos* que representa el panel de vuelos de aeropuerto. Se utilizará en *Lleg* y en *TEmb*
5	Pa_Any → 1'(est_p, p_emb, vuelo)	Se envían los datos a las transiciones para que sean leídos y usados para las comprobaciones del guarda. Esta expresión de arco se usará para las transiciones *TEmb*, *Ini_Ctrl* y *Emb.*
6	TEmb_Pa → 1'(1, pta, vuelo_info)	Se devuelve un token al nodo *Pasajero* con el estado puesto a 1, ya tiene asignada la tarjeta de embarque con el número de

		vuelo y puerta para dirigirse hacia su avión de destino.
7	Ctr_Ini → 1'(id_op, false)	La transición *Ini_Ctrl* espera un token del nodo *Control* que tenga su estado en *false*, es decir que esté libre. En el caso de la transición *Fin_Ctrl*, devuelve un token con ese estado para almacenarlo en su nodo lugar.
8	Ini_Ctr → 1'(id_op, true)	En este caso es a la inversa, *Ini_Ctrl* devuelve un token con el estado a *true*, es decir, ocupado y *Fin_Ctrl* busca en el nodo lugar *Control*, un token con el estado en *true*.
9	Ctr_Pa → 1'(est_p+1, p_emb, vuelo)	Una vez finalizado el control, se devuelve el token que representa al pasajero utilizado a su nodo lugar con el estado aumentado en 1 para representar la actividad.
0	Ptas → 1'(id_pta)	Se envía un token del nodo *Puertas* para que sea leído desde la transición *Emb*. Una vez leído se utiliza la misma expresión para devolver el token en el mismo estado.

Tabla 8: Funciones Guarda para la RPC versión_1

Nombre	Definición	Descripción
Lleg	vuelo_a=vuelo & n_pas>0	Se compara que el identificador del avión a aterrizar aparezca en el panel de información del aeropuerto.
TEmb	est_p=0	Sólo es necesario comprobar que el pasajero todavía no tenga asignada la tarjeta de embarque.
Ini_Ctr	(est_p=1 & est_o=false & id_o<=ADN) | (est_p=2 & est_false=0 & id_o>ADN & id_o<=SEG)	Este guarda lleva a cabo dos actividades: a) Comprueba que el pasajero ya tenga asignada la tarjeta de embarque, que el controlador sea de aduanas y que esté libre. b) Comprueba que el pasajero haya finalizado el paso por aduanas, que el controlador sea de seguridad y que esté libre.
Fin_Ctr	(est_p=1 & est_o=true & id_o<=ADN) | (est_p=2 & est_true=0 & id_o>ADN & id_o<=SEG)	Este guarda lleva a cabo dos actividades: a) Comprueba que el pasajero ya tenga asignada la tarjeta de embarque, que el controlador sea de aduanas y esté ocupado. b) Comprueba que el pasajero haya finalizado el paso por aduanas, que el controlador sea de seguridad y esté ocupado.
Emb	Id_pta=p_emb & est_p=3 & vuelo=vuelo_a	En este caso se analiza que el pasajero ya haya finalizado ambos controles y que su número de vuelo y puerta coincida con el avión y su puerta de embarque.

Finalmente se representa gráficamente la RPC en función a los elementos definidos.

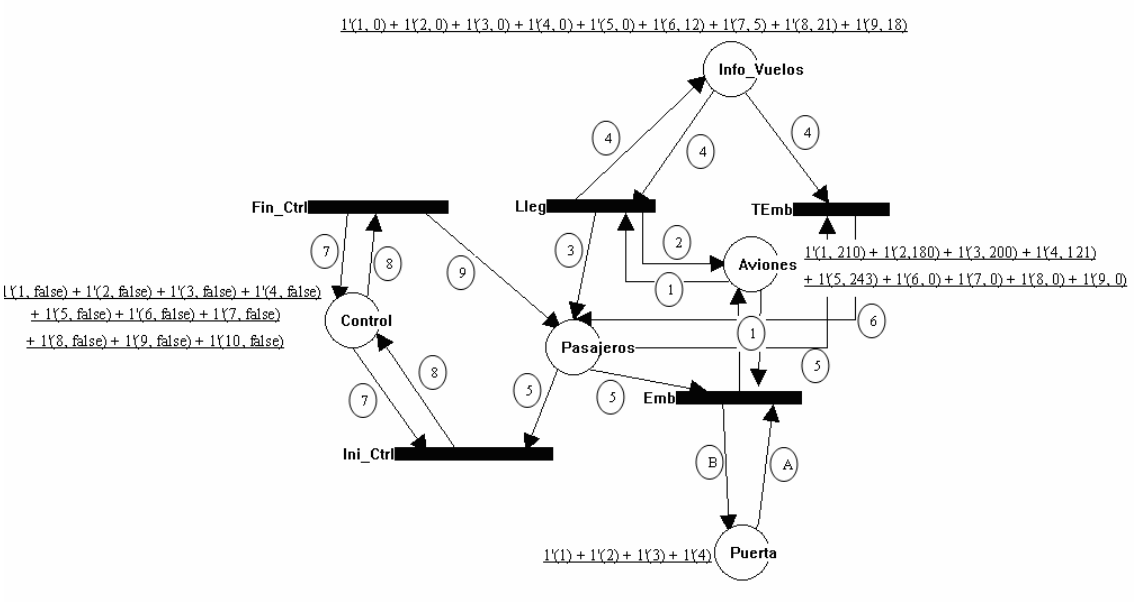


Figura 9: Representación Gráfica de la RPC versión_1

4.2.2. Transfer versión_2:

Una vez modelada y simulada la versión versión_1, se procedió a localizar los fallos y corregirlos en esta segunda versión, donde se han realizado muchos cambios de sintaxis, pero también algunos de concepto muy importantes, como simplificar la actividad de control en un solo evento, que engloba el inicio y el final de la misma. Además se añade un contador de pasajeros para llevar un mayor control de la ejecución y del estado del proceso en cada momento. Como apunte final hay que destacar que se añaden muchos más atributos para los tokens de cada Nodo Lugar con el objetivo de poder definir más estados y características de cada token y hacer el modelo mucho más completo, aprovechando al máximo esta característica de las RPC. De la *Tabla 9* a la *12,* se pueden observar todas estas mejoras, tales como los nuevos colores, nodos lugar y nodos transición modificados, expresiones de arco y funciones guarda.

En la *Tabla 9* se redefinen todos los colores para esta versión de la RPC, puesto que ha habido demasiados cambios de la primera a la segunda versión. En la *Tabla 10* se muestra sólo el nodo lugar que se ha creado nuevo respecto a la anterior versión. De la misma forma, en la *Tabla 11* se define el único nodo transición que ha cambiado respecto a la versión_1. Finalmente, en las *Tablas 12* y *13* se muestran todas las

expresiones de arco y funciones guarda aunque ya existieran en la versión anterior, puesto que se mantiene el concepto pero la especificación ha variado notablemente.

Tabla 9: Definición de Colores para la RPC versión_2

COLOR	DEFINICIÓN	DESCRIPCIÓN
CONT	int 0..1000	Se trata de una variable interna que especificará el número de pasajeros en cada instante.
CONT1	int 0..1000	Se trata de un contador que se utilizará para saber cuántos pasajeros ha atendido un controlador.
ESTPA	int 0..3	Estados del pasajero antes de embarcar (0-sin datos,1-tarjeta de embarque, 2-aduanas, 3-seguridad)
HOR1	int 0..23	Representa el componente hora de la hora de llegada del avión de origen al aeropuerto.
HOR2	int 0..23	Representa el componente hora de la hora de salida del avión en el panel de información.
HORAS1	int 0..23	Representa el componente hora de la hora de llegada del avión para el nodo lugar *Pasajeros*.
HORAS2	int 0..23	Representa el componente hora de la hora de salida del avión para el nodo lugar *Pasajeros*.
IDOP	int 1..100	Describe el código interno del controlador. Es un identificador global, no por tipo.
IDPA	int 1..1000	Describe la identificación del pasajero en el aeropuerto, podría utilizarse el DNI; por ejemplo.
MIN1	int 0..59	Representa el componente minuto de la hora de llegada del avión de origen al aeropuerto.
MIN2	int 0..59	Representa el componente minuto de la hora de salida del avión en el panel de información.
MINUTOS1	int 0..59	Representa el componente minuto de la hora de llegada del avión para el nodo lugar *Pasajeros*.
MINUTOS2	int 0..59	Representa el componente minuto de la hora de salida del avión para el nodo lugar *Pasajeros*.
NPAS1	int 0..1000	Define el número de pasajeros de cada avión que deberán hacer escala en el aeropuerto.
NPAS2	int 0..1000	Define el número de pasajeros que ya han subido al avión según el panel informativo.
PTAEMB	int 0..30	Representa el número de puerta de embarque asignada al pasajero en su tarjeta de embarque.
PTAEMB1	int. 1..30	Muestra el número de puerta de embarque de cada vuelo según el panel informativo.

PTAEMB2	int. 1..30	Describe el número de cada puerta de embarque del nodo *Puertas*.
TIPOOP	int 0..1	Describe al controlador (0 aduanas, 1 seguridad)
TOP	int 1000..2000	Define el tiempo de servicio de cada controlador.
TPAS	int 1000..2000	Hace referencia al tiempo de tránsito de cada pasajero, desde que aterriza hasta que embarca
VUELO	int 1..1000	Representa el número de vuelo de destino asignado al pasajero en su tarjeta de embarque.
VUELO1	int 1..1000	Es el código interno del vuelo del panel informativo.
VUELO2	int 1..1000	Representa el número de vuelo del avión que acaba de aterrizar.
AVION	product VUELO2*NPAS1*HOR1 *MIN1	Representa los diferentes aviones que van aterrizando en el aeropuerto
CONTROLADOR	product TOP*IDOP*TIPOOP	Representa a cada controlador del aeropuerto
INFO	product NPAS2*VUELO1*PTAEMB1*HOR2*MIN2	Representa la información de cada vuelo mostrada en el panel de información de vuelos.
PASAJERO	product IDPA*ESTPA*PTAEMB *VUELO* HORAS1*MINUTOS1* HORAS2*MINUTOS2	Representa al pasajero que desembarca y debe hacer escala en el aeropuerto, mostrando el tiempo total utilizado para dicho proceso.

Tabla 10: Nodos Lugar que cambian de versión_1 a versión_2

LUGAR	COLOR	DESCRIPCIÓN
CntPsj	IDPA	Se trata de un contador de pasajeros interno. Cada vez que aterriza un pasajero, se aumenta en 1 el valor del token que tiene dentro. De esta forma, se puede controlar en cada momento el número de pasajeros que hay en el aeropuerto.

Tabla 11: Nodos Transición que cambian de versión_1 a versión_2

TRANS.	DESCRIPCIÓN
Control	Representa la actividad de los Controladores, tanto de aduanas como de seguridad.

Tabla 12: Expresiones de Arco para la RPC versión_1.0

ID	Definición	Descripción
1*	AvAny → 1'(vueloav, npas, h1, m1)	La transición correspondiente hace la lectura de los datos del nodo lugar Aviones. Se utiliza tanto en para *Lleg* como para *Emb*
2*	LlAv → 1'(vueloav, npas-1,h1,m1)	Representa el desembarque de un pasajero, el avión pasa a tener un pasajero menos.
3	LlPa → 1'(0, contador+1, 0, vueloav, h1, m1, 0, 0, 1000)	Al mismo tiempo que se representa el desembarque del pasajero, se crea un token nuevo ya inicializado en el nodo *Pasajeros* y se le asigna su ID.
4*	InAny → 1'(numero, vueloinfo, pta, h2, m2)	Esta expresión de arco se utilizará para representar la lectura del panel de información del aeropuerto, *Info_Vuelos.*
5*	EnvPas → 1'(contador)	Esta expresión se va a utilizar para leer del contador la cantidad de pasajeros aterrizados antes del evento *Lleg* y así poder asignar un ID nuevo al siguiente que aterrice.
6*	DevPas → 1'(contador+1)	Registra la llegada de un nuevo pasajero.
7	PaAny → 1'(estpa, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas)	Se envían los datos de cada pasajero a las transiciones para que sean leídos y usados para las comprobaciones del guarda. Esta expresión de arco se usa en las transiciones *Temb*, *Control* y *Emb.*
8	TEmbPa → 1'(1, idpa, pta, vueloinfo, horas1, minutos1, h2, m2, tpas)	Se devuelve un token al nodo pasajero con el estado puesto a 1, significa que ya tiene asignados la tarjeta de embarque, el número de vuelo y puerta de embarque para dirigirse hacia su avión de partida.
9*	TEmbInfo → 1'(numero+1, vueloinfo, pta, h2, m2)	Se devuelve la consulta de información al panel pero con el valor de un color aumentado, este color controla el número de pasajeros que deberán embarcar en ese vuelo en concreto.
10	EnvCtr → 1'(top, idop, tipoop)	La transición *Control* hace una lectura de los tokens del nodo *Controlador.* A partir de esta versión ya hay un identificador de tipo (0 aduanas y 1 seguridad). Además se ha eliminado el estado del controlador, puesto que tanto el inicio como el fin del control se efectúa en una sola transición.
11	DevCtr → if(tipoop=0 & top>=tpas) then 1'(top+15, idop, tipoop) else if(tipoop=0 & top<tpas) then	A partir de esta versión se controlan los tiempos de trabajo, por lo que al devolver el token a su nodo lugar *Controlador*, debe compararse el tiempo del

	1'(tpas+15, idop, tipoop) else if(tipoop=1 & top>=tpas) then 1'(top+12, idop, tipoop) else 1'(tpas+12,idop, tipoop)	pasajero y el del controlador para asignar el mayor de los dos, puesto que así se actualiza el reloj en el tiempo de cada token.
12	CtrPa → if(tipoop=0 & top>=tpas) then 1'(estpa+1, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, top+15) else if(tipoop=0 & top<tpas) then 1'(estpa+1, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas+15) else if(tipoop=1 & top>=tpas) then 1'(estpa+1, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, top+12) else 1'(estpa+1, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas+12)	En esta expresión, al igual que en la anterior, se debe comprobar antes de devolver el token a su nodo lugar, que el tiempo asignado es el correspondiente en función del tiempo del pasajero y el del controlador además de diferenciarlo por tipo de controlador.
13	RestPas → 1'(contador-1)	Devuelve al nodo lugar *CntPsj* el token contador con un pasajero menos, puesto que ha embarcado y es necesario restarlo para controlar el número de pasajeros en el aeropuerto en todo momento.
14*	Ptas → 1'(idpta)	Se envía un token del nodo *Puertas* para que sea leído desde la transición *Emb*. Una vez leído, se utiliza la misma expresión para devolver el token en el mismo estado.

Tabla 13: Funciones Guarda para la RPC versión_2

Nombre	Definición	Descripción
Lleg	(vueloav=vueloinfo) & (npas>0)	Se compara que el identificador del avión que llega aparezca en el panel informativo y que el avión tenga pasajeros por desembarcar todavía.
Temb	(estpa=0) & (h2>horas1)	Comprueba que el token escogido no tenga ya tarjeta de embarque y que la hora de salida de la tarjeta asignada sea mayor que su hora de llegada.
Control	(estpa=1 & estop=0 & idop<=3) |(estpa=2 & estop=1 & idop<=3) |(estpa=3 & estop=0 & idop>3 & idop<=10) |(estpa=4 & estop=1 & idop>3 & idop<=10)	Este guarda lleva a cabo dos actividades: a) Comprueba que el pasajero tenga tarjeta de embarque y que exista un controlador libre de aduanas. Además, controla que finalice correctamente esta actividad cambiando posteriormente el estado del

		pasajero y dejando libre al controlador. b) Comprueba que el pasajero haya pasado aduanas y que exista un controlador libre de seguridad. Además, controla que finalice correctamente esta actividad cambiando posteriormente el estado del pasajero y dejando libre al controlador.
Emb	(idpta=ptaemb) & (estpa=5) & (vuelo=vueloav)	Finalmente, se analiza que el pasajero haya pasado los controles y que su número de vuelo y puerta coincidan con el avión y su puerta de embarque.

Tras especificar esta simplificación de funciones guarda y complicación de alguna expresión de arco, se observa en la *Figura 10* cómo queda el diseño de la nueva RPC.

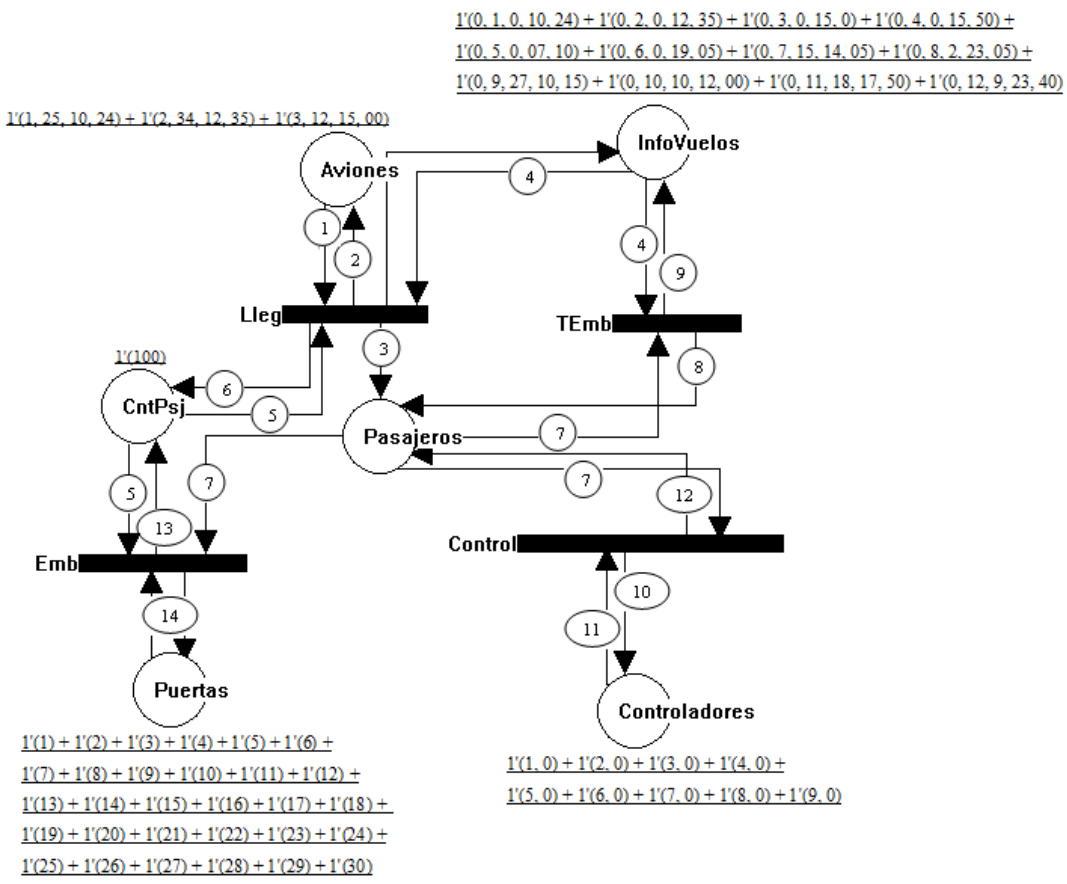


Figura 10: Representación Gráfica de la RPC versión_2

4.2.4. Transfer versión_3 (final):

Después de dos versiones de las cuales sólo con la segunda se obtuvieron resultados coherentes y cercanos a lo buscado, se ha diseñado la especificación final del modelado de la RPC que representa el sistema objeto del caso de estudio. Con la versión anterior se realizaron pruebas para observar qué resultados se obtenían, y aunque no eran malos, no eran los deseados, puesto que no se acercaban demasiado a la realidad y los tiempos de simulación eran demasiado elevados.

Por esa razón, y tras analizarlo exhaustivamente, se llegó a una conclusión que ayudaría mucho a minimizar el tiempo de obtención de resultados: dividir el proceso de simulación en dos submodelos. De esta manera, el primer submodelo obtiene los datos de entrada (*input*) necesarios para simular el segundo submodelo, donde realmente se manipulan los tiempos de espera, que es lo que interesa reducir y analizar.

Tabla 14: Definición de Colores para la RPC versión_3

COLOR	DEFINICIÓN	DESCRIPCIÓN
CONT	int 0..1000	Se trata de una variable interna que marcará el número de pasajeros en cada instante.
ESTPA	int 0..3	Representa los estados por los que debe pasar el pasajero antes de embarcar (0 sin datos, 1 con tarjeta de embarque, 2 control de aduanas finalizado, 3 control de seguridad finalizado).
HOR1	int 0..23	Representa el componente hora de la hora de llegada del avión de origen al aeropuerto.
HOR2	int 0..23	Representa el componente hora de la hora de salida del avión en el panel de información.
HORAS1	int 0..23	Representa el componente hora de la hora de llegada del avión para el nodo lugar *Pasajeros*.
HORAS2	int 0..23	Representa el componente hora de la hora de salida del avión para el nodo lugar *Pasajeros*.
IDCOLA	int 1..1000	Es la posición del pasajero en cada una de las colas.
IDPA	int 1..1000	Describe la identificación del pasajero en el aeropuerto, podría utilizarse el DNI; por ejemplo.
MIN1	int 0..59	Representa el componente minuto de la hora de llegada del avión de origen al aeropuerto.
MIN2	int 0..59	Representa el componente minuto de la hora de salida del avión en el panel de información.

MINUTOS1	int 0..59	Representa el componente minuto de la hora de llegada del avión para el nodo lugar *Pasajeros*.
MINUTOS2	int 0..59	Representa el componente minuto de la hora de salida del avión para el nodo lugar *Pasajeros*.
NPAS1	int 0..1000	Define el número de pasajeros de cada avión que deberán hacer escala en el aeropuerto.
NPAS2	int 0..1000	Define el número de pasajeros que ya han subido al avión según el panel informativo.
PTAEMB	int 1..30	Representa el número de puerta de embarque asignada al pasajero en su tarjeta de embarque.
PTAEMB1	int 1..30	Muestra el número de puerta de embarque de cada vuelo según el panel informativo.
TACT	int 0..10000	Representa el *clock* del sistema, indicando en cada instante el tiempo del mismo.
TPAS	int 0..2000	Hace referencia al tiempo de tránsito de cada pasajero, desde que aterriza hasta que embarca.
TTOTAL	int 0..10000	Representa el tiempo final de la simulación. Se actualiza con el valor más alto disponible.
VUELO	int 1..1000	Representa el número de vuelo de destino asignado al pasajero en su tarjeta de embarque.
VUELO1	int 1..1000	El código de cada vuelo según el panel informativo.
VUELO2	int 1..1000	Representa el vuelo del avión que acaba de aterrizar.
CONT1	product CONT*TACT*TTOTAL	Además de usarse como contador, se almacenan el tiempo y el *clock* del sistema, de esta manera se puede controlar cuánto tiempo simulado lleva ejecutado y en caso de haber solución, se puede observar el tiempo total simulado.
AVION	product VUELO2*NPAS1*HOR1*MIN1	Representa los diferentes aviones que van aterrizando en el aeropuerto
PASAJERO	product ESTPA*IDCOLA*PTAEMB*VUELO*HORAS1*MINUTOS1*HORAS2*MINUTOS2*TPAS*IDPA	Representa al pasajero que desembarca y debe hacer escala en el aeropuerto, mostrando el tiempo total utilizado para dicho proceso.
INFO	product NPAS2*VUELO1*PTAEMB1*HOR2*MIN2	Representa el panel de información del aeropuerto.
COLA1	int 1..10000	Representa la cola generada en el control de aduanas.

COLA2	int 1..10000	Representa la cola generada en el control de seguridad.
COLA3	int 1..10000	Representa la cola generada en la sala de embarque.
ATE1	int 1..10000	Representa el último pasajero atendido en la cola de aduanas.
ATE2	int 1..10000	Representa el último pasajero atendido en la cola de seguridad.
ATE3	int 1..10000	Representa el último pasajero embarcado en la sala de embarque.
IDOP	int 1..100	Describe el código interno del controlador. Es un identificador global, no por tipo.
TIPOOP	int 0..1	Define el tipo de controlador de cada token del nodo *Controladores* (0 aduanas, 1 seguridad)
TOP	int 0..1000	Define la posición del pasajero en cada una de las colas por las que pasa.
TOCIO	int 0..1000	Describe el acumulado del tiempo en el que el controlador ha estado desocupado.
NPAS3	Int 0..1000	Identifica el número de pasajeros atendidos por el controlador al que pertenece este atributo.
PTAEMB2	int. 1..30	Describe el número de cada puerta de embarque del nodo *Puertas*.
CONTROLADOR	Product TOP*TOCIO*IDOP*TIPOOP*NPAS3	Representa a cada controlador del aeropuerto, definiendo sus tiempos de trabajo y descanso, junto al número de pasajeros atendidos.
COLAS	product COLA1*ATE1* COLA2*ATE2* COLA3*ATE3	Este color representa todas las colas del sistema junto al pasajero atendido en cada momento.

Debido a que se han modelado dos RPCs, existen cambios sustanciales en transiciones y lugares, por lo que los estados iniciales y finales son muy diferentes. Y para una mejor comprensión durante toda esta versión, se tratarán las dos RPC como submodelos independientes, generando sus especificaciones en tablas por separado.

Se puede observar en la *Tabla 16*, cómo desaparece la transición *TEmb* y se integra junto con la llegada de aviones, *Lleg*, con el objetivo de sintetizar más el código.

Además, otra heurística muy importante que ayudó a minimizar tiempos de simulación fue la jerarquización de las Funciones Guarda. De esta manera se limitaban los tokens a

comparar a medida que se iba avanzando en el guarda, evitando comparaciones de 4 o 5 tokens a la vez, hecho que enlentecía mucho el proceso de simulación.

En las tablas que hacen referencia a las Funciones Guarda y a las Expresiones de Arco se utiliza el asterisco como marca diferenciadora de las funciones y expresiones que no se modifican respecto a la anterior versión.

Tabla 15: Nodos Lugar del primer submodelo de la RPC versión_3

LUGAR	COLOR	DESCRIPCIÓN
Pasajeros	PASAJERO	En este nodo se almacenan los pasajeros que van aterrizando en el aeropuerto, pero en este caso ya incluyen la tarjeta de embarque asignada.
Aviones	AVION	Este Nodo Lugar no tiene ninguna modificación, sigue mostrando código interno, número de pasajeros en tránsito y hora de llegada.
InfoVuelos	INFO	Tampoco sufren cambios sus colores, aunque sí sus valores, como el número de pasajeros que deben embarcar, que ahora empieza en 1000.
CntPsj	CONT1	Se trata de un contador de pasajeros interno. Cada vez que aterriza un pasajero, se aumenta en 1 el valor del token que tiene dentro. Además, lleva la cuenta del tiempo total y el *clock* del sistema.

Tabla 16: Nodos Transición del primer submodelo de la RPC versión_3

TRANS.	DESCRIPCIÓN
Lleg	Muestra la llegada de los aviones al aeropuerto e inmediatamente le asigna a cada pasajero una tarjeta de embarque con los datos de destino.

Tabla 17: Expresiones de Arco para el primer submodelo de la RPC versión_3

ID	Definición	Descripción
1*	AvAny → 1'(vueloav, npas, h1, m1)	Esta expresión sirve para hacer la lectura de datos del nodo lugar *Aviones*.
2*	InAny → 1'(numero, vueloinfo, pta, h2, m2)	Esta expresión de arco se utiliza para enviar el panel de *Info_Vuelos.*
3	EnvPas → 1'(contador,tact,ttotal)	Expresa la lectura del contador de pasajeros, tiempo actual y tiempo total.
4	LlPa → 1'(1, contador+1, pta, vueloinfo, h1, m1, h2, m2, 1120, contador+1)	Representa el desembarque, se inicializa un token nuevo en el nodo *Pasajeros* y se le asigna una tarjeta de embarque e identificador que no variarán, además de su posición en la cola y su primer tiempo de paseo (120seg.).
5	DevPas → 1'(contador+1,tact,ttotal)	Se utiliza para guardar el actual número de

		pasajeros aterrizados, que ha aumentado en una unidad.
6*	LlAv → 1'(vueloav, npas-1,h1,m1)	Representa el desembarque de un pasajero, el avión tiene uno menos.
7	LlInfo → 1'(numero+1, vueloinfo, pta, h2, m2)	Equivale a la anterior *TembInfo*, representa el aumento de pasajeros que deberán embarcar en un vuelo.

Tabla 18: Funciones Guarda para el primer submodelo de la RPC versión_3

Nombre	Definición	Descripción
Lleg*	(npas:NPAS1): bool=(npas>0) ((h1*60+m1+45)<(h2*60+m2))	Se comprueba que queden pasajeros por aterrizar en algún avión. Y que entre la hora de llegada y la hora de salida haya al menos 45 minutos para hacer el tránsito correctamente.

El archivo de salida de este primer submodelo de la RPC contiene la lista de todos los pasajeros creados ya con la tarjeta de embarque asignada. Esto se hace para reducir el tiempo de simulación, puesto que al ejecutar la aplicación de simulación, se perdía muchísimo tiempo combinando pasajeros con tarjetas de embarque. Como en este caso de estudio se trata de un proceso aleatorio que no afecta la solución, se coge la primera solución que da la simulación sin importar las distintas combinaciones.

Una vez obtenido este *output*, se utilizará de *input* en el segundo submodelo, para ello se van a describir los Nodos Lugar y Transición, las Expresiones de Arco y las Funciones Guarda de este segundo submodelo de la RPC. Al igual que en casos anteriores, sólo se definirán aquellos nodos que varíen o sean nuevos respecto al primer submodelo. En el caso de las Expresiones de Arco se expondrán por mera información y distinguidas con un asterisco aquellas que no varíen de anteriores versiones. Cabe señalar que en el caso del Nodo Lugar *Pasajeros*, en el estado inicial del primer submodelo estaba vacío, en cambio ahora contiene un número determinado de tokens que representan el número de pasajeros aterrizados en el aeropuerto, cada uno con su tarjeta de embarque y resto de información, correspondiente al estado final del mismo nodo en el primer submodelo.

Tabla 19: Nodos Lugar del segundo submodelo de la RPC versión_3

LUGAR	COLOR	DESCRIPCIÓN
Pasajeros	PASAJERO	A través de este nodo se representa a todos los pasajeros que hay en el aeropuerto en cada instante.
Controladores	CONTROLADOR	Este nodo representa los controladores de aduanas y seguridad.
Colas	COLAS	A través de este nodo se gestionan todas las colas del sistema. El primer valor, que corresponde al número de pasajeros que ya tienen tarjeta de embarque, está inicializado directamente con la forma 1X, donde X es ese número de pasajeros (*output* del primer submodelo.
Puertas	PTAEMB2	Este nodo representa las puertas de embarque del aeropuerto con un único color que define el número de puerta.
CntPsj	CONT1	Se trata de un contador de pasajeros interno. Cada vez que aterriza un pasajero, se aumenta en 1 el valor de su token. Además acumula el tiempo total y el *clock* del sistema, que se actualiza con cada evento activado. En esta segunda parte se inicializa directamente con el valor del número de pasajeros aterrizados para ir restando cada vez que embarca uno.

Tabla 20: Nodos Transición del segundo submodelo de la RPC versión_3

TRANS.	DESCRIPCIÓN
Control	Esta transición gestiona la actividad de los dos controles, el de Aduanas y el de Seguridad.
Emb	A través de esta transición se representa el embarque de los pasajeros en su avión de destino.

Tabla 21: Expresiones de Arco para el segundo submodelo de la RPC versión_3

ID	Definición	Descripción
8*	EnvPas → 1' (contador,tact,ttotal)	Marca la cantidad de pasajeros en el aeropuerto.
9	RestPas → if (tpas>tact) then 1'(contador-1, tpas-10000, ttotal+tpas-10000) else 1'(contador-1, tact-10000, ttotal+tpas-1000)	Representar el embarque de pasajeros en el nodo *CntPsj*.
10*	PaAny → 1'(estpa, idpa, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas)	Se envían los datos de cada pasajero a las transiciones para que sean leídos y usados para las comprobaciones del guarda.
11	ReadColas→ 1'(cola1, ate1, cola2, ate2, cola3, ate3)	Se devuelve un token al nodo pasajero con el estado puesto a 1, significa que ya tiene asignados la tarjeta de embarque, el número de vuelo y puerta de embarque para dirigirse hacia su avión de partida.

12*	WriteAte1y2 → if(tipoop=0) then 1'(cola1, idcola, cola2+1, ate2, cola3, ate3) else 1'(cola1, ate1, cola2, idcola, cola3+1, ate3)	Representa el avance de las colas y el cambio de pasajero atendido. En función del tipo de controlador que se haya usado, moverá una cola u otra.
13*	EnvCtr → 1'(top, tocio, idop, tipoop, pasajeros)	Esta expresión permite leer el tipo de controlador, el tiempo de trabajo, el de ocio y el número de pasajeros atendidos.
14	DevCtr → if(tipoop=0 & top>=tpas) then 1'(top+30, tocio, idop, tipoop, pasajeros+1) else if(tipoop=0 & top<tpas) then 1'(tpas+30, tocio+tpas-top, idop, tipoop, pasajeros+1) else if(tipoop=1 & top>=tpas) then 1'(top+37, tocio, idop, tipoop, pasajeros+1) else 1'(tpas+37, tocio+tpas-top, idop, tipoop, pasajeros+1)	En esta versión ya se aplican los tiempos conocidos según el ADRM, 30seg para inspección de aduanas y 37seg para control de seguridad. Además, al devolver el token a su nodo lugar *Controlador*, deben compararse el tiempo del pasajero y el del controlador para asignar el más alto, tanto al pasajero como al controlador.
15	CtrPa → if(tipoop=0 & top>=tpas) then 1'(2, cola2+1, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, top+30, idpa) else if(tipoop=0 & top<tpas) then 1'(2, cola2+1, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas+30, idpa) else if(tipoop=1 & top>=tpas) then 1'(3, cola3+1, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, top+37+120+240+300, idpa) else 1'(3, cola3+1, ptaemb, vuelo, horas1, minutos1, horas2, minutos2, tpas+37+120+240+300, idpa)	Al igual que en la expresión anterior, aquí ya se aplican los tiempos de referencia del ADRM, pero a diferencia del arco anterior, aquí se deben añadir los tiempos de paseo del pasajero, 240seg para llegar desde aduanas a seguridad y 300seg para ir del control de seguridad a la puerta de embarque.
16	WriteAte3 → 1'(cola1, ate1, cola2, ate2, cola3, idcola)	Devuelve al nodo lugar *CntPsj* el token con un pasajero menos, puesto que ya ha embarcado y es necesario restarlo para controlar el número de pasajeros en todo momento.
17*	Ptas → 1'(idpta)	Se utiliza la misma expresión para enviar datos del nodo *Puertas* a la transición *Emb*, y recogerlos de la misma transición para enviarlos al mismo nodo.

Tabla 22: Funciones Guarda para el segundo submodelo de la RPC versión_3

Nombre	Definición	Descripción
Control	(tipoop=1 & pasajeros<45 & tocio<10560) | (tipoop=0 & pasajeros<45 & tocio<10560) (estpa=2 & tpas<=14500) | (estpa=1) (estpa=2 & idcola=(ate2+1)) | (estpa=1 & idcola=(ate1+1)) (estpa=2 & tipoop=1) | (estpa=1 & tipoop=0)	Esta función guarda se jerarquiza en 4 niveles: -El tiempo libre del controlador no debe exceder de un tiempo marcado para considerarlo como buena solución. También debe haber un balance de carga de trabajo por controlador. -Se comprueba que el pasajero no lleve demasiado tiempo en el aeropuerto o se descarta. -Se busca al siguiente pasajero de cada cola. -Se comprueba que el pasajero sea atendido por el controlador correcto.
Emb	(ate2=2130) ((estpa=3) & (tpas<=16000)) (idcola=(ate3+1)) (idpta=ptaemb)	Esta función guarda también está jerarquizada en 4 niveles: -Se comprueba que estén todos los pasajeros listos para embarcar. -Que el tiempo de tránsito final no sea elevado. -Que los pasajeros embarcan en orden de llegada a la cola. -Como última comprobación, se asegura que el pasajero embarca en la puerta correcta.

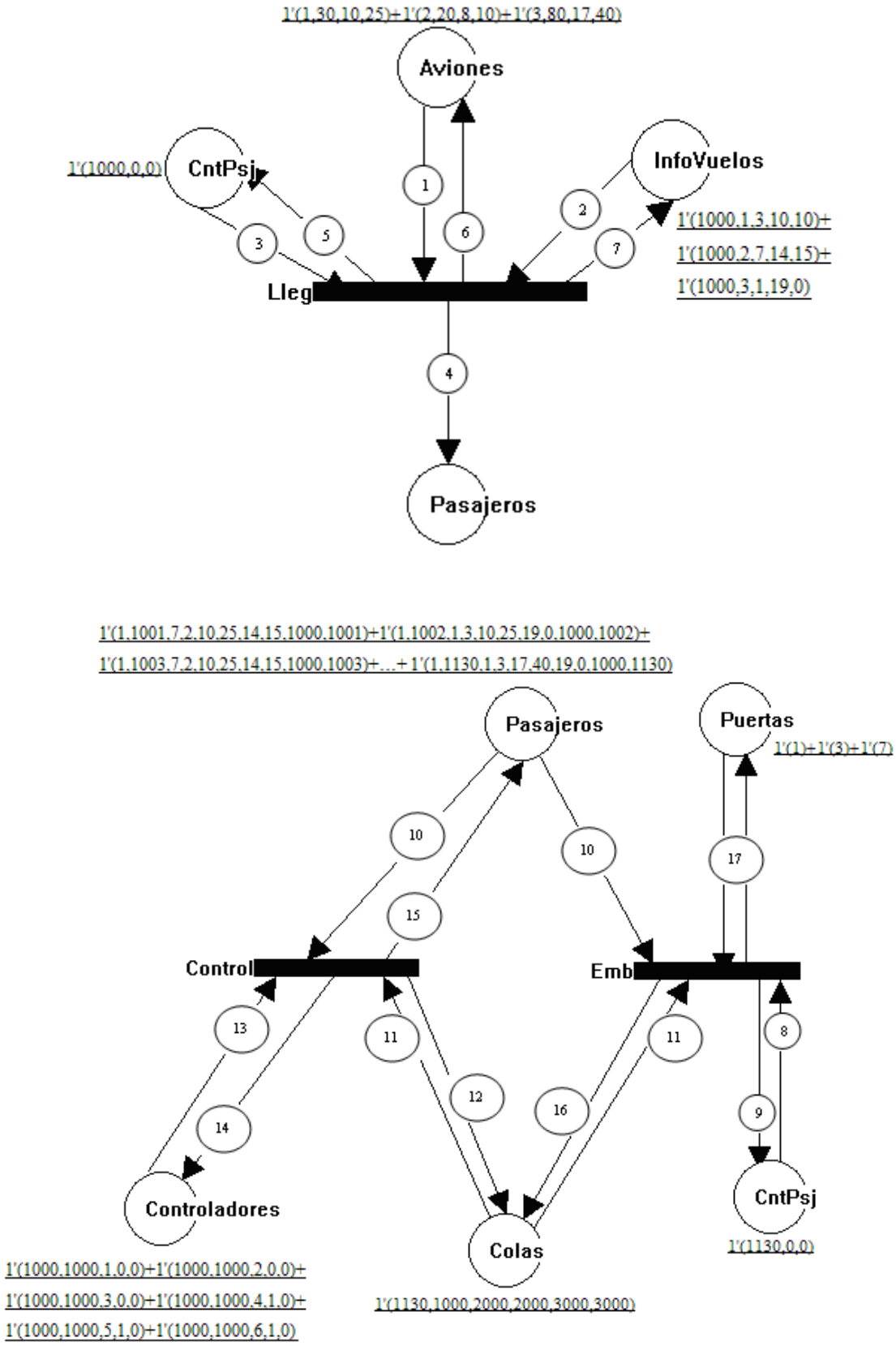


Figura 11: Representación Gráfica en dos submodelos de la RPC versión_3

4.3 Fase 3: Representación del Conocimiento

En esta fase debe especificarse todo el conocimiento posible con el objetivo de evitar que los eventos ocurran bajo determinadas condiciones que se sabe de antemano que no conducirán a una buena solución. Este conocimiento se usa como heurística y se puede adquirir con el estudio del entorno real. Estas heurísticas se implementarán en el diseño de la RPC mediante el uso de guardas.

En el caso del sistema en estudio, se puede determinar como conocimiento el tiempo total que determina el ADRM como aceptable para poder hacer un tránsito en un aeropuerto[4]. Sin tener en cuenta los tiempos de espera y sumando todos los tiempos de actividad de los diferentes eventos: el tiempo de actividad medio de un pasajero durante un tránsito se marca en 67 segundos (30 para el paso por aduanas y 37 para el paso por seguridad), además se le debe sumar el tiempo de paseo, que determina lo que tarda el pasajero en ir de la puerta de llegada al control de aduanas, del control de aduanas al control de seguridad y del control de seguridad a la puerta de embarque; dicho tiempo se ha marcado en 660 segundos (11 minutos). Está claro que estos tiempos varían mucho en función del aeropuerto y de cada pasajero, por lo que es muy complicado establecer una media, así que este tiempo se ha cogido tomando como fuente las recomendaciones del ADRM. Por lo tanto el tiempo total de tránsito, sin esperas, de un pasajero en un aeropuerto medio se ha establecido en 727 segundos.

A partir de estos datos, cualquier solución que incluya un tiempo superior al especificado en cada fase más un lógico retardo por una espera admisible es descartada como solución buena. Esta heurística queda definida en cada guarda de la segunda parte de la RPC de manera que se pueda supervisar en todo momento si la solución que dará será buena o no.

Otra heurística que se ha aplicado afecta al uso de los recursos, ya que se ha determinado que todos los controladores deben ser usados proporcionalmente al número de pasajeros, con esto se intenta conseguir un balanceo de carga de los recursos de manera que ninguno cause baja laboral por exceso de carga, lo cual podría ser mucho más perjudicial para la gestión del control. Para ello, la condición del guarda de la Transición *Control* especifica que en función del número de pasajeros, cada controlador

deberá atender como máximo un número determinado de los mismos, por ejemplo, en el caso de que haya 130 pasajeros, 3 controladores de aduanas y 2 controladores de seguridad, cada controlador de aduanas deberá atender como máximo 45 pasajeros y cada controlador de seguridad 67 pasajeros. Como se puede observar, se deja un margen de pasajeros atendidos para cada controlador de manera que no se limite en exceso el número de soluciones y así se puede reducir el riesgo de perder buenas soluciones. En un sistema real cada controlador tiene su propio tiempo de actividad, por lo que puede atender a más o menos pasajeros en el mismo periodo de tiempo. Para reducir esa variable, se ha tomado un tiempo medio de servicio igual para todos los controladores.

4.4 Fase 4: Construcción del Modelo de Decisión

4.4.1. Especificación del estado inicial

En este caso de estudio, el modelo final de la RPC se ha dividido en dos submodelos. Por ello, consta de dos estados iniciales y dos estados finales, quedando claro que el estado final del primer submodelo forma parte del estado inicial del segundo submodelo en los Nodos Lugar que tienen en común ambas partes. Por lo que el estado inicial del primer submodelo es el estado incial del sistema modelado y el estado final del segundo submodelo es el estado final del sistema modelado. Aun así, se definirán ambos para detallar mejor el funcionamiento de la RPC.

$$M_0 = \begin{Bmatrix} \left[1'(1,30,10,25)+1'(2,20,8,10)+1'(3,80,17,40)\right], \\ \left[1'(1000,1,3,10,10)+1'(1000,2,7,14,15)+1'(1000,3,1,19,0)\right], \\ \left[1'(1000,0,0)\right] \end{Bmatrix}$$

Figura 12: Estado Inicial del primer submodelo de la RPC versión_3

$$M_0 = \begin{Bmatrix} \left[1'(1,1001,7,2,10,25,14,15,1000,1001)+\ldots+1'(1,1130,1,3,17,40,19,0,1000,1130)\right], \\ \left[1'(1000,1000,1,0,0)+\ldots+1'(1000,1000,6,1,0)\right], \\ \left[1'(1130,0,0)\right], \\ \left[1'(1130,1000,2000,2000,3000,3000)\right], \\ \left[1'(1)+1'(3)+1'(7)\right] \end{Bmatrix}$$

Figura 13: Estado Inicial del segundo submodelo de la RPC versión_3

4.4.2. Especificación del Objetivo de Optimización

El objetivo de optimización viene marcado por los estados finales para cada submodelo de la RPC que se ha diseñado. Estos estados finales representan estados a los que se desea llegar para encontrar una solución. Posteriormente, mediante la función de costo y el análisis de resultados, ya se determinará cuáles de entre las soluciones son las mejores soluciones.

$$M_f = \left\{ \left[1'(1,0,10,25) + 1'(2,0,8,10) + 1'(3,0,17,40) \right], \left[*'(**) \right], \left[1'(1130,*,*) \right] \right\}$$

Figura 14: Estado final del primer submodelo de la RPC versión_3

$$M_f = \left\{ \left[*'(**) \right], \left[1'(1000,*,*,) \right], \left[1'(*,*,*,*,*,3130) \right], \left[*'(**) \right] \right\}$$

Figura 15: Estado final del segundo submodelo de la RPC versión_3

En el caso del primer submodelo de la RPC, lo que marca que se ha conseguido una solución es que los aviones estén vacíos y que todos los pasajeros estén ya situados en la primera cola, la que hace referencia al control de aduanas. En el segundo submodelo la solución se consigue cuando todos los pasajeros han llegado a la sala de embarque y ya han embarcado, quedando el aeropuerto sin más pasajeros en tránsito.

4.4.3. Especificación de una Función de Costo

La función de costo pondera las variables de decisión de manera que aquellos estados que no convienen tengan un costo elevado. Para ello es necesario decidir cuál de los nodos lugar de la RPC contiene las variables de decisión más críticas para este modelo. En este caso, lo que se está buscando optimizar es el tiempo de tránsito de un pasajero, representado en la RPC como el color *TPAS* de los tokens del nodo lugar *Pasajeros*, por esa razón la función de costo se le asignará a dicho nodo lugar de manera que penalice valores de *TPAS* altos.

$$FCP(tpas : TPAS) : real = a * tpas'(*,*,*,*,*,*,*,*,*,*)$$

Figura 16: Función de Costo diseñada

donde *a* es una constante con valor 1, definida al inicio de la simulación.

En el caso que ocupa a este proyecto, la función de costo no se utiliza para obtener una solución de costo menor, si no como heurística para acotar el espacio de estados, de manera que cada vez que la función determina un costo elevado para un camino del espacio de estados, deja de generar más marcados en ese camino y busca otro, de esta manera el espacio de estados se reduce, consiguiendo menores tiempos de simulación, objetivo principal de esta función de costo. Hay que aclarar que, aunque se aplicó en la segunda versión del modelo, no se aplicó en la versión final por no generar mejoras sustanciales en el tiempo de simulación ni en las soluciones.

4.5 Fase 5: Simulación del Modelo de Decisión

En esta fase ya se procede a ejecutar el modelo de decisión y a tomar datos con las soluciones al problema de búsqueda generado. Para ello se ha utilizado una interfaz diseñada con el objetivo de convertir la representación gráfica del modelo de decisión en código fuente, utilizando un lenguaje propio y por lo tanto optimizado para estas funciones. De esta manera se evita tener que teclear directamente todo el código, con la simple especificación del modelo gráfico, la interfaz determina el código que luego se simulará utilizando el intérprete diseñado para tal efecto.

En la *Tabla 23* de la sección Fase 6, Análisis de Resultados, se pueden observar las principales variables de decisión (tiempos de pasajeros y controladores) y determina su valor en función de las variables de entrada (número de controladores y de pasajeros). Se toma como base para la simulación la llegada al aeropuerto de tres aviones que contienen 30, 20 y 80 pasajeros en tránsito respectivamente, lo que hace un total de 130 pasajeros en tránsito en el aeropuerto.

A partir de aquí, se irá modificando el número de controladores del aeropuerto con el objetivo de decidir la mejor configuración de controladores de aduanas y de seguridad para dicho aeropuerto. Una vez finalizada la prueba con 130 pasajeros, se aumentarán hasta llegar a los 1000 pasajeros pasando también por los 400 pasajeros y se observará la variación del tiempo de tránsito de los pasajeros. Una vez realizadas todas estas pruebas, ya se podrá tomar una decisión sobre la mejor configuración de controladores para obtener una buena relación coste_recursos/tiempo_tránsito. Para ello, se tendrá en consideración el tiempo medio y tiempo máximo de tránsito obtenido para un pasajero

con el objetivo de compararlo con el tiempo medio máximo permitido para el nivel de calidad escogido para las pruebas en este teórico aeropuerto, que según los estudios realizados en varios aeropuertos[2 y 4] no debe exceder de 45-50 minutos. Paralelamente, también se recogerán un valor de referencia de los tiempos de los controladores y el tiempo total desde que desembarca el primer pasajero hasta que embarca el último en un espacio de tiempo concreto.

4.6. Fase 6: Análisis de Resultados

Para las tres versiones que componen este caso de estudio, se ha evaluado esta fase con el objetivo de analizar los resultados, pero sólo en la última versión se han obtenido buenos resultados, sobre los que sí se podía hacer un análisis. Sí que es cierto que con la segunda versión se obtuvieron resultados, pero estaban muy lejos de los deseados e incluso con algunas combinaciones de controladores y pasajeros eran resultados incoherentes, por eso se optó por mejorarla y volver a analizar los resultados. En cuanto a la primera versión se pudo llegar hasta la fase de simulación, pero los resultados analizados eran totalmente incoherentes, seguramente propiciado por un diseño incorrecto del modelo, por esa razón se optó por realizar profundos cambios y no reflejar dichos resultados en este análisis.

Antes de empezar con el análisis de los distintos escenarios, es necesario aclarar que en el ADRM no se ha podido localizar el volumen de pasajeros ni de cuántos minutos es la muestra para obtener los resultados que aquí se usan como estándar para comparar. Por lo tanto no se podrá concluir de forma general si se han mejorado los resultados o no, pero se analizará concretamente cada caso.

En la *Tabla 23* se recogen los resultados de la mejor de las soluciones obtenidas durante la fase de simulación, aplicada a su vez sobre la versión final del problema. Se puede observar que para pocos controladores el tiempo medio de tránsito de un pasajero está por encima del estándar marcado por el ADRM (50 minutos), pero cuanto más nos acercamos al número de controladores recomendado para un aeropuerto medio (3 controladores de aduanas y 4 de seguridad) el tiempo de tránsito medio se coloca bastante por debajo del estándar del ADRM, consiguiendo que incluso el tiempo de tránsito máximo de un pasajero se sitúe bastante por debajo de este estándar. Según este

análisis se puede concluir que para volúmenes de tránsito menores de 130 pasajeros simultáneos no sería necesario invertir en tantos recursos para alcanzar lo marcado por el ADRM, se podría estar dentro de los límites marcados con sólo dos controladores de aduanas y dos controladores de seguridad. Además, se puede comprobar que aumentando un recurso de cada tipo no se mejoran notablemente los resultados.

Otro dato interesante que se puede advertir es el efecto de cuello de botella que generan los controladores de seguridad. Se puede observar como los resultados obtenidos cuando hay menos controladores de seguridad que de aduanas son los mismos que cuando el número de recursos está igualado. Esto se debe a que por muchos controladores de aduanas que estén de servicio, como el tiempo de control de aduanas es menor que el tiempo de control de seguridad, si los controladores de seguridad no pueden atenderlos al tiempo que les llegan, se generan colas y aumenta el tiempo de espera de los pasajeros en dichas colas. En cambio, en el caso contrario el tiempo mejora, ya que cuantos más controladores de seguridad hay, menores son las colas y antes llegan los pasajeros a su puerta de embarque de destino.

Observado este cuello de botella, para el resto de casos se obviarán los escenarios en que hay menos controladores de seguridad que de aduanas.

Para los escenarios con 400 pasajeros se puede observar que cuesta mucho más alcanzar el objetivo marcado por el ADRM. Se observa que es necesario alcanzar el número de recursos recomendado por el ADRM para conseguir estar entre los 45-50 min. Aún así estaría dentro del objetivo de nivel de calidad marcado. Se puede observar que la diferencia entre tener 3+3 o 3+4 es bastante amplia, es decir, es conveniente mantener 3+4 para conseguir los niveles de calidad exigidos. Pero si se analiza el siguiente escalón, 4+4, se puede observar que ya no es tan rentable, puesto que por el coste de un recurso más sólo se mejoran 2,5 minutos el tiempo medio de tránsito. En cambio, para el siguiente escalón, 4+5, vuelve a observarse que se ganan 5,5 minutos. Según este análisis, se puede concluir que es más rentable la contratación de un controlador de seguridad que uno de aduanas.

El siguiente paso es analizar los resultados para 1000 pasajeros. Lo primero que se puede observar es el importante aumento del tiempo de simulación, se ha pasado de

escasos 4 minutos a casi 40 minutos (aproximadamente un 1000% de incremento). Se ha observado que este hecho, en parte, viene motivado por el retardo en la escritura en disco de los resultados, puesto que se generan archivos solución de más de 100MB. Además, se observa un hecho curioso, a medida que la simulación avanza, la velocidad de simulación disminuye, puesto que tiene más carga de datos que analizar, pero a medida que los pasajeros van embarcando (Transición *Emb*) la velocidad vuelve a aumentar, llegando a generar los últimos 200 nodos en menos de 10 segundos, cuando el tiempo durante el resto del proceso de simulación es de un nodo por segundo.

Tras varios escenarios simulados, se vuelve a observar que la mejora de pasar de 4+4 a 4+5 es superior a pasar de 4+5 a 5+5, esto confirma la observación hecha en los escenarios con 400 pasajeros que es más rentable un controlador de seguridad que uno de aduanas y por lo tanto no es muy beneficioso invertir en un quinto controlador de aduanas.

Se debe remarcar que en ninguno de los escenarios analizados con 1000 pasajeros se ha conseguido alcanzar el tiempo medio de tránsito deseado para cumplir el estándar del ADRM. Por esta razón, se puede concluir que son necesarias otras mejoras sobre el modelo si el objetivo es analizar gran cantidad de pasajeros.

En cuanto a los tiempos de simulación, en todos los casos son tiempos muy asumibles, ya que en el peor de los casos estamos hablando de 38 minutos, un tiempo muy por debajo de los tiempos obtenidos en las anteriores versiones del modelo para los escenarios de 130 pasajeros, por ejemplo. Esto demuestra que se ha conseguido un código fuente que ha permitido hacer una simulación para todos los escenarios propuestos.

Tabla 23: Resultados de la versión_3

Nº PSJ	AD+ SEG	Nº Nodos	(seg) $T_{serv}SEG$	(seg) $T_{max}PSJ$	(min) $T_{max}PSJ$	(seg) $T_{total}PSJ$	(min) $T_{medio}PSJ$	(seg) T_{Sim}
130	1+1	391	4840	5500	**91.66**	404755	**51.90**	**8**
130	1+2	391	3922	4597	**76.62**	346060	**44.36**	**9**
130	2+1	391	4840	5500	**91.66**	404755	**51.90**	**7**
130	2+2	391	2435	3095	**51.58**	248430	**31.85**	**8**
130	2+3	391	1981	2654	**44.23**	219758	**28.17**	**13**
130	3+2	391	2435	3095	**51.58**	248430	**31.85**	**16**
130	3+3	391	1633	2318	**38.63**	196334	**25.17**	**18**
130	3+4	391	1340	2017	**33.61**	177945	**22.81**	**22**
130	4+3	391	1633	2318	**38.63**	196334	**25.17**	**26**
130	4+4	391	1262	1911	**31.85**	170286	**21.83**	**28**
400	1+1	1201	24830	15490	**258.16**	3243400	**135.14**	**127**
400	2+2	1201	17430	8090	**134.83**	1755310	**73.14**	**150**
400	2+3	1201	16032	6704	**111.73**	1486193	**61.92**	**180**
400	3+3	1201	14951	5648	**94.13**	1264431	**52.64**	**226**
400	3+4	1201	14039	4717	**78.61**	1082858	**45.11**	**275**
400	4+4	1201	13730	4390	**73.17**	1023400	**42.64**	**150**
400	4+5	1201	13046	3718	**61.97**	888930	**37.04**	**245**
1000	2+2	3001	28530	19190	**319,83**	9939310	**165,65**	**1380**
1000	3+3	3001	22363	13048	**217,46**	6862131	**114,36**	**2028**
1000	4+4	3001	19280	9940	**165,67**	5323560	**88,73**	**1409**
1000	4+5	3001	17543	8218	**138,02**	4464212	**74,40**	**1431**
1000	5+5	3001	17430	8090	**134,83**	4408500	**73,47**	**2310**
1000	5+6	3001	16210	6885	**114,75**	3805075	**63,41**	**1689**
1000	6+6	3001	16196	6869	**114,49**	3791858	**63,20**	**2035**
1000	6+7	3001	15327	6004	**100,07**	3356689	**55,94**	**1866**

Nº PSJ: Número de pasajeros en tránsito utilizados para cada simulación.

AD+SEG: Número de controladores de aduanas y de seguridad.

Nº Nodos: Número de nodos creados durante cada simulación.

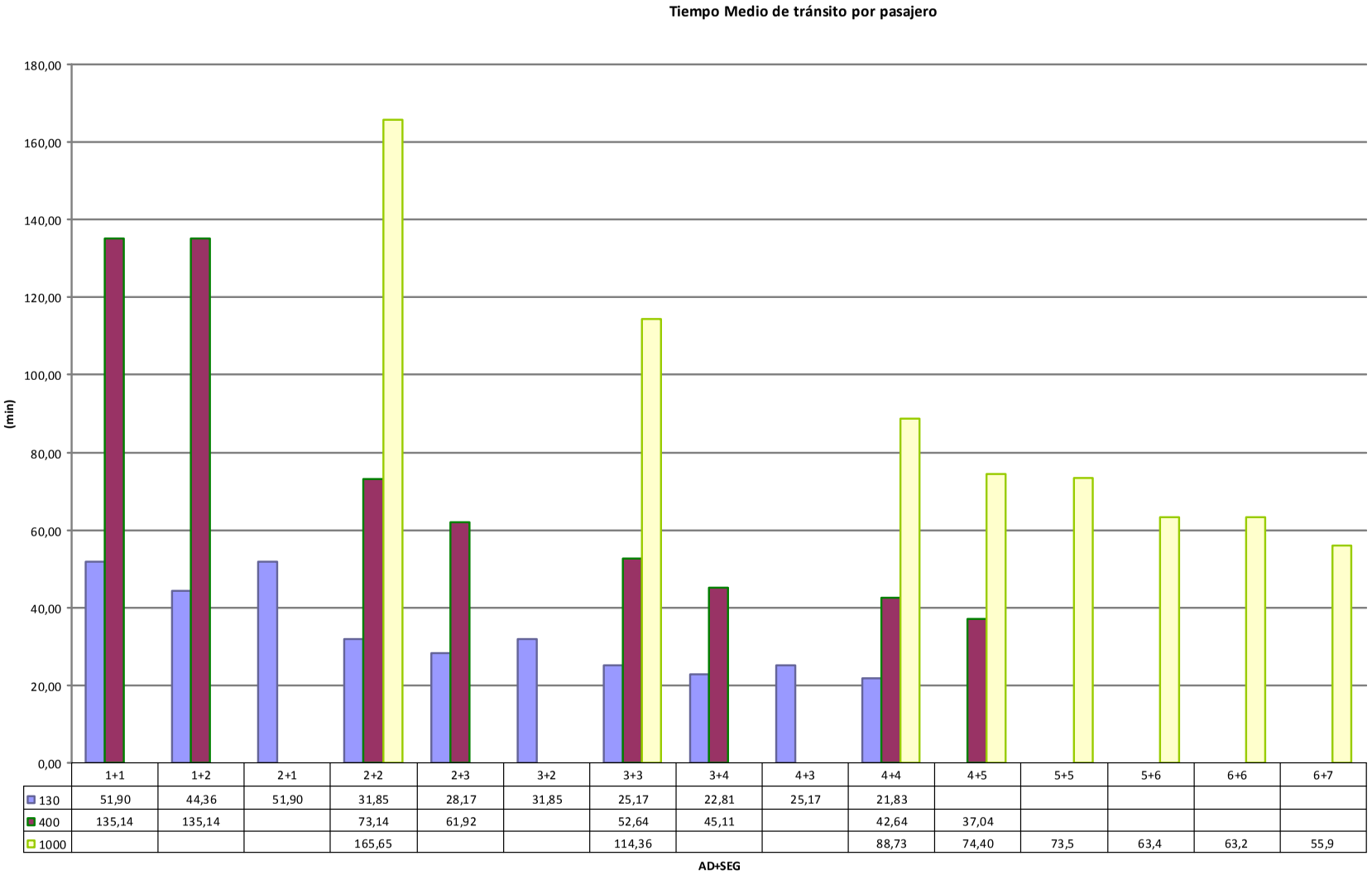
$T_{serv}SEG$*:* Tiempo medio de servicio de los controladores de seguridad.

$T_{max}PSJ$*:* Tiempo máximo (en segundos, columna 5, y en minutos, columna 6) que podría tardar un pasajero en hacer su tránsito.

$T_{total}PSJ$*:* Suma de todos los tiempos de tránsito de los pasajeros en segundos.

$T_{medio}PSJ$*:* Tiempo medio en minutos que tarda un pasajero en hacer su escala.

T_{sim}*:* Tiempo que tarda la herramienta de simulación en hallar todas las soluciones.



	1+1	1+2	2+1	2+2	2+3	3+2	3+3	3+4	4+3	4+4	4+5	5+5	5+6	6+6	6+7
130	51,90	44,36	51,90	31,85	28,17	31,85	25,17	22,81	25,17	21,83					
400	135,14	135,14		73,14	61,92		52,64	45,11		42,64	37,04				
1000				165,65			114,36			88,73	74,40	73,5	63,4	63,2	55,9

Figura 17: Comparación de tiempos según nº de pasajeros analizados

CAPÍTULO V Conclusiones

Tras evaluar la metodología siguiendo las seis fases que la componen, se concluye que esta metodología se puede utilizar de manera general para resolver cualquier problema de optimización, de transporte o de rutas.

Se puede observar que el objetivo del proyecto se ha cumplido, ya que se ha conseguido describir un problema de optimización como un SED para posteriormente modelarlo utilizando el formalismo de las RPC y simularlo utilizando las herramientas de simulación y análisis disponibles hasta conseguir unos resultados interpretables y acordes con el objetivo del problema.

Se ha podido comprobar la gran utilidad de las RPC para resolver problemas de optimización o de decisión, ya que ponen a disposición del investigador una manera de representar dichos problemas como un modelo gráfico, fácil de interpretar y de simular.

En cuanto al caso de estudio utilizado para evaluar esta metodología, se puede concluir que, con el modelo presentado, se ha obtenido un tiempo de tránsito por debajo del estándar en las pruebas con poca cantidad de pasajeros, pero para grandes cantidades de pasajeros, el resultado ha estado muy por encima del estándar ADRM para el nivel de calidad escogido. Como no se dispone de la cantidad de pasajeros para la que el ADRM realizó las pruebas, no se puede concluir si estos resultados han mejorado o no lo marcado por el ADRM.

Para finalizar se expone la planificación real de tareas llevada a cabo y las futuras mejoras que se pueden establecer para el sistema modelado.

5.1. Planificación Real de Tareas

Es muy habitual que durante el desarrollo de un proyecto, las etapas del mismo se reubiquen en el tiempo alargándose o acortándose según necesidades o situaciones no controlables. Por esa razón, en la *Figura 18* se especifica la planificación final que ha tenido el proyecto destacando los cambios realizados.

Algunos retrasos sufridos se deben a la necesidad de dedicar más tiempo del planificado a asimilar los conceptos básicos relativos al contexto del proyecto, así como para el aprendizaje de las herramientas de modelado, análisis y simulación. Debe destacarse que hasta principios de mayo de 2009 el proyecto cumplía las fechas marcadas para cada etapa.

Pero las nuevas responsabilidades laborales del alumno sumado a problemas con los tiempos de simulación en alguna de las versiones del modelo que dificultaron la fase de análisis de resultados, hicieron que el proyecto no se pudiera finalizar hasta septiembre de 2010.

Finalmente, debido a los problemas comentados, el proyecto se vio prolongado en el tiempo, aunque se le dedicaron el mismo número de horas, puesto que durante varios meses enteros no se le pudo dedicar horas de trabajo.

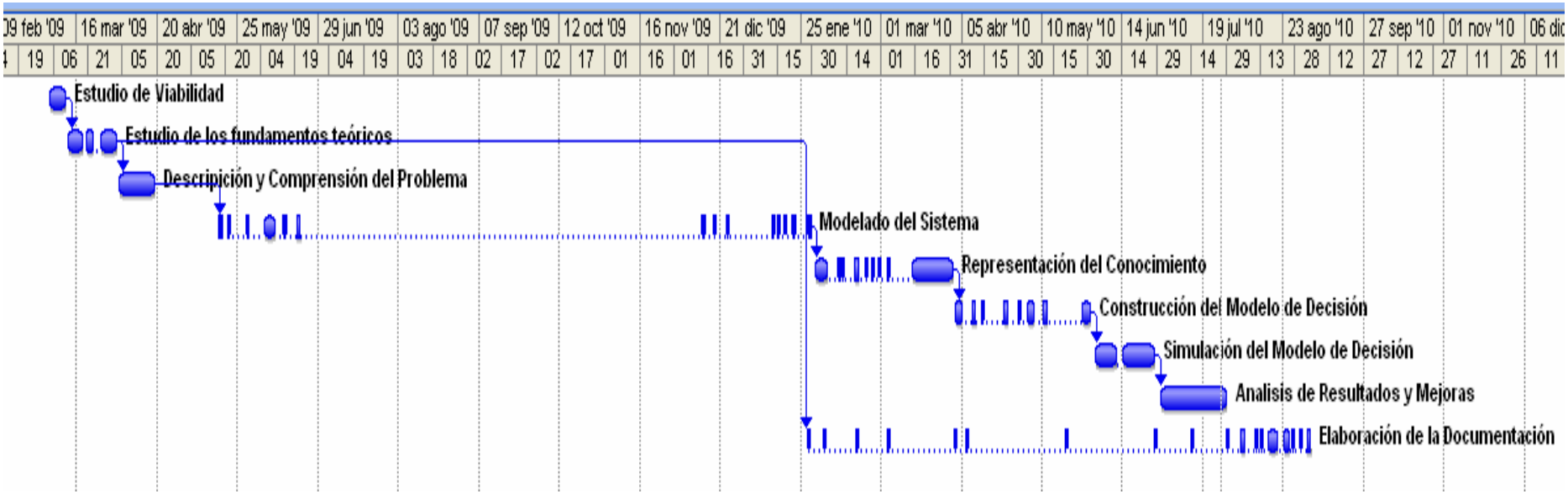


Figura 18: Planificación Final de las Tareas del Proyecto

5.2. Líneas Futuras de Investigación

Dentro de la evaluación de la metodología no hay alternativas de investigación, puesto que es una metodología ya establecida, por lo que las mejoras para el caso de estudio presentado deben centrarse en la aplicación de alguna fase en concreto de la metodología, sería el caso de la Representación del Conocimiento, incluyendo conceptos que podrían mejorar los datos obtenidos en posteriores fases; como podría ser la comparación de horas de llegada y horas de salida para mejorar la gestión de tarjetas de embarque. También, durante la cuarta fase, Construcción del Modelo de Decisión, se podría mejorar la función de costo analizando sus efectos al variar la constante o cambiando la fórmula matemática utilizada.

También se podrían utilizar nuevas heurísticas o nuevas funciones guarda que no acoten tanto el espacio de estados desplegado por la interfaz de simulación y así garantizar la posibilidad de no perder mejores soluciones.

A parte de las posibles mejoras sobre este modelo y tras comprobar, analizando los resultados, que no mejora los resultados ya existentes en el mundo real, se podría retroceder hasta la segunda fase de la metodología para diseñar otro modelado del sistema basándonos en el mismo SED, lo que implicaría una nueva concepción del sistema y podría dar mejores resultados.

5.3. Valoración Personal

Embarcarme en un proyecto de investigación como éste ha sido todo un reto para mí porque, aunque tenía una base de conocimientos sobre Redes de Petri, no conocía la metodología evaluada ni el formalismo de RPC. Este hecho ha dificultado mucho la comprensión de los conceptos básicos y la asimilación de la metodología, puesto que no había utilizado nada parecido anteriormente. Gracias a la ayuda de la directora de proyecto pude asimilar esos conocimientos y avanzar con el proyecto.

Es sorprendente las ventajas que da seguir una metodología a la hora de resolver problemas de optimización, es mucho más fácil seguir el hilo del proyecto y permite identificar mejor los posibles errores de cada fase. La ventaja de dividir un problema en

seis pequeños problemas es que resulta más fácil localizar errores de diseño, modelado, simulación, etc. Y de la misma manera es más fácil mejorar el proyecto, puesto que eliges en qué fase realizar la mejora, pruebas y si no lo consigues, intentas mejorar otra fase.

Ha sido un período de aprendizaje de mucho provecho y que valoro muy positivamente, ya que he aprendido un procedimiento a aplicar para resolver problemas de optimización que podría encontrarme a lo largo de mi vida laboral.

Bibliografía

[1] ADRM (Airport Development Reference Manual), IATA Online, Prod. Nr. 9044-09 [http://www.iata.org] y [http://www.iata.org/ps/publications/Pages/adrm.aspx]

[2] Análisis de los parámetros de los procesos de facturación y embarque bajo simulación en el Aeropuerto de Barcelona, Víctor Gayubar Arnaldo, TFC, UPC, 2008. *[http://upcommons.upc.edu/pfc/handle/2099.1/5539]*

[3]Análisis de las operaciones hub&spoke en el transporte aéreo. Aplicación al Aeropuerto de Barcelona. Juan Martínez Romero, Minor Thesis, UPC, 2003 *[http://upcommons.upc.edu/pfc/handle/2099.1/3353]*

[4] A Framework for Evaluating Level of Service for Airport Terminals. Transportation Planning and Technology, Vol. 16, Pág. 45-61. Muller, C. y Gosling, G.D., 1991 *[http://www.informaworld.com/smpp/content~db=all~content=a773498707]*

[5] Interfaz de Usuario para un Simulador de Redes de Petri Coloreadas. Marcos de Miguel Cruz, TFC, ETSE-UAB, 2008. *[http://ddd.uab.cat/pub/trerecpro/2007/hdl_2072_9034/PFCMiguelCruz.pdf]*

[6] Introducción a la Simulación de Eventos Discretos, Gabriel A. Wainer, Departamento de Computación, Universidad de Buenos Aires, 2005. *[http://www.ucema.edu.ar/~rst/Simulacion_de_Sistemas/Teoria/Introduccion_a_la_Simulacion_de_Eventos_Discretos.pdf]*

[7] An Introduction to the Theoretical Aspects of Coloured Petri Nets. K. Jensen, Aarhus University. *[http://www.daimi.au.dk/~kjensen/papers_books/rex.pdf]*

[8] Metodología para la Resolución de Problemas de Optimización Mediante la Exploración del Espacio de Estados Generado a partir de Modelos de Redes de Petri Coloreadas. Mercedes E. Narciso Farias, Tesis doctoral, UAB, 2007.'

[9] Metodologia i Gestió de Projectes, Teoria. EUIS-UAB. 2007

[10] Modelado de workflow con Redes de Petri Coloreadas Condicionales. Samuel Garrido Daniel, Tesis doctoral, Instituto Politécnico Nacional, México D.F, 2005. *[www.cs.cinvestav.mx/Estudiantes/TesisGraduados/2005/tesisSamuelGarrido.pdf]*

[11] Planifiació de la Producció, Teoria. Román Buil Gine, EUIS-UAB, 2007.