Resum

Aquest projecte pretén elaborar una eina per a la generació automàtica de codi en la programació de robots, a partir de les xarxes de Petri. Les xarxes de Petri són una eina gràfica formada per llocs, arcs i transicions que, a part de moltes altres finalitats, permet la modelització de cel·les de treball on intervenen tot tipus d'elements automatitzats, principalment robots manipuladors i elements de transport i emmagatzematge.

Es vol que el programa permeti l'assimilació de l'entorn real de treball a la xarxa de Petri, dotant-la de les característiques necessàries per a la diferenciació de varis robots i la capacitat d'emmagatzemar i generar posteriorment el codi per programar-los, deixant per a l'usuari només la feina referent a la introducció del codi relacionat estrictament amb el moviment del robots.

D'altra banda es vol que estigui desenvolupat com a programari lliure i que també es desenvolupi a través d'aquest.

Es parteix d'un programa lliure que permet l'edició i anàlisi de xarxes de Petri i es desenvolupa un mòdul totalment integrat a l'aplicació que genera el codi de programació de forma automàtica.

Aquest codi generat és capaç d'implementar tots els elements necessaris per tal d'establir i mantenir la comunicació entre els robots, eximint a l'usuari d'aquesta feina complexa. També és capaç de gestionar espais i recursos accedits per varis robots, així com d'executar tasques en paral·lel al fil principal de cada robot per tal de realitzar càlculs o l'activació de dispositius externs.

Sumari

1. Introducció

1.1. Objectius del projecte

L'objectiu principal d'aquest projecte de fi de carrera és desenvolupar una eina de programació de robots a partir de les xarxes de Petri que modelen el sistema en estudi. Concretament es parteix d'un programa de software lliure que permet el modelat i anàlisi de les xarxes de Petri i se n'amplien les seves funcions a través d'un mòdul. Es tracta d'aprofitar les capacitats que proporcionen les xarxes de Petri tan de síntesi com d'anàlisi per crear el que serà l'esquelet del programa que gestionarà el comportament dels robots. A més, amb aquest projecte es facilitarà la programació de robots que treballen en una cel·la compartint diferents recursos, així com també el mateix espai de treball.

En el capítol dos d'aquest projecte es fa una introducció a les xarxes de Petri, les seves aplicacions i la seva relació amb els robots. El capítol tres està destinat a explicar l'elecció del programa amb que es desenvoluparà el projecte i els requeriments d'aquest per tal d'aconseguir les noves capacitats. En el capítol quatre s'expliquen els robots als que es pot aplicar aquest projecte i la seva programació. El cinquè presenta el mòdul de programació creat, les característiques del codi que genera i es fa una validació experimental del projecte on es posa de manifest de manera més entenedora la seva utilitat. Seguidament en el sisè capítol s'explica la metodologia seguida en l'elaboració del projecte així com els llenguatges de programació necessaris. La implementació del mòdul de programació i les modificacions fetes al programa original es presenten en el capítol set. Finalment les conclusions del treball recullen una valoració del projecte realitzat.

1.2. Abast del projecte

Aquest projecte està pensat per a ser utilitzat en una cel·la de treball on intervinguin varis robots que es puguin programar amb el llenguatge de programació escollit. En concret, està pensat perquè treballi amb els robots dels laboratoris de l'Institut d'Organització i Control situat a la planta 11 de l'ETSEIB, on es disposa de dos braços robotitzats de la marca Stäubli, però és extensible a qualsevol cel·la de treball que utilitzi aquests robots.

2. Les xarxes de Petri

2.1. Què són les xarxes de Petri?

Les xarxes de Petri són un formalisme matemàtic basat en simples objectes, relacions i regles que permet representar comportaments complexes. Són especialment útils per modelar i analitzar sistemes dinàmics basats en esdeveniments discrets que tenen evolucions en paral·lel i comportaments caracteritzats per sincronitzacions i compartiment de recursos [1].

Un model en xarxa de Petri d'un sistema dinàmic consta de dues parts:

1- L'estructura de la xarxa: que és un graf dirigit, bipartit i ponderat que representa la part estàtica del sistema.

2- El marcat: que representa la part dinàmica del sistema.

L'estructura de la xarxa està formada per dos tipus d'objectes que es relacionen amb un flux ponderat: els *llocs*, que representen l'estat del sistema, i les *transicions*, que representen els canvis d'estat (*Figura 2.1, a* i *b*).

El marcat és l'assignació d'una sèrie de marques a alguns dels llocs de l'estructura de la xarxa (*Figura 2.1, c*). Aquestes marques indiquen que una acció està realitzant-se o que un recurs es troba disponible.

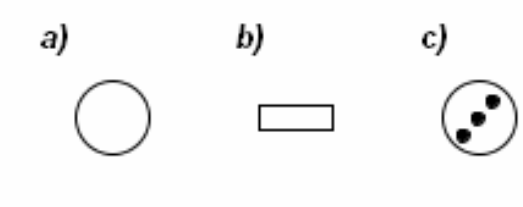


Figura 2.1. Símbol corresponent a:
a) un lloc; b) una transició; c) un lloc amb tres marques

L'evolució d'aquestes marques es defineix a través de la regla d'ocurrència o dispar, anomenada *joc de marques*.

Segons aquesta regla:

1- Una transició està habilitada per un determinat marcat si tots els seus llocs d'entrada tenen tantes marques (o més) com el pes de l'arc que connecta amb la transició. En la *Figura 2.2* es pot veure un exemple de transició habilitada i de transició no habilitada.

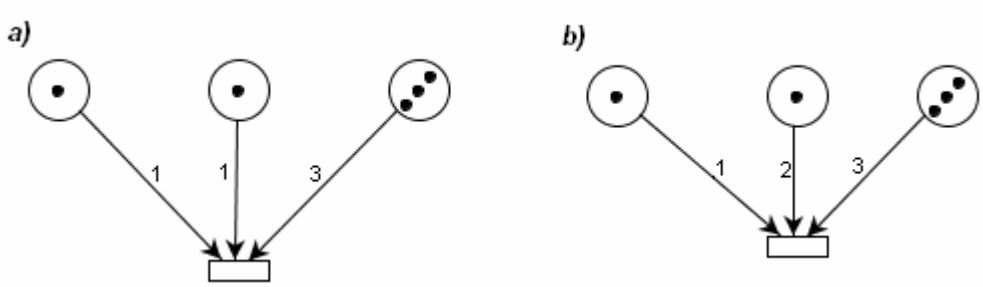


Figura 2.2 Exemple de transició: a) habilitada, b) no habilitada

2- El dispar d'una transició provoca que, de forma instantània, es tregui de cada lloc d'entrada un nombre de marques igual al pes de l'arc de la connexió i es posi a cada lloc de sortida un nombre de marques igual al pes de l'arc de la connexió. Si el pes no està indicat, se sobreentén que és u. En la *Figura 2.3* es pot veure un exemple de dispar.

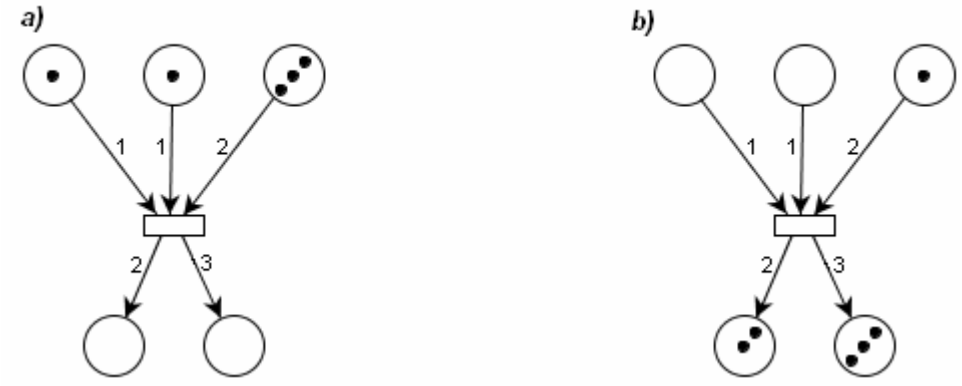


Figura 2.3 Exemple de dispar d'una transició:
a) Situació abans del dispar; b) Situació després del dispar

2.2. Definicions i característiques

2.2.1. Definicions

Un cop vist el funcionament bàsic de les xarxes de Petri cal conèixer algunes de les característiques bàsiques que les defineixen per entendre millor aquest projecte.

Les funcions de *pre-incidència* (o d'entrada a les transicions) i de *post-incidència* (o de sortida de les transicions) descriuen les connexions entre els llocs i les transicions de manera formal i se solen representar en forma de matrius. Si una xarxa no forma llaços llavors es pot caracteritzar per la ***matriu d'incidència C***, resultat de restar la pre-incidència a la post-incidència, on els valors positius representen les post-incidències i els valors negatius les pre-incidències.

Un ***P-Invariant*** és un vector X que compleix l'expressió $X^TC = 0$. Indica que el nombre de marques en el conjunt de llocs senyalat es manté constant. Així mateix, un vector Y és un ***T-Invariant*** si CY = 0. Indica el conjunt de la xarxa que té un comportament de marcat cíclic.

2.2.2. Propietats Qualitatives

Acotació *("boundness"):* Un sistema està acotat si l'espai d'estats és finit, sabent que el concepte cota es refereix al nombre màxim de marques d'un lloc.

Vivacitat *("liveness"):* Un sistema té la propietat de vivacitat si des de qualsevol marcat assolible existeix una seqüència de dispars que habilita qualsevol transició. Que un sistema no tingui bloquetjos ("deadlocks") és una restricció més fluixa que la vivacitat ja que només verifica que alguna transició es pugui disparar.

Reversibilitat *("reversibility"):* Un sistema és reversible si des de qualsevol marcat assolible es pot tornar al marcat inicial. Si un sistema té bloquetjos, no és reversible.

Exclusió Mútua *("mututal exclusion"):* En un sistema existeix exclusió mútua si:

- no es poden disparar simultàniament dues transicions (t-mutex)
- no es poden marcar simultàniament dos llocs (p-mutex)

2.2.3. Classificació

Una xarxa de Petri és **no autònoma** si se la dota d'interpretació, és a dir, si s'associa una semàntica a les entitats de la xarxa i a les condicions d'evolució. La interpretació considera l'entorn del sistema. Altrament, la xarxa és **autònoma**. Partint d'aquesta classificació general s'obté el següent esquema, on es mostra breument els diferents tipus de xarxes de Petri:

1-*Xarxes autònomes*:

1.1- *Xarxes de Petri Ordinàries*. Són aquelles en que tots els pesos valen 1, només hi ha un tipus de marca, la capacitat dels llocs no està limitada i els dispars de les transicions poden ocórrer si els corresponents llocs estan habilitats.

1.1.1- *Màquina d'estats o Graf d'estats (SG)*: cada transició té exactament un lloc d'entrada i un de sortida.

1.1.2- *Graf d'esdeveniments o Graf Marcat (MG)*: cada lloc té exactament una transició d'entrada i una de sortida.

1.1.3- *Xarxa d'elecció lliure (FCN)*: són un compromís entre els dos tipus anteriors sempre i quan no s'interfereixin:

- Si dos llocs comparteixen transició de sortida, aquesta és l'única transició de sortida.

- Si dos transicions comparteixen un mateix lloc d'entrada, aquest és l'únic lloc d'entrada.

1.1.4- *Xarxes simples (SN)*: cada transició té com a molt un lloc compartit amb d'altres transicions.

1.2- *Abstraccions*: són representacions abreviades per ressaltar la part gràfica. Es pot trobar la corresponent xarxa de Petri ordinària.

1.2.1- *Xarxes de Petri generalitzades*: consideren pesos més grans que 1.

1.2.2- *Xarxes de Petri colorejades*: consideren diferents tipus de marques.

1.2.3- *Xarxes de Petri amb capacitat finita*: consideren un nombre finit de marques en cada lloc.

2- *Xarxes no autònomes*:

2.1- *Xarxes de Petri Sincronitzades*: les transicions es disparen (si estan habilitades) quan ocorre un esdeveniment.

2.2- *Xarxes de Petri Interpretades*: les transicions es disparen quan ocorre un esdeveniment i es compleix una condició.

2.2.1- *Xarxes de Petri Condició/Esdeveniment*: xarxes de Petri interpretades i que contenen marcat booleà (Grafcets).

2.3- *Xarxes de Petri temporitzades (Timed PN)*: consideren la duració de les accions (associades als llocs o a les transicions).

2.4- *Xarxes de Petri estocàstiques*: es considera un temps aleatori al disparar les transicions i se'ls associa un procés de Markov.

2.3. El modelat amb xarxes de Petri

A l'hora de poder avaluar un sistema físic s'acostuma a emprar un model amb el que poder fer assajos. El poder disposar d'un model permet fer assajos que poden estalviar malmetre el material original. Per a que això sigui cert, però, cal que el model s'ajusti el més possible a la realitat.

Sovint els resultats no poden ser traslladats a la realitat ja sigui per poca precisió del model, com per la naturalesa de la prova que faci que fins que no es realitza un assaig sobre el sistema real no es pugui garantir la seva fiabilitat.

Les xarxes de Petri són considerades com una metodologia per modelar els sistemes de fabricació flexibles. En aquest apartat s'estudia amb més detall les capacitats de les xarxes de Petri en el modelat de sistemes de fabricació.

Per modelar un sistema amb xarxes de Petri s'ha de saber quina interpretació se li dóna a cada element. Aquests elements però, no poden tenir qualsevol significat sinó que estan delimitats. Així els llocs d'una xarxa de Petri poden representar dues coses: activitats o recursos.

Activitats: Una marca en un lloc indica que l'activitat s'està realitzant. Aquests llocs s'anomenen de tipus A. Un camí que conté només llocs de tipus A s'anomena camí de tipus A.

Recursos: Un nombre determinat de marques indica el nombre de recursos disponibles. Quan el lloc no està marcat indica que el recurs no està disponible. Es classifiquen en llocs tipus B, que representen recursos fixes com màquines, robots, etc., i llocs tipus C que representen recursos variables com palets, fixacions, etc.

Per altra part, les transicions només solen representar l'inici o la fi d'una operació.

Per procedir a la descripció d'un sistema mitjançant una xarxa de Petri cal seguir els següents passos: 1. Identificar les activitats i recursos requerits per fabricar el producte. 2. Ordenar les activitats per les relacions de precedència. 3. Per a cada activitat, crear un lloc, que descriu l'activitat, i una transició d'entrada i una de sortida, que descriuen l'inici i la fi de l'activitat. 4. Per a cada activitat, crear els llocs (si encara no existeixen) dels recursos que necessita per començar i connectar-los a la transició d'entrada. Fer l'equivalent amb els recursos que allibera, connectant la transició de sortida a ells. 5. Especificar el marcat inicial.

Una altra manera de construir el model en xarxa de Petri és la síntesi híbrida, que és un procediment que tracta separadament els llocs de tipus A dels llocs de tipus B i C. El

procediment és un enfoc combinat jeràrquic i modular que permet introduir els detalls de forma incremental i que garanteix el compliment de les propietats qualitatives bàsiques. Aquest procediment consta de dues fases:

Fase 1 *(Top-down):* S'inicia amb una xarxa d'alt nivell acotada, viva i reversible que captura l'estructura fonamental del sistema. Després se substitueixen els llocs i transicions per subxarxes, tot garantint la preservació de les propietats qualitatives. Aquestes propietats es poden garantir usant mòduls bàsics de llocs i transicions, tals com mòduls de seqüència per a operacions successives, mòduls paral·lel per a operacions que comencen simultàniament, mòduls de conflicte per representar diferents opcions d'una operació, mòduls d'alternança que representen varies alternatives per una operació que es succeeixen en ordre, etc. En aquest pas es van incrementant els llocs de tipus A. Seguidament s'afegeixen els llocs corresponents a recursos no compartits.

Fase 2 *(Bottom-up):* S'afegeixen els magatzems, els quals també es modelen com una subxarxa que preserva les propietats mitjançant mòduls concrets per a cada tipus de magatzem. Després s'afegeixen els recursos compartits que formen exclusions mútues en paral·lel i seguidament els recursos compartits que formen exclusions mútues en sèrie. L'ús de recursos compartits pot donar lloc a problemes de bloquejos si no es garantitza que quan un procés accedeix al recurs, sigui capaç d'alliberar-lo una vegada finalitzada l'activitat.

2.4. Aplicació al projecte

Com s'ha mencionat en l'apartat anterior, les xarxes de Petri són considerades com una eina per l'estudi dels sistemes. Amb la seva ajuda es pot modelar el comportament i l'estructura d'un sistema, i portar el model a condicions límit, que en un sistema real són difícils d'aconseguir o molt costoses. La teoria de xarxes de Petri ha arribat a ser reconeguda com una metodologia establerta en la literatura de la robòtica per modelar els sistemes de fabricació flexibles. Les xarxes de Petri ofereixen una forma d'expressar processos que requereixen sincronia, element indispensable en la fabricació flexible. També són molt interessants ja que poden ser analitzades de manera formal i obtenir informació del comportament dinàmic del sistema modelat.

Una idea fonamental en un sistema és que es composa de mòduls que interactuen entre sí, els quals poden ser considerats per sí mateixos un sistema, i es pot estudiar el seu comportament per separat i d'aquesta manera aïllar-los, però sempre tenint en compte la interacció que guarden amb els altres. Aquests mòduls presenten unes condicions internes que determinen l'estat en que es troben, així s'entén que un sistema sigui un arranjament dinàmic que en el transcurs del temps té variacions i no roman estàtic. Donat que els mòduls interactuen entre sí, les accions de varis mòduls poden ocórrer simultàniament i es fa

necessari sincronitzar els esdeveniments. Això pot resultar en que les condicions d'un mòdul en el temps necessiten com a entrades les sortides d'altres mòduls, els quals necessiten més temps per generar les sortides. Aquí és on intervenen el paral·lelisme i la concurrència, elements que també es representen mitjançant xarxes de Petri.

Per modelar un sistema s'han de conèixer les condicions i els esdeveniments que s'hi donen, d'aquesta manera es pot fer una analogia entre el sistema i el model, i relacionar cada lloc i transició de la xarxa amb les corresponents accions i condicions.

Així s'entén que les xarxes de Petri siguin una eina adequada pel modelat de sistemes de fabricació flexible, ja que amb elles es té l'habilitat de representar de manera fàcil la concurrència, la causalitat, la sincronització, la compartició de recursos, etc. Existeix la possibilitat d'associar als objectes del model un ampli rang de significats diferents, el que permet tenir una eina comú per les diferents fases del disseny i operació dels sistemes de fabricació flexible. A més, la representació gràfica facilita la documentació i la monitorització, i la semàntica, formal i precisa, permet un rigorós anàlisi, així com l'automatització de la generació de codi pel control o per la simulació.

El sistema genèric d'estudi d'aquest projecte es correspon a una cel·la de treball on intervenen tot tipus d'elements automatitzats, principalment robots manipuladors i elements de transport i emmagatzematge. Per programar les tasques que duran a terme aquests robots es necessita modelar els moviments de tot el sistema, incloent també la interacció dinàmica entre els diferents mecanismes, trajectòries planejades, punts d'assignació, etc. Com s'ha explicat, aquesta és una feina que es pot facilitar a través del model del sistema en xarxa de Petri.

Es tracta doncs, d'agrupar les possibilitats que ens aporten les xarxes de Petri en el modelat i anàlisi dels sistemes amb la necessitat d'obtenir un codi de programació que controli el comportament dels robots i elements que hi intervenen i gestioni de manera eficient els recursos, ja siguin cintes transportadores, magatzems o espais de treball, facilitant així tant el modelat i l'anàlisi com la programació en una sola eina. És per aquest motiu que es parteix de la xarxa de Petri que representa el sistema d'estudi en qüestió i es desenvolupa una eina per a la generació automàtica de codi per a la programació dels robots i el control dels elements que hi intervenen.

3. El software

Un cop conegut el punt de partida del projecte, s'ha d'escollir la plataforma amb la que es durà a terme. Es requereix d'un programa capaç de suportar l'eina que es desenvoluparà i que ajudi a fer del modelat i l'anàlisi del sistema una feina senzilla i intuïtiva, aportant el màxim de prestacions addicionals possibles.

Degut a que en els objectius del projecte no figura el de desenvolupar tot el programa de modelat i anàlisi de les xarxes de Petri, sinó desenvolupar només la part que permeti la generació del codi de programació, el que es fa és utilitzar un programa ja existent que aporti el màxim de facilitats a l'hora de desenvolupar les modificacions que se li faran.

3.1. Requeriments del software

3.1.1. Programari lliure

El primer que es demana al programa és que es pugui tenir accés al codi font i que aquest es pugui modificar. Aquesta és precisament una de les condicions que defineixen el programari lliure i la principal causa per escollir aquest tipus de software en front del programari propietari.

El terme de programari lliure (en anglès *free software*) no s'ha de confondre amb el de programari gratuït. En anglès el terme *free* vol dir tant lliure com gratuït. En aquest context es pren l'accepció de lliure. Així, que el programari sigui lliure no implica que sigui gratuït.

El concepte de programari lliure fa referència al que es pot fer amb ell i no a quan es paga per ell. El que es pot fer amb el programari lliure sol agrupar-se en quatre llibertats:

- *Llibertat 0*. El programari es pot fer servir per qualsevol propòsit.
- *Llibertat 1*. Poder estudiar com funciona el programa i adaptar-ho a les teves necessitats.
- *Llibertat 2*. Poder realitzar i distribuir còpies del programari.
- *Llibertat 3*. Millorar i publicar les millores fetes al programari.

Les llibertats 1 i 3 impliquen accés al codi font. Això és una condició necessària, però no suficient. És necessari poder adaptar-lo a les necessitats i, a més, poder distribuir les còpies,

de forma gratuïta o no. En general, quan un programari compleix les quatre llibertats rep el nom programari lliure.

3.1.2. Gratuïtat

Que el programari sigui lliure ja s'ha vist que no obliga a que sigui gratuït. No obstant això, i estudiades la majoria d'aplicacions gratuïtes i de pagament que hi ha a disposició, no es troba cap motiu pel qual s'hagi de triar un programa de pagament. Així doncs, també es demana que el programa sigui gratuït, tan en l'obtenció com en la distribució.

3.1.3. Components del programa

Com s'ha esmentat anteriorment, es vol que el programa suporti el modelat i l'anàlisi de les xarxes de Petri, i que el seu ús sigui senzill i intuïtiu, no escatimant però, en el número de prestacions addicionals possibles per a poder desenvolupar una eina de màxima utilitat. Cal doncs saber quins components ha de tenir el programa. Una possible llista seria:

- *Editor de gràfics:* que permeti representar visualment les xarxes de Petri.
- *Simulació animada del joc de marques:* que permeti la simulació gràfica de l'evolució de les marques per la xarxa.
- *Simulació ràpida:* que faci la simulació sense gràfics per aconseguir informació a llarg plaç i en condicions límit.
- *Espai d'estats:* que suporti la generació d'espai d'estats (o els grafs d'ocurrència o d'abastament).
- *Invariants:* que permeti trobar els invariants, tan de llocs com de transicions (P-invariants i T-invariants).
- *Classificació:* que distingeixi el tipus de xarxa de Petri.
- *Anàlisi de les simulacions:* que faci estudis estadístics de les simulacions.

N'hi poden haver molts d'altres, però aquests components poden ser la base dels requeriments que s'utilitzin per escollir el programa.

No s'ha esmentat res sobre el tipus de xarxes suportades. Per a portar a terme la generació de codi no es necessita que les xarxes siguin d'alt nivell (xarxes de Petri colorejades) ni tan

sols que suportin la variable temps (xarxes temporitzades o estocàstiques), ja que el que es farà és dotar els elements de les xarxa de les propietats necessàries per a què suportin el codi. Això vol dir que partint de la representació d'una xarxa autònoma el que es farà és donar-li una interpretació posteriorment, quan es vulgui generar el codi. Així doncs, només cal que el programa suporti xarxes de Petri ordinàries o generalitzades.

3.2. Elecció del programa

La quantitat de programes a disposició de l'usuari per a l'edició i anàlisi de les xarxes de Petri no és petita: fent una cerca per internet es troben una gran quantitat de programes que serveixen per aquest propòsit. Com ja s'ha comentat anteriorment, es busca un programa del qual es tingui accés al codi font per a poder-lo editar i que a més sigui gratuït. Per fer un anàlisi dels programes que compleixen aquestes característiques es pren com a lloc de partida la base de dades de Petri Nets World [2] on es recullen les principals eines relacionades amb les xarxes de Petri. Aprofitant les eines de cerca d'aquesta base de dades, es busquen tots els programes que siguin gratuïts i que tinguin tots els components esmentats anteriorment. De les més de 60 aplicacions (presentades en l'annex A), aquesta cerca dóna un únic resultat: el **Platform Independent Petri Net Editor**, conegut com a **PIPE**, que compleix en escreix tots els requeriments imposats.

Un cop feta aquesta primera aproximació es proven tan aquest com d'altres programes que, tot i no complir amb les condicions posades, podrien ser igual d'interessants. Es prova el programa **Tina** però la interfície gràfica no es intuïtiva i per fer diferents accions s'han de córrer varis programes alhora. També es prova el **Visual Object Net++** i tot i resultar ser bastant millor que l'anterior, les carències gràfiques i sobretot d'anàlisi fan desestimar-lo. Finalment també es prova el **Petri Net Kernel** i els resultats són similars: poques propietats d'anàlisi i un editor de xarxes mitjançant menús poc visuals. Les conclusions a les que s'arriben un cop provats no difereixen de la primera aproximació: el PIPE és el programa que més funcions té, més intuïtiu resulta d'utilitzar i més facilitats posa a l'hora de modificar-lo i ampliar-lo. S'escull doncs, el PIPE.

3.3. Platform Independent Petri Net Editor (PIPE)

3.3.1. Introducció

PIPE és un projecte començat al desembre de l'any 2002 al Department of Computing, Imperial College, de Londres, portat a terme per un grup de 5 alumnes de postgrau capitanejats per James D. Bloom, que en l'actualitat s'encarrega del manteniment del programa.

Un dels motius principals per desenvolupar el nou programa va ser que tots els editors de xarxes de Petri disponibles eren difícils d'utilitzar i poc intuïtius. Estava clar que aconseguir una aplicació fàcil d'usar era la clau a l'hora de considerar el disseny, de manera que el programa fós una forma simple, ràpida i eficient de crear xarxes de Petri (l'annex E inclou l'informe del projecte PIPE).

De totes maneres, les funcionalitats del programes existents solen estar limitades a les que el programador creu convenient incloure en el moment d'escriure'l. El principal punt d'atenció al desenvolupar el projecte PIPE va ser crear una eina que suportés la càrrega i descàrrega de mòduls d'anàlisi en temps d'execució. Usant una interfície de mòdul, els usuaris tenen la capacitat d'escriure els seus propis mòduls d'anàlisi implementant simplement aquesta interfície. Això es tradueix en que l'usuari és capaç de dur a terme tot tipus d'anàlisi utilitzant una sola aplicació. A més, el programa utilitza el nou format estàndard de xarxes de Petri basat en *XML*, el *PNML* (explicat en l'apartat 6.3 i ampliat en l'annex E), fent que el programa sigui capaç d'analitzar xarxes de Petri creades amb altres aplicacions que utilitzin el mateix estàndard.

3.3.2. Components i característiques de PIPE

Interfície gràfica d'usuari:

Editor. Aquest permet a l'usuari crear, guardar i obrir xarxes de Petri gràfiques que utilitzen el Petri Net Markup Language (*PNML*). Des de l'editor l'usuari és capaç de crear i editar xarxes de Petri de manera gràfica mitjançant els menús, les barres d'eines i el ratolí.

Animació. L'usuari pot disparar manualment les transicions habilitades. Aquestes es remarquen quan s'utilitza el mode d'animació per tal que l'usuari distingeixi les transicions que es poden disparar i les que no. La xarxa pot tornar fàcilment a l'estat inicial, ja sigui al final de l'animació o en el moment que l'usuari ho demana, així com desfer una a una la seqüència de dispars.

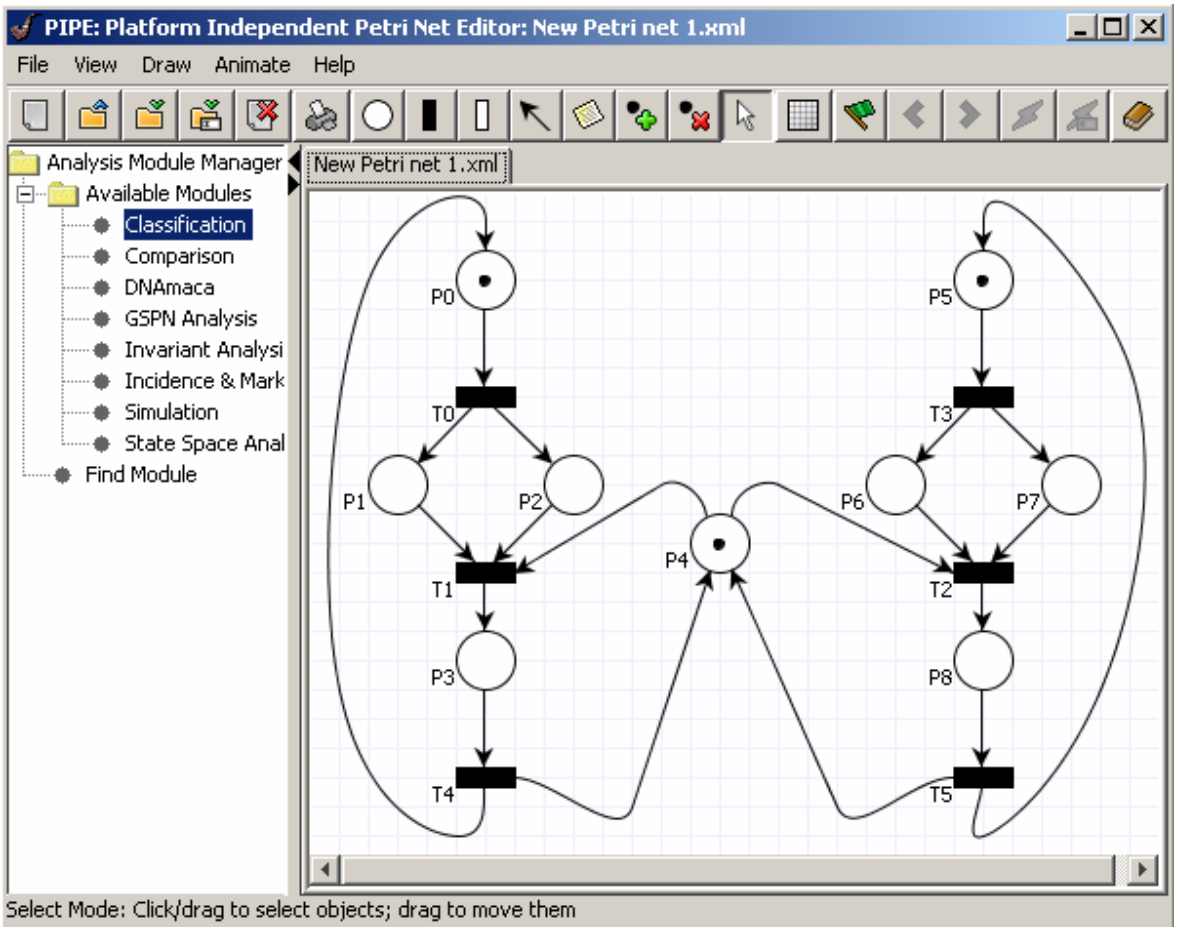


Figura 3.1 Interfície gràfica del programa PIPE.

Mòduls d'anàlisi carregables:

Com es veu a la *Figura 3.1*, a la part esquerra de la pantalla es troba el gestor de mòduls representat amb un arbre (*Tree View*). Com es pot observar, l'última opció de l'arbre serveix per carregar un nou mòdul en temps d'execució. Seguidament es comenten les funcionalitats bàsiques d'aquests mòduls.

Anàlisi d'invariants. Aquest mòdul (*Figura 3.2*) analitza la xarxa determinant els vectors dels P-Invariants i els T-Invariants, donant a més informació sobre l'acotació i vivacitat de la xarxa.

Mòdul de simulació. Aquest mòdul (*Figura 3.3*) treballa sense que l'usuari intervingui en la simulació i calcula de forma ràpida paràmetres estadístics de cada lloc per al seu anàlisi com el número promig de marques que té cada lloc i l'interval de confiança d'aquest valor.

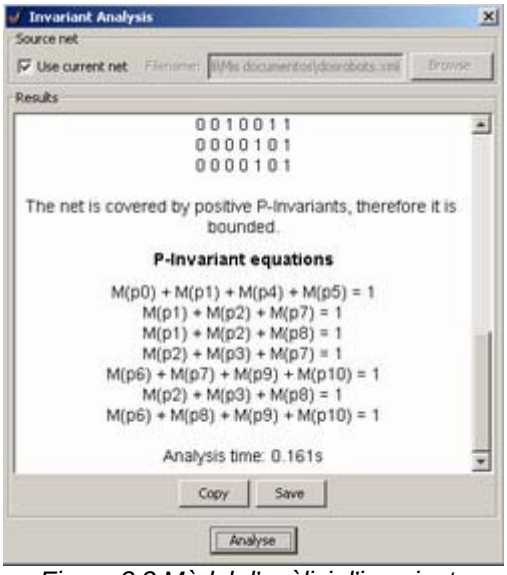


Figura 3.2 Mòdul d'anàlisi d'invariants

Incidència i marcat. Calcula les matrius de Pre-Incidència, de Post-Incidència i la combinada de les dues (*Figura 3.4*). A més crea una taula on diu si els llocs estaven marcats inicialment o no i si ho estan actualment. També crea una taula amb l'habilitació de cada transició en l'estat actual.

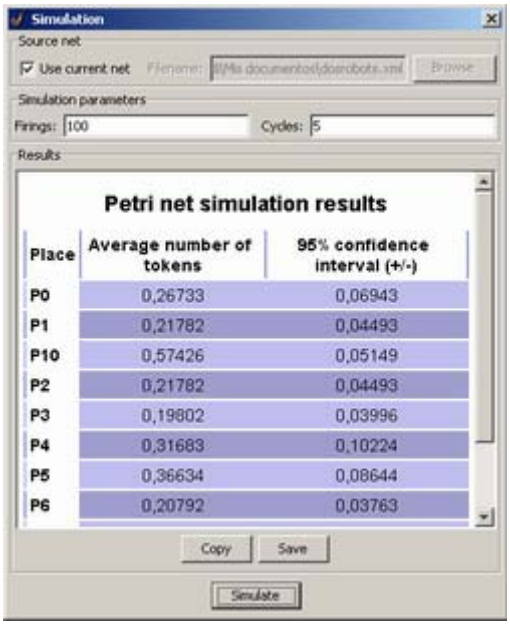


Figura 3.3 Mòdul de simulació

Comparació. Aquest mòdul serveix per veure les diferències entre dos xarxes de Petri a partir dels seus fitxers .

Classificació. Donada una xarxa de Petri la classifica en un dels següents tipus: Màquina d'estats, Graf marcat, xarxa d'elecció lliure o xarxes simples.

Anàlisi de l'espai d'estats. Aquest mòdul investiga les propietats de la xarxa, com la vivacitat, l'acotació i els bloquejos en cas que els tingui.

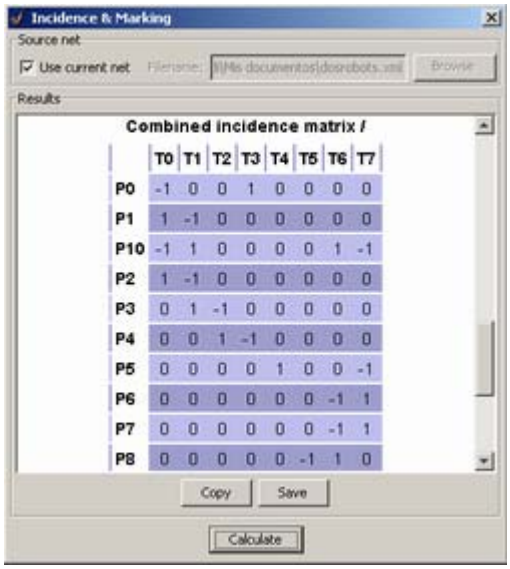


Figura 3.4 Mòdul d'incidència i marcat

El funcionament general del programa s'inclou en el manual d'usuari de l'annex D.

3.4. Especificacions per a les noves funcionalitats

Aquesta secció detalla les especificacions que es plantegen al començament del projecte. Aquestes especificacions indiquen les funcionalitats bàsiques de les que s'ha de dotar el programa PIPE per acomplir els objectius. Com ja s'ha mencionat, aquest programa té la capacitat de carregar un nou mòdul programat independentment, facilitant així les tasques de programació. Això és una gran ajuda però degut a que es necessita dotar d'interpretació els elements de la xarxa no serà suficient per a dur a terme el projecte. Així doncs, primerament es plantegen les especificacions que estan estrictament relacionades amb el nou mòdul i que no requereixen modificacions del codi font del programa, i tot seguit s'expliquen les modificacions necessàries vinculades al codi intern de PIPE per tal d'acomplir amb les especificacions del mòdul.

3.4.1. Especificacions generals del mòdul de programació

Desenvolupar un nou mòdul amb tots els elements necessaris per a la generació automàtica de codi. Aquest mòdul processarà l'estructura del sistema d'estudi representat per la xarxa de Petri corresponent i diferenciarà els seus elements a partir de la interpretació que se'n faci de cada un d'ells. En principi està pensat per a suportar xarxes que modelin un sistema d'un o de dos robots treballant en paral·lel. Obrirà una finestra on posarà la informació generada i tindrà la capacitat de estructurar la pantalla en funció del número de robots que hi intervenen. A més, des d'aquesta finestra també es tindrà accés al codi introduït per l'usuari i es podrà modificar si s'escau. El codi es generarà en llenguatge de programació Val3 que és el llenguatge que utilitzen els robots de la marca Stäubli, dels quals se'n disposen dues unitats a l'Institut d'Organització i Control situat a la planta 11 de l'ETSEIB. Es podrà guardar en fitxers de text independents, copiar-lo en la memòria interna si s'escau o generar els projectes necessaris per obrir-lo posteriorment amb l'eina de programació adient.

3.4.2. Modificacions internes de PIPE

- Dotar els elements de la xarxa (llocs i transicions) de les variables necessàries per donar-los la interpretació correcta, és a dir, diferenciar cada element en funció del robot que executarà aquella tasca. Així s'haurà de definir el tipus dels elements.

- Diferenciar els elements que no pertanyen a cap robot en concret sinó que seran compartits per varis robots.

- Modificar les definicions internes dels llocs i/o transicions perquè tinguin la capacitat d'emmagatzemar cada un el seu codi particular.

- Distingir gràficament de manera fàcil els diferents elements del sistema.

- Ampliar si s'escau la definició interna dels elements de la xarxa per tal de poder guardar i obrir les xarxes. Fer que aquestes modificacions compleixin amb l'estàndard *PNML* per tal que els fitxers es puguin utilitzar en altres aplicacions.

- Aconseguir que cap d'aquestes modificacions interfereixi en el bon funcionament de la resta de funcionalitats del programa i que no condicionin el desenvolupament de futures ampliacions.

4. Els robots

4.1. Els robots i altres mecanismes

Un robot és una eina complexa en la que es barregen diverses tecnologies. Hi ha una part mecànica que li dóna entitat i que el permet moure's o moure altres objectes. També hi ha una part elèctrica (i electrònica) que permet controlar la part mecànica.

Khalil i Dombre [3] proposen la següent definició: Un robot és un sistema mecànic amb diversos graus de llibertat, controlat automàticament, reprogramable, multipropòsit, que pot estar fix o pot ser mòbil. Segons aquests autors hi ha tres tipus genèrics de robots: robots manipuladors, que imiten el braç humà, robots caminadors, que imiten el sistema locomotriu dels humans o dels animals, i els robots mòbils que semblen vehicles.

4.1.1. Els robots industrials

A nivell industrial hi ha molts tipus de robots. El més comú és el manipulador mecànic o robot manipulador. J. Craig [4] proposa que per diferenciar un robot industrial del que no ho és, cal fixar-se en la sofisticació de la programabilitat del dispositiu: aquell que només es pot programar per fer un tipus de tasca, és considerat una automatització simple i no un robot industrial.

4.1.2. Els components d'un sistema robotitzat

Un sistema robotitzat és aquell que està constituït per un o més robots o perifèrics en forma d'estructures mecàniques articulades en un espai delimitat i que treballa en conjunt. Als sistemes robotitzats es coneixen com a cel·les de treball (*workcells*).

Una estructura articulada és un mecanisme sobre el qual actua un actuador elèctric, pneumàtic o hidràulic, que transmet un moviment a les articulacions utilitzant un sistema de transmissió.

Un robot pot tenir sensors de contacte, mesuradors de distància o visió artificial que li aporten capacitat de percepció, fet que l'ajuda a adaptar-se al sistema on es troba. Els sensors interns proveeixen d'informació sobre l'estat del mateix, com per exemple les posicions de les articulacions i les seves velocitats.

El controlador és aquella part que genera els senyals d'entrada per als actuadors en funció de les instruccions de l'usuari i recull els senyals de sortida dels sensors de forma que es puguin assolir els objectius a realitzar.

4.2. La programació dels robots

La sofisticació de la programabilitat del dispositiu fa que els robots industrials puguin fer un gran ventall de feines, però per poder-ho fer s'han de programar. Hi ha múltiples llenguatges de programació per robots; cada fabricant de robots sol subministrar-ne un, i per tant no hi ha un estàndard comú.

Malgrat aquest fet, en general tots tenen un enfocament comú. La programació de robots fa servir un llenguatge explícit. Això vol dir que el programa consta d'una seqüència d'ordres concretes que van definint amb rigor les operacions necessàries per dur a terme l'aplicació. Aquesta explicitat en el llenguatge implica que la persona que ha de programar el robot ha de definir els moviments exactes que el robot ha d'efectuar. La metodologia de programació consisteix en definir les posicions on ha d'arribar el robot, programar el dispositiu que tingui incorporat a la punta (una pinça per agafar, un làser per soldar, etc.) i explicitar com s'ha de fer el moviment entre posició i posició.

Aquesta forma de programació fa que, malgrat la seva simplicitat, sigui una feina laboriosa. Per aquest motiu, els llenguatges de programació de robots solen incorporar funcions que ajuden a la programació. també sol incorporar-se al robot un comandament per manejar el robot. Aquests comandaments solen ser una botonera on l'operari pot moure el robot efectuant petits desplaçaments en els motors que fan girar les articulacions.

En general per programar un robot s'ha de tenir accés a aquest. D'aquesta manera fent moviments en el robot, es poden guardar els valors de les posicions i així fer un programa vagi anant de posició en posició.

4.2.1. Les xarxes de Petri en la programació de robots

La programació de sistemes i en especial els sistemes multirobot és una feina complexa. Els sistemes robotitzats de muntatge d'elements requereixen un profund estudi per tal de sincronitzar les tasques, o determinar per exemple la seqüència d'operacions òptima per desenvolupar un producte. Arnold [5] presenta les bases per desenvolupar una interfície gràfica basada en grafcet per la programació de robots industrials. Tal com s'ha vist en l'apartat 2.3 les xarxes de Petri són un formalisme adequat per modelar sistemes, com per

exemple els sistemes multirobot. Rosell [6] aborda la planificació de tasques i el control de robots a través de xarxes de Petri. Suh [7] aborda aquest tema basant-se en el Petri net Graphical Robot Language (PGRL).

El desenvolupament d'aquest projecte intenta aportar una solució a aquesta problemàtica, dotant dels instruments necessaris per simplificar al màxim possible la feina feixuga del control i la sincronització dels robots.

4.3. Els robots Stäubli

En el mercat existeixen moltes marques de robots articulats i cada fabricant té el seu llenguatge de programació. Aquest projecte parteix d'una idea general, i d'aplicació a qualsevol robot però degut a aquesta problemàtica en la programació i a la complexitat de desenvolupar una eina vàlida per a qualsevol marca de robot s'escull centrar-se en robots de la marca Stäubli, deixant oberta la possibilitat d'ampliació a altres robots en altres projectes. El perquè d'aquests robots i no uns altres ja s'ha comentat: es disposa de dues unitats de robots Stäubli a l'Institut d'Organització i Control de Sistemes Industrials, situat a la planta 11 de l'ETSEIB.

El 1982 Stäubli Faverges va portar a terme l'opció estratègica de la robòtica. Aquesta diversificació en el producte va passar en un principi per la compra de la societat americana Unimation, pionera en la robòtica industrial. Deu anys després Stäubli va plasmar les seves noves competències en el sector amb el llançament del braç RX totalment desenvolupat i fabricat a les instal·lacions de Faverges.

Els robots Stäubli es caracteritzen per la seva fiabilitat, robustesa i el seu reduït tamany i conjuguen velocitats elevades amb alta precisió, destresa i fàcil posada en marxa. Actualment hi ha dues gammes de braços robot, la RX i la nova TX (*Figura 4.1*). Els robots necessiten d'un controlador, que en el cas de robots Stäubli es coneix com a CS8, del qual també se'n disposa un a l'Institut d'Organització i Control de Sistemes Industrials.

Figura 4.1 Braç robot TX-90 de la marca Stäubli

4.3.1. El llenguatge de programació Val3

Stäubli ha canviat recentment el llenguatge de programació dels seus robots, substituint l'antic V+ pel nou llenguatge **Val3**. Es tracta d'un llenguatge estructurat i interpretat, perfectament adaptat a la robòtica que permet gestionar el conjunt de l'aplicació. Ha conservat els aspectes fonamentals de les possibilitats d'un llenguatge informàtic en temps real estàndard, integrant-hi les funcionalitats específiques per al maneig d'un robot en una cel·la industrial: eines de control del robot, eines de modelització geomètrica i eines de control d'entrades i sortides.

La programació de les aplicacions en Val3 es fa a través d'un entorn robotitzat per a PC, el *Stäubli Robotics Studio*, format per un conjunt d'eines en un entorn Windows que a més de la programació també facilita al màxim la posada en marxa. Dels varis elements que formen aquest entorn, en aquest projecte se n'utilitzen dos: el *Val3 Studio* per a la creació i la modificació d'aplicacions i el *CS8 Emulator* que ens serveix per a la simulació completa del controlador i del comandament manual.

Una aplicació Val3 és un software autònom de programació del robot amb les entrades i sortides vinculades a un controlador CS8. Aquesta aplicació està constituïda per:

- Un conjunt de programes: les instruccions Val3 que han d'executar-se.
- Un conjunt de variables globals: les dades de l'aplicació.

- Un conjunt de biblioteques: les instruccions i dades externes utilitzats per l'aplicació.
- Un conjunt de tasques: els programes en curs d'execució.

Totes les aplicacions han de tenir els programes **Start**() i **Stop**() per l'engegada i parada, així com un sistema de referència *World* (tipus *frame*) i una eina *flange* (tipus *tool*). Els programes estan constituïts per instruccions, variables locals i paràmetres, com a qualsevol llenguatge de programació i permeten agrupar seqüències d'instruccions susceptibles de ser utilitzades en varis llocs a l'aplicació. Tots els programes son reentrants, el que vol dir que un programa pot cridar-se a sí mateix de manera recursiva (instrucció **call**) o ser cridat simultàniament per varies tasques. Tots els programes han de començar amb l'instrucció begin i acabar amb la instrucció **end**. Els tipus de dades de Val3 s'agrupen en tipus senzills i tipus estructurats. Els tipus senzills són:

- Tipus *bool*: per a valors booleans
- Tipus *num*: per a valors numèrics
- Tipus *string*: per a cadenes de caràcters.
- Tipus *dio*: per a les entrades/sortides tot o res.
- Tipus *aio*: per a les entrades/sortides numèriques (analògiques o digitals).
- Tipus *sio*: per les entrades/sortides en connexió sèrie.

Un tipus estructurat és un tipus que reuneix varies dades caracteritzades, els camps del tipus estructurat, que són accessibles individualment pel seu nom. Els tipus estructurats que suporta són:

- Tipus *trsf* : per a les transformacions geomètriques cartesianes.
- Tipus *frame* : per als sistemes de referència geomètrics cartesians.
- Tipus *tool* : per a les eines ajustades en un robot.
- Tipus *point* : per a les posicions cartesianes d'una eina.
- Tipus *joint* : per a les posicions articulars del robot.
- Tipus *config* : per a les configuracions del robot.
- Tipus *mdesc* : per als paràmetres de desplaçament del robot.

De tots els aspectes generals de Val3 es fa menció seguidament dels elements més importants que serviran per a la generació automàtica de codi. Aquests elements són bàsicament els mètodes de comunicació entre els robots i la gestió de les tasques que corren en paral·lel.

La comunicació entre els robots es pot implementar de moltes maneres però d'entre d'elles la utilització del tipus *SIO* és la que es considera més adient. El tipus *sio* permet vincular una variable Val3 a l'entrada-sortida sèrie del sistema o una connexió a través de socket Ethernet. Una entrada *sio* es caracteritza per uns paràmetres propis al tipus de comunicació, definits en el sistema, un caràcter de fi de cadena per fer possible la utilització del tipus *string* i un temps d'espera de comunicació. Les connexions mitjançant socket Ethernet s'activen en el primer accés en lectura o escriptura a través d'un programa Val3 i es desactiven automàticament quan s'acaba l'aplicació. Les entrades i sortides declarades en el sistema són directament utilitzables en una aplicació Val3, sense haver de declarar-les com a variable global o local a l'aplicació. Qualsevol instrucció que utilitzi una variable de tipus *sio* no vinculada a una entrada-sortida declarada al sistema genera un error d'execució. Per tan, en el cas de sistemes amb dos robots, la generació automàtica de codi serà capaç de crear les instruccions necessàries que utilitzin les variables tipus *sio* per a fer possible aquesta comunicació.

En la programació de robots és molt important la gestió de les tasques que s'executen, ja que freqüentment es necessiten varis programes corrent en paral·lel. En Val3, quan una aplicació te varies tasques en curs d'execució, aquestes semblen executar-se simultàniament. Això només és cert si s'observa l'aplicació globalment en intervals de temps suficientment llargs, però no és exacte quan un s'interessa en el comportament precís d'un interval de temps reduït ja que el sistema no té sinó un sol processador. La simultaneïtat se simula a través d'una seqüenciació ràpida de les tasques, que executen unes després d'altres unes instruccions, abans que el sistema passi a la tasca següent.

A vegades es fa necessari sincronitzar varies tasques abans que aquestes segueixin la seva execució. Si es coneix a priori la durada de cada una d'aquestes tasques, aquesta sincronització es pot fer senzillament per l'espera d'un senyal emès per la tasca més lenta. Però quan no se sap quina de les tasques serà la més lenta, s'ha d'utilitzar un mecanisme de sincronització més complex.

Quan varies tasques utilitzen un mateix recurs del sistema o de la cel·la, és indispensable assegurar-se que no van a pertorbar-se mútuament. Per això es pot utilitzar un mecanisme d'exclusió mútua que protegeix un recurs autoritzant el seu accés a una sola tasca a la vegada.Val3 és capaç de suportar totes aquestes característiques orientades a la gestió de tasques i serà un dels punts importants en la generació automàtica de codi.

Per informació detallada de Val3 veure el manual de referència de l'annex E.

4.3.2. Creació de projectes: Val3 Studio

Val3 Studio és el programa utilitzat per crear i modificar aplicacions en el llenguatge Val3 (*Figura 4.2*). Aquestes aplicacions s'estructuren en programes, agrupant un conjunt d'instruccions a executar conjuntament.

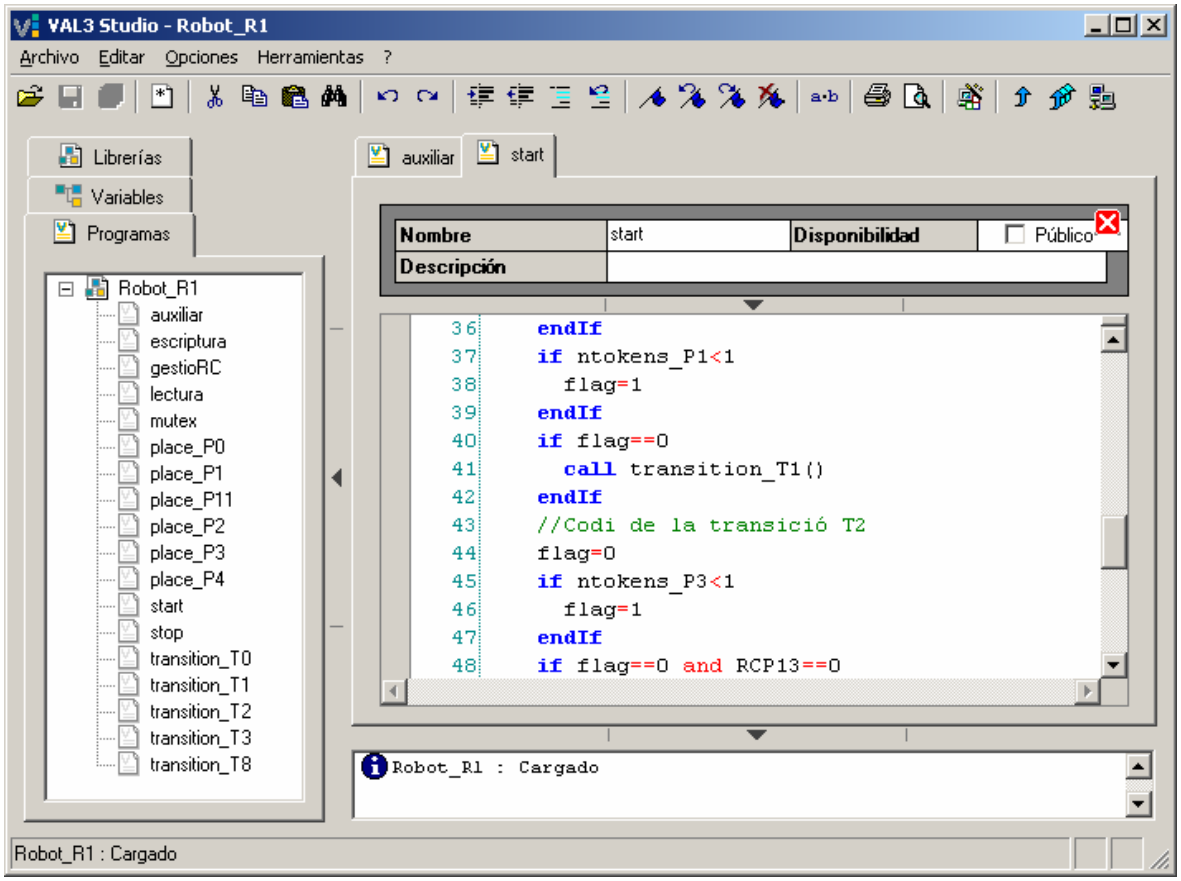


Figura 4.2 Programa Val3 Studio per a la creació i edició d'aplicacions Val3

Quan es crea una aplicació en Val3 Studio aquesta s'empaqueta en forma de projecte i es guarda en un directori amb el nom d'aquest projecte. Dins el directori es guarden tots els fitxers generats que formen part de l'aplicació en quatre formats diferents: *pgx*, *dtx*, *ltx* i *pjx*. En realitat tots quatre són fitxers *XML*, i utilitza aquests formats per diferenciar els quatre tipus de fitxers que genera:

- *Fitxers pgx*: tots els programes que formen part de l'aplicació es guarden en aquest format. Les etiquetes *XML* del que es forma contenen el nom del programa, una descripció, els paràmetres que utilitza, les variables locals i el codi del programa.

- *Fitxers dtx*: cada projecte té un fitxer *dtx* amb el nom del projecte. Aquest fitxer conté la definició de totes les variables globals que utilitza l'aplicació en tots els programes. Per tan, les etiquetes *XML* que el formen son tots els tipus de variables possibles en Val3.

- *Fitxers ltx*: un projecte pot tenir o no aquest fitxer, i de tenir-lo, només en tindrà un que serveix per informar de les variables de les biblioteques externes que utilitza l'aplicació. Només conté una etiqueta *XML* que conté aquestes variables.

- *Fitxers pjx*: igual que el *dtx*, cada projecte té un fitxer *pjx* amb el nom del projecte. Aquest fitxer és el que empaqueta tots els programes de l'aplicació. En obrir un projecte en *Val3 Studio* aquest crida el fitxer *pjx*, que a dins conté la crida a tots el programes que utilitza l'aplicació. Les etiquetes *XML* principals de que consta son tres: la que conté les crides als programes *pgx*, la que conté les crides al fitxer de dades *dtx* i una etiqueta que crida les biblioteques externes utilitzades per l'aplicació.

La creació d'aquests fitxers serà necessària ja que es vol que el mòdul de programació generi un codi editable des de *Val3 Studio*.

5. El mòdul de programació

En l'apartat 3 s'ha justificat l'elecció del programa PIPE i en l'apartat 4 s'ha vist el llenguatge de programació Val3 dels robots, que serà la manera de mostrar els resultats del mòdul. Seguidament s'expliquen les característiques del mòdul de programació creat i es mostra un exemple com a validació experimental del treball. La seva implementació s'explica en el capítol 7.

5.1. Descripció i funcionament del mòdul

Aquest mòdul de programació té com a finalitat generar de forma automàtica una estructura de codi que permeti controlar els robots del sistema. La creació de la xarxa que modela el sistema i la introducció de les accions de moviment està a l'abast de qualsevol usuari (que pot ser un operari), mentre que la programació del control dels robots és molt més complexa. D'aquesta manera, el mòdul de programació minimitza el procés complex i feixuc de programar el comportament dels robots, deixant per a l'usuari només la introducció de forma senzilla del codi del moviment dels robots a través de la xarxa de Petri.

El mòdul s'anomena **Code Generation** i apareix a l'arbre de selecció dels mòduls amb aquest nom. Quan es prem sobre el nom a l'arbre s'obra la finestra del mòdul (*Figura 5.1*) on es poden distingir 4 parts. A la part superior hi ha un quadre amb el nom de *Source Net*, que conté un selector de fitxers. Per defecte surt marcada l'opció d'utilitzar la xarxa de Petri actual que es mostra en el programa, però des d'aquest selector de fitxers es pot escollir qualsevol fitxer guardat en format *PNML*. La part central i principal es divideix en dues parts iguals: són les finestres a on es mostrarà el codi per a cada robot. Cada finestra té dos botons, un per copiar el text mostrat a la memòria i un per guardar el text com a fitxer de text. Finalment a la part de sota hi ha tres botons més: un botó que genera el codi, un botó que serveix per revisar el codi dels llocs i modificar-lo, i un últim botó que guarda tota la informació com a un projecte, en el format entenedor del *Val3 Estudio* (apartat 4.3.2),

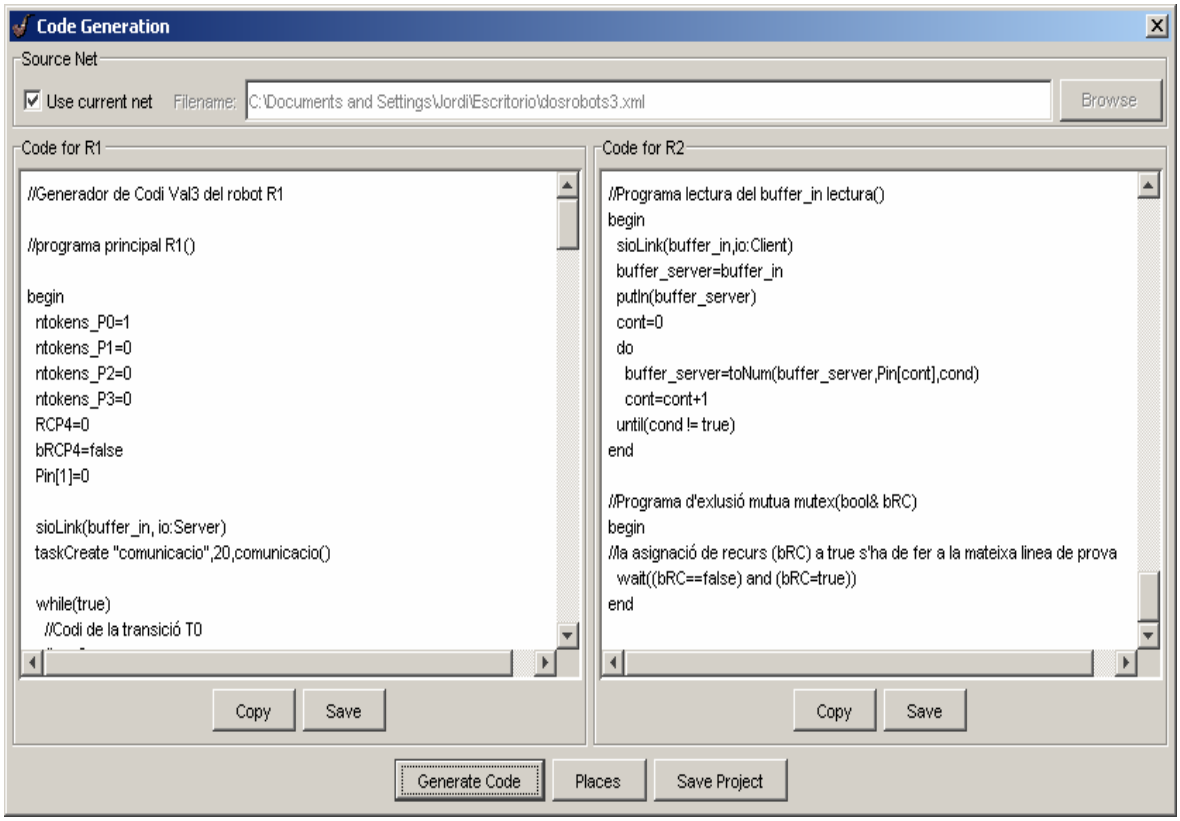


Figura 5.1 Finestra del mòdul de programació on es mostra el codi generat.

El funcionament és molt simple: es crea una xarxa de Petri des de l'editor, o s'obra una xarxa guardada en format *PNML*, s'assigna el tipus a cada lloc de la xarxa i es guarda el corresponent codi a cada lloc, ja sigui des de l'editor o des de dins del mòdul a través del menú creat per a aquesta finalitat. Seguidament es prem el botó *Generate Code* de dins el mòdul i es visualitza el codi en la primera finestra si només hi ha un robot o en les dues si la xarxa representa el funcionament de dos robots. Després, el codi es pot guardar com a fitxers de text o com a projecte de Val3. Es recomana fer servir aquesta opció ja que així es facilita la modificació, la declaració de variables que poden ser necessàries (incloses per l'usuari en el codi dels llocs) i la transferència de la informació als robots.

5.2. Característiques del codi generat

El codi que ha de generar el mòdul depèn de la complexitat de la xarxa, ja que serà diferent si hi ha un sol robot o si n'hi ha dos interactuant entre ells. Si hi ha dos robots, la diferència principal és l'existència o no de recursos compartits a la xarxa, ja que en cas d'existir, es necessita un sistema de comunicació continu entre els robots per tal de gestionar correctament l'ús dels recursos.

Tan si hi ha un com dos robots, el propi modelatge del sistema en xarxa de Petri facilita la inclusió de codi auxiliar que s'executi en paral·lel al fil principal de cada robot. Aquest codi permetrà realitzar càlculs necessaris per l'aplicació però que no formen part de les accions dels robots, així com la possible activació de diferents dispositius d'entrada/sortida.

5.2.1. Cas de dos robots interactuant entre ells

Es comença explicant aquest cas perquè és el més complex però a la vegada el de major utilitat, i també perquè el següent cas serà una simplificació d'aquest.

En el cas de tenir un sistema amb dos robots (que a partir d'ara s'anomenen R1 i R2) el que es farà és intentar generar una estructura de codi que sigui simètrica pels dos, encara que el robot R1 serà el que gestionarà els recursos compartits. Així el robot R1 tindrà el paper de servidor en la comunicació mitjançant socket Ethernet, mentre que el R2 serà un client. L'estructura general de programes que tindran les aplicacions serà similar, si no la mateixa. D'acord amb l'apartat 4.3.1 totes les aplicacions Val3 contenen un programa **start**() que serveix per iniciar l'aplicació. El programa start() que es generarà serà l'encarregat de fer el cicle general del robot, repetint-lo indefinidament fins que no s'executi el programa **stop**(). Per a poder implementar la dinàmica d'aquest llaç es defineixen un conjunt de variables on moltes tenen un comportament similar:

1. Un conjunt de variables tipus *num* anomenades ntokens_+"nom del lloc". Hi haurà una variable per a cada lloc, excepte pels llocs de tipus recurs compartit que es gestionen a part. Aquestes variables es definiran inicialment amb el marcat inicial de cada lloc, i s'aniran actualitzant dins el llaç amb el valor del marcat actual.

2. Una variable tipus *num* anomenada flag que servirà per gestionar el cicle principal del robot.

3. Un conjunt de variables tipus *num* anomenades RC+"nom del lloc". Hi haurà una variable per cada lloc de tipus recurs compartit i podrà tenir tres valors, 0, 1 o 2. Que una variable RC valgui 0 voldrà dir que el recurs compartit està lliure i no l'està fent servir cap robot, que valgui 1 voldrà dir que l'està fent servir el robot R1 i que valgui 2 que l'està fent servir el robot R2. Aquests valors els assignarà el robot R1 que fa de servidor en al comunicació, i per tan, les variables RC del robot R2 contindran una copia del valor de les variables amb el mateix nom del robot R1 que aquest li haurà enviat.

4. Un conjunt de variables tipus *bool* anomenades bRC+"nom del lloc". Hi haurà una variable per cada lloc de tipus recurs compartit i servirà per implementar l'exclusió

mútua de les variables RC, ja que aquestes variables seran accedides contínuament per diferents tasques.

5. Dues variables tipus *sio* anomenades buffer_in i buffer_out que serviran per a establir la connexió entre els dos robots.

6. Dues variables tipus *string* anomenades buffer_server i buffer_client que s'usaran per obtenir els valors de les variables *sio* buffer_in i buffer_out respectivament i així poder-los modificar.

7. A més de les anteriors, per a fer possible la comunicació del robots també seran necessàries les següents variables: una variable tipus *bool* anomenada cond, un vector Pin[] de tipus *num* (també Pout[] en el robot R2), una variable tipus *num* anomenada cont, una variable tipus *string* anomenada SRC0 i un conjunt de variables tipus *string* anomenades SRC+"nom del lloc", una per a cada lloc compartit.

El codi generat estarà organitzat en mètodes separats. Cada mètode generat es mostrarà seqüencialment a la finestra de text del robot que l'utilitzarà. En Val3 però, es guarda cada mètode en fitxers independents en format *XML* de manera que cada mètode forma un programa. Aquests mètodes/programes que es mostraran en la finestra de codi i que també haurà de generar el mòdul (amb l'extensió *pgx*) seran els següents:

1. El programa **start()** i **stop()** ja mencionats. El programa start() tindrà un patró igual per a qualsevol model.

```
begin
        // A - INICIALITZACIÓ DE LES VARIABLES
        // B - ESTABLIMENT DE LA COMUNICACIÓ
        while (TRUE)
                //C - GESTIÓ DE LES TRANSICIONS
                        //C.1 - GESTIÓ TRANSICIONS AMB RECURS COMPARTIT A L'ENTRADA
                        //C.2 -  GESTIÓ DE TRANSICIONS PRINCIPALS (NO AUXILIARS)
                        // C.3 - GESTIÓ DE TRANSICIONS AMB RECURS COMPARTIT A LA SORTIDA
        endWhile
end
```

La dinàmica dels robots s'implementarà seguint el següent patró de gestió de les transicions corresponent al punt C.2 de l'esquema anterior:

```
//Tros de codi corresponent a una transició del punt C.2
  flag=0
```

```
if ntokens_P0<1
        flag=1
endIf
if flag==0
        call transition_T1()
endIf
```

Aquest és l'aspecte general que tindrà el tros de codi del llaç principal de cada robot per a cada transició. La variable flag es farà servir per saber si tots els llocs de l'entrada a la transició estan marcats o no i en cas d'estar marcats s'executaran les instruccions de la transició.

2. Un conjunt de programes anomenats **place_"nom del lloc"()**, un per a cada lloc, que simplement contindrà el codi que l'usuari haurà introduït al lloc corresponent.

```
//codi d'exemple corresponent a un programa de un lloc
begin
        movej(j[0],flange,nom_speed)
        movej(j[1],flange,nom_speed)
        waitEndMove()
end
```

3. Un conjunt de programes anomenats **transition_"nom de la transició"()**, un per a cada transició, que s'encarregaran de tres coses: primer restar una marca a la variable ntokens dels llocs d'entrada, després cridar els programes dels llocs de sortida i seguidament afegir una marca a les variables ntokens dels llocs de sortida.

```
//Programa transition_T0() del robot R1
begin
        ntokens_P0=ntokens_P0-1
        call place_P1()
        ntokens_P1=ntokens_P1+1
end
```

Les transicions que entrin a una branca auxiliar (*Figura 5.2*), és a dir, que no formen part del fil principal del robot, seran una particularitat ja que en aquest cas el codi dels llocs d'aquesta branca s'executarà en una altra tasca en paral·lel.

```
//Programa transition T0() del robot R1
begin
        ntokens_P0=ntokens_P0-1
        call place_P1()
        ntokens_P1=ntokens_P1+1
        taskCreate "auxR1",10,place_P2()
end
```

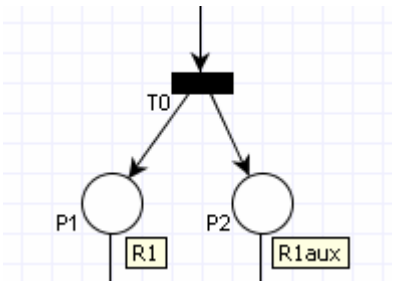


Figura 5.2 Exemple de branca auxiliar

4. Un programa anomenat **comunicacio()** que s'executarà en una tasca en paral·lel al cicle principal del robot i que permetrà la comunicació entre els robots cridant successivament els programes **lectura()**, **escriptura()** i **gestioRC()**. Aquest programa no dependrà del número de recursos compartits.

```
//Programa comunicacio() del robot R1
begin
    while(true)
        call lectura()
        call gestioRC()
        call escriptura()
    endWhile
end
```

```
//Programa comunicacio() del robot R2
begin
    while(true)
        call escriptura()
        call lectura()
        call gestioRC()
    endWhile
end
```

Degut a la diferent naturalesa dels programes que executa, aquest programa serà lleugerament diferent per a cada robot. El robot R1 primer llegirà, després gestionarà els recursos i després escriurà, mentre que el robot R2 primer escriurà, després llegirà i després gestionarà. Això és així perquè el robot R1 primer ha de llegir si R2 vol el recurs, el vol alliberar, o no el demana, i després ha d'assignar o alliberar el recurs.

5. Un programa anomenat **lectura()** que s'encarregarà d'actualitzar contínuament la variable buffer_server amb el valor de la variable *sio* buffer_in.

```
//Programa lectura() del robot R1
begin
    sioLink(buffer_in,io:Server)
    buffer_server=buffer_in
    cont=0
    do
        buffer_server=toNum(buffer_server,Pin
        [cont],cond)
        cont=cont+1
    until(cond != true)
end
```

```
//Programa lectura() del robot R2
begin
    sioLink(buffer_in,io:Client)
    buffer_server=buffer_in
    cont=0
    do
        buffer_server=toNum(buffer_server,Pin[c
        ont],cond)
        cont=cont+1
    until(cond != true)
end
```

Tal com es mostra, aquest codi generat no dependrà de la xarxa de Petri modelada.

6. Un programa anomenat **escriptura()** que s'encarregarà d'escriure contínuament a la variable *sio* buffer_out el valor de la variable buffer_client corresponent. A diferència de l'anterior, aquest programa sí que depèn del número de recursos compartits de la xarxa..

```
//Programa escriptura() del robot R1
begin
    SRC0=toString("1",9)
    call mutex(bRCP4)
    SRCP4=toString("1",RCP4)
    bRCP4=false
    sioLink(buffer_out,io:Server)
    buffer_client=SRC0+" "+SRCP4
    buffer_out=buffer_client
end
```

```
//Programa escriptura() del robot R2
begin
    SRC0=toString("1",9)
    call mutex(bPout)
    SRCP4=toString("1",Pout[1])
    bPout=false
    buffer_client=SRC0+" "+SRCP4
    sioLink(buffer_out,io:Client)
    buffer_out=buffer_client
end
```

En aquest exemple el lloc P4 és l'únic recurs compartit de la xarxa; es mostra com la seva variable SRC conté el valor de la variable RC en forma de *string*, seguidament es concatena el *string* buffer_client amb tots els valors de les variables SRC (en aquest exemple només n'hi ha un) i s'assigna aquest valor a la variable *sio* buffer_out. La variable SRC0 fa la funció d'inici de cadena i valdrà "9" per defecte. Com que les variables RC i Pout s'utilitzen en varies tasques, s'ha d'accedir al seu valor a través de l'exclusió mútua.

7. Un programa anomenat **gestioRC()** que tindrà diferent significat per a cada robot. Al robot R1 aquest programa mirarà si els recursos compartits estan lliures i a la vegada el robot R2 l'està demanant. Si és així assignarà el recurs lliure al robot R2. Un cop finalitzada l'acció del robot R2 amb aquest recurs, l'alliberarà. En canvi al robot R2, aquest programa simplement copiarà els valors de les variables RC del robot R1 a les corresponents variables de R2. Hi haurà una part de codi com la següent per a cada recurs compartit que tingui el sistema.

```
//Programa gestioRC() del robot R1
begin
    call mutex(bRCP4)
    if RCP4==0 and Pin[1]==2
        RCP4=2
    else
        if RCP4==2 and Pin[1]==0
            RCP4=0
```

```
//Programa gestioRC() del robot R2
begin
    call mutex(bRCP4)
    RCP4=Pin[1]
    bRCP4=false
end
```

```
            endIf
        endIf
        bRCP4=false
end
```

8. Un programa anomenat **mutex()** que implementarà l'exclusió mútua de totes les variables que es fan servir en diferents tasques.

```
begin
            //l'assignació de recurs (bRC) a true s'ha de fer a la mateixa línia de prova
            wait((bRC==false) and (bRC=true))
end
```

9. Excepcions.

9.1 Com es veu en l'esquema del punt 1 i ha dues excepcions al cicle de les transicions en el programa **start()**: les transicions que a la seva entrada tenen un recurs compartit (*Figura 5.3*) i les transicions que tenen el recurs compartit a la sortida (*Figura 5.4*). La implementació d'aquests dos casos serà lleugerament diferent per a cada robot, ja que per accedir al recurs compartit el robot R2 ha de fer una petició a R1, i aquest assignar-li o no. En el primer cas, el codi tindrà el següent aspecte:

```
//codi del punt C.1 de R1
flag=0
if ntokens_P1<1
            flag=1
endIf
call mutex(bRCP4)
if flag==0 and RCP4==0
            RCP4=1
            bRCP4=false
            call transition_T1()
else
            bRCP4=false
endIf
```

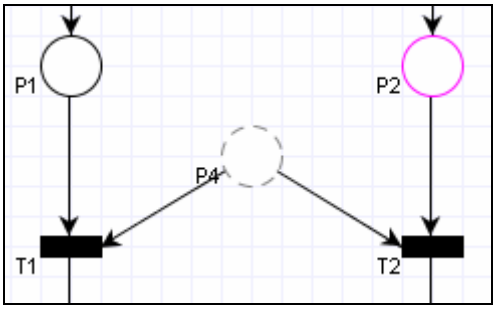


Figura 5.3 Exemple de recurs compartit a l'entrada de les transicions. El lloc P4 representa un recurs compartit (gris a ratlles). El lloc P1 pertany al robot R1 (negre) mentre que el lloc P2 pertany al robot R2 (magenta).

```
//codi del punt C.1 de R2
flag=0
if ntokens_P2<1
        flag=1
endIf
if flag==0
      call mutex(bPout)
       Pout[1]=2
       bPout=false
       call mutex(bRCP4)
       if RCP4==2
            bRCP4=false
            call transition_T2()
       else
            bRCP4=false
       endIf
endIf
```

Aquest exemple mostra com seria un mateix tros de codi per R1 i R2. La primera part és igual però en el moment de demanar el recurs compartit, mentre el robot R1

comprova si està lliure la seva variable del recurs compartit RCP4 directament , el robot R2 ha de fer una petició del recurs a través del vector Pout. Un cop feta la petició, la seva variable RCP4 ja s'haurà actualitzat amb el valor enviat de R1 a través de la tasca de comunicació. Com que les variables RC i Pout s'utilitzen en varies tasques, la resta de codi són les crides a l'exclusió mútua i actualitzacions dels valors usats per aquesta.

En el cas que la transició tingui a la sortida un recurs compartit, el codi a generar serà similar a l'anterior, però mentre R1 allibera el recurs compartit directament (assigna el valor 0 a la variable RC), R2 envia a R1 l'alliberació en forma de vector Pout i seguidament assigna 0 a la seva variable RC.

```
//codi del punt C.3 de R1
flag=0
if ntokens_P1<1
        flag=1
endIf
if flag==0
        call mutex(bRCP4)
        RCP4=0
        bRCP4=false
        call transition_T3()
endIf
```

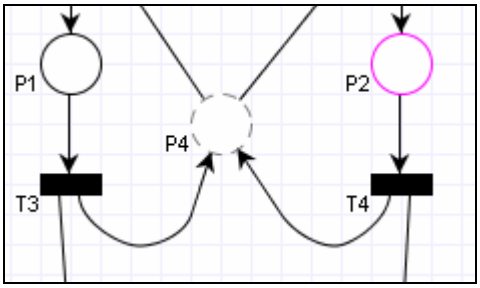


Figura 5.4 Exemple de recurs compartit a la sortida de les transicions.

```
//codi del punt C.3 de R2
flag=0
if ntokens_P2<1
        flag=1
endIf
if flag==0
        call mutex(bPout)
        Pout[1]=0
        bPout=false
        call mutex(bRCP4)
        RCP4=0
        bRCP4=false
        call transition_T4()
endIf
```

9.2 Per tal d'implementar xarxes amb la possibilitat d'elecció, és a dir, que des d'un lloc es pot accedir a més d'una transició de sortida, s'hauria de disposar d'elements complexes que dificultarien tan la generació automàtica del codi com la creació del model. No serà possible doncs, implementar aquest tipus de xarxa, amb l'excepció que una de les transicions de sortida només tingui com a lloc d'entrada un recurs compartit (*Figura 5.5*). Això permetrà desenvolupar sistemes complexes sense un gran esforç en la generació de codi, ja que el que es farà és col·locar la part de codi d'aquestes transicions en primer terme del llaç principal del robot. Així, el conflicte se soluciona donant preferència sempre a les accions a executar que utilitzen el recurs compartit per sobre de les accions de l'altre transició.

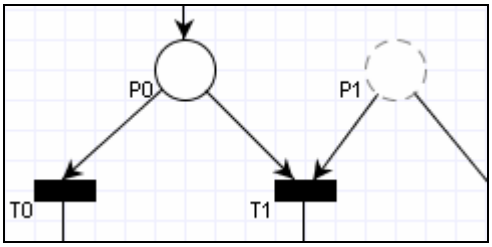


Figura 5.5 Exemple de lloc amb possibilitat d'elecció

En aquest exemple, el lloc P0 podria tenir un conflicte d'elecció entre les transicions de sortida T0 i T1, però com que T1 només té a l'entrada el lloc P1 de tipus recurs compartit i es codificarà a l'inici del llaç, pel simple fet de ser una programació seqüencial es donarà preferència automàticament a les accions de T1 en cas que el recurs compartit estigui disponible.

A més dels programes comentats, per poder guardar el codi com a un projecte Val3 es necessiten un conjunt de fitxers més: es tracta dels fitxers *pjx*, *dtx* i *ltx* comentats en l'apartat 4.3.2 El contingut d'aquests fitxers dependrà del sistema modelat, però no el número de fitxers a generar ja que els únics fitxers dels quals no se sap quants se n'han de generar son els de format *pgx*.

5.2.2. Cas d'un sol robot o dos robots sense interactuar entre ells

Aquest cas és molt més senzill i es pot explicar com a una simplificació del cas anterior. La diferència entre modelar un o dos robots només serà que en el cas de tenir-ne dos es generaran dos codis independents (o dos projectes independents en cas de gravar-ho com a projecte Val3). Del conjunt de variables explicades només farà servir les mencionades en els punts 1 i 2. En quan als programes usats, com que no hi haurà recursos compartits només generarà els programes dels punts 1, 2 i 3, obviant tots els programes que implementen la comunicació i l'exclusió mútua. Els programes dependents únicament de guardar el codi com a projecte de Val3 s'han de generar igualment.

5.3. Validació experimental

Un cop explicat el funcionament del mòdul generador de codi es comprova la seva efectivitat a través d'un exemple complert del que seria la creació d'una aplicació qualsevol per part de l'usuari.

5.3.1. Sistema a modelar

El sistema a modelar està format per dos robots Stäubli que treballen en paral·lel en una mateixa cel·la de treball. Les accions que han de realitzar son molt senzilles però a la vegada suficientment entenedores per il·lustrar la funcionalitat del projecte.

Si es mira el cicle de treball dels dos robots de manera independent s'observa un comportament simètric: es defineix per a cada robot vuit posicions a l'espai que representen els espais de treball dels robots i els llocs d'espera (*Figura 5.6*). Cada robot es desplaça des del seu punt J0 fins al seu punt J7 passant per tots els punts intermitjos i tanca el cicle de treball tornant de J7 a J0 per el mateix camí de tornada.

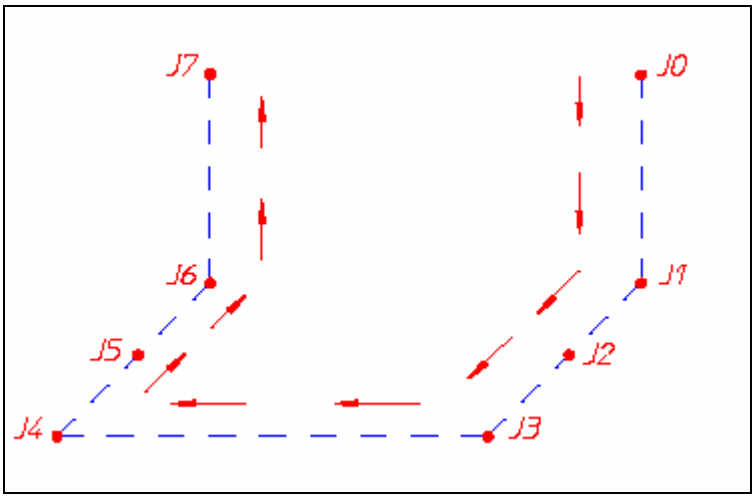


Figura 5.6 Moviment d'anada de cada robot

Quan es miren els dos cicles conjuntament el que passa és que existeix una zona de treball a la qual han d'accedir els dos robots. L'espai comprès entre J3 i J4 de cada robot és una zona compartida pels dos robots tan en el camí d'anada com en el de tornada. Com és evident no poden accedir a la zona compartida a la vegada. Per això es defineixen els llocs J2 i J5 com els punts d'espera de l'anada i de la tornada en cas que el camí estigui ocupat per l'altre robot.

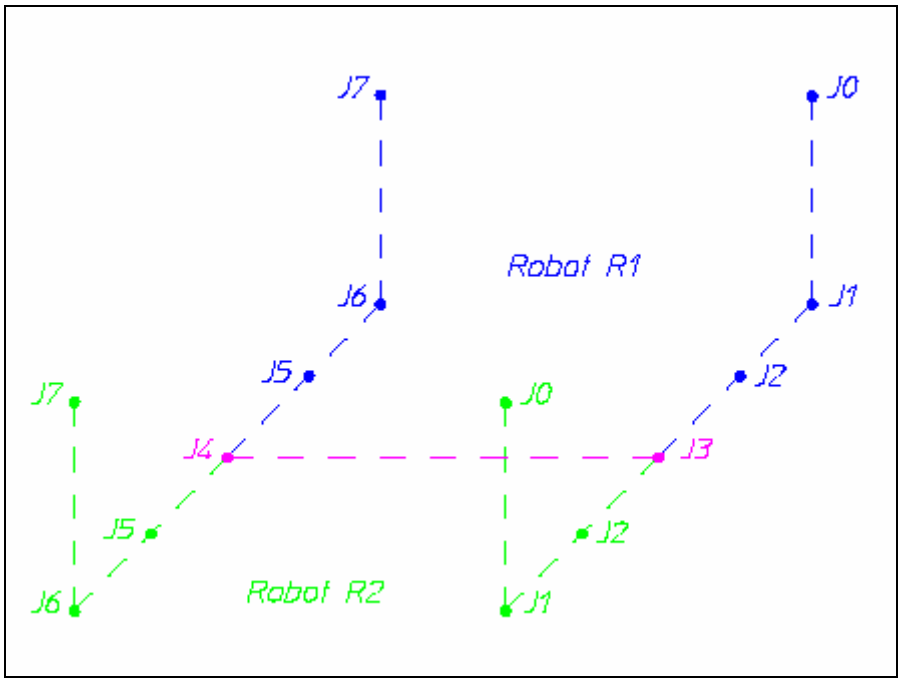


Figura 5.7 Cicle dels dos robots. La zona central J3-J4 és compartida

En aquest exemple els robots només es desplacen d'un lloc a l'altre, però en una aplicació real en el moment d'arribar als llocs determinats podrien fer les accions que volgués l'usuari.

5.3.2. Descripció del material

Per desenvolupar aquesta aplicació es disposa dels següents elements:

- Dos robots de la marca Stäubli, model TX-90. Un està dotat d'una mà robotitzada i l'altre d'una pinça, però en aquesta aplicació no es fan servir cap dels dos elements terminals.
- Dos controladors CS8 marca Stäubli, un per a cada robot que gestionen el control.
- Una botonera marca Stäubli que serveix per ensenyar de manera manual les posicions que han d'aprendre els robots.
- Un PC connectat en xarxa amb els dos controladors. Aquest ordinador disposa dels següents programes:
    - *Platform Independent Petri Net Editor* amb les modificacions desenvolupades en el present projecte.

- *Stäubli Robotics Studio*, que conté un conjunt de programes dels quals s'usaran el *Val3 Studio* i el *CS8 Emulator*.

5.3.3. Modelat del sistema en xarxa de Petri

Un cop definit el sistema i els elements que hi intervindran es comença a desenvolupar l'exemple. El primer pas és modelar el sistema descrit mitjançant una xarxa de Petri, que es realitza a través del programa PIPE modificat.

El sistema té dos robots que tenen un comportament cíclic, per tan el model ha de tenir dues branques que representin aquests cicles. Es separen les accions dels robots en cinc grups:

1- Moviments des de J0 fins a J2.

2- Moviments des de J2 fins a J5.

3- Moviments des de J5 fins a J7 i retorn a J5.

4- Moviments des de J5 fins a J2.

5- Moviments des de J2 fins al punt inicial J0.

Aquesta separació és deguda a la necessitat de tenir dos punts d'espera en cas que el camí compartit estigui ocupat per l'altre robot. Cada un d'aquests moviments es modela amb un lloc per cada robot, els quals seran de tipus R1 o R2 en funció del robot que executa les accions. A més d'aquests llocs es modela un lloc addicional per cada robot que representa l'inici del cicle i permet la seva repetició; aquests llocs tindran una marca inicialment.

El punt clau és el modelat de l'espai compartit. Es podria pensar d'usar dos recursos compartits perquè s'accedeix de dues maneres diferents a aquest espai, una a l'anada i una a la tornada. Això no és correcte ja que realment el recurs que es comparteix és el mateix encara que un robot hi vagi d'anada i l'altre de tornada del cicle. Per tan es modela aquest espai compartit com un lloc que se li assignarà el tipus de recurs compartit. Cada branca dels robots fa una petició del recurs un cop finalitzats els moviments 1 i 3 de la llista anterior, mentre que allibera el recurs un cop finalitzats els moviments 2 i 4. En definitiva, tots aquests aspectes es modelen amb la xarxa de Petri de la *Figura 5.8.*

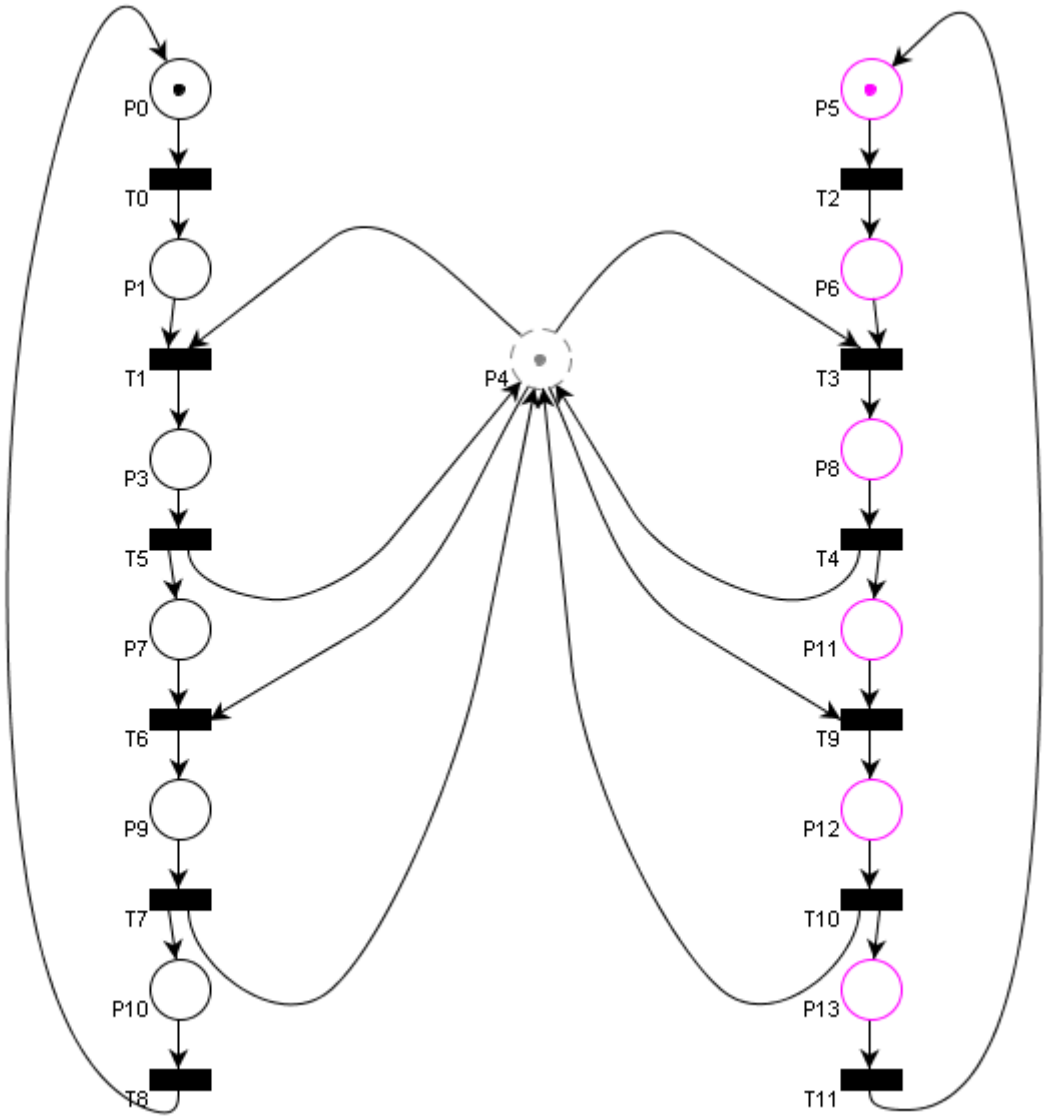


Figura 5.8 Xarxa de Petri que modela el sistema d'exemple

Com s'observa, la branca de l'esquerra el formen llocs de tipus *R1* (color negre) mentre que la branca de la dreta el formen llocs de tipus *R2* (color magenta). El lloc P4 del mig representa el recurs compartit (color gris a ratlles) i comença marcat ja que a l'inici cap dels robots utilitza el recurs. Totes dues branques accedeixen i alliberen el recurs dues vegades, com s'ha explicat. Els llocs P0 i P5 simulen l'inici del cicle i per tan comencen marcats. La resta de llocs, cinc a cada branca, representen les cinc accions dels robots llistades anteriorment i en el mateix ordre, per tan el codi que contindran serà simètric, igual que la xarxa. Per escollir el tipus de cada lloc es fa a través del menú desplegable quan es prem el botó dret del ratolí sobre un lloc (*Figura 7.4*). Es recomana llegir l'annex D (manual d'usuari) a l'hora de realitzar la xarxa en el programa *PIPE*.

Un cop simplificat el sistema en una xarxa de Petri, es poden usar els mòduls d'anàlisi de *PIPE* per trobar els elements que la representen, com la matriu d'incidència, espais d'estats, etc., i comprovar les propietats. Es verifica que la xarxa és de tipus *xarxa de Petri simple* (apartat 2.2.3), està acotada i no te bloquejos, del qual es pot concloure que es tracta d'una xarxa vàlida per modelar un sistema cíclic i procedir a la introducció del codi.

5.3.4. Generació i edició del codi

Des de l'editor de PIPE s'accedeix al camp *View/Edit Code* del menú desplegable de cada lloc (*Figura 7.3*) a través del botó dret del ratolí i s'introdueix un per un el codi. Com que el sistema és simètric, s'introdueix el codi a tots els llocs de la branca de R1 i es copia el codi al lloc corresponent de la branca de R2. Així als llocs P0 i P5 se'ls introdueix un comentari d'inici de cicle, ja que realment no executen codi. La instrucció que s'usa per moure els robots és **movej()**. Als llocs P1 i P6 s'introdueix:

```
movej(j[0],flange,nom_speed)
movej(j[1],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
```

on la instrucció **waitEndMove()** assegura que el robot finalitza el moviment i no procedirà a fer cap acció més fins que s'aturi per complert al lloc j[2]. Aquesta nomenclatura indica que s'ha de declarar un vector j de vuit elements per a cada robot que representaran les seves posicions. La variable nom_speed serà l'encarregada de definir les característiques del moviment, com la velocitat i l'acceleració. Finalment flange és la variable de tipus *tool* que obligatòriament han de tenir totes les aplicacions, com s'ha vist a l'apartat 4.3.1. Aquests tres elements s'hauran de definir des del *Val3 Studio* un cop generat el codi.

Als successius llocs de cada branca s'introdueix una estructura de codi igual que l'anterior, però canviant els índexs del vector j i afegint si cal alguna instrucció de moviment més per tal de seguir els cinc moviments descrits anteriorment (per entendre el codi d'aquesta aplicació, i també per desenvolupar altres aplicacions més complexes es recomana llegir el manual de referència de Val3 inclòs a l'annex E).

Tot seguit s'entra al mòdul de programació i es prem el botó de generació de codi (*Figura 5.1*). Els resultats són molt destacables, ja que només amb la creació d'una xarxa de Petri i un simple clic a un botó s'obté de manera senzilla tot el codi de control dels robots per aquesta aplicació, que es presenta a continuació:

```
//Generador de Codi Val3 del robot R1
```

```
//Generador de Codi Val3 del robot R2
```

```
//programa start()
begin
  //Inicialitzacions
  ntokens_P0=1
  ntokens_P1=0
  ntokens_P10=0
  ntokens_P3=0
  RCP4=0
  bRCP4=false
  Pin[1]=0
  ntokens_P7=0
  ntokens_P9=0

  sioLink(buffer_in, io:Server)
  taskCreate "comunicacio",20,comunicacio()

  //Cicle principal
  while(true)
    //Codi de la transició T1
    flag=0
    if ntokens_P1<1
      flag=1
    endIf
    call mutex(bRCP4)
    if flag==0 and RCP4==0
      RCP4=1
      bRCP4=false
      call transition_T1()
    else
      bRCP4=false
    endIf
    //Codi de la transició T6
    flag=0
    if ntokens_P7<1
      flag=1
    endIf
    call mutex(bRCP4)
    if flag==0 and RCP4==0
      RCP4=1
      bRCP4=false
      call transition_T6()
    else
      bRCP4=false
    endIf
    //Codi de la transició T0
    flag=0
    if ntokens_P0<1
      flag=1
    endIf
    if flag==0
      call transition_T0()
    endIf
    //Codi de la transició T5
    flag=0
    if ntokens_P3<1
      flag=1
    endIf
    if flag==0
      call mutex(bRCP4)
      RCP4=0
      bRCP4=false
      call transition_T5()
    endIf
```

```
//programa start()
begin
  //Inicialitzacions
  ntokens_P11=0
  ntokens_P12=0
  ntokens_P13=0
  RCP4=0
  bRCP4=false
  ntokens_P5=1
  ntokens_P6=0
  ntokens_P8=0

  sioLink(buffer_out, io:Client)
  taskCreate "comunicacio",20,comunicacio()

 //Cicle principal
  while(true)
    //Codi de la transició T3
    flag=0
    if ntokens_P6<1
      flag=1
    endIf
    if flag==0
      call mutex(bPout)
      Pout[1]=2
      bPout=false
      call mutex(bRCP4)
      if RCP4==2
        bRCP4=false
        call transition_T3()
      else
        bRCP4=false
      endIf
    endIf
    //Codi de la transició T9
    flag=0
    if ntokens_P11<1
      flag=1
    endIf
    if flag==0
      call mutex(bPout)
      Pout[1]=2
      bPout=false
      call mutex(bRCP4)
      if RCP4==2
        bRCP4=false
        call transition_T9()
      else
        bRCP4=false
      endIf
    endIf
    //Codi de la transició T10
    flag=0
    if ntokens_P12<1
      flag=1
    endIf
    if flag==0
      call mutex(bPout)
      Pout[1]=0
      bPout=false
      call mutex(bRCP4)
      RCP4=0
      bRCP4=false
```

```
    //Codi de la transició T7
    flag=0
    if ntokens_P9<1
      flag=1
    endIf
    if flag==0
      call mutex(bRCP4)
      RCP4=0
      bRCP4=false
      call transition_T7()
    endIf
    //Codi de la transició T8
    flag=0
    if ntokens_P10<1
      flag=1
    endIf
    if flag==0
      call transition_T8()
    endIf
  endWhile
end

//Subprogrames de les transicions
begin
  ntokens_P0=ntokens_P0-1
  call place_P1()
  ntokens_P1=ntokens_P1+1
end

begin
  ntokens_P1=ntokens_P1-1
  call place_P3()
  ntokens_P3=ntokens_P3+1
end

begin
  ntokens_P3=ntokens_P3-1
  call place_P7()
  ntokens_P7=ntokens_P7+1
end

begin
  ntokens_P7=ntokens_P7-1
  call place_P9()
  ntokens_P9=ntokens_P9+1
end

begin
  ntokens_P9=ntokens_P9-1
  call place_P10()
  ntokens_P10=ntokens_P10+1
end

begin
  ntokens_P10=ntokens_P10-1
  call place_P0()
  ntokens_P0=ntokens_P0+1
end

//Subprogrames dels llocs
//Programa de place_P1()
begin
movej(j[0],flange,nom_speed)
```

```
    call transition_T10()
  endIf
  //Codi de la transició T11
  flag=0
  if ntokens_P13<1
    flag=1
  endIf
  if flag==0
    call transition_T11()
  endIf
  //Codi de la transició T2
  flag=0
  if ntokens_P5<1
    flag=1
  endIf
  if flag==0
    call transition_T2()
  endIf
  //Codi de la transició T4
  flag=0
  if ntokens_P8<1
    flag=1
  endIf
  if flag==0
    call mutex(bPout)
    Pout[1]=0
    bPout=false
    call mutex(bRCP4)
    RCP4=0
    bRCP4=false
    call transition_T4()
  endIf
 endWhile
end

//Subprogrames de les transicions
begin
  ntokens_P12=ntokens_P12-1
  call place_P13()
  ntokens_P13=ntokens_P13+1
end

begin
  ntokens_P13=ntokens_P13-1
  call place_P5()
  ntokens_P5=ntokens_P5+1
end

begin
  ntokens_P5=ntokens_P5-1
  call place_P6()
  ntokens_P6=ntokens_P6+1
end

begin
  ntokens_P6=ntokens_P6-1
  call place_P8()
  ntokens_P8=ntokens_P8+1
end

begin
  ntokens_P8=ntokens_P8-1
  call place_P11()
```

```
movej(j[1],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end

//Programa de place_P3()
begin
movej(j[3],flange,nom_speed)
movej(j[4],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end

//Programa de place_P7()
begin
movej(j[6],flange,nom_speed)
movej(j[7],flange,nom_speed)
waitEndMove()
movej(j[6],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end

//Programa de place_P9()
begin
movej(j[4],flange,nom_speed)
movej(j[3],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end

//Programa de place_P10()
begin
movej(j[1],flange,nom_speed)
movej(j[0],flange,nom_speed)
waitEndMove()
end

//Programa de place_P0()
begin
putln("Inici CICLE Robot 1")
end

//Programa auxiliar comunicacio()
begin
  while(true)
    call lectura()
    call gestioRC()
    call escriptura()
  endWhile
end

//Programa gestio dels recursos compartits gestioRC()
begin
  call mutex(bRCP4)
  if RCP4==0 and Pin[1]==2
    RCP4=2
  else
    if RCP4==2 and Pin[1]==0
      RCP4=0
    endIf
  endIf
  bRCP4=false
  ntokens_P11=ntokens_P11+1
end

begin
  ntokens_P11=ntokens_P11-1
  call place_P12()
  ntokens_P12=ntokens_P12+1
end

//Subprogrames dels llocs
//Programa de place_P13()
begin
movej(j[1],flange,nom_speed)
movej(j[0],flange,nom_speed)
waitEndMove()
end

//Programa de place_P5()
begin
putln("Inici CICLE Robot 2")
end

//Programa de place_P6()
begin
movej(j[0],flange,nom_speed)
movej(j[1],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end

//Programa de place_P8()
begin
movej(j[3],flange,nom_speed)
movej(j[4],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end

//Programa de place_P11()
begin
movej(j[6],flange,nom_speed)
movej(j[7],flange,nom_speed)
waitEndMove()
movej(j[6],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end

//Programa de place_P12()
begin
movej(j[4],flange,nom_speed)
movej(j[3],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end

//Programa auxiliar comunicacio()
begin
  while(true)
    call escriptura()
    call lectura()
    call gestioRC()
  endWhile
```

```
end

//Programa escriptura del buffer_out escriptura()
begin
  SRC0=toString("1",9)
  call mutex(bRCP4)
  SRCP4=toString("1",RCP4)
  bRCP4=false
  sioLink(buffer_out,io:Server)
  buffer_client=SRC0+" "+SRCP4
  buffer_out=buffer_client
  putln(buffer_client)
end

//Programa lectura del buffer_in lectura()
begin
  sioLink(buffer_in,io:Server)
  buffer_server=buffer_in
  putln(buffer_server)
  cont=0
  do
    buffer_server=toNum(buffer_server,Pin[cont],cond)
    cont=cont+1
  until(cond != true)
end

//Programa d'exlusió mutua mutex(bool& bRC)
begin
  wait((bRC==false) and (bRC=true))
end
```

```
end

//Programa gestio dels recursos compartits gestioRC()
begin
  call mutex(bRCP4)
  RCP4=Pin[1]
  bRCP4=false
end

//Programa escriptura del buffer_out escriptura()
begin
  SRC0=toString("1",9)
  call mutex(bPout)
  SRCP4=toString("1",Pout[1])
  bPout=false
  buffer_client=SRC0+" "+SRCP4
  sioLink(buffer_out,io:Client)
  buffer_out=buffer_client
end

//Programa lectura del buffer_in lectura()
begin
  sioLink(buffer_in,io:Client)
  buffer_server=buffer_in
  putln(buffer_server)
  cont=0
  do
    buffer_server=toNum(buffer_server,Pin[cont],cond)
    cont=cont+1
  until(cond != true)
end

//Programa d'exlusió mutua mutex(bool& bRC)
begin
  wait((bRC==false) and (bRC=true))
end
```

Un cop revisat el codi presentat en les dues finestres del mòdul es prem el botó de creació de projectes (els fitxer generats s'adjunten en l'annex C) i seguidament s'escull el directori que es desitgi des de l'explorador de carpetes que apareix (*Figura 7.7*).

Com s'ha mencionat, el codi que s'ha introduït requereix de la declaració de les variables flange i nom_speed, a més de definir el vector j amb els vuit punts de l'espai per a cada robot. Les dues primeres variables s'introdueixen des de *Val3 Studio* (*Figura 4.2*) tot obrint els dos projectes generats que el propi mòdul ha anomenat Robot_R1 i Robot_R2. També es defineixen els paràmetres de la comunicació, indicant que el robot R1 fa de servidor i que R2 de client, tot introduint les direccions IP i la seva relació en la comunicació a través de socket ethernet.

Per enviar els programes als robots s'obren dues finestres del *CS8 emulator* (*Figura 5.9*), una per a cada robot i des del Val3 Studio s'envia el projecte Robot_R1 a la direcció IP del primer robot i el projecte Robot_R2 a la IP corresponent al segon robot. Des de cada finestra del *CS8 emulator* es carrega el corresponent projecte i ja només cal definir els vectors de posicions per executar l'aplicació. Aquestes posicions a l'espai les aprèn cada robot a través

de la botonera, ja que per fer-ho des de *Val3 Studio* s'haurien de conèixer les coordenades concretes de cada posició, que en aquest cas es desconeixen. Se situen els robots manualment a les vuit posicions, una a una, i des de la botonera mateixa es fa que les guardin en les variables corresponents al vector j[].

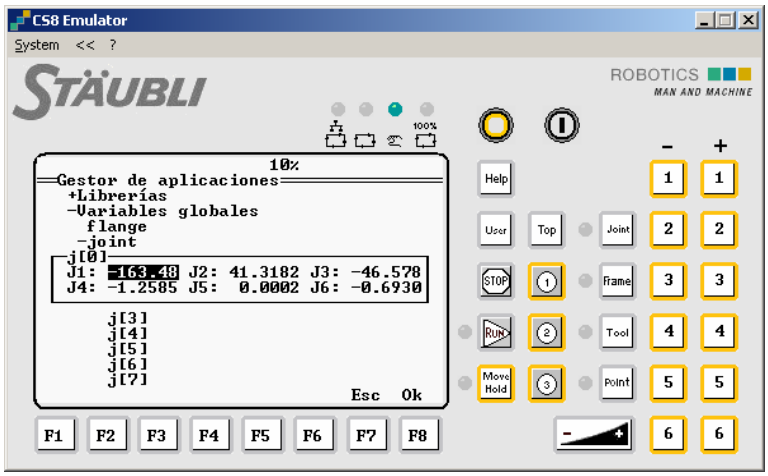


Figura 5.9 Programa CS8 Emulator amb les coordenades de la posició j[0]

Ara ja es pot executar l'aplicació, cridant el programa **start()** a cada un dels robots des de l'emulador. La validesa d'aquest exemple queda reflectida en el vídeo adjuntat amb la memòria del present projecte (*Figura 5.10*).

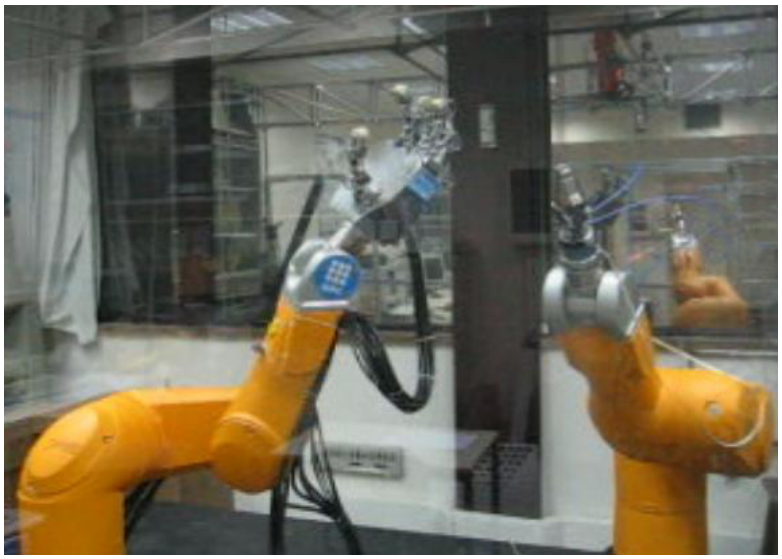


Figura 5.10 Captura del vídeo que mostra el funcionament de l'aplicació

6. Eines de desenvolupament

6.1. Introducció

En aquest capítol es descriuen les eines necessàries per a desenvolupar el mòdul presentat en l'apartat anterior. Quan es parla de programació aquestes eines són evidentment els llenguatges i els entorns de programació. Es diferencien els diferents llenguatges que intervenen en el desenvolupament del projecte. El primer és el llenguatge de programació de PIPE, que es tracta de *Java*. Després també s'ha de considerar el format en què es guarden les xarxes. Aquest és el *PNML* que com s'ha comentat en les especificacions també s'haurà de modificar per tal que la xarxa guardi la informació relacionada amb el codi a generar. S'ha de tenir en compte que en aquest apartat no es fa referència al llenguatge de programació dels robots, el Val3, ja que no es tracta de la metodologia a seguir per dur a terme el projecte, sinó la manera de presentar els resultats.

6.2. El llenguatge de programació Java

El llenguatge de programació Java va ser dissenyat el 1990 per James Gosling de *Sun Microsystems* pensat en primera instància per dispositius electrònics com calculadores, microones i televisió interactiva. Sun es va adonar que la televisió interactiva no seria un gran èxit a curt plaç i es va plantejar noves estratègies per produir beneficis, entre elles es trobava l'aplicació de Java a Internet ja que el llenguatge s'adaptava perfectament a les seves necessitats. Java aporta a Internet la interactivitat que s'havia buscat durant molt de temps entre usuari i aplicació.

6.2.1. Principals característiques de Java

Orientació a objectes

L'orientació a objectes és un paradigma de programació que facilita la creació de software de qualitat pels seus factors que potencien el manteniment, l'extensió i la reutilització de software generat sota aquest paradigma. La programació orientada a objectes intenta assemblar-se a la manera de pensar de l'home i no al de la màquina. Això es possible gràcies a la forma racional amb la que es manegen les abstraccions que representen les entitats del domini del problema, i a propietats com la jerarquia o l'encapsulament. L'element

bàsic d'aquest paradigma no és la funció com a la programació estructurada, sinó una entitat denominada objecte. Un objecte és la representació d'un concepte per un programa i conté tota la informació necessària per abstraure les dades que descriuen l'estat i les operacions que poden modificar-lo. Java incorpora l'orientació a objectes com un dels pilars bàsics del seu llenguatge.

Robust

Java verifica el seu codi al mateix temps que s'escriu, i una vegada més abans d'executar-se, de manera que s'aconsegueix un alt marge de codificació sense errors. Al ser estricte en quant a tipus i declaracions, es descobreixen la major part dels errors durant el temps de compilació, i així el que sembla rigidesa i falta de flexibilitat es converteix en eficàcia. Respecte a la gestió de memòria, Java allibera el programador del compromís d'haver de controlar especialment l'assignació que se'n fa a les necessitats específiques. Posseeix una gestió avançada de memòria i el maneig d'excepcions orientat a objectes integrat.

Simple

L'únic requisit per aprendre Java és tenir un coneixement dels conceptes bàsics de programació orientada a objectes. Així el llenguatge permet mostrar qualsevol plantejament per part del programador sense que les interioritats subjacents siguin desvelades. Java és més senzill que qualsevol altre entorn de programació, i l'únic obstacle que es pot presentar és comprendre la programació orientada a objectes. A més el paquet d'utilitats de Java ve amb un conjunt complet d'estructures de dades complexes i els seus mètodes associats, que son d'inestimable ajuda per implementar aplicacions complexes.

Interactiu i orientat a la xarxa

Java té la capacitat de crear programes en xarxa interactius i és capaç de fer varies tasques a la vegada mitjançant la utilització de múltiples fils de programació (*multithread*). També és capaç de situar figures animades en les pàgines Web amb textos que es desplacen per la pantalla. En Java es poden desenvolupar *applets*, que són petits programes transferits dinàmicament a través d'Internet que poden presentar comportament intel·ligent, podent reaccionar a l'entrada d'un usuari i canviar de forma dinàmica.

6.2.2. Java: llenguatge de programació de PIPE

PIPE, com la majoria d'aplicacions d'edició de xarxes de Petri (annex A), està programat en Java. La principal raó rau en la facilitat d'adequar les característiques dels components de les xarxes de Petri a l'orientació a objectes. Gràcies a l'orientació a objectes es defineix tota una jerarquia entre els elements que formen les xarxes, que mitjançant l'herència de classes

comunes facilita la programació i la reutilització de codi. Així per exemple els llocs i les transicions es defineixen com a objectes que hereten de la mateixa classe pare (*PlaceTransitionObject* en aquest cas) propietats com el nom, la posició x i y de la representació gràfica, el color, la capacitat de ser seleccionats, etc. a més de les funcions per modificar o obtenir aquestes propietats. L'orientació d'objectes també és essencial en el desenvolupament dels mòduls d'anàlisi implementats, en la capacitat de carregar nous mòduls i en la interfície gràfica. A més, com s'ha comentat Java proporciona una simplicitat a l'hora de programar que altres llenguatges de programació orientats a objectes (com *C++*) no tenen.

6.2.3. Entorn de programació: Plataforma Eclipse

Un cop conegut el llenguatge de programació que s'utilitza s'ha d'escollir una plataforma d'edició adient. Com és de suposar les possibilitats són moltes i l'elecció s'ha de fer en criteris de facilitat alhora de desenvolupar el nou mòdul i en la modificació del codi intern de PIPE. Ja que PIPE és un programa de codi lliure i gratuït té sentit pensar en una plataforma que compleixi les mateixes característiques.

Existeixen molts entorns de programació de software molt potents però pels quals s'han de pagar costoses llicències com *Visual Studio* (Microsoft) o *Visual Age* (IBM). Des de fa un temps es disposa d'una eina de potència similar i de lliure distribució. Es tracta d**'Eclipse** [8], una plataforma d'eines universal, que com suggereixen definir els seus creadors, és un entorn integrat de desenvolupament (*IDE*) obert i extensible, per a qualsevol cosa i res en particular. Així Eclipse és la plataforma que compleix amb les característiques desitjades, i a més té l'avantatge de ser l'entorn de programació en el qual es va desenvolupar PIPE, evitant així problemes derivats de la càrrega del codi o de la compilació.

Eclipse és neutral i adaptable a qualsevol tipus de llenguatge com per exemple *C/C++, Cobol*, *XML*, etc. tot i que està escrit en la seva major part en Java, s'executa sobre la màquina virtual d'aquesta i el seu ús més popular és com a IDE per a Java. La característica clau d'Eclipse és l'extensibilitat. Eclipse és una gran estructura formada per un nucli i molts *plug-ins* que van conformant la funcionalitat final. La forma en que els plug-ins interactuen és mitjançant interfícies o punts d'extensió, així, les noves aportacions s'integren sense dificultat ni conflictes.

La Plataforma Eclipse és un subprojecte integrat en un projecte de desenvolupament de software de codi obert anomenat *Projecte Eclipse* dedicat a proporcionar una plataforma industrial robusta, amb àmplies característiques i amb qualitat comercial pel desenvolupament d'eines altament integrades. Els altres subprojectes són el *Java Development Tool* i el *Plug-in Development Environment*.

Eclipse el formen el nucli, l'entorn de treball (*workspace*), l'àrea de desenvolupament (*workbench*), l'ajuda al equip (*team support*) i l'ajuda o documentació (*help*). En la *Figura 6.1* es mostra l'arquitectura d'Eclipse.

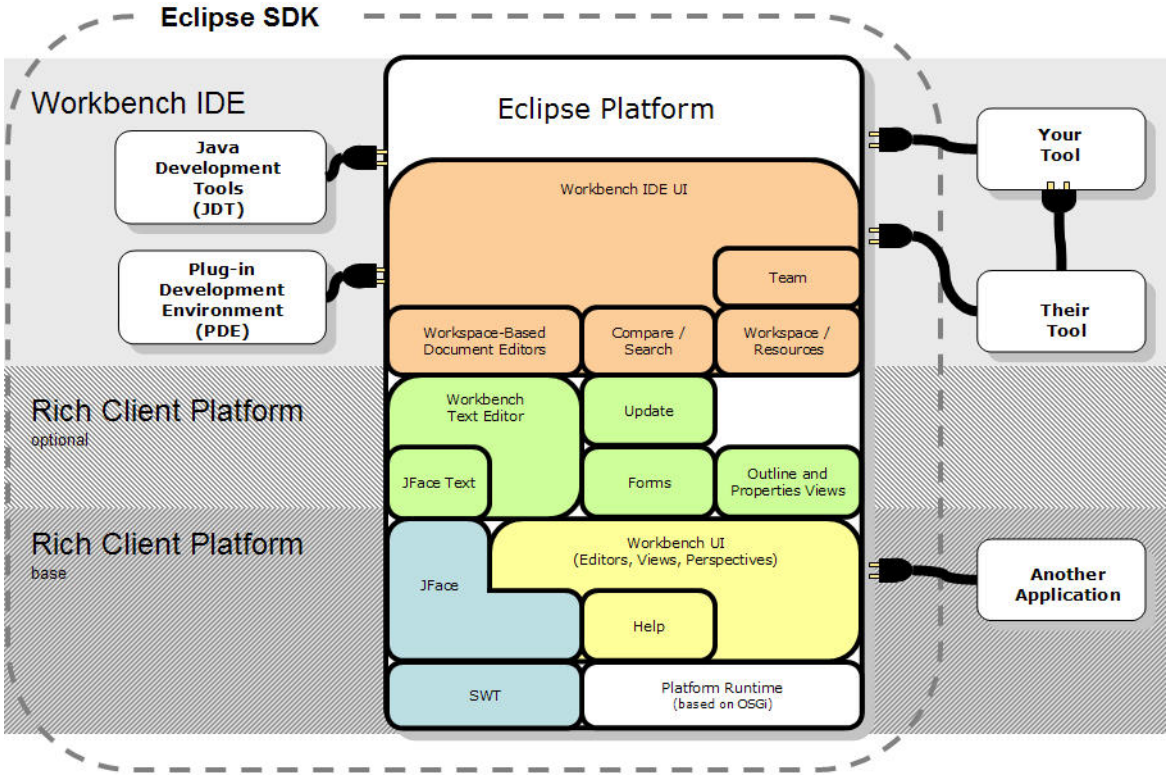


Figura 6.1 Arquitectura d'Eclipse

Nucli: la seva tasca és determinar quins són els *plug-ins* disponibles en el directori de *plug-ins* d'Eclipse. Com que la quantitat de *plug-ins* pot ser molt gran, només es carreguen els necessaris en el moment de ser utilitzats per tal de minimitzar el temps d'arrencada d'Eclipse i recursos.

Entorn de treball: maneja els recursos d'usuari, organitzats en un o més projectes. Cada projecte es correspon a un directori en el directori de treball d'Eclipse i conté arxius i carpetes.

Interfície d'usuari: mostra els menús i eines, i s'organitza en perspectives que configuren els editors de codi i les vistes. A diferència de moltes aplicacions escrites en Java, Eclipse té l'aspecte i es comporta com una aplicació nativa, és a dir, emula els gràfics natius a cada sistema operatiu (*Figura 6.2*).

Ajuda al grup: aquest *plug-in* facilita l'ús d'un sistema de control de versions per controlar els recursos en un projecte de l'usuari i defineix el procés necessari per guardar i recuperar d'un repositori. Eclipse inclou un client per a *CVS*.

Documentació: igual que el propi Eclipse, el component d'ajuda és un sistema de documentació extensible. Els proveïdors d'eines poden afegir documentació en format *HTML* i, usant *XML*, definir una estructura de navegació.

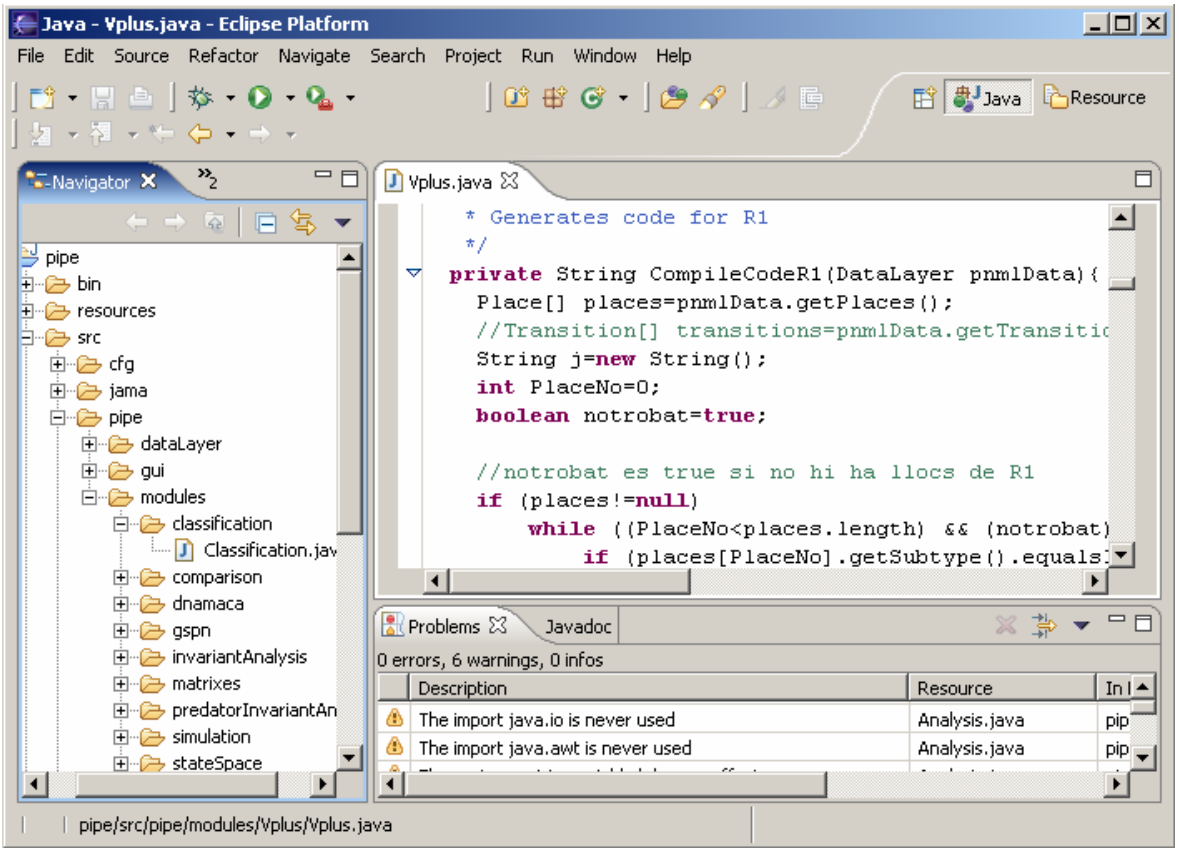


Figura 6.2 Interfície d'usuari de la Plataforma Eclipse

6.3. PNML: format estàndard per a xarxes de Petri

Una de les característiques principals necessàries per als programes d'edició de xarxes de Petri són les funcions per exportar-les a altres eines i importar-les des d'aquestes. El problema amb aquesta necessitat aparentment simple i purament tècnica és la multitud de diferents tipus de xarxes de Petri i la multitud de diferents eines, com es mostra en l'annex A, amb el consegüent nombre de tipus de fitxers per diferents xarxes. Això fa impossible de dotar els programes de totes les funcions d'importació i exportació desitjades. Aquesta situació va ser el punt de partida de la "*International Conference on Application and Theory of Petri Nets 2000*" per fer un esforç en l'estandardització dels formats i on es van fer diferents propostes de formats intercanviables basats en el codi *XML*.

El **PNML** (*Petri Net Markup Language*) va ser concebut per resoldre aquest problema. El seu disseny es va governar per les següents premisses:

Llegibilitat: El format hauria de ser llegible i editable amb eines convencionals.

Universalitat: El format no hauria d'excloure cap tipus de xarxa de Petri. Per tant, hauria de ser capaç de representar totes les versions de xarxes de Petri amb qualsevol tipus d'extensió.

Mutualitat: El format hauria de permetre extreure tanta informació com fos possible a partir de la xarxa, fins i tot si el seu tipus és desconegut.

Clarament, l'ús del *XML* garanteix la llegibilitat del format. La universalitat es pot garantir afegint tota la informació addicional d'un tipus particular de xarxa de Petri als objectes de la xarxa. Això s'aconsegueix a través d'etiquetar aquests objectes i la pròpia xarxa. Les etiquetes permeses i els seus possibles valors estan definits per un document anomenat **PNTD** (*Petri Net Type Definition*). La mutualitat es pot garantir amb les convencions, que són un conjunt d'etiquetes estandarditzades. Tècnicament, les convencions són una col·lecció extensible d'etiquetes amb la seva semàntica i el seu ús mes freqüent. Llavors, a partir d'aquestes etiquetes es pot construir un nou *PNTD*.

6.3.1. Conceptes generals del PNML

La universalitat és un dels aspectes principals del *PNML*, per tant, ha de ser suficientment general com per representar totes les versions de xarxes de Petri. Per altre part, la mutualitat requereix capturar l'essència de les xarxes i excloure tot tipus de sense sentit. Això s'aconsegueix mitjançant un format general, que està restringit a les necessitats d'una versió particular de xarxa de Petri definint un tipus de xarxa de Petri. Bàsicament el format general del *PNML* és un graf etiquetat amb dos tipus de nodes: llocs i transicions, però hi ha molts més conceptes, que es representen en la *Figura 6.3*.

Xarxes de Petri i objectes: Un fitxer que compleix els requeriments del format s'anomena fitxer de xarxa de Petri i pot contenir varies xarxes de Petri. Cada xarxa consisteix en objectes, on aquests bàsicament representen el graf d'estructura de la xarxa. Un objecte per tant, és un lloc, una transició o un arc. Per poder estructurar correctament la xarxa existeixen uns altres tres tipus d'objectes: pàgines, llocs de referència i transicions de referència. Cada objecte del fitxer de xarxa de Petri té un únic identificador, que es pot usar per a referir-se a aquest objecte. Per conveniència s'anomenen nodes als llocs o transicions.

Etiquetes: Per tal d'assignar un significat a un objecte, cada objecte pot tenir diferents etiquetes. Típicament una etiqueta representa el nom d'un node, el marcat inicial d'un lloc, la inscripció d'un arc o la guarda d'una transició. A més, la pròpia xarxa pot tenir diferents etiquetes. Existeixen dos tipus d'etiquetes, anotacions i atributs. Una anotació és una

etiqueta que té un domini infinit de valors legals, per exemple noms, marcats, inscripcions en arcs o diferents anotacions. Per altre part un atribut és una etiqueta amb un domini finit (i petit) de valors legals. Típicament el valor d'un atribut no es mostra textualment, però està representat en la forma, estil o color d'un objecte.

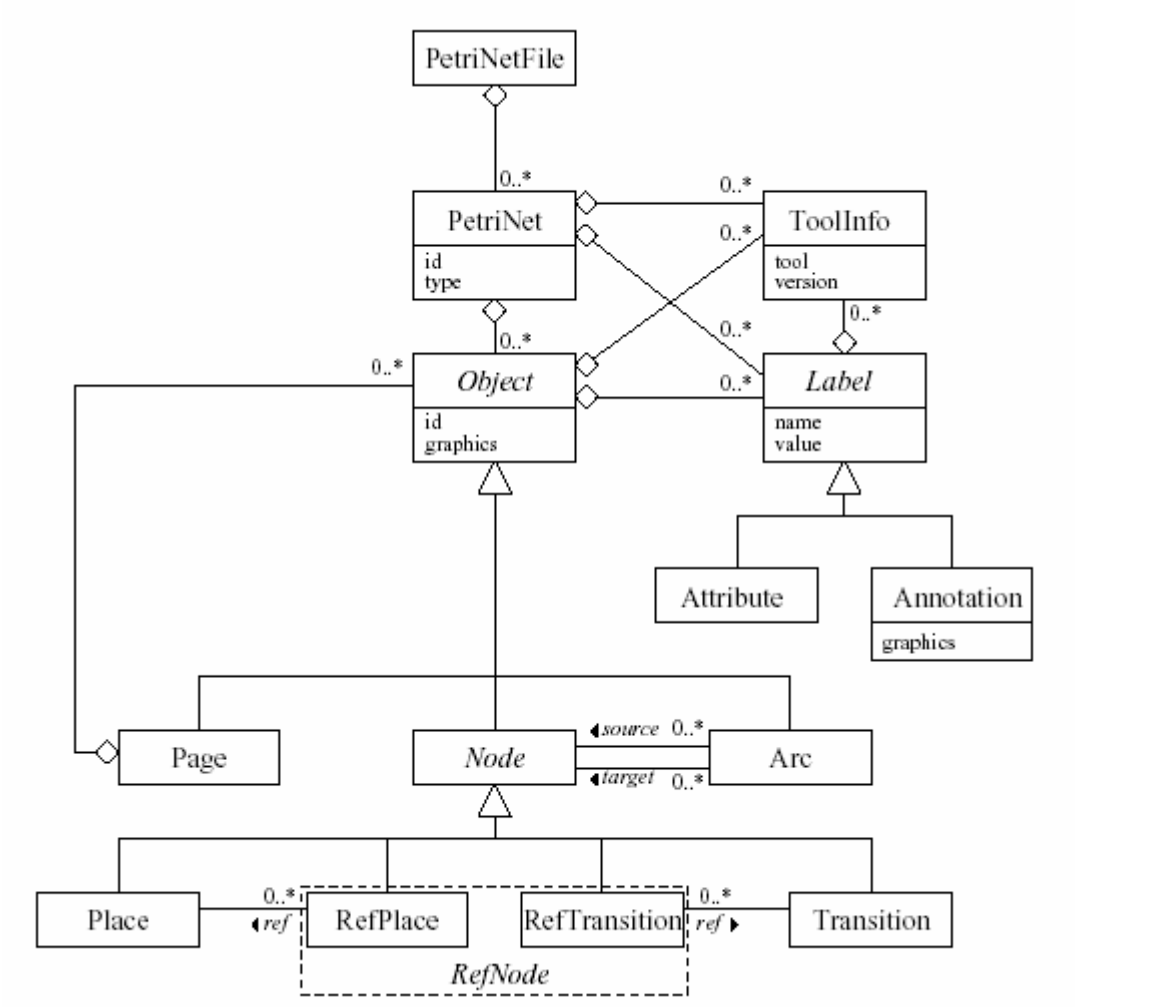


Figura 6.3 Esquema dels elements del PNML

Informació gràfica. Cada objecte està dotat d'algun tipus d'informació gràfica. Per un node, aquesta informació és la seva posició, per un arc és una llista de posicions que defineixen els punts intermitjos de l'arc i per una anotació és la seva posició relativa respecte a l'objecte corresponent. Totes aquestes posicions es representen en coordenades cartesianes *(x,y)*.

Informació específica de l'eina. Per algunes eines pot ser necessari guardar algun tipus d'informació interna, que se suposa no serà usada en altres eines. Per tal de guardar aquesta informació cada objecte i cada etiqueta es pot dotar amb informació específica d'aquesta eina. La forma interna d'aquesta informació depèn de l'eina que la usi, però ha

d'estar clarament etiquetada i assignada a un nom de l'eina específica. D'aquesta manera altres eines poden ignorar sense problemes aquesta informació.

Pàgines i nodes de referència. Una xarxa de Petri es pot estructurar amb l'ajuda de pàgines, que son objectes que poden consistir d'altres objectes. Un arc però, només pot connectar nodes de la mateixa pàgina. Per tal de connectar nodes de diferents pàgines es poden usar nodes de referència, que es pot referir a qualsevol node de la xarxa amb la única condició que no hi hagi referències cícliques.

6.3.2. Definició de tipus (PNTD) i convencions

Les etiquetes disponibles i les combinacions legals d'aquestes per un objecte particular estan definides per un tipus de xarxa de Petri. Tècnicament un tipus de xarxa de Petri es un document que defineix la sintaxi *XML* de les etiquetes; aquest document és un *DTD* (*document type definition*) que s'anomena **PNTD**. Conceptualment, un *PNTD* és una especificació del format general mostrat a l'apartat anterior i afegeix les definicions de les etiquetes als objectes i a la xarxa respectivament.

En principi un *PNTD* es pot definir lliurement. A la pràctica però, un *PNTD* escull les etiquetes d'una col·lecció predefinida que se subministra en un document separat: les convencions. Aquestes convencions garanteixen que una mateixa etiqueta té el mateix significat en tots els *PNTD*. Això permet intercanviar xarxes a través de diferents eines amb un tipus de xarxa diferents però similars entre sí. Aquesta col·lecció de convencions no forma part del *PNML*, el *PNML* només dóna el mecanisme per definir les convencions i per incloure parts de les convencions a dins d'un tipus de xarxa de Petri.

La definició i el manteniment de les convencions és un procés obert que està en continua evolució. En el moment que es va desenvolupar el programa PIPE s'estava començant a instaurar el *PNML* com a format estàndard per a les aplicacions, per tant, el present projecte es basa en la definició de xarxa que el PIPE té definida i es limita a afegir les propietats necessàries per guardar la informació addicional que requereix, tot assegurant la compatibilitat entre les diferents versions de PIPE (amb, o sense el nou mòdul desenvolupat) i amb diferents aplicacions que utilitzen el *PNML*.

6.3.3. Sintaxi i exemples del PNML

En aquesta secció es presenta breument la sintaxi del *PNML* comentant alguns exemples més freqüents. Com s'ha vist el *PNML* està basat en el *XML* (*Extensible Markup Language*). La xarxa de Petri, els objectes i les etiquetes estan representats com a elements del *XML*.

Un element del *XML* està inclòs en un parell d'etiquetes, una inicial <*element*> i una final <*/element*>. Aquests elements poden tenir atributs *XML* que el doten d'informació addicional (no confondre amb els atributs dels objectes de les xarxes de Petri). Un atribut *XML* d'un element *XML* està representat per una assignació d'un valor a una clau (el nom de l'atribut) a l'etiqueta inicial de l'element <*element clau=valor*>. Aquests elements poden contenir text o mes elements *XML*. Un element que no conté ni text ni sub-elements es denota per una sola marca <*element/*>.

En els exemples següents, les paraules clau relacionades amb el *PNML* estan subratllades, mentre que les etiquetes que no estan subratllades son elements d'una determinada definició de tipus de xarxa de Petri, no del propi *PNML*. El primer exemple mostra la representació d'un lloc amb el identificador p1. El lloc té dos etiquetes; concretament són dues anotacions. La primera representa el nom del lloc <*name*>, mentre la segona representa el marcat inicial <*initialMarking*>. Una anotació consisteix en el seu valor <*value*> i, possiblement, alguna informació gràfica.

```
<place id="p1">
        <graphics>
                <position x="20" y="40"/>
        </graphics>
        <name>
                <value>ready to produce</value>
                <graphics>
                        <offset x="-10" y="10"/>
                </graphics>
        </name>
        <initialMarking>
                <value>P</value>
                <graphics>
                        <offset x="-1" y="-1"/>
                </graphics>
        </initialMarking>
</place>
```

El segon exemple mostra la representació d'una transició, similar a la representació del lloc anterior però aquesta conté informació d'una eina específica imaginària.

```
<transition id="t1">
        ...
        <toolspecific tool="PN4all" version="0.1">
                <hidden/>
```

```
        </toolspecific>
</transition>
```

Finalment s'inclou un exemple de la representació d'un arc: l'origen i el destí es mostren com a atributs *XML* del corresponent element *<arc>.* La representació gràfica de l'arc conté una llista de punts que representen els punts intermitjos de l'arc. Aquest arc té un atribut addicional *<type>*, que no forma part del *PNML* pròpiament dit, sinó de la definició de tipus de la xarxa corresponent.

```
<arc id="a1" source="p1" target="t1">
        <graphics>
                <position x="10" y="30"/>
                <position x="10" y="10"/>
        </graphics>
        <inscription>
                <value>x</value>
                <graphics>
                        <offset x="-6" y="-16"/>
                </graphics>
        </inscription>
        <type value="inhibitor"/>
</arc>
```

Una documentació més detallada sobre el *PNML* i el *PNTD* s'inclou en l'annex E.

7. Implementació del mòdul de programació

En l'apartat 5 es presenta el mòdul desenvolupat i les característiques del codi generat i en l'apartat 6 s'expliquen les eines necessàries per desenvolupar-lo. Seguidament s'explica com s'ha fet la implementació de tot això. Aquest capítol es recolza en l'annex B, on s'inclou el codi sencer de la implementació.

7.1. Breus nocions de l'estructura interna de PIPE

PIPE està separat en tres àrees de manera lògica: la interfície gràfica d'usuari (*GUI*), els mòduls d'anàlisi i una capa que gestiona les interaccions entre la *GUI* i els mòduls (*Figura 7.1*). Aquesta capa actua com una mena d'adaptador o traductor, dotant els mòduls d'un format estàndard de dades a on actuar. Així és com l'estructura de PIPE en termes de relació entre la *GUI* i la capa de dades es van implementar a través de l'arquitectura de Model-Vista-Controlador (*MVC*).

7.1.1. L'arquitectura de Model-Vista-Controlador

El patró de MVC és simple i efectiu, i permet separar l'aplicació en tres àrees diferents:

Model: És la part principal de l'aplicació i conté l'estat i les dades que està representant l'aplicació. Quan ocorren canvis significatius al model, aquest actualitza totes les vistes. En el PIPE el model és la capa de dades que s'anomena *datalayer*.

Controlador: És la interfície d'usuari presentada a l'usuari per a què manipuli l'aplicació. En PIPE es tracta d'un contenidor de la classe *Swing* de Java, que controla les xarxes de Petri de manera individual.

Vista: És la interfície d'usuari, que mostra informació sobre el model a l'usuari. Tot objecte que necessita informació del model necessita ser registrat al model. En PIPE, la vista és la representació gràfica d'una xarxa de Petri.

Aquest patró de disseny té molts avantatges, en particular la claredat del disseny, la modularitat, la extensibilitat, i el més important, la flexibilitat.

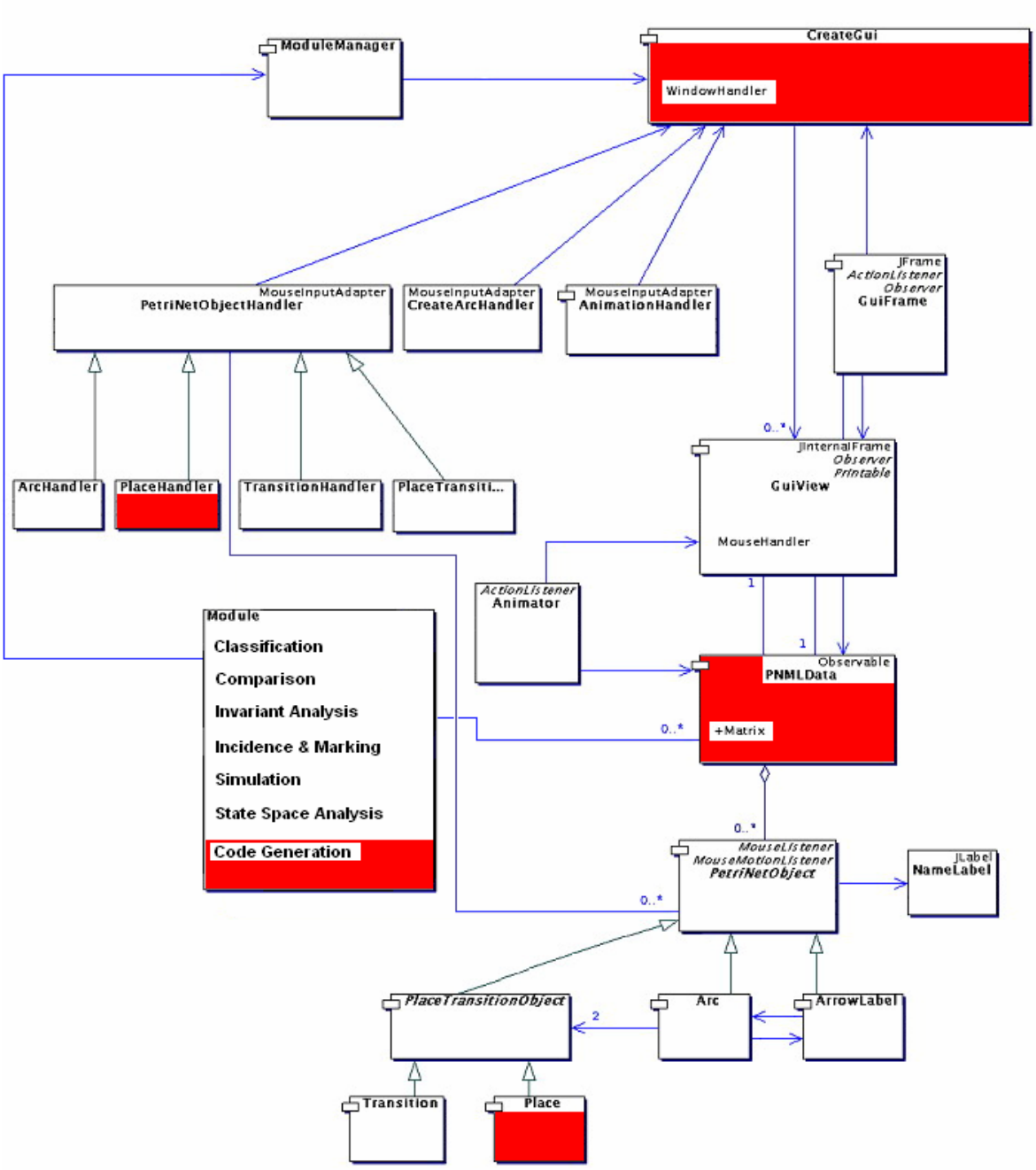


Figura 7.1 Esquema general de l'estructura interna de PIPE. Els elements creats o modificats en el present projecte es ressalten en vermell.

7.1.2. La capa de dades (datalayer)

La capa de dades està formada bàsicament per un paquet anomenat *PNMLData*. Aquest paquet conté totes les classes usades per representar una xarxa de Petri i té 4 rols: 1- guardar tota la informació sobre la xarxa de Petri i calcular la matriu d'incidència i el marcat,

2-guardar el model de xarxa de Petri de forma que faciliti la representació gràfica, 3-disparar les transicions i 4-guardar i obrir les xarxes de Petri des d'un fitxer.

Un model de xarxa de Petri està format per llocs, transicions, arcs, matriu de marcat i matriu d'incidència. Tan els llocs com les transicions tenen alguns atributs que caracteritzen les seves propietats, com el nom, la posició, etc. Aquests atributs es guarden en variables internes a cada objecte. L'ús de l'herència assegura que les variables comunes i els seus mètodes d'accés només es codifiquin una vegada (*Figura 7.2*).

Cada xarxa de Petri s'encapsula per una instància de la classe *PNMLData*. La classe *PNMLData* conté tots els objectes de xarxes de Petri guardats en *Arrays*, conté mètodes d'accés a tots els objectes interns i mètodes per calcular el marcat actual, el marcat inicial, les matrius de pre-incidència i post-incidència i les transicions habilitades.

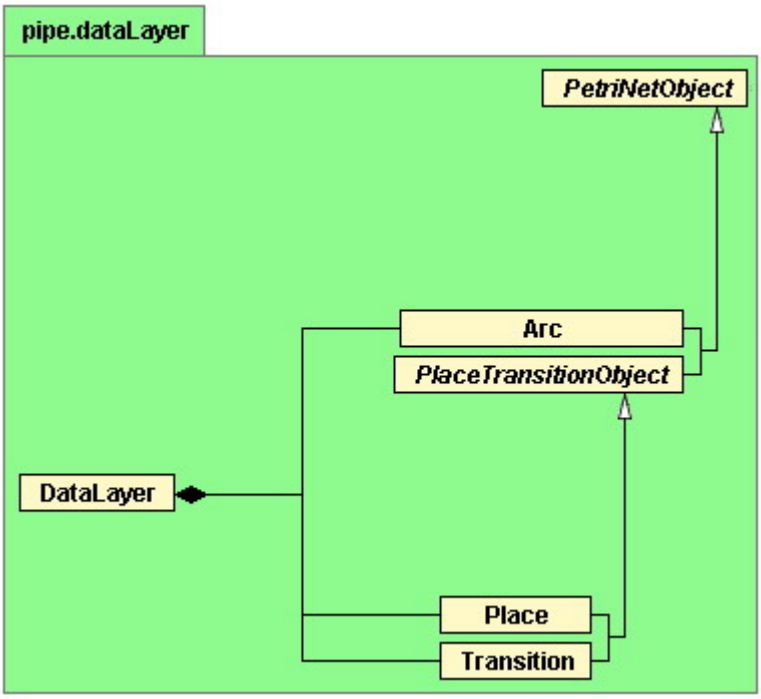


Figura 7.2 Herència dels elements usats per la capa de dades

La classe *PNMLData* també gestiona el dispar de les transicions; conté mètodes que permeten disparar tan una transició específica seleccionada com una transició aleatòria. El dispar aleatori s'aconsegueix a través del generador de números aleatoris de Java que s'usen per escollir aleatòriament una transició del conjunt de transicions habilitades. El càlcul del resultat de disparar una transició es fa a través de la matriu d'incidència calculant el vector de dispar, i la matriu del marcat actual.

Els canvis generats en el *PNMLData*, com per exemple afegir un nou objecte o disparar una transició, es mostren immediatament a la *GUI* mitjançant l'ús del patró de disseny *MVC*. El *PNMLData* s'utilitza com el model, guardant la informació de la representació gràfica. Quan el *PNMLData* pateix un canvi, el notifica a la classe *GuiView* que actualitza la representació visual de la xarxa de Petri.

El format dels fitxers per guardar i obrir xarxes de Petri és el *PNML*, vist en l'apartat 6.3. Per obrir una xarxa de Petri guardada en *PNML*, PIPE utilitza un fitxer *XSLT* (*Extensible Stylesheet Language Transformation)* per transformar-lo en un fitxer *XML* estàndard. Un cop feta la transformació, només ha d'extreure la informació del fitxer *XML* utilitzant alguna de les funcionalitats de Java dissenyades per a aquest fi. El procés per guardar una xarxa és l'invers: primer guarda la informació de la xarxa en un fitxer *XML* estàndard i després es transforma a *PNML* a través del corresponent *XSLT*. Els dos fitxers *XSLT* que fan aquesta transformació, tan els originals com els modificats es mostren a l'annex B.1.

7.1.3. La interfície gràfica d'usuari (GUI)

Tots els components de la *GUI* s'implementen a través dels components de Java *Swing*, prenent avantatge de les classes predissenyades i adaptant-les a les necessitats de PIPE. Cada element de les xarxes de Petri (llocs, arcs i transicions) i les seves respectives representacions en *Swing* estan pensades per a que s'ajustin a les moltes operacions que poden necessitar, com per exemple el pes dels arcs, afegir o treure marques als llocs, donar nom a tots els elements de la xarxa, etc. Així, els elements de la xarxa deriven del component *JLabel* de la biblioteca *Swing*.

La classe *GuiView* crea la representació visual a partir de les dades del model i suporta les interaccions amb l'usuari. *GuiView* es declara com un observador del *PNMLData* de manera que el *PNMLData* pot avisar automàticament la vista dels canvis soferts. Això és així gràcies a l'arquitectura *MVC*, que a més permet a diferents models i vistes de ser intercanviats fàcilment, cosa que és particularment útil per a tenir diferents finestres obertes amb diferents xarxes de Petri, que a la *GUI* s'implementen com a elements *JTabbedPane* i es visualitzen a través de la selecció de pestanyes a la part superior.

Per guardar i obrir fitxers a través de la interfície gràfica d'usuari es fa servir la classe *JFileChooser* del paquet *javax.swing*, que permet la creació de manera senzilla de les finestres de diàleg que serveixen per seleccionar directoris i fitxers i donar nom als fitxers a guardar.

7.1.4. Els mòduls d'anàlisi

El disseny general de l'aplicació imposa poques restriccions als dissenyadors per implementar nous mòduls d'anàlisi. Això vol dir que els canvis relacionats a adaptar codi existent per a usar-lo en el nou mòdul haurien de ser mínims. Tots els mòduls han d'implementar la interfície *Module*, que només conté dos mètodes:

```
public void run (PNMLData PetriNet) {…}
public String getName() {…}
```

A més d'això, els mòduls han de tenir un constructor sense paràmetres. Cada mòdul implementat d'aquesta manera es mostra com a un element més a l'arbre de mòduls de la *GUI*, situat a la part esquerra de la finestra. No hi ha més restriccions referents als mòduls, així que el dissenyador pot extreure la informació necessària de l'objecte *PNMLData* de la xarxa analitzada i és lliure de mostrar els resultats en qualsevol format.

7.2. Modificacions realitzades per implementar el mòdul de programació

Seguidament es comenten les modificacions fetes que serveixen per a crear posteriorment el mòdul de programació. Se separen en quatre apartats, les modificacions relacionades a les declaracions dels objectes que utilitza la capa de dades, les modificacions dels fitxers *XSLT* per tal que el format *PNML* en que es guarden les xarxes suporti les noves propietats, les modificacions a la capa de dades i finalment les modificacions a la interfície gràfica d'usuari.

7.2.1. Modificacions en les declaracions d'objectes

Cada element d'una xarxa de Petri està guardat en el corresponent objecte que després utilitza el *datalayer* per gestionar la informació. Cal pensar ara els elements addicionals que es necessiten per la generació de codi Val3 i si algun dels elements de que es disposen s'han de canviar.

Bàsicament es necessiten dues coses addicionals: definir un nou tipus per a cada lloc per poder distingir si pertany a una acció d'un robot o a un recurs compartit i que aquests llocs tinguin la capacitat d'emmagatzemar el codi que introduirà l'usuari. Aquests dos conceptes

addicionals estan relacionats només amb la definició de lloc, així és fàcil pensar que només s'han de definir els atributs addicionals al lloc i els mètodes per obtenir i guardar aquesta informació.

La classe *Place* és una extensió d'una classe comuna tan per llocs i transicions (*PlaceTransitionObject*) que conté tots els atributs i mètodes comuns a aquests dos elements. Per exemple, tan els llocs com les transicions tenen un nom, els mètodes per obtenir i guardar aquest nom, una posició x i y, els mètodes per obtenir i guardar aquestes posicions, tenen la capacitat de ser seleccionats, etc. Aquesta classe *PlaceTransitionObject* és a la vegada una extensió de la classe *PetriNetObject* que és una classe abstracta amb els components comuns a tots els elements d'una xarxa de Petri. Aquí és on l'herència té una gran rellevància i s'ha d'anar en compte de modificar un element que no afecti als seus fills de manera equivocada.

La classe *place* ja conté un atribut de tipus *string* que es diu "type" que hereta de la classe pare, així es defineix un nou atribut de tipus *string* que s'anomenarà "**subtype**". Els valors que podrà tenir aquest atribut es defineixen com a ***R1*** si el lloc pertany a una acció relacionada amb el primer robot, ***R2*** si el lloc pertany a una acció relacionada amb el segon robot i ***Shared Resource*** si el lloc es correspon amb un element o espai compartit. Per tal de poder diferenciar les branques de la xarxa que s'executaran en paral·lel a la branca principal (tasques en paral·lel a l'acció principal del robot que poden realitzar accions auxiliars, per exemple, càlculs independents) es defineixen dos tipus més, ***R1aux*** i ***R2aux***.

```
public String subtype="R1";
```

S'ha de tenir clar un concepte: Si l'usuari defineix una xarxa de Petri que no té a veure amb la generació de codi, aquests atributs no tenen sentit. Per tan, per defecte es defineixen tots els llocs amb el tipus R1 que té les mateixes propietats visuals que els llocs per defecte i si posteriorment l'usuari vol canviar el tipus només haurà de canviar el tipus dels llocs corresponents al robot 2 i als recursos compartits.

També s'han de definir els mètodes per obtenir i guardar aquest tipus:

```
public String getSubtype( ) {
        return this.subtype;
}
public void setSubtype(String subtype) {
        this.subtype = new String(subtype);
}
```

De la mateixa manera es defineix l'atribut que guardarà el codi, el *string* **codeSnipet** i els mètodes relacionats a ell:

```
private String codeSnipet = null;
public void setCodeSnipet(String codeSnipet) {
        this.codeSnipet = new String(codeSnipet);
}
public String getCodeSnipet( ) {
    if(this.codeSnipet != null) {
            return this.codeSnipet;}
     else{
            return("");}
}
```

El mètode per crear els llocs a partir de fitxers *PNML* també es veurà modificat, de manera que ara tindrà dos variables més, subtype i CodeSnipet:

```
public Place(double positionXInput, double positionYInput, String idInput, String nameInput, double
nameOffsetXInput, double nameOffsetYInput, int initialMarkingInput, double markingOffsetXInput, double
markingOffsetYInput, String subtypeInput, String codeSnipetInput ){
    ...
    subtype = new String(subtypeInput);
    codeSnipet = new String(codeSnipetInput);
    ...
}
```

Així mateix, el mètode per dibuixar i pintar els llocs a la interfície gràfica també canviarà. Es defineix que els llocs de tipus R1 seran de color negre, així si no es vol generar codi l'aplicació tindrà el mateix aspecte que abans de les modificacions. Els llocs de tipus R2 seran de color magenta i els llocs de tipus Shared Resource seran de color gris però amb la ratlla a traços. Les modificacions queden de la següent manera:

```
public void paintComponent(Graphics g) {
            super.paintComponent(g);
            ...
    if (this.getSubtype().equalsIgnoreCase("R1")|this.getSubtype().equalsIgnoreCase("R1aux"))
            g2.setStroke(new BasicStroke(1.0f));
    else if (this.getSubtype().equalsIgnoreCase("R2")|this.getSubtype().equalsIgnoreCase("R2aux"))
            g2.setStroke(new BasicStroke(1.0f));
    else
            g2.setStroke(new
            BasicStroke(1.0f,BasicStroke.CAP_SQUARE,BasicStroke.JOIN_MITER,10.0f,new float[]
```

```
                {6.0f,6.0f},0.0f));
        if (selected && !ignoreSelection)
                g2.setColor(SELECTION_FILL_COLOUR);
        else
                g2.setColor(ELEMENT_FILL_COLOUR);
                g2.fill(place);
                        if (selected && !ignoreSelection)
                                g2.setPaint(SELECTION_LINE_COLOUR);
                        else
                                if
                                (this.getSubtype().equalsIgnoreCase("R2")|this.getSubtype().equalsIgnoreC
                                ase("R2aux"))
                                        g2.setPaint(Color.MAGENTA);
                                else if
                                (this.getSubtype().equalsIgnoreCase("R1")|this.getSubtype().equalsIgnoreC
                                ase("R1aux"))
                                        g2.setPaint(ELEMENT_LINE_COLOUR);
                                else
                                        g2.setPaint(Color.GRAY);
                        g2.draw(place);
                        ...
        this.setToolTipText(this.getSubtype());
}
```

Com es pot observar, a part del color i tipus de línia per a cada tipus de lloc, a l'última línia de codi s'activa un *tooltip* que mostrarà el tipus del lloc. Aquests *tooltips* són etiquetes que s'activen quan es passa el ratolí per sobre dels elements, així serà més fàcil la seva identificació.

No es necessiten més modificacions en la declaració d'objectes ja que amb aquests dos atributs definits el mòdul en té suficient per generar el codi.

7.2.2. Modificacions del format PNML

Per a què els fitxers que contenen la informació de les xarxes de Petri en format *PNML* puguin guardar el valor d'aquests dos nous atributs es necessita ampliar la definició interna del format que el PIPE entén com a vàlid. En l'apartat 7.1.2 s'ha comentat la manera en que el programa PIPE entén el *PNML*, que és mitjançant la transformació a un format entenedor per Java a través de dos fitxers *XSLT*, un per guardar i un per obrir (annex B.1).

Per guardar una xarxa de Petri la capa de dades guarda de manera interna la informació de tota la xarxa en un fitxer *XML* estàndard a través de la biblioteca *javax.xml*, i seguidament

crida al fitxer *XSLT* anomenat *GeneratePNML* que es troba dins el directori on s'instal·la el PIPE. Aquest fitxer assigna a cada element guardat la seva transformació al format *PNML*. Es tracta doncs, d'afegir a la definició de l'element lloc els dos nous atributs, subtype i codeSnipet. Per evitar confusions es crea un fitxer nou anomenat *GeneratePNMLMod*.

```
 ...
<xsl:template match="net/place">
        <xsl:element name="place">
                <xsl:call-template name="place-transition" />
                <xsl:call-template name="initialMarking" />
                <xsl:call-template name="subtype" />
                <xsl:call-template name="codeSnipet" />
        </xsl:element>
</xsl:template>
 ...
```

A cada un d'aquests atributs també s'ha de definir els atributs que el formen. En els dos casos només tenen com a valor un camp que té el mateix nom que el propi atribut.

```
...
<xsl:template name="subtype">
        <xsl:element name="subtype">
                <xsl:element name="value">
                        <xsl:value-of select="@subtype" />
                </xsl:element>
         </xsl:element>
</xsl:template>
<xsl:template name="codeSnipet">
        <xsl:element name="codeSnipet">
                <xsl:element name="value">
                        <xsl:value-of select="@codeSnipet" />
                </xsl:element>
        </xsl:element>
</xsl:template>
...
```

De la mateixa manera, per obrir una xarxa guardada, la capa de dades crida al fitxer de la transformació inversa anterior anomenat *GenerateObjectList*. Aquest el que fa és transformar la informació del *PNML* al format que Java entén. Es crea un fitxer nou anomenat *GenerateObjectListMod* i s'afegeixen els dos nous atributs a la definició de l'element lloc.

```
...
<xsl:template match="place">
        <xsl:element name="place">
                <xsl:call-template name="place-transition" />
                  ...
                  <xsl:attribute name="subtype">
                          <xsl:value-of select="subtype/value" />
                  </xsl:attribute>
                  <xsl:attribute name="codeSnipet">
                          <xsl:value-of select="codeSnipet/value" />
                  </xsl:attribute>
                  ...
        </xsl:element>
 </xsl:template>
...
```

7.2.3. Modificacions a la capa de dades

Com s'ha comentat la capa de dades és el nucli de PIPE i la part que guarda i gestiona tota la informació relacionada amb les xarxes. Un cop modificats els fitxers anteriors s'han d'implementar certes modificacions als mètodes que serveixen per guardar i obrir les xarxes i que utilitzen els fitxers anteriors.

Per guardar una xarxa es fa servir el mètode *savePNML*. Aquest mètode primer crea el document intern *XML* amb la informació de la xarxa fent un escombrat als llocs, transicions i arcs que la formen i seguidament fa la crida al fitxer que fa la transformació. Es canvia aquest fitxer per el *GeneratePNMLMod* creat anteriorment.

```
public void savePNML(File file) {
        ...
        for(i = 0 ; i < places.length ; i++) {
                NET.appendChild(createPlaceElement(places[i], pnDOM));
        }
        places = null;
        ...
         xsltSource = new StreamSource(CreateGui.appPath+"xslt"+
        System.getProperty("file.separator")+"GeneratePNMLMod.xsl");
         ...
}
```

Com es veu, l'escombrat que guarda la informació dels llocs fa una crida al mètode *createPlaceElement*. S'afegeixen els dos camps necessaris a dins aquest mètode.

```
private Element createPlaceElement(Place inputPlace, Document document){
        Element placeElement = null;
        ...
        if(inputPlace != null ) {
        ...
        String subtypeInput = inputPlace.getSubtype();
        String codeSnipetInput = inputPlace.getCodeSnipet();
        ...
        placeElement.setAttribute("subtype", (subtypeInput != null ? String.valueOf(subtypeInput) : ""));
        placeElement.setAttribute("codeSnipet", (codeSnipetInput != null ? String.valueOf( codeSnipetInput) :
        ""));}
        return placeElement;
}
```

Per obrir una xarxa de Petri es fa servir el mètode *loadPNML*. Aquest mètode fa el procés invers al de guardar, és a dir, primer crida el fitxer de la transformació corresponent i després guardar la informació en el format que la capa de dades ja pot gestionar.

```
public void loadPNML(String filename) {
        ...
        StreamSource xsltSource = new
        StreamSource(CreateGui.appPath+"xslt"+System.getProperty("file.separator")+"GenerateObjectListMod.x
        sl");
        ...
        for(int i = 0 ; i < nodeList.getLength() ; i++) {
        ...
                if("place".equals(element.getNodeName())){
                        addPlace(createPlace(element));
        }
        ...
    }
}
```

D'igual manera que en el cas de guardar, quan es fa l'escombrat que crea els llocs es fa una crida al mètode *createPlace* que també s'ha de modificar.

```
private Place createPlace(Element inputPlaceElement){
        String subtypeInput = null;
        String codeSnipetInput = null;
        ...
        String subtypeTempStorage = inputPlaceElement.getAttribute("subtype");
        String codeSnipetTempStorage = inputPlaceElement.getAttribute("codeSnipet");
```

```
            ...
            if (subtypeTempStorage.length() > 0)
                        subtypeInput = subtypeTempStorage;
            else
                        subtypeInput="R1";
            if (codeSnipetTempStorage.length() > 0)
                        codeSnipetInput = codeSnipetTempStorage;
             else
                        codeSnipetInput = "";
            ...
            return new Place(positionXInput, positionYInput, idInput, nameInput, nameOffsetXInput,
            nameOffsetYInput, initialMarkingInput, markingOffsetXInput, markingOffsetYInput, subtypeInput,
            codeSnipetInput);
}
```

7.2.4. Modificacions a la interfície gràfica d'usuari

Totes les modificacions anteriors queden reflectides finalment en la interfície gràfica. S'ha de dissenyar una manera en que l'usuari pugui guardar i modificar tan el tipus com el codi relacionat als llocs. En aquest apartat només es mostren les modificacions fetes a la interfície gràfica que no es corresponen al nou mòdul, ja que aquestes es veuran en el següent apartat.

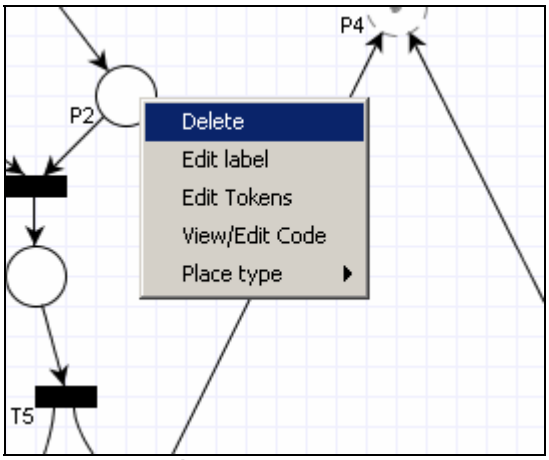
Primerament es decideix que la manera que tindrà l'usuari per escollir el tipus dels llocs serà a través del menú desplegable tipus *pop-up* que ja disposen els elements lloc, en el que s'hauran d'afegir més opcions. Per visualitzar i editar el codi també es farà a través d'aquest menú desplegable, a més de poder-ho fer a través de la finestra del mòdul.

La classe usada per implementar els mètodes que es corresponen a accions del ratolí sobre els llocs s'anomena *PlaceHandler*. Aquesta classe és la que crea el menú desplegable de tipus *JPopupMenu* de la biblioteca *swing* de Java i conté tres mètodes, el constructor de la classe, el que crea el menú i el que li diu al lloc si ha d'afegir o no una marca si l'opció d'afegir marques està seleccionada en el moment de prémer el botó del ratolí sobre el lloc. Es modifica només el segon mètode afegint les opcions necessàries.

```
public class PlaceHandler extends PlaceTransitionObjectHandler {
            public PlaceHandler(Container contentpane, Place obj) {...}
            public JPopupMenu getPopup(MouseEvent e) {
                        ...
                        menuItem= new JMenuItem(new EditCodeAction(contentPane,(Place)myObject));
                        menuItem.setText("View/Edit Code");
                        popup.add(menuItem);
```

```
                JMenu subtypeMenu=new JMenu("Place type");
                subtypeMenu.add(new JMenuItem(new EditSubtypeAction(contentPane,
                (Place)myObject,"R1")));
                subtypeMenu.add(new JMenuItem(new
                EditSubtypeAction(contentPane,(Place)myObject,"R2")));
                subtypeMenu.add(new JMenuItem(new EditSubtypeAction(contentPane, (Place)myObject,
                "R1aux")));
                subtypeMenu.add(new JMenuItem(new EditSubtypeAction(contentPane, (Place)myObject,
                "R2aux")));
                subtypeMenu.add(new JMenuItem(new EditSubtypeAction(contentPane, (Place)myObject,
                "Shared Resource")));
                popup.add(subtypeMenu);
                return popup;
                }
        public void mouseClicked(MouseEvent e) {...}
}
```

Es creen dos elements més del menú, que s'anomenen *View/Edit Code* i *Place type*. El primer crida el mètode *EditCodeAction* i el segon crea un submenú on es pot escollir el tipus del lloc. Cada opció d'aquest submenú crida el mètode *EditSubtypeAction*.



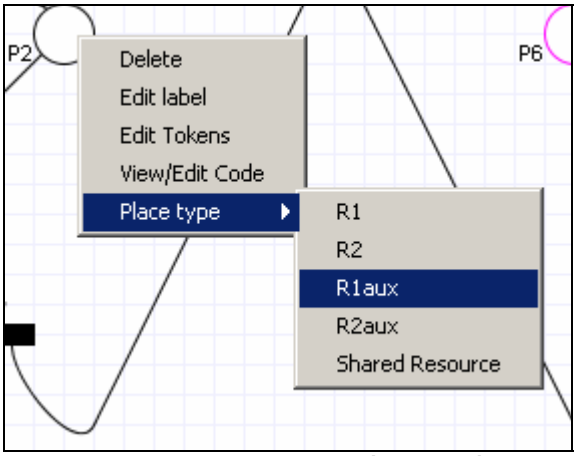


Figura 7.3 Menú desplegable dels llocs amb els nous camps. Figura 7.4 Submenú d'elecció de tipus

Els dos mètodes que criden aquests menús són nous però es programen seguint el mateix patró que els altres elements del menú, com per exemple *EditTokenAction*. En el cas de *EditSubtypeAction* es tracta d'una classe on el constructor de la classe assigna els paràmetres que se li passen (el contenidor, el lloc seleccionat i el tipus seleccionat) i té un altre mètode que simplement assigna al lloc seleccionat el tipus seleccionat, mitjançant el mètode definit a l'objecte lloc per a tal finalitat.

```
public class EditSubtypeAction extends AbstractAction {
        ...
        public EditSubtypeAction (Container contentPane, Place p, String name) {...}
```

```
        public void actionPerformed(ActionEvent e){
                selected.setSubtype(Subtype);
         }
}
```

El cas de *EditCodeAction* és similar a l'anterior, però el mètode que executa és una mica més complex ja que aquest el que fa és cridar una finestra que mostrarà el codi. Aquesta finestra també s'ha de dissenyar de zero. Es crea primer aquesta finestra, que es tracta d'una classe nova anomenada *ObjectCodePane* que deriva del contenidor *JPanel* de la biblioteca gràfica *swing* de Java i conté un objecte *JEditorPane* que simplement és una finestra de text editable que té dos mètodes, un per assignar el text que mostrarà i un altre per agafar-lo.

```
public class ObjectCodePane extends JPanel {
        JEditorPane results;
        Clipboard clipboard=this.getToolkit().getSystemClipboard();
        public ObjectCodePane() {
                super(new BorderLayout());
                results=new JEditorPane();
                results.setEditable(true);
                results.setMargin(new Insets(5,5,5,5));
                results.setContentType("text/plain");
                JScrollPane scroller=new JScrollPane(results);
                scroller.setPreferredSize(new Dimension(400,300));
                scroller.setBorder(new BevelBorder(BevelBorder.LOWERED));
                this.add(scroller);
                this.setBorder(new TitledBorder(new EtchedBorder(),"Codi"));}
        public void setText(String text) {
                results.setText(text); }
        public String getText(){
                return results.getText();}
}
```

Ara ja es pot definir la classe *EditCodeAction* de la següent manera:

```
public class EditCodeAction extends AbstractAction {
        ...
          public EditCodeAction (Container Pane, Place p) {...}
          public void actionPerformed(ActionEvent e){
                String codi=selected.getCodeSnipet();
                guiDialog = new JDialog(CreateGui.getApp(),"Code For Place "+ selected.getName(), ,true);
                Container Pane=guiDialog.getContentPane();
```

```
                Pane.setLayout(new BoxLayout(Pane,BoxLayout.PAGE_AXIS));
                Pane.add(resultats=new ObjectCodePane());
                resultats.setText(codi);
                Pane.add(new ButtonBar("Save",SaveCodeButtonClick),BorderLayout.PAGE_END);
                ...}
        ActionListener SaveCodeButtonClick=new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                            selected.setCodeSnipet(resultats.getText());
                           guiDialog.setVisible(false);}
        };}
```

D'aquesta classe es pot destacar que fa una crida a un objecte de la classe *ObjectCodePane* i l'encapsula dins una finestra tipus *JDialog* de la biblioteca *swing*. Es mostra el codi del lloc que se li passa i s'afegeix un botó anomenat *save*, que assigna el text mostrat en pantalla a la variable codeSnipet del lloc. La *Figura 7.5* mostra aquesta finestra.

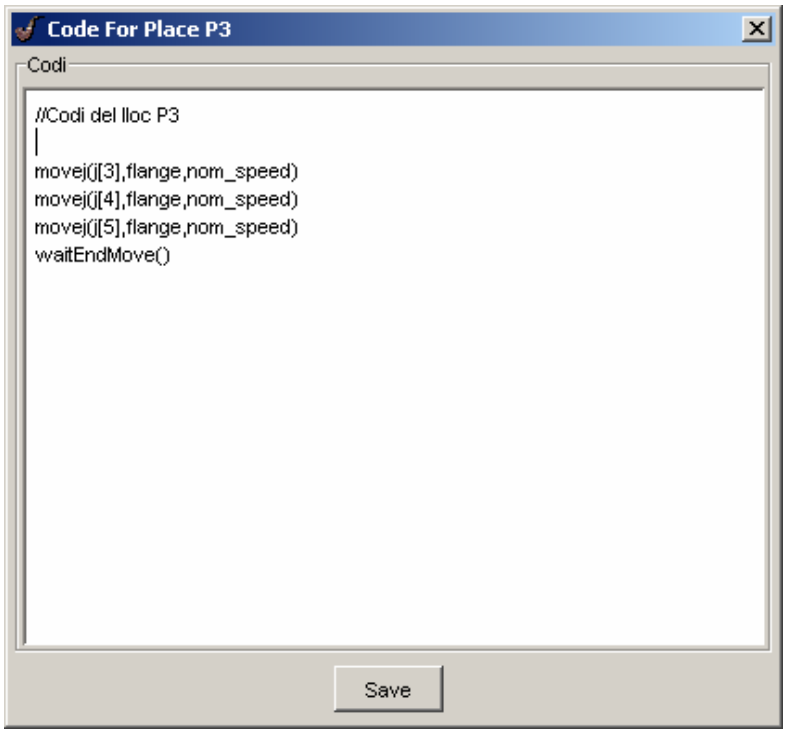


Figura 7.5 Finestra de creació i edició de codi d'un lloc anomenat P3

7.3. La implementació

D'acord amb l'apartat 7.1.4, per programar el mòdul s'ha d'implementar la interfície *Module* amb els seus dos mètodes *run* i *getName*. La classe que implementarà el mòdul s'anomena *Val3* (annex B.2) i conté cinc objectes a part dels dos mètodes anteriors:

- Una variable de tipus *string* que conté el nom del mòdul (MODULE_NAME).
- Un objecte que serà el selector de fitxers (*sourceFilePanel*).
- Dos objectes iguals, un per a cada robot que són les finestres que contenen el codi (results1 i results2).
- Una finestra, que és la que es mostra quan es vol editar el codi dels llocs des de dins del mòdul.

El mètode *getName* simplement retorna la variable nom de la classe *Val3*, mentre que el mètode *run* crea la interfície gràfica amb els cinc objectes mencionats tot afegint els botons pertinents i les accions a realitzar quan es premen.

```
public void run(DataLayer pnmlData) {
        JDialog guiDialog = new JDialog(CreateGui.getApp(),MODULE_NAME,true);
        Container contentPane=guiDialog.getContentPane();
        contentPane.setLayout(new BoxLayout(contentPane,BoxLayout.PAGE_AXIS));
        contentPane.add(sourceFilePanel=new PetriNetChooserPanel("Source Net",pnmlData));
        JPanel Codi=new JPanel();
        Codi.setLayout(new BoxLayout(Codi,BoxLayout.LINE_AXIS));
        Codi.add(results1=new GeneradorCodiPane());
        Results1.setBorder(new TitledBorder(new EtchedBorder(),"Code for R1"));
        Codi.add(results2=new GeneradorCodiPane());
        Results2.setBorder(new TitledBorder(new EtchedBorder(),"Code for R2"));
        contentPane.add(Codi);
        contentPane.add(new ButtonBar(new String[]{"Generate Code","Places","Save Project"},new
        ActionListener[]{Val3ButtonClick,CodePlaceButtonClick,MakeProjectButtonClick}));
        guiDialog.pack();
        guiDialog.setLocationRelativeTo(null);
        guiDialog.setVisible(true);
 }
```

La part principal del mòdul serà la programació dels tres *ActionListener* que implementen les accions que executen els tres botons d'aquesta finestra: el generador de codi, l'editor del codi dels llocs i el generador del projecte Val3 (com es veu en la *Figura 5.1)*.

7.3.1. Implementació de la generació de codi

Per poder generar el codi es crea un conjunt de mètodes que es poden separar en dos grups: els mètodes que serveixen per a fer escombrats a la xarxa i saber les característiques de llocs i transicions i els mètodes que tenen en comú que tots retornen una variable de tipus

string, ja que les funcions específiques d'aquest mòdul fan que tots els resultats siguin en format de text, ja sigui el text que es mostrarà en pantalla com el text per generar els fitxers del projecte en Val3.

Mètodes d'anàlisi de la xarxa

Els mètodes d'aquest primer grup es basen en les següents dues funcions:

```
private int[] ForwardsPlaceSet(DataLayer pnmlData, int TransitionNo) {...}
private int[] BackwardsPlaceSet(DataLayer pnmlData, int TransitionNo) {...}
```

S'observa que tenen dos paràmetres, un element del tipus *DataLayer* que representa la xarxa d'estudi i un una variable de tipus *int* que representa l'índex d'una transició en la representació interna. Es tracta de dues funcions que retornen un vector de números enters que indiquen l'índex dels llocs en la representació interna de la xarxa d'estudi: el primer mètode retorna els llocs als que apunten tots els arcs que surten de la transició *TransitionNo*, i el segon retorna els llocs als que apunten tots els arcs que entren a aquesta transició.

Amb aquests dos mètodes es poden definir totes les funcions següents que implementen comprovacions que es fan sobre les transicions per poder fer un escombrat correcte dels elements de la xarxa; tenen en comú que tots retornen un valor booleà.

```
private boolean transitionR1(DataLayer pnmlData, int TransitionNo){...}
private boolean transitionR1aux(DataLayer pnmlData, int TransitionNo){...}
private boolean transitionR1Only(DataLayer pnmlData, int TransitionNo){...}
private boolean transitionR2(DataLayer pnmlData, int TransitionNo){...}
private boolean transitionR2aux(DataLayer pnmlData, int TransitionNo){...}
private boolean transitionR2Only(DataLayer pnmlData, int TransitionNo){...}
```

La primera funció retorna *true* si la transició *TransitionNo* donada no té cap lloc ni al conjunt de llocs d'entrada ni al conjunt de llocs de sortida que pertanyi a R2, és a dir, la branca que conté *TransitionNo* pertany al robot R1. La segona funció retorna *true* si la transició donada només té llocs de tipus *R1aux* tan a l'entrada com a la sortida, és a dir, la branca que conté *TransitionNo* pertany al robot R1 però no és una branca del fil principal d'execució del robot. I la tercera funció retorna *true* si la transició donada només te llocs de tipus R1 i es diferencia de la primera funció en que no té llocs de tipus *R1aux*, així la transició donada només formarà part del fil principal d'execució del robot R1. Les altres tres funcions són exactament iguals que les anteriors però fent les comprovacions corresponents al robot R2.

També es defineixen les dues funcions booleanes següents:

```
private boolean transitionRC(DataLayer pnmlData, int TransitionNo){...}
private boolean NexTransAux(DataLayer pnmlData, int PlaceNo){...}
```

on la primera retorna *true* si la transició conté un lloc de tipus recurs compartit en el conjunt de llocs d'entrada i la segona retorna *true* si la següent transició del lloc donat forma part d'una branca auxiliar. Aquesta funció implica una restricció del mòdul: les branques auxiliars dels dos robots no poden tenir més d'una branca en paral·lel al fil principal a la vegada.

Mètodes que obtenen el codi resultant

Formen part d'aquest segon grup tots els mètodes que serveixen per obtenir el codi i per tan, tenen en comú entre sí que retornen una cadena de caràcters que representa el tros de codi generat per aquell mètode. Totes aquestes funcions segueixen el mateix patró: a partir de les funcions del grup anterior es van fent escombrats concrets per obtenir les característiques de la xarxa i saber en cada moment a quin robot pertanyen les transicions o llocs i quin codi ha de generar.

Per fer més entenedor el procés de generació de codi i les parts en que es dividirà el codi generat, es crea un mètode principal (un per cada robot) on se separen molt clarament les parts del codi generat i es comenta l'inici de cada part (els comentaris són les línies que comencen per doble barra):

```
private String CompileCodeR1(DataLayer pnmlData){
  Place[] places=pnmlData.getPlaces();
  String j=new String();
  int PlaceNo=0;
  int Procedencia=1;
        boolean trobat=false;
        boolean RC=false;
        //trobat es fals si no hi ha llocs de R1
        if (places!=null)
                while ((PlaceNo<places.length) && (!trobat)){
                        if (places[PlaceNo].getSubtype().equalsIgnoreCase("R1")){
                        trobat=true;}
                        PlaceNo+=1;
                }
        if (trobat){
                j="//Generador de Codi Val3 del robot R1"+"\n\n"+"//programa start() de R1"+"\n\n";
                //1-crea les inicialitzacions
                j+="begin"+"\n";
                j+=InitializeCodeR1(pnmlData);
```

```
                j+="\n";
                //2-crea les crides a les tasques de gestió de recursos compartits
                j+=RcTaskCreateR1(pnmlData);
                //3-crea el llaç principal
                j+=PrincipalCodeR1(pnmlData,Procedencia);
                j+="end"+"\n"+"\n";
                //4-crea tots els subprogrames de transició i lloc
                j+=TransitionCodeR1(pnmlData);
                //Mira si hi ha recursos compartits i posa el valor a RC
                PlaceNo=0;
                while ((PlaceNo<places.length) && (!RC)){
                        if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                RC=true;}
                        PlaceNo+=1;
                }
                if (RC){
                        //5-crea el subprograma  de comunicació
                        j+="//Programa auxiliar comunicacio()"+"\n"+"begin"+"\n";
                        j+=comunicacio();
                        j+="\n";
                        //6-crea el subprograma de gestió de recursos compartits
                        j+="//Programa gestio dels recursos compartits gestioRC()"+"\n";
                        j+=gestioRC(pnmlData);
                        j+="\n";
                        //7-crea el subprograma d'escriptura
                        j+="//Programa escriptura del buffer_out escriptura()"+"\n";
                        j+=escriptura(pnmlData);
                        j+="\n";
                        //8-crea el subprograma de lectura
                        j+="//Programa lectura del buffer_in lectura()"+"\n";
                        j+=lectura();
                        j+="\n";
                        //9-crea el subprograma d'exclusió mutua
                        j+="//Programa d'exlusió mutua mutex(bool& bRC)"+"\n";
                        j+=mutex();
                }
        }else{
                j="No hi ha llocs associats a R1";}
  return(j);
}
```

Tots els comentaris següents faran referència a aquesta funció. El primer que fa és comprovar que existeixin llocs a la xarxa. Després mira lloc per lloc si n'hi ha algun de tipus R1, ja que si no fos així no tindria sentit continuar, i seguidament comença la generació de codi. Aquest codi tindrà un programa principal (*start*) i molts subprogrames depenent dels elements de la xarxa (llocs, transicions, recursos compartits, etc). Com es veu en els comentaris, el codi generat se separa en nou parts i cada part crida a una funció de text, seguint la llista de programes explicats a l'apartat 5.2.1. El codi de tots aquests mètodes es troba a l'annex B.2 i s'omet ara d'exposar-lo sencer, però sí que se'n comenta el funcionament general.

La primera part de codi a generar són les inicialitzacions de les variables que es fa dins del programa *start*. Mitjançant el mètode:

```
private String InitializeCodeR1 (DataLayer pnmlData){...}
```

s'inicialitzen les variables ntokens amb el valor del marcat inicial i en cas d'haver-hi recursos compartits, les variables RC s'inicialitzen a 0 si el lloc està marcat perquè voldrà dir que el recurs està lliure. També s'inicialitzen les variables booleanes bRC a fals i un vector Pin[ ] (Pin[ ] i Pout[ ] en el cas de R2).

Seguidament, en cas que la xarxa tingui llocs de tipus recurs compartit, es fa la crida a les tasques per la seva gestió.

```
private String RcTaskCreateR1(DataLayer pnmlData){...}
```

La comunicació entre els robots es fa a través de les dues variables de tipus *sio* anomenades buffer_in i buffer_out. Així en el robot R1 primer es vincula l'entrada/soritda anomenada *Server* a buffer_in, mentre que en el robot R2 es vincula l'entrada/sortida anomenada *Client* a buffer_out. La següent línia de codi a generar és la crida a la tasca que permetrà la comunicació mitjançant la instrucció *taskCreate*. En els dos robots es crea una tasca que executa el programa comunicacio() corresponent a cada robot.

Un cop arribat en aquest punt, es genera el llaç principal del programa *start()*. Aquest llaç generat a través del següent mètode:

```
private String PrincipalCodeR1 (DataLayer pnmlData,int Procedencia){...}
```

implementa totes les parts de codi com les mencionades en l'apartat 5.2.1 que fan referència a la dinàmica dels robots. Primerament fa un recorregut per totes les transicions buscant les que formin part de R1 sense tenir llocs auxiliars i que tinguin un recurs compartit a l'entrada per tal de donar preferència a les accions que utilitzin recursos compartits, i després busca

les que formen part de R1 sense llocs auxiliars ni recursos compartits. La variable Procedencia s'utilitza per saber quin mètode ha cridat aquesta funció, ja que en el cas de generar un projecte en Val3 es faran servir aquests mateixos mètodes però amb alguna modificació, com per exemple els signe de "més petit que", que té una representació diferent en el codi intern de Val3 *(&lt*) al propi signe *"<".* En cada cas es crida un mètode diferent, però que compleix el mateix esquema general:

```
private String CreateTransitionRCCode(DataLayer pnmlData,int TransitionNo,int Procedencia){...}
private String CreateTransitionCode(DataLayer pnmlData,int TransitionNo,int Procedencia){...}
```

diferenciant-se entre sí en els elements extra que s'han d'afegir per a la gestió dels recursos compartits. Aquests dos mètodes executen successius llaços a través de la transició *TransitionNo* que reben i generen pas per pas les parts de codi necessàries.

Després d'executar aquests mètodes s'acaba el programa *Start()* i el següent codi a generar són els programes de transicions i de llocs mitjançant la funció *TransitionCodeR1()*, que crea dos llaços i encapsula les funcions que realment generen el codi:

```
private String ProgramCodeTran(DataLayer pnmlData,int TransitionNo){...}
private String ProgramCodePlace(DataLayer pnmlData,int TransitionNo){...}
```

El primer mètode obté els llocs d'entrada i de sortida de la transició *TransitionNo* donada. Per als llocs d'entrada escriu el codi que actualitza el marcat i per als llocs de sortida escriu el codi que fa la crida als programes de lloc, actualitzant seguidament el marcat afegint-los les marques adients. També fa les crides a les tasques auxiliars de les branques que contenen llocs de tipus *R1aux* (*R2aux* en el cas del robot R2).

El segon mètode comprova que els llocs de sortida de la transició donada són de tipus diferent a recurs compartit. En cas afirmatiu, crea els programes de cada lloc, escrivint simplement el codi intern del lloc que està guardat a la variable codeSnipet definida a l'apartat 7.2.1.

Un cop generats tots aquests programes necessaris per a qualsevol xarxa modelada, es mira l'existència o no de recursos compartits i en cas afirmatiu, es criden seqüencialment les funcions que implementen els programes de comunicació, gestió, escriptura, lectura i d'exclusió mútua:

```
private String comunicacio(){...}
private String lectura(){...}
private String escriptura(DataLayer pnmlData){...}
private String gestioRC(DataLayer pnmlData){...}
private String mutex(){...}
```

Els mètodes que implementen la comunicació, la lectura i l'exclusió mútua no depenen de la xarxa de Petri, només de l'existència dels recursos compartits. Per tan, cap d'aquests tres mètodes té paràmetres ja que assegurada l'existència dels recursos compartits el codi generat sempre serà el mateix. En el cas dels programes d'escriptura i gestió, al tenir la mateixa part de codi repetit per a cada recurs compartit, necessita rebre la xarxa com a paràmetre per poder fer els recorreguts per tots els elements en busca dels recursos compartits.

7.3.2. Implementació de l'editor de codi dels llocs

Aquest apartat descriu la implementació de les accions que realitza el segon botó del mòdul: l'editor del codi que guarda internament cada lloc. Es tracta d'elements que només implementen la interfície gràfica per editar i guardar el codi dels llocs (guardat a la variable codeSnipet) des de la finestra del mòdul de programació ja que la classe *place* ja disposa de les modificacions adients.

Com es menciona a l'inici de l'apartat 7.3, la classe *Val3* que implementa el mòdul té una variable de classe que és la finestra que mostra el codi dels llocs. Quan es prem sobre el botó d'edició del codi dels llocs primerament es comprova que existeixi una xarxa a l'editor ja que si no fos així no tindria sentit editar el codi d'uns llocs que no existeixen com a xarxa. Un cop feta aquesta comprovació es crea un nou objecte de la variable de classe *SelectCodePane* mencionada i que es diu codepanel. Per tan, aquest botó no fa res més que crear aquest objecte; fins ara però, encara no ha estat implementat. Cal doncs fer la implementació de la nova classe *SelectCodePane*.

Aquesta finestra serà molt semblant a la que es fa servir des de l'editor per modificar el codi (*Figura 7.5*), amb la particularitat que en aquell cas s'accedia al codi del lloc a través d'un menú desplegable del propi lloc; des del mòdul de programació però, no tenim accés directe a l'editor, per tan s'ha de modificar la finestra que s'obrirà per tal que es pugui escollir el lloc a editar des de dins la pròpia finestra. La nova classe serà:

```
public class SelectCodePane extends JPanel implements ActionListener {
        private ObjectCodePane resultspane
        ...
        public SelectCodePane(DataLayer pnmlData) {
                places=pnmlData.getPlaces();
                String[] placeStrings=new String[places.length];
                if (places!=null)
                        for (int placeNo=0; placeNo<places.length; placeNo++){
                                placeStrings[placeNo]=places[placeNo].getName();}
```

```
                    JComboBox placelist = new JComboBox(placeStrings);
                    placetype=new JLabel("  Type: "+selected.getSubtype()+"  ",JLabel.RIGHT);
                    placelist.setSelectedIndex(0);
                    placelist.addActionListener(this);
                    guiDialog = new JDialog(CreateGui.getApp(),"Code For All Places",true);
                    ...
}
```

Com es veu, té un objecte de tipus *ObjectCodePane* com el definit en l'apartat 7.2.4 que suportarà el codi dels llocs i el mostrarà en la finestra. Per tal de possibilitar la selecció de llocs des de la pròpia finestra, es tria implementar un objecte de la classe *JComboBox* de la biblioteca gràfica *Swing* de Java. Aquesta classe presenta un sistema ideal per a la selecció d'elements mostrats en forma de llista desplegable. Així, es crearà aquest desplegable a sota la finestra de codi i s'afegirà un camp de text al costat d'aquest menú que mostrarà el tipus del lloc seleccionat, per tal de no confondre's de robot; aquest camp serà del tipus *JLabel*. Es crearà un mètode dins la classe *SelectCodePane* que actualitzarà el seu valor amb el tipus del lloc seleccionat al menú i a la vegada actualitzarà el codi a mostrar en pantalla. Finalment tindrà un botó per guardar el codi. La seva implementació queda determinada de la següent manera:

```
public void updatePanel(String name,Place[] places) {
        if (places!=null)
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        if(places[placeNo].getName().equalsIgnoreCase(name))
                                selected=places[placeNo];
                                resultspane.setText(selected.getCodeSnipet());
                                placetype.setText("  Type: "+selected.getSubtype()+"  ");}
}
```

Aquesta finestra amb tots els seus camps queda representada amb la *Figura 7.6*.

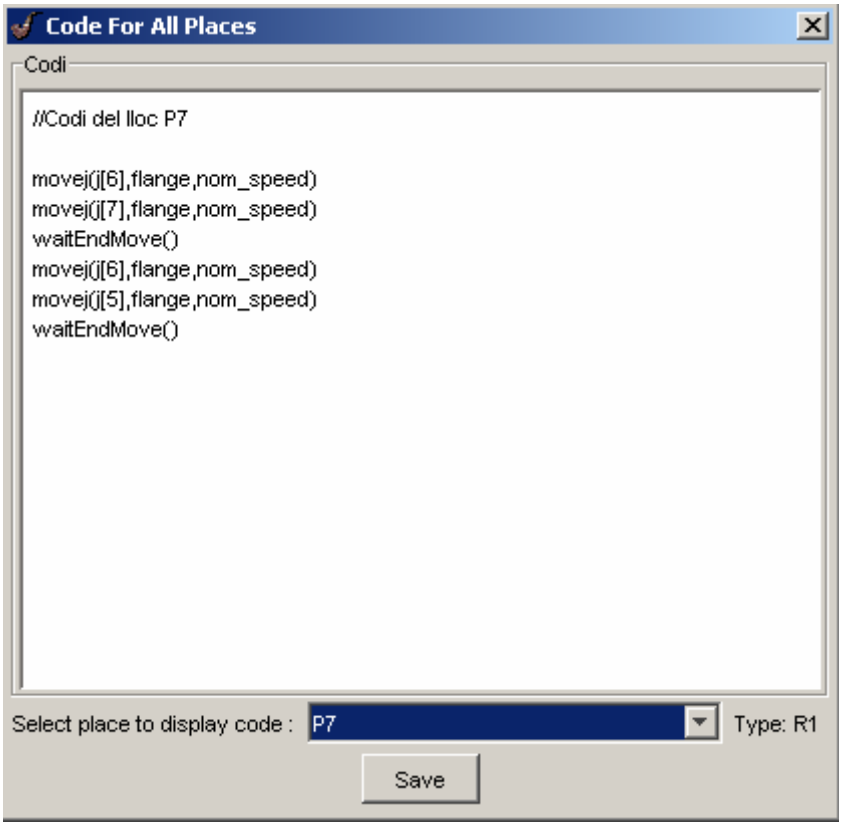


Figura 7.6 Finestra de creació i edició cridada des del mòdul de programació

7.3.3. Implementació de la generació de projectes

En aquest darrer apartat es descriu la codificació de les accions que realitza el tercer botó del mòdul de programació: es tracta de la generació dels fitxers que formen part dels projectes en format entenedor de *Val3 Studio* tal com s'ha mencionat en l'apartat 4.3.2.

La implementació d'aquesta part resulta molt facilitada per la reutilització dels mètodes desenvolupats per generar el codi. Es tracta doncs, de crear només els mètodes que afegeixen les etiquetes *XML* a cada fitxer i els que permeten la selecció del directori on es guardaran els fitxers en forma de projecte, degut a que ja es disposen dels mètodes per generar el codi.

El primer que s'ha de mostrar quan es prem el botó és la finestra que actuï com a selector del directori a guardar els fitxers. Aquesta finestra s'implementa a través de la classe *FileBrowser* que ja està definida en el codi intern de PIPE i que és una extensió de la classe *JFileChooser* de la biblioteca *Swing* de Java que permet guardar fitxers.

```
FileBrowser fileBrowser=new FileBrowser("Directory","",null);
fileBrowser.setFileSelectionMode(FileBrowser.DIRECTORIES_ONLY);
```

```
fileBrowser.showDialog(fileBrowser, "Save in this folder");
```

Aquest és la part de codi que mostrarà el selector de fitxers (*Figura 7.7*), que com es veu a la segona línia s'imposa la restricció que mostri només directoris, ja que el nom dels fitxers generats estaran definits automàticament.

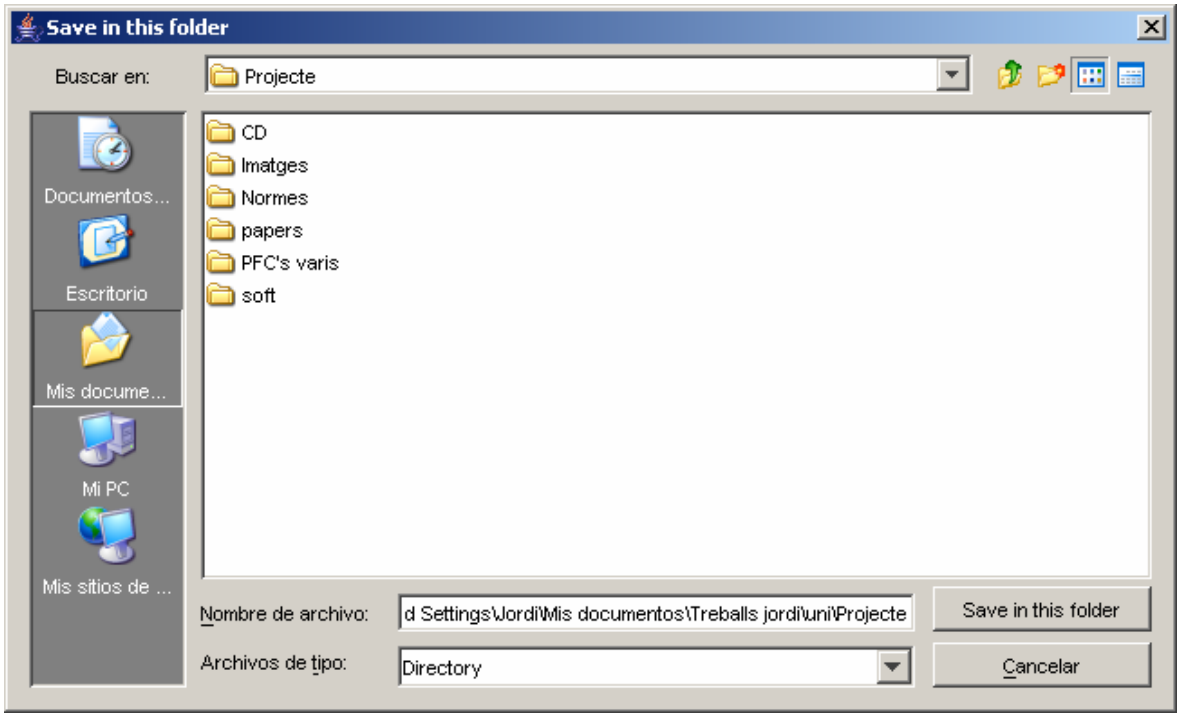


Figura 7.7 Finestra de selecció del directori on es guarden els projectes en format Val3

S'han de crear dos projectes, un per a cada robot i tindran per defecte els noms de **Robot_R1** i **Robot_R2**. Per a cada projecte es tindrà el codi següent:

```
String destFN=fileBrowser.getSelectedFile().getPath();
File directory = new File(destFN+"/"+"Robot_R1");
directory.mkdir();
destFN+="/"+"Robot_R1/";
```

Que crea el directori i es guarda la ruta on s'hauran de guardar els fitxers de cada projecte. A partir d'aquí es segueix el mateix patró per a cada fitxer a generar: es crea el fitxer, s'assigna el text corresponent a una variable *string* anomenada filetxt, s'escriu aquesta variable dins el fitxer i es tanca el fitxer. El següent exemple crea el programa *Start()* del robot R1.

```
FileWriter writer1=new FileWriter(new File(destFN+"start.pgx"));
filetxt=CreateStartR1File(sourceDataLayer);
writer1.write(filetxt);
writer1.close();
```

Tots els mètodes assignats a la variable filetxt retornen el text que contindrà cada fitxer. Els mètodes de l'apartat 7.3.1 que serveixen per generar el codi son usats per les funcions que afegeixen les etiquetes *XML* corresponents, de forma que en el moment d'omplir el contingut de l'etiqueta de codi dels fitxers *pgx* es criden els corresponents mètodes que generen el codi. Les excepcions seran els fitxers *dtx*, *ltx* i *pjx* que no depenen directament del codi generat i s'han d'implementar de nou. Els mètodes nous implementats son els següents:

```
private String InitialFile(String name){...}
private String CreateStopFile(DataLayer sourceDatalayer){...}
private String CreateStartR1File(DataLayer sourceDataLayer){...}
private String CreateStartR2File(DataLayer sourceDataLayer){...}
private String CreateVariableR1File(DataLayer pnmlData){...}
private String CreateVariableR2File(DataLayer pnmlData){...}
private String CreateProjectR1File(DataLayer pnmlData){...}
private String CreateProjectR2File(DataLayer pnmlData){...}
```

De tots aquests mètodes, se'n mostra el primer ja que s'utilitza en tots els altres. La seva implementació és la següent:

```
private String InitialFile(String name){
        String j=new String();
        j="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>"+"\n";
        j+="<programList xmlns=\"ProgramNameSpace\" >"+"\n";
        j+="    <program name=\""+name+"\" public=\"false\" >"+"\n";
        j+="        <description></description>"+"\n";
        j+="        <paramSection>"+"\n";
        j+="        </paramSection>"+"\n";
        j+="        <localSection>"+"\n";
        j+="        </localSection>"+"\n";
        j+="        <source>"+"\n";
        j+="            <code>"+"\n";
return j;}
```

on se li passa com a variable el nom del fitxer a generar i crea la capçalera i les primeres etiquetes dels fitxers, ja que el contingut de tots ells es el mateix fins a la etiqueta de codi *<code>.* Aquest mètode també serveix per il·lustrar la manera en que estan implementats els

altres, definint una variable de tipus *string* anomenada j i assignant-li a aquesta el text corresponent.

Per més detalls de la implementació sencera de tot el mòdul consultar el codi de l'annex B.2.

Conclusions

Aquest projecte ha assolit les fites proposades. La tasca que s'ha dut a terme ha estat complexa i moltes vegades ha generat més interrogants que respostes, però complint al final amb les expectatives dipositades.

El mòdul de programació implementat s'ha integrat totalment en l'aplicació, dotant-la d'una eina idònia per facilitar la programació dels robots, i ampliant el ventall d'usuaris que podran accedir a una programació complexa amb uns coneixements mínims. La integració dels elements en la interfície gràfica es posa de manifest en el fet que desenvolupar una aplicació complerta, des de la creació de la xarxa fins a la obtenció de codi i el posterior funcionament dels robots, resulta una feina senzilla i intuïtiva.

Gràcies a que el modelat en xarxa de Petri d'una cel·la de treball complexa permet una simplificació considerable del sistema, la feina típica de la programació es redueix només a conèixer les accions que realitzen els robots, deixant per a la generació automàtica de codi que aporta el mòdul tota la feina de control, gestió de recursos compartits i sincronització amb altres dispositius.

A més, s'ha aconseguit adaptar el projecte al format estàndard de xarxes de Petri de manera que no interfereix en el funcionament de les altres aplicacions que l'utilitzen, permetent realitzar anàlisis externs al programa.

L'aplicació s'ha desenvolupat fent servir programari lliure i aquest s'ha disposat com a programari lliure. La memòria ha estat confeccionada amb un estil curós i amb un llenguatge de forma que qualsevol persona amb un mínim nivell tècnic pugui continuar la tasca realitzada.

Els futurs treballs interessats en ampliar les funcionalitats presentades en aquest projecte podrien centrar els esforços en adaptar les característiques del programa per tal que sigui capaç de suportar sistemes amb més de dos robots que comparteixin l'espai de treball.

Bibliografia

Referències bibliogràfiques

[1] T. MURATA, *Petri Nets: Properties, Analysis and Applications*, in Proc. Of the IEEE, Vol.77, No. 4, pp 541-580, 1989.

[2] PETRI NETS WORLD
[http://www.informatik.uni-hamburg.de/TGI/PetriNets/, visitat el novembre de 2004]

[3] W.KHALIL I E.DOMBRE, *Modeling indetification and Control of Robots*. Hermes Penton Science, 2002.

[4] J. J. CRAIG, *Introduction to Robotics: mechanical and control*. Addison Wesley, 1989.

[5] G. V. ARNOLD, *A graphical interface based on grafcet for programming industrial robots off-line*, PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATION AND ROBOTICS. Vol. II, p. 117-118. Barcelona, Setembre 14 –17, 2005

[6] J. ROSELL, *Assembly and Task Planning Using Petri Nets: A Survey* , JOURNAL OF ENGINEERING MANUFACTURE - PART B, (ISSN 09544054), Vol. 218, issue B7, p. 987-994, 2004

[7] I. H. SUH et al, *Design of a supervisory control system for multiple robotic systems*, in: Proc. of the 1996 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'96, 1996, p. 332–339.

[8] ECLIPSE PROJECT
[http://www.eclipse.org/, visitat el novembre de 2004]

ANNEXOS

A. Taula dels programes relacionats amb xarxes de Petri

Aplicació	Característiques		Entorns
	PN suportades	Components	
ALPHA/Sim	PN d'alt nivell PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi simple de comportament	SunOS Solaris MS Windows
ARP Gratuït	PN lloc/transició PN temporitzades	Simulació ràpida Espai d'estats P-Invariants T-Invariants Anàlisi estructural Anàlisi simple de comportament	PC, MS DOS
Artifex	PN orientades a objectes PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi estructural Anàlisi avançat de comportament	Sun, SunOS HP, HP-UX IRIX 6.5 PC, Linux PC, MS Windows
CoopnBuilder Gratuït	PN orientades a objectes	Editor de gràfics Animació del joc de marques Simulació ràpida Generació de codi	Java
COSA BPM	PN d'alt nivell	Editor de gràfics Animació del joc de marques Sistema de gestió de flux	Sun, SunOS HP, HP-UX PC, Linux PC, MS Windows Java
CPN-AMI Gratuït	PN d'alt nivell PN lloc/transició	Editor de gràfics Simulació ràpida Espai d'estats P-Invariants T-Invariants Anàlisi estructural	Linux Macintosh
CPN Tools Gratuït	PN d'alt nivell PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Anàlisi simple de comportament	PC, MS Windows

		Intercanvi de format de fitxers	
Design/CPN Gratuït	PN d'alt nivell PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Anàlisi simple de comportament Intercanvi de format de fitxers	Sun HP Silicon Graphics Linux
ExSpect	PN d'alt nivell PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi simple de comportament Anàlisi avançat de comportament Rapid Prototyping Sistema de gestió de flux	MS Windows
FLOWer	PN lloc/transició	Editor de gràfics Rapid Prototyping Sistema de gestió de flux	HP, HP-UX PC, Linux PC, MS Windows Java
F-net	PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats P-Invariants T-Invariants Anàlisi estructural Anàlisi simple de comportament Anàlisi avançat de comportament	MS Windows OS/2
GDToolkit	PN d'alt nivell PN lloc/transició	Editor de gràfics	Sun Linux MS Windows
GreatSPN	PN d'alt nivell PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants Anàlisi estructural Anàlisi avançat de comportament	Sun Linux
Helena Gratuït	PN d'alt nivell	Espai d'estats Reducció d'espai d'estats Reducció de xarxes	Linux
HiQPN-Tool Gratuït	PN d'alt nivell PN estocàstiques	Editor de gràfics Animació del joc de marques Espai d'estats P-Invariants	Sun

		T-Invariants Anàlisi avançat de comportament Intercanvi de format de fitxers	
HPSim Gratuït	PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi simple de comportament	PC, MS Windows
INA Gratuït	PN d'alt nivell PN lloc/transició PN temporitzades	Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants Reducció de xarxes Anàlisi estructural Anàlisi simple de comportament Anàlisi avançat de comportament Intercanvi de format de fitxers	Sun Linux MS Windows
Income Suite	PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Sistema de gestió de flux	Sun, SunOS HP, HP-UX PC, Linux PC, MS Windows Java
JARP Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Espai d'estats Intercanvi de format de fitxers	Java
JFern Gratuït	PN orientades a objectes PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Anàlisi simple de comportament Intercanvi de format de fitxers	Java
JPetriNet Gratuït	PN lloc/transició PN temporitzades	Editor de gràfics Anàlisi estructural	Java
Maria Gratuït	PN d'alt nivell PN lloc/transició PN modular d'alt nivell	Animació del joc de marques Simulació ràpida Espai d'estats Alt nivell de dades i operacions	Sun, SunOS 5.7 Digital, UNIX 4.0 IRIX 6.5 HP-UX 11.22 PC, Linux PC, MS Windows Apple, Mac OS
MISS-RdP	PN estocàstiques PN temporitzades PN colorejades	Editor de gràfics Simulació ràpida	Sun, SunOS PC, MS Windows
The Model Gratuït	PN d'alt nivell PN lloc/transició	Espai d'estats Reducció d'espai d'estats	Sun Linux
Netlab	PN lloc/transició	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants	PC, MS Windows

		Anàlisi estructural Intercanvi de format de fitxers	
Nevod	PN d'alt nivell	Editor de gràfics Animació del joc de marques Simulació ràpida	MS DOS
Opera	PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants Reducció de xarxes Anàlisi estructural Anàlisi simple de comportament Anàlisi avançat de comportament Intercanvi de format de fitxers	MS DOS
P3 Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Reducció d'espai d'estats Intercanvi de format de fitxers	PC, MS Windows
PACE	PN d'alt nivell PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Reducció de xarxes Tècniques difuses, optimització	PC, MS Windows
PED Gratuït	PN lloc/transició PN temporitzades	Editor de gràfics	Sun Linux
PEP Gratuït	PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants Reducció de xarxes Anàlisi estructural Intercanvi de format de fitxers Model Checking Petri Net Generators	Sun Linux
Petrigen Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Síntesi	PC, Linux PC, MS Windows
Petri-lld Gratuït	PN elementals	Editor de gràfics Animació del joc de marques Rapid Prototyping	Java
Petri Net Kernel Gratuït	PN d'alt nivell PN lloc/transició	Editor de gràfics Animació del joc de marques Intercanvi de format de fitxers	Java

Petri .NET Simulator	PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi simple de comportament	PC, MS Windows
Petri Net Toolbox	PN lloc/transició PN estocàstiques PN temporitzades PN estocàstiques	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats P-Invariants T-Invariants Anàlisi estructural Anàlisi simple de comportament Anàlisi avançat de comportament Intercanvi de format de fitxers	PC, MS Windows
PetriSim Gratuït	PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Simulació ràpida	MS DOS
Platform Independent Petri Net Editor (PIPE) Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats Reducció d'espai d'estats P-Invariants T-Invariants Anàlisi estructural Anàlisi simple de comportament Intercanvi de format de fitxers Mòduls d'anàlisi ampliables Formats de fitxers ampliables	Java
PNSim Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Anàlisi estructural Anàlisi de xarxes simples	MS Windows Java
PNtalk Gratuït	PN d'alt nivell PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi simple de comportament	Sun MS Windows
Poses++ Gratuït	PN d'alt nivell	Simulació ràpida	Sun Linux MS Windows
Predator Gratuït	PN lloc/transició PN estocàstiques PN geràrquiques	Editor de gràfics P-Invariants T-Invariants Intercanvi de format de fitxers Mòduls d'anàlisi ampliables	Java
PROD Gratuït	PN d'alt nivell PN lloc/transició	Espai d'estats Reducció d'espai d'estats	Sun, SunOS HP, HP-UX PC, Linux PC, MS Windows
ProM	PN lloc/transició	Espai d'estats	PC, Linux

framework Gratuït		P-Invariants T-Invariants Intercanvi de format de fitxers	PC, MS Windows Macintosh, Mac OS Java
PROTOS	PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Simulació ràpida Anàlisi estructural Anàlisi simple de comportament Rapid Prototyping Sistema de gestió de flux	PC, MS Windows
Renew Gratuït	PN orientades a objectes PN d'alt nivell PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Intercanvi de format de fitxers Rapid Prototyping Sistema de gestió de flux	Java
Romeo	PN temporitzades	Editor de gràfics Espai d'estats	PC, Linux Macintosh, Mac OS
SEA Gratuït	PN d'alt nivell PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida	Sun
SIPN-Editor Gratuït	PN interpretades	Editor de gràfics Generació de codi per PLC's	Java
Simulaworks	PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Simulació ràpida	PC, MS Windows
SPNP	PN d'alt nivell PN estocàstiques	Anàlisi avançat de comportament	PC, MS Windows
StpnPlay Gratuït	PN estocàstiques PN temporitzades	Editor de gràfics Simulació ràpida Anàlisi simple de comportament Intercanvi de format de fitxers	PC, MS Windows
SYROCO	PN d'alt nivell PN temporitzades Instanciació dinàmica de PN Codi C++ associat a llocs i transicions	Editor de gràfics Simulació ràpida Anàlisi simple de comportament Intercanvi de format de fitxers Generació de codi C++	C++
TimeNET	PN d'alt nivell PN lloc/transició PN estocàstiques PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Espai d'estats P-Invariants Anàlisi estructural Anàlisi simple de comportament Anàlisi avançat de comportament Intercanvi de format de fitxers	Sun Linux
Tina Gratuït	PN lloc/transició PN temporitzades Time Petri Nets	Editor de gràfics Animació del joc de marques Espai d'estats Reducció d'espai d'estats	Sun, SunOS PC, Linux PC, MS Windows Macintosh, Mac OS

		P-Invariants T-Invariants	
Visual Object Net ++ Gratuït	PN lloc/transició PN temporitzades	Editor de gràfics Animació del joc de marques Simulació ràpida Anàlisi estructural Anàlisi simple de comportament	MS Windows
WebSPN Gratuït	PN estocàstiques	Editor de gràfics Animació del joc de marques Anàlisi avançat de comportament Intercanvi de format de fitxers	PC, Linux Java
WINSIM Gratuït	PN d'alt nivell PN temporitzades	Simulació ràpida Anàlisi simple de comportament	PC, MS Windows
WoPeD Gratuït	PN lloc/transició	Editor de gràfics Animació del joc de marques Intercanvi de format de fitxers Sistema de gestió de flux	Java
XRL/flower Gratuït	PN d'alt nivell	Sistema de gestió de flux	PC, MS Windows Java
YAWL Gratuït	PN d'alt nivell	Editor de gràfics Animació del joc de marques Intercanvi de format de fitxers Sistema de gestió de flux	Java

B. EL CODI

B.1. Fitxers XSLT

En aquest apartat es mostren els fitxers *XSLT* serveixen per guardar i obrir les xarxes des del programa PIPE, tan els originals com els modificats.

El següent fitxer *GeneratePNML* transforma la informació de la capa de dades al format *PNML.*

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" encoding="iso-8859-1" omit-xml-
      declaration="no" indent="yes" version="1.0" />
  - <xsl:template match="/">
    - <xsl:element name="pnml">
        <xsl:apply-templates select="pnml" />
      </xsl:element>
    </xsl:template>
  - <xsl:template match="pnml">
    - <xsl:element name="net">
      - <xsl:attribute name="id">
          <xsl:value-of select="net/@id" />
        </xsl:attribute>
      - <xsl:attribute name="type">
          <xsl:value-of select="net/@type" />
        </xsl:attribute>
      - <xsl:apply-templates select="net/labels">
          <xsl:sort select="@text" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/place">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/transition">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/arc">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net/place">
    - <xsl:element name="place">
        <xsl:call-template name="place-transition" />
        <xsl:call-template name="initialMarking" />
      </xsl:element>
    </xsl:template>
```

```
- <xsl:template match="net/labels">
  - <xsl:element name="labels">
    - <xsl:attribute name="x">
        <xsl:value-of select="@positionX" />
      </xsl:attribute>
    - <xsl:attribute name="y">
        <xsl:value-of select="@positionY" />
      </xsl:attribute>
    - <xsl:attribute name="width">
        <xsl:value-of select="@width" />
      </xsl:attribute>
    - <xsl:attribute name="height">
        <xsl:value-of select="@height" />
      </xsl:attribute>
    - <xsl:attribute name="border">
        <xsl:value-of select="@border" />
      </xsl:attribute>
    - <xsl:element name="text">
        <xsl:value-of select="@text" />
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template match="net/transition">
  - <xsl:element name="transition">
      <xsl:call-template name="place-transition" />
      <xsl:call-template name="orientation" />
      <xsl:call-template name="myrate" />
      <xsl:call-template name="timed" />
    </xsl:element>
  </xsl:template>
- <xsl:template name="place-transition">
  - <xsl:attribute name="id">
      <xsl:value-of select="@id" />
    </xsl:attribute>
    <xsl:call-template name="graphics" />
    <xsl:call-template name="name" />
  </xsl:template>
- <xsl:template match="net/arc">
  - <xsl:element name="arc">
    - <xsl:attribute name="id">
        <xsl:value-of select="@id" />
      </xsl:attribute>
    - <xsl:attribute name="source">
        <xsl:value-of select="@source" />
      </xsl:attribute>
    - <xsl:attribute name="target">
        <xsl:value-of select="@target" />
      </xsl:attribute>
      <xsl:call-template name="graphics" />
      <xsl:call-template name="inscription" />
```

```
      - <xsl:apply-templates select="arcpath">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      </xsl:element>
    </xsl:template>
  - <xsl:template match="arcpath">
    - <xsl:element name="arcpath">
      - <xsl:attribute name="id">
          <xsl:value-of select="@id" />
        </xsl:attribute>
      - <xsl:attribute name="x">
          <xsl:value-of select="@xCoord" />
        </xsl:attribute>
      - <xsl:attribute name="y">
          <xsl:value-of select="@yCoord" />
        </xsl:attribute>
      - <xsl:attribute name="curvePoint">
          <xsl:value-of select="@arcPointType" />
        </xsl:attribute>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="graphics">
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@positionX) > 0) and (string-
          length(@positionY) > 0)">
        - <xsl:element name="position">
          - <xsl:attribute name="x">
              <xsl:value-of select="@positionX" />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@positionY" />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="name">
    - <xsl:element name="name">
      - <xsl:element name="value">
          <xsl:value-of select="@name" />
        </xsl:element>
      - <xsl:element name="graphics">
        - <xsl:if test="(string-length(@nameOffsetX) > 0) and
            (string-length(@nameOffsetY) > 0)">
          - <xsl:element name="offset">
            - <xsl:attribute name="x">
                <xsl:value-of select="@nameOffsetX" />
              </xsl:attribute>
            - <xsl:attribute name="y">
                <xsl:value-of select="@nameOffsetY" />
```

```
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="initialMarking">
  - <xsl:element name="initialMarking">
    - <xsl:element name="value">
        <xsl:value-of select="@initialMarking" />
      </xsl:element>
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@markingOffsetX) > 0) and
          (string-length(@markingOffsetY) > 0)">
        - <xsl:element name="offset">
          - <xsl:attribute name="x">
              <xsl:value-of select="@markingOffsetX" />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@markingOffsetY" />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="inscription">
  - <xsl:element name="inscription">
    - <xsl:element name="value">
        <xsl:value-of select="@inscription" />
      </xsl:element>
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@inscriptionOffsetX) > 0)
          and (string-length(@inscriptionOffsetY) > 0)">
        - <xsl:element name="offset">
          - <xsl:attribute name="x">
              <xsl:value-of select="@inscriptionOffsetX"
                />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@inscriptionOffsetY"
                />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="myrate">
  - <xsl:element name="rate">
```

```
      - <xsl:element name="value">
            <xsl:value-of select="@rate" />
        </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="timed">
  - <xsl:element name="timed">
    - <xsl:element name="value">
            <xsl:value-of select="@timed" />
        </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="orientation">
  - <xsl:element name="orientation">
    - <xsl:element name="value">
            <xsl:value-of select="@angle" />
        </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

El següent fitxer és el *GeneratePNMLMod*, resultat d'adaptar-lo a les funcionalitats d'aquest projecte, les parts marcades són les que varien de l'anterior.

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" encoding="iso-8859-1" omit-xml-
      declaration="no" indent="yes" version="1.0" />
  - <xsl:template match="/">
    - <xsl:element name="pnml">
        <xsl:apply-templates select="pnml" />
      </xsl:element>
    </xsl:template>
  - <xsl:template match="pnml">
    - <xsl:element name="net">
      - <xsl:attribute name="id">
          <xsl:value-of select="net/@id" />
        </xsl:attribute>
      - <xsl:attribute name="type">
          <xsl:value-of select="net/@type" />
        </xsl:attribute>
      - <xsl:apply-templates select="net/labels">
          <xsl:sort select="@text" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/place">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/transition">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      - <xsl:apply-templates select="net/arc">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net/place">
    - <xsl:element name="place">
        <xsl:call-template name="place-transition" />
        <xsl:call-template name="initialMarking" />
        <xsl:call-template name="subtype" />
        <xsl:call-template name="codeSnipet" />
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net/labels">
    - <xsl:element name="labels">
      - <xsl:attribute name="x">
          <xsl:value-of select="@positionX" />
        </xsl:attribute>
      - <xsl:attribute name="y">
```

```
            <xsl:value-of select="@positionY" />
          </xsl:attribute>
        - <xsl:attribute name="width">
            <xsl:value-of select="@width" />
          </xsl:attribute>
        - <xsl:attribute name="height">
            <xsl:value-of select="@height" />
          </xsl:attribute>
        - <xsl:attribute name="border">
            <xsl:value-of select="@border" />
          </xsl:attribute>
        - <xsl:element name="text">
            <xsl:value-of select="@text" />
          </xsl:element>
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net/transition">
    - <xsl:element name="transition">
        <xsl:call-template name="place-transition" />
        <xsl:call-template name="orientation" />
        <xsl:call-template name="myrate" />
        <xsl:call-template name="timed" />
      </xsl:element>
    </xsl:template>
  - <xsl:template name="place-transition">
    - <xsl:attribute name="id">
        <xsl:value-of select="@id" />
      </xsl:attribute>
      <xsl:call-template name="graphics" />
      <xsl:call-template name="name" />
    </xsl:template>
  - <xsl:template match="net/arc">
    - <xsl:element name="arc">
      - <xsl:attribute name="id">
          <xsl:value-of select="@id" />
        </xsl:attribute>
      - <xsl:attribute name="source">
          <xsl:value-of select="@source" />
        </xsl:attribute>
      - <xsl:attribute name="target">
          <xsl:value-of select="@target" />
        </xsl:attribute>
        <xsl:call-template name="graphics" />
        <xsl:call-template name="inscription" />
      - <xsl:apply-templates select="arcpath">
          <xsl:sort select="@id" data-type="text" />
        </xsl:apply-templates>
      </xsl:element>
    </xsl:template>
  - <xsl:template match="arcpath">
```

```
    - <xsl:element name="arcpath">
      - <xsl:attribute name="id">
          <xsl:value-of select="@id" />
        </xsl:attribute>
      - <xsl:attribute name="x">
          <xsl:value-of select="@xCoord" />
        </xsl:attribute>
      - <xsl:attribute name="y">
          <xsl:value-of select="@yCoord" />
        </xsl:attribute>
      - <xsl:attribute name="curvePoint">
          <xsl:value-of select="@arcPointType" />
        </xsl:attribute>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="graphics">
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@positionX) > 0) and (string-
          length(@positionY) > 0)">
        - <xsl:element name="position">
          - <xsl:attribute name="x">
              <xsl:value-of select="@positionX" />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@positionY" />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="name">
    - <xsl:element name="name">
      - <xsl:element name="value">
          <xsl:value-of select="@name" />
        </xsl:element>
      - <xsl:element name="graphics">
        - <xsl:if test="(string-length(@nameOffsetX) > 0) and
            (string-length(@nameOffsetY) > 0)">
          - <xsl:element name="offset">
            - <xsl:attribute name="x">
                <xsl:value-of select="@nameOffsetX" />
              </xsl:attribute>
            - <xsl:attribute name="y">
                <xsl:value-of select="@nameOffsetY" />
              </xsl:attribute>
            </xsl:element>
          </xsl:if>
        </xsl:element>
      </xsl:element>
    </xsl:template>
```

```
- <xsl:template name="initialMarking">
  - <xsl:element name="initialMarking">
    - <xsl:element name="value">
        <xsl:value-of select="@initialMarking" />
      </xsl:element>
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@markingOffsetX) > 0) and
          (string-length(@markingOffsetY) > 0)">
        - <xsl:element name="offset">
          - <xsl:attribute name="x">
              <xsl:value-of select="@markingOffsetX" />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@markingOffsetY" />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="inscription">
  - <xsl:element name="inscription">
    - <xsl:element name="value">
        <xsl:value-of select="@inscription" />
      </xsl:element>
    - <xsl:element name="graphics">
      - <xsl:if test="(string-length(@inscriptionOffsetX) > 0)
          and (string-length(@inscriptionOffsetY) > 0)">
        - <xsl:element name="offset">
          - <xsl:attribute name="x">
              <xsl:value-of select="@inscriptionOffsetX"
                />
            </xsl:attribute>
          - <xsl:attribute name="y">
              <xsl:value-of select="@inscriptionOffsetY"
                />
            </xsl:attribute>
          </xsl:element>
        </xsl:if>
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="myrate">
  - <xsl:element name="rate">
    - <xsl:element name="value">
        <xsl:value-of select="@rate" />
      </xsl:element>
    </xsl:element>
  </xsl:template>
- <xsl:template name="timed">
```

```
    - <xsl:element name="timed">
      - <xsl:element name="value">
          <xsl:value-of select="@timed" />
        </xsl:element>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="orientation">
    - <xsl:element name="orientation">
      - <xsl:element name="value">
          <xsl:value-of select="@angle" />
        </xsl:element>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="subtype">
    - <xsl:element name="subtype">
      - <xsl:element name="value">
          <xsl:value-of select="@subtype" />
        </xsl:element>
      </xsl:element>
    </xsl:template>
  - <xsl:template name="codeSnipet">
    - <xsl:element name="codeSnipet">
      - <xsl:element name="value">
          <xsl:value-of select="@codeSnipet" />
        </xsl:element>
      </xsl:element>
    </xsl:template>
  </xsl:stylesheet>
```

Seguidament es presenta el fitxer *GenerateObjectList* que transforma la informació d'una xarxa guardada en *PNML* en un format entenedor per la capa de dades.

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" encoding="iso-8859-1" omit-xml-
      declaration="no" indent="yes" />
    <xsl:strip-space elements="*" />
  - <xsl:template match="pnml">
    - <xsl:element name="pntools">
      - <xsl:attribute name="id">
          <xsl:value-of select="net/@id" />
        </xsl:attribute>
      - <xsl:attribute name="type">
          <xsl:value-of select="net/@type" />
        </xsl:attribute>
        <xsl:apply-templates select="net" />
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="page" />
      <xsl:apply-templates select="module" />
    </xsl:template>
  - <xsl:template match="page">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="module" />
    </xsl:template>
  - <xsl:template match="module">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="interface" />
    </xsl:template>
  - <xsl:template match="labels">
    - <xsl:element name="labels">
      - <xsl:attribute name="xPosition">
          <xsl:value-of select="@x" />
        </xsl:attribute>
      - <xsl:attribute name="yPosition">
          <xsl:value-of select="@y" />
        </xsl:attribute>
```

```
      - <xsl:attribute name="w">
          <xsl:value-of select="@width" />
        </xsl:attribute>
      - <xsl:attribute name="h">
          <xsl:value-of select="@height" />
        </xsl:attribute>
      - <xsl:attribute name="border">
          <xsl:value-of select="@border" />
        </xsl:attribute>
      - <xsl:attribute name="txt">
          <xsl:value-of select="text" />
        </xsl:attribute>
    </xsl:element>
  </xsl:template>
- <xsl:template match="place">
  - <xsl:element name="place">
      <xsl:call-template name="place-transition" />
    - <xsl:attribute name="initialMarking">
        <xsl:value-of select="initialMarking/value" />
      </xsl:attribute>
    - <xsl:attribute name="markingOffsetX">
        <xsl:value-of select="initialMarking/graphics/offset/@x"
          />
      </xsl:attribute>
    - <xsl:attribute name="markingOffsetY">
        <xsl:value-of select="initialMarking/graphics/offset/@x"
          />
      </xsl:attribute>
    </xsl:element>
  </xsl:template>
- <xsl:template match="transition">
  - <xsl:element name="transition">
    - <xsl:attribute name="rate">
        <xsl:value-of select="rate/value" />
      </xsl:attribute>
    - <xsl:attribute name="timed">
        <xsl:value-of select="timed/value" />
      </xsl:attribute>
    - <xsl:attribute name="angle">
        <xsl:value-of select="orientation/value" />
      </xsl:attribute>
      <xsl:call-template name="place-transition" />
    </xsl:element>
  </xsl:template>
- <xsl:template match="arc">
  - <xsl:element name="arc">
      <xsl:call-template name="place-transition-arc" />
    - <xsl:attribute name="source">
        <xsl:value-of select="@source" />
      </xsl:attribute>
```

```
      - <xsl:attribute name="target">
          <xsl:value-of select="@target" />
        </xsl:attribute>
      - <xsl:attribute name="inscription">
          <xsl:value-of select="inscription/value" />
        </xsl:attribute>
      - <xsl:attribute name="inscriptionOffsetX">
          <xsl:value-of select="inscription/graphics/offset/@x" />
        </xsl:attribute>
      - <xsl:attribute name="inscriptionOffsetY">
          <xsl:value-of select="inscription/graphics/offset/@x" />
        </xsl:attribute>
        <xsl:apply-templates select="arcpath" />
      </xsl:element>
    </xsl:template>
  - <xsl:template name="place-transition">
      <xsl:call-template name="place-transition-arc" />
    - <xsl:attribute name="name">
        <xsl:value-of select="name/value" />
      </xsl:attribute>
    - <xsl:attribute name="nameOffsetX">
        <xsl:value-of select="name/graphics/offset/@x" />
      </xsl:attribute>
    - <xsl:attribute name="nameOffsetY">
        <xsl:value-of select="name/graphics/offset/@y" />
      </xsl:attribute>
    </xsl:template>
  - <xsl:template name="place-transition-arc">
    - <xsl:attribute name="id">
        <xsl:value-of select="@id" />
      </xsl:attribute>
    - <xsl:attribute name="positionX">
        <xsl:value-of select="graphics/position/@x" />
      </xsl:attribute>
    - <xsl:attribute name="positionY">
        <xsl:value-of select="graphics/position/@y" />
      </xsl:attribute>
    </xsl:template>
  - <xsl:template match="arcpath">
    - <xsl:element name="arcpath">
      - <xsl:attribute name="x">
          <xsl:value-of select="@x" />
        </xsl:attribute>
      - <xsl:attribute name="y">
          <xsl:value-of select="@y" />
        </xsl:attribute>
      - <xsl:attribute name="arcPointType">
          <xsl:value-of select="@curvePoint" />
        </xsl:attribute>
      </xsl:element>
```

```
    </xsl:template>
  - <xsl:template match="interface">
    - <xsl:element name="arc">
        <xsl:attribute name="id">InterfaceArc</xsl:attribute>
        <xsl:attribute name="positionX" />
        <xsl:attribute name="positionY" />
      - <xsl:attribute name="source">
          <xsl:value-of select="../referencePlace/@id" />
        </xsl:attribute>
      - <xsl:attribute name="target">
          <xsl:value-of select="importPlace/@target" />
        </xsl:attribute>
        <xsl:attribute name="inscription" />
        <xsl:attribute name="inscriptionOffsetX" />
        <xsl:attribute name="inscriptionOffsetY" />
      </xsl:element>
    - <xsl:element name="arc">
        <xsl:attribute name="id">InterfaceArc</xsl:attribute>
        <xsl:attribute name="positionX" />
        <xsl:attribute name="positionY" />
      - <xsl:attribute name="source">
          <xsl:value-of select="exportPlace/@id" />
        </xsl:attribute>
      - <xsl:attribute name="target">
          <xsl:value-of select="exportPlace/@ref" />
        </xsl:attribute>
        <xsl:attribute name="inscription" />
        <xsl:attribute name="inscriptionOffsetX" />
        <xsl:attribute name="inscriptionOffsetY" />
      </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

Finalment es presenta el fitxer anterior modificat: *GenerateObjectListMod.*

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" encoding="iso-8859-1" omit-xml-
      declaration="no" indent="yes" />
    <xsl:strip-space elements="*" />
  - <xsl:template match="pnml">
    - <xsl:element name="pntools">
      - <xsl:attribute name="id">
          <xsl:value-of select="net/@id" />
        </xsl:attribute>
      - <xsl:attribute name="type">
          <xsl:value-of select="net/@type" />
        </xsl:attribute>
        <xsl:apply-templates select="net" />
      </xsl:element>
    </xsl:template>
  - <xsl:template match="net">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="page" />
      <xsl:apply-templates select="module" />
    </xsl:template>
  - <xsl:template match="page">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="module" />
    </xsl:template>
  - <xsl:template match="module">
      <xsl:apply-templates select="labels" />
      <xsl:apply-templates select="place" />
      <xsl:apply-templates select="transition" />
      <xsl:apply-templates select="arc" />
      <xsl:apply-templates select="interface" />
    </xsl:template>
  - <xsl:template match="labels">
    - <xsl:element name="labels">
      - <xsl:attribute name="xPosition">
          <xsl:value-of select="@x" />
        </xsl:attribute>
      - <xsl:attribute name="yPosition">
          <xsl:value-of select="@y" />
        </xsl:attribute>
```

```
      - <xsl:attribute name="w">
          <xsl:value-of select="@width" />
        </xsl:attribute>
      - <xsl:attribute name="h">
          <xsl:value-of select="@height" />
        </xsl:attribute>
      - <xsl:attribute name="border">
          <xsl:value-of select="@border" />
        </xsl:attribute>
      - <xsl:attribute name="txt">
          <xsl:value-of select="text" />
        </xsl:attribute>
    </xsl:element>
  </xsl:template>
- <xsl:template match="place">
  - <xsl:element name="place">
      <xsl:call-template name="place-transition" />
    - <xsl:attribute name="initialMarking">
        <xsl:value-of select="initialMarking/value" />
      </xsl:attribute>
    - <xsl:attribute name="markingOffsetX">
        <xsl:value-of select="initialMarking/graphics/offset/@x"
          />
      </xsl:attribute>
    - <xsl:attribute name="markingOffsetY">
        <xsl:value-of select="initialMarking/graphics/offset/@x"
          />
      </xsl:attribute>
    - <xsl:attribute name="subtype">
        <xsl:value-of select="subtype/value" />
      </xsl:attribute>
    - <xsl:attribute name="codeSnipet">
        <xsl:value-of select="codeSnipet/value" />
      </xsl:attribute>
    </xsl:element>
  </xsl:template>
- <xsl:template match="transition">
  - <xsl:element name="transition">
    - <xsl:attribute name="rate">
        <xsl:value-of select="rate/value" />
      </xsl:attribute>
    - <xsl:attribute name="timed">
        <xsl:value-of select="timed/value" />
      </xsl:attribute>
    - <xsl:attribute name="angle">
        <xsl:value-of select="orientation/value" />
      </xsl:attribute>
      <xsl:call-template name="place-transition" />
    </xsl:element>
  </xsl:template>
```

```
- <xsl:template match="arc">
  - <xsl:element name="arc">
      <xsl:call-template name="place-transition-arc" />
    - <xsl:attribute name="source">
        <xsl:value-of select="@source" />
      </xsl:attribute>
    - <xsl:attribute name="target">
        <xsl:value-of select="@target" />
      </xsl:attribute>
    - <xsl:attribute name="inscription">
        <xsl:value-of select="inscription/value" />
      </xsl:attribute>
    - <xsl:attribute name="inscriptionOffsetX">
        <xsl:value-of select="inscription/graphics/offset/@x" />
      </xsl:attribute>
    - <xsl:attribute name="inscriptionOffsetY">
        <xsl:value-of select="inscription/graphics/offset/@x" />
      </xsl:attribute>
      <xsl:apply-templates select="arcpath" />
    </xsl:element>
  </xsl:template>
- <xsl:template name="place-transition">
    <xsl:call-template name="place-transition-arc" />
  - <xsl:attribute name="name">
      <xsl:value-of select="name/value" />
    </xsl:attribute>
  - <xsl:attribute name="nameOffsetX">
      <xsl:value-of select="name/graphics/offset/@x" />
    </xsl:attribute>
  - <xsl:attribute name="nameOffsetY">
      <xsl:value-of select="name/graphics/offset/@y" />
    </xsl:attribute>
  </xsl:template>
- <xsl:template name="place-transition-arc">
  - <xsl:attribute name="id">
      <xsl:value-of select="@id" />
    </xsl:attribute>
  - <xsl:attribute name="positionX">
      <xsl:value-of select="graphics/position/@x" />
    </xsl:attribute>
  - <xsl:attribute name="positionY">
      <xsl:value-of select="graphics/position/@y" />
    </xsl:attribute>
  </xsl:template>
- <xsl:template match="arcpath">
  - <xsl:element name="arcpath">
    - <xsl:attribute name="x">
        <xsl:value-of select="@x" />
      </xsl:attribute>
    - <xsl:attribute name="y">
```

```
            <xsl:value-of select="@y" />
          </xsl:attribute>
        - <xsl:attribute name="arcPointType">
            <xsl:value-of select="@curvePoint" />
          </xsl:attribute>
        </xsl:element>
      </xsl:template>
    - <xsl:template match="interface">
      - <xsl:element name="arc">
          <xsl:attribute name="id">InterfaceArc</xsl:attribute>
          <xsl:attribute name="positionX" />
          <xsl:attribute name="positionY" />
        - <xsl:attribute name="source">
            <xsl:value-of select="../referencePlace/@id" />
          </xsl:attribute>
        - <xsl:attribute name="target">
            <xsl:value-of select="importPlace/@target" />
          </xsl:attribute>
          <xsl:attribute name="inscription" />
          <xsl:attribute name="inscriptionOffsetX" />
          <xsl:attribute name="inscriptionOffsetY" />
        </xsl:element>
      - <xsl:element name="arc">
          <xsl:attribute name="id">InterfaceArc</xsl:attribute>
          <xsl:attribute name="positionX" />
          <xsl:attribute name="positionY" />
        - <xsl:attribute name="source">
            <xsl:value-of select="exportPlace/@id" />
          </xsl:attribute>
        - <xsl:attribute name="target">
            <xsl:value-of select="exportPlace/@ref" />
          </xsl:attribute>
          <xsl:attribute name="inscription" />
          <xsl:attribute name="inscriptionOffsetX" />
          <xsl:attribute name="inscriptionOffsetY" />
        </xsl:element>
      </xsl:template>
    </xsl:stylesheet>
```

B.2. Implementació del mòdul

B.2.1. La classe Val3

Aquesta classe és la encarregada d'implementar el mòdul i conté els dos mètodes públics *run* i *getName*.

```
/**
 * Val3 code generation Module
 * @author Jordi Garcia 2004-11-16
 * les notes estan en anglès per millorar la documentació de l'ajuda
 *
 */

package pipe.modules.Val3;

import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;
import javax.swing.BoxLayout;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;
import pipe.dataLayer.DataLayer;
import pipe.dataLayer.Place;
import pipe.dataLayer.Arc;
import pipe.dataLayer.Transition;
import pipe.gui.CreateGui;
import pipe.gui.widgets.ButtonBar;
import pipe.gui.widgets.FileBrowser;
import pipe.gui.widgets.SelectCodePane;
import pipe.gui.widgets.PetriNetChooserPanel;
import pipe.gui.widgets.GeneradorCodiPane;
import pipe.modules.Module;

/**
 * Val3 class implements Val3 code generation module
 */
public class Val3 implements Module {
  private static final String MODULE_NAME = "Code Generation";

  private PetriNetChooserPanel sourceFilePanel;
  private GeneradorCodiPane results1;
  private GeneradorCodiPane results2;
  private SelectCodePane codepanel;

  public void run(DataLayer pnmlData) {
    // Crea la interficie
```

```
  JDialog guiDialog = new JDialog(CreateGui.getApp(),MODULE_NAME,true);

  // 1 Defineix el layout
  Container contentPane=guiDialog.getContentPane();
  contentPane.setLayout(new BoxLayout(contentPane,BoxLayout.PAGE_AXIS));

  // 2 Afageix el file browser
  contentPane.add(sourceFilePanel=new PetriNetChooserPanel("Source Net",pnmlData));

  // 3 Afageix els dos panells de resultats
  JPanel Codi=new JPanel();
  Codi.setLayout(new BoxLayout(Codi,BoxLayout.LINE_AXIS));
  Codi.add(results1=new GeneradorCodiPane());
  results1.setBorder(new TitledBorder(new EtchedBorder(),"Code for R1"));
  Codi.add(results2=new GeneradorCodiPane());
  results2.setBorder(new TitledBorder(new EtchedBorder(),"Code for R2"));
  contentPane.add(Codi);

  // 4 Afageix els butons
  contentPane.add(new ButtonBar(new String[]{"Generate Code","Places","Save Project"},new
   ActionListener[]{Val3ButtonClick,CodePlaceButtonClick,MakeProjectButtonClick}));

  // 5 Ajusta el tamany de la finestra als conitnguts
  guiDialog.pack();

  // 6 Mou la finestra al mig de la pantalla i la fa visible
  guiDialog.setLocationRelativeTo(null);
  guiDialog.setVisible(true);
 }

 public String getName() {
   return MODULE_NAME;
 }
```

B.2.2. Els tres botons del mòdul

```
 /**
  * Display a window where code for all places is shown and can be modified
  */
 ActionListener CodePlaceButtonClick=new ActionListener(){
         public void actionPerformed(ActionEvent arg0){
         DataLayer sourceDataLayer=sourceFilePanel.getDataLayer();
         if(sourceDataLayer==null) return;
         if(!sourceDataLayer.getPetriNetObjects().hasNext()){
                 results1.setText("No Petri net objects defined!");
                 results2.setText("No Petri net objects defined!");}
         else {
         codepanel=new SelectCodePane(sourceDataLayer);}

         };
 };
```

```
/**
* Generate code button click handler
*/
ActionListener Val3ButtonClick=new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		DataLayer sourceDataLayer=sourceFilePanel.getDataLayer();
		String codi1=new String();
		String codi2=new String();
		if(sourceDataLayer==null) return;
		if(!sourceDataLayer.getPetriNetObjects().hasNext()){
			codi1="No Petri net objects defined!";
			codi2="No Petri net objects defined!";}
		else {
			codi1=CompileCodeR1(sourceDataLayer) ;
			codi2=CompileCodeR2(sourceDataLayer) ;}
		results1.setText(codi1);
		results2.setText(codi2);
	}
};

/**
 * Save documents as a Val3 project
 */
ActionListener MakeProjectButtonClick=new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
	DataLayer sourceDataLayer=sourceFilePanel.getDataLayer();
	Transition[] transitions=sourceDataLayer.getTransitions();
	Place[] places=sourceDataLayer.getPlaces();
	Arc[] arcs=sourceDataLayer.getArcs();
	String filetxt=new String();
	String source=new String();
	String target=new String();
	int pos=0;
	int i=0;
	if(sourceDataLayer==null) return;
	if(!sourceDataLayer.getPetriNetObjects().hasNext()){
		results1.setText("No Petri net objects defined!");
		results2.setText("No Petri net objects defined!");}
	else {
	    try{
		//Es crea un fileBrowser per triar el directori a guardar
		FileBrowser fileBrowser=new FileBrowser("Directory","",null);
		fileBrowser.setFileSelectionMode(FileBrowser.DIRECTORIES_ONLY);
		fileBrowser.showDialog(fileBrowser, "Save in this folder");
		//Es creen els fitxers per R1
		String destFN=fileBrowser.getSelectedFile().getPath();
		File directory = new File(destFN+"/"+"Robot_R1");
		directory.mkdir();
		destFN+="/"+"Robot_R1/";
		//fitxer start per R1
		FileWriter writer1=new FileWriter(new File(destFN+"start.pgx"));
		filetxt=CreateStartR1File(sourceDataLayer);
		writer1.write(filetxt);
		writer1.close();
		//fitxer stop per R1
		FileWriter writer2=new FileWriter(new File(destFN+"stop.pgx"));
```

```
            filetxt=CreateStopFile(sourceDataLayer);
        writer2.write(filetxt);
        writer2.close();
        //fitxers de subprogrames de transició per R1
        for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
            //si la transicio te algun lloc de R1 vol dir que pertany a R1
            //i genera els programes de transicions
            if (transitionR1(sourceDataLayer,TransitionNo)){
                FileWriter writer3=new FileWriter(new
                File(destFN+"transition_"+transitions[TransitionNo].getName()+".pgx"));
                filetxt=InitialFile("transition_"+transitions[TransitionNo].getName());
                filetxt+=ProgramCodeTran(sourceDataLayer,TransitionNo);
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer3.write(filetxt);
                writer3.close();
            }
    }
    //fitxers de subprogrames de lloc per R1
    for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
            //si els llocs pertanyen a R1 gener els programes de lloc
            if
            ((places[PlaceNo].getSubtype().equalsIgnoreCase("R1"))|(places[PlaceNo].getSubtyp
            e().equalsIgnoreCase("R1aux"))){
                FileWriter writer4=new FileWriter(new
                File(destFN+"place_"+places[PlaceNo].getName()+".pgx"));
                filetxt=InitialFile("place_"+places[PlaceNo].getName());
                filetxt+="begin"+"\n";
                filetxt+=places[PlaceNo].getCodeSnipet()+"\n";
                for (int ArcNo=0; ArcNo<arcs.length; ArcNo++){
                    source=arcs[ArcNo].getId();
                    pos=source.indexOf(" to ");
                    if (pos==-1){
                        filetxt+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
                    else{
                        source=arcs[ArcNo].getId().substring(0,pos);
                        target=arcs[ArcNo].getId().substring(pos+4);}
                    if (places[PlaceNo].getName().equalsIgnoreCase(target)){
                        if (NexTransAux(sourceDataLayer,PlaceNo)){
                            filetxt+=CodeAux(sourceDataLayer,PlaceNo);}
                        filetxt+="end"+"\n";
                    }

                }
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer4.write(filetxt);
                writer4.close();
            }
    }
    //crea els fitxers relacionats amb els recursos compartits per R1
    for (int placeNo=0; placeNo<places.length; placeNo++){
            if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                i+=1;
                if (i==1){
                    //Fitxers que no depenen de la xarxa de Petri
                    //fitxer comunicació per a R1
```

```
                                FileWriter writer5=new FileWriter(new
                                File(destFN+"comunicacio.pgx"));
                                filetxt=InitialFile("comunicacio");
                                filetxt+="begin"+"\n";
                                filetxt+=comunicacio();
                                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                                filetxt+="    </program>"+"\n"+"</programList>";
                                writer5.write(filetxt);
                                writer5.close();
                                //fitxer d'exlusió mutua per a R1
                                FileWriter writer6=new FileWriter(new File(destFN+"mutex.pgx"));
                                filetxt="<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>"+"\n";
                                filetxt+="<programList xmlns=\"ProgramNameSpace\">"+"\n";
                                filetxt+="  <program name=\"mutex\" public=\"false\">"+"\n";
                                filetxt+="    <description />"+"\n";
                                filetxt+="    <paramSection>"+"\n";
                                filetxt+="      <param name=\"bRC\" type=\"bool\" byVal=\"false\"
                                />"+"\n";
                                filetxt+="    </paramSection>"+"\n";
                                filetxt+="    <localSection />"+"\n";
                                filetxt+="    <source>"+"\n";
                                filetxt+="      <code>"+"\n";
                                filetxt+=mutex();
                                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                                filetxt+="    </program>"+"\n"+"</programList>";
                                writer6.write(filetxt);
                                writer6.close();
                                //Fitxer de lectura per a R1
                                FileWriter writer7=new FileWriter(new File(destFN+"lectura.pgx"));
                                filetxt=InitialFile("lectura");
                                filetxt+=lectura();
                                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                                filetxt+="    </program>"+"\n"+"</programList>";
                                writer7.write(filetxt);
                                writer7.close();
                        }
                }
        }
        //fitxers que depenen de la xarxa de Petri
        if (i>0){
                //fitxer d'escriptura per a R1
                FileWriter writer8=new FileWriter(new File(destFN+"escriptura.pgx"));
                filetxt=InitialFile("escriptura");
                filetxt+=escriptura(sourceDataLayer);
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer8.write(filetxt);
                writer8.close();
                //fitxer de gestió dels recursos compartits per a R1
                FileWriter writer9=new FileWriter(new File(destFN+"gestioRC.pgx"));
                filetxt=InitialFile("gestioRC");
                filetxt+=gestioRC(sourceDataLayer);
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer9.write(filetxt);
                writer9.close();
        }
```

```
//fitxer de dades R1
FileWriter writer10=new FileWriter(new File(destFN+"Robot_R1.dtx"));
filetxt=CreateVariableR1File(sourceDataLayer);
writer10.write(filetxt);
writer10.close();
//crea el fitxer ltx
FileWriter writer11=new FileWriter(new File(destFN+"Robot_R1.ltx"));
filetxt="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>"+"\n";
filetxt+="<externList>"+"\n";
filetxt+="    <data alias=\"io\" type=\"sio\" name=\"Server\" />"+"\n";
filetxt+="</externList>"+"\n";
writer11.write(filetxt);
writer11.close();
//fitxer del projecte R1
FileWriter writer12=new FileWriter(new File(destFN+"Robot_R1.pjx"));
filetxt=CreateProjectR1File(sourceDataLayer);
writer12.write(filetxt);
writer12.close();
//a partir d'aqui val per R2
destFN=fileBrowser.getSelectedFile().getPath();
File directory2 = new File(destFN+"/"+"Robot_R2");
directory2.mkdir();
destFN+="/"+"Robot_R2/";
//fitxer start per a R1
FileWriter writer13=new FileWriter(new File(destFN+"start.pgx"));
filetxt=CreateStartR2File(sourceDataLayer);
writer13.write(filetxt);
writer13.close();
//fitxer stop per a R1
FileWriter writer14=new FileWriter(new File(destFN+"stop.pgx"));
filetxt=CreateStopFile(sourceDataLayer);
writer14.write(filetxt);
writer14.close();
//fitxers de subprogrames de transició per a R2
for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
        //si la transicio te algun lloc de R2 vol dir que pertany a R2
        if (transitionR2(sourceDataLayer,TransitionNo)){
                FileWriter writer15=new FileWriter(new
                File(destFN+"transition_"+transitions[TransitionNo].getName()+".pgx"));
                filetxt=InitialFile("transition_"+transitions[TransitionNo].getName());
                filetxt+=ProgramCodeTran(sourceDataLayer,TransitionNo);
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer15.write(filetxt);
                writer15.close();
        }
}
//fitxers de subprogrames de lloc per a R2
for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
        if
        ((places[PlaceNo].getSubtype().equalsIgnoreCase("R2"))|(places[PlaceNo].getSubtyp
        e().equalsIgnoreCase("R2aux"))){
                FileWriter writer16=new FileWriter(new
                File(destFN+"place_"+places[PlaceNo].getName()+".pgx"));
                filetxt=InitialFile("place_"+places[PlaceNo].getName());
                filetxt+="begin"+"\n";
                filetxt+=places[PlaceNo].getCodeSnipet()+"\n";
```

```
                    for (int ArcNo=0; ArcNo<arcs.length; ArcNo++){
                        source=arcs[ArcNo].getId();
                        pos=source.indexOf(" to ");
                        if (pos==-1){
                            filetxt+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
                        else{
                            source=arcs[ArcNo].getId().substring(0,pos);
                            target=arcs[ArcNo].getId().substring(pos+4);}
                        if (places[PlaceNo].getName().equalsIgnoreCase(target)){
                            if (NexTransAux(sourceDataLayer,PlaceNo)){
                                filetxt+=CodeAux(sourceDataLayer,PlaceNo);}
                            filetxt+="end"+"\n";
                        }
                    }
                    filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                    filetxt+="    </program>"+"\n"+"</programList>";
                    writer16.write(filetxt);
                    writer16.close();
                }
            }
            //fitxer de dades per a R2
            FileWriter writer17=new FileWriter(new File(destFN+"Robot_R2.dtx"));
            filetxt=CreateVariableR2File(sourceDataLayer);
            writer17.write(filetxt);
            writer17.close();
            //crea el fitxer ltx
            FileWriter writer18=new FileWriter(new File(destFN+"Robot_R2.ltx"));
            filetxt="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>"+"\n";
            filetxt+="<externList>"+"\n";
            filetxt+="    <data alias=\"io\" type=\"sio\" name=\"Client\" />"+"\n";
            filetxt+="</externList>"+"\n";
            writer18.write(filetxt);
            writer18.close();
            //fitxer del projecte R2
            FileWriter writer19=new FileWriter(new File(destFN+"Robot_R2.pjx"));
            filetxt=CreateProjectR2File(sourceDataLayer);
            writer19.write(filetxt);
            writer19.close();
            //crea els fitxers relacionats amb els recursos compartits
            i=0;
            for (int placeNo=0; placeNo<places.length; placeNo++){
                //mira si hi ha recursos compartits i els compta
                if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                    i+=1;
                }
            }
            if (i>0){
                FileWriter writer20=new FileWriter(new File(destFN+"lectura.pgx"));
                //fitxer de lectura per a R2
                filetxt=InitialFile("lectura");
                filetxt+=lecturaR2();
                filetxt+="          </code>"+"\n"+"       </source>"+"\n";
                filetxt+="    </program>"+"\n"+"</programList>";
                writer20.write(filetxt);
                writer20.close();
                //fitxer d'escriputra per a R2
                FileWriter writer21=new FileWriter(new File(destFN+"escriptura.pgx"));
```

```
            filetxt=InitialFile("escriptura");
            filetxt+=escriptura_R2(sourceDataLayer);
            filetxt+="          </code>"+"\n"+"       </source>"+"\n";
            filetxt+="    </program>"+"\n"+"</programList>";
            writer21.write(filetxt);
            writer21.close();
            //fitxer de gestió de recursos compartits per a R2
            FileWriter writer22=new FileWriter(new File(destFN+"gestioRC.pgx"));
            filetxt=InitialFile("gestioRC");
            filetxt+=gestioRC_R2(sourceDataLayer);
            filetxt+="          </code>"+"\n"+"       </source>"+"\n";
            filetxt+="    </program>"+"\n"+"</programList>";
            writer22.write(filetxt);
            writer22.close();
            //fitxer de comunicació per a R2
            FileWriter writer23=new FileWriter(new File(destFN+"comunicacio.pgx"));
            filetxt=InitialFile("comunicacio");
            filetxt+="begin"+"\n";
            filetxt+=comunicacioR2();
            filetxt+="          </code>"+"\n"+"       </source>"+"\n";
            filetxt+="    </program>"+"\n"+"</programList>";
            writer23.write(filetxt);
            writer23.close();
            //fitxer d'exclusió mútua per a R2
            FileWriter writer24=new FileWriter(new File(destFN+"mutex.pgx"));
            filetxt="<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>"+"\n";
            filetxt+="<programList xmlns=\"ProgramNameSpace\">"+"\n";
            filetxt+="  <program name=\"mutex\" public=\"false\">"+"\n";
            filetxt+="    <description />"+"\n";
            filetxt+="    <paramSection>"+"\n";
            filetxt+="      <param name=\"bRC\" type=\"bool\" byVal=\"false\" />"+"\n";
            filetxt+="    </paramSection>"+"\n";
            filetxt+="    <localSection />"+"\n";
            filetxt+="    <source>"+"\n";
            filetxt+="      <code>"+"\n";
            filetxt+=mutex();
            filetxt+="          </code>"+"\n"+"       </source>"+"\n";
            filetxt+="    </program>"+"\n"+"</programList>";
            writer24.write(filetxt);
            writer24.close();
    }
} catch (Exception e) {
             System.out.println("Error saving file");
 }
}
};
};
```

B.2.3. Els mètodes que realitzen comprovacions

Els següents mètodes serveixen per comprovar les característiques dels elements de la xarxa.

```
/**
 * transitionR1 is true if TransitionNo doesn't have any R2 place in its forwards
 * and backwards incidence matrixes
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private boolean transitionR1(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean notrobat=true;
        while ((PlaceNo<forwardset.length) && (notrobat)){
                if
                ((places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[forwards
                et[PlaceNo]].getSubtype().equalsIgnoreCase("R2aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        PlaceNo=0;
        while ((PlaceNo<backwardset.length) && (notrobat)){
                if
                ((places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[backw
                ardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        return notrobat;
}

/**transitionR1Only is true if TransitionNo doesn't have any R2 or auxiliar
 * place in its forwards and backwards incidence matrixes
 * @param pnmlData
 * @return
 */
private boolean transitionR1Only(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean notrobat=true;
        while ((PlaceNo<forwardset.length) && (notrobat)){
        if
        ((places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[forwarset[PlaceN
        o]].getSubtype().equalsIgnoreCase("R2aux"))|transitionR1aux(pnmlData,TransitionNo)){
```

```
                notrobat=false;
        }
        PlaceNo+=1;
}
PlaceNo=0;
while ((PlaceNo<backwardset.length) && (notrobat)){
        if
        ((places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[backwardset[P
        laceNo]].getSubtype().equalsIgnoreCase("R2aux"))|transitionR1aux(pnmlData,TransitionNo)){
                notrobat=false;
        }
        PlaceNo+=1;
}
return notrobat;}

/**
 * transitionR2 is true if TransitionNo doesn't have any R1 place in its forwards
 * and backwards incidence matrixes
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private boolean transitionR2(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean notrobat=true;
        while ((PlaceNo<forwardset.length) && (notrobat)){
                if
                (places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1")|(places[forwardse
                t[PlaceNo]].getSubtype().equalsIgnoreCase("R1aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        while ((PlaceNo<backwardset.length) && (notrobat)){
                if
                (places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1")|(places[backwa
                rdset[PlaceNo]].getSubtype().equalsIgnoreCase("R1aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        return notrobat;
}

/**
 * return true if TransitionNo doesn't have any R1 or auxiliar place in
 * its forwards and backwards incidence matrixes
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private boolean transitionR2Only(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
```

```
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean notrobat=true;
        while ((PlaceNo<forwardset.length) && (notrobat)){
                if
                ((places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[forwards
                et[PlaceNo]].getSubtype().equalsIgnoreCase("R1aux"))|transitionR2aux(pnmlData,Tra
                nsitionNo)){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        PlaceNo=0;
        while ((PlaceNo<backwardset.length) && (notrobat)){
                if
                ((places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[backw
                ardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1aux"))|transitionR2aux(pnmlDat
                a,TransitionNo)){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        return notrobat;
}

/**
 * returns true if TransitionNo have a Shared Resource in its backwards set
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private boolean transitionRC(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean trobat=false;
        while ((PlaceNo<backwardset.length) && (!trobat)){
                if (places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("Shared
                Resource")){
                        trobat=true;
                }
                PlaceNo+=1;
        }
        return trobat;
}

/**
 * return true if TransitionNo belongs to R1aux only
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private boolean transitionR1aux(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
```

```
        boolean notrobat=true;

        while ((PlaceNo<forwardset.length) && (notrobat)){
                if
                ((places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[forwards
                et[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[forwardset[PlaceNo]].get
                Subtype().equalsIgnoreCase("R2aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        PlaceNo=0;
        while ((PlaceNo<backwardset.length) && (notrobat)){
                if
                ((places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[backw
                ardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[backwardset[PlaceN
                o]].getSubtype().equalsIgnoreCase("R2aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        return notrobat;
}

 /**
  * return true if TransitionNo belongs to R2aux only
  * @param pnmlData
  * @param TransitionNo
  * @return
  */
 private boolean transitionR2aux(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        int PlaceNo=0;
        boolean notrobat=true;
        while ((PlaceNo<forwardset.length) && (notrobat)){
                if
                ((places[forwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[forwards
                et[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[forwardset[PlaceNo]].get
                Subtype().equalsIgnoreCase("R1aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        PlaceNo=0;
        while ((PlaceNo<backwardset.length) && (notrobat)){
                if
                ((places[backwardset[PlaceNo]].getSubtype().equalsIgnoreCase("R1"))|(places[backw
                ardset[PlaceNo]].getSubtype().equalsIgnoreCase("R2"))|(places[backwardset[PlaceN
                o]].getSubtype().equalsIgnoreCase("R1aux"))){
                        notrobat=false;
                }
                PlaceNo+=1;
        }
        return notrobat;
}
```

```
/**
 * return a set of forward places for TransitionNo
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private int[] ForwardsPlaceSet(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[][] forwards=pnmlData.getForwardsIncidenceMatrix();
        ArrayList postsetArrayList = new ArrayList();
        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++) {
                if(forwards[PlaceNo][TransitionNo] != 0) {
                        postsetArrayList.add(new Integer(PlaceNo));
                }
        }
        int[] postset = new int[postsetArrayList.size()];
        for(int postsetPosition = 0 ; postsetPosition < postset.length ; postsetPosition++) {
                postset[postsetPosition] = ((Integer)postsetArrayList.get(postsetPosition)).intValue();
        }
 return postset;}

/**
 * return a set of backwards places for TransitionNo
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private int[] BackwardsPlaceSet(DataLayer pnmlData, int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        int[][] backwards=pnmlData.getBackwardsIncidenceMatrix();
        ArrayList postsetArrayList = new ArrayList();
        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++) {
                if(backwards[PlaceNo][TransitionNo] != 0) {
                        postsetArrayList.add(new Integer(PlaceNo));
                }
        }
        int[] postset = new int[postsetArrayList.size()];
        for(int postsetPosition = 0 ; postsetPosition < postset.length ; postsetPosition++) {
                postset[postsetPosition] = ((Integer)postsetArrayList.get(postsetPosition)).intValue();
        }
 return postset;}
```

B.2.4. Els mètodes que generen el codi

Els següents mètodes són realment els que generen el codi, utilitzant els mètodes de l'apartat anterior per fer comprovacions sobre els elements de la xarxa i escriure el text corresponent per cada part.

```
/**
* Generates code for R1
*/
//serveix per visualitzar el codi dins del modul
private String CompileCodeR1(DataLayer pnmlData){
	Place[] places=pnmlData.getPlaces();
	String j=new String();
	int PlaceNo=0;
	int Procedencia=1;
	boolean notrobat=true;
	boolean RC=false;
	//notrobat es true si no hi ha llocs de R1
	if (places!=null)
		while ((PlaceNo<places.length) && (notrobat)){
			if (places[PlaceNo].getSubtype().equalsIgnoreCase("R1")){
				notrobat=false;
			}
			PlaceNo+=1;
		}
	if (!notrobat){
		j="//Generador de Codi Val3 del robot R1"+"\n\n" +
		"//programa principal R1()"+"\n\n";
		//1-crea les inicialitzacions. En val3 fa falta declarar les variables
		j+="begin"+"\n";
		j+=InitializeCodeR1(pnmlData);
		j+="\n";
		//2-crea les crides a les tasques de gestió de recursos compartits
		j+=RcTaskCreateR1(pnmlData);
		//3-crea el bucle principal
		j+=PrincipalCodeR1(pnmlData,Procedencia);
		j+="end"+"\n"+"\n";
		//4-crea tots els subprogrames de transició i lloc
		j+=TransitionCodeR1(pnmlData);
		//Mira si hi ha recursos compartits i posa el valor a RC
		PlaceNo=0;
		while ((PlaceNo<places.length) && (!RC)){
			if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared Resource")){
				RC=true;
			}
			PlaceNo+=1;
		}
		if (RC){
			//5-crea el subprograma  auxiliar de comunicació
			j+="//Programa auxiliar comunicacio()"+"\n"+"begin"+"\n";
			j+=comunicacio();
			j+="\n";
			//6-crea el subprograma de gestió de recursos compartits
			j+="//Programa gestio dels recursos compartits gestioRC()"+"\n";
			j+=gestioRC(pnmlData);
			j+="\n";
			//7-crea el subprograma d'escriptura
			j+="//Programa escriptura del buffer_out escriptura()"+"\n";
			j+=escriptura(pnmlData);
			j+="\n";
			//8-crea el subprograma d'escriptura
			j+="//Programa lectura del buffer_in lectura()"+"\n";
			j+=lectura();
```

```
                        j+="\n";
                        //9-crea el subprograma d'exclusió mutua
                        j+="//Programa d'exlusió mutua mutex(bool& bRC)"+"\n";
                        j+=mutex();
                }
        }else{
                j="No hi ha llocs associats a R1";}
 return(j);
}

/**
 * Creates tokens' declarations for R1
 * @param pnmlData
 * @return
 */
private String InitializeCodeR1 (DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0;
        if (places!=null)
                //recorregut a tots els places
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si els llocs pertanyen a R1
                        if (!(places[placeNo].getSubtype().equalsIgnoreCase("R2"))&
                        !(places[placeNo].getSubtype().equalsIgnoreCase("R2aux"))&
                        !(places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                                //si es que si, posa les variables de cada lloc R1 amb el seu marcat
                                //com a valor
                                j=j+"
                                ntokens_"+places[placeNo].getName()+"="+places[placeNo].getInitia
                                lMarking()+"\n";}
                        //si es de tipus recurs compartit ha de declarar altres variables
                        else if (places[placeNo].getSubtype().equalsIgnoreCase("Shared
                        Resource")){
                                i+=1;
                                if (places[placeNo].getInitialMarking()>0){
                                        j=j+"  RC"+places[placeNo].getName()+"=0"+ "\n";
                                        j=j+"  bRC"+places[placeNo].getName()+"=false"+ "\n";
                                        j=j+"  Pin["+i+"]=0"+ "\n";
                                }
                                else {
                                        j=j+"  RC"+places[placeNo].getName()+"=2"+ "\n";
                                        j=j+"  bRC"+places[placeNo].getName()+"=false"+ "\n";
                                        j=j+"  Pin["+i+"]=2"+ "\n";
                                }
                        }
                }
        return j;}

/**
 * Creates the main loop for R1
 * @param pnmlData
 * @return
 */
private String PrincipalCodeR1 (DataLayer pnmlData,int Procedencia){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
```

```
        String j=new String();
        if (transitions!=null)
                j+="  while(true)"+"\n";
                //recorregut a totes les transicions
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transició té un recurs compartit, la posem al principi perquè en //cas
                        //d'elecció lliure tingui preferència. Genera el codi asociat
                        if
                        ((transitionR1Only(pnmlData,TransitionNo))&(transitionRC(pnmlData,Transiti
                        onNo))){
                                j+=CreateTransitionRCCode(pnmlData,TransitionNo,Procedencia);}
                }
                //si la transició només pertany a R1 (no R1aux) crea els bucles
                //i genera el codi associat
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        if
                        ((transitionR1Only(pnmlData,TransitionNo))&(!(transitionRC(pnmlData,Transit
                        ionNo)))){
                                j+=CreateTransitionCode(pnmlData,TransitionNo,Procedencia);
                        }
                }
        j+="  endWhile"+"\n";
        return j;}



/**
 * Creates routines for transition TransitionNo
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private String CreateTransitionCode(DataLayer pnmlData,int TransitionNo,int Procedencia){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        Arc[] arcs=pnmlData.getArcs();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        String j=new String();
        String source=new String();
        String target=new String();
        int i=0;
        int pos=0;
        j="    //Codi de la transició "+transitions[TransitionNo].getName()+"\n"+"    flag=0"+"\n";
        //per cada arc, troba el source i el target
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
                //guardem source i target de cada arc
                source=arcs[ArcNo].getId();
                pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
                source=arcs[ArcNo].getId().substring(0,pos);
                target=arcs[ArcNo].getId().substring(pos+4);
                //mira si el target es TransitionNo
                if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
```

```
                    if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                        if (Procedencia==1){
                            j+="    if
                            ntokens_"+source+"<"+arcs[ArcNo].getWeight()+"\n"+"
                            flag=1"+"\n"+"    endIf"+"\n";
                        }
                    else{
                        j+="    if ntokens_"+source+"&lt;"+arcs[ArcNo].getWeight()+"\n"+"
                        flag=1"+"\n"+"    endIf"+"\n";
                    }
            }
}}}
}
j+="    if flag==0"+"\n";
for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
source=arcs[ArcNo].getId();
pos=source.indexOf(" to ");
if (pos==-1){
        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc te com a source la transició
        if (transitions[TransitionNo].getName().equalsIgnoreCase(source)){
            //recurregut a places per trobar el lloc a que apunta
            for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if (places[PlaceNo].getName().equalsIgnoreCase(target)){
                    //si el target es un recurs compartit primer es posa
                    //el codi que només pertany a R2
                    if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                    Resource")){
                        if (transitionR2(pnmlData,TransitionNo)){
                            i+=1;
                            j+="     call mutex(bPout)"+"\n";
                            j+="     Pout["+i+"]=0"+"\n";
                            j+="     bPout=false"+"\n";}
                    }
                }
            }
        }
}
}
//  recorregut a tots els arcs
for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
//guardem source i target de cada arc
source=arcs[ArcNo].getId();
pos=source.indexOf(" to ");
if (pos==-1){
        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc te com a source la transicio
        if (transitions[TransitionNo].getName().equalsIgnoreCase(source)){
            //recurregut a places per trobar place target
            for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if (places[PlaceNo].getName().equalsIgnoreCase(target)){
```

```
                                        //si el target es un recurs compartit
                                        if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                        Resource")){
                                                //posem el codi que deixa lliure el recurs compartit
                                                //tan per R1 com per R2
                                                j+="     call
                                                mutex(bRC"+places[PlaceNo].getName()+")"+"\n";
                                                j+="     RC"+places[PlaceNo].getName()+"=0"+"\n";
                                                j+="     bRC"+places[PlaceNo].getName()+"=false"+"\n";
                                        }
                                }
                        }
                }
                }
                }
                //es fa la crida a la transició següent
                j+="     call transition_"+transitions[TransitionNo].getName()+"()"+"\n";
                j+="    endIf"+"\n";
                return j;}

        /**
         * Creates routines for transition TransitionNo if it has a RC
         * @param pnmlData
         * @param TransitionNo
         * @return
         */
        private String CreateTransitionRCCode(DataLayer pnmlData,int TransitionNo,int Procedencia){
                Place[] places=pnmlData.getPlaces();
                Transition[] transitions=pnmlData.getTransitions();
                Arc[] arcs=pnmlData.getArcs();
                int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
                int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
                String j=new String();
                String source=new String();
                String target=new String();
                int i=0;
                int i2=0;
                int i3=0;
                int i4=0;
                int pos=0;
                j="    //Codi de la transició "+transitions[TransitionNo].getName()+"\n"+"    flag=0"+"\n";
                //es fan varis bucles "for" independents degut a que hi pot haver més d'un RC
                //en la mateixa transició d'entrada
                //primer bucle:mira si els llocs d'entrada estan marcats
                for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
                source=arcs[ArcNo].getId();
                pos=source.indexOf(" to ");
                if (pos==-1){
                        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
                else{
                source=arcs[ArcNo].getId().substring(0,pos);
                target=arcs[ArcNo].getId().substring(pos+4);
                //mira si el target es TransitionNo
                if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                                if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                                        //mira si el source es diferent a RC
```

```
                                if (!(places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                Resource"))){
                                        if (Procedencia==1){
                                                j+="    if
                                                ntokens_"+source+"<"+arcs[ArcNo].getWeight()+"\n
                                                "+"     flag=1"+"\n"+"    endIf"+"\n";}
                                        else{
                                                j+="    if
                                                ntokens_"+source+"&lt;"+arcs[ArcNo].getWeight()+"
                                                \n"+"     flag=1"+"\n"+"    endIf"+"\n";}
                                }
                        }
                }
}}
}
//segon bucle:per a cada robot fa una acció diferent
for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
source=arcs[ArcNo].getId();
pos=source.indexOf(" to ");
if (pos==-1){
        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
else{
source=arcs[ArcNo].getId().substring(0,pos);
target=arcs[ArcNo].getId().substring(pos+4);
//mira si el target es TransitionNo
if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                        //mira si el source es igual a shared resource
                        if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                        Resource")){
                                if (transitionR1(pnmlData,TransitionNo)){
                                        //a R1 es criden els mutex de cada RC
                                        j+="    call mutex(bRC"+source+")"+"\n";}
                                else {
                                        //a R2 es demana el recurs compartit
                                        i2+=1;
                                        if (i2==1){
                                                j+="    if flag==0"+"\n";
                                        }
                                        j+="     call mutex(bPout)"+"\n";
                                        j+="     Pout["+i2+"]=2"+"\n";
                                        j+="     bPout=false"+"\n";
                                }
                        }
                }

        }
}}
}
//tercer bucle: per a cada robot fa una acció diferent
for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
```

```
            source=arcs[ArcNo].getId().substring(0,pos);
            target=arcs[ArcNo].getId().substring(pos+4);
            //mira si el target es TransitionNo
            if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                  for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                        if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                              //mira si el source es igual a shared resource
                              if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                              Resource")){
                                    if (transitionR1(pnmlData,TransitionNo)){
                                          //a R1 es crea la primera part del "if"
                                          i+=1;
                                          if (i==1){
                                                j+="    if flag==0";
                                          }
                                          j+=" and RC"+source+"==0";
                                          }
                                          else {
                                          //a R2 es criden les exlusions
                                          //mutues
                                          j+="     call
                                          mutex(bRC"+source+")"+"\n";
                                          }
                                    }
                              }
                        }
                  }
            }
      }
      //quart bucle: per a cada robot fa una acció diferent
      for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
            source=arcs[ArcNo].getId();
            pos=source.indexOf(" to ");
            if (pos==-1){
                  j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
            else{
            source=arcs[ArcNo].getId().substring(0,pos);
            target=arcs[ArcNo].getId().substring(pos+4);
            //mira si el target es TransitionNo
            if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                  for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                  if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                              //mira si el source es igual a shared resource
                              if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                              Resource")){
                                    i3+=1;
                                    if (transitionR1(pnmlData,TransitionNo)){
                                          //a R1 es crea el cos del if i es crida el
                                          //subprograma de transició
                                          j+="\n";
                                          j+="     RC"+source+"=1";
                                          j+="\n"+"     bRC"+source+"=false";
                                          if (i3==i){
                                                j+="\n"+"     call
                                                transition_"+transitions[TransitionN
                                                o].getName()+"()"+"\n";
                                          }
```

```
                                                }
                                                else {
                                                        //a R2 es crea la primera part del
                                                        //"if"
                                                        i4+=1;
                                                        if (i4==1){
                                                                j+="     if";
                                                        }
                                                        j+=" RC"+source+"==2";
                                                        if (i4==i2){
                                                                j+="\n";
                                                        }
                                                        else {
                                                                j+=" and";
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
}
//cinquè bucle:només per a R2
i4=0;
for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //mira si el target es TransitionNo
        if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                                //mira si el source es igual a shared resource
                                if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                Resource")){
                                        if (transitionR1(pnmlData,TransitionNo)){
                                                //no fa res
                                        }
                                        else {
                                                //a R2 es crea el cos del if i es crida el
                                                //subprograma de transició
                                                i4+=1;
                                                j+="      bRC"+source+"=false"+"\n";
                                                if (i4==i2){
                                                        j+="      call
                                                        transition_"+transitions[TransitionN
                                                        o].getName()+"()"+"\n";
                                                }
                                        }
                                }
                        }
                }
        }
```

```
        }
        }
        //sisè bucle:en cas de no poder fer servir el recurs alliberem la variable
        //de l'exclusió mutua
        i3=0;
        i4=0;
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
                source=arcs[ArcNo].getId();
                pos=source.indexOf(" to ");
                if (pos==-1){
                        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
                else{
                source=arcs[ArcNo].getId().substring(0,pos);
                target=arcs[ArcNo].getId().substring(pos+4);
                //mira si el target es TransitionNo
                if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                                if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                                        //mira si el source es igual a shared resource
                                        if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                        Resource")){
                                                //es crea el cos de else
                                                if (transitionR1(pnmlData,TransitionNo)){
                                                        i3+=1;
                                                        if (i3==1){
                                                                j+="    else"+"\n";
                                                        }
                                                        j+="     bRC"+source+"=false"+"\n";
                                                }
                                                else {
                                                        i4+=1;
                                                        if (i4==1){
                                                                j+="     else"+"\n";
                                                        }
                                                        j+="      bRC"+source+"=false"+"\n";
                                                        if (i4==i2){
                                                                j+="     endIf"+"\n";
                                                        }
                                                }
                                        }
                                }
                        }
                }
                }
}
j+="    endIf"+"\n";
return j;}

/**
 * Creates subprograms for each transition and place for R1
 * @param pnmlData
 * @return
 */
private String TransitionCodeR1 (DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        String j=new String();
```

```
        if (transitions!=null)
                //recorregut a totes les transicions
                j+="//Subprogrames de les transicions"+"\n";
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transicio te algun lloc de R1 vol dir que pertany a R1
                        //i genera els programes de les transicions
                        if (transitionR1(pnmlData,TransitionNo)){
                                j+=ProgramCodeTran(pnmlData,TransitionNo)+"\n";
                        }
                }
                j+="//Subprogrames dels llocs"+"\n";
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transicio te algun lloc de R1 vol dir que pertany a R1
                        //i genera els programes de llocs
                        if (transitionR1(pnmlData,TransitionNo)){
                                j+=ProgramCodePlace(pnmlData,TransitionNo)+"\n";
                        }
                }

        return j;}

/**
 * Creates code for transitions' subprograms
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private String ProgramCodeTran(DataLayer pnmlData,int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        Arc[] arcs=pnmlData.getArcs();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        String j=new String();
        String source=new String();
        String target=new String();
        int pos=0;
        j="begin"+"\n";
        //  recorregut a tots els arcs
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        //guardem source i target de cada arc
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc te com a target la transició
        if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                //recorregut per trobar el lloc source de l'arc
                for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                        if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                                //si el lloc és del tipus recurs compartit
                                if (!(places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                Resource"))){
                                        //actualitzem les variables de marcat
```

```
                                        j+="  ntokens_"+source+"="+"ntokens_"+source+"-
                                        "+arcs[ArcNo].getWeight()+"\n";
                                }
                        }
                }
        }}
        }
        //fem les crides als subprgrames de llocs
        j+=CallPlaces(pnmlData,TransitionNo);
        j+="end"+"\n";
        return j;}

/**
 * Creates calls to all target places
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private String CallPlaces(DataLayer pnmlData,int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        Arc[] arcs=pnmlData.getArcs();
        int[] forwardset=ForwardsPlaceSet(pnmlData,TransitionNo);
        int[] backwardset=BackwardsPlaceSet(pnmlData,TransitionNo);
        String j=new String();
        String source=new String();
        String target=new String();
        int pos=0;
        //  recorregut a tots els arcs
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        //guardem source i target de cada arc
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc te com a source la transició
        if (transitions[TransitionNo].getName().equalsIgnoreCase(source)){
                //recurregut a places per trobar place target
                for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                        if (places[PlaceNo].getName().equalsIgnoreCase(target)){
                                //si el target és un lloc auxiliar, es creen les tasques en paral·lel
                                if
                                ((places[PlaceNo].getSubtype().equalsIgnoreCase("R1aux"))|(places
                                [PlaceNo].getSubtype().equalsIgnoreCase("R2aux"))){
                                        if (transitionR1Only(pnmlData,TransitionNo)){
                                                j+="  taskCreate
                                                \"auxR1\",10,place_"+places[PlaceNo].getName()+"
                                                ()"+"\n";}
                                        else if (transitionR2Only(pnmlData,TransitionNo)){
                                                j+="  taskCreate
                                                \"auxR2\",10,place_"+places[PlaceNo].getName()+"
                                                ()"+"\n";}
                                        else{
```

```
                                                j+="  call
                                                place_"+places[PlaceNo].getName()+"()"+"\n";
                                                j+="
                                                ntokens_"+places[PlaceNo].getName()+"=ntokens_
                                                "+places[PlaceNo].getName()+"+1"+"\n";}
                                        }
                                        //sinó, es fan les crides als subprogrames de lloc i s'actualitza el
                                        //marcat
                                        else if (!(places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                        Resource"))){
                                                j+="  call place_"+places[PlaceNo].getName()+"()"+"\n";
                                                j+="
                                                ntokens_"+places[PlaceNo].getName()+"=ntokens_"+places
                                                [PlaceNo].getName()+"+1"+"\n";}
                                }
                        }
                }}
                }
 return j;}

/**
 * Creates code for places' subprograms
 * @param pnmlData
 * @param TransitionNo
 * @return
 */
private String ProgramCodePlace(DataLayer pnmlData,int TransitionNo){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        Arc[] arcs=pnmlData.getArcs();
        String j=new String();
        String source=new String();
        String target=new String();
        int pos=0;
        //  recorregut a tots els arcs
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        //guardem source i target de cada arc
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc té com a source la transició
        if (transitions[TransitionNo].getName().equalsIgnoreCase(source)){
                //recorregut per trobar el lloc target de l'arc
                for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                        if (places[PlaceNo].getName().equalsIgnoreCase(target)){
                                //si el lloc és del tipus recurs compartit
                                if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared
                                Resource")){
                                        //no fa res
                                }
                                //si el lloc es normal crea el subprograma
                                else {
```

```
                                        j+="//Programa de
                                        place_"+places[PlaceNo].getName()+"()"+"\n"+"begin"+"\n";
                                        //posa el codi del lloc
                                        j+=places[PlaceNo].getCodeSnipet()+"\n";
                                        //si la següent transició pertany a la branca auxiliar, s'ha de
                                        //fer la crida
                                        //dins del subprograma del lloc anterior
                                        if (NexTransAux(pnmlData,PlaceNo)){
                                                j+=CodeAux(pnmlData,PlaceNo);}
                                        j+="end"+"\n";}
                                }
                        }}}
                }
                return j;}

        /**
         * Returns true if next transition is auxiliar
         * @param pnmlData
         * @param PlaceNo
         * @return
         */
         private boolean NexTransAux(DataLayer pnmlData, int PlaceNo){
                Place[] places=pnmlData.getPlaces();
                Transition[] transitions=pnmlData.getTransitions();
                Arc[] arcs=pnmlData.getArcs();
                String j=new String();
                String source=new String();
                String target=new String();
                boolean trobat=false;
                int pos=0;
                //  recorregut a tots els arcs
                for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
                //guardem source i target de cada arc
                source=arcs[ArcNo].getId();
                pos=source.indexOf(" to ");
                if (pos==-1){
                        j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
                else{
                source=arcs[ArcNo].getId().substring(0,pos);
                target=arcs[ArcNo].getId().substring(pos+4);
                //si l'arc te com a source el place
                if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                        //recorregut per trobar la transició target de l'arc
                        for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                                if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                                        //si la transicio és només auxiliar
                                        if
                                        ((transitionR1aux(pnmlData,TransitionNo))|(transitionR2aux(pnmlDat
                                        a,TransitionNo))){
                                                trobat=true;}
                                }
                        }}
                }
                }
                return trobat;}
```

```
/**
 * Creates code for auxiliar places inside an auxiliar brench
 * @param pnmlData
 * @param PlaceNo
 * @return
 */
private String CodeAux(DataLayer pnmlData,int PlaceNo){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        Arc[] arcs=pnmlData.getArcs();
        String j=new String();
        String source=new String();
        String target=new String();
        boolean trobat=false;
        int pos=0;
        //  recorregut a tots els arcs
        for (int ArcNo=0; ArcNo<arcs.length; ArcNo++) {
        //guardem source i target de cada arc
        source=arcs[ArcNo].getId();
        pos=source.indexOf(" to ");
        if (pos==-1){
                j+="ERROR EN LA XARXA: TIPUS 2"+"\n";}
        else{
        source=arcs[ArcNo].getId().substring(0,pos);
        target=arcs[ArcNo].getId().substring(pos+4);
        //si l'arc té com a source el place
        if (places[PlaceNo].getName().equalsIgnoreCase(source)){
                //recorregut per trobar la transició target de l'arc
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        if (transitions[TransitionNo].getName().equalsIgnoreCase(target)){
                                //si la transicio es només auxiliar
                                if
                                ((transitionR1aux(pnmlData,TransitionNo))|(transitionR2aux(pnmlDat
                                a,TransitionNo))){
                                        //s'actualitza el marcat dels llocs i es crida la següent
                                        //transició
                                        j+="ntokens_"+places[PlaceNo].getName()+"="+"ntokens_"+
                                        places[PlaceNo].getName()+"+"+arcs[ArcNo].getWeight()+"\
                                        n";
                                        j+="call
                                        transition_"+transitions[TransitionNo].getName()+"()"+"\n";}
                        }
                }
        }}
        }
        return j;}

/**
 * Creates initialization for RC tascs for R1
 * @param pnmlData
 * @return
 */
private String RcTaskCreateR1(DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0; //conta el numero de recursos compartits
        if (places!=null)
```

```
		//recorregut a tots els places
		for (int placeNo=0; placeNo<places.length; placeNo++){
			//mira si es igual a recurs compartit
			if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
				i+=1;
				//crea el taskcreates i asigna les variables d'entrada/sortida
				if (i==1){
					j+="  sioLink(buffer_in, io:Server)"+"\n";
					j+="  taskCreate \"comunicacio\",20,comunicacio()"+"\n";
					}
			}
		}
 j+="\n";
	return j;}

/**
 * Creates initialization for RC tascs for R2
 * @param pnmlData
 * @return
 */
private String RcTaskCreateR2(DataLayer pnmlData){
	Place[] places=pnmlData.getPlaces();
	String j=new String();
	 int i=0; //conta el numero de recursos compartits
	if (places!=null)
		//recorregut a tots els places
		for (int placeNo=0; placeNo<places.length; placeNo++){
			//mira si es igual a recurs compartit
			if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
				i+=1;
				//crea el taskcreates i asigna les variables d'entrada/sortida
				if (i==1){
					j+="  sioLink(buffer_out, io:Client)"+"\n";
					j+="  taskCreate \"comunicacio\",20,comunicacio()"+"\n";
				}
			}
		}
 j+="\n";

	return j;}

/**
 * Creates program comunicacio for robot R1
 * @param pnmlData
 * @return
 */
private String comunicacio(){
 String j=new String();
		j+="  while(true)"+"\n";
		j+="    call lectura()"+"\n";
		j+="    call gestioRC()"+"\n";
		j+="    call escriptura()"+"\n";
		j+="  endWhile"+"\n"+"end"+"\n";
	return j;}
```

```
/**
 * Creates program comunicacio for robot R2
 * @param pnmlData
 * @return
 */
private String comunicacioR2(){
 String j=new String();
                j+="  while(true)"+"\n";
                j+="    call escriptura()"+"\n";
                j+="    call lectura()"+"\n";
                j+="    call gestioRC()"+"\n";
                j+="  endWhile"+"\n"+"end"+"\n";
        return j;}

/**
 * creates program lectura for R1
 * @param pnmlData
 * @return
 */
private String lectura(){
        String j=new String();
        j+="begin"+"\n"+"  sioLink(buffer_in,io:Server)"+"\n";
        j+="  buffer_server=buffer_in"+"\n";
        j+="  putln(buffer_server)"+"\n";
        j+="  cont=0"+"\n";
        j+="  do"+"\n"+"    buffer_server=toNum(buffer_server,Pin[cont],cond)"+"\n";
        j+="    cont=cont+1"+"\n";
        j+="  until(cond != true)"+"\n"+"end"+"\n";
        return j;}

/**
 * creates program lectura for R2
 * @param pnmlData
 * @return
 */
private String lecturaR2(){
        String j=new String();
        j+="begin"+"\n"+"  sioLink(buffer_in,io:Client)"+"\n";
        j+="  buffer_server=buffer_in"+"\n";
        j+="  putln(buffer_server)"+"\n";
        j+="  cont=0"+"\n";
        j+="  do"+"\n"+"    buffer_server=toNum(buffer_server,Pin[cont],cond)"+"\n";
        j+="    cont=cont+1"+"\n";
        j+="  until(cond != true)"+"\n"+"end"+"\n";
        return j;}

/**
 * Creates program escriptura for R1
 * @param pnmlData
 * @return
 */
private String escriptura(DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
         if (places!=null)
                j+="begin"+"\n"+"  SRC0=toString(\"1\",9)"+"\n";
                //recorregut a tots els places
```

```
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                //crea els strings q formaran tot el buffer_client
                                j+="  call mutex(bRC"+places[placeNo].getName()+")"+"\n";
                                j+="
                                SRC"+places[placeNo].getName()+"=toString(\"1\",RC"+places[place
                                No].getName()+")"+"\n";
                                j+="  bRC"+places[placeNo].getName()+"=false"+"\n";
                        }
                }
                j+="  sioLink(buffer_out,io:Server)"+"\n";
                j+="  buffer_client=SRC0";
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                //concatena l'string buffer_client
                                        j+="+\" \"+SRC"+places[placeNo].getName();
                        }
                }
                j+="\n";
                j+="  buffer_out=buffer_client"+"\n";
                j+="  putln(buffer_client)"+"\n"+"end"+"\n";
        return j;}

/**
 * Creates program escriptura for R2
 * @param pnmlData
 * @return
 */
private String escriptura_R2(DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0;
        if (places!=null)
                //recorregut a tots els places
                j+="begin"+"\n"+"  SRC0=toString(\"1\",9)"+"\n";
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                //crea els strings q formaran tot el buffer_client
                                if (i==1){
                                        j+="  call mutex(bPout)"+"\n";
                                }
                                j+="
                                SRC"+places[placeNo].getName()+"=toString(\"1\",Pout["+i+"])"+"\n;

                        }
                }
                j+="  bPout=false"+"\n";
                j+="  buffer_client=SRC0";
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                //concatena l'string buffer_client
                                j+="+\" \"+SRC"+places[placeNo].getName();
```

```
                                        }
                                }
        j+="\n";
        j+="  sioLink(buffer_out,io:Client)"+"\n";
        j+="  buffer_out=buffer_client"+"\n"+"end"+"\n";
        return j;}

/**
 * creates program gestioRC for R1
 * @param pnmlData
 * @return
 */
private String gestioRC(DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0;
        if (places!=null)
                //recorregut a tots els places
                j+="begin"+"\n";
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                //crea els bucles de cada recurs
                                j+="  call mutex(bRC"+places[placeNo].getName()+")"+"\n";
                                j+="  if RC"+places[placeNo].getName()+"==0 and
                                Pin["+i+"]==2"+"\n";
                                j+="    RC"+places[placeNo].getName()+"=2"+"\n";
                                j+="  else"+"\n";
                                j+="    if RC"+places[placeNo].getName()+"==2 and
                                Pin["+i+"]==0"+"\n";
                                j+="      RC"+places[placeNo].getName()+"=0"+"\n";
                                j+="    endIf"+"\n"+"  endIf"+"\n";
                                j+="  bRC"+places[placeNo].getName()+"=false"+"\n";
                        }
                }
        j+="end"+"\n";
        return j;}

/**
 * creates program gestioRC for R2
 * @param pnmlData
 * @return
 */
private String gestioRC_R2(DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0;
        if (places!=null)
                //recorregut a tots els places
                j+="begin"+"\n";
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es igual a recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                //crea els bucles de cada recurs
                                j+="  call mutex(bRC"+places[placeNo].getName()+")"+"\n";
```

```
                                        j+="  RC"+places[placeNo].getName()+"=Pin["+i+"]"+"\n";
                                        j+="  bRC"+places[placeNo].getName()+"=false"+"\n";
                                }
                        }
                j+="end"+"\n";
                return j;}

        /**
         * creates program for mutual exclusion
         * @return
         */
        private String mutex(){
                String j=new String();
                j+="begin"+"\n";
                j+="//la asignació de recurs (bRC) a true s'ha de fer a la mateixa linea de prova"+"\n";
                j+="  wait((bRC==false) and (bRC=true))"+"\n";
                j+="end"+"\n";
                return j;}

        /**
         * Generates code for R2, if any
         * @param pnmlData
         * @return
         */
       private String CompileCodeR2(DataLayer pnmlData){
                Place[] places=pnmlData.getPlaces();
                Transition[] transitions=pnmlData.getTransitions();
                String j=new String();
                int PlaceNo=0;
                int Procedencia=1;
                boolean notrobat=true;
                boolean RC=false;
                if (places!=null)
                        while ((PlaceNo<places.length) && (notrobat)){
                                if (places[PlaceNo].getSubtype().equalsIgnoreCase("R2")){
                                notrobat=false;}
                        PlaceNo+=1;
                        }
          if (!notrobat){
                j="//Generador de Codi Val3 del robot R2"+"\n\n" +
                        "//programa principal R2"+"\n\n";
                        //1-crea les inicialitzacions
                        j+="begin"+"\n";
                        j+=InitializeCodeR2(pnmlData);
                        //2-crea les crides a les tasques de comunicació
                        j+=RcTaskCreateR2(pnmlData);
                        //3-crea el bucle principal
                        j+=PrincipalCodeR2(pnmlData,Procedencia);
                        j+="end"+"\n"+"\n";
                        //4-crea tots els subprogrames de llocs i transicions
                        j+=TransitionCodeR2(pnmlData);;
                        //Mira si hi ha recursos compartits i guarda el valor a RC
                        PlaceNo=0;
                        while ((PlaceNo<places.length) && (!RC)){
                                if (places[PlaceNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                RC=true;}
                        PlaceNo+=1;
```

```
                }
                if (RC){
                        //5-crea el subprograma  auxiliar de comunicació
                        j+="//Programa auxiliar comunicacio()"+"\n"+"begin"+"\n";
                        j+=comunicacioR2();
                        j+="\n";
                        //6-crea el subprograma de gestió de recursos compartits
                        j+="//Programa gestio dels recursos compartits gestioRC()"+"\n";
                        j+=gestioRC_R2(pnmlData);
                        j+="\n";
                        //7-crea el subprograma d'escriptura
                        j+="//Programa escriptura del buffer_out escriptura()"+"\n";
                        j+=escriptura_R2(pnmlData);
                        j+="\n";
                        //8-crea el subprograma de lectura
                        j+="//Programa lectura del buffer_in lectura()"+"\n";
                        j+=lecturaR2();
                        j+="\n";
                        //9-crea el subprograma d'exclusió mutua
                        j+="//Programa d'exlusió mutua mutex(bool& bRC)"+"\n";
                        j+=mutex();
                }
        }
        else{
                j="No hi ha llocs associats a R2";
        }
 return(j);}

/**
 * Creates tokens' declarations for R2
 * @param pnmlData
 * @return
 */
private String InitializeCodeR2 (DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        String j=new String();
        int i=0;
        if (places!=null)
                //recorregut a tots els places
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si son de R2
                        if (!(places[placeNo].getSubtype().equalsIgnoreCase("R1"))&
                        !(places[placeNo].getSubtype().equalsIgnoreCase("R1aux"))&
                        !(places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                                //si es que si, posa les variables de cada lloc R2 amb el seu marcat
                                //com a valor
                                j=j+"
                                ntokens_"+places[placeNo].getName()+"="+places[placeNo].getInitia
                                lMarking()+ "\n";}
                        //si es de tipus recurs compartit ha de declarar altres variables
                        else if (places[placeNo].getSubtype().equalsIgnoreCase("Shared
                        Resource")){
                                if (places[placeNo].getInitialMarking()>0){
                                        j=j+"  RC"+places[placeNo].getName()+"=0"+"\n";
                                        j=j+"  bRC"+places[placeNo].getName()+"=false"+"\n";
                                }
                                else {
```

```
                                        j=j+"  RC"+places[placeNo].getName()+"=1"+"\n";
                                        j=j+"  bRC"+places[placeNo].getName()+"=false"+"\n";
                                }
                        }
                }
        j+="\n";
        return j;}


/**
 * Creates the main loop for R2
 * @param pnmlData
 * @return
 */
private String PrincipalCodeR2 (DataLayer pnmlData,int Procedencia){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        String j=new String();
        if (transitions!=null)
                j+="  while(true)"+"\n";
                //  recorregut a totes les transicions
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transició te un recurs compartit, la posem al principi perquè en cas
                        //d'elecció lliure tingui preferència
                        if
                        ((transitionR2Only(pnmlData,TransitionNo))&(transitionRC(pnmlData,Transiti
                        onNo))){
                                j+=CreateTransitionRCCode(pnmlData,TransitionNo,Procedencia);}
                }
                //si la transició només pertany a R2 (no R2aux) crea els bucles
                //i genera el codi associat
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        if
                        ((transitionR2Only(pnmlData,TransitionNo))&(!(transitionRC(pnmlData,Transit
                        ionNo)))){
                                j+=CreateTransitionCode(pnmlData,TransitionNo,Procedencia);
                        }
                }
                j+="  endWhile"+"\n";
        return j;}


private String TransitionCodeR2 (DataLayer pnmlData){
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        String j=new String();
        if (transitions!=null)
                //recorregut a totes les transicions
                j+="//Subprogrames de les transicions"+"\n";
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transició te algun lloc de R2 vol dir que pertany a R2
                        //i genera els subprogrames de transicions
                        if (transitionR2(pnmlData,TransitionNo)){
                                j+=ProgramCodeTran(pnmlData,TransitionNo)+"\n";
                        }
                }
                j+="//Subprogrames dels llocs"+"\n";
```

```
                for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                        //si la transició té algun lloc de R2 vol dir que pertany a R2
                        //i genera els programs de llocs
                        if (transitionR2(pnmlData,TransitionNo)){
                                j+=ProgramCodePlace(pnmlData,TransitionNo)+"\n";
                        }
                }
        return j;}
```

B.2.5. Els mètodes per a la creació de projectes Val3

```
/**
 * Creates start file for R1
 * @param sourceDataLayer
 * @param destFN
 * @return
 */
private String CreateStartR1File(DataLayer sourceDataLayer){
        String j=new String();
        int Procedencia=2;
        j=InitialFile("start");
        j+="begin"+"\n";
        j+=InitializeCodeR1(sourceDataLayer);
        j+="\n";
        j+=RcTaskCreateR1(sourceDataLayer);
        j+=PrincipalCodeR1(sourceDataLayer,Procedencia);
        j+="end"+"\n";
        j+="        </code>"+"\n"+"      </source>"+"\n";
        j+="   </program>"+"\n"+"</programList>";
        return j;}

/**
 * Creates start file for R2
 * @param sourceDataLayer
 * @param destFN
 * @return
 */
private String CreateStartR2File(DataLayer sourceDataLayer){
        String j=new String();
        int Procedencia=2;
        j=InitialFile("start");
        j+="begin"+"\n";
        j+=InitializeCodeR2(sourceDataLayer);
        j+="\n";
        j+=RcTaskCreateR2(sourceDataLayer);
        j+=PrincipalCodeR2(sourceDataLayer,Procedencia);
        j+="end"+"\n";
        j+="        </code>"+"\n"+"      </source>"+"\n";
        j+="   </program>"+"\n"+"</programList>";
        return j;}

/**
 * Creates stop file
 * @param sourceDatalayer
```

```
 * @return
 */
private String CreateStopFile(DataLayer sourceDatalayer){
        String j=new String();
        j=InitialFile("stop");
        j+="begin"+"\n";
        j+="  popUpMsg(&quot;Pending movement commands have been canceled&quot;)"+"\n";
        j+="  taskKill(\"comunicacio\")"+"\n";
        j+="  resetMotion()"+"\n";
        j+="end"+"\n";
        j+="          </code>"+"\n"+"       </source>"+"\n";
        j+="    </program>"+"\n"+"</programList>";
        return j;}

/**
 * Creates the initial code for file "name"
 * @param name
 * @return
 */
private String InitialFile(String name){
        String j=new String();
        j="<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>"+"\n";
        j+="<programList xmlns=\"ProgramNameSpace\" >"+"\n";
        j+="    <program name=\""+name+"\" public=\"false\" >"+"\n";
        j+="       <description></description>"+"\n";
        j+="       <paramSection>"+"\n";
        j+="       </paramSection>"+"\n";
        j+="       <localSection>"+"\n";
        j+="       </localSection>"+"\n";
        j+="       <source>"+"\n";
        j+="          <code>"+"\n";
        return j;}

/**
 * Creates data file for R1
 * @param pnmlData
 * @return
 */
private String CreateVariableR1File(DataLayer pnmlData){
        String j=new String();
        int i=0;
        Place[] places=pnmlData.getPlaces();
        j="<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>"+"\n";
        j+="<dataList xmlns=\"DataNameSpace\" >"+"\n";
        j+="  <aioSection>"+"\n"+"  </aioSection>"+"\n"+"  <boolSection>"+"\n";
        for (int placeNo=0; placeNo<places.length; placeNo++){
                //mira si es igual a recurs compartit
                if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                        i+=1;
                        j+="    <bool name=\"bRC"+places[placeNo].getName()+"\" public=\"false\"
                        privilege=\"0\" >"+"\n";
                        j+="      <valueBool index=\"0\" value=\"false\" />"+"\n"+"    </bool>"+"\n";
                }
        }
        if (i>0){
                j+="    <bool name=\"cond\" public=\"false\" privilege=\"0\" >"+"\n";
                j+="      <valueBool index=\"0\" value=\"false\" />"+"\n"+"    </bool>"+"\n";
```

```
}
i=0;
j+="  </boolSection>"+"\n"+"  <configSection>"+"\n"+"  </configSection>"+"\n";
j+="  <dioSection>"+"\n";
j+="  </dioSection>"+"\n"+"  <frameSection>"+"\n"+"  </frameSection>"+"\n";
j+="  <jointSection>"+"\n"+"  </jointSection>"+"\n"+"  <mdescSection>"+"\n";
j+="  </mdescSection>"+"\n"+"  <numSection>"+"\n";
j+="    <num name=\"flag\" public=\"false\" privilege=\"0\" >"+"\n";
j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
for (int placeNo=0; placeNo<places.length; placeNo++){
        //mira si son diferents de R2 i de recurs compartit
        if (!(places[placeNo].getSubtype().equalsIgnoreCase("R2"))&
        !(places[placeNo].getSubtype().equalsIgnoreCase("R2aux"))&
        !(places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                //si es que si, declara les variables de cada lloc R1 amb el seu marcat com a
                valor
                j+="    <num name=\"ntokens_"+places[placeNo].getName()+"\"
                public=\"false\" privilege=\"0\" >"+"\n";
                j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";}
                //si es de tipus recurs compartit ha de declarar altres variables
        else if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                i+=1;
                j+="    <num name=\"RC"+places[placeNo].getName()+"\" public=\"false\"
                privilege=\"0\" >"+"\n";
                j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
        }
}
//variables que només es creen en cas de recurs compartit
if (i>0){
        j+="    <num name=\"cont\" public=\"false\" privilege=\"0\" >"+"\n";
        j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
        j+="    <num name=\"Pin\" public=\"false\" privilege=\"0\" >"+"\n";
        for (int z=0; z<i+2; z++){
                j+="      <valueNum index=\""+z+"\" value=\"0\" />"+"\n";
        }
        j+="    </num>"+"\n";}
j+="  </numSection>"+"\n"+"  <pointSection>"+"\n"+"  </pointSection>"+"\n";
j+="  <sioSection>"+"\n";
if (places!=null)
        i=0;
        //recorregut a tots els places
        for (int placeNo=0; placeNo<places.length; placeNo++){
                //mira si es recurs compartit
                if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                        i+=1;
                        if (i==1){
                                j+="    <sio name=\"buffer_in\" public=\"false\" privilege=\"0\"
                                size=\"1\" />"+"\n";
                                j+="    <sio name=\"buffer_out\" public=\"false\"
                                privilege=\"0\" size=\"1\" />"+"\n";
                        }
                }
        }
j+="  </sioSection>"+"\n"+"  <stringSection>"+"\n";
if (places!=null)
        i=0;
        //recorregut a tots els places
```

```
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                j+="    <string name=\"SRC"+places[placeNo].getName()+"\"
                                public=\"false\" privilege=\"0\" >"+"\n";
                                j+="      <valueString index=\"0\" value=\"0\" />"+"\n"+"
                                </string>"+"\n";
                                if (i==1){
                                        j+="    <string name=\"buffer_server\" public=\"false\"
                                        privilege=\"0\" >"+"\n";
                                j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                </string>"+"\n";
                                j+="    <string name=\"buffer_client\" public=\"false\" privilege=\"0\"
                                >"+"\n";
                                j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                </string>"+"\n";
                                j+="    <string name=\"SRC0\" public=\"false\" privilege=\"0\" >"+"\n";
                                j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                </string>"+"\n";
                                }
                        }
                }
        j+="  </stringSection>"+"\n"+"  <toolSection>"+"\n";
        j+="    <tool name=\"flange\" public=\"false\" privilege=\"0\" >"+"\n";
        j+="      <tFather alias=\"\" name=\"\" fatherIndex=\"0\" />"+"\n";
        j+="      <valueTool index=\"0\" >"+"\n";
        j+="        <ttValue x=\"0\" y=\"0\" z=\"0\" rx=\"0\" ry=\"0\" rz=\"0\" />"+"\n";
        j+="        <io alias=\"io\" name=\"valve1\" ioIndex=\"0\" open=\"0\" close=\"0\" />"+"\n";
        j+="      </valueTool>"+"\n"+"    </tool>"+"\n";
        j+="  </toolSection>"+"\n";
        j+="  <trsfSection>"+"\n"+"  </trsfSection>"+"\n"+"</dataList>"+"\n";
        return j;}

/**
 * Creates file project for R1
 * @param pnmlData
 * @return
 */
private String CreateProjectR1File(DataLayer pnmlData){
        String j=new String();
        int i=0;
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        j="<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>"+"\n";
        j+="<project xmlns=\"ProjectNameSpace\" >"+"\n";
        j+="    <parameters version=\"~VAL3:s5.0.1~Motion:s5.0.1\" stackSize=\"5000\"
        millimeterUnit=\"true\" />"+"\n";
        j+="    <programSection>"+"\n";
        //fa les crides a les subrutines de llocs i de recurs compartit
        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if
                ((places[PlaceNo].getSubtype().equalsIgnoreCase("R1"))|(places[PlaceNo].getSubtyp
                e().equalsIgnoreCase("R1aux"))){
                        j+="        <program file=\""+"place_"+places[PlaceNo].getName()+".pgx\"
                        />"+"\n";
                }
```

```
                else if((places[PlaceNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                        i+=1;
                        if (i==1){
                                j+="       <program file=\""+"comunicacio.pgx\" />"+"\n";
                                j+="       <program file=\""+"lectura.pgx\" />"+"\n";
                                j+="       <program file=\""+"escriptura.pgx\" />"+"\n";
                                j+="       <program file=\""+"gestioRC.pgx\" />"+"\n";
                                j+="       <program file=\""+"mutex.pgx\" />"+"\n";
                        }
                }
        }
        //fa les crides a les subrutines de transició
         for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
                if (transitionR1(pnmlData,TransitionNo)){
                        j+="       <program
                        file=\""+"transition_"+transitions[TransitionNo].getName()+".pgx\" />"+"\n";
                }
        }
        j+="       <program file=\"start.pgx\" />"+"\n";
        j+="       <program file=\"stop.pgx\" />"+"\n";
        j+="    </programSection>"+"\n";
        j+="    <dataSection>"+"\n"+"       <data file=\"Robot_R1.dtx\" />"+"\n";
        j+="    </dataSection>"+"\n"+"    <aliasSection>"+"\n";
        j+="       <alias name=\"io\" interface=\"io\" autoload=\"true\" />"+"\n";
         j+="    </aliasSection>"+"\n"+"</project>"+"\n";
        return j;
        }

/**
 * Creates data file for R2
 * @param pnmlData
 * @return
 */
private String CreateVariableR2File(DataLayer pnmlData){
        String j=new String();
        int i=0;
        Place[] places=pnmlData.getPlaces();
        j="<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>"+"\n";
        j+="<dataList xmlns=\"DataNameSpace\" >"+"\n";
        j+="  <aioSection>"+"\n"+"  </aioSection>"+"\n"+"  <boolSection>"+"\n";
        for (int placeNo=0; placeNo<places.length; placeNo++){
                //mira si es igual a recurs compartit
                if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                        i+=1;
                        j+="    <bool name=\"bRC"+places[placeNo].getName()+"\" public=\"false\"
                        privilege=\"0\" >"+"\n";
                        j+="      <valueBool index=\"0\" value=\"false\" />"+"\n"+"    </bool>"+"\n";
                }
        }
        //variables que només es creen en cas de recurs compartit
        if (i>0){
                j+="    <bool name=\"cond\" public=\"false\" privilege=\"0\" >"+"\n";
                j+="      <valueBool index=\"0\" value=\"false\" />"+"\n"+"    </bool>"+"\n";
                j+="    <bool name=\"bPout\" public=\"false\" privilege=\"0\" >"+"\n";
                j+="      <valueBool index=\"0\" value=\"false\" />"+"\n"+"    </bool>"+"\n";
        }
        i=0;
```

```
        j+="  </boolSection>"+"\n"+"  <configSection>"+"\n"+"  </configSection>"+"\n";
        j+="  <dioSection>"+"\n";
        j+="  </dioSection>"+"\n"+"  <frameSection>"+"\n"+"  </frameSection>"+"\n";
        j+="  <jointSection>"+"\n"+"  </jointSection>"+"\n"+"  <mdescSection>"+"\n";
        j+="  </mdescSection>"+"\n"+"  <numSection>"+"\n";
        j+="    <num name=\"flag\" public=\"false\" privilege=\"0\" >"+"\n";
   j    +="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
        for (int placeNo=0; placeNo<places.length; placeNo++){
                //mira si son diferents de R1 i de recurs compartit
        if (!(places[placeNo].getSubtype().equalsIgnoreCase("R1"))&
        !(places[placeNo].getSubtype().equalsIgnoreCase("R1aux"))&
        !(places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                //si es que si, declara les variables de cada lloc R2 amb el seu marcat com a valor
                j+="    <num name=\"ntokens_"+places[placeNo].getName()+"\" public=\"false\"
                privilege=\"0\" >"+"\n";
                 j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";}
                //si es de tipus recurs compartit ha de declarar altres variables
        else if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                i+=1;
                j+="    <num name=\"RC"+places[placeNo].getName()+"\" public=\"false\"
                privilege=\"0\" >"+"\n";
                j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
        }
}
//variables que només es creen en cas de recurs compartit
if (i>0){
        j+="    <num name=\"cont\" public=\"false\" privilege=\"0\" >"+"\n";
        j+="      <valueNum index=\"0\" value=\"0\" />"+"\n"+"    </num>"+"\n";
        j+="    <num name=\"Pin\" public=\"false\" privilege=\"0\" >"+"\n";
        for (int z=0; z<i+2; z++){
                j+="      <valueNum index=\""+z+"\" value=\"0\" />"+"\n";
        }
        j+="    </num>"+"\n";
        j+="    <num name=\"Pout\" public=\"false\" privilege=\"0\" >"+'\n";
        for (int z=0; z<i+2; z++){
                j+="      <valueNum index=\""+z+"\" value=\"0\" />"+"\n";
        }
        j+="    </num>"+"\n";}
        j+="  </numSection>"+"\n"+"  <pointSection>"+"\n"+"  </pointSection>"+"\n";
        j+="  <sioSection>"+"\n";
        if (places!=null)
                i=0;
                //recorregut a tots els places
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                if (i==1){
                                        j+="    <sio name=\"buffer_in\" public=\"false\" privilege=\"0\"
                                        size=\"1\" />"+"\n";
                                        j+="    <sio name=\"buffer_out\" public=\"false\"
                                        privilege=\"0\" size=\"1\" />"+"\n";
                                }
                        }
                }
        j+="  </sioSection>"+"\n"+"  <stringSection>"+"\n";
        if (places!=null)
```

```
                i=0;
                //recorregut a tots els places
                for (int placeNo=0; placeNo<places.length; placeNo++){
                        //mira si es recurs compartit
                        if (places[placeNo].getSubtype().equalsIgnoreCase("Shared Resource")){
                                i+=1;
                                j+="    <string name=\"SRC"+places[placeNo].getName()+"\"
                                public=\"false\" privilege=\"0\" >"+"\n";
                                j+="      <valueString index=\"0\" value=\"0\" />"+"\n"+"
                                </string>"+"\n";
                                if (i==1){
                                        j+="    <string name=\"buffer_server\" public=\"false\"
                                        privilege=\"0\" >"+"\n";
                                        j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                        </string>"+"\n";
                                        j+="    <string name=\"buffer_client\" public=\"false\"
                                        privilege=\"0\" >"+"\n";
                                        j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                        </string>"+"\n";
                                        j+="    <string name=\"SRC0\" public=\"false\" privilege=\"0\"
                                        >"+"\n";
                                        j+="      <valueString index=\"0\" value=\"\" />"+"\n"+"
                                        </string>"+"\n";
                                }
                        }
                }
        j+="  </stringSection>"+"\n"+"  <toolSection>"+"\n";
        j+="    <tool name=\"flange\" public=\"false\" privilege=\"0\" >"+"\n";
        j+="      <tFather alias=\"\" name=\"\" fatherIndex=\"0\" />"+"\n";
        j+="      <valueTool index=\"0\" >"+"\n";
        j+="        <ttValue x=\"0\" y=\"0\" z=\"0\" rx=\"0\" ry=\"0\" rz=\"0\" />"+"\n";
        j+="        <io alias=\"io\" name=\"valve1\" ioIndex=\"0\" open=\"0\" close=\"0\" />"+"\n";
        j+="      </valueTool>"+"\n"+"    </tool>"+"\n";
        j+="  </toolSection>"+"\n";
        j+="  <trsfSection>"+"\n"+"  </trsfSection>"+"\n"+"</dataList>"+"\n";
        return j;}

/**
 * Creates project file for R2
 * @param pnmlData
 * @return
 */
private String CreateProjectR2File(DataLayer pnmlData){
        String j=new String();
        int i=0;
        Place[] places=pnmlData.getPlaces();
        Transition[] transitions=pnmlData.getTransitions();
        j="<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>"+"\n";
        j+="<project xmlns=\"ProjectNameSpace\" >"+"\n";
        j+="    <parameters version=\"~VAL3:s5.0.1~Motion:s5.0.1\" stackSize=\"5000\"
        millimeterUnit=\"true\" />"+"\n";
        j+="    <programSection>"+"\n";
        for (int PlaceNo=0; PlaceNo<places.length; PlaceNo++){
                if
                ((places[PlaceNo].getSubtype().equalsIgnoreCase("R2"))|(places[PlaceNo].getSubtyp
                e().equalsIgnoreCase("R2aux"))){
                        //fa les crides a les subrutines de lloc
```

```
                    j+="       <program file=\""+"place_"+places[PlaceNo].getName()+".pgx\"
                    />"+"\n";
                }
                else if((places[PlaceNo].getSubtype().equalsIgnoreCase("Shared Resource"))){
                    i+=1;
                    if (i==1){
                        //fa les crides a les subrutines relacionades amb els recursos
                        compartits
                        j+="       <program file=\""+"comunicacio.pgx\" />"+"\n";
                        j+="       <program file=\""+"lectura.pgx\" />"+"\n";
                        j+="       <program file=\""+"escriptura.pgx\" />"+"\n";
                        j+="       <program file=\""+"gestioRC.pgx\" />"+"\n";
                        j+="       <program file=\""+"mutex.pgx\" />"+"\n";
                    }
                }
        }
 //fa les crides a les subrutines de transició
 for (int TransitionNo=0; TransitionNo<transitions.length; TransitionNo++){
        if (transitionR2(pnmlData,TransitionNo)){
                j+="       <program file=\""+"transition_"+transitions[TransitionNo].getName()+".pgx\"
                />"+"\n";
        }
 }
  j+="      <program file=\"start.pgx\" />"+"\n";
  j+="      <program file=\"stop.pgx\" />"+"\n";
  j+="   </programSection>"+"\n";
  j+="   <dataSection>"+"\n"+"       <data file=\"Robot_R2.dtx\" />"+"\n";
  j+="   </dataSection>"+"\n"+"   <aliasSection>"+"\n";
  j+="      <alias name=\"io\" interface=\"io\" autoload=\"true\" />"+"\n";
  j+="   </aliasSection>"+"\n"+"</project>"+"\n";
 return j;}
```

C. Fitxers dels projectes generats en la validació experimental

Seguidament es presenten els fitxers generats a l'apartat 5.3.4 un cop premut el botó de creació de projectes Val3.

Robot R1

Fitxer Start.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="start" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P0=1
 ntokens_P1=0
 ntokens_P10=0
 ntokens_P3=0
 RCP4=0
 bRCP4=false
 Pin[1]=0
 ntokens_P7=0
 ntokens_P9=0

 sioLink(buffer_in, io:Server)
 taskCreate "comunicacio",20,comunicacio()

 while(true)
  //Codi de la transició T1
  flag=0
  if ntokens_P1&amp;lt;1
   flag=1
  endIf
  call mutex(bRCP4)
  if flag==0 and RCP4==0
   RCP4=1
   bRCP4=false
   call transition_T1()
  else
   bRCP4=false
  endIf
  //Codi de la transició T6
  flag=0
  if ntokens_P7&amp;lt;1
```

```
    flag=1
  endIf
  call mutex(bRCP4)
  if flag==0 and RCP4==0
    RCP4=1
    bRCP4=false
    call transition_T6()
  else
    bRCP4=false
  endIf
  //Codi de la transició T0
  flag=0
  if ntokens_P0&lt;1
    flag=1
  endIf
  if flag==0
    call transition_T0()
  endIf
  //Codi de la transició T5
  flag=0
  if ntokens_P3&lt;1
    flag=1
  endIf
  if flag==0
    call mutex(bRCP4)
    RCP4=0
    bRCP4=false
    call transition_T5()
  endIf
  //Codi de la transició T7
  flag=0
  if ntokens_P9&lt;1
    flag=1
  endIf
  if flag==0
    call mutex(bRCP4)
    RCP4=0
    bRCP4=false
    call transition_T7()
  endIf
  //Codi de la transició T8
  flag=0
  if ntokens_P10&lt;1
    flag=1
  endIf
  if flag==0
    call transition_T8()
  endIf
 endWhile
end
        </code>
      </source>
  </program>
</programList>
```

Fitxer stop.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="stop" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  popUpMsg(&amp;quot;Pending movement commands have been canceled&amp;quot;)
  taskKill("comunicacio")
  resetMotion()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer de comunicacio.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="comunicacio" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  while(true)
    call lectura()
    call gestioRC()
    call escriptura()
  endWhile
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer de escriptura()

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="escriptura" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  SRC0=toString("1",9)
  call mutex(bRCP4)
```

```
  SRCP4=toString("1",RCP4)
  bRCP4=false
  sioLink(buffer_out,io:Server)
  buffer_client=SRC0+" "+SRCP4
  buffer_out=buffer_client
  putln(buffer_client)
end
          </code>
        </source>
    </program>
</programList>
```

Fitxer de lectura.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="lectura" public="false" >
        <description></description>
        <paramSection>
        </paramSection>
        <localSection>
        </localSection>
        <source>
            <code>
begin
  sioLink(buffer_in,io:Server)
  buffer_server=buffer_in
  putln(buffer_server)
  cont=0
  do
    buffer_server=toNum(buffer_server,Pin[cont],cond)
    cont=cont+1
  until(cond != true)
end
            </code>
        </source>
    </program>
</programList>
```

Fitxer de GestioRC.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="gestioRC" public="false" >
        <description></description>
        <paramSection>
        </paramSection>
        <localSection>
        </localSection>
        <source>
            <code>
begin
  call mutex(bRCP4)
  if RCP4==0 and Pin[1]==2
    RCP4=2
  else
```

```
    if RCP4==2 and Pin[1]==0
      RCP4=0
    endIf
  endIf
  bRCP4=false
end
          </code>
        </source>
    </program>
</programList>
```

Fitxer mutex.pgx

```
<?xml version="1.0" encoding="iso-8859-1"?>
<programList xmlns="ProgramNameSpace">
  <program name="mutex" public="false">
    <description />
    <paramSection>
      <param name="bRC" type="bool" byVal="false" />
    </paramSection>
    <localSection />
    <source>
      <code>
begin
//la asignació de recurs (bRC) a true s'ha de fer a la mateixa linea de prova
  wait((bRC==false) and (bRC=true))
end
          </code>
        </source>
    </program>
</programList>
```

Fitxer Robot_R1.dtx

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<dataList xmlns="DataNameSpace" >
  <aioSection>
  </aioSection>
  <boolSection>
    <bool name="bRCP4" public="false" privilege="0" >
      <valueBool index="0" value="false" />
    </bool>
    <bool name="cond" public="false" privilege="0" >
      <valueBool index="0" value="false" />
    </bool>
  </boolSection>
  <configSection>
  </configSection>
  <dioSection>
  </dioSection>
  <frameSection>
  </frameSection>
  <jointSection>
    <joint name="j" public="false" privilege="0">
      <valueJoint index="0">
```

```
      <jointValue j1="-163.486643" j2="41.318266" j3="-46.578086" j4="-1.258564" j5="0.000244"
j6="-0.693054" />
    </valueJoint>
    <valueJoint index="1">
      <jointValue j1="-163.486643" j2="65.345998" j3="-72.150082" j4="-1.258564" j5="0.000122"
j6="-0.693054" />
    </valueJoint>
    <valueJoint index="2">
      <jointValue j1="-146.386595" j2="65.345998" j3="-72.15002" j4="-1.258564" j5="0.000244"
j6="-0.693054" />
    </valueJoint>
    <valueJoint index="3">
      <jointValue j1="-124.254064" j2="56.430818" j3="-25.106049" j4="-4.183044" j5="-
37.704956" j6="-20.63141" />
    </valueJoint>
    <valueJoint index="4">
      <jointValue j1="-37.79171" j2="73.05903" j3="-23.345891" j4="-4.753189" j5="-16.905274"
j6="-21.502809" />
    </valueJoint>
    <valueJoint index="5">
      <jointValue j1="-10.617829" j2="73.059084" j3="-23.345776" j4="-4.753189" j5="-16.905274"
j6="-21.502809" />
    </valueJoint>
    <valueJoint index="6">
      <jointValue j1="8.488884" j2="65.044158" j3="-39.017831" j4="-4.753132" j5="-16.905396"
j6="-21.502809" />
    </valueJoint>
    <valueJoint index="7">
      <jointValue j1="8.488884" j2="42.972453" j3="-43.439887" j4="-81.099016" j5="-41.718995"
j6="-21.502809" />
    </valueJoint>
  </joint>
 </jointSection>
 <mdescSection>
  <mdesc name="nom_speed" public="false" privilege="0">
   <valueMdesc index="0">
    <mdescValue accel="100" vel="30" decel="100" tmax="9999" rmax="99999" blend="joint"
leave="50" reach="50" />
   </valueMdesc>
  </mdesc>
 </mdescSection>
 <numSection>
  <num name="flag" public="false" privilege="0" >
   <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P0" public="false" privilege="0" >
   <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P1" public="false" privilege="0" >
   <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P10" public="false" privilege="0" >
   <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P3" public="false" privilege="0" >
   <valueNum index="0" value="0" />
  </num>
```

```
    <num name="RCP4" public="false" privilege="0" >
      <valueNum index="0" value="0" />
    </num>
    <num name="ntokens_P7" public="false" privilege="0" >
      <valueNum index="0" value="0" />
    </num>
    <num name="ntokens_P9" public="false" privilege="0" >
      <valueNum index="0" value="0" />
    </num>
    <num name="cont" public="false" privilege="0" >
      <valueNum index="0" value="0" />
    </num>
    <num name="Pin" public="false" privilege="0" >
      <valueNum index="0" value="0" />
      <valueNum index="1" value="0" />
      <valueNum index="2" value="0" />
    </num>
  </numSection>
  <pointSection>
  </pointSection>
  <sioSection>
    <sio name="buffer_in" public="false" privilege="0" size="1" />
    <sio name="buffer_out" public="false" privilege="0" size="1" />
  </sioSection>
  <stringSection>
    <string name="SRCP4" public="false" privilege="0" >
      <valueString index="0" value="0" />
    </string>
    <string name="buffer_server" public="false" privilege="0" >
      <valueString index="0" value="" />
    </string>
    <string name="buffer_client" public="false" privilege="0" >
      <valueString index="0" value="" />
    </string>
    <string name="SRC0" public="false" privilege="0" >
      <valueString index="0" value="" />
    </string>
  </stringSection>
  <toolSection>
    <tool name="flange" public="false" privilege="0" >
      <tFather alias="" name="" fatherIndex="0" />
      <valueTool index="0" >
        <ttValue x="0" y="0" z="0" rx="0" ry="0" rz="0" />
        <io alias="io" name="valve1" ioIndex="0" open="0" close="0" />
      </valueTool>
    </tool>
  </toolSection>
  <trsfSection>
  </trsfSection>
</dataList>
```

Fitxer Robot_R1.pjx

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<project xmlns="ProjectNameSpace" >
   <parameters version="~VAL3:s5.0.1~Motion:s5.0.1" stackSize="5000" millimeterUnit="true" />
   <programSection>
      <program file="place_P0.pgx" />
      <program file="place_P1.pgx" />
      <program file="place_P10.pgx" />
      <program file="place_P3.pgx" />
      <program file="comunicacio.pgx" />
      <program file="lectura.pgx" />
      <program file="escriptura.pgx" />
      <program file="gestioRC.pgx" />
      <program file="mutex.pgx" />
      <program file="place_P7.pgx" />
      <program file="place_P9.pgx" />
      <program file="transition_T0.pgx" />
      <program file="transition_T1.pgx" />
      <program file="transition_T5.pgx" />
      <program file="transition_T6.pgx" />
      <program file="transition_T7.pgx" />
      <program file="transition_T8.pgx" />
      <program file="start.pgx" />
      <program file="stop.pgx" />
   </programSection>
   <dataSection>
      <data file="Robot_R1.dtx" />
   </dataSection>
   <aliasSection>
      <alias name="io" interface="io" autoload="true" />
   </aliasSection>
</project>
```

Fitxer Robot_R1.ltx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<externList>
   <data alias="io" type="sio" name="Server" />
</externList>
```

Fitxer transition_T0.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T0" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P0=ntokens_P0-1
 call place_P1()
 ntokens_P1=ntokens_P1+1
end
         </code>
      </source>
```

```
   </program>
</programList>
```

Fitxer transition_T1.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T1" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P1=ntokens_P1-1
  call place_P3()
  ntokens_P3=ntokens_P3+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T5.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T5" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P3=ntokens_P3-1
  call place_P7()
  ntokens_P7=ntokens_P7+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T6.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T6" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
```

```
      </localSection>
      <source>
         <code>
begin
  ntokens_P7=ntokens_P7-1
  call place_P9()
  ntokens_P9=ntokens_P9+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T7.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T7" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P9=ntokens_P9-1
  call place_P10()
  ntokens_P10=ntokens_P10+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T8.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T8" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P10=ntokens_P10-1
  call place_P0()
  ntokens_P0=ntokens_P0+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T10.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T10" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P13=ntokens_P13-1
 call place_P0()
 ntokens_P0=ntokens_P0+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P0.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P0" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
putln("Inici CICLE Robot 1")
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P1.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P1" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
```

```
movej(j[0],flange,nom_speed)
movej(j[1],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end
        </code>
      </source>
   </program>
</programList>
```

Fitxer place_P3.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P3" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[3],flange,nom_speed)
movej(j[4],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P7.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P7" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[6],flange,nom_speed)
movej(j[7],flange,nom_speed)
movej(j[6],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P9.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P9" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[4],flange,nom_speed)
movej(j[3],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P10.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P10" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[1],flange,nom_speed)
movej(j[0],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P13

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P13" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
```

```
      <source>
        <code>
begin
  movej(j[1],flange,nom_speed)
  movej(j[0],flange,nom_speed)
  waitEndMove()
end
        </code>
      </source>
    </program>
</programList>
```

Robot R2

Fitxer Start.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="start" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
        <code>
begin
  ntokens_P11=0
  ntokens_P12=0
  ntokens_P13=0
  RCP4=0
  bRCP4=false
  ntokens_P5=1
  ntokens_P6=0
  ntokens_P8=0


  sioLink(buffer_out, io:Client)
  taskCreate "comunicacio",20,comunicacio()

  while(true)
    //Codi de la transició T3
    flag=0
    if ntokens_P6&lt;1
      flag=1
    endIf
    if flag==0
      call mutex(bPout)
      Pout[1]=2
      bPout=false
      call mutex(bRCP4)
      if RCP4==2
        bRCP4=false
        call transition_T3()
      else
```

```
    bRCP4=false
  endIf
endIf
//Codi de la transició T9
flag=0
if ntokens_P11&lt;1
  flag=1
endIf
if flag==0
  call mutex(bPout)
  Pout[1]=2
  bPout=false
  call mutex(bRCP4)
  if RCP4==2
    bRCP4=false
    call transition_T9()
  else
    bRCP4=false
  endIf
endIf
//Codi de la transició T10
flag=0
if ntokens_P12&lt;1
  flag=1
endIf
if flag==0
  call mutex(bPout)
  Pout[1]=0
  bPout=false
  call mutex(bRCP4)
  RCP4=0
  bRCP4=false
  call transition_T10()
endIf
//Codi de la transició T11
flag=0
if ntokens_P13&lt;1
  flag=1
endIf
if flag==0
  call transition_T11()
endIf
//Codi de la transició T2
flag=0
if ntokens_P5&lt;1
  flag=1
endIf
if flag==0
  call transition_T2()
endIf
//Codi de la transició T4
flag=0
if ntokens_P8&lt;1
  flag=1
endIf
if flag==0
  call mutex(bPout)
```

```
      Pout[1]=0
      bPout=false
      call mutex(bRCP4)
      RCP4=0
      bRCP4=false
      call transition_T4()
    endIf
  endWhile
end
          </code>
        </source>
    </program>
</programList>
```

Fitxer stop.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="stop" public="false" >
        <description></description>
        <paramSection>
        </paramSection>
        <localSection>
        </localSection>
        <source>
            <code>
begin
  popUpMsg(&amp;quot;Pending movement commands have been canceled&amp;quot;)
  taskKill("comunicacio")
  resetMotion()
end
            </code>
        </source>
    </program>
</programList>
```

Fitxer comunicacio.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="comunicacio" public="false" >
        <description></description>
        <paramSection>
        </paramSection>
        <localSection>
        </localSection>
        <source>
            <code>
begin
  while(true)
    call escriptura()
    call lectura()
    call gestioRC()
  endWhile
end
            </code>
```

```
      </source>
   </program>
</programList>
```

Fitxer escriptura.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="escriptura" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  SRC0=toString("1",9)
  call mutex(bPout)
  SRCP4=toString("1",Pout[1])
  bPout=false
  buffer_client=SRC0+" "+SRCP4
  sioLink(buffer_out,io:Client)
  buffer_out=buffer_client
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer lectura.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="lectura" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  sioLink(buffer_in,io:Client)
  buffer_server=buffer_in
  putln(buffer_server)
  cont=0
  do
    buffer_server=toNum(buffer_server,Pin[cont],cond)
    cont=cont+1
  until(cond != true)
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer gestioRC.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="gestioRC" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  call mutex(bRCP4)
  RCP4=Pin[1]
  bRCP4=false
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer mutex.pgx

```
<?xml version="1.0" encoding="iso-8859-1"?>
<programList xmlns="ProgramNameSpace">
  <program name="mutex" public="false">
    <description />
    <paramSection>
      <param name="bRC" type="bool" byVal="false" />
    </paramSection>
    <localSection />
    <source>
      <code>
begin
//la asignació de recurs (bRC) a true s'ha de fer a la mateixa linea de prova
  wait((bRC==false) and (bRC=true))
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer Robot_R2.dtx

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<dataList xmlns="DataNameSpace" >
  <aioSection>
  </aioSection>
  <boolSection>
    <bool name="bRCP4" public="false" privilege="0" >
      <valueBool index="0" value="false" />
    </bool>
    <bool name="cond" public="false" privilege="0" >
      <valueBool index="0" value="false" />
```

```
    </bool>
    <bool name="bPout" public="false" privilege="0" >
      <valueBool index="0" value="false" />
    </bool>
  </boolSection>
  <configSection>
  </configSection>
  <dioSection>
  </dioSection>
  <frameSection>
  </frameSection>
  <jointSection>
    <joint name="j" public="false" privilege="0">
      <valueJoint index="0">
        <jointValue  j1="23.115006"  j2="47.160072"  j3="-36.903078"  j4="0.000114"  j5="-0.000244"
j6="0.000183" />
      </valueJoint>
      <valueJoint index="1">
        <jointValue j1="19.898472" j2="73.90669" j3="-34.668046" j4="5.7E-05" j5="0" j6="-0.000183"
/>
      </valueJoint>
      <valueJoint index="2">
        <jointValue j1="3.051739" j2="73.906745" j3="-34.668046" j4="5.7E-05" j5="0" j6="-0.000366"
/>
      </valueJoint>
      <valueJoint index="3">
        <jointValue j1="-19.90654" j2="73.90669" j3="-34.668046" j4="5.7E-05" j5="0" j6="0" />
      </valueJoint>
      <valueJoint index="4">
        <jointValue j1="-111.026863" j2="85.253219" j3="-68.288388" j4="49.882337" j5="26.265749"
j6="-20.657776" />
      </valueJoint>
      <valueJoint index="5">
        <jointValue j1="-131.703191" j2="59.146501" j3="-66.188392" j4="49.882337" j5="25.095704"
j6="-119.291567" />
      </valueJoint>
      <valueJoint index="6">
        <jointValue  j1="-170.25159"  j2="59.146501"  j3="-66.188392"  j4="49.882337"  j5="25.095704"
j6="-119.291749" />
      </valueJoint>
      <valueJoint index="7">
        <jointValue  j1="-169.531634"  j2="33.025419"  j3="-52.45434"  j4="49.882395"  j5="25.096437"
j6="-119.291749" />
      </valueJoint>
    </joint>
  </jointSection>
  <mdescSection>
    <mdesc name="nom_speed" public="false" privilege="0">
      <valueMdesc index="0">
        <mdescValue  accel="100"  vel="30"  decel="100"  tmax="9999"  rmax="99999"  blend="joint"
leave="50" reach="50" />
      </valueMdesc>
    </mdesc>
  </mdescSection>
  <numSection>
    <num name="flag" public="false" privilege="0" >
      <valueNum index="0" value="0" />
```

```
  </num>
  <num name="ntokens_P11" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P12" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P13" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="RCP4" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P5" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P6" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="ntokens_P8" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="cont" public="false" privilege="0" >
    <valueNum index="0" value="0" />
  </num>
  <num name="Pin" public="false" privilege="0" >
    <valueNum index="0" value="0" />
    <valueNum index="1" value="0" />
    <valueNum index="2" value="0" />
  </num>
  <num name="Pout" public="false" privilege="0" >
    <valueNum index="0" value="0" />
    <valueNum index="1" value="0" />
    <valueNum index="2" value="0" />
  </num>
</numSection>
<pointSection>
</pointSection>
<sioSection>
  <sio name="buffer_in" public="false" privilege="0" size="1" />
  <sio name="buffer_out" public="false" privilege="0" size="1" />
</sioSection>
<stringSection>
  <string name="SRCP4" public="false" privilege="0" >
    <valueString index="0" value="0" />
  </string>
  <string name="buffer_server" public="false" privilege="0" >
    <valueString index="0" value="" />
  </string>
  <string name="buffer_client" public="false" privilege="0" >
    <valueString index="0" value="" />
  </string>
  <string name="SRC0" public="false" privilege="0" >
    <valueString index="0" value="" />
  </string>
</stringSection>
<toolSection>
```

```
    <tool name="flange" public="false" privilege="0" >
      <tFather alias="" name="" fatherIndex="0" />
      <valueTool index="0" >
        <ttValue x="0" y="0" z="0" rx="0" ry="0" rz="0" />
        <io alias="io" name="valve1" ioIndex="0" open="0" close="0" />
      </valueTool>
    </tool>
  </toolSection>
  <trsfSection>
  </trsfSection>
</dataList>
```

Fitxer Robot_R2.pjx

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<project xmlns="ProjectNameSpace" >
    <parameters version="~VAL3:s5.0.1~Motion:s5.0.1" stackSize="5000" millimeterUnit="true" />
    <programSection>
        <program file="place_P11.pgx" />
        <program file="place_P12.pgx" />
        <program file="place_P13.pgx" />
        <program file="comunicacio.pgx" />
        <program file="lectura.pgx" />
        <program file="escriptura.pgx" />
        <program file="gestioRC.pgx" />
        <program file="mutex.pgx" />
        <program file="place_P5.pgx" />
        <program file="place_P6.pgx" />
        <program file="place_P8.pgx" />
        <program file="transition_T10.pgx" />
        <program file="transition_T11.pgx" />
        <program file="transition_T2.pgx" />
        <program file="transition_T3.pgx" />
        <program file="transition_T4.pgx" />
        <program file="transition_T9.pgx" />
        <program file="start.pgx" />
        <program file="stop.pgx" />
    </programSection>
    <dataSection>
        <data file="Robot_R2.dtx" />
    </dataSection>
    <aliasSection>
        <alias name="io" interface="io" autoload="true" />
    </aliasSection>
</project>
```

Fitxer Robot_R2.ltx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<externList>
    <data alias="io" type="sio" name="Client" />
</externList>
```

Fitxer transition_T2.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T2" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P5=ntokens_P5-1
  call place_P6()
  ntokens_P6=ntokens_P6+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T3.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T3" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  ntokens_P6=ntokens_P6-1
  call place_P8()
  ntokens_P8=ntokens_P8+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T4.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T4" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
```

```
  ntokens_P8=ntokens_P8-1
  call place_P11()
  ntokens_P11=ntokens_P11+1
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer transition_T5.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="transition_T5" public="false" >
       <description></description>
       <paramSection>
       </paramSection>
       <localSection>
       </localSection>
       <source>
          <code>
begin
  ntokens_P3=ntokens_P3-1
  call place_P4()
  ntokens_P4=ntokens_P4+1
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer transition_T7.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="transition_T7" public="false" >
       <description></description>
       <paramSection>
       </paramSection>
       <localSection>
       </localSection>
       <source>
          <code>
begin
  ntokens_P10=ntokens_P10-1
  call place_P11()
  ntokens_P11=ntokens_P11+1
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer transition_T9.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T9" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P11=ntokens_P11-1
 call place_P12()
 ntokens_P12=ntokens_P12+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T10.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T10" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P12=ntokens_P12-1
 call place_P13()
 ntokens_P13=ntokens_P13+1
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer transition_T11.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="transition_T11" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 ntokens_P13=ntokens_P13-1
```

```
  call place_P5()
  ntokens_P5=ntokens_P5+1
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer place_P5.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="place_P5" public="false" >
       <description></description>
       <paramSection>
       </paramSection>
       <localSection>
       </localSection>
       <source>
          <code>
begin
putln("Inici CICLE Robot 2")
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer place_P6.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
    <program name="place_P6" public="false" >
       <description></description>
       <paramSection>
       </paramSection>
       <localSection>
       </localSection>
       <source>
          <code>
begin
movej(j[0],flange,nom_speed)
movej(j[1],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end
          </code>
       </source>
    </program>
</programList>
```

Fitxer place_P8.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P8" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[3],flange,nom_speed)
movej(j[4],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P10.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P10" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
 movej(j[6],flange,nom_speed)
 movej(j[7],flange,nom_speed)
 movej(j[6],flange,nom_speed)
 movej(j[5],flange,nom_speed)
 waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P11.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P11" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
```

```
        <code>
begin
movej(j[6],flange,nom_speed)
movej(j[7],flange,nom_speed)
movej(j[6],flange,nom_speed)
movej(j[5],flange,nom_speed)
waitEndMove()
end
        </code>
      </source>
   </program>
</programList>
```

Fitxer place_P12.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P12" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[4],flange,nom_speed)
movej(j[3],flange,nom_speed)
movej(j[2],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

Fitxer place_P13.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P13" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
movej(j[1],flange,nom_speed)
movej(j[0],flange,nom_speed)
waitEndMove()
end
         </code>
      </source>
   </program>
```

```
</programList>
```

Fitxer place_P14.pgx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<programList xmlns="ProgramNameSpace" >
   <program name="place_P14" public="false" >
      <description></description>
      <paramSection>
      </paramSection>
      <localSection>
      </localSection>
      <source>
         <code>
begin
  movej(j[1],flange,nom_speed)
  movej(j[0],flange,nom_speed)
  waitEndMove()
end
         </code>
      </source>
   </program>
</programList>
```

D. Manual d'usuari del Platform Independent Petri Net Editor

El present manual vol ser una guia que permeti mostrar als usuaris de PIPE les instruccions bàsiques del seu funcionament.

Quan s'executa el programa PIPE apareix una pantalla com la de la *Figura 3.1*. La barra de menú té cinc elements i just a sota apareix una barra amb la majoria de funcionalitats de la barra de menú en forma de botons com es mostra en la *Figura D.1*

Figura D.1 Barra de menú i barra de menú en forma de botons

Els sis primers botons fan referència a obrir i guardar documents i son les instruccions que es troben en el menú File:

- El primer serveix per crear una nova xarxa de Petri. Cada xarxa de Petri que es creï o s'obri es situa a una finestra independent, seleccionables a través de les pestanyes de la *Figura D*.2.

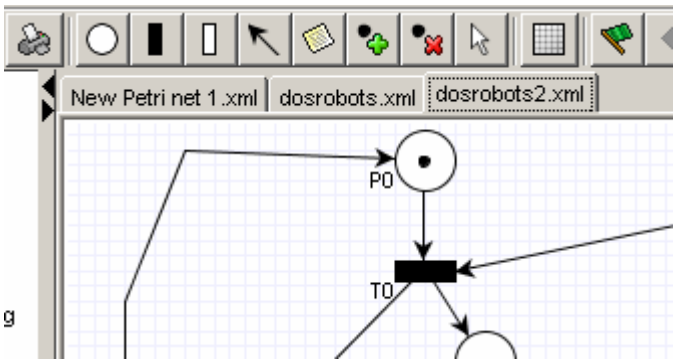


Figura D.2 Tres pestanyes que contenen tres xarxes obertes

- El segon serveix per obrir una xarxa guardada al disc dur.
- El tercer i el quart tenen les funcions de guardar i guardar com... típics de tots els programes de l'entorn Windows.

- El cinquè botó serveix per tancar una xarxa oberta, eliminant la pestanya corresponent de l'editor.

- El sisè botó imprimeix la xarxa que es té en pantalla.

Seguidament es troba un conjunt de vuit botons que presenten les instruccions de creació de les xarxes:

- El primer serveix per afegir un lloc a l'editor.

- El segon i tercer botó afegeixen una transició. La diferència és que la transició de color blanc accepta un temps de dispar.

- El quart botó afegeix un arc. Per posar-lo, s'ha de prémer primer sobre un element, lloc o transició i seguidament sobre un element contrari a l'anterior.

- El cinquè botó afegeix anotacions en forma de text i només tenen un significat informatiu per l'usuari.

- El sisè i setè botons serveixen per afegir o treure marques sobre els llocs. Per a fer-ho, s'ha de prémer sobre el lloc un cop es té seleccionada l'acció que es vol fer.

- El vuitè botó serveix per seleccionar elements. Es poden seleccionar els elements individualment o dibuixant una quadre sobre els objectes que es volen seleccionar. Un cop seleccionats, els objectes es poden moure arrastrant-los amb el ratolí.

El següent botó en forma de graella ens permet ajustar la definició de la quadrícula de l'editor, permetent la creació dels elements en posicions més concretes.

Seguidament, el grup de cinc botons permet activar el mode d'animació.

- El primer, que està activat per defecte a diferència dels altres quatre (*Figura D.1*) ens permet precisament activar la resta i engegar el mode d'animació. Un cop s'activa, les transicions que estan habilitades es ressalten en vermell i a sota l'arbre de mòduls s'activa una finestra de seguiment de l'animació on es mostren les transicions disparades, per ordre de dispar.

- El tercer botó permet seleccionar manualment una de les transicions habilitades per tal de disparar-la, i el segon botó retrocedeix l'últim dispar.

- El quart botó dispara una de les transicions habilitades de forma aleatòria, mentre que el cinquè permet disparar també de forma automàtica un nombre determinat de transicions introduïdes per l'usuari.

Finalment l'últim botó activa la documentació d'ajuda.

A més d'aquestes opcions, tots els elements mostrats a l'editor presenten un menú desplegable amb el seguit d'accions a realitzar sobre aquest element. Així els arcs tenen dos opcions, eliminar l'arc i editar el pes de l'arc. Les transicions tenen la opció d'eliminar, de canviar el nom, de girar-les i de permetre la temporització. Finalment els llocs presenten les opcions d'eliminar, canviar el nom, editar el número de marques i els dos elements afegits en el present projecte: veure/editar el codi i escollir el tipus del lloc (*Figures 7.3 i 7.4*).

E. Documentació addicional

En el CD que s'adjunta amb la memòria es troba tota la documentació relacionada amb el Val3, el PNML i l'informe del PIPE.

F. Cost del projecte

En l'estudi del cost del present projecte s'analitza la despesa que ha suposat la realització del mateix. No es tindran en compte doncs, les inversions i despeses derivades de l'ús de l'aplicació ni de la validació experimental amb els robots i el Val3 Studio.

El cost derivat de les hores de treball invertides per a la realització del projecte es detallen en la *Taula F.1*.

Personal	**Mesos**	**Hores/Mes**	**Hores totals** *(h)*	**Cost/hora** *(€/h)*	**Cost total** *(€)*
Programador	10	32	320	30	9600

Taula F.1. Cost derivat de les hores de treball invertides per el personal.

El cost de l'ús de l'equip informàtic es realitza a partir del preu de compra tenint en compte que l'amortització en un centre com l'Institut d'Organització i Control es calcula a 3 anys (*Taula F.2*).

Cost d'adquisició *(€)*	**Període d'utilització**	**Amortització**	**Cost total** *(€)*
1200	10 mesos	27,8%	333,3

Taula F.2. Cost derivat de l'ús de l'equip informàtic.

Així doncs, el cost total del projecte, resultat de sumar les dues anteriors quantitats és de:

9933,3 €

G. Impacte ambiental

La utilització del present projecte genera un impacte mínim en l'entorn, però en tot cas de tipus positiu. El fet de ser una aplicació de software lliure, haver-se desenvolupat a través d'un programari lliure i distribuir-se de manera gratuïta implica una menor utilització de dispositius d'emmagatzematge tals com CD's o DVD's, ja que l'obtenció es fa a través d'Internet.

A més, si es té en compte que aquesta aplicació serveix per reduir la feina de programació i optimitzar el funcionament dels robots, es conclou que l'ús d'aquest programa implica un estalvi energètic tan en l'equip informàtic utilitzat per a la programació com en l'electricitat que se subministra als robots.