Universitat Autònoma de Barcelona

escola d'enginyeria

LLIBRERIA DE COMPONENTS PER A LA MODELITZACIÓ DEL PROCÉS DE SORTIDES DE PASSATGERS D'UN AEROPORT

Memòria del Projecte Fi de Carrera d'Enginyeria en Informàtica

realitzat per:

Marta García Roig

i dirigit per:

Juan José Ramos González

Bellaterra, 17 de juny de 2010

Índex

Annexes (format electrònic)

A1: Documentació de les classes entitat (Javadoc).

A2: Codi font d'Anylogic del model de la terminal T1 de l'Aeroport de Barcelona.

A3: Llibreria de components (llibreria.jar).

1. Introducció

En aquest capítol es presentarà el projecte i es definiran els seus objectius. A més, es farà una breu introducció a l'estat de l'art i s'explicarà l'organització de la memòria.

1.1. Presentació

Avui dia, en organitzacions que tenen sistemes definits per un gran nombre de variables, resulta difícil trobar eines analítiques que ajudin en els processos de presa de decisions, especialment quan els sistemes presenten un comportament total o parcialment estocàstic.

L'enfocament analític dels processos de presa de decisions implica, bàsicament, l'obtenció del model analític que representa el comportament del sistema (conjunt d'equacions) i el seu ús per part d'algorismes d'optimització analítica, amb l'objectiu de minimitzar una certa funció de cost. La debilitat principal d'aquest enfocament radica en que, sovint, l'obtenció del model analític implica simplificacions molt importants del comportament real del sistema, per tal de ser emprats de forma factible pels algorismes d'optimització.

En sistemes d'aquest nivell de complexitat, la presa de decisions basada en simulació per computador es presenta com una alternativa molt útil a l'ús de tècniques analítiques, doncs permet l'ús de models que representen de forma molt més fidel el comportament real del sistema. De vegades, fins i tot, pot ser l'únic enfocament pràctic realista.

L'ús de tècniques de simulació per computador satisfà un gran ventall de necessitats, des de l'estudi de la influència en el comportament d'un sistema que pot suposar la variació de determinades variables que el defineixen, fins a la imitació (simulació) de situacions perilloses, com per exemple la manipulació de materials explosius, amb l'objectiu d'estudiar els possibles efectes de determinades actuacions.

En aquest projecte es farà ús d'aquesta tècnica per modelitzar el procés de sortides de passatgers d'un aeroport. Per fer-ho, s'usarà un model de simulació orientat a esdeveniments discrets, ja que les variacions d'estat d'aquest sistema es realitzen en instants no periòdics de temps, pel que el procés de sortides de passatgers es pot definir com un model dinàmic i estocàstic.

L'eina de simulació que s'usarà en aquest projecte és l'Anylogic. Aquesta eina permet la modelització de sistemes orientats a esdeveniments discrets i de dinàmica de sistemes, pel que s'ha considerat una bona eina per complir els objectius del projecte.

El producte final d'aquest projecte és una llibreria de components totalment reutilitzable, que faciliti la modelització del procés de sortides de passatgers de qualsevol aeroport comercial. La idea principal en que es basa la construcció d'aquesta llibreria és la separació entre les característiques particulars de cada

aeroport, i el comportament comú a tots els aeroports que experimenten certs elements del procés de sortides quan es miren de manera individualitzada, és a dir, sense tenir en compte com es relacionen amb els altres elements del sistema en cada aeroport concret que s'estigui tractant.

Addicionalment, com a cas d'aplicació d'aquesta llibreria, s'ha realitzat la modelització del procés de sortides de la terminal T1 de l'Aeroport de Barcelona. Val a dir, però, que no s'ha realitzat una validació del model com a tal, és a dir, pel que fa als temps mitjos de cada subprocés, ja que només pretén servir d'exemple en l'ús dels components de la llibreria que s'ha creat i la seva interrelació amb els altres objectes genèrics de que disposa l'Anylogic.

La simulació del procés de sortides de passatgers d'un aeroport permet prendre decisions importants sobre l'assignació de recursos del procés per tal d'obtenir un funcionament òptim del mateix, en els termes en que es defineixi aquest concepte.

Val a dir que la definició d'aquest funcionament òptim en general resulta complexa i sovint implica un compromís entre elements contradictoris, com, per exemple, minimització de les cues d'espera en els mostradors de facturació (qualitat de servei) i a l'hora minimització del nombre de mostradors disponibles (minimització de costos). La presa de decisions òptima en aquest context implica l'ús combinat de la simulació amb tècniques d'optimització que no són purament analítiques (p.e. ús d'heurístiques). Aquest tipus de treball, òbviament, queda fora de l'abast d'aquest projecte.

1.2. Objectius del projecte

El principal objectiu d'aquest projecte és proveir d'una llibreria de components reutilitzables per facilitar la modelització, amb l'eina Anylogic, del procés de sortides de passatgers d'un aeroport comercial.

A més, aquesta llibreria ha de permetre modelitzar el procés de sortides:

a) Des d'una perspectiva de microsimulació, tant pel que fa al moviment individual de cada passatger, com al flux de persones.

b) Parametritzant tant les condicions de contorn (taula d'arribades de passatgers a l'aeroport, nº de vol), com l'assignació de mitjans (número de taulells de facturació, filtres de seguretat i portes d'embarcament).

I finalment, la modelització del procés de sortides fent ús d'aquesta llibreria ha de facilitar l'avaluació de forma simple de les diferents polítiques d'assignació de mitjans, i del seu impacte sobre factors de qualitat i KPI's[1] rellevants establerts per l'aeroport.

[1] El **KPI** (Key Performance Indicator) és una mètrica definida per una organització que descriu l'estat actual de l'organització en una activitat concreta.

1.3. Breu introducció a l'estat de l'art

Per complir amb els objectius dels projecte es farà servir l'aplicació AnyLogic. Aquesta eina fou desenvolupada per XJ Tecnologies per a donar suport a la modelització de la dinàmica de sistemes, sistemes d'esdeveniments discrets, i a la simulació basada en agents.

En concret ens permetrà modelitzar el moviment de persones, tant a nivell individual com de flux, que es produeix en el procés de sortides de passatgers d'un aeroport.

Aquesta eina inclou un llenguatge de modelització gràfica, però també permet ampliar els models de simulació fent servir codi Java.

Al mercat podem trobar altres eines per simular el moviment de persones, com per exemple:

- **SimWalk Transport** és una solució en 3D capdavantera per a la simulació i l'anàlisi dels desplaçaments de passatgers i el disseny d'estacions de ferrocarril, metro i autobusos. Aquesta eina permet optimitzar la fluïdesa dels moviments dels passatgers, el disseny de les estacions i els seus elements (andanes, passadissos, escales, escales mecàniques, ascensors, etc.), així com millorar els horaris, els temps de transbordament i les connexions.

- **STEP software** és una eina de simulació dissenyada pel grup Mott MacDonald que permet predir el moviment dels vianants en circumstàncies normals i d'emergència. S'originà gràcies a l'àmplia experiència del Grup en el disseny d'edificis i en el desenvolupament d'eines de simulació per al disseny d'enginyeria. Aquesta eina produeix en temps real simulacions en 3D de molt fàcil comprensió, pel que els resultats obtinguts poden ser interpretats tant per experts com per persones no especialistes en el tema. A més, és molt útil per identificar colls d'ampolla en l'assignació de medis.

- **VISSIM** és una eina desenvolupada pel grup PTV que ofereix la tecnologia per a la simulació microscòpica de vehicles i vianants. Concretament, permet mostrar la interacció real entre vianants i vehicles, i el mobiliari urbà, semàfors, passos de vianants i les parts normals dels carrers d'una ciutat. A més, aquesta eina posseeix una gran capacitat de visualització, en 3D, pel que serveix de gran ajuda als arquitectes que participen en el disseny urbà, ja que permet una presentació dels seus plans urbanístics d'una manera gràfica i entenedora per al públic en general. Per tant, aquesta eina constitueix un pont entre els enginyers de trànsit, arquitectes i encarregats de prendre decisions d'assignació de medis, que tenen poc o cap coneixement de planificació de trànsit.

1.4. Organització de la memòria

La memòria d'aquest projecte està organitzada en 7 capítols.

En el capítol 1 es fa una presentació del projecte, una descripció dels seus objectius i s'explica la organització de la memòria.

En el capítol 2 es presenta l'estudi de la viabilitat del projecte i es detalla la seva planificació temporal, juntament amb una comparació amb la seva execució.

En el capítol 3 s'expliquen el fonaments de l'Anylogic per construir models, i es fa una breu introducció a la interfície d'usuari de l'eina.

En el capítol 4 s'explica en que consisteix el procés de sortides de passatgers d'un aeroport i la relació de cada bloc del procés amb la llibreria de components que s'ha desenvolupat.

En el capítol 5 s'explica en detall cada component de la llibreria per tal que qualsevol usuari de l'Anylogic pugui fer-lo servir, així com modificar-lo i/o ampliar-lo en un futur.

En el capítol 6 s'explica la modelització del procés de sortides de la terminal T1 de l'Aeroport de Barcelona com un exemple d'ús de la llibreria de components que s'ha creat en aquest projecte.

En el capítol 7 s'expliquen les conclusions del projecte en quant a objectius complerts i no complerts, les lliçons apreses i possibles línies de continuació.

2. Planificació del projecte

En aquest capítol es presenta l'estudi de viabilitat del projecte així com la seva planificació temporal i una comparació amb la seva execució.

2.1. Estudi de viabilitat

Pel que fa als medis tècnics necessaris per a realitzar aquest projecte, es farà ús de l'eina Anylogic (es pot descarregar una versió de prova de l'adreça web: *http://www.xjtek.com/anylogic/download/*). Aquesta eina ja s'utilitza al Departament de Telecomunicació i d'Enginyeria de Sistemes (al qual pertany el director del projecte) i, tal com s'explicita a la planificació, se'm donarà formació en la mateixa.

D'altra banda, com a fonaments teòrics necessaris per a la realització del projecte, bàsicament tenim, d'una banda, la modelització de sistemes orientats a esdeveniments discrets i, d'altra, coneixements de probabilitat i estadística. Pel que fa al darrer punt, vaig adquirir els coneixements necessaris a les corresponents assignatures de matemàtiques, mentre que el coneixement relatiu als sistemes orientats a esdeveniments discrets el vaig adquirir a l'assignatura (optativa) de Planificació de la producció.

Finalment, la programació de l'eina objecte del projecte es completarà mitjançant programació en Java, llenguatge emprat a la carrera en diverses assignatures.

Per tant, la viabilitat tècnica del projecte es considera garantida en base als punts comentats.

Pel que fa a la viabilitat econòmica, aquest anàlisi no té rellevància, atès que es tracta d'un projecte acadèmic que no es pretén comercialitzar.

Tot i així, a continuació es farà una petita simulació del càlcul dels costos que s'haurien d'imputar al projecte en cas que el desenvolupament hagués anat a càrrec d'una empresa dedicada a l'elaboració de projectes informàtics per a d'altres empreses.

Treballarem amb el supòsit que l'empresa desenvolupadora del projecte disposa d'un local comercial on hi treballa una sola persona, l'empresari, dedicada exclusivament a la part de producció, durant 8 hores al dia, de dilluns a divendres. A més, l'empresa té subcontractada una gestoria per a que porti la seva comptabilitat i li faci de consultora en la presa de decisions a nivell comptable i financer.

També suposarem que per a la realització d'aquest projecte aquesta empresa ha necessitat disposar de 15 hores de consultoria amb un expert en simulació per computador. D'aquesta manera l'anàlisi també contemplarà el cost de les hores dedicades al projecte que ha invertit el director del projecte acadèmic.

A més, el calendari laboral del treballador és de 222 dies laborables a l'any.

A continuació es detalla l'anàlisi de costos que s'ha dut a terme, desglossat segons la tipologia de cada cost:

Costos fixes

Aquests tipus de costos, també coneguts com a costos estructurals, no depenen del volum de producció d'aplicacions informàtiques que tingui l'empresa.

A continuació s'enumeren els conceptes que es consideren costos fixes de l'empresa:

- Lloguer local : 600 euros /mes.
- Subministraments (aigua, llum, gas, telèfon i Internet) : 90 euros/mes.
- Material d'oficina i d'altres : 30 euros/mes.
- Gestoria : 70 euros/mes.
- Amortització d'equipament informàtic (25%) : 1.000 * 25% = 250 euros/any.
- Amortització llicència d'Anylogic (33%) : 12.250 * 33% = 4.042.5 euros/any.

La imputació d'aquests costos es farà en funció dels dies que s'ha trigat en realitzar aquest projecte. En la taula 2.1 es mostra el cost diari que suposa cadascun d'aquests conceptes:

Concepte	Cost anual	Cost Diari (222 dies/any)
Lloguer local	7200,00 euros	32,43 euros
Subministraments	1.080,00 euros	4,86 euros
Material oficina	360,00 euros	1,60 euros
Gestoria	840,00 euros	3,80 euros
Amort. Equip informàtic	250,00 euros	1,12 euros
Amort. Llicència AnyLogic	4.042,50 euros	18,20 euros

Taula 2.1: desglossament de costos fixes

Costos variables

Aquests costos depenen del volum de producció d'aplicacions informàtiques que faci el treballador de l'empresa.

Els costos variables d'aquest projecte són:

- Despeses salarials: 14 pagues de 3.000 euros.

De manera anàloga als costos fixes, la imputació d'aquest cost es farà en funció dels dies que s'ha trigat en realitzar aquest projecte.

A la taula 2.2 es mostra el cost diari que suposa aquest concepte:

Concepte	Cost anual	Cost Diari (222 dies/any)
Despeses salarials	42.000,00 euros	189,20 euros

Taula 2.2: desglossament de costos variables

- Consultoria amb expert en simulació: 15 hores, a 60 euros/hora.

La totalitat de la quantia d'aquest cost serà imputada al cost total del projecte.

Cost total del projecte

El cost que ha suposat la realització d'aquest projecte es calcula fent la suma del costos fixes i variables imputables al temps que s'ha trigat en la seva execució, que ha estat de 356 hores equivalents a 44,5 dies laborables.

A la taula 2.3 es detalla el càlcul del cost total, desglossat per conceptes.

Concepte	Cost Diari (sobre 222 dies laborables/any)	Cost del projecte (44,5 dies)
COSTOS FIXES		
Lloguer local	32,43 euros	1.443,13 euros
Subministraments	4,86 euros	216,27 euros
Material Oficina	1,60 euros	71,20 euros
Gestoria	3,80 euros	169,10 euros
Amort. Equip informàtic	1,12 euros	49,84 euros
Amort. Llicència AnyLogic	18,2 euros	809,90 euros
COSTOS VARIABLES		
Despeses salarials	189,20 euros	8.419,40 euros
Consultoria amb expert		900,00 euros
COST TOTAL PROJECTE		
Total cost projecte	251,21 euros	**12.078,84 euros**

Taula 2.3: detall del càlcul del cost del projecte

2.2. Planificació temporal

La planificació del projecte està basada en dues premisses fonamentals:

- Una dedicació total aproximada de 300 hores.
- Una dedicació diària de 3 hores (en promig) d'un únic recurs de treball: MGR (inicials del nom de la projectista).

Sota aquestes premisses, a la taula 2.4 es mostra l'estimació de durada de les diferents fases i activitats identificades per aquest projecte, així com les seves relacions de precedència:

Jornada Laboral (JL): 3 hores al dia (*)

Fases / Activitats		**Hores totals**	**Precedent**	**Hores al dia (*)**	**% JL**	**Dies**
Codi	Nom					
1	**Definició del projecte**	**15**				
2	Idea general. Definició dels objectius	3	-	3,0	100%	1,00
3	Revisió estat del art	6	2	3,0	100%	2,00
4	Estudi de viabilitat	3	3	3,0	100%	1,00
5	Planificació	3	4	3,0	100%	1,00
6	**Aprenentatge d'Anylogic**	**24**				
7	Curs introductori	3	5	3,0	100%	1,00
8	Aprenentatge autònom	21	7	1,5	50%	14,00
9	**Definició del procés de sortides**	**12**				
10	Divisió en blocs	3	7	1,5	50%	2,00
11	Identificació i definició dels components	9	10	1,5	50%	6,00
12	**Creació de la llibreria de components**	**108**				
13	Disseny dels components	9	11	3,0	100%	3,00
14	Codificació dels components	84	13	2,4	80%	35,00
15	Validació: proves unitàries dels components	15	14CC+10 dias	0,6	20%	25,00
16	**Cas d'aplicació: terminal T1**	**69**				
17	Recollida d'informació	9	14;15	3,0	100%	3,00
18	Modelització de la planta 3	30	17	3,0	100%	10,00
19	Modelització de la planta 1	21	18	3,0	100%	7,00
20	Validació: proves d'integració	9	19	3,0	100%	3,00
21	**Redacció de la memòria**	**60**	20	3,0	100%	20,00
22	**Preparació de la presentació**	**21**	21	3,0	100%	7,00

Total hores = 309

(*): Promig estimat

Taula 2.4: Dades bàsiques de la planificació del projecte.

La informació de la taula anterior és la base per a la realització de la planificació temporal del projecte que s'ha efectuat amb MS Project. Addicionalment, en aquesta planificació s'han considerat certes restriccions per compatibilitat amb la resta de la meva activitat laboral, acadèmica i personal que es tradueixen en el següent:

- Tot i que la data d'inici del projecte és l'1 de desembre de 2009, en aquesta data només es va fer una reunió per definir els objectius i l'abast general del projecte, atès que per les restriccions esmentades no podia dedicar-me al projecte fins després de Nadal.

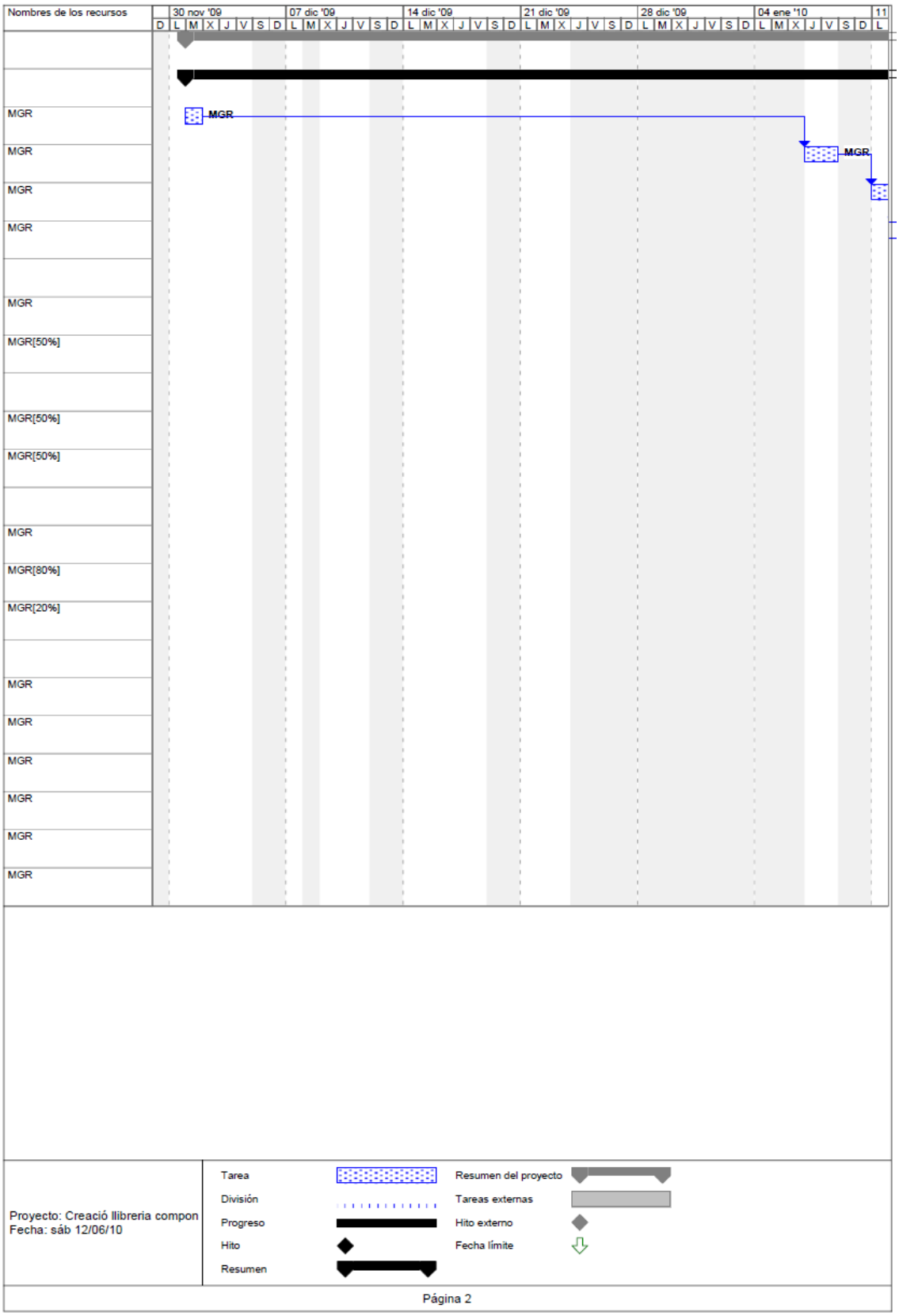
- A mitjans de gener s'ha planificat un període de 10 dies de no realització d'activitat del projecte per exàmens.
- Les vacances de Nadal i Setmana Santa es consideren també com a període de no realització d'activitat del projecte.

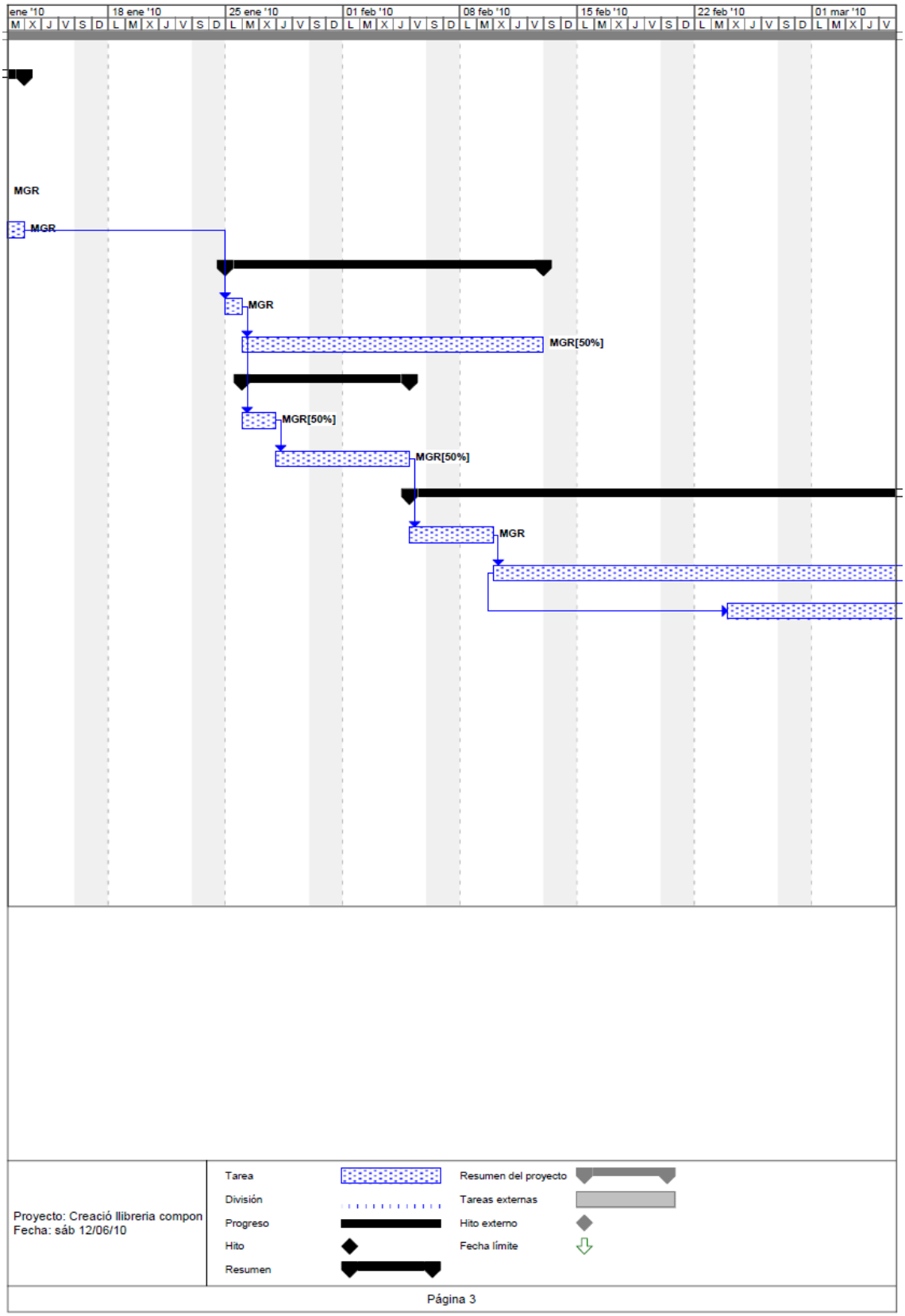
Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
0	**Creació llibreria components: procés de sortides d**	**125 días**	**mar 01/12/09**	**mié 16/06/10**	
1	**Definició del projecte**	**20 días**	**mar 01/12/09**	**mar 12/01/10**	
2	Idea general. Definició dels objectius	1 día	mar 01/12/09	mar 01/12/09	
3	Revisió estat del art	2 días	jue 07/01/10	vie 08/01/10	2
4	Estudi de viabilitat	1 día	lun 11/01/10	lun 11/01/10	3
5	Planificació	1 día	mar 12/01/10	mar 12/01/10	4
6	**Aprenentatge d'anylogic**	**15 días**	**lun 25/01/10**	**vie 12/02/10**	
7	Curs introductori	1 día	lun 25/01/10	lun 25/01/10	5
8	Aprenentatge autònom	14 días	mar 26/01/10	vie 12/02/10	7
9	**Definició del procés de sortides**	**8 días**	**mar 26/01/10**	**jue 04/02/10**	
10	Divisió en blocs	2 días	mar 26/01/10	mié 27/01/10	7
11	Identificació i definició dels components	6 días	jue 28/01/10	jue 04/02/10	10
12	**Creació de la llibreria de components**	**38 días**	**vie 05/02/10**	**mié 07/04/10**	
13	Disseny dels components	3 días	vie 05/02/10	mar 09/02/10	11
14	Codificació dels components	35 días	mié 10/02/10	mié 07/04/10	13
15	Validació: proves unitàries dels components	25 días	mié 24/02/10	mié 07/04/10	14CC+10 días
16	**Cas d'aplicació: terminal T1**	**23 días**	**jue 08/04/10**	**lun 10/05/10**	
17	Recollida d'informació	3 días	jue 08/04/10	lun 12/04/10	14;15
18	Modelització de la planta 3	10 días	mar 13/04/10	lun 26/04/10	17
19	Modelització de la planta 1	7 días	mar 27/04/10	mié 05/05/10	18
20	Validació: proves d'integració	3 días	jue 06/05/10	lun 10/05/10	19
21	Redacció de la memòria	20 días	mar 11/05/10	lun 07/06/10	20
22	Preparació de la presentació	7 días	mar 08/06/10	mié 16/06/10	21

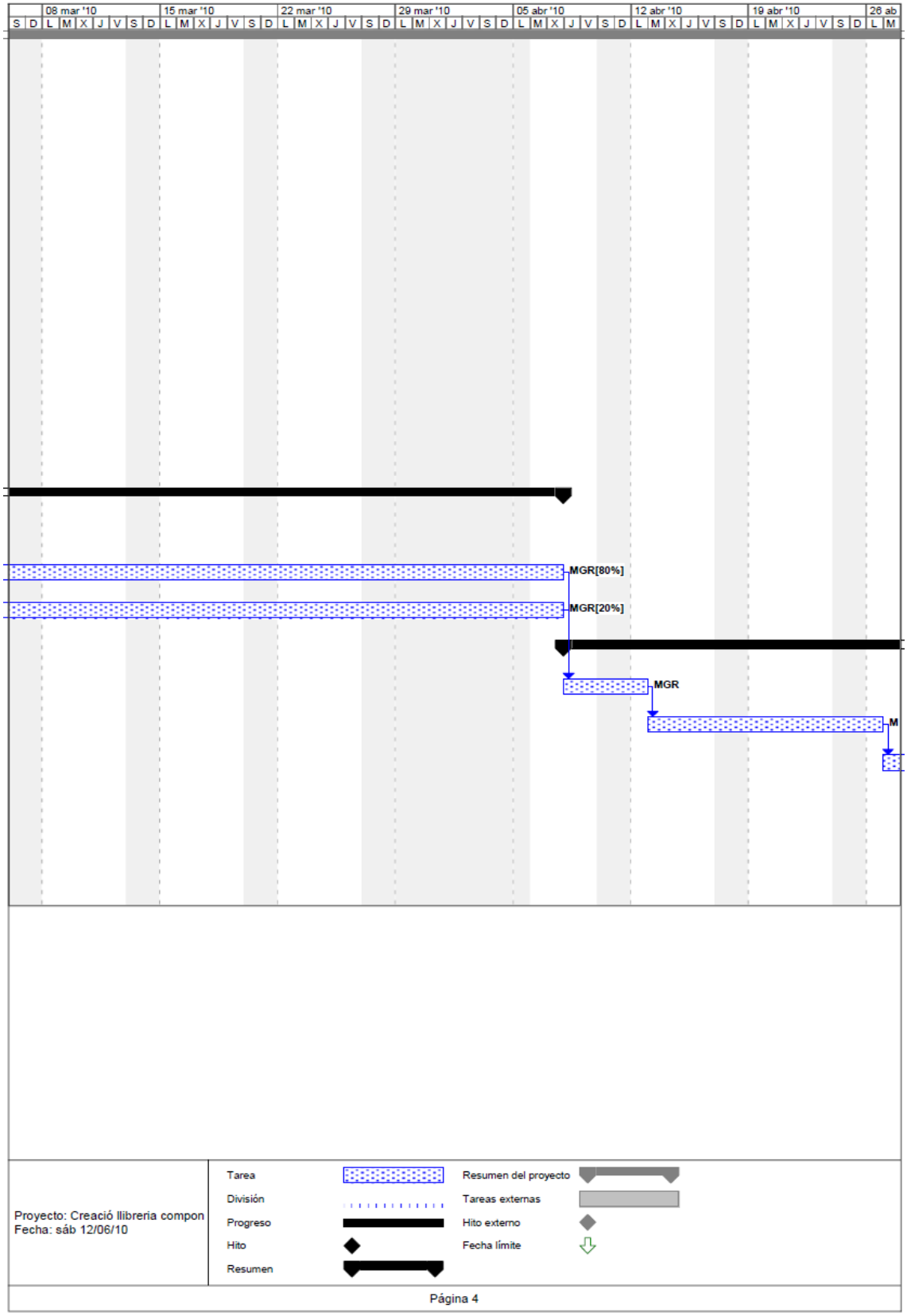
Proyecto: Creació llibreria compon
Fecha: sáb 12/06/10

Tarea — Resumen del proyecto
División — Tareas externas
Progreso — Hito externo
Hito — Fecha límite
Resumen

Página 1


Nombres de los recursos
30 nov '09
07 dic '09
14 dic '09
21 dic '09
28 dic '09
04 ene '10
D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L
MGR
MGR
MGR
MGR
MGR
MGR
MGR[50%]
MGR[50%]
MGR[50%]
MGR
MGR[80%]
MGR[20%]
MGR
MGR
MGR
MGR
MGR
MGR
MGR
MGR
Proyecto: Creació llibreria compon
Fecha: sáb 12/06/10
Tarea
División
Progreso
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite
Página 2


ene '10
18 ene '10
25 ene '10
01 feb '10
08 feb '10
15 feb '10
22 feb '10
01 mar '10
MGR
MGR
MGR
MGR[50%]
MGR[50%]
MGR[50%]
MGR
Proyecto: Creació llibreria compon
Fecha: sáb 12/06/10
Tarea
División
Progreso
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite
Página 3


08 mar '10
15 mar '10
22 mar '10
29 mar '10
05 abr '10
12 abr '10
19 abr '10
26 ab
S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L M
MGR[80%]
MGR[20%]
MGR
M
Proyecto: Creació llibreria compon
Fecha: sáb 12/06/10
Tarea
División
Progreso
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite
Página 4

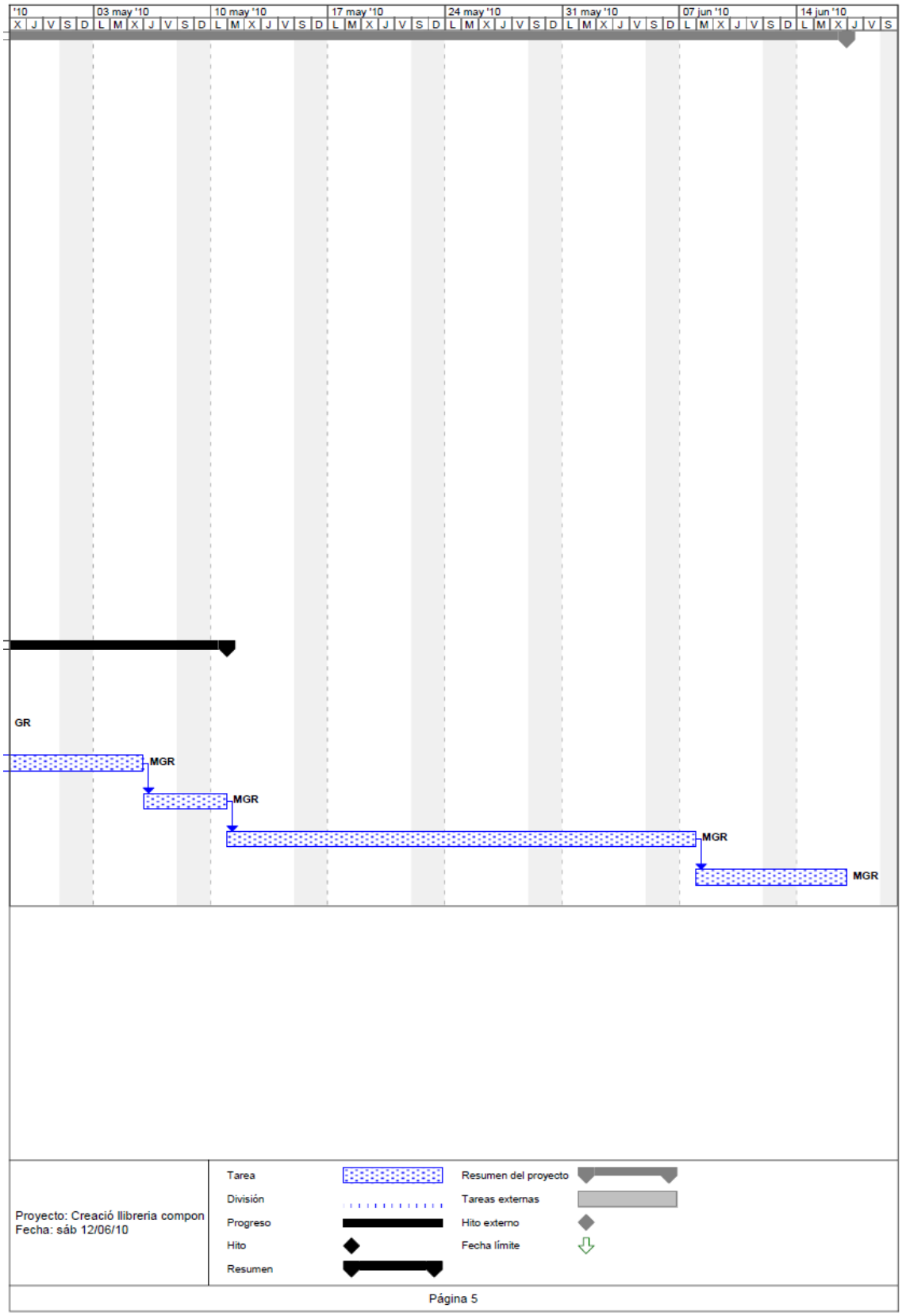


Figura 2.1: Planificació del projecte amb MS Project: dades bàsiques i diagrama de Gantt.

Així, la figura 2.1 mostra la informació bàsica d'aquesta planificació, així com el diagrama de Gantt del projecte, en format d'informe de MS Project. Com es pot veure, sota aquesta planificació la data de finalització del projecte és el 16 de

juny de 2010, el que, amb l'objectiu de presentar el projecte a la convocatòria de juny, deixa un cert marge en previsió de possibles contingències.

2.3. Comparació de la planificació amb l'execució

La taula 2.5 mostra les hores reals dedicades a cada activitat del projecte, comparades amb la planificació inicial.

Fases / Activitats		**Hores planifi-cades**	**Hores reals**	**Dif.**	**Dif. (%)**
Codi	Nom				
1	**Definició del projecte**	**15**	**11**	**-4,0**	**-26,7%**
2	Idea general. Definició dels objectius	3	1	-2,0	-66,7%
3	Revisió estat del art	6	3	-3,0	-50,0%
4	Estudi de viabilitat	3	3	0,0	0,0%
5	Planificació	3	4	1,0	33,3%
6	**Aprenentatge d'Anylogic**	**24**	**18**	**-6,0**	-25,0%
7	Curs introductori	3	3	0,0	0,0%
8	Aprenentatge autònom	21	15	-6,0	-28,6%
9	**Definició del procés de sortides**	**12**	**15**	3,0	25,0%
10	Divisió en blocs	3	3	0,0	0,0%
11	Identificació i definició dels components	9	12	3,0	33,3%
12	**Creació de la llibreria de components**	**108**	**122**	**14,0**	**13,0%**
13	Disseny dels components	9	12	3,0	33,3%
14	Codificació dels components	84	92	8,0	9,5%
15	Validació: proves unitàries dels components	15	18	3,0	20,0%
16	**Cas d'aplicació: terminal T1**	**69**	**75**	**6,0**	**8,7%**
17	Recollida d'informació	9	7	-2,0	-22,2%
18	Modelització de la planta 3	30	37	7,0	23,3%
19	Modelització de la planta 1	21	19	-2,0	-9,5%
20	Validació: proves d'integració	9	12	3,0	33,3%
21	**Redacció de la memòria**	**60**	**94**	**34,0**	**56,7%**
22	**Preparació de la presentació**	**21**	**21**	**0,0**	**0,0%**
	Total hores =	**309**	**356**	**47,0**	**15,2%**

Taula 2.5: Comparació de la planificació temporal del projecte amb la seva execució.

Els comentaris més rellevants d'aquesta comparativa serien els següents:

- El treball global ha estat un 15,2% superior al planificat. Aquesta desviació es considera raonable, atesa la poca experiència en planificació i execució de treballs d'aquesta envergadura.

- La major desviació s'ha produït en la redacció de la memòria, que ha necessitat un 56,7% més del treball planificat. Això serà un referent valuós de cara a futurs treballs d'aquest tipus.
- La tasca de codificació (tasca 14: Codificació dels components –de la llibreria) que sovint és la que més s'allunya, per excés, de la planificació només ha necessitat un 9,5 % més del treball planificat.
- La resta de tasques han seguit la planificació de manera bastant acurada, amb petites diferències, per excés o per defecte, que si bé en alguns casos són importants en termes percentuals, es deu a que són tasques de curta durada.

Pel que fa a la distribució temporal del treball, la planificació ha constituït una guia molt valuosa. S'ha tractar de seguir al màxim, i s'ha aconseguit fer-ho amb bastant precisió. Així, la confecció de la memòria es va finalitzar el dia 16 de juny. Aquesta era la data prevista per a la finalització total del projecte (incloent-hi la preparació de la presentació) segons la planificació. Val a dir que en aquesta data la preparació de la presentació estava, aproximadament, a un terç de la seva realització, de manera que, en funció del temps invertit en aquest primer terç, es va estimar que la realització total d'aquesta tasca podria complir el temps planificat.

3. Fonaments: Anylogic

En aquest capítol s'explicaran els fonaments en que es basa l'Anylogic per a la construcció de models, i s'explicarà breument l'entorn de desenvolupament de l'eina, a nivell d'usuari. D'aquesta manera es pretén facilitar el seguiment de les explicacions realitzades ens els següents capítols, per part de persones no usuàries de l'eina, però en cap cas es té per objectiu que aquest capítol constitueixi un manual o un tutorial d'Anylogic. Amb aquest objectiu es poden consultar els manuals i tutorials d'Anylogic continguts a l'adreça web *http://www.xjtek.com/anylogic/resources/documentation/.*

3.1. Fonaments

L'Anylogic és una eina de simulació basada en el paradigma de programació orientada a objectes que constitueix un entorn obert programable totalment en Java.

Per la realització d'aquest projecte s'ha fet ús de la versió 6.4.1 d'aquesta eina.

Principalment, hi ha dos tipus de classes amb les que treballa l'Anylogic:

- ActiveObject class

  Aquesta classe, pertanyent al package com.xj.anylogic.engine de l'Anylogic, es el principal bloc de construcció de models Anylogic, ja que disposa del mètode main().

  És una classe base, tant per a les llibreries de classes que proveeix l'Anylogic, per exemple la llibreria Pedestrian, com per a totes les classes d'objectes actius creats per l'usuari, que pot tenir paràmetres, variables, ports, esdeveniments, gràfics d'estat, i d'altres objectes actius incrustats.

- Java class

  L'Anylogic permet crear noves classes Java que seran usades pels objectes actius, de la classe ActiveObject, creats per l'usuari o per objectes actius de llibreries pròpies de l'Anylogic.

  Mitjançant herència, es pot crear una jerarquia de classes a partir de l'especialització de classes mare que l'Anylogic proveeix, com per exemple: la classe Ped, que pertany a la llibreria de classes Pedestrian, i les classes Entity i resourceUnit, que pertanyent a la llibreria de classes Enterprise.

  A més, fent ús del polimorfisme es pot adaptar el comportament dels mètodes de les classes mare de la jerarquia.

A part de les classes que l'usuari pot crear, l'Anylogic disposa de llibreries de classes a disposició de l'usuari, que contenen classes predissenyades i parametritzables per tal que l'usuari les pugui adaptar al model que estigui desenvolupant. Les més importants pel que fa a l'ús en aquest projecte són:

Pedestrian Library

Aquesta llibreria conté un conjunt de classes, del tipus ActiveObject class, dedicades a simular els fluxos de vianants dins d'un entorn físic.

Fent ús d'aquestes classes es poden crear models com per exemple una estació de tren, un aeroport, etc.

Aquests models permeten recollir estadístiques sobre la densitat de vianants en les diferents àrees de l'entorn físic que s'ha creat al model, per tal de detectar, per exemple, possibles problemes amb la geometria interior (en l'adició d'obstacles), per assegurar un rendiment acceptable dels punt de servei amb càrrega hipotètica que s'hagin modelitzat, per calcular els temps d'estància en àrees específiques, etc.

En els models creats fent ús d'aquestes classes, com és el cas de la llibreria de components que s'ha creat en aquest projecte per modelar el procés de sortides de passatgers d'un aeroport, els vianants es mouen per un espai continu, i reaccionen davant els diferents tipus d'obstacles que contingui l'entorn físic, inclosos altres vianants, fent ús de regles físiques.

La classe principal d'aquesta llibreria és la classe Ped, que és una classe del tipus Java class, que representa un vianant, i que es usada per les classes del tipus ActiveObject class que conté la llibreria. D'aquesta classe es poden crear tantes subclasses com es vulgui, mitjançant herència, que continguin les característiques i el comportament específic dels vianants que es moguin per l'entorn físic del model concret que s'estigui dissenyant.

Enterprise Library

Aquesta llibreria conté un conjunt de classes, del tipus ActiveObject class, que es fan servir en la modelització d'esdeveniments discrets. Per tant, amb l'ús d'aquesta llibreria es pot modelitzar la realitat en termes d'entitats (productes, clients, transaccions, etc.), processos (seqüències d'operacions que acostumen a implicar l'ús de recursos) i recursos.

Aquests processos s'especifiquen en forma de diagrames de fluxos.

Alguns exemples de models que farien ús d'aquesta llibreria són: una planta industrial, un centre de trucades, un determinat procés de negoci, etc.

A més, aquesta llibreria permet crear animacions molt sofisticades dels models de processos.

La classe base d'aquesta llibreria és la classe Entity, que és una classe del tipus Java class que representa les entitats del procés, com per exemple els productes d'una fàbrica, els clients d'una tenda, etc, que fan servir altres classes del tipus ActiveObject class que conté la llibreria.

En un mateix model es poden combinar objectes d'aquestes dues llibreries, ja que la classe Ped de la llibreria Pedestrian és una classe que hereta directament de la classe Entity de la llibreria Enterprise.

La principal diferència entre les dues llibreries és el tipus de moviment que experimenten durant la simulació els objectes Entity i els objectes Ped. És a dir, fent ús de la llibreria Enterprise, el moviment de les entitats depèn dels objectes que s'hagin fet servir en la construcció del diagrama de flux. Per exemple, si s'ha usat un objecte que simula una cinta transportadora, quan els objectes Entity passin per la cinta, durant la simulació, es mouran seguin el moviment determinat per l'objecte cinta transportadora (un darrere l'altre, a una velocitat determinada, etc). En canvi, el moviment que experimenten els objectes Ped de la llibreria Pedestrian, depèn de l'entorn físic del model; és a dir, dels obstacles, parets i altres vianants que hi hagi a la simulació.

3.2. Interfície d'usuari

L'entorn de desenvolupament de l'eina es basa en l'IDE Eclipse, pel que s'hi poden trobar moltes semblances en quant a les interfícies d'usuari de les dues aplicacions.

Tal i com es mostra a la figura 3.1, la interfície d'usuari de l'Anylogic disposa de diferents vistes i perspectives que permeten una navegació fàcil per la informació.

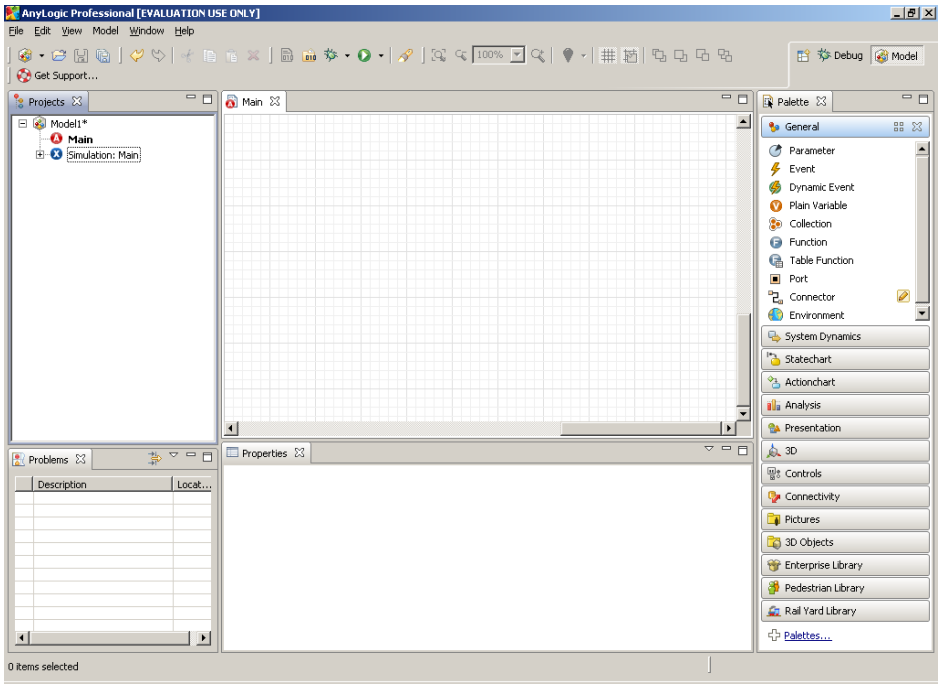


Figura 3.1: Interfície gràfica de l'Anylogic.

Les vistes més destacades en quant a la seva funcionalitat són:

- Vista *Projects*: que proporciona accés als projectes Anylogic que hagi actualment oberts a l'espai de treball.

- Vista *Properties*: que permet la visualització i/o modificació de les propietats de l'element del model seleccionat.

- Vista *Palette*: que conté la llista d'elements que proveeix l'Anylogic per construir un model, agrupats per categories. Aquests elements predissenyats es poden configurar i personalitzar mitjançant codi Java, per tal d'adaptar-los al model concret que s'estigui desenvolupant.

  Les principals categories de la Vista *Palette* que s'han usat en aquest projecte són les següents:

  General: conté els elements per definir la dinàmica del model, la seva estructura i les dades.

  Presentation: conté les formes, per exemple: línies, polylínies, rectangles, etc, que es poden usar per crear les presentacions dels models.

  Connectivity: conté els elements necessaris per a la connexió a fitxers de dades externs, i la seva posterior lectura i/o escriptura.

  A més, la vista *Palette* també permet l'accés a objectes de les llibreries que proveeix l'Anylogic, que són: Pedestrian Library, Enterprise Library i Rail Yard Library, i llibreries realitzades per tercers.

Dins la interfície d'usuari, també trobem un editor gràfic associat a cada objecte de la classe ActiveObject que es crea al model. En la figura 3.1 es pot veure l'editor gràfic associat a l'objecte actiu Main que, com es pot apreciar, és una vista que conté una quadrícula.

Dins l'editor gràfic de cada objecte ActiveObject es defineix la seva estructura, que pot estar composta de diversos elements, com, per exemple: altres objectes actius incrustats, interconnexions entre aquests objectes incrustats, presentació del model mitjançant objectes de la categoria *Presentation* de la vista *Palette*, gràfics d'estat, esdeveniments, etc.

Per incrustar altres objectes actius dins l'editor gràfic d'un objecte del tipus ActiveObject class, només cal triar un element d'alguna de les categories de la vista *Palette*, incloses les llibreries, i arrossegar-lo amb el ratolí a dins de l'editor.

Per exemple, cada component de la llibreria que s'ha creat en aquest projecte és un objecte del tipus ActiveObject class, que té una estructura formada per un conjunt d'objectes del tipus ActiveObject class pertanyents a les llibreries Pedestrian i Enterprise, incrustats i interconnectats dins l'editor gràfic del

component. A més, aquests objectes incrustats fan servir una classe del tipus Java class, creada per herència de la classe Ped de l'Anylogic: la classe Passenger, que simula un grup d'1 a 5 passatgers que viatgen junts.

A grans trets, la creació d'un model Anylogic que simuli el desplaçament de persones dins un entorn físic determinat, com ara la modelització del procés de sortides de passatgers d'un aeroport tractat en aquest projecte, consisteix bàsicament en el següent:

- Creació de la presentació del model.

  La presentació del model consisteix en la definició de l'entorn físic per on es mouran durant la simulació els objectes de la classe Ped, o d'una subclasse d'aquesta, que hagi creat l'usuari.

  L'Anylogic permet construir presentacions 2D fent ús d'un objecte de la llibreria Pedestrian: PedGround, que guarda l'estructura de l'entorn físic del model. La presentació es realitza mitjançant un dibuix, dins l'editor gràfic de l'objecte principal del model, compost de varies formes: cercles, rectangles, línies, etc.

  Les formes de dibuix de que disposa l'eina es troben dins la categoria *Presentation* de la vista *Palette*. Cadascuna d'aquestes formes són objectes que tenen una sèrie de propietats que defineixen el seu aspecte visual: posició, altura, amplada, color, etc. A més, les formes que composen la presentació del model poden ser animades, ja que disposen de propietats dinàmiques que permeten a l'usuari definir el seu valor real durant la simulació.

- Creació de la lògica del model.

  La lògica del model defineix el comportament que ha de tenir el model durant la simulació.

  Aquesta lògica es defineix mitjançant la incrustació, dins l'editor gràfic de l'objecte Main del model, d'un conjunt d'objectes interconnectats entre sí, que poden pertànyer a llibreries o packages que proveeix l'Anylogic, o bé poden ser creats pels mateixos usuaris, com, per exemple, els components de la llibreria que s'ha creat en aquest projecte.

  La interconnexió d'aquests objectes, per formar la lògica del model, es realitza mitjançant l'ús de línies connectores que enllacen els ports d'entrada d'alguns d'aquests objectes amb els ports de sortida d'alguns altres.

  Val a dir que els objectes poden disposar d'un nombre variable de ports d'entrada i sortida, depenent de la funcionalitat de cada objecte.

A més, cal definir les propietats de cada objecte incrustat, dins la vista Properties que proveeix la interfície d'usuari del simulador, per tal d'adaptar el seu comportament al model concret que s'estigui dissenyant.

4. Descripció del procés de sortides d'un aeroport

En aquest capítol s'explicaran totes les operacions que composen el procés de sortides d'un aeroport, des del punt de vista d'aplicació en simulació amb l'Anylogic. És a dir, s'explicarà en què consisteix cadascuna de les operacions i, addicionalment, es farà referència als components de la llibreria (explicats en el capítol següent) que implementen l'operació. D'aquesta manera els components de la llibreria queden emmarcats en el seu context global del procés de sortides d'un aeroport.

En aquest projecte s'ha considerat que el procés de sortides d'un aeroport abasta des de l'arribada d'un passatger a una determinada terminal, fins a l'embarcament. En aquest procés s'estudia només el comportament del passatger, el procés que segueix l'equipatge un cop s'ha facturat no és objecte d'estudi.

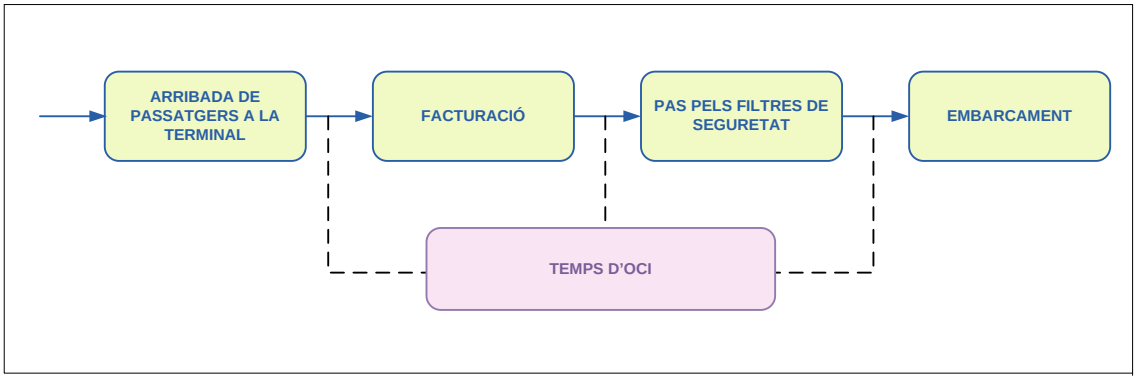


Figura 4.1: Esquema del procés de sortida de passatgers d'un aeroport

Aquest procés el podem desglossar en els 5 blocs mostrats a la figura 4.1 i que s'expliquen a continuació.

4.1. Arribada de passatgers a la terminal

En la realitat, els passatgers poden arribar a la terminal fent ús de diversos medis de transport, com poden ser: el vehicle propi, el taxi, l'autobús, el metro, el tren, etc., en funció de l'accessibilitat de cada aeroport.

Depenent dels mitjans de transport utilitzats, l'accés a la terminal es realitza per una entrada o una altra. Per exemple, en cas d'arribar a la terminal amb un vehicle propi, l'aeroport pot disposar d'un aparcament soterrat des del que es pot accedir a la terminal mitjançant ascensors.

En relació amb això, definirem el *tipus d'accés* com una agrupació de mitjans físics a triar pel passatger per entrar a la terminal; per exemple, conjunt de portes a peu de carrer o ascensors i cintes mecàniques, de manera que hi hagi coherència entre el mitjà de transport usat pel passatger per arribar a la terminal i el mitjà físic usat per entrar-hi.

Pel que fa a la forma en que arriben els passatgers, s'ha considerat que les arribades es faran en grups familiars d'1 a 5 persones, que viatgen juntes en el mateix vol, i que es mouran per la terminal de manera agrupada. En aquest sentit, és important controlar tant el número de grups que es crearan durant la simulació, com el nombre de passatgers que formarà cada grup, ja que tots els passatgers han de tenir un vol assignat. El color de cada objecte passatger que creï la font d'arribada indicarà el tipus de grup, d'1 a 5 persones, que representa l'objecte.

No s'ha considerat la creació d'acompanyants de passatgers, tot i l'impacte que poden tenir sobre les zones públiques de l'aeroport, ja que s'han centrat els esforços en la modelització del consum dels recursos crítics del procés de sortides que realitzen els passatgers. Aquests recursos són principalment: els taulells de facturació, els filtres de seguretat i les portes d'embarcament.

Per modelitzar aquest bloc s'han creat dos components a la llibreria, que són els següents:

Font de passatgers

És un component que injecta passatgers a la simulació. Hi ha una font de passatgers per cada tipus d'accés a la terminal.

La quantitat i freqüència de les aparicions de passatgers a la simulació depèn de les taules d'arribades. Aquestes dades s'obtenen d'un arxiu extern de dades del model que conté la programació de vols d'una determinada terminal per un dia en concret. Aquestes dades canviaran en funció de la terminal i del dia que es vulgui modelitzar.

Porta d'entrada a la terminal

És un component que serveix per modelitzar qualsevol medi físic d'entrada a la terminal, per exemple una porta, unes escales mecàniques, etc.

4.2. Facturació

El bloc de facturació consisteix bàsicament en la recollida de la targeta d'embarcament i el lliurament de l'equipatge que el passatger no pugui dur a sobre quan embarqui.

Donat que els medis de que disposa l'aeroport per a realitzar la facturació de passatgers i equipatge són limitats, diàriament es realitza una programació de medis en la que s'assigna un conjunt de mostradors de facturació a cada operador de handling[2]. A la seva vegada, cada operador de handling pot decidir, en funció de la càrrega diària prevista de passatgers per franges

[2] L'operador de handling de passatge és l'encarregat, dins l'aeroport, de realitzar les operacions relacionades directament amb els passatgers, com són la facturació de passatgers i dels seus equipatges, i les operacions d'embarcament i desembarcament d'avions.

horàries i companyies aèries, quins mostradors assignar a cada companyia aèria a cada franja horària.

Per tant, en un mateix dia i segons la franja horària, un mateix mostrador de facturació pot passar de disponible a no disponible, i de disponible per a una companyia aèria a disponible per a una altra.

S'ha considerat que el passatger té dues alternatives per facturar presencialment: La primera consisteix en acudir a un dels mostradors de facturació del rang de mostradors que tingui assignat el seu vol, en funció de la seva disponibilitat. La segona alternativa consisteix en usar els caixers d'autofacturació de que disposen la majoria d'aeroports. En aquest segon cas es suposa que el passatger no ha de facturar equipatge.

Avui dia, però, la facturació es pot realitzar online, pel que en cas que el passatger no necessiti facturar maletes, es saltarà aquest bloc.

Cada vol disposa d'un rang de mostradors en els quals els passatgers poden realitzar la facturació, en funció de la seva disponibilitat. Els números de mostradors que defineixen els límits del rang s'obtenen d'un arxiu d'entrada de dades. Aquest fitxer també conté les taules de disponibilitat dels mostradors, ja que depenent de la franja horària, el passatger es trobarà un o altre subconjunt de mostradors disponibles per poder facturar.

Segons com s'organitzin les cues de facturació, hi ha dues maneres de facturar:

4.2.1. Mostradors de facturació individuals.

En aquest mètode de facturació, els passatgers acudeixen directament a un mostrador de facturació del rang de mostradors assignats al vol del passatger.

Per modelitzar aquesta situació s'ha creat el següent component:

Taulell de facturació

Aquest element modelitza la lògica de la cua prèvia que ha de realitzar el passatger per facturar en un determinat mostrador, el temps d'atenció al taulell que rep el passatger per recollir la targeta d'embarcament, i la facturació de l'equipatge.

A més, per tal de tractar el tema de la disponibilitat dels mostradors de facturació, s'ha afegit el mètode *GoToCounter()* a la classe Passenger (que modelitza els grups de passatgers que es mouran pel model). Aquest mètode permet decidir a cada objecte Passenger cap a quin mostrador a d'anar a facturar en funció de la programació de medis del dia que es vulgui simular, i que es llegeix del fitxer d'entrada de dades del model.

4.2.2. Grups de multifacturació.

Un grup de multifacturació es defineix com un conjunt de mostradors que comparteixen una mateixa cua. Dins la seva dinàmica, el passatger que està primer a la cua ha d'esperar a que hi hagi un mostrador del grup disponible per a poder facturar.

Hi ha vols que tenen assignat un grup de multifacturació per a que els passatgers realitzin la facturació presencial. Aquesta situació es llegeix de l'arxiu extern de dades del model.

Per modelitzar aquesta situació s'han creat dos elements a la llibreria de components:

Serpentí

És un component que modelitza una cua d'espera de passatgers. En aquest cas la cua d'espera es compartida pels diferents mostradors del grup de multifacturació.

Taulell multifacturació

És el component que modelitza cadascun dels mostradors que formen el grup de multifacturació. En concret modelitza la lògica del procés d'atenció que rep el passatger al mostrador i el procés de facturació de l'equipatge.

Per tant, un grup de multifacturació estarà compost d'un component de tipus *Serpentí*, i un conjunt de components de tipus *Taulell de MultiFacturació*.

A més, per tal de tractar el tema de la disponibilitat dels mostradors que composen el grup de multifacturació, s'ha creat el mètode *availabilityTMF()* dins la classe Passenger (classe que modelitza els grups de passatgers que es mouran pel model). Aquest mètode permet que cada objecte Passenger que està al cap de la cua del *Serpentí* sàpiga quins mostradors del grup de multifacturació estan disponibles, en funció de la programació de medis del dia que es vulgui simular, per tal que pugui decidir a quin mostrador dirigir-se per a facturar.

4.3. Pas pels filtres de seguretat

Un cop el passatger ja ha facturat, el següent punt del procés de sortida és el pas per filtres de seguretat.

La zona de filtres de seguretat està formada per un conjunt d'arcs de seguretat pels quals han de passar obligatòriament els passatgers que volen anar a la zona d'embarcament.

Igual que els taulells de facturació, els filtres de seguretat són recursos de l'aeroport que depenen d'una programació diària de medis que en condiciona la

seva disponibilitat segons franges horàries. D'aquesta manera l'aeroport intenta realitzar una gestió el més eficient possible dels medis que es necessiten a la zona de filtres de seguretat, com són els arcs de seguretat, el personal de control, etc. Aquesta programació s'extreu de les taules de disponibilitat que conté l'arxiu de dades del model.

Per modelitzar aquest subprocés s'ha creat el següent component a la llibreria:

Filtre de seguretat

És el component que modelitza de manera simplificada el pas per l'arc de seguretat i posterior recollida d'objectes personals dipositats prèviament a la cinta de l'scanner, com un únic temps d'espera en un punt de la zona de filtres.

A més, es fa ús del component *Serpentí* per modelitzar la cua de persones que es forma davant dels arcs de seguretat. Segons l'aeroport que s'estigui modelitzant, la zona de filtres de seguretat disposarà d'un o més serpentins.

Per tractar el tema de la disponibilitat dels components *Filtres de Seguretat*, s'ha creat el mètode *availabilityFilters()* dins la classe Passenger. Aquest mètode permet que cada objecte Passenger que està al cap de la cua del *Serpentí* pugui consultar quins filtres de seguretat estan disponibles, en funció de la programació de medis del dia que es vulgui simular, per tal de que pugui decidir cap a quin filtre de seguretat ha de dirigir-se.

4.4. Embarcament

Un cop el passatger ha creuat la zona de filtres de seguretat del terminal, arriba a la zona d'embarcament. Aquesta zona sol estar dividida en subzones segons la procedència i destinació dels vols, normalment identificades amb lletres de l'abecedari, per exemple, zona A.

En aquest subprocés del procés general de sortides d'un aeroport, un cop el passatger arriba a la porta que té assignada a la seva targeta d'embarcament, s'espera a la sala que sol estar situada davant la porta d'embarcament fins que s'obri la porta. Un cop es pot embarcar, el passatger ensenya la seva targeta a la persona encarregada de fer la revisió i posteriorment desapareix de la simulació. En cas de que coincideixin alhora varis passatgers davant la porta d'embarcament, es formarà una cua per ordre d'arribada davant del mostrador de revisió de les targetes.

Per modelitzar aquest comportament s'ha creat el següent component a la llibreria:

Porta d'embarcament

Aquest component està format per una zona d'espera, una cua d'espera davant la porta i un punt de parada en el que es modelitza el temps que es triga en revisar la targeta d'embarcament i embarcar.

En la realitat, es podria donar el cas de que un passatger no arribi a temps a la porta d'embarcament que te assignada, i per tant perdi el seu vol. Aquesta situació està contemplada dins aquest component per tal de que en aquest cas es no comptabilitzi com a passatger que a finalitzat amb èxit el procés d'embarcament.

Cada vol té assignada una porta d'embarcament determinada. Aquesta informació s'extreu d'un arxiu d'entrada de dades a la simulació.

4.5. Temps d'oci

Segons la destinació del vol, els passatgers arriben a l'aeroport amb més o menys marge de temps respecte a l'hora de sortida del vol. Per exemple, en vols internacionals es recomana arribar entre 2 i 3 hores abans de la sortida prevista del vol.

Això fa que els passatgers que arriben a l'aeroport amb bastant marge disposen de temps que poden aprofitar per visitar els establiments comercials que hi ha dins de la terminal.

Per tant, tot i que el procés de sortida es pot definir en 4 grans blocs seqüencials: arribada de passatgers al terminal, facturació, pas per filtres de seguretat i embarcament, per explicar el procés complet cal disposar d'un cinquè bloc que modeli el desplaçament lliure que sol realitzar el passatger durant el temps d'oci de que disposa entre mig de cada bloc.

Aquest desplaçament lliure per dins l'aeroport que realitzen els passatgers, segueix un comportament totalment estocàstic, ja que, depenen de decisions personals i circumstancials, els passatgers opten per dirigir-se cap a un tipus d'establiments o cap a uns altres.

A més, la definició d'un model de desplaçament lliure durant el temps d'oci de que disposa cada passatger, ve fortament condicionat pel passatger. Per exemple, l'hora d'arribada del passatger a l'aeroport, o el que és el mateix, el temps que li queda fins a l'embarcament, condiciona el temps de lleure de que disposarà i, per tant, el nombre d'establiments comercials que podrà visitar . També, les característiques personals de cada passatger, com l'edat, el sexe, les aficions, etc. poden determinar les preferències en quan al tipus d'establiments comercials a visitar.

Una altra qüestió important que cal tenir present a l'hora de modelitzar aquest bloc és la dependència de l'aeroport pel que fa a la ubicació, tipus i nombre d'establiments comercials de que disposarà el model.

Tal i com es resumeix a la figura 4.1, el passatger pot disposar de temps d'oci en els següents estats:

- Un cop ha arribat a l'aeroport i està pendent de facturar.

- Un cop ha facturat i està pendent de passar per la zona de filtres de seguretat.
- Un cop ha passat per la zona de filtres de seguretat i està pendent d'embarcar.

El desplaçament lliure que realitza el passatger durant el seu temps d'oci és materialitza de moltes maneres; per exemple, prenent un cafè al restaurant, passejant per la terminal mirant els aparadors de les tendes, comprant, etc.

Per tant, per modelitzar aquest bloc s'han creat tres components que defineixen diferents mides de rutes d'establiments comercials, per les quals anirà passant el passatger mentre disposi de temps lliure entre cada bloc principal del procés de sortides.

Aquests components permetran al passatger prendre decisions del tipus: si pot continuar consumint temps d'oci mitjançant el desplaçament dins una ruta comercial o bé ha de marxar cap al següent bloc del procés de sortides de passatgers. Aquesta decisió depèn del temps màxim que li quedi per anar fins al proper bloc.

A més, es podrà simular el comportament estocàstic del desplaçament lliure de passatgers creant varies rutes d'establiments comercials, i assignant a cadascuna d'elles una probabilitat de ser visitades pels passatgers durant el seu temps d'oci.

Aquests components són:

ShopsTwo

Aquest component modelitza el desplaçament lliure dels passatgers dins d'una ruta de dos establiments comercials. La ubicació i mida de cada establiment comercial de la ruta es passa com a paràmetre d'entrada al component, ja que hi ha una forta dependència d'aquest component amb l'aeroport que s'estigui modelitzant.

A més, cada cop que el passatger es situa davant l'entrada de cadascun dels dos establiments comercials, fa una consulta, mitjançant una crida a un mètode de la classe Passenger (que és una classe Java que s'ha creat per modelitzar al passatger), dels temps màxim de que disposa per anar fins al següent bloc del procés de sortides.

En cas que no disposi de més temps per continuar la ruta, sortirà d'ella i es dirigirà cap el següent bloc del procés de sortides que li correspongui segons l'estat en que es trobi, que pot ser: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament.

A més, abans d'entrar a cada establiment comercial fa una consulta de l'aforament màxim de l'establiment, que es parametritzable, i en cas d'estar ple, es dirigeix cap al següent establiment comercial o, en cas de ser l'últim, surt de la ruta.

ShopsThree

Aquest component modelitza el desplaçament lliure dels passatgers dins d'una ruta de tres establiments comercials. Igual que al component *ShopsTwo* i per la mateixa raó, la ubicació i mida de cada establiment comercial de la ruta es passa com a paràmetre d'entrada al component.

El comportament que té el passatger dins d'aquesta ruta és el mateix que el del component *ShopsTwo*.

ShopsFive

Aquest component modelitza el desplaçament lliure dels passatgers dins d'una ruta de cinc establiments comercials. Igual que als components *ShopsTwo* i *ShopsThree*, la ubicació i mida de cada establiment comercial de la ruta es passa com a paràmetre d'entrada al component.

El comportament que té el passatger dins d'aquesta ruta és el mateix que el dels dos components anteriors.

Aquests tres components es poden anar enllaçant per formar rutes més llargues. A la figura 4.2 es mostra un exemple de com es podrien crear diferents rutes amb 6 establiments.

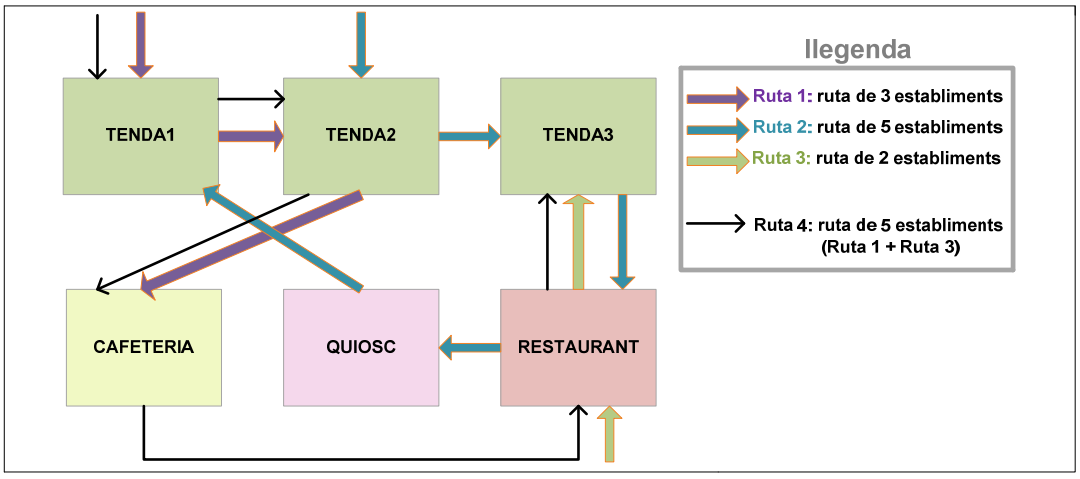


Figura 4.2: exemple de creació de rutes de tendes

5. Llibreria de components

En aquest capítol s'explicarà la llibreria de components que s'ha creat, amb la finalitat de complir amb un doble objectiu: per una banda explicar el treball realitzat en aquest projecte, i per l'altra banda aportar un document que faciliti la comprensió i la possible modificació i/o ampliació futura de la llibreria per part dels usuaris de l'Anylogic.

La llibreria de components conté els elements bàsics que intervenen en el procés de sortides de passatgers de qualsevol aeroport.

La idea clau en la que es basa la llibreria és la separació entre les característiques particulars de cada aeroport, com pot ser la ubicació de cada element que intervé en el procés de sortides de passatgers i la seva interrelació amb els altres elements, i la lògica interna que segueix cadascun d'aquests elements, que és comuna a tots els aeroports.

Per exemple, dins la lògica interna d'un taulell de facturació, el passatger arriba al taulell, es posa a la cua en cas que n'hi hagi, i quan li arriba el seu torn s'acosta al mostrador on factura l'equipatge i recull la seva targeta d'embarcament. Aquest comportament és comú als taulells de facturació de tots els aeroports, però depenent de les característiques particulars de l'aeroport que s'estigui modelitzant, aquest mostrador tindrà una forma, ubicació i relació amb els altres elements determinada.

D'aquesta manera s'ha aconseguit crear una llibreria de components totalment reutilitzable que permet la modelització del procés de sortides de passatgers de qualsevol aeroport.

5.1. Classes Entitat i components comuns

5.1.1. Passenger

A la llibreria Pedestrian es troba la classe Ped. Aquesta classe que proveeix l'Anylogic modelitza el comportament d'un vianant segons les regles físiques que formen l'entorn que es modeli.

Heretant d'aquesta classe Ped, s'ha creat la classe entitat Passenger. Aquesta classe Java conté els atributs i mètodes necessaris per a modelitzar el comportament conjunt d'un grup de passatgers, d'1 a 5 membres, des de la seva arribada a l'aeroport fins a l'embarcament.

Les consultes i decisions més importants que realitza cada objecte Passenger fent ús dels seus atributs i mètodes són les següents:

- Un cop surt del bloc d'arribades, ha de consultar si ha de facturar o ja la porta feta, per tal de decidir el següent bloc del procés de sortides cap el que haurà de dirigir-se.

- Dins del bloc de temps d'oci, decideix si te temps per continuar comprant o ha d'anar cap al següent bloc del procés de sortides de passatgers. Aquesta decisió ve condicionada, per una banda, per la consulta de l'estat en que es trobi l'objecte Passenger, que pot ser: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament, i, per altra banda, pel temps màxim de que disposa per anar fins al proper bloc del procés.

- Dins del bloc de facturació, pot realitzar les següents consultes i decisions:

  - En cas de tenir assignat un rang de mostradors per realitzar la facturació, decideix a quin mostrador ha de dirigir-se per facturar; decisió que ve condicionada per la consulta de disponibilitat dels mostradors que s'han assignat al seu vol.

  - En cas que tingui assignat un grup de multifacturació per a realitzar la facturació, ha de consultar quins mostradors del grup estan disponibles, per decidir en quin facturarà.

  - Un cop s'ha situat davant del mostrador de facturació, decideix quant de temps ha de romandre facturant, en funció del tipus de vol, Schengen o No Schengen, i del nombre de components del grup familiar que representa.

- Dins del bloc de pas pels filtres de seguretat, ha de consultar quins filtres, de la zona de filtres cap a la que s'ha dirigit, estan disponibles, per tal de decidir a quin filtre a d'anar.

A l'Annex 1 es troba la documentació (Javadoc) d'aquesta classe.

5.1.2. Flight

La classe entitat Flight és una classe Java que inclou els atributs i mètodes necessaris per a la modelització d'un vol. Durant el procés d'inicialització de la simulació (veure apartat 5.1.3. *Components Comuns*) es llegeix de l'arxiu extern de dades del model el llistat de vols programats en el dia que es vol simular i es crea un objecte Flight per a cada vol, amb els seus atributs.

Després, durant la injecció de passatgers a la simulació, realitzada pel component *Font de passatgers,* es relaciona cada objecte Passenger amb un objecte Flight. D'aquesta manera, durant tota la simulació es tenen disponibles totes les dades del vol que són necessàries per a que cada objecte Passenger es vagi movent pels diferents blocs del procés de sortides.

A l'Annex 1 es troba la documentació (Javadoc) d'aquesta classe.

5.1.3. Components comuns

En aquest apartat s'explicaran en detall els processos d'inicialització que s'han de realitzar al crear un model concret d'aeroport fent ús de la llibreria de components que s'ha creat en aquest projecte, per tal que serveixi als usuaris de la llibreria com a complement a l'explicació que es farà de cada component en aquest mateix capítol.

En primer lloc, cal recordar que l'objecte Main, que és la classe principal que s'executa a la simulació, contindrà dins el seu espai de disseny tant l'entorn físic, en forma de plànol, com la lògica interna del procés de sortides de l'aeroport que es vulgui modelitzar. A més, dins el seu paràmetre *Startup Code* s'ha d'afegir tot el codi que calgui executar abans d'iniciar la simulació.

Els processos d'inicialització necessaris per poder fer ús de la llibreria de components es divideixen en els quatre blocs següents:

- Connexió a la base de dades.
- Càrrega de dades al simulador.
- Inicialització de les taules d'arribades de cada font.
- Creació de llistats de recursos.

Connexió a la base de dades

Aquest procés permet la connexió amb un fitxer extern de dades que conté la programació de vols d'un dia de l'aeroport que es vulgui modelitzar, i altres dades d'interès per la simulació com: per exemple, les taules d'arribades de passatgers, les taules de disponibilitat de mostradors i portes d'embarcament i una pàgina de configuració que conté el número de fonts de passatgers i els percentatges de passatgers que arribaran per cada font.

Dins l'editor gràfic de l'objecte Main del model cal tenir un objecte de la classe Database, pertanyent al package com.xj.anylogic.engine.connectivity de l'AnyLogic, mostrat a la figura 5.1, que s'encarregarà d'establir una connexió amb el fitxer excel que conté les dades del model.

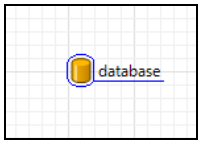


Figura 5.1: Objecte de la classe Database de l'AnyLogic.

La configuració de l'objecte database és la següent:

Type:	Excel/acces
File:	ruta on estigui el fitxer de dades extern

La resta de paràmetres mantenen els valors per defecte.

Fent una crida al mètode *getResultset()* d'aquest objecte, es poden fer consultes sql sobre les dades guardades al fitxer.

El contingut del fitxer de dades està estructurat en 12 fulls de càlcul que es detallen a continuació:

- *Paso1:* conté una relació de vols i els seus atributs principals que estan programats per un determinat dia.

- *Paso2:* conté la relació de vols ordenats per franja horària en la que sortirà el vol.

- *Paso 3:* conté la corba de presentació de passatgers.

- *DISTRIBUCIONES:* conté els percentatges usats per calcular la corba de presentació de passatgers segons la distribució triada al Paso2.

- *IATA:* conté la taula d'equivalències de les abreviatures usades en els fulls Paso1 i Paso2.

- *Paso3a:* conté la taula d'arribades previstes de passatgers desglossades per vols, des de la franja horària 1, que es correspon amb el rang horari [00:00 h – 00:05 h), fins la franja horària 200.

- Paso3b: conté la taula d'arribades previstes de passatgers desglossades per vols, des de la franja horària 201 fins la franja horària 288, que es correspon amb el rang horari [23:55 h – 00:00 h).

- *Setup:* conté dades de configuració de la simulació com són el número de fonts de passatgers que conté el model i el percentatge de passatgers que s'estima que arribarà per cada font.

- *DispMostradoresA:* conté la taula de disponibilitat dels mostradors de facturació desglossada per franges horàries, des de la franja horària 1 fins la franja horària 200.

- *DispMostradoresB:* conté la taula de disponibilitat dels mostradors de facturació desglossada per franges horàries, des de la franja horària 201 fins la franja horària 288.

- *DispFiltresA:* conté la taula de disponibilitat dels filtres de seguretat desglossada per franges horàries, des de la franja horària 1 fins la franja horària 200.

- *DispFiltresB:* conté la taula de disponibilitat dels filtres de seguretat desglossada per franges horàries, des de la franja horària 201 fins la franja horària 288.

Càrrega de dades al simulador

Les dades amb les que haurà de treballar el simulador estan emmagatzemades dins del fitxer extern de dades del model, que, com ja s'ha explicat a l'apartat

anterior, conté la programació de vols per un dia concret i altres dades d'interès per la creació d'un model d'aeroport concret.

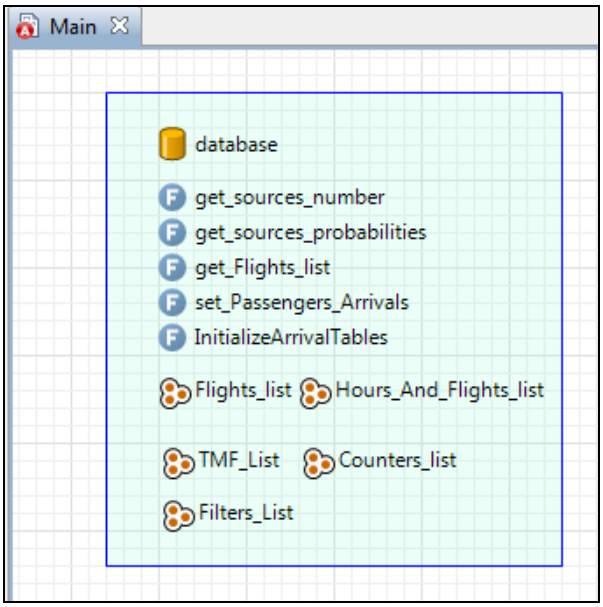


Figura 5.2: Funcions i llistes definides dins l'objecte Main del model

Per a realitzar la càrrega de dades al simulador, l'objecte Main del model fa una crida a les següents funcions que s'han definit dins del seu editor gràfic, tal i com es mostra a la figura 5.2:

***public int get_sources_number()* :**

Funció que llegeix del full de càlcul *Setup,* el número de components *Fonts de Passatgers* que ha de tenir el model i el guarda dins una variable entera.

Els paràmetres d'aquesta funció són:

- *database*: objecte de la classe Database de l'Anylogic que conté la connexió amb el fitxer de dades extern.

***public float[] get_sources_probabilities ()* :**

Funció que llegeix del full de càlcul *Setup,* les probabilitats de que arribin passatgers per cadascun dels components *Fonts de Passatgers* que ha de tenir el model i les guarda dins un array de variables de tipus float.

Els paràmetres d'aquesta funció són:

- *database*: objecte de la classe Database de l'Anylogic que conté la connexió amb el fitxer de dades extern.

***public ArrayList get_Flights_list()* :**

Funció que crea un objecte Flight per cada vol llegit del fitxer de dades extern, i el guarda dins una llista anomenada Flights_list, que prèviament s'ha definit

dins l'espai de disseny de l'objecte Main del model que s'estigui desenvolupant, ja que pot ser usada per varis components de la llibreria.

A més, aquesta funció també elabora una relació de franges horàries i vols, de manera que per cada franja horària tenim guardada una llista dels identificadors dels vols que tenen arribades previstes de passatgers en aquella franja. Aquestes arribades es preveuen que es produiran durant les 28 franges immediatament anteriors a la franja horària de sortida del vol, també inclosa, pel que cada vol està relacionat amb 29 franges horàries.

La funció parteix de la idea de que un dia es divideix en 288 franges horàries de 5 minuts cadascuna, pel que la franja horària 1 comença a les 00:00:00 h i finalitza a les 00:04:59 h.

Aquesta relació de vols i franges es guarda dins una llista anomenada Hours_And_Flights_list, creada prèviament dins l'espai de disseny de l'objecte Main del model que s'estigui desenvolupant, on cada posició de la llista es correspon amb una franja horària, començant per l'índex 0 i finalitzant a l'índex 287.

Els paràmetres d'aquesta funció són:

- *database*: objecte de la classe Database de l'Anylogic que conté la connexió amb el fitxer de dades extern.
- *Flights_list*: objecte de la classe ArrayList on es guardarà el llistat d'objectes Flight.
- *Hours_And_Flights_list*: objecte de la classe ArrayList on es guardarà la relació de vols i franges horàries.

public void set_Passengers_Arrivals():

Funció que llegeix les taules d'arribades de passatgers dels fulls *Paso3a* i *Paso3b* del fitxer de dades extern per omplir la matriu d'arribades, atribut *groups*, de cada objecte Flight.

Els paràmetres d'aquesta funció són:

- *database*: objecte de la classe Database de l'Anylogic que conté la connexió amb el fitxer de dades extern.
- *Flights_list*: objecte de la classe ArrayList que conté el llistat d'objectes Flight.

Inicialització de les taules d'arribades de cada font

Cada component *Font de Passatgers* que usi el model que es vulgui implementar necessita una taula d'arribades, objecte de la classe TableFuncition pertanyent al package com.xj.anylogic.engine de l'Anylogic

mostrat a la figura 5.3, des d'on llegir el nombre d'objectes Passenger que s'han d'injectar a la simulació a cada franja horària definida a la taula.

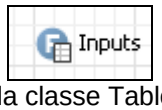


Figura 5.3: Objecte de la classe TableFunction de l'Anylogic.

Durant els processos d'inicialització, s'omplen els objectes TableFunction que s'hagin creat dins l'editor gràfic de l'objecte Main del model, fent ús de la funció *public void InitializeArrivalTables*(), que extreu les dades d'arribades de passatgers del fitxer de dades extern i les carrega a les taules.

Aquesta funció usa els següents paràmetres:

- *Flights_list*: objecte de la classe ArrayList que conté una llista d'objectes vols programats.
- *Hours_And_Flights_list*: objecte de la classe ArrayList que conté una relació de franges horàries i identificadors de vols que tenen arribades previstes de passatgers en cadascuna de les franges.
- *num_source*: variable de tipus enter que conté el número de font d'arribada de passatgers. El rang de valors d'aquest paràmetre és [0, nombre total de fonts – 1].
- *table*: objecte de la classe *TableFunction*, del package com.xj.anylogic.engine de l'aplicació AnyLogic, que conté una taula d'arribades inicialment buida.

Dins d'aquesta funció es recorren les llistes Flights_list i Hours_And_Flights_list per calcular la taula d'arribades de passatgers, en forma de grups, que pertanyi al component *Font de Passatgers* passat com a paràmetre d'entrada.

Per tant, dins l'espai de disseny de l'objecte *Main*, es definiran tants objectes de la classe *TableFunction* com components *Font de Passatgers* tingui l'aeroport que s'estigui modelitzant.

Creació de llistats de recursos

Dins del procés d'inicialització de la simulació, cal crear dos llistats que guardin, respectivament, els components *Taulell de Facturació* i *Taulell de MultiFacturació* que s'usin en el procés de sortides de l'aeroport que s'estigui modelitzant. Per exemple, a la figura 5.2 es mostren els objectes Counters_list i TMF_List, que llisten els components *Taulell de Facturació* i *Taulell de MultiFacturació,* respectivament.

També cal crear un llistat que guardi els recursos *Filtres de Seguretat* de que disposa la zona de filtres de seguretat de l'aeroport que s'estigui modelitzant. A la figura 5.2 es pot veure un exemple d'aquest llistat anomenat Filters_List.

Aquests llistats ha de ser objectes de la classe ActiveObjectCollection, pertanyent al package com.xj.anylogic.engine de l'Anylogic, i s'han d'ubicar dins l'editor gràfic de l'objecte Main del model, per tal que siguin accessibles des de tots els components de la llibreria i des d'altres objectes de l'Anylogic.

Fent una crida al mètode *add(Object o)* d'aquests objectes es poden anar afegint tots els components al llistat.

El llistat que guarda els components *Taulell de Facturació* servirà per recórrer, dins del mètode GoToCounter() de la classe Passenger, tots els components d'aquest tipus que s'hagin creat al model, amb l'objectiu de trobar el número de mostrador al que s'haurà de dirigir l'objecte Passenger que hagi fet aquesta consulta, en funció de la disponibilitat dels mostradors a cada franja horària.

El llistat que guarda els components Taulell de MultiFacturació servirà per recórrer, dins del mètode availabilityTMF() de la classe Passenger, tots els components d'aquest tipus que s'hagin creat al model, amb l'objectiu d'actualitzar la seva disponibilitat mitjançant la lectura de les taules de disponibilitat de mostradors que conté el fitxer d'entrada de dades del model.

De forma anàloga, el llistat que guarda els components *Filtres de Seguretat* servirà per recórrer, dins del mètode availabilityFilters() de la classe Passenger, tots els components d'aquest tipus que s'hagin creat al model, amb l'objectiu d'actualitzar la seva disponibilitat mitjançant la lectura de les taules de disponibilitat de filtres de seguretat que conté el fitxer d'entrada de dades del model.

5.2. Arribada de passatgers a la terminal

En aquest apartat s'explicaran els components que s'han creat a la llibreria per modelitzar el primer bloc del procés de sortides de passatgers de l'aeroport: l'arribada de passatgers a la terminal.

5.2.1. Font de passatgers

Figura 5.4: Component Font de passatgers (FP)

Aquest component va creant objectes de la classe Passenger segons una taula d'arribades passada com a paràmetre. En el procés d'inicialització de la simulació (veure apartat 5.1.3. *Components Comuns*) s'omplen les taules d'arribades de cada font.

Com es mostra a la figura 5.4, aquest component disposa d'un únic port de sortida.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterNumFont*: enter que indica quin número d'ordre li correspon a la font dins del conjunt de fons que tingui el model concret d'aeroport que s'estigui modelitzant. La numeració comença pel 0.

- *parameterArrivalTable*: taula d'arribades que usa la font per generar objectes de la classe Passenger.

  La taula d'arribades és un objecte de la classe TableFunction de l'AnyLogic, que defineix funcions a partir d'un conjunt de pars (argument, valor). En aquest cas, l'argument de la funció conté la relació d'unitats de temps en les que es produiran les injeccions de passatgers, i el valor de la funció conté la relació de quantitats de passatgers que es crearan a cada unitat de temps.

- *parameterAppearsAtLine*: objecte de la classe ShapeLine de L'AnyLogic que representa el dibuix d'una línia per on apareixeran els passatgers creats per la font. Tot i que amb el component *Font de passatgers* es crea però es no visualitzen els passatgers a la simulació, aquest paràmetre és necessari per a que la configuració de l'objecte *source* de la lògica interna del component, del que es parlarà més endavant, pugui funcionar correctament.

- *Hours_And_Flights_list:* objecte de la classe ArrayList que conté una relació dels vols implicats en la generació d'arribades de passatgers a cada franja horària.

- *Flights_list*: objecte de la classe ArrayList que conté la llista vols, objectes de la classe Fligh, que intervindran en la simulació del procés de sortida de passatgers d'un determinat dia.

Lògica Interna

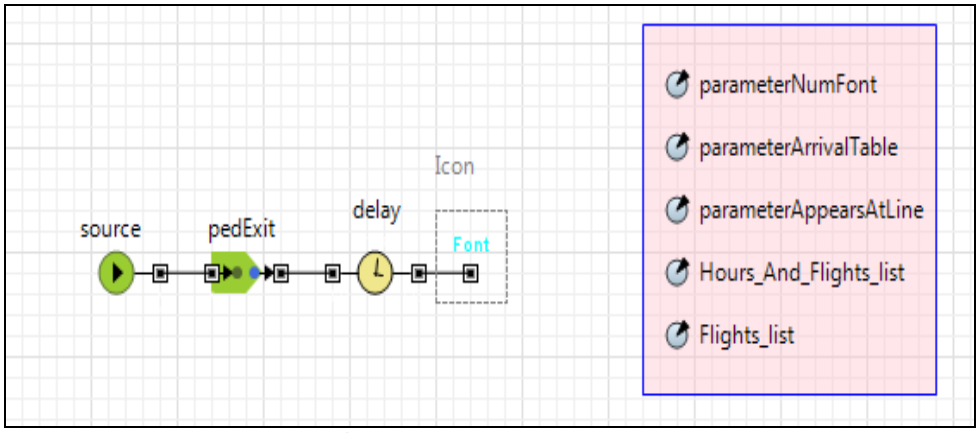


Figura 5.5: Lògica interna de la Font de passatgers

Tal i com es mostra a la figura 5.5, la lògica interna d'aquest component està formada per 3 objectes:

a) source

Objecte de la classe PedSource. Aquesta classe, pertanyent a la llibreria Pedestrian de l'Anylogic, permet crear objectes de la classe Ped (pedestrian), o subclasses d'aquesta, durant la simulació.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Creation mode:	Ped arrivals
Ped arrivals defined by:	Arrival table
Arrival table:	paramaterArrivalTable
New Ped:	new passenger();
Appears At:	parameterAppearsAtLine
On Exit:	*codi per relacionar l'objecte Passenger amb el vol i i el tipus d'agrupació familiar que representa mitjançant el color

*

```
 ArrayList gv = ped.Calculate_Family_Group(Hours_And_Flights_list,
Flights_list,source.getArrivalCount(),parameterNumFont);
int color = ((Integer)gv.get(0)).intValue();
Flight v = (Flight)gv.get(1);
ped.setFlight(v);
ped.setCol(color);
```

La resta de paràmetres mantenen els valors per defecte.

b) *pedExit*

Objecte de la classe PedExit. Aquesta classe de la llibreria Pedestrian de l'Anylogic, entre d'altres coses, permet connectar amb objectes d'altres llibreries. En aquest cas s'usa per connectar l'objecte *Source* amb l'objecte *delay*, que pertany a la llibreria Enterprise de l'Anylogic.

La configuració d'aquest objecte és la següent:

Ped class: Passenger

La resta de paràmetres mantenen els valors per defecte.

c) *delay*

L'element *source* injecta de cop un número determinat d'objectes Passenger a cada unitat de temps, segons la taula d'arribades que se li passa com a paràmetre. Però, en la realitat, les arribades de passatgers no es produeixen a tongades, pel que cal aplicar un decalatge.

L'objecte de la classe Delay, pertanyent a la llibreria Enterprise de l'Anylogic, simula aquest decalatge fent esperar entre 0 a 5 minuts (valor generat amb la

funció de distribució uniforme) a cada objecte Passenger abans d'enviar-se al següent component de la llibreria: *Porta d'entrada a la terminal.*

La configuració d'aquest objecte és la següent:

Delay time is:	Specified explicitly
Delay time:	uniform(0.0 * minute(), 5.0 * minute());

La resta de paràmetres mantenen els valors per defecte.

5.2.2. Porta d'entrada a la terminal

Figura 5.6: Porta d'entrada a la terminal (PEntTerminal)

Aquest component modelitza la lògica d'una entrada física a l'aeroport que s'estigui modelitzant; per exemple, una porta a peu de carrer, o unes escales mecàniques que pugin del pàrquing.

Un cop el component *Font de Passatgers* ha creat els objectes Passenger, mitjançant aquest nou component es fan aparèixer per l'entrada que s'hagi dibuixat al model amb una velocitat i un diàmetre específics.

Aquest component també permet canviar l'entorn físic, entès com el conjunt de parets que no poden ser travessades, i les velocitats amb les que es mouran els passatgers en un punt concret del plànol de l'aeroport que s'estigui modelitzant.

Tal i com es mostra a la figura 5.6, aquest component disposa d'un port d'entrada, situat a l'esquerra del component, i d'un port de sortida, situat a la dreta del component.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterLiniaEntrada:* objecte de la classe ShapeLine, pertanyent al package com.xj.anylogic.engine.presentation de l'Anylogic, que representa el dibuix de la línia per on entraran els objectes Passenger, creats pel component *Font de Passatgers,* que es dirigeixin a aquesta entrada del model d'aeroport que s'estigui simulant.

- *parameterGround:* objecte de la classe PedGround, pertanyent a la llibreria Pedestrian de l'Anylogic, que representa els límits físics, definits com un grup de línies que representaran les parets, de l'entorn per on es mouran els passatgers durant la simulació.

- *parameterDiametre:* variable de tipus double que indica el diàmetre en metres que tindrà el dibuix d'un objecte Passenger.

- *parameterVelocitat: variable de tipus* double que indica la velocitat, en metres per segon, que portarà cada objecte Passenger durant la simulació. La velocitat a pas normal d'una persona està entre 0.8 i 1 m/s.

Lògica Interna

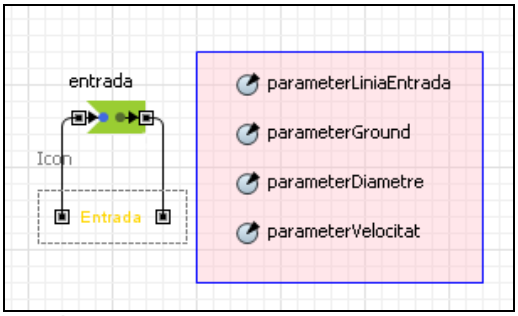


Figura 5.7: Lògica interna de la Porta d'entrada a la terminal

Tal i com es mostra a la figura 5.7, la lògica interna d'aquest component està formada per un objecte de la classe PedEnter de la llibreria Pedestrian de l'Anylogic. Aquest objecte permet fer canvis de velocitat i de medi físic en punts concrets de l'entorn de simulació.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Diameter, meters:	parameterDiametre
Comfortable speed:	parameterVelocitat
Arrival Ground:	parameterGround
Appears At:	line,polyline
Appears at:	parameterLiniaEntrada
On Enter:	*codi per colorejar els objectes Passenger segons els grup familiar que representin. La llegenda de colors és la següent:

Llegenda de colors

Black	Grup d'1 persona
Blue	Grup de 2 persones
Red	Grup de 3 persones
Yellow	Grup de 4 persones
Green	Grup de 5 persones

*

```
switch(ped.getCol()){
case 1: ped.setColor(BLACK);
break;
case 2: ped.setColor(BLUE);
break;
case 3: ped.setColor(RED);
break;
case 4: ped.setColor(YELLOW);
break;
case 5: ped.setColor(GREEN);
break;
default: ped.setColor(WHITE);
break;
}
```

La resta de paràmetres mantenen els valors per defecte.

5.3. Facturació

En aquest apartat s'explicaran els components que s'han creat a la llibreria per modelitzar el segon bloc del procés de sortides de passatgers de l'aeroport: la facturació.

5.3.1. Taulell de Facturació

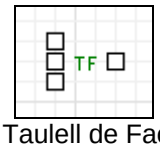


Figura 5.8: Taulell de Facturació (TF)

Aquest component modelitza la lògica d'un taulell de facturació format per tres elements: un mostrador on s'atén al passatger durant un temps determinat, una cua, amb política FIFO, on romandran els passatgers que han de facturar a aquest mostrador mentre aquest estigui ocupat, i una cinta transportadora de maletes.

El temps d'atenció davant el mostrador de facturació depèn dels següents factors:

- El tipus de vol que ha d'agafar el passatger: Schengen o No Schengen[3], definit com un atribut de l'objecte Flight relacionat amb cada objecte Passenger.

- El color de l'objecte Passenger, que indica el tipus de grup familiar que representa.

La classe entitat Passenger disposa d'un mètode per calcular el temps d'atenció segons aquests dos paràmetres. La signatura d'aquest mètode és la següent:

public double timeWaitCheking (int color, boolean schengen)

A la figura 5.9 es mostra la relació de temps d'atenció al mostrador i número de components del grup familiar que hi ha definida a la funció *timeWaitCheking()* pels vols Schengen i pels vols No Schengen:

[3] L' **Acord de Schengen** constitueix un dels avenços més importants en la història de la construcció de la Unió Europea (UE). L' acord té com a objectiu finalitzar amb els controls fronterers dintre de l' **espai de Schengen** —format per la majoria dels Estats membres de la Unió i alguns tercers països— i harmonitzar els controls fronterers externs. Els països que apliquen en la seva totalitat l' acord de Schengen constitueixen un territori denominat **espai Schengen**.

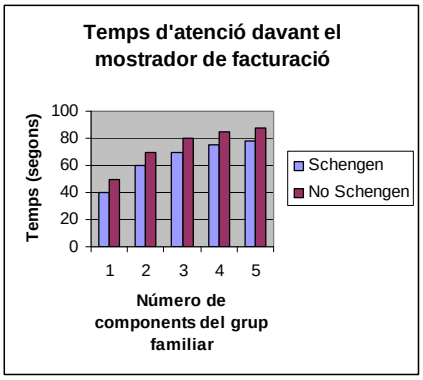


Figura 5.9: Gràfica de temps d'atenció al mostrador de facturació

Aquests temps d'atenció són orientatius, pel que en cas de voler validar un model concret d'aeroport, caldrà revisar aquests valors segons els temps mitjos d'espera que es produeixin als mostradors de facturació, en funció dels grups de persones que s'atenguin i del tipus de vol, Schengen o No Schengen.

La facturació de maletes es defineix com la injecció d'un número determinat de maletes dins la cinta transportadora. Dins la lògica d'aquest component, aquest número està definit per una funció de distribució uniforme discreta limitada pel rang [num_min_maletes, num_max_maletes]. El color de l'objecte Passenger que arriba al mostrador determina el rang usat per fer la crida a la funció de distribució. D'aquesta manera es manté la coherència entre el tipus de grup que representa l'objecte Passenger i el número de maletes facturades.

La disponibilitat del mostrador està definida fora de la lògica d'aquest component. Un mostrador es considera disponible en els següents supòsits:

- A la taules de disponibilitat de mostradors apareix com a disponible, valor 1, per a la franja horària en la que s'està realitzant la consulta.

- A les taules de disponibilitat de mostradors apareix com a No disponible, valor 0, per a la franja horària en la que s'està realitzant la consulta, però la cua davant del mostrador encara no està buida.

La classe Passenger disposa del mètode amb signatura: *public int GoToCounter(int counterF, int counter L, int minute, Database database, ArrayList Counters_List,)* , que fa ús del fitxer d'entrada del model per calcular el número de mostrador al que ha d'anar el passatger a facturar, entre un rang inicial de mostradors assignats al vol, segons la franja horària actual de la simulació. A més, aquesta funció calcula d'entre els mostradors que té disponibles l'objecte Passenger, el que te la cua més curta.

Per tant, tots els components Taulell de Facturació que es facin servir en un model concret d'aeroport sempre estaran disponibles i caldrà cridar a la funció *GoToCounter()* per dirigir l'objecte Passenger cap a un mostrador concret.

Tal i com es mostra a la figura 5.8, aquest component disposa d'un port d'entrada, situat a la seva part esquerra, i d'un port de sortida, situat a la seva part dreta. També té un port d'entrada i un port de sortida per la lògica de la

cinta transportadora de maletes, que es tan situats, respectivament, per sobre i per sota del port d'entrada del component.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterPunt_Atencio:* objecte de la classe ShapeLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa el dibuix de la línia on s'esperaran els objectes Passenger davant el mostrador de facturació.

- *parameterCua_Facturacio:* objecte de la classe ShapePolyLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que guarda el dibuix de la línia que representa la cua que faran els passatgers mentre s'esperen a que arribi el seu torn.

- *parameterCintaM_E:* objecte de la classe ShapePolyLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa una part de la cinta transportadora de maletes.

- *parameterCintaM_S:* objecte de la classe ShapePolyLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa una part de la cinta transportadora de maletes.

Lògica interna

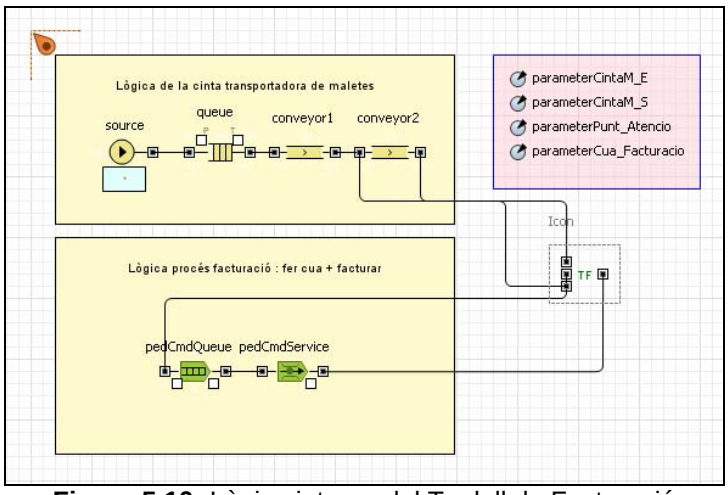


Figura 5.10: Lògica interna del Taulell de Facturació

Tal i com es mostra a la figura 5.10, la lògica d'aquest component està dividida en dos blocs:

1. Lògica de la cinta transportadora de maletes

La cinta transportadora de maletes s'ha modelitzat en dos trams. En el primer tram de cinta es modelitza la facturació d'equipatge que es realitza a cada

taulell. En el segon tram es modelitza el transport de l'equipatge facturat a cada mostrador fins la zona de tractament de maletes (SATE).

Aquest bloc està format per 4 objectes de la llibreria Enterprise:

a) source

Instància de la classe Source que modelitza una font d'objectes Entity.

La configuració d'aquest objecte és la següent:

Entity class:	Entity
Arrivals defined by:	Manual (call inject() method)
New entity:	new Entity();
Entity animation shape:	bag // objecte de la classe ShapeRectangle, del package com.xj.anylogic.engine.presentation de l'Anylogic, que dibuixa una forma estàndard de l'equipatge

La resta de paràmetres mantenen els valors per defecte.

b) queue

Instància de la classe Queue que modelitza la lògica d'una cua. És necessari posar una cua davant d'objectes de la classe Conveyor per adaptar la velocitat d'entrada d'instàncies Entity al conveyor i la velocitat amb la que aquest objecte les pot assimilar.

La configuració d'aquest objecte és la següent:

Entity class:	Entity
Maximum capacity:	Sí

La resta de paràmetres mantenen els valors per defecte.

c) conveyor1

Instància de la classe Conveyor que modelitza la primera part de la cinta transportadora de maletes. La configuració d'aquest objecte és la següent:

Entity class:	Entity
Length is:	Specified explicitly
Length:	10
Space between entities:	5
Speed:	75
Accumulating:	Sí
Animation guide shape:	parameterCintaM_E
Animation direction:	Backward

La resta de paràmetres mantenen els valors per defecte.

d) conveyor2

Instància de la classe Conveyor que modelitza la segona part de la cinta transportadora de maletes. Aquest element permet enllaçar les cintes transportadores d'un grup de mostradors, formant així una illa de facturació.

Per tal que no s'acumuli l'equipatge a l'objecte conveyor2 de l'últim mostrador del grup, s'haurà de connectar el port de sortida de maletes d'aquest últim component del tipus *Taulell de facturació* a una instància de la classe Sink, pertanyent a la llibreria Enterprise, que anirà eliminant del sistema totes les instàncies Entity, que representen l'equipatge facturat, que li vagin arribant.

La configuració d'aquest objecte és la següent:

Entity class:	Entity
Length is:	Specified explicitly
Length:	10
Space between entities:	2
Speed:	75
Accumulating:	Sí
Animation guide shape:	parameterCintaM_S
Animation direction:	Forward

La resta de paràmetres mantenen els valors per defecte.

2. Lògica del procés de facturació

En aquest bloc es modelitza la lògica de la cua d'espera davant el mostrador de facturació i el temps d'atenció que rep el passatger al mostrador. Els elements que conformen aqueta lògica són els següents:

a) pedCmdQueue

Instància de la classe PedCmdQueue, pertanyent a la llibreria Pedestrian, que modelitza una cua d'espera, la forma de la qual es passa com a paràmetre d'entrada.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Type:	Line,polyline
Capacity:	Integer.MAX_VALUE;
Shape:	parameterCua_Facturacio

La resta de paràmetres mantenen els valors per defecte.

b) pedCmdService

Instància de la classe PedCmdService, pertanyent a la llibreria Pedestrian de l'Anylogic, que modelitza diferents tipus de serveis, com per exemple una venda de tiquets en una estació de tren, un pas per un torniquet, o, en el nostre cas, el servei de facturació d'un aeroport.

El dibuix de la línia de parada davant el mostrador de facturació es passa com a paràmetre d'entrada de l'objecte *Taulell de facturació.*

Per modelitzar aquest servei s'ha configurat l'objecte de la següent manera:

Ped class: Passenger
Type: Just delay
Shape: parameterPunt_Atencio
Depay time: ped.timeWaitCheking(ped.getCol(),ped.getFlight().getSchengen())*second();
// Funció de la classe Passenger que retorna el temps que l'objecte Passenger haurà d'esperar davant el mostrador de facturació
On begin service: *codi per injectar maletes a la cinta.

```
*  switch(ped.getCol()){
   case 1:this.source.inject(uniform_discr(0,1));
   break;
   case 2:this.source.inject(uniform_discr(1,2));
   break;
   case 3:this.source.inject(uniform_discr(1,3));
   break;
   case 4:this.source.inject(uniform_discr(2,3));
   break;
   case 5:this.source.inject(uniform_discr(2,4));
   break;
   default:
   }
```

La resta de paràmetres mantenen els valors per defecte.

5.3.2. Taulell de MultiFacturació

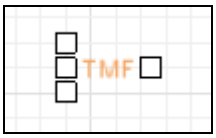


Figura 5.11: Taulell de MultiFacturació (TMF)

Aquest component modelitza un mostrador de facturació pertanyent a un grup de multifacturació.

Un grup de multifacturació està format per un *Serpentí* i dos o més components del tipus T*aulell de MultiFacturació*.

La disponibilitat d'aquest component es llegeix de les taules de disponibilitat de mostradors, per franges horàries, que conté l'arxiu de dades extern.

Dins del *Serpentí* del grup de multifacturació es crida al mètode availabilityTMF(), de la classe Passenger, per actualitzar la disponibilitat de cadascun dels components *Taulell de MultiFacturació* que el component *Serpentí* hi té connectats.

Tal i com es mostra a la figura 5.11, aquest component disposa d'un port d'entrada, situat a la seva part esquerra, i d'un port de sortida, situat a la seva

part dreta. També té un port d'entrada i un port de sortida per la lògica de la cinta transportadora de maletes, que es tan situats, respectivament, per sobre i per sota del port d'entrada del component.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterCintaM_E:* objecte de la classe ShapePolyLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa la part inicial de la cinta transportadora de maletes.

- *parameterCintaM_S:* objecte de la classe ShapePolyLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa la part final de la cinta transportadora de maletes.

- *parameterPunt_Atencio:* objecte de la classe ShapeLine del Package com.xj.anylogic.engine.presentation de l'Anylogic que representa el dibuix de la línia on s'esperaran els objectes Passenger davant el mostrador de facturació.

Lògica interna

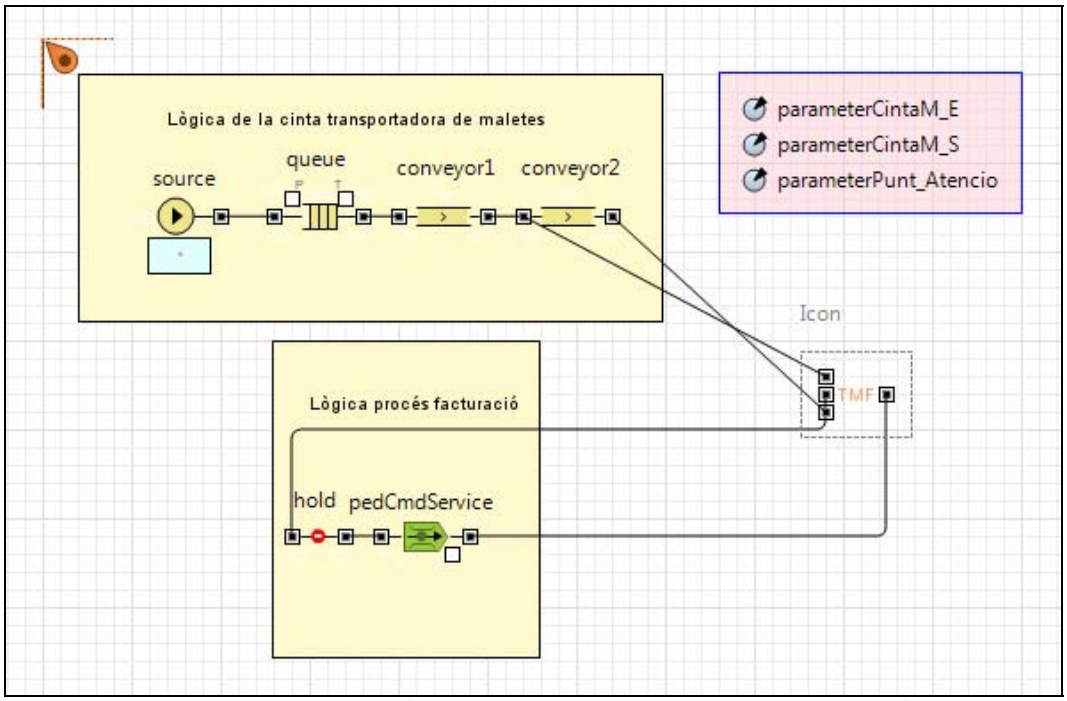


Figura 5.12: Lògica interna del Taulell de MultiFacturació

Tal i com es veu a la figura 5.12, la lògica interna és la mateixa que la del *Taulell de Facturació,* a excepció del següent:

a) L'objecte pedCmdQueue no s'usa, ja que el grup de multifacturació conté una cua compartida per tots els mostradors, gestionada a través del component extern *Serpentí.*

b) S'ha afegit l'objecte hold de la classe Hold, pertanyent a la llibreria Enterprise de l'Anylogic, que simula la disponibilitat del mostrador. La configuració d'aquest objecte manté tots els valors per defecte.

Aquest objecte disposa del mètode setBlocked() per bloquejar-lo i desbloquejar-lo. Quan l'objecte hold queda bloquejat, cap objecte Passenger pot entrar-hi, i d'aquesta manera es simula que el mostrador no està disponible.

Com ja s'ha comentat en aquest mateix apartat, el grup de multifacturació conté un component Serpentí que està connectat a tants components *Taulell de MultiFacturació* com mostradors d'aquest tipus es vulguin modelitzar dins d'aquest grup. Llavors, dins la lògica del component *Serpentí,* es bloqueja o desbloqueja l'objecte hold de cada component *Taulell de MultiFacturació*, en funció de les taules de disponibilitat de mostradors, per franges horàries, que conté el fitxer de dades extern que usa el model.

5.4. Serpentí

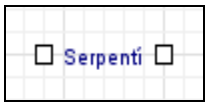


Figura 5.13: Serpentí

Aquest component modelitza la lògica interna d'una cua d'espera de persones amb política FIFO, que està delimitada per barreres que li donen la forma i que no es poden travessar; per exemple, suports de peu amb cintes. La forma de la cua es passa al component com a paràmetre d'entrada.

Com es veurà amb més detall a l'apartat de Lògica interna, per modelitzar aquest component ha estat necessari combinar objectes de dues llibreries, ja que tot i que l'Anylogic proveeix una classe PedQueue, de la llibreria Pedestrian, que modelitza cues de persones, el comportament d'aquest element no s'ajusta al que tenen els passatgers en el cas particular d'una cua dins un Serpentí. El motiu és que el comportament de la cua que modelitza L'Anylogic és tal que, els passatgers que hi arriben tenen la possibilitat d'anar cap al seu inici formant petites aglomeracions de gent que poc a poc es van posant en fila, en canvi, en un serpentí no pot haver-hi la possibilitat de que els passatgers arribin fins al cap de la cua i formin aglomeracions, pel que per modelitzar una cua de persones disposades en fila índia en tot el seu tram, i que a més tingui el contorn limitat físicament, ha calgut crear un component específic.

La cua d'espera que modelitza el *Serpentí* pot situar-se tant dins d'un grup de multifacturació com dins d'una zona de filtres de seguretat. Segons el seu ús, que és parametritzable, la seva lògica interna farà el tractament de la disponibilitat dels components *Taulell de MultiFacturació* o dels components *Filtre de Seguretat* que hi tingui connectats.Tal i com es mostra a la figura 5.13, aquest component disposa d'un port d'entrada, situat a la seva part esquerra, i d'un port de sortida, situat a la seva part dreta.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterCuaAbansSerpenti:* objecte de la classe ShapePolyLine del package com.xj.anylogic.engine.presentation de l'Anylogic que guarda el dibuix de la línia que representa la forma de la cua davant l'entrada del serpentí que hauran de fer els passatgers si aquest està ple.

- *ParameterLiniaSerpenti:* objecte de la classe ShapePolyLine de l'Anylogic que guarda el dibuix de la línia que representa la forma de la cua que faran els passatgers dins del serpentí.

- *ParameterLongitudSerpenti:* enter que indica la longitud que tindrà la cua interna del serpentí.

- *ParameterVelocitat:* enter que guarda la velocitat amb la que es desplaçaran els passatgers dins del serpentí. L'Anylogic no especifica les unitats amb les que treballa l'objecte Conveyor, pertanyent a la lògica interna d'aquest component, que fa ús d'aquest paràmetre.

- *parameterGround:* objecte de la classe PedGround, pertanyent a la llibreria Pedestrian de l'Anylogic, que representa els límits físics, definits com un grup de línies que representaran les parets de l'entorn per on es mouran els passatgers quan surtin del serpentí.

- *parameterAppearsAtLine:* objecte de la classe ShapeLine de l'Anylogic que representa la línia per on apareixeran a la simulació els objectes Passenger un cop surtin del Serpentí.

- *parameterDiameter:* double que indica el diàmetre en metres que tindrà el dibuix d'un objecte Passenger.

- *parameterRecursInicial:* enter que guarda el número del primer recurs, del tipus *Taulell de MultiFacturació* o *Filtre de Seguretat*, que té connectat el *Serpentí*.

- *parameterRecursFinal:* enter que guarda el número de l'últim recurs, del tipus *Taulell de MultiFacturació* o *Filtre de Seguretat*, que té connectat el *Serpentí*.

- *parameterDatabase:* objecte de la classe Database, pertanyent al package com.xj.anylogic.engine.connectivity de l'AnyLogic, que guarda la connexió amb el fitxer de dades extern que conté la taula de disponibilitats tant dels mostradors de facturació com dels filtres de seguretat que contingui el model.

- *parameterLlista_Recursos:* ArrayList que guarda la llista de recursos, del tipus *Taulell de MultiFacturació* o *Filtre de Seguretat*, que té connectat el

Serpentí i que haurà de recórrer dins la seva lògica interna per actualitzar la disponibilitat de cadascun d'ells.

- *Disponibilitat:* enter que pot prendre els valors 0 ó 1 per indicar quin tipus de components té connectats el *Serpentí*:

  - Si el valor és igual a 0: indica que els components són del tipus *Filtre de Seguretat.*

  - Si el valor és igual a 1: indica que els components són del tipus *Taulell de MultiFacturació.*

Lògica interna

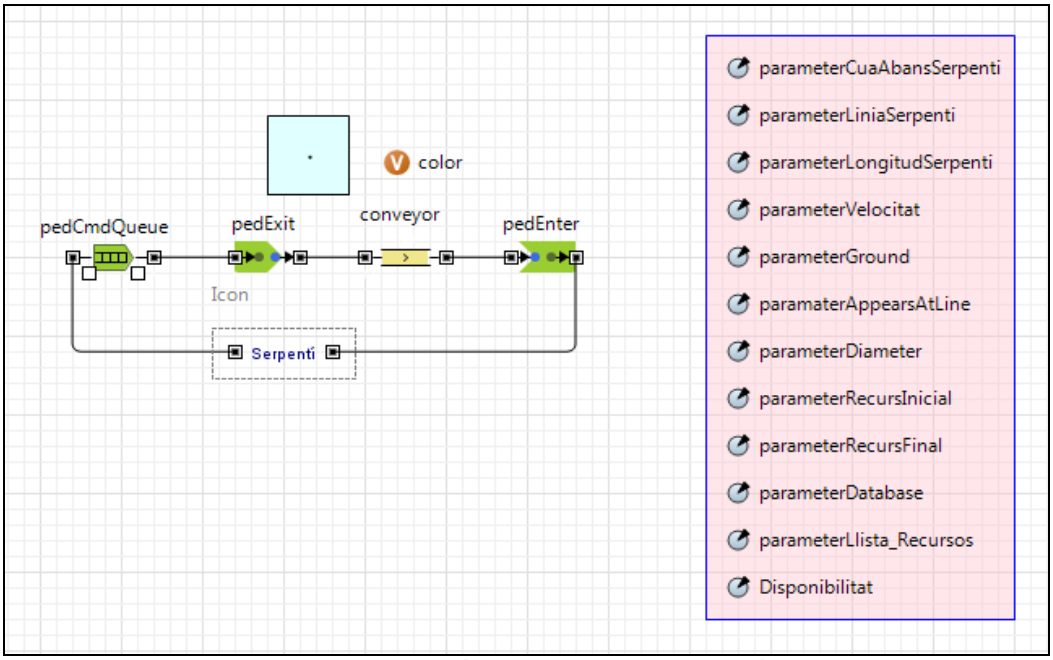


Figura 5.14: Lògica interna del Serpentí

Com es pot veure a la figura 5.14, dins la lògica interna del *Serpentí* es combinen objectes de classes pertanyents a la llibreria Pedestrian i Enterprise de l'Anylogic.

a) pedCmdQueue

Instància de la classe PedCmdQueue, pertanyent a la llibreria Pedestrian de l'Anylogic, que modelitza la cua d'espera que es forma davant l'entrada del Serpentí quan aquest està ple, la forma de la qual es passa com a paràmetre d'entrada.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Type:	Line,polyline
Capacity:	Integer.MAX_VALUE
Shape:	parameterCuaAbansSerpenti

La resta de paràmetres mantenen els valors per defecte.

b) pedExit

Objecte de la classe PedExit. Aquesta classe de la llibreria Pedestrian de l'Anylogic, entre d'altres coses, permet connectar amb objectes d'altres llibreries. En aquest cas s'usa per connectar l'objecte *pedCmdQueue* amb l'objecte conveyor, que pertany a la llibreria Enterprise de l'Anylogic.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
On exit environment:	color = ped.getColor(); (amb aquest codi actualitzem la variable amb el color de l'objecte Passenger que passa per aquest element ja que aquesta variable serà utilitzada en el següent element de la lògica interna).

La resta de paràmetres mantenen els valors per defecte.

c) conveyor

Instància de la classe Conveyor, pertanyent a la llibreria Enterprise de l'Anylogic, que modelitza una cua de passatgers dins d'un serpentí.

Aquest objecte fa ús de la variable *color* per colorejar els objectes Passenger segons el tipus de grup familiar que representen.

La configuració d'aquest objecte és la següent:

Entity class:	Entity
Length is:	Specified explicitly
Length:	parameterLongitudSerpenti
Space between entities:	1
Speed:	parameterVelocitat
Accumulating:	Sí
On Enter:	***** codi per dibuixar i colorejar cada objecte Passenger que entri al serpentí, ja que la forma i el color que te mentre passa pels elements de la llibreria Pedestrian es perd al entrar a l'objecte d'una altra llibreria.
Animation guide shape:	parameterLiniaSerpenti
Animation direction:	Backward

*****
```
ShapeCurve o = curve.clone();
o.setLineColor(color);
o.setLineWidth(0.5);
o.setFillColor(color);
entity.setShape(o);
```

La resta de paràmetres mantenen els valors per defecte.

d) pedEnter

Instància de la classe PedEnter, de la llibreria Pedestrian de l'Anylogic, que modelitza l'entrada de passatgers dins l'entorn en que es trobaran a la sortida

del serpentí. Tant l'entorn com la línia per on entren el passatgers a aquest nou entorn es passen a l'objecte mitjançant paràmetres d'entrada del component *Serpentí*.

El paràmetre *On Enter* d'aquest objecte conté el codi necessari per simular les disponibilitats dels recursos que tingui connectats el serpentí, ja siguin del tipus *Taulell de MultiFacturació* com del tipus *Filtre de Seguretat*, mitjançant la crida al mètode setBlocked() de l'objecte Hold de que disposen aquests dos tipus de recursos dins la seva lògica interna.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Diameter, meters:	parameterDiameter
Comfortable speed:	0.8
Arrival Ground:	parameterGround
Appears at:	line, polyline
Appears at:	parameterAppearsAtLine
Initial direction, radian:	West
On Enter:	*codi que s'executa cada cop que entra un objecte Passenger dins d'aquest objecte. Serveix per actualitzar les disponibilitats dels taulells de multifacturació o dels filtres de seguretat que tingui connectats el component *Serpentí*.

*

```
if(Disponibilitat==1){

boolean[] availability = ped.availabilityTMF(parameterRecursInicial,
parameterRecursFinal, (this.getHourOfDay()*60)+this.getMinute(),
parameterDatabase );

for (int i=0;i<=(parameterRecursFinal-parameterRecursInicial);i++){
((TMF)parameterLlista_Recursos.get(i)).hold.setBlocked(!availability
[i]);

}
}else{

boolean[] availability =
ped.availabilityFilters(parameterRecursInicial,
parameterRecursFinal, (this.getHourOfDay()*60)+this.getMinute(),
parameterDatabase );

for (int i=0;i<=(parameterRecursFinal-parameterRecursInicial);i++){
((Filtre)parameterLlista_Recursos.get(i)).hold_disponibilitat.setBlo
cked(!availability[i]);

}
```

La resta de paràmetres mantenen els valors per defecte.

5.5. Filtre de seguretat

Figura 5.15: Filtre

Aquest component modelitza una simplificació del procés de pas de passatgers per la zona de filtres, consistent en fer esperar a l'objecte Passenger en un punt dins la zona de filtres durant un temps determinat.

La durada del temps d'espera depèn del que triga el grup de passatgers, segons el color de l'objecte Passenger, en realitzar les següents accions:

- Dipositar d'objectes personals dins una safata sobre la cinta de l'scanner.
- Pas per l'arc de seguretat.
- Recollida de la safata a la sortida de la cinta de l'scanner.

Per modelitzar la zona de filtres de seguretat d'un determinat aeroport caldrà fer ús d'un component d'aquest tipus per cada arc de seguretat de que disposi aquesta zona.

La disponibilitat d'aquest component es llegeix de les taules de disponibilitat de filtres de seguretat, per franges horàries, que conté l'arxiu de dades extern.

Per modelitzar una zona de filtres de seguretat en la seva totalitat, caldrà usar tants components del tipus *Filtre de Seguretat* com filtres de seguretat tingui la zona a modelitzar i, a més, caldrà connectar-hi un o més components Serpentí que modelitzin les cues d'espera que es produeixen abans d'arribar als filtres de seguretat.

Dins la lògica interna de cada component *Serpentí* que s'usi per modelitzar la zona de filtres de seguretat, es gestiona la disponibilitat dels components *Filtres de Seguretat* que hi te connectats mitjançant la crida a la funció availabilityFilters() de la classe Passenger. Aquesta funció llegeix les taules de disponibilitat de filtres de seguretat que conté el fitxer de dades extern que usa el model.

Tal i com es mostra a la figura 5.15, aquest component disposa d'un port d'entrada, situat a la seva part esquerra, i d'un port de sortida, situat a la seva part dreta.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterLiniaArc:* objecte de la classe ShapeLine de la llibreria Pedestrian que guarda el dibuix de la línia on s'haurà d'esperar l'objecte Passenger dins la zona de filtres.

- *parameterTempsMin:* enter que representa el límit inferior de l'interval que delimita la funció de distribució que es fa servir per calcular el temps d'espera, dins la zona de filtres, d'un determinat objecte Passenger.

- *parameterTempsMax:* enter que representa el límit superior de l'interval que delimita la funció de distribució que es fa servir per calcular el temps d'espera, dins la zona de filtres, d'un determinat objecte Passenger.

Lògica interna

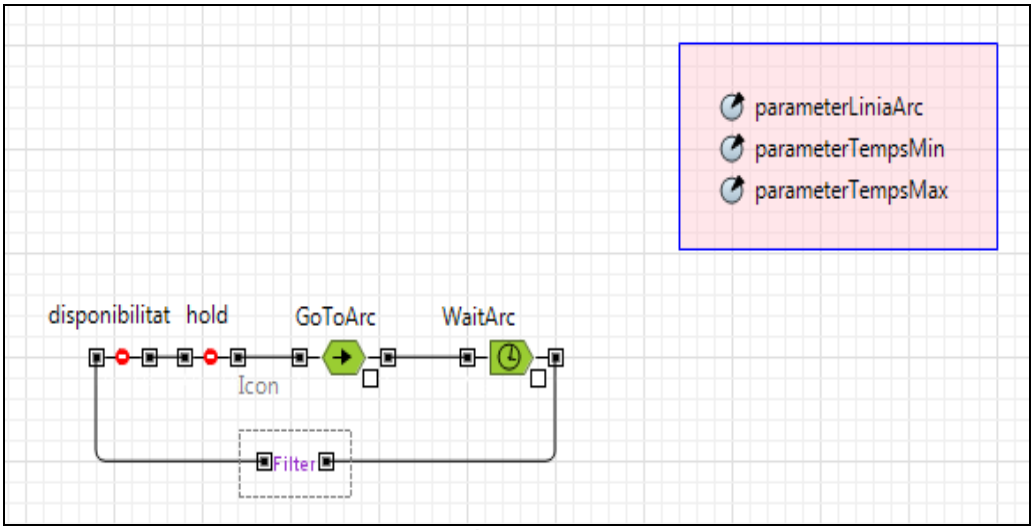


Figura 5.16: Lògica interna del Filtre

Tal i com es mostra a la figura 5.16, s'ha definit la lògica interna de manera que permeti ampliar el nivell de detall del procés de pas de passatgers per la zona de filtres que modelitza el component, sense haver de modificar els objectes que ja hi ha implementats.

Aquesta lògica està implementada amb els següents objectes de l'Anylogic:

a) disponibilitat

Instància de la classe Hold, pertanyent a la llibreria Enterprise de l'Anylogic, que simula la disponibilitat del filtre de seguretat. La configuració d'aquest objecte manté tots els valors per defecte.

Aquest objecte disposa del mètode setBlocked() per bloquejar-lo i desbloquejar-lo. Quan l'objecte hold queda bloquejat, cap objecte Passenger pot entrar-hi, i d'aquesta manera es simula que el filtre de seguretat no està disponible.

Dins la lògica del component *Serpentí,* es bloqueja o desbloqueja l'objecte *disponibilitat* de cada component *Filtre de Seguretat* que hi tingui connectat. Aquesta acció es realitza en funció de les taules de disponibilitat de filtres de seguretat per franges horàries que conté el fitxer de dades extern que usa el model.

b) hold

Instància de la classe Hold de la llibreria Enterprise de l'Anylogic que bloqueja el pas cap el filtre de seguretat modelitzat pel component mentre aquest estigui ocupat.

La configuració d'aquest objecte manté tots els valors per defecte.

c) GoToArc

Instància de la classe PedGoTo de la llibreria Pedestrian que envia els objectes Passenger cap a la zona d'espera del filtre, que es passa al component com un paràmetre d'entrada.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Target:	parameterLiniaArc
On enter:	Hold.setBlocked(true); //codi que bloqueja l'entrada al filtre mentre aquest estigui ocupat

La resta de paràmetres mantenen els valors per defecte.

d) WaitArc

Instància de la classe PedCmdWait de la llibreria Pedestrian que fa esperar als objectes Passenger en la localització actual en la que es trobin, durant un temps determinat per una funció de distribució uniforme contínua que genera valors dins de l'interval: [parameterTempsMin, parameterTempsMax].

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Type:	Depay
Delay time:	uniform (parameterTempsMin*second(), parameterTempsMax*second());

On exit:	hold.setBlocked(false); //codi per desbloquejar el filtre un cop queda desocupat.
	Ped.setState(2); //codi per actualitzar l'estat de l'objecte Passenger.

La resta de paràmetres mantenen els valors per defecte.

5.6. Temps d'oci

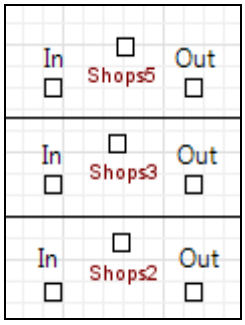


Figura 5.17: Component ShopsTwo, ShopsThree i ShopsFive.

Per tal de modelitzar el desplaçament lliure dels passatgers per l'aeroport, mentre fan temps per anar cap al següent bloc crític del procés de sortides (que pot ser: facturació, pas per filtres de seguretat o embarcament), s'han creat tres components, que es mostren a la figura 5.17.

Aquests components defineixen diferents mides de rutes d'establiments comercials: tendes, bars, restaurants, quioscs, etc., que visiten els passatgers durant el temps de lleure de que disposen durant el procés de sortida.

Depenent del número d'establiments comercials que defineix la ruta, tenim:

- ShopsTwo: Aquest component modelitza el desplaçament lliure dels passatgers dins d'una ruta de dos establiments comercials. La ubicació i mida de cada establiment comercial de la ruta es passa com a paràmetre d'entrada al component, ja que hi ha una forta dependència d'aquest component amb l'aeroport que s'estigui modelitzant.

- ShopsThree: Equivalent a l'anterior pel cas d'una ruta de tres establiments comercials.

- ShopsFive: Equivalent a l'anterior pel cas d'una ruta de cinc establiments comercials.

Aquests tres components es poden enllaçar per formar rutes més llargues, de diferents mides, segons l'aeroport que s'estigui modelitzant.

En aquest apartat analitzarem els paràmetres d'entrada i la lògica d'un d'ells, ShopsTwo, ja que els altres dos components segueixen el mateix criteri de disseny.

Tal i com es mostra a la figura 5.17, cadascun d'aquests components disposa d'un port d'entrada, situat a la seva part esquerra, d'un port de sortida, situat a la seva part dreta, i d'un port de sortida, situat a la part de dalt del component, per encaminar els objectes Passenger que hagin de sortir de la ruta sense haver pogut finalitzar-la.

Paràmetres d'entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterSizeShops:* enter que indica l'aforament màxim dels establiments comercials que comprenen la ruta.

- *parameterGround:* objecte de la classe PedGround, pertanyent a la llibreria Pedestrian, que representa els límits físics, definits com un grup de línies que representaran les parets, de l'entorn per on es mouran els passatgers. Dins d'aquest entorn estan definides les parets que delimiten els establiments comercials.

- *parameterlineShop1:* objecte de la classe ShapeLine, pertanyent al package com.xj.anylogic.engine.presentation de l'Anylogic, que guarda la línia on es pararan els objectes Passenger davant el primer establiment comercial de la ruta, per simular que miren el seu aparador. Hi ha un paràmetre d'aquest tipus per a cada establiment de la ruta, pel que els components *ShopsThree* i *ShopsFive* tindran tres i cinc paràmetres d'aquest tipus, respectivament.

- *parametreShop1:* objecte de la classe PedArea, de la llibreria Pedestrian, que guarda l'espai públic del primer establiment comercial de la ruta, on els objectes Passenger poden realitzar les seves compres. Hi ha un paràmetre d'aquest tipus per a cada establiment que composa la ruta, de manera que els components ShopsThree i ShopsFive tindran tres i cinc paràmetres d'aquest tipus, respectivament.

- *parameterlineShop2:* objecte de la classe ShapeLine, que pertany al package com.xj.anylogic.engine.presentation de l'Anylogic, que guarda la línia on es pararan els objectes Passenger davant el segon establiment de la ruta.

- *parametreShop2:* objecte de la classe PedArea, de la llibreria Pedestrian, que guarda l'espai públic del segon establiment comercial de la ruta, on els objectes Passenger poden realitzar les seves compres o consumicions.

- *parameterTimeFrame:* enter que indica el temps de folgança que tenen els passatgers abans de cancel·lar les seves compres per anar al següent punt del procés de sortida: facturació, filtres o embarcament, segons l'estat en que es trobi objecte Passatger.

Els components *ShopsThree* i *ShopsFive* tenen els mateixos paràmetres d'entrada que el component *ShopsTwo*, més alguns addicionals, però que ja han estat explicats en aquest apartat.

Lògica interna

La lògica que segueix aquest component, i que es mostra a la figura 5.18, modelitza el següent comportament:

Cada cop que un objecte Passenger es situa davant l'entrada de cadascun dels establiments comercials que té la ruta fa una consulta, mitjançant una crida al mètode cancelShopping(), de la classe Passenger, del temps màxim de que disposa per anar fins al següent bloc del procés de sortides, que va en funció de l'estat del passatger i de la franja horària en la que es realitza la consulta.

En cas que disposi de temps per continuar la ruta, entra a l'establiment i hi roman a dins durant un temps determinat, d'entre 1 a 2 minuts.

En cas contrari, és a dir, que no disposi de més temps per continuar la ruta, sortirà d'ella i es dirigirà cap el següent bloc del procés de sortides que li

correspongui segons l'estat en que es trobi, que pot ser: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament.

A més, abans d'entrar a cada establiment comercial fa una consulta de l'aforament màxim de l'establiment, que es passa com a paràmetre d'entrada al component i, en cas d'estar ple, es dirigeix cap al següent establiment comercial o, en cas de ser l'últim, surt de la ruta.

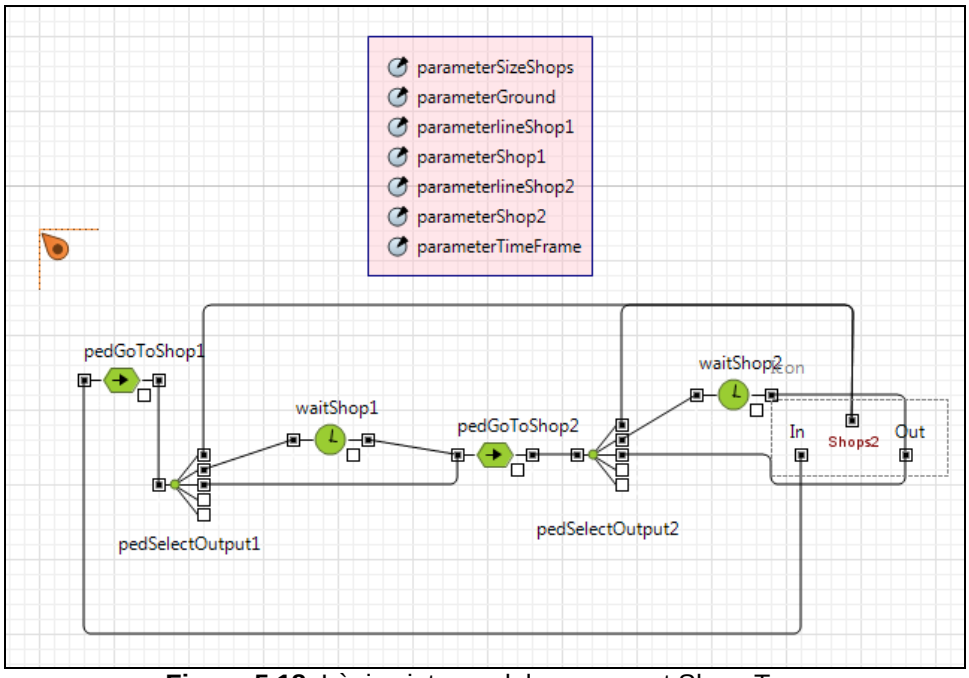


Figura 5.18: Lògica interna del component ShopsTwo

Per modelitzar aquest comportament s'ha fet ús dels següents objectes:

a) pedGoToShop1

Instància de la classe PedGoTo, de la llibreria Pedestrian de l'Anylogic, que guarda la línia on els objectes Passenger s'aturaran davant el primer establiment de la ruta abans d'entrar-hi o de continuar cap al següent, si aquest està ple.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Target:	parameterlineShop1

La resta de paràmetres mantenen els valors per defecte.

b) PedSelectOutput1

Instància de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, que permet a l'objecte Passenger seleccionar una de les tres alternatives següents:

- Sortir de la ruta per anar al següent bloc del procés de sortida: facturació, pas per filtres de seguretat o embarcament. Aquesta alternativa s'avalua tenint en compte l'estat en que es troba l'objecte Passenger: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament; l'hora actual de simulació i el temps màxim de que disposa el passatger per anar fins al següent bloc.
- Entrar al primer establiment comercial, en cas que aquest no estigui ple.
- Dirigir-se cap el segon establiment en, cas que el primer estigui ple.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Selection mode:	Conditions
Condition1:	ped.cancelShopping(getMinute(),parameterTimeFrame,ped.getState()) // funció de la classe Passenger que avalua si l'objecte Passenger pot continuar la ruta o ha de sortir cap el següent bloc del procés de sortida, segons el seu estat: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament.
Condition 2:	parameterShop1.size()<parameterSizeShops // sentència que retorna cert si el primer establiment comercial de la ruta està per sota de l'aforament màxim passat com a paràmetre d'entrada.
Condition 3:	parameterShop1.size()>=parameterSizeShops // sentència que retorna cert si el primer establiment comercial de la ruta ha arribat o ja ha sobrepassat l'aforament màxim passat com a paràmetre d'entrada.
Condition 4:	false

La resta de paràmetres mantenen els valors per defecte.

c) waitShop1

Instància de la classe PedWait, pertanyent a la llibreria Pedestrian de l'Anylogic, que fa esperar als objectes Passenger dins l'àrea del primer establiment de la ruta entre un i dos minuts. Aquest temps d'espera està definit mitjançant una funció de distribució uniforme contínua que genera valors dins de l'interval: [1,2].

Els temps d'espera dins del primer establiment comercial no és parametritzable, per tal d'evitar que es pugui passar al component un temps d'espera massa gran, que faci sobrepassar el temps màxim de que disposa el Passatger per anar fins al següent bloc del procés de sortides i, per tant, perdi el vol.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Waiting location:	Random inside area
Area to wait:	parameterShop1
Type:	delay
Delay time:	uniform (1.0*minute(), 2.0*minute());

La resta de paràmetres mantenen els valors per defecte.

d) pedGoToShop2

Instància de la classe PedGoTo, de la llibreria Pedestrian de l'Anylogic, que guarda la línia on els objectes Passenger s'aturaran davant el segon establiment comercial de la ruta abans d'entrar-hi o de continuar cap al següent, si aquest està ple.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Target:	parameterlineShop2

La resta de paràmetres mantenen els valors per defecte.

e) PedSelectOutput2

Instància de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, que permet a l'objecte Passenger seleccionar una de tres alternatives que també permet l'objecte *PedSelectOutput1* (veure punt b) d'aquest mateix apartat).

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Selection mode:	Conditions
Condition1:	ped.cancelShopping(getMinute(),parameterTimeFrame,ped.getState()) // funció de la classe Passenger que avalua si l'objecte Passenger pot continuar la ruta o ha de sortir cap el següent bloc del procés de sortida, segons el seu estat: pendent de facturar, pendent de passar per filtres de seguretat o pendent d'embarcament.
Condition 2:	parameterShop2.size()<parameterSizeShops // sentència que retorna cert si el segon establiment comercial de la ruta està per sota de l'aforament màxim passat com a paràmetre d'entrada.
Condition 3:	parameterShop2.size()>=parameterSizeShops // sentència que retorna cert si el segon establiment comercial de la ruta ha arribat o ja ha sobrepassat l'aforament màxim passat com a paràmetre d'entrada.
Condition 4:	false

La resta de paràmetres mantenen els valors per defecte.

f) waitShop2

Instància de la classe PedWait, pertanyent a la llibreria Pedestrian de l'Anylogic, que fa esperar als objectes Passenger dins l'àrea del segon establiment de la ruta entre un i dos minuts. Aquest temps d'espera està definit mitjançant una funció de distribució uniforme contínua que genera valors dins de l'interval: [1,2].

Igual que en l'objecte waitShop1, els temps d'espera dins del segon establiment comercial no és parametritzable. Amb aquesta mesura es pretén evitar que es pugui passar al component un temps d'espera massa gran, que

faci sobrepassar el temps màxim de que disposa el Passatger per anar fins al següent bloc del procés de sortides i, per tant, perdi el vol.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Waiting location:	Random inside area
Area to wait:	parameterShop2
Type:	delay
Delay time:	uniform (1.0*minute(), 2.0*minute());

La resta de paràmetres mantenen els valors per defecte.

La lògica interna dels components *ShopsThree* i *ShopsFive* és la mateixa que la del component *ShopsTwo.* Simplement afegeix els objectes addicionals de les classes PedGoTo, PedSelectOutput i PedWait necessaris, en funció del nombre d'establiments comercials que tingui la ruta.

Per tal de modelitzar el comportament estocàstic que ha de tenir cada objecte Passenger a l'hora de triar la ruta d'establiments comercials a visitar, cal usar un objecte selector de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, que tingui connectada un ruta, o un conjunt de rutes enllaçades, a cadascuna de les seves branques. També caldrà assignar-li a aquest objecte una probabilitat de pas per cada branca, de manera que quan un objecte Passenger entri dins d'aquest selector, fent ús d'aquestes probabilitats, pugui fer la tria de la ruta a seguir.

Per a la modelització d'un aeroport concret, un ús adequat d'aquests components ha de permetre establir les rutes de tendes més adients, atenent tant a les característiques pròpies dels diferents grups de passatgers que es puguin considerar, com a l'oferta particular d'establiments comercials d'aquell aeroport.

Per exemple, en cas que l'aeroport que s'estigui modelitzant disposi de tendes infantils, es podria crear una ruta d'establiments comercials que passi per aquestes tendes, per tal de satisfer les preferències dels passatgers que viatgin amb nens petits.

5.7. Porta d'Embarcament

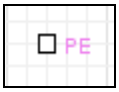


Figura 5.19: Porta d'Embarcament (PE)

Aquest component modelitza la lògica d'una porta d'embarcament, mitjançant una zona d'espera, una cua d'espera de persones, amb política FIFO, i un punt d'atenció on es simula el temps que triga el personal de l'aeroport en revisar la targeta d'embarcament de cada passatger. La situació del punt d'atenció i la cua d'espera es passen com a paràmetres d'entrada del component.

Tal i com es mostra a la figura 5.19, aquest component només disposa d'un port d'entrada.

Paràmetres entrada

Els paràmetres d'entrada que té aquest component són els següents:

- *parameterPunt_Atencio:* objecte de la classe ShapeLine, que pertany al package com.xj.anylogic.engine.presentation de l'Anylogic, que guarda el dibuix de la línia que representa el punt d'atenció on es pararan els objectes Passenger durant un temps determinat passat com a paràmetre d'entrada del component.

- *parameterCua:* objecte de la classe ShapePolyLine, que pertany al package com.xj.anylogic.engine.presentation de l'Anylogic, que guarda el dibuix de la línia que representa la cua d'espera que es forma davant del punt d'atenció de la porta d'embarcament.

- *parameterTempsRevisarTargMin:* variable de tipus double que guarda el límit inferior del rang de temps d'espera.

- *parameterTempsRevisarTargMax:* variable de tipus double que guarda el límit superior del rang de temps d'espera.

- *parameterAreaEspera:* objecte de la classe pedArea, pertanyent a la llibreria Pedestrian, que guarda l'àrea que ocupa la zona d'espera que hi ha situada al costat de la porta d'embarcament.

- *parameterEntradaAreaEspera:* objecte de la classe ShapeLine, que pertany al package com.xj.anylogic.engine.presentation de l'Anylogic, que guarda la línia d'entrada a la zona d'espera on es farà l'avaluació de si entrar a la zona d'espera o si embarcar.

- *parameterTempsEmbarcament:* variable entera que guarda el temps que dura l'embarcament. El valor per defecte d'aquest paràmetre és de 6 franges horàries, que equival a 30 minuts.

Lògica interna

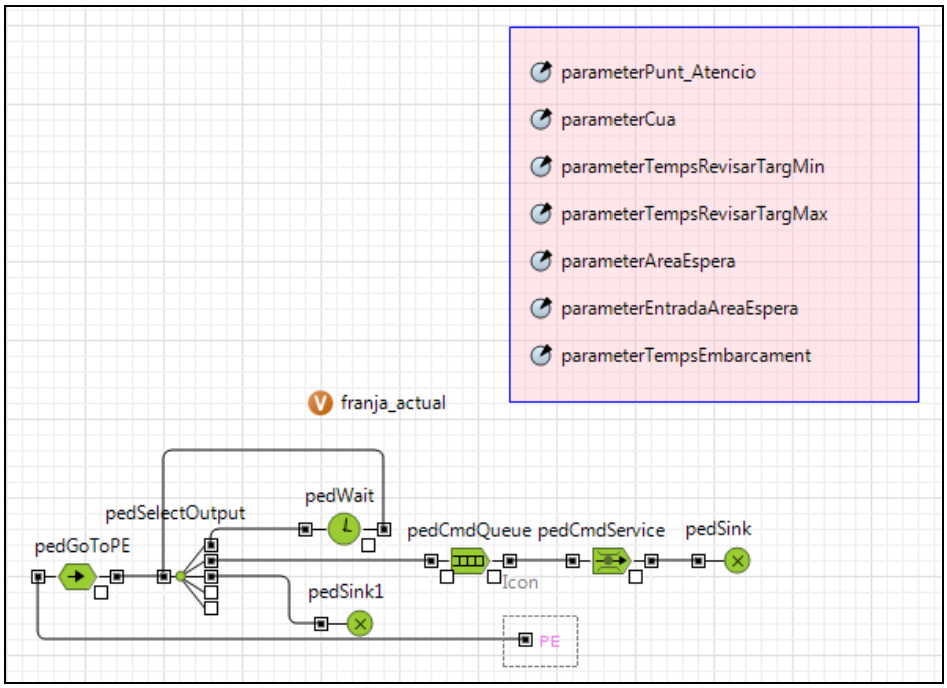


Figura 5.20: Lògica interna Porta Embarcament

Tal i com es mostra a la figura 5.20, la lògica interna de la *Porta d'Embarcament* està formada per 7 objectes, que s'expliquen a continuació:

a) pedGoToPE

Instància de la classe PedGoTo, de la llibreria Pedestrian de l'Anylogic, que dirigeix els objectes Passenger cap a la línia d'entrada de la zona d'espera.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Target:	parameterEntradaAreaEspera

La resta de paràmetres mantenen els valors per defecte.

b) PedSelectOutput

Instància de la classe PedSelectOutput de la llibreria Pedestrian de l'Anylogic, on cada objecte Passenger avalua si ha d'entrar dins la zona d'espera o bé s'ha de dirigir cap a la porta d'embarcament.

Aquesta avaluació es realitza en funció de si la porta d'embarcament està oberta o no en el moment de l'avaluació. Si el resultat de l'avaluació és que ha d'anar a la zona d'espera, l'objecte Passenger surt per la primera branca del selector, fins arribar a l'objecte lògic pedWait. En canvi, en cas que pugui començar el procés d'embarcament, l'objecte Passenger surt per la segona branca del selector fins arribar a l'objecte *pedCmdQueue.*

Dins d'aquest objecte selector també s'avalua si l'objecte Passenger ha arribat tard, o no, a l'embarcament, pel que en cas d'haver perdut el vol, es dirigeix cap a l'objecte *pedSink1,* per tal que no quedi comptabilitzat dins del procés d'embarcament d'un vol posterior al seu.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Selection mode.	Conditions
Condition1:	franja_actual<(ped.getFlight().getFranja()-parameterTempsEmbarcament) // codi que retorna cert si la porta d'embarcament està tancada
Condition2:	franja_actual >= (ped.getFlight().getFranja() - parameterTempsEmbarcament) && (franja_actual <= ped.getFlight().getFranja()) // codi que retorna cert si ja es pot començar el procés d'embarcament i, a més, no s'ha perdut el vol.
Condition3:	franja_actual > ped.getFlight().getFranja() // codi que retorna cert si l'objecte Passenger ha arribat tard a l'embarcament del seu vol.
Condition4:	false
On enter:	franja_actual = ((getMinute()+(getHourOfDay()*60))/5)+1; // codi que calcula la franja actual de simulació per poder fer les comparacions en els aparats, Condition1, Condition2 i Condition3.
On Exit2:	pedWait.freeAll(); // En cas que la porta d'embarcament estigui oberta, s'executa aquest codi per alliberar a tots els objectes Passenger que estan dins la sala d'espera. Per tant, quan un objecte Passenger ha obtingut com a resultat de l'avaluació que ha de dirigir-se cap a la porta d'embarcament, allibera als altres objectes Passenger, per tal que tots es dirigeixin a fer cua davant la porta.

La resta de paràmetres mantenen els valors per defecte.

c) pedWait

Instància de la classe PedWait, de la llibreria Pedestrian de l'Anylogic, que simula un temps d'espera dins la zona d'espera, determinat per una funció de distribució uniforme contínua que genera valors, en minuts, dins de l'interval:
[1 , 5].

L'àrea que compren la zona d'espera es passa com a paràmetre d'entrada al component.

Quan el temps d'espera finalitza, l'objecte Passenger es dirigit cap al selector, anomenat *pedSelectOutput,* per a que torni a realitzar l'avaluació del camí a seguir: tornar a la zona d'espera o començar l'embarcament.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Waiting location:	Random inside area
Area to wait in:	parameterAreaEspera
Type:	Delay
Delay time:	uniform (1.0*minute(), 5.0*minute())
Delay starts when:	Ped stops at the point

La resta de paràmetres mantenen els valors per defecte.

d) pedSink1

Instància de la classe PedSink, de la llibreria Pedestrian de l'Anylogic, que treu del sistema els objectes Passenger que han perdut el vol i, per tant, no poden continuar el procés d'embarcament. Aquest objecte permet portar una comptabilització de tots els objectes Passenger que han perdut el seu vol.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger

La resta de paràmetres mantenen els valors per defecte.

e) pedCmdQueue

Instància de la classe PedCmdQueue, de la llibreria Pedestrian de l'Anylogic, que representa la cua d'espera que es forma davant la porta d'embarcament, quan aquesta és oberta. La situació i forma que ha de tenir aquesta cua es passa com a paràmetre d'entrada al component.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Type:	line
Shape:	parameterCua
On exit:	pedWait.freeAll(); // codi que allibera els objectes Passenger de la sala d'espera. Aquest codi es posa com a reforç per evitar que quedin passatgers esperant a la sala d'espera quan la porta d'embarcament s'hagi obert.

La resta de paràmetres mantenen els valors per defecte.

f) pedCmdService

Instància de la classe PedCmdService, pertanyent a la llibreria Pedestrian de l'Anylogic, que modelitza diferents tipus de serveis, com ara una venda de tiquets en una estació de tren, un pas per un torniquet o, en el nostre cas, el servei de revisió de targetes d'embarcament que es realitza just abans de l'embarcament.

El dibuix de la línia de parada davant la porta d'embarcament es passa com a paràmetre d'entrada de l'objecte *Porta d'Embarcament.*

La configuració d'aquest objecte és la següent:

Ped class:	Passenger
Type:	Delay
Shape:	parameterPunt_Atencio
Delay time:	uniform(parameterTempsRevisarTargMin*second(), parameterTempsRevisarTargMax*second())
On Exit:	pedWait.freeAll() //codi per desbloquejar els objectes Passenger que hagin quedat a la sala d'espera per a que procedeixin a l'embarcament.

La resta de paràmetres mantenen els valors per defecte.

g) pedSink

Instància de la classe PedSink, de la llibreria Pedestrian de l'Anylogic, que treu del sistema els objectes Passenger un cop han acabat el procés d'embarcament.

Aquest objecte permet portar una comptabilització de tots els objectes Passenger que han completat amb èxit el procés complet de sortides.

La configuració d'aquest objecte és la següent:

Ped class:	Passenger

La resta de paràmetres mantenen els valors per defecte.

h) franja_actual

Variable entera que guarda el resultat del càlcul de la franja_actual de simulació. Aquest càlcul es realitza dins l'objecte pedSelectOutput cada cop que un objecte Passenger realitza una avaluació del camí que ha de seguir dins la lògica del component *Porta d'Embarcament*.

6. Cas d'aplicació: Terminal T1 de l'Aeroport de Barcelona

En aquest apartat s'explicarà un exemple d'aplicació de la llibreria de components per modelitzar el procés de sortides de passatgers d'un aeroport.

S'ha triat la nova terminal T1 de l'Aeroport de Barcelona com a sistema a modelitzar, ja que, a més de ser un dels projectes d'enginyeria civil més importants realitats a Europa, del seu atractiu en quant al seu disseny innovador, combinat amb l'ús d'alta tecnologia, la seva proximitat a la Universitat Autònoma de Barcelona permet que es pugui visitar per tal d'obtenir informació útil per a la seva modelització. Per exemple, tot i que en els plànols que hi a la web d'AENA (www.aena.es), es veu que la terminal disposa de 6 illes de facturació, el dia que es va fer la visita a la terminal es va comprovar que la primera illa de facturació encara no està operativa, pel que el model que s'ha desenvolupat no contempla cap mostrador de facturació dins la primera illa. En general, s'ha mirat de respectar al màxim la topologia d'elements del sistema a modelitzar, així com la seva funcionalitat. Només s'han efectuat dues simplificacions, relatives al nombre d'establiments comercials i al nombre de portes d'embarcament incorporades al model.

Val a dir, però, que no s'ha realitzat una validació del model com a tal. És a dir, els temps mitjans d'espera que s'han aplicat als taulells de facturació, zona de filtres de seguretat i portes d'embarcament són aproximats, i no han estat validats al sistema real, atès que aquest cas d'aplicació només pretén servir d'exemple en l'ús dels components de la llibreria que s'ha creat i la seva interrelació amb els altres objectes genèrics de que disposa l'Anylogic, per tal de modelitzar el procés de sortides d'un aeroport. En aquest sentit, el que sí resulta imprescindible és validar els resultats de la simulació, i això sí que s'ha fet, tant pel que fa a la lògica del moviment de passatgers a dins de la terminal com a la comptabilització global de passatgers, de manera que es comprova que tots els passatgers associats a la programació de vols objecte de la simulació, o bé acaben embarcant al seu avió, o bé arriben tard i, per tant, no embarquen. És a dir, garantint que cap passatger es "perdi" a la simulació.

6.1. Terminal T1: informació general

Amb la posada en servei de la terminal T1[4], Barcelona es consolida com l'aeroport de referència de l'espai mediterrani, donant resposta a les necessitats plantejades per una de les àrees metropolitanes europees amb major creixement del trànsit aeri.

Tal i com es mostra a la figura 6.1, la T1 és un edifici de gran complexitat logística i tècnica, que ocupa una superfície de quasi 600.000 m^2, on es preveu que cada dia hi passin més de 100.000 passatgers.

[4] La terminal T1 de l'Aeroport de Barcelona va ser inaugurada el dia 16 de juny de 2009.

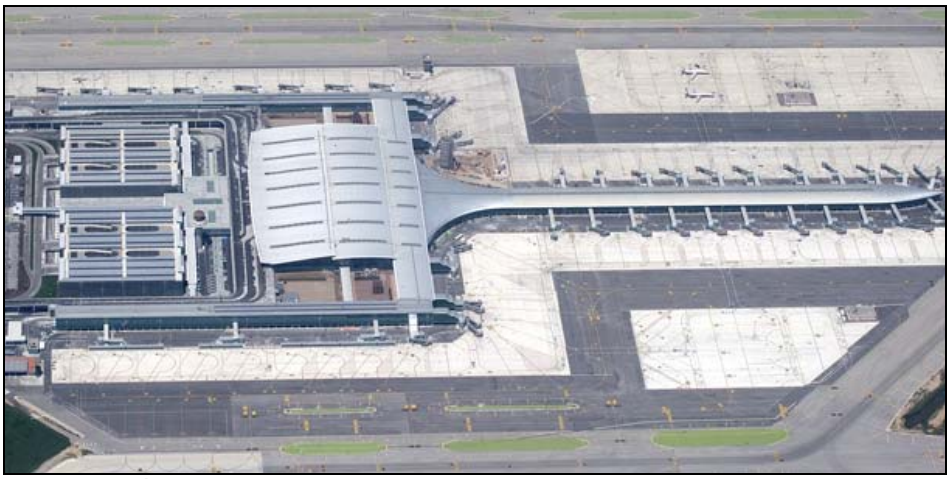


Figura 6.1: Vista aèria de la nova terminal T1 de l'Aeroport de Barcelona (Font: www.aena.es)

La proposta arquitectònica duta a terme per l'arquitecte Ricardo Bofill conjuga un caràcter internacional i multicultural amb la lluminositat i calidesa ambiental de l'arquitectura mediterrània.

L'edifici està perfectament adaptat a la línia del paisatge, amb la il·luminació natural com a protagonista a totes les zones destinades als passatgers. D'aquesta manera, s'ha aconseguit integrar aquest gran nus de comunicacions i gran centre de serveis, a una zona de gran interès ecològic minimitzant-ne l'impacte mediambiental.

El resultat és un complex que s'estructura al voltant de tres grans elements: un edifici processador que allotja la facturació, la recollida d'equipatges i la zona comercial, dos dics laterals i un de longitudinal dedicats a l'embarcament de passatgers, i un vestíbul on confluiran els diferents sistemes de transport.

Aquesta estructura gaudeix d'una ordenació que dóna una enorme funcionalitat pels passatgers i les companyies aèries.

Tal i com es mostra a la figura 6.2, la terminal T1 compta amb 5 plantes, dues de les quals estan dedicades a les sortides de passatgers (plantes 1 i 3) i dues a les arribades (plantes 0 i 2).

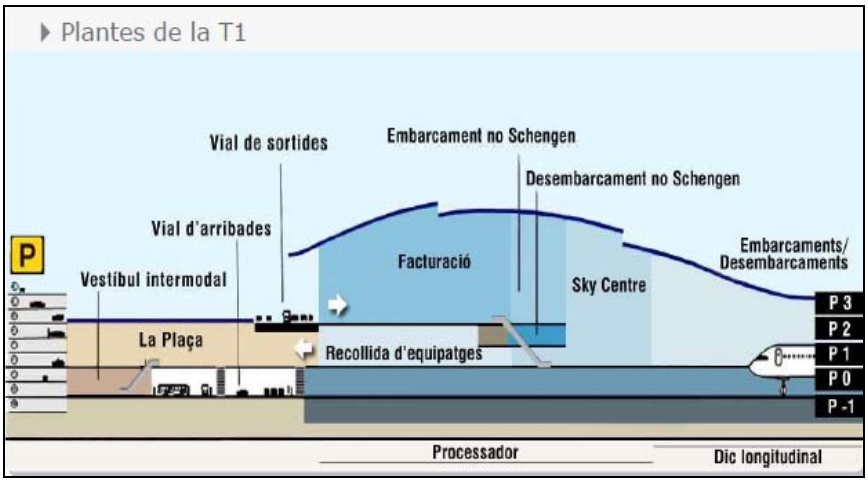


Figura 6.2: Esquema de la terminal T1 de Barcelona (Font: www.aena.es)

Pel que fa a l'organització de medis que intervenen en el procés de sortides de passatgers de la terminal, la planta 3 disposa de 6 illes de facturació (5 operatives), on hi ha actualment 140 mostradors operatius. A més, el Corredor Barcelona-Madrid disposa dels seus propis taulells de facturació al vestíbul. A la mateixa planta 3, després de la zona de facturació s'hi ubica la zona de filtres de seguretat, tal com mostra la figura 6.3.

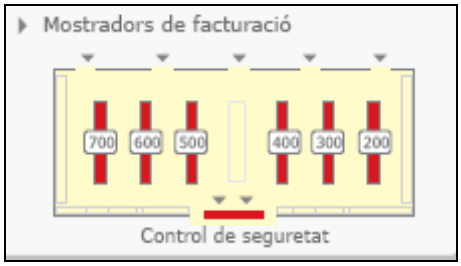


Figura 6.3: Esquema de la planta 3 de la terminal T1 (Font: www.aena.es)

Un cop passats els filtres, els passatgers poden accedir a la seva zona d'embarcament. Les portes d'embarcament estan agrupades en 5 zones: les zones A, B i C, situades a la planta 1, constitueixen l'àrea principal d'embarcament. A aquesta àrea s'hi accedeix des de la planta 3 baixant per unes escales mecàniques, mostrades a la figura 6.2. Addicionalment, les zones D i E, que es troben al mateix nivell que facturació i filtres (planta 3), constitueixen una segona àrea d'embarcament, molt menor, però, que l'anterior. A la figura 6.4 es mostra un esquema de les diferents zones d'embarcament.

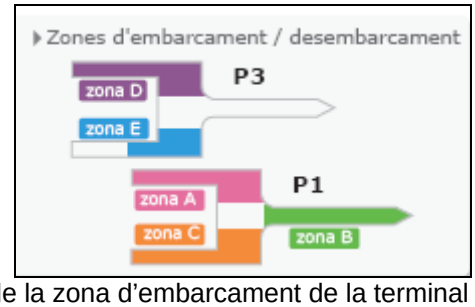


Figura 6.4: Esquema de la zona d'embarcament de la terminal T1 (Font: www.aena.es)

En quant a l'accessibilitat, la nova terminal permet el seu accés a través de vehicles privats (per la carretera C-31 o C-32 a través de la desviació B-22); taxi (hi ha dues parades); autobusos urbans, interurbans, línies regionals i Aerobús (parades a les plantes 0 i 3). A més, un servei permanent de llançadores comunica la terminal T1 amb la terminal T2.

Actualment, es troba en construcció l'estació de Renfe de la T1, que se situarà al vestíbul ferroviari, a la planta 0 de la nova terminal.

La T1 disposa també de dues àrees d'aparcament públic que sumen més de 12.000 places. Una formada per dos edificis de nou plantes (9.400 places) més un aparcament a la superfície (1.600 places) amb accés directe a la terminal; i una altra de llarga estada (1.059 places), a dos quilòmetres de la T1.

Pel que fa a la zona comercial, la terminal disposa de 81 botigues i punts de restauració que fan d'aquesta terminal una de les zones comercials més atractives d'Europa, on es poden fer compres i gaudir d'una variada selecció de restaurants i cafeteries.

La figura 6.5 resumeix les principals característiques de la nova terminal T1 de Barcelona.

PRINCIPALS CARACTERÍSIQUES DE LA T1	
CAPACITAT OPERATIVA	
Passatgers	30 milions a l'any. T1 + T2, 55 milions de passatgers
PLATAFORMA	74 places d'estacionament
Capacitat màxima del camp de vols	90 operacions / hora
SUPERFÍCIE	
T1	544.066 m^2
Zones públiques	155.200 m^2
Les botigues de l'aeroport	23.866 m^2
Sales VIP	6.066 m^2
Sales de recollida d'equipatges	20.000 m^2
Aparcament i vials de servei	34.500 m^2
PLATAFORMA	600.000 m^2
SERVEIS AL PASSATGER	
Taulells de facturació	166
Pantalles informatives	256
Taulells d'informació	8
Punts d'atenció al passatger	14
Controls de seguretat	28
Control de passaports	52
Portes d'embarcament	101
Passarel·les	43 (ampliables a 50)
Hipòdroms de recollida d'equipatges	15

Figura 6.5: Resum de característiques més destacades de la terminal T1 (Font: www.aena.es)

6.2. Simulació del procés de sortides

En aquest apartat s'explicarà com s'ha modelitzat la lògica del procés de sortides de passatgers de la terminal T1 de l'Aeroport de Barcelona.

L'explicació s'ha dividit en 8 blocs:

- Creació de l'entorn físic del model
- Modelització de l'arribada de passatgers a la terminal
- Modelització del procés de facturació
- Modelització del pas pels filtres de seguretat
- Modelització del procés d'embarcament

- Modelització del temps d'oci
- Procés d'inicialització
- Validació del resultat de la simulació

La parametrització dels objectes continguts a les diferents seccions del model, que s'aniran comentant en els apartats següents, es pot consultar (en format electrònic) a l'annex 2 del projecte: Codi font Anylogic del model de la terminal T1 de l'Aeroport de Barcelona.

L'equip informàtic usat a les simulacions és un ordinador portàtil HP Pavilion Entertainment PC amb dos processadors Intel Core i5 M450 a 2.27 GHz cadascun, i una memòria RAM de 4 GB. El sistema operatiu instal·lat a l'equip és Windows 7 de 64 bits.

6.2.1. Creació de l'entorn físic del model

Tal com s'ha comentat a l'apartat 6.1, el procés de sortides de la terminal T1 implica operacions a les plantes 1 i 3. Així, per tal de disposar d'un entorn físic realista per a la simulació del moviment de passatgers en aquest procés de sortides, s'ha treballat amb un plànol a escala de la terminal T1, contingut en un arxiu Autocad.

Mitjançant una versió d'avaluació del programa Autocad 2011, s'ha fet un tractament del plànol per tal de visualitzar només les capes necessàries pel model. A més, tot i que l'Anylogic permet importar plànols fets en Autocad, degut a un problema d'incompatibilitat amb la versió del fitxer .cad, s'ha optat per convertir els plànols a format d'imatge .png, en tamany DIN A0.

Un cop importades les imatges de les plantes 1 i 3 dins l'editor gràfic de l'objecte Main del model, s'han redibuixat els dos plànols mitjançant línies i polylínies que proveeix l'Anylogic per crear els dibuixos dels models a implementar.

D'aquesta manera s'han creat els entorns físics de cada planta, fent ús d'objectes de la classe PedGround de l'Anylogic. Aquests objectes estan formats per un conjunt de línies que simulen les parets que cap objecte Passenger pot travessar.

Finalment, i per completar l'explicació de com s'ha modelitzat la terminal, cal dir que s'ha fet ús d'un objecte de la classe PedConfiguration, pertanyent a la llibreria Pedestrian de l'Anylogic, que conté les dades de configuració del model, com són per exemple, la forma que prendran els objectes Passenger durant la simulació, o l'escala del model, representada en píxels per metre.

6.2.2. Modelització de l'arribada de passatgers a la terminal

Segons el mitjà de transport usat per arribar l'aeroport, considerarem que els passatgers poden accedir a la zona de facturació de la terminal, situada a la planta 3, per dues vies:

1) 5 portes d'entrada situades a peu de carrer (planta 3): usades pels passatgers que arriben en transport públic.

2) 2 cintes mecàniques que pugen des de la planta inferior: usades pels passatgers que arriben en vehicle particular i el deixen estacionat a l'aparcament soterrat de que disposa la mateixa terminal.

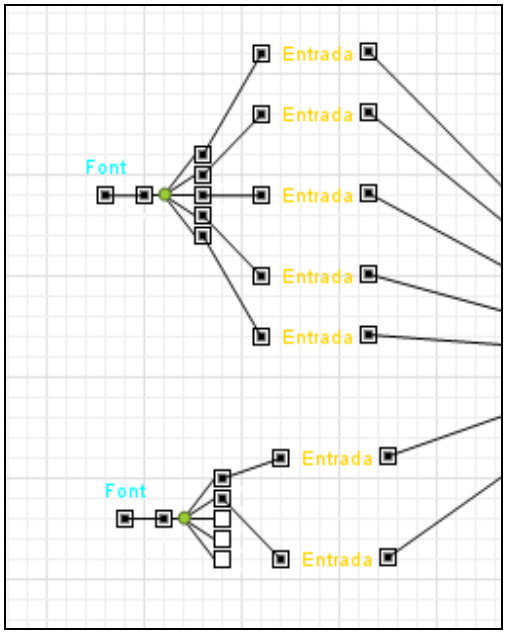


Figura 6.6: Components de la llibreria que modelitzen l'arribada de passatgers a la terminal T1.

Per tal de simplificar la modelització d'aquesta part del procés de sortides, tot i que la planta 3 també disposa d'ascensor i dues escales, aquests mitjans d'accés a la planta no s'han considerat en el model.

Per tant, tal i com es mostra a la figura 6.6, per modelitzar el primer bloc del procés de sortides ha calgut usar els següents components de la llibreria:

- 2 Fonts de passatgers.

- 7 Portes d'entrada al terminal: 5 per modelitzar les portes d'entrada a peu de carrer i 2 per modelitzar les cintes mecàniques.

Per a la definició de l'entorn físic de la planta 3 s'ha usat un objecte (anomenat P3, que es mostra a la figura 6.17) de la classe PedGround, pertanyent a la llibreria Pedestrian de l'Anylogic, el qual es passa com a paràmetre d'entrada a cada component del tipus *Porta d'entrada a la terminal* que s'ha utilitzat al model.

Finalment, per acabar el bloc d'arribada de passatgers a la terminal, s'han enllaçat tots els components *Porta d'entrada a la terminal*, fent ús del seu port de sortida, a un objecte de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, que s'ha anomenat CheckIn_Or_Shopping, tal i com es mostra a la figura 6.7. Mitjançant l'ús de probabilitats, cada objecte Passenger que entra dins d'aquest objecte selector, decideix passar al següent

bloc del procés de sortides, que és la facturació, o bé realitzar un desplaçament lliure per la planta 3 de la terminal (veure apartat *Modelització del temps d'oci*)

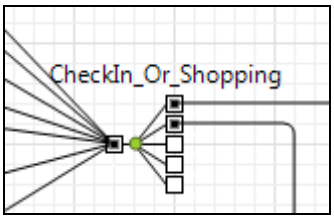


Figura 6.7: Objecte de la classe PedSelectOutput de la llibreria Pedestrian de l'Anylogic situat entre els blocs d'arribada de passatgers i facturació.

6.2.3. Modelització del procés de facturació

Actualment, la terminal disposa de 140 mostradors de facturació operatius, repartits en 5 illes de facturació, dels quals s'han modelitzat 133 en forma de facturació individual i 7 en forma de grup de multifacturació.

Facturació individual

Per modelitzar la lògica de la facturació en mostradors individuals, s'ha fet ús de 133 components del tipus *Taulell de Facturació*.

A part d'aquests components, s'ha considerat necessari fer ús d'un objecte de la classe ActiveObject de l'Anylogic, anomenat *Facturació,* que empaqueta tota la lògica necessària per conduir cada objecte Passenger fins al mostrador de facturació individual que li correspongui. D'aquesta manera és més fàcil resseguir i comprendre tota la lògica del procés de sortides que s'ha implementat dins l'espai de disseny de l'objecte Main del model.

Com ja s'ha explicat al capítol 4, la disponibilitat dels mostradors de facturació depèn d'una programació de medis que es realitza diàriament a l'aeroport, i que va en funció de les assignacions de mostradors que realitzi l'operador de handling a cada companyia aèria, segons la franja horària del dia. Per això, tot i que cada objecte Passenger té un rang de mostradors assignats a priori al vol, en els que pot facturar, a la pràctica només podrà facturar en aquells mostradors del rang que estiguin disponibles en la franja horària en la que realitzi aquesta operació.

A la figura 6.8 es mostra el detall d'una part de la lògica implementada dins l'objecte Facturació. Com es pot observar, s'usen objectes selectors, pertanyents a la classe PedSelectOutput de la llibreria Pedestrian de l'Anylogic, que estableixen condicions de pas per cada camí de la lògica interna del procés de facturació.

En concret, hi ha un primer objecte selector, situat a l'esquerra de la figura 6.8, que fa una crida al mètode GoToCounter(), pertanyent a la classe Passenger, cada cop que hi passa un objecte Passenger, per calcular el número de mostrador de destí en funció dels criteris de disponibilitat explicats en el capítol 5, dins l'apartat de Facturació.

D'aquest primer objecte selector surten cinc branques, cadascuna de les quals està enllaçada a un subconjunt d'objectes PedSelectOutput, que simulen l'agrupació dels mostradors entre les 5 illes de facturació que hi ha operatives a la terminal T1 de Barcelona.

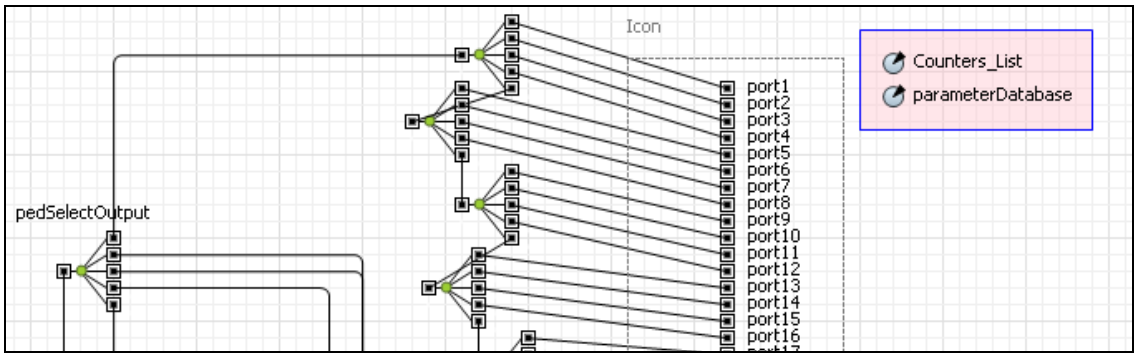


Figura 6.8: Vista de l'editor gràfic de l'objecte Facturació

A més, com es mostra a la mateixa figura 6.8, cada branca d'aquests selectors està connectada a un objecte de la classe Port, pertanyent al package com.xj.anylogic.engine de l'Anylogic, que permetrà connectar-hi un component *Taulell de Facturació* quan l'objecte Facturació s'arrossegui dins l'editor gràfic de l'objecte Main del model (veure figura 6.9).

Per tant, l'objecte Facturació tindrà tants objectes Port per connectar-hi components del tipus *Taulell de Facturació* com mostradors de facturació, que no pertanyin a cap grup de multifacturació, té la terminal T1 de Barcelona.

Els dos paràmetres d'entrada que té l'objecte Facturació són: Counters_List i parameterDatabase, que permeten passar-li el llistat de mostradors del tipus *Taulell de Facturació* i la connexió amb el fitxer de dades extern que conté les taules de disponibilitat de mostradors. Tant el llistat com la connexió amb el fitxer s'han creat dins l'objecte Main del model. Aquests paràmetres són necessaris per fer la crida al mètode *GoToCounter(),* que es realitza dins el primer objecte selector, i que s'encarrega de calcular el mostrador al que s'ha de dirigir l'objecte Passenger que hagi fet la crida.

Un cop arrossegat l'objecte Facturació dins la finestra de disseny de la terminal; és a dir, l'editor gràfic de l'objecte Main del model, a cada port de l'objecte Facturació s'hi connecta un component *Taulell de Facturació* (TF) tal i com es mostra a la figura 6.9, menys pels ports que van de l'1 al 7, ja que els mostradors que es corresponen amb aquesta numeració pertanyen a un grup de multifacturació, pel que els objectes Passenger que tinguin com a mostrador de facturació un d'aquests 7 mostradors, hi arribaran a través del camí implementat a la lògica del grup de multifacturació que s'explica en el següent punt.

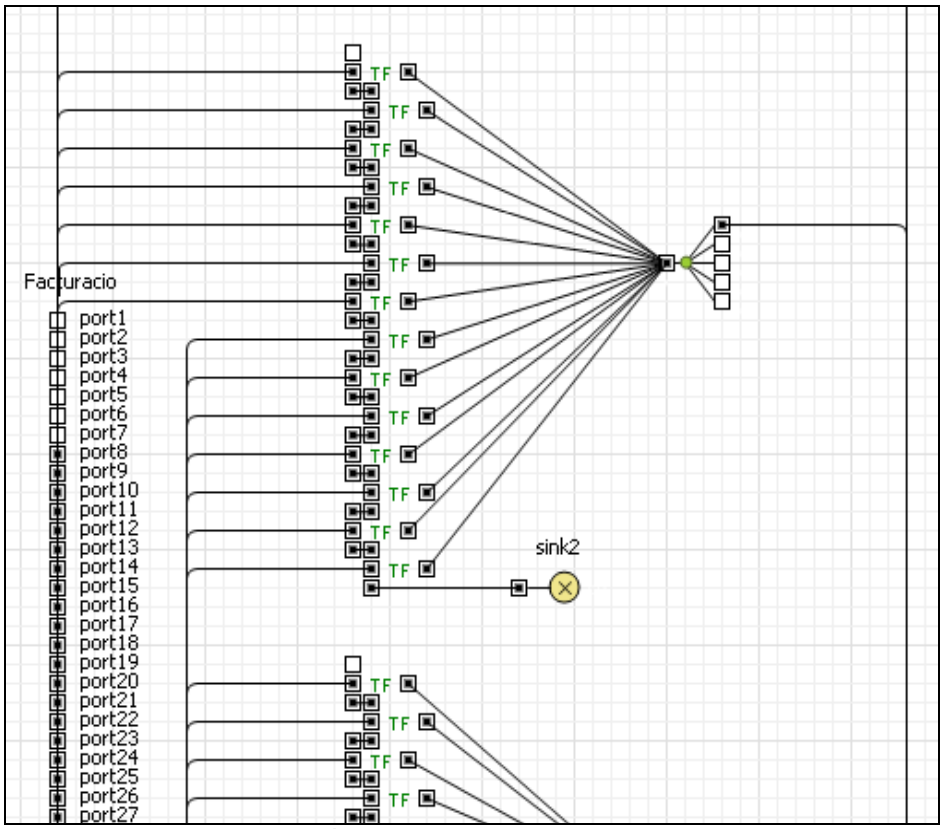


Figura 6.9: Vista de l'editor gràfic de l'objecte Main del model on es veu una part de l'objecte Facturació i un subconjunt dels components Taulell de Facturació connectats.

Dins la lògica del bloc de facturació, també cal fer ús d'objectes de la classe Sink de la llibreria Enterprise de l'Anylogic, com per exemple l'objecte sink2 que es mostra a la figura 6.9. Aquest tipus d'objectes s'han de connectar al port de sortida de maletes dels dos components *Taulell de Facturació* que s'hagin ubicat al model just al costat de la zona SATE (Sistema Automatitzat de Tractament d'Equipatges) de cada illa de facturació. Gràcies a aquests objectes, es treuen del sistema les maletes que arriben a la zona SATE, ja que el tractament d'equipatges no és objecte d'estudi en el model desenvolupat.

Grup de multifacturació

La lògica del grup de multifacturació, que s'ha creat a la primera illa de facturació (mostradors de l'1 al 7), està formada per 1 component *Serpentí* connectat amb 7 objectes del tipus *Taulell de MultiFacturació*, tal com es mostra a la figura 6.10.

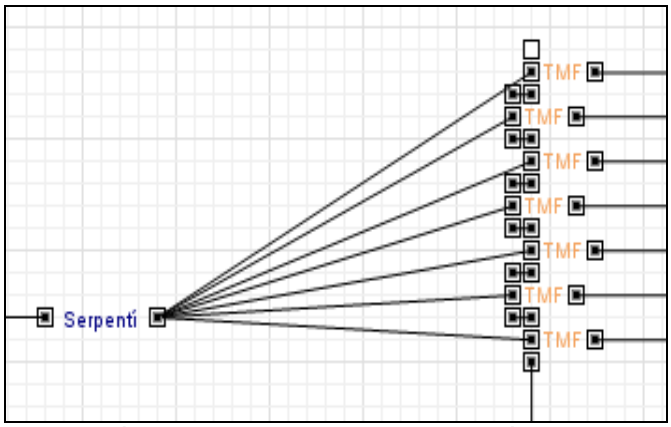


Figura 6.10: Lògica d'un grup de multifacturació de 7 mostradors.

Com ja s'ha explicat al capítol 5, dins la lògica del component *Serpentí* es realitza la gestió de la disponibilitat dels mostradors, per franges horàries, del grup de multifacturació, pel que quan un objecte Passenger surti del component Serpentí, es dirigirà al component del tipus *Taulell de MultiFacturació* més proper que estigui disponible segon la franja horària actual i desocupat.

Addicionalment, cal assenyalar que a l'inici de la lògica del bloc de facturació s'ha fet ús d'un objecte de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, anomenat Facturació_O_Multifacturació, tal i com es mostra a la figura 6.11. Dins d'aquest selector, cada objecte Passenger avalua si ha d'anar cap a la lògica de facturació o cap a al lògica del grup de multifacturació. Aquesta avaluació es realitza mitjançant la crida al mètode getMultiFact(), de la classe Flight, que determina si el vol que té associat l'objecte Passenger té assignat, o no, un grup de multifacturació. Aquesta informació s'extreu del fitxer de dades extern que usa el model.

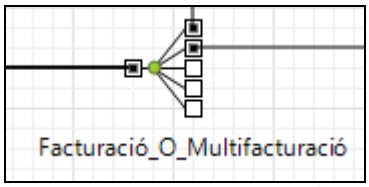


Figura 6.11: Objecte selector situat a l'inici de la lògica del bloc de facturació.

Finalment, per acabar el bloc de facturació s'han enllaçat tots els components *Taulell de Facturació i Taulell de MultiFacturació*, fent ús dels seus ports de sortida, a un objecte selector, pertanyent a la llibreria Pedestrian de l'Anylogic, que s'ha anomenat Shopping_Or_Filtres, tal i com es mostra a la figura 6.12. Mitjançant l'ús de probabilitats, els objectes Passenger que arriben al selector decideixen passar al següent bloc del procés de sortides, que és el pas per la zona de filtres de seguretat, o bé realitzar un desplaçament lliure per la planta 3 de la terminal (veure apartat *Modelització del temps d'oci*).

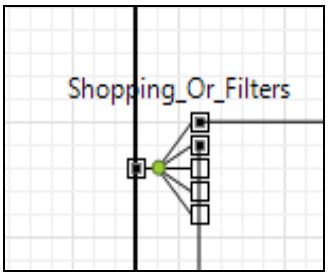


Figura 6.12: Objecte selector situat entre els blocs de facturació i pas per filtres de seguretat.

6.2.4. Modelització del pas pels filtres de seguretat

Tal i com es mostra a la figura 6.13, per modelitzar el pas pels filtres de seguretat s'han usat els següents components de la llibreria:

- 2 Serpentins
- 10 Filtres de seguretat

Aquests components modelitzen la configuració d'un dia concret de la zona de filtres de seguretat, on hi ha dos serpentins: un que dóna accés als primers cincs filtres de seguretat, i altre que dóna accés als cinc filtres de seguretat restants.

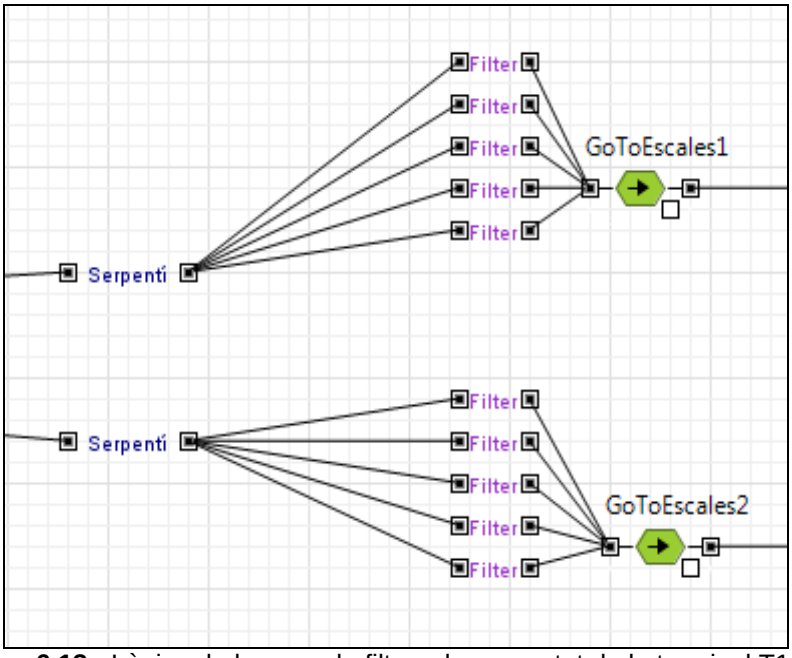


Figura 6.13: Lògica de la zona de filtres de seguretat de la terminal T1.

Com ja s'ha explicat al capítol 5, dins dels dos components *Serpentí*, es realitza la gestió de la disponibilitat segon franja horària dels components *Filtres de Seguretat* que hi tinguin connectats cadascun d'ells.

A més, per a que cada objecte Passenger pugui decidir cap a quin component *Serpentí* ha de dirigir-se, a l'inici de la lògica del bloc de pas per filtres de seguretat s'ha disposat un objecte selector, anomenat Go_To-Filters, que es

mostra a figura 6.14. Aquest selector té connectades les dues primeres branques de sortida als dos serpentins. L'avaluació del camí a triar que realitza cadascun dels objectes Passenger que passa pel selector, s'efectua en funció de la cua que hi hagi a cada serpentí en el moment de l'avaluació, pel que, igual que passa en la realitat, l'objecte Passenger triarà el serpentí que tingui la cua més curta.

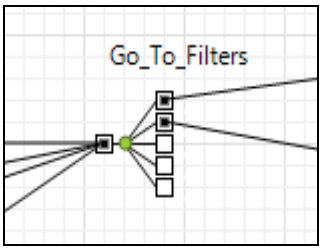


Figura 6.14: Objecte selector situat a l'inici del bloc de pas per filtres de seguretat.

Finalment, dins la lògica de la zona de filtres de seguretat de la terminal, s'han afegit dos objectes de la classe PedGoTo, que pertanyen a la llibreria Pedestrian de l'Anylogic: GoToEscales1 i GoToEscales2. Aquests dos objectes serveixen per dirigir els objectes Passenger, un cop han passat pels filtres de seguretat, cap a les escales que donen accés a la planta 1 de la terminal on hi ha les zones d'embarcament A, B i C, tal com es va comentar a l'apartat 6.1.

Per tant, un cop els objectes Passenger arriben a les escales de la zona de filtres, el model ha de permetre simular que baixen per les escales fins a la planta 1. Per fer-ho, s'ha fet ús d'objectes de la classe PedExit i PedEnter, pertanyents a la llibreria Pedestrian de l'Anylogic, que es mostren a la figura 6.15. La lògica conjunta d'aquests dos objectes transporta els passatgers d'una planta a una altra. Per exemple, un cop els objectes Passenger arriben a les primeres escales de la zona de filtres de seguretat, l'objecte ExitEscales1Planta3 s'encarrega de fer-los desaparèixer d'aquest punt del plànol de la terminal, i l'objecte EnterEscales1Planta1 s'encarrega de fer-los reaparèixer al final de les primeres escales que s'han dibuixat a la planta1.

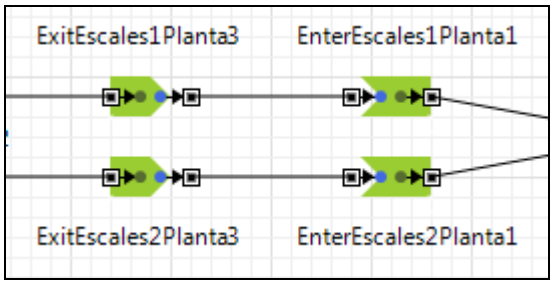


Figura 6.15: Objectes de la llibreria Pedestrian de l'Anylogic que simulen la baixada d'escales per anar de la planta 3 a la planta1.

A la figura 6.20 d'aquest capítol es mostra l'objecte P1, que és una instància de la classe PedGround, pertanyent a la llibreria Pedestrian de l'Anylogic, que defineix l'entorn físic de la planta 1, format per un conjunt de línies que simulen les parets que cap objecte Passenger pot travessar. Aquest objecte es passa com a paràmetre als objectes EnterEscales1Planta1 i EnterEscales2Planta1.

6.2.5. Modelització del procés d'embarcament

Com ja s'ha enunciat a la introducció d'aquest capítol, el model del procés de sortides de passatgers de la terminal T1 de Barcelona que s'ha realitzat en aquest projecte només pretén servir d'exemple en l'ús dels components de la llibreria que s'ha creat, i com es poden enllaçar entre ells fent servir altres objectes de llibreries que proveeix l'Anylogic.

Per aquest motiu, la modelització del procés d'embarcament s'ha simplificat, de manera que, d'una banda, només s'han inclòs al model les zones d'embarcament de la planta 1 (zones A, B i C), i d'altra banda, només s'han inclòs 3 portes d'embarcament.

A la figura 6.16 es mostra la lògica del bloc d'embarcament que s'ha dissenyat. Com es pot veure, s'ha fet ús dels següents elements:

- 3 components del tipus *Porta d'Embarcament*

  Per fer la simulació, s'ha treballat amb un fitxer de dades d'entrada del model que conté la programació real de vols d'un interval d'hores corresponent a un dia en concret, de manera que tots els vols tenen assignades portes d'embarcament a les zones A, B o C. Així, els vols s'han agrupat de la següent manera:

  - El primer component *Porta d'Embarcament* gestiona totes les portes d'embarcament de la zona A.

  - El segon component *Porta d'Embarcament* gestiona totes les portes d'embarcament de la zona B.

  - I finalment, el tercer component *Porta d'Embarcament* gestiona totes les portes d'embarcament de la zona C.

  Aquesta agrupació s'ha fet en funció de la situació real d'aquestes portes dins la terminal.

  A la figura 6.20 és mostren els objectes de la classe PedArea, pertanyent a la llibreria Pedestrian de l'Anylogic, que s'han usat per definir les àrees d'espera de que disposa cada porta d'embarcament. Aquests objectes es passen com a paràmetres d'entrada a cada component del tipus *Porta d'Embarcament* (mostrats com PE a la figura 6.16) que s'ha utilitzat en el model de la terminal.

- 1 objecte selector

  Aquest element és una instància de la classe PedSelectOutput, de la llibreria Pedestrian de l'Anylogic, que serveix per seleccionar la Porta d'Embarcament a la que s'ha de dirigir l'objecte Passenger, en funció de la lletra que contingui la numeració de la porta d'embarcament assignada al seu vol. Aquest objecte també es mostra a la figura 6.16.

- 1 franja_actual

  Aquest element és una variable entera, mostrada a la figura 6.16, que guarda la franja horària en la que l'objecte Passenger realitza l'avaluació, dins del selector, de quina porta d'embarcament ha de triar. El càlcul d'aquest enter es fa dins de l'objecte selector cada cop que entra un objecte Passenger.

  El valor d'aquesta variable es compara, dins l'objecte selector, amb l'hora de sortida del vol, pel que, si falta més d'una hora, l'objecte Passenger es dirigit cap endarrere, concretament a la lògica que modelitza el seu desplaçament lliure per les rutes d'establiments comercials que s'han modelitzat a la planta 1.

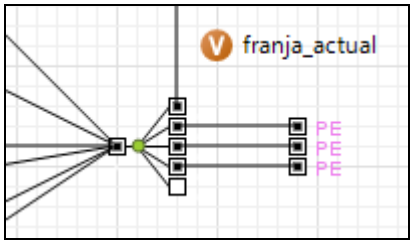


Figura 6.16: Components que modelitzen el bloc d'embarcament de la terminal.

6.2.6. Modelització del temps d'oci

El desplaçament lliure que realitzen els passatgers durant el temps d'oci de que disposen mentre estan a la terminal s'ha modelitzat mitjançant la creació de diverses rutes d'establiments comercials a cada planta. La lògica implementada a cada planta és la següent:

- <u>Lògica del desplaçament lliure de passatgers a la planta 3:</u>

Per simplificar el model, s'han inclòs només 5 tendes a la planta 3. Com és mostra a la figura 6.17, s'han utilitzat 5 objectes de la classe PedArea, de la llibreria Pedestrian de l'Anylogic, per definir l'espai físic que cada tenda té disponible al públic.

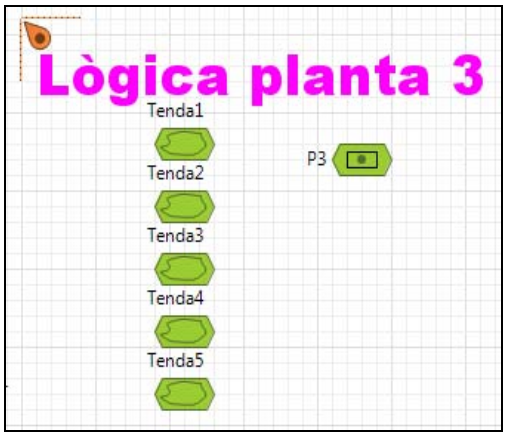


Figura 6.17: Objectes de la llibreria Pedestrian usats en la modelització de les rutes d'establiments comercials de la planta 3.

Dins la lògica implementada per a la planta 3, s'han considerat dos moments en els que els passatgers poden realitzar aquest desplaçament lliure:

a) Entre el bloc d'entrada de passatgers a la terminal i el bloc de facturació:

Entre aquests dos blocs s'han definit dues rutes de tendes, fent ús d'un component *ShopsTwo* i d'un component *ShopsThree*, que tenen definides dues i tres tendes, respectivament.

Tal i com es mostra a la figura 6.18, aquests dos components estan connectats a un objecte de la classe PedSelectOutput de la llibreria Pedestrian de l'Anylogic, anomenat Shopping_Before_CkekIn, que, amb unes determinades probabilitats, dirigeix els passatgers cap a una ruta o una altra, per tal de simular el comportament estocàstic que segueixen els passatgers a l'hora de triar visitar unes tendes o unes altres.

A més, com ja s'ha explicat amb anterioritat en aquest capítol, s'ha fet ús d'un altre objecte de la classe PedSelectOutput, anomenat ChekIn_Or_Shopping, pel que s'encaminen els objectes Passenger cap a la lògica del procés de facturació, o bé cap a la lògica del temps d'oci, amb certes probabilitats. Aquestes probabilitats són aproximades, tenint en compte que en la realitat, la majoria de passatgers que arriben a la terminal decideixen facturar abans que visitar tendes.

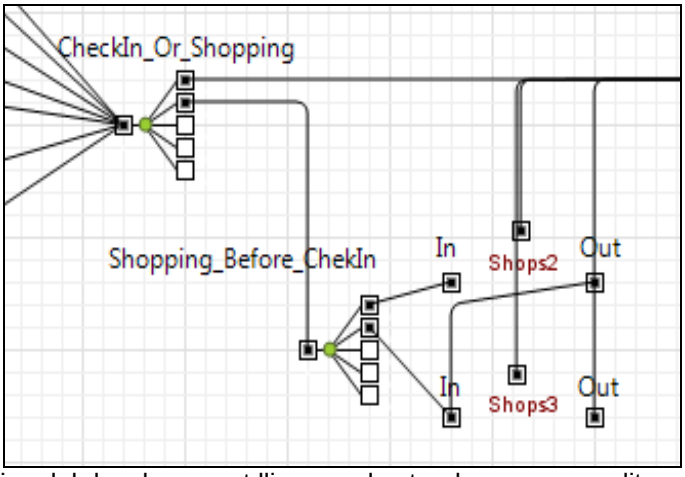


Figura 6.18: Lògica del desplaçament lliure per les tendes que es realitza abans de facturar

b) Entre el bloc de facturació i el bloc de pas per filtres de seguretat:

Tal com es mostra a la figura 6.19, la lògica d'aquest desplaçament lliure de passatgers és anàloga a l'explicada a l'apartat a). Així, consta de dos objectes selectors: Shopping_Or_Filters i Shopping_Before_Filters. El primer selector permet que cada objecte Passenger decideixi si s'ha d'encaminar cap al següent bloc del procés de sortides (pas per filtres) o bé realitzar un desplaçament lliure abans d'efectuar-lo. El segon selector, en cas que la primera opció sigui la de realitzar un desplaçament lliure, permet simular un comportament estocàstic en la tria de la ruta comercial a seguir.

S'han definit dues rutes comercials, fent ús d'un component *ShopsTwo* i d'un component *ShopsThree*, que tenen definides dues i tres tendes, respectivament, i que són diferents a les definides en les rutes comercials explicades a l'apartat a).

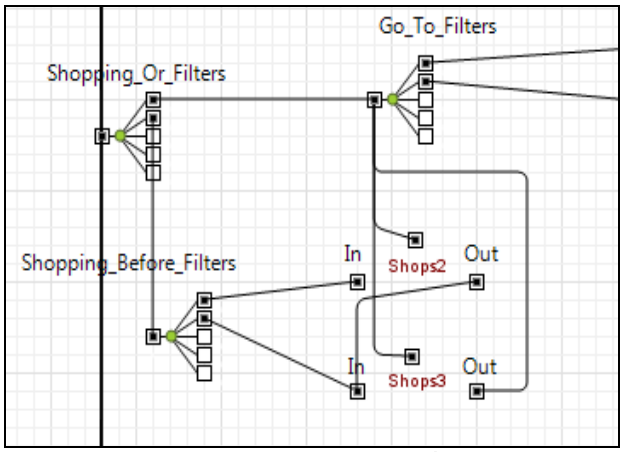


Figura 6.19: Lògica del temps d'oci entre la facturació i el pas pels filtres de seguretat.

- Lògica del desplaçament lliure de passatgers a la planta 1:

Quan els passatgers travessen la zona de filtres de seguretat, agafen les escales per baixar a la planta 1. En aquesta planta realitzaran l'embarcament, però mentre el temps que els hi resta fins l'hora d'embarcament és major que un cert llindar (paràmetre del model), es dediquen a passejar per les tendes que la terminal té a la planta 1.

Igual que per a la planta 3, per tal de simplificar el model, només s'ha considerat un subconjunt dels establiments comercials que té la terminal en aquesta planta. Concretament s'han modelitzat 14 tendes i 2 bars/restaurants.

S'han utilitzat 16 objectes de la classe PedArea, de la llibreria Pedestrian de l'Anylogic, per definir l'espai físic que té disponible al públic cada establiment comercial. Aquests objectes es poden veure a la figura 6.20.

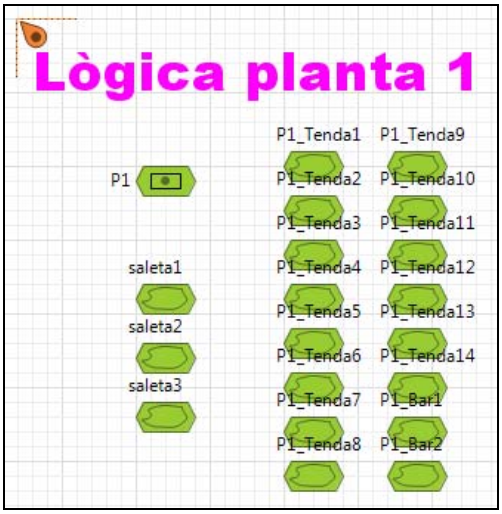


Figura 6.20: Objectes de la llibreria Pedestrian usats en la modelització de les rutes d'establiments comercials i zones d'espera d'embarcament de la planta 1.

Tal i com es pot veure a la figura 6.21, a la lògica pròpia de la planta 1 s'han definit dues rutes comercials, amb 8 establiments cadascuna, mitjançant l'enllaç de components del tipus *ShopsFive* i *ShopsThree*.

A més, les dues rutes estan connectades a un objecte selector de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, anomenat Route_Shopping que conté les probabilitats de que un objecte Passenger agafi una ruta o l'altra. D'aquesta manera es simula el comportament estocàstic del desplaçament lliure per la planta 1 que realitzen els passatgers mentre fan temps per dirigir-se cap a la porta d'embarcament.

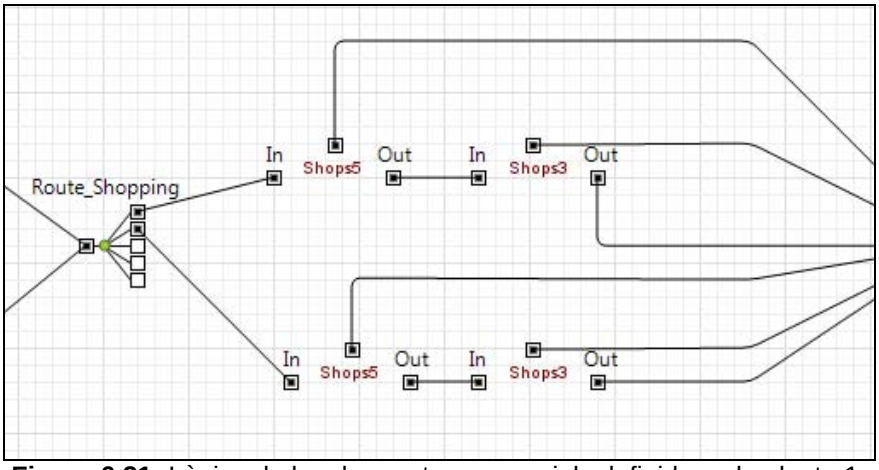


Figura 6.21: Lògica de les dues rutes comercials definides a la planta 1.

6.2.7. Procés d'inicialització

Els processos d'inicialització de la simulació que s'han creat en la modelització de la terminal T1 de Barcelona, es divideixen en els quatre blocs següents:

- Connexió a la base de dades:

  La connexió al fitxer de dades s'ha realitzat fent ús de l'objecte database, de la classe Database de l'Anylogic, tal i com es mostra a la figura 6.22.

- Càrrega de dades al simulador:

  Per carregar al simulador les dades d'inicialització que conté el fitxer extern, que són: les taules d'arribades de passatgers, les probabilitats d'arribades de passatgers per cada font i la programació de vols del dia que es vulgui simular, s'han de cridar, dins el paràmetre *Startup Code* de l'objecte Main del model, les següents funcions:

  - get_sources_number()
  - get_sources_probabilities()
  - get_Flights_list()
  - set_Passengers_Arrivals()

Tal i com es mostra a la figura 6.22, aquestes funcions s'han creat dins l'editor gràfic de l'objecte Main del model de la terminal.

El detall de la funcionalitat i els paràmetres d'entrada que usen aquestes funcions es troben explicats al capítol 5 d'aquesta memòria, concretament dins l'apartat *5.1.3. Components Comuns*, i el codi Java que s'ha implementat a cada funció es pot consultar en format electrònic a l'annex 2.

- Inicialització de les taules d'arribades de cada font:

  Com ja s'ha explicat en aquest capítol, per a la modelització de la terminal T1 de Barcelona s'han usat dos components *Font de passatgers*, on cadascun d'ells usa la seva pròpia taula d'arribades. Per tant, en els processos d'inicialització s'han creat dues taules d'arribades, anomenades Inputs 1 i Inputs2, tal i com es mostra a la figura 6.22, que són objectes de la classe TableFunction de l'Anylogic.

  Per carregar el nombre d'arribades de passatgers a cada franja horària que ha de contenir cada taula, s'ha fet ús de la següent funció:

  InitializeArrivalTables()

  Aquesta funció s'ha creat dins l'editor gràfic de l'objecte Main del model de la terminal, tal i com mostra la figura 6.22.

  El detall de la funcionalitat i els paràmetres d'entrada que usa aquesta funció es troben a l'apartat *5.1.3.* i el codi Java que s'ha implementat dins d'aquesta funció es pot consultar en format electrònic a l'annex 2.

- Creació de llistats de recursos:

  Dins l'editor gràfic de l'objecte Main del model de la terminal, com es pot veure a la figura 6.22, s'han creat 6 objectes de la classe Collection de l'Anylogic que contenen els llistats següents:

  - Objecte Flights_list: conté el llistat d'objectes Flight que s'han llegit de la programació de vols del fitxer de dades extern.

    Aquest llistat s'omple durant el procés de càrrega de dades al simulador, mitjançant la funció get_Flights_list().

  - Objecte Hours-And_Flights_list: conté una relació dels vols que intervenen en l'arribada de passatgers a cada franja horària.

    Igual que amb l'anterior, aquest llistat s'omple durant el procés de càrrega de dades al simulador, mitjançant la funció get_Flights_list().

  - Objecte Counters_list: conté el llistat de components de tipus Taulell de *Facturació* que usa el model.

Aquest llistat s'omple dins el paràmetre *Startup code* de l'objecte Main del model.

- Objecte TMF_list: conté el llistat de components de tipus *Taulell de MultiFacturació* que usa el model.

  Aquest llistat s'omple dins el paràmetre *Startup code* de l'objecte Main del model.

- Objecte Llistat_Filtres1: conté el llistat de components de tipus *Filtres de Seguretat* que usa el model, i que estan connectats al primer component *Serpentí* de la lògica que modelitza la zona de filtres de seguretat de la terminal.

  Aquest llistat s'omple dins el paràmetre *Startup code* de l'objecte Main del model.

- Objecte Llistat_Filtres2: conté el llistat de components de tipus *Filtres de Seguretat* que usa el model, i que estan connectats al segon component *Serpentí* de la lògica que modelitza la zona de filtres de seguretat de la terminal.

  Aquest llistat s'omple dins el paràmetre *Startup code* de l'objecte Main del model.

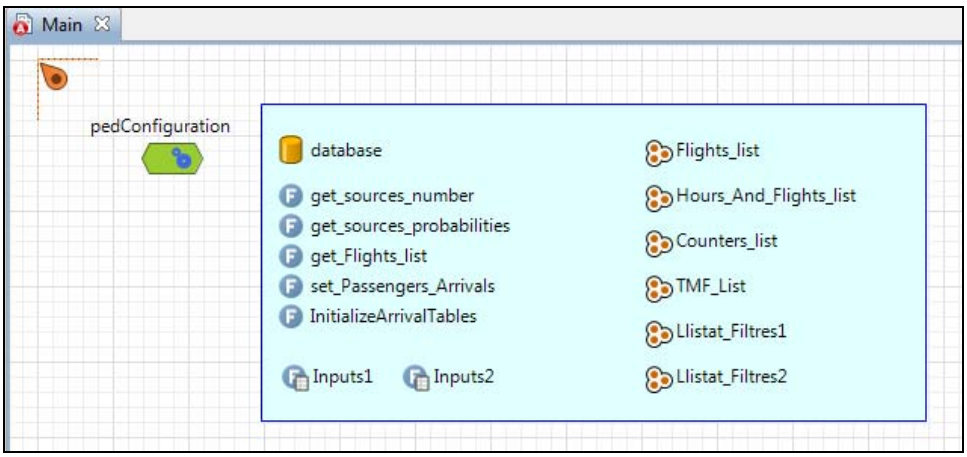


Figura 6.22: Objectes de l'Anylogic usats en el procés d'inicialització.

6.2.8. Validació del resultat de la simulació

A l'hora de validar el resultat de la simulació, s'ha de verificar el següent:

Per una banda, cal assegurar el correcte funcionament dels components que s'han utilitzat en el model; és a dir, comprovar que durant la simulació els objectes Passenger es comporten segons la lògica implementada.

Aquesta comprovació s'ha fet de manera visual, durant diverses simulacions.

Per altra banda, cal verificar que el recompte de passatgers que han finalitzat amb èxit el procés de sortides (han embarcat a l'avió que els corresponia), més el nombre de passatgers que han perdut el seu vol, és igual al nombre total de passatgers que han estat creats durant la simulació. D'aquesta manera s'assegura que l'execució de la simulació no genera cap pèrdua de passatgers.

Per realitzar aquesta segona comprovació s'ha introduït codi Java de control dins la lògica del model, que s'encarrega de comptar tant el nombre de passatgers creats durant la simulació que finalment embarquen, com el nombre de passatgers que surten del sistema perquè, o bé no han arribat a temps a facturar, o bé han arribat tard a la porta d'embarcament, i per tant han perdut el seu vol.

S'han creat dues variables que guarden els dos recomptes:

PassatgersFinalitzacióSenseExit: acumulat que guarda del nombre de passatgers que surten del sistema sense haver pogut finalitzar el procés de sortides.

PassatgersFinalitzacióAmbExit: acumulat que guarda el nombre de passatgers que surten del sistema havent pogut finalitzar amb èxit el procés de sortides.

Segons el bloc del procés de sortides modelitzat, el codi de control que s'ha implementat és el següent:

Bloc de facturació:

Pel cas de la facturació que es realitza en un mostrador individual, s'ha implementat codi de control dins l'objecte Facturació.

Concretament, l'element Facturació disposa d'un objecte de la classe PedSink, de la llibreria Pedestrian de l'Anylogic, al que van a parar tots els objectes Passenger que no tenen cap mostrador de facturació disponible en el moment en que fan la consulta de cap a quin mostrador han d'anar a facturar mitjançant la crida al mètode GoToCounter().

Dins d'aquest objecte PedSink s'ha implementat un codi de control que s'encarrega de llegir el color dels objectes Passenger que hi entren, abans d'eliminar-los del sistema, i actualitzar la variable *PassatgersFinalitzacióSenseExit.*

Pel cas de la facturació que es realitza fent ús d'un grup de multifacturació, s'ha implementat codi Java de control dins l'objecte selector de la classe PedSelectOutput, pertanyent a la llibreria Pedestrian de l'Anylogic, que connecta amb el Serpentí del grup de multifacturació que s'ha modelitzat.

Concretament, dins d'aquest selector, cada objecte Passenger consulta la disponibilitat dels mostradors del grup de multifacturació, i en cas que en el moment de la consulta no n'hi hagi cap de disponible, s'encamina cap a un objecte PedSink.

De manera anàloga al procés de facturació individual, dins l'objecte PedSink s'actualitza la variable *PassatgersFinalitzacióSenseExit.*

Bloc d'embarcament

Com ja s'ha explicat al capítol 5, la lògica interna de cada component del tipus *Porta d'Embarcament*, disposa de dues instàncies de la classe PedSinK, que són pedSink i pedSink1. Aquests dos objectes s'encarreguen de treure els objectes Passanger del sistema. L'entrada dels objectes Passenger dins d'un d'aquests dos objectes depèn del tipus de finalització del procés de sortides que tinguin; és a dir, finalització amb èxit, en cas que puguin embarcar, o sense èxit, en cas que hagin perdut el vol.

El codi de control que s'ha implementat dins de l'objecte pedSink1 actualitza la variable *PassatgersFinalitzacióSenseExit,* en funció del color de l'objecte Passenger que llegeix abans d'eliminar-lo del sistema.

En canvi, el codi de control implementat dins de l'objecte pedSink, s'encarrega d'actualitzar la variable *PassatgersFinalitzacióAmbExit* en funció del valor de la lectura del color de cada objecte Passenger que ha d'eliminar del sistema.

7. Conclusions

En aquest projecte s'ha dissenyat i implementat una llibreria de components reutilitzables que permeten la modelització del procés de sortides d'un aeroport comercial amb l'eina de simulació Anylogic.

Aquesta llibreria permet la modelització del procés de sortides, tant des d'un punt de vista de microsimulació, en el que es pot veure el moviment individual de cada passatger, com des d'un punt de vista de macrosimulació, on es pot apreciar la interacció dels passatgers entre ells i amb els elements que composen l'entorn propi del procés de sortides: taulells de facturació, filtres de seguretat, portes d'embarcament, etc.

A més, permet la parametrització de les condicions de contorn del model a simular, com són la programació de vols i l'assignació de mitjans (taulells de facturació, filtres de seguretat i portes d'embarcament) d'un determinat aeroport per un determinat dia.

Tot això fa aquesta llibreria faciliti una avaluació, de manera simple, de les diferents polítiques d'assignació de mitjans de que disposa l'aeroport que s'estigui modelant, així com el seu impacte sobre factors de qualitat i KPI's rellevants establerts per l'aeroport.

Per tant, es pot dir que els objectius que es van definir a l'inici del projecte s'han complert en la seva totalitat.

En relació a la metodologia de desenvolupament del projecte, en primer lloc cal remarcar la importància de l'etapa de planificació. En aquest sentit, podem dir que la planificació realitzada ha estat una referència molt útil pel desenvolupament del projecte i, tal com es va comentar al capítol 2, en general s'ha seguit de manera bastant acurada.

La principal desviació s'ha produït en la redacció de la memòria, que ha necessitat un 56,7% més del treball planificat. Això serà un referent valuós de cara a futurs treballs d'aquest tipus.

La tasca de codificació (en aquest cas, dels components de la llibreria), que sovint és la que més s'allunya, per excés, de la planificació, només ha necessitat un 9,5 % més del treball planificat.

Addicionalment, altres lliçons apreses durant l'execució del projecte serien les següents:

- L'Anylogic és una eina que facilita la modelització de sistemes orientats a esdeveniments discrets, ja que disposa de llibreries predeterminades molt útils per aquesta finalitat.

  En aquest sentit, la parametrització que ofereixen els objectes d'aquestes llibreries, així com la possibilitat de combinar-los per formar objectes amb un comportament específic diferent a l'ofert a priori, ha permès la creació

dels diferents components de la llibreria que ha donat com a resultat aquest projecte.

- En les diferents simulacions que s'han realitzat del model de la terminal T1 de l'Aeroport de Barcelona, que s'ha creat com exemple d'ús de la nova llibreria, la càrrega de treball, pel que fa al nombre de passatgers que arriben al terminal per agafar un vol i als recursos disponibles, tant per facturar, com per passar per la zona de seguretat, ha estat representativa de la realitat d'aquest procés.

  L'equip informàtic usat a les simulacions és un portàtil HP Pavilion Entertainment PC amb dos processadors Intel Core i5 M450 a 2.27 GHz cadascun, i una memòria RAM de 4 GB. El sistema operatiu instal·lat a l'equip és Windows 7 de 64 bits.

  En aquestes condicions, ha quedat palesa la necessitat de disposar d'un bon equipament informàtic que pugui satisfer els forts requeriments de processament d'aquest tipus d'aplicacions (simuladors).

- La consulta dels tutorials i manuals de que disposa l'Anylogic, tant en el web del fabricant, com a l'ajuda de l'aplicació, ha resultat molt útil tant a la fase d'aprenentatge de l'eina com a la fase de creació dels components de la llibreria, sobretot durant el procés de tria dels objectes més apropiats pel disseny de cada component.

Finalment, aquest projecte podria tenir diverses línies de continuació. En relació a la pròpia llibreria de components del procés de sortides, una possibilitat seria l'ampliació de la lògica del component *Filtre de Seguretat*, per tal de permetre la modelització en detall de tots els passos que segueix un passatger des de que s'acosta al filtre de seguretat fins que surt i, d'aquesta manera, possibilitar la minimització del temps mig d'aquest subprocés.

Una altra línia de continuació força interessant i que, personalment, m'hauria agradat realitzar, és la finalització del model complet del procés de sortides de la terminal T1 de l'Aeroport de Barcelona, i la seva posterior validació amb dades reals de temps de processament dels diferents subprocessos. Això, d'una banda, no era objectiu d'aquest projecte i, d'altra banda, per limitacions de temps, no ha estat possible fer-ho.

8. Bibliografia

Material imprès

- Deitel H.M., Deitel P.J. *Como programar en Java*. Prentice Hall, México, 2004.

- Guasch A., Piera M.A., Casanovas J., Figueras J. *Modelado y simulación. Aplicación a procesos logísticos de fabricación y servicios.* Edicions UPC, Barcelona, 2002.

- Reselman B., Peasley R., Prunchniak W. *Descubre Visual Basic 6*. Prentice Hall Iberia, Madrid, 1999.

- Tejada I. *Descubrir los aeropuertos.* Centro de Documentación y Publicaciones de Aena, 2002.

Enllaços web

- Pàgina oficial d'Aena, on hi ha informació de la terminal T1 de l'Aeroport de Barcelona: http://www.aena.es/csee/Satellite?pagename=Home

- Manuals i tutorials d'Anylogic: http://www.xjtek.com/

- Eines de simulació del moviment de persones:

  - SIMWALK TRANSPORT: http://www.simwalk.com/

  - STEP: http://www.mottmac.com/skillsandservices/software/stepssoftware/

  - VISSIM: http://www.vissim.com/