SDLPS

TÍTOL DEL DOCUMENT

LIAM.SDLPS01

VERSIÓ ??

CREAT PER: MÀXIM COLLS

CREACIÓ: 09/11/2009 12:27:00

EDITAT PER: MÀXIM COLLS

ÚLTIMA EDICIÓ: 14/03/2011 19:28:00

ÍNDEX

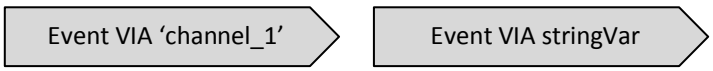
1. *SDL SUBSET* UTILITZAT

CARACTERÍSTIQUES DEL SUBSET

Per tal de poder representar qualsevol tipus de model mitjançant els diagrames s'ha utilitzat una variant de SDL amb les següents característiques:

- **Ús de codi C++** en lloc del pseudocodi que utilitza SDL. Això afecta als agents *Task*, *DCL*, *Procedure*, *Decision* i *Output* (*Text Extension*)
- Especificació del destí a on s'enviarà l'event. D'aquesta manera l'agent Output es complementarà:
  - Mitjançant la cadena **TO** seguida del Procés al qual es vol enviar la senyal. Als exemples, en el primer cas s'enviarà la senyal cap al Procés amb identificador 'Block_Process', en el segon exemple la senyal s'enviarà al mateix Procés i al tercer exemple la senyal s'enviarà al Procés que tingui com identificador el contingut de la variable stringVar.

  - Mitjançant la cadena **VIA** seguida del canal pel qual es vol enviar la senyal. El canal es pot especificar literalment mitjançant el seu nom entre cometes simples o bé utilitzant una variable char*.



- **Extensió de l'agent *Output*** mitjançant la forma (Visio Shape) *Text Extension.* La extensió serveix per completar l'enviament de l'event especificant el valor de variables associades a l'event

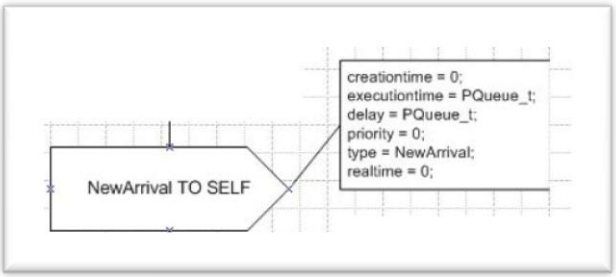


Figura 1 - Exemple d'utilització del text extension

Les variables pròpies de l'event que es podran modificar seran:

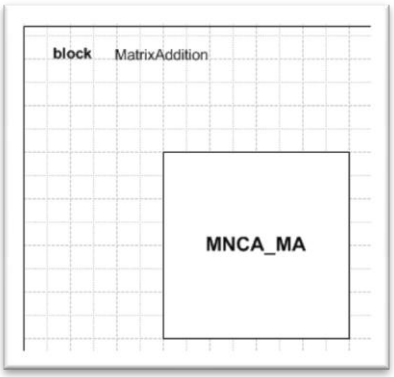
- double delay; //indica el temps que tarda a arribar al destí
- int priority; //prioritat en cas d'arribada de dos events simultaniament

REPRESENTACIÓ D'UN MNCA^K EN SDL

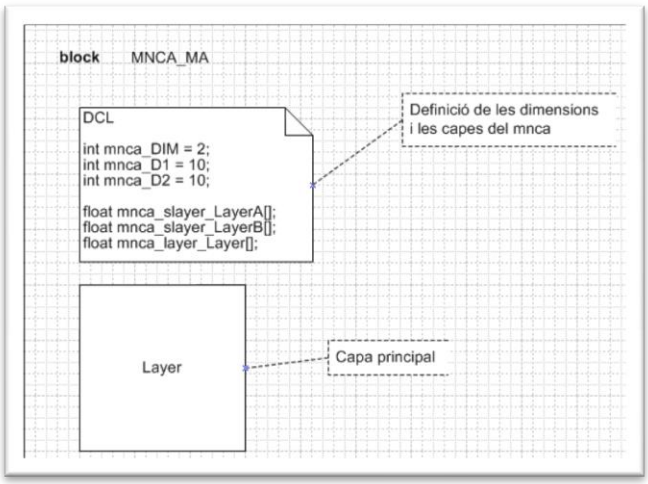
REPRESENTACIÓ DEL MNCA^K

Un mnCA^k és una generalització d'un autòmat cel·lular on *m* és el nombre de capes, *n* les dimensions de cada capa i *k* el nombre de capes principals.

Per representar un mnCA^k s'utilitzarà un *Shape* de tipus *Block* on el nom utilitzarà el prefix "MNCA_". Així si volem definir un mnCA^k anomenat "MA":

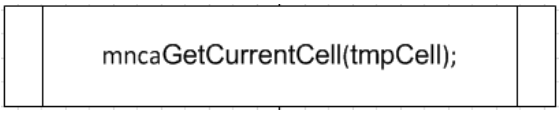


Tot seguit es mostra el contingut del *Block* "MNCA_MA". Els elements principals són, les dimensions, el tamany de cada dimensió, les capes secundàries i les capes principals:

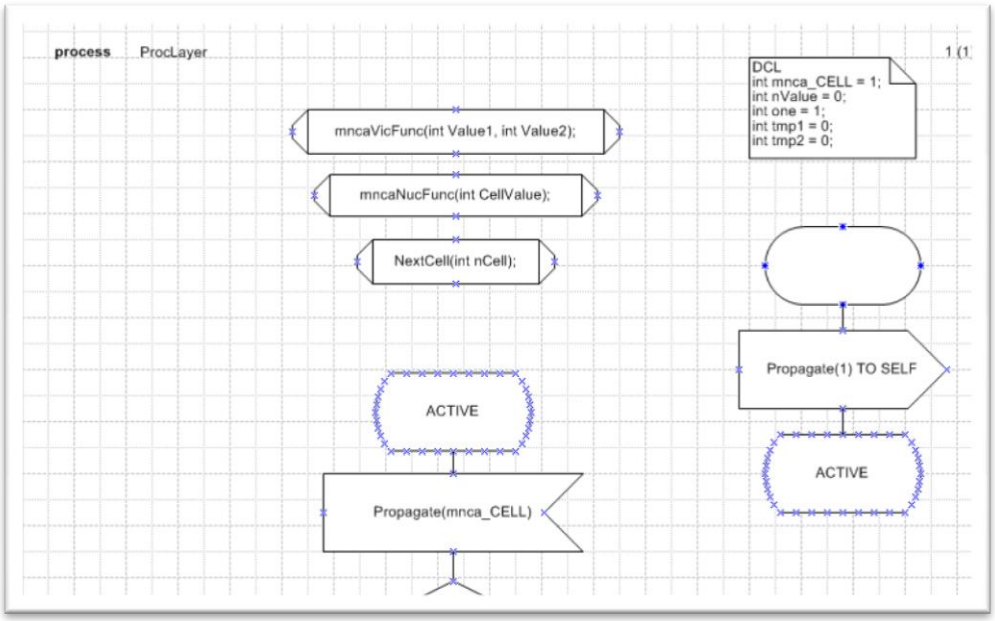


REPRESENTACIÓ D'UNA CAPA PRINCIPAL

- Un sol *Process* ens representarà cada una de les cel·les d'una mateixa capa principal.
- Per tal de saber quin és l'identificador de la cel·la que s'està tractant en un moment donat, s'utilitzarà el *Procedure Call*:



  De manera que assignarà a la variable *tmpCell* el valor de l'identificador de la cel·la actual.
- La **funció de veïnatge** ens permetrà consultar els estats (valors de la capa) de les cel·les veïnes. La funció de veïnatge s'implementarà mitjançant un *Procedure,* aquest, es podrà definir utilitzant diagrames SDL o bé implementant-ho en codi a l'SDLPS.
- La **funció de nucli** ens permet actualitzar l'estat de les cel·les de nucli (usualment el nucli és únicament una cel·la, l'actual). La funció de nucli s'implementarà mitjançant un *Procedure,* aquest, es podrà definir utilitzant diagrames SDL o bé implementant-ho en codi a l'SDLPS.


process ProcLayer
1 (1
DCL
int mnca_CELL = 1;
int nValue = 0;
int one = 1;
int tmp1 = 0;
int tmp2 = 0;
mncaVicFunc(int Value1, int Value2);
mncaNucFunc(int CellValue);
NextCell(int nCell);
ACTIVE
Propagate(mnca_CELL)
Propagate(1) TO SELF
ACTIVE

2. MANUAL D'ÚS DEL PLUGIN DE VISIO

Documentació referent al Visio Editor SDLPS. Aquest *plugin* del Microsoft Visio permet generar el fitxer .sdlps a partir dels diagrames SDL.

Els diagrames SDL es representen amb l'ajut del Microsoft Visio i el *plugin* SanDriLa (http://www.sandrila.co.uk/), que permet la utilització de les formes estàndards de l'SDL.

A l'executar el *plugin* apareix un únic bloc i el menú SDLPS. Per tal d'exportar els diagrames SDL a un fitxer .sdlps només cal posar el nom del model en el bloc i executar Export to XML en el menú SDLPS.

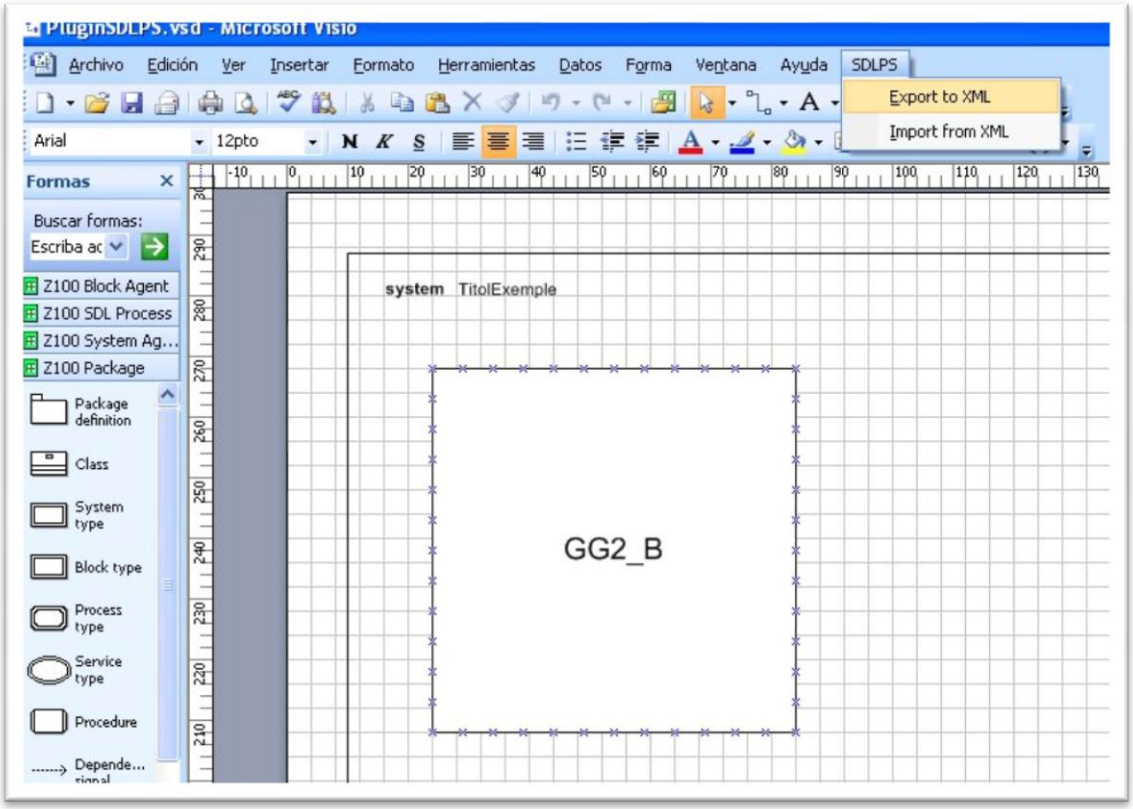
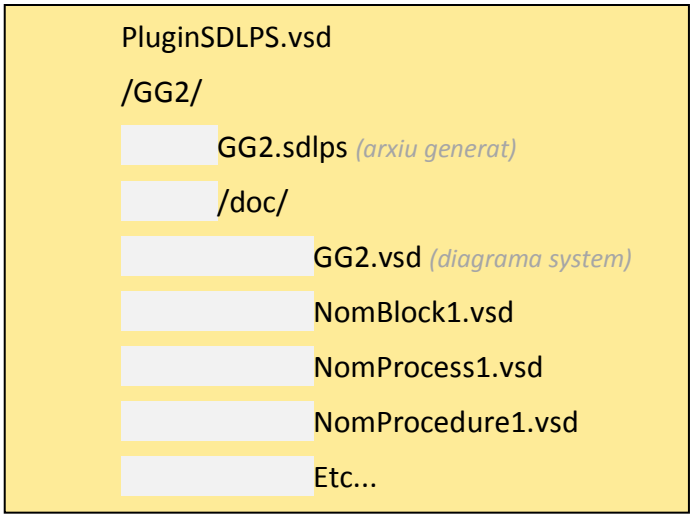


Figura 2 – Captura de pantalla del plugin

ESTRUCTURA DE CARPETES

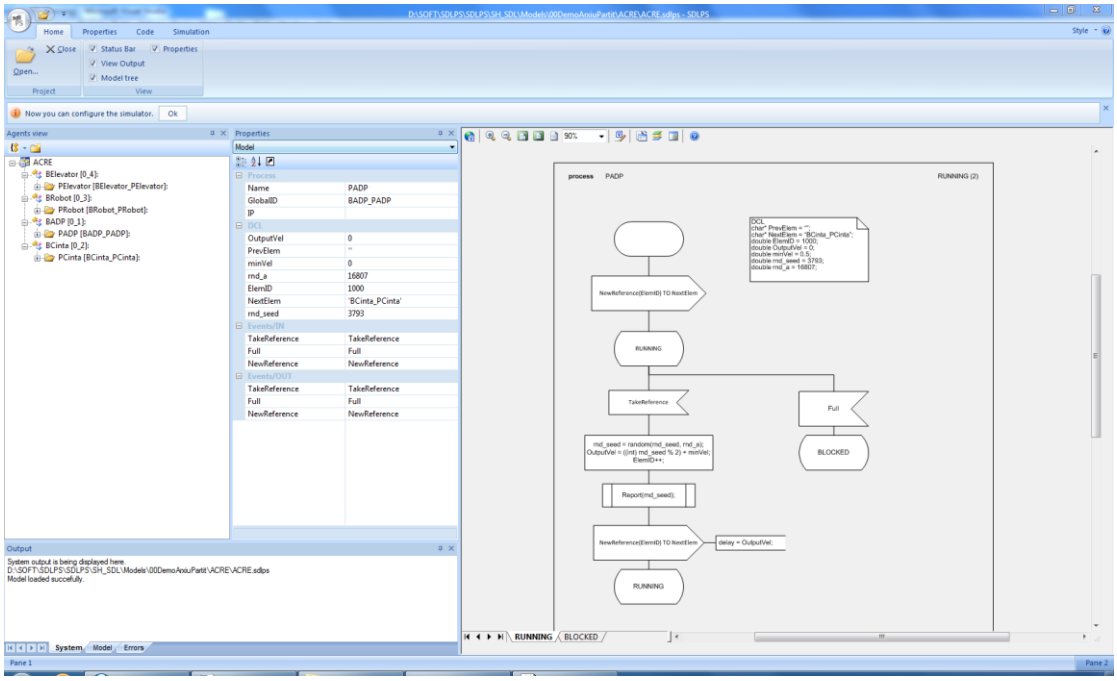
Si volem generar el fitxer .sdlps a partir dels diagrames SDL (fitxers .vsd) caldrà tenir els fitxers correctament localitzats.

En l'exemple següent es mostra el cas que vulguem exportar el model anomenat GG2:



La carpeta que conté els diagrames del model ha de tenir el nom del propi model, en aquest cas /GG2/. Dins del directori del model hi haurà la carpeta /doc/ que contindrà tots els arxius dels diagrames SDL. El nom dels fitxers ha de correspondre amb el nom de l'agent que representa.

3. MANUAL D'ÚS DE L'SDLPS



Per a realitzar una simulació primer cal disposar de l'arxiu .sdlps que conté la estructura i la lògica del model. El model .sdlps es pot crear manualment o mitjançant el *plugin* descrit a l'apartat anterior.

INSTRUCCIONS PAS A PAS

Carregar el model .sdlps en el programa SDLPS

Per carregar un model prèviament generat mitjançant el Plugin del Visio explicat al tema anterior només cal obrir el fitxer des del menú principal de l'SDLPS. En el menú principal apareixerà també la llista dels últims models carregats.

El fitxer del model ha de ser un fitxer en format XML sintàcticament correcte i amb un node arrel anomenat <system>.

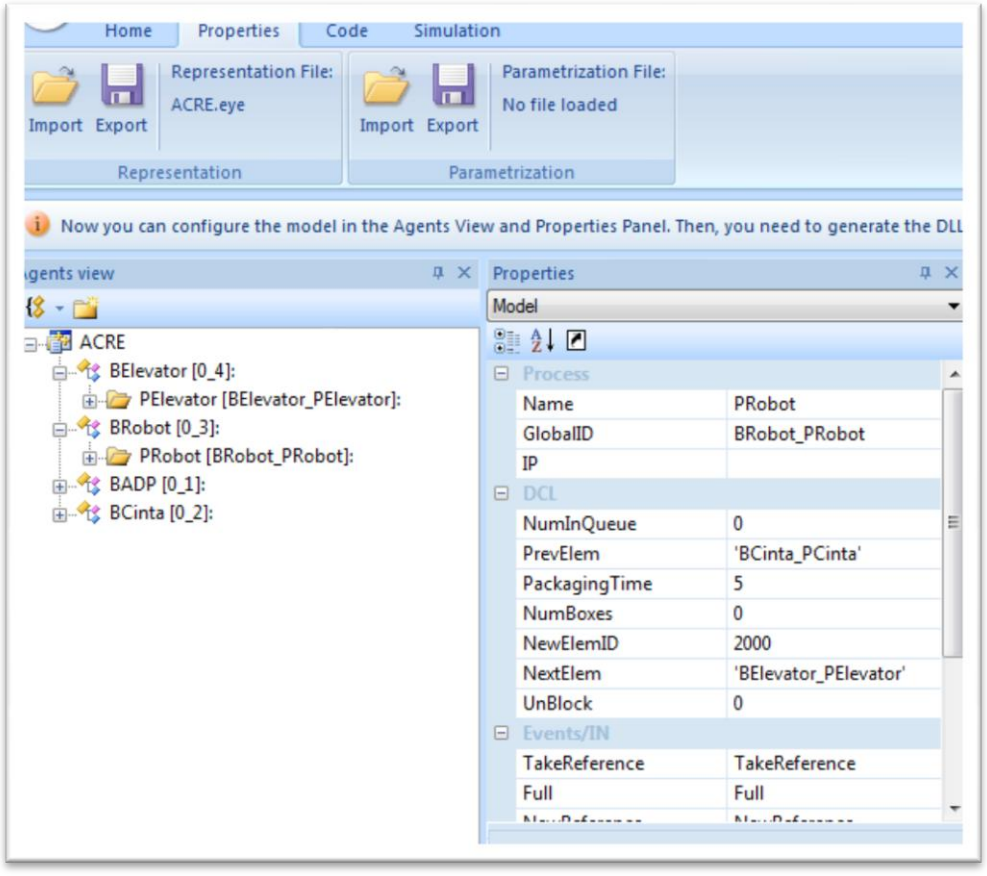
Si no hi ha cap error durant la càrrega apareixerà l'estructura dels agents en el panell "Agents View".

Parametritzar el model

El procés de parametrització del model consisteix en aplicar una parametrització concreta. Aquest procés es pot ometre si volem executar la parametrització per defecte que està especificada al fitxer .sdlps.

Per aplicar una parametrització es pot fer manualment o important un fitxer de parametrització (.param).

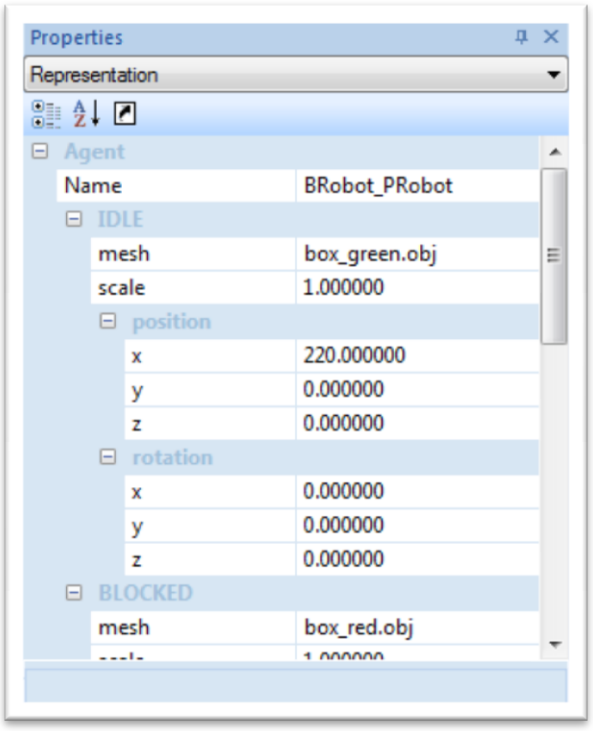
Per crear una parametrització només cal navegar per l'arbre d'Agents i en el panell *Properties* i modificar les variables DCL que desitgem. Un com s'hagin acabat de parametritzar, aquesta parametrització es podrà exportar des de la pestanya *Properties*. Des del mateix panell, es podran importar parametritzacions ja existents.



Definir una representació 3D del model.

De manera similar es poden crear/exportar/importar fitxers de representació. Per modificar la representació d'un *Agent Process* només cal seleccionar-lo al panell *Agents View* i en el panell eleccionar en el desplegable del

panell *Properties* la opció *Representation*. Des del panell, es podran modificar la representació de cada possible estat del Process.



Crear el codi, compilar i enllaçar la dll.

Un canvi en la parametrització o representació del model implica generar de nou la DLL que executa el model. Aquest procés es realitza des de la pestanya Code mitjançant el botó: *Create, Compile and Link DLL*.

Configurar la simulació

Abans d'inicialitzar la simulació es pot definir el final de la simulació. A la pestanya *Simulation* es troba el botó *Simulation time*. Des del menú que apareix, podem definir com s'acabarà la simulació: per temps, nombre d'iteracions o mai.

Executar la simulació

Un cop s'ha acabat tot el procés previ a la simulació es podrà inicialitzar i executar. L'execució del model pot ser pas a pas o contínua. Per definir quin tipus d'execució es vol, abans d'inicialitzar la simulació es marcarà si es vol una simulació

step by step o no. Un cop s'hagi configurat la simulació és moment d'inicialitzar-la (*Initialize*) i posteriorment executar-la (*Run*).

L'execució de la simulació genera una traça que es guarda en un fitxer extern (.trace) dins la carpeta traces relativa al fitxer del model.

TIPUS DE FITXERS

MODEL FILE (.sdlps)

Aquest fitxer conté la estructura de Blocs i Processos i les seves connexions (canals). Aquest fitxer es pot generar manualment o mitjançant el plugin de visio (veure el següent apartat).

```
<?xml version="1.0"?>
<system id="0" name="ACRE" implementation="" IP="" portRead="">
    <channels>
        <channel name="channel1" start="BADP" end="BCinta" dual="yes">
            <event name="NewReference"></event>
            <event name="TakeReference"></event>
            <event name="Full"></event>
        </channel>
        <channel name="channel2" start="BCinta" end="BRobot" dual="yes">
            <event name="NewReference"></event>
            <event name="TakeReference"></event>
            <event name="Full"></event>
        </channel>
        <channel name="channel3" start="BRobot" end="BElevator" dual="yes">
            <event name="NewReference"></event>
            <event name="TakeReference"></event>
        </channel>
    </channels>
    <block id="1" name="BADP" implementation="" IP="" portRead="">
        <channels>
            <channel name="ADPCh" start="BADP" end="PADP" dual="yes">
                <event name="NewReference"></event>
                <event name="TakeReference"></event>
                <event name="Full"></event>
            </channel>
        </channels>
        <process id="1" name="PADP" implementation="ADP.sdlprocess" IP="" portRead="">
        <DCLS>
            <DCL name="PrevElem" type="char*" value="''"></DCL>
            <DCL name="NextElem" type="char*" value="'BCinta_PCinta'"></DCL>
            <DCL name="ElemID" type="double" value="1000"></DCL>
            <DCL name="OutputVel" type="double" value="0"></DCL>
            <DCL name="minVel" type="double" value="0"></DCL>
            <DCL name="rnd_seed" type="double" value="3793"></DCL>
            <DCL name="rnd_a" type="double" value="16807"></DCL>
        </DCLS>
        </process>
    </block>
    <block id="2" name="BCinta" implementation="" IP="" portRead="">
        <channels>
            <channel name="CintaCh" start="BCinta" end="PCinta" dual="yes">
                <event name="NewReference"></event>
                <event name="TakeReference"></event>
                <event name="Full"></event>
```

REPRESENTATION FILE (.eye)

Aquest fitxer conté la informació referent a la representació del model dins un espai 3D. Cada agent que vol ser representat tindrà definida una representació per a cada un dels seus estats. La informació de la representació consta de:

- Escalat i fitxer de la malla (format Wavefront Object File *.obj)

- Posició x, y, z
- Rotació x, y, z

```
<ModelInfo>
 <ModelName>Ausonia</ModelName>
<Agents>
  <Agent name='BRobot_PRobot'>
    <state name='IDLE'>
      <mesh scale='1'>box_green.obj</mesh>
      <pos x='220' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
    <state name='BLOCKED'>
      <mesh scale='1'>box_red.obj</mesh>
      <pos x='220' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
    <state name='WORKING'>
      <mesh scale='1'>box_yellow.obj</mesh>
      <pos x='220' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
  </Agent>
  <Agent name='BElevator_PElevator'>
    <state name='RUNNING'>
      <mesh scale='1'>box_green.obj</mesh>
      <pos x='250' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
  </Agent>
  <Agent name='BADP_PADP'>
    <state name='RUNNING'>
      <mesh scale='1'>box_green.obj</mesh>
      <pos x='-20' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
    <state name='BLOCKED'>
      <mesh scale='1'>box_red.obj</mesh>
      <pos x='-20' y='0' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
  </Agent>
  <Agent name='BCinta_PCinta'>
    <state name='ROLLING'>
      <mesh scale='1'>cinta200x20.obj</mesh>
      <pos x='100' y='-10' z='0' />
      <rot x='0' y='0' z='0' />
    </state>
  </Agent>
</Agents>
</ModelInfo>
```

SDL PROCESS File (.sdlprocess)

Un *Process* pot estar implementat en un fitxer extern d'aquest tipus. D'aquesta manera es pot construir una llibreria d'objectes i reutilitzar-los en diferents models.

```
<sdlprocess>
    <process id="1" name="PElevator" implementation="" IP="" portRead="">
        <DCLS>
            <DCL name="PrevElem" type="char*" value="'PID'"></DCL>
            <DCL name="ElemID" type="double" value=""></DCL>
        </DCLS>
        <start>
            <setstate id="1" name="RUNNING"></setstate>
        </start>
        <state name = "RUNNING">
            <input id="1" name="NewReference"></input>
            <procedurecall id="2" name="RemoveAt">
                <param name="PARAM_1" value="ElemID"></param>
            </procedurecall>
            <output id="2" name="TakeReference" self="" to="PrevElem" via="">
            </output>
            <setstate id="3" name="RUNNING"></setstate>
        </state>
    </process>
```

PARAMETRIZATION File (.param)

Aquest fitxer conté les parametritzacions d'un model concret. Una parametrització concreta pot ser la configuració d'una execució en un joc d'experiments. Normalment, s'especifiquen valors concrets de variables DCL que caracteritzen el model.

```
<Parametrization>
    <Agent name='BRobot_PRobot'>
        <DCL name='PackagingTime' type='double' value='7' />
    </Agent>
</Parametrization>
```

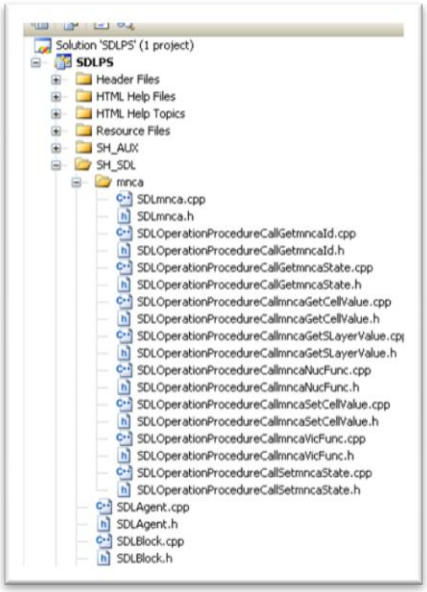
TRACE FILE (.trace)

Aquest fitxer conté la sortida de l'execucció del simulador SDLPS.

Es composa de dues parts diferenciades:

- La primera, encapçalada pel tag **ModelInfo** conté la informació del model així com el nom, la seva representació (.eye), tipus d'execució, parametrització, etc.
- La segona part està formada per una seqüència **d'Events**. D'aquests en tenim de dos classes:
  - Events de l'execució del model. (*input, output, setstate*)
  - Events de representació (*Eye_AnimTo, Eye_Remove, Eye_SetState*, *Eye_Report*)

IMPLEMENTACIÓ D'UN *PROCEDURE* A L'SDLPS



Molts cops ens interessarà implementar un *Procedure* directament a l'SDLPS. Per això haurem de crear una subclasse de la classe *CSDLOperationProcedureCall* i reimplementar el mètode *Execute()*.

Un cas pot ser la implementació de les classes *CSDLOperationProcedureCallmncaNucFunc* i *CSDLOperationProcedureCallmncaVicFunc* que implementen les funcions de nucli i veïnatge respectivament.

Sempre que s'afegeix la implementació d'un *Procedure* cal modificar el mètode *LoadOperationProcedureCall* de la classe *CSDLLoader*. Cal indicar que si a l'XML

s'especifica que el Procedure té una implementació (atribut *implementation*) aquest està implementat en una classe de l'SDLPS:

```
else if(Procedure->m_Implementation==_T("CSDLOperationProcedureCallmncaVicFunc"))
{
    CSDLOperationProcedureCallmncaVicFunc * Operation=new CSDLOperationProcedureCallmncaVicFunc();
    return LoadOperationProcedureCallAux(name, ID, XMLOperation, Operation, Process, Procedure);
}
else if(Procedure->m_Implementation==_T("CSDLOperationProcedureCallmncaNucFunc"))
{
    CSDLOperationProcedureCallmncaNucFunc * Operation=new CSDLOperationProcedureCallmncaNucFunc();
    return LoadOperationProcedureCallAux(name, ID, XMLOperation, Operation, Process, Procedure);
}
```

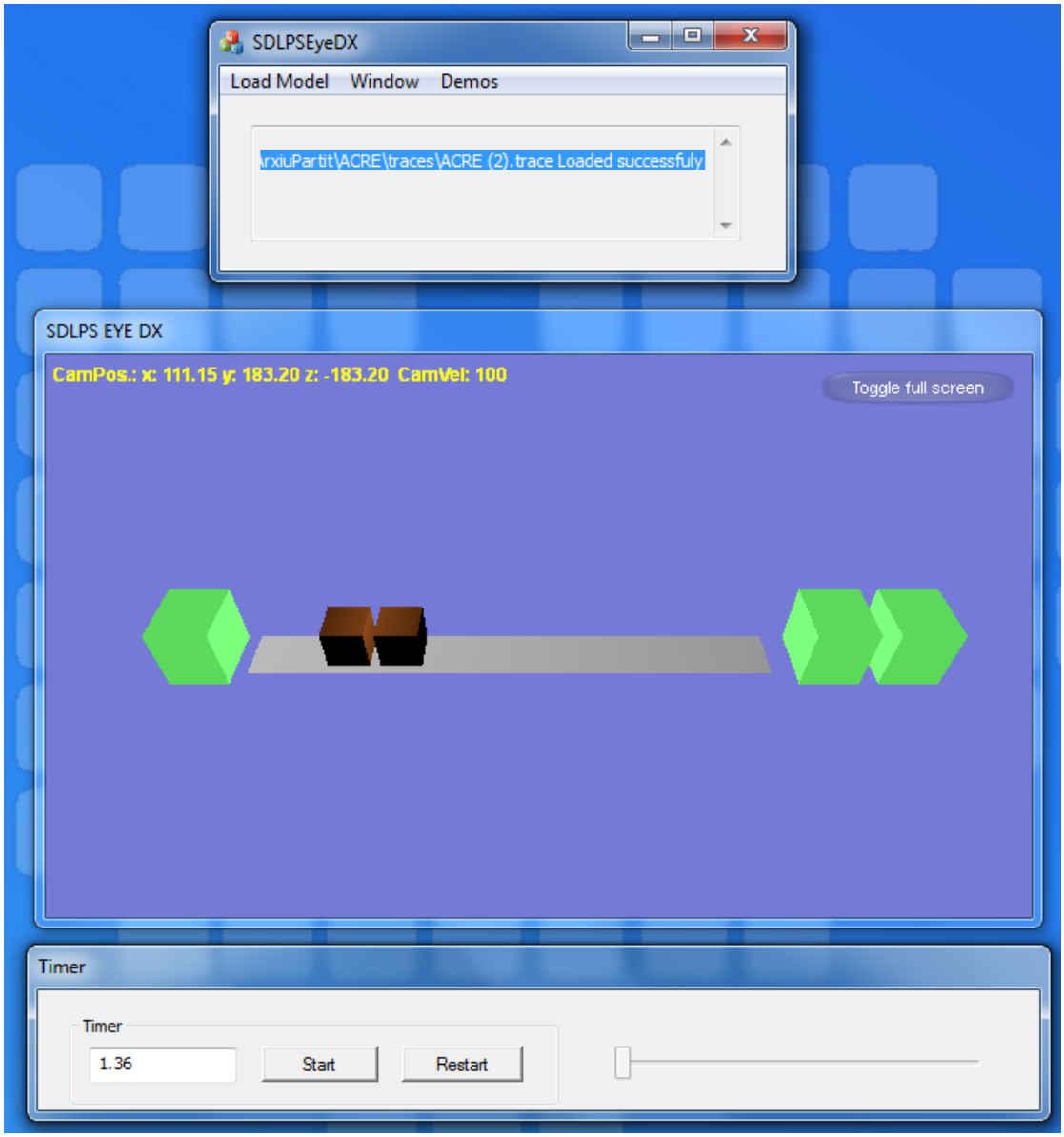
PROCEDURES INTEGRATS A L'SDLPS

Per tal de poder implementar mitjançant diagrames qualsevol funció de veïnatge o nucli, s'han implementat 4 funcions bàsiques a l'SDLPS:

- mncaSetCellValue
- mncaGetCellValue
- mncaGetCurrentCell
- mncaGetSLayerValue

4. MANUAL D'ÚS DE L'EYE DX

L'EyeDX o SDLPSEye és un client per l'SDLPS que interpreta i representa en un espai virtual la traça que genera el simulador.



EVENTS DE REPRESENTACIÓ DE L'EYEDX