5 Disseny

En l'etapa anterior d'especificació s'ha definit quin es el problema a resoldre, la definició del "que", en l'etapa de disseny s'han de prendre les decisions del "com" es resoldrà el problema.

Les decisions s'han de prendre en funció dels recursos disponibles, de les eines que es tenen a disposició i de les especificacions del problema.

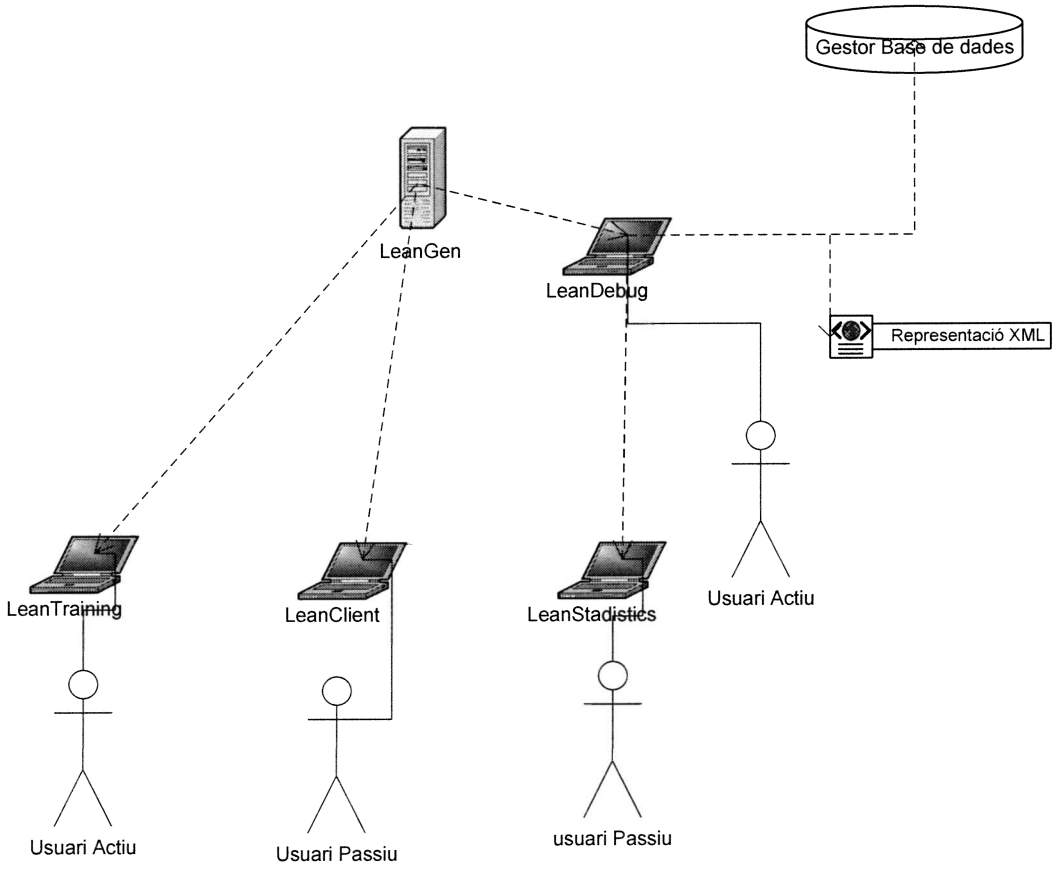
Donat que LeanDebug ha de ser integrat en un sistema ja existent, ja es disposen de certes restriccions de cara al disseny.

A continuació es farà una descripció de l'arquitectura de LeanDebug i com aquesta s'integra dins l'arquitectura de LeanSim.

5.1 Arquitectura LeanDebug

L'arquitectura de LeanDebug s'integra fàcilment dins l'arquitectura de LeanSim degut a que de bon principi aquest ha treballat amb mòduls que interactuen enviant-se missatges, així doncs tenim certa llibertat en el desenvolupament.

En el següent gràfic veiem un esquema de l'arquitectura:

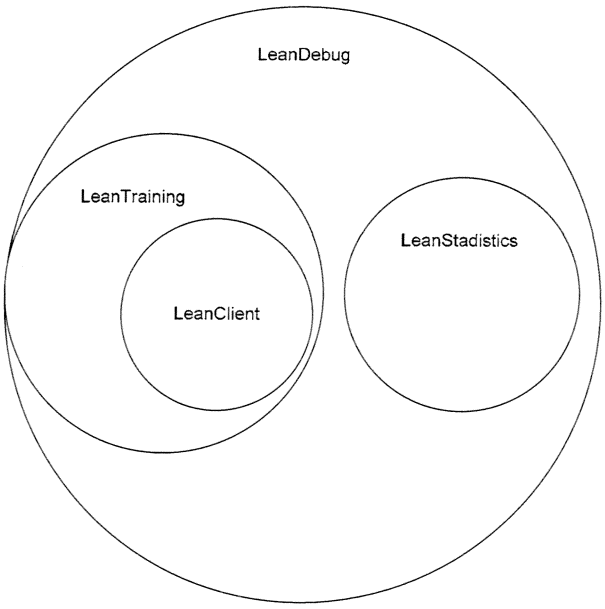


Com podem veure en el gràfic LeanGen es comunica amb tots els seus "clients" mitjançant missatges. Cada client ha de llegir només els missatges que necessita per la seva tasca. Així LeanClient només farà servir aquells missatges relatius a la representació VRML, LeanTraining rebrà també aquests missatges ja que es una versió modificada de LeanClient i LeanStadistics rebrà

tots els missatges d'estadistics que va enviant LeanGen per generar els resultats estadístics de la simulació. LeanDebug ha de rebre tots els missatges que efectuïn qualsevol canvi sobre el model de dades per tal d'anar guardant tots els canvis al gestor de Bases de dades per poder executar RollBack en cas de ser necessari i per anar mostrant en tot moment quins son els atributs i valors de tots els objectes que participen en la simulació per a que l'usuari que està fent el debug pugui trobar els errors.

Com es pot veure en el gràfic hem dividit als usuaris en Usuaris Actius i Usuaris Passius, això es degut a que els usuaris de LeanDebug i LeanTraining interaccionen amb LeanGen, es a dir els usuaris tenen una interacció amb el sistema i poden modificar la simulació. LeanStadistics i LeanClient no tenen cap interacció directe amb LeanGen.

Tant LeanStadistics com LeanClient no tenen cap interacció directe però indirectament sí que poden influir en la simulació, a primera vista sembla impossible ja que no envien cap missatge a LeanGen però de fet aquestes dues eines son un complement a l'hora de fer un debug d'un model de simulació.



Això es deu a que no hi ha prou amb una representació de les dades, amb els atributs, valors i objectes. Si volem tenir una visió global necessitem una representació 3D de les dades que es oferta per LeanClient i uns estadístics que informin de manera resumida el funcionament global del sistema. Així doncs els resultats d'aquests dos mòduls si estem debugant el model si que poden influir en el model i els usuaris que estiguessin treballant amb aquestes eines alertarien d'anomalies detectades.

5.2 Arquitectura Representació de les dades

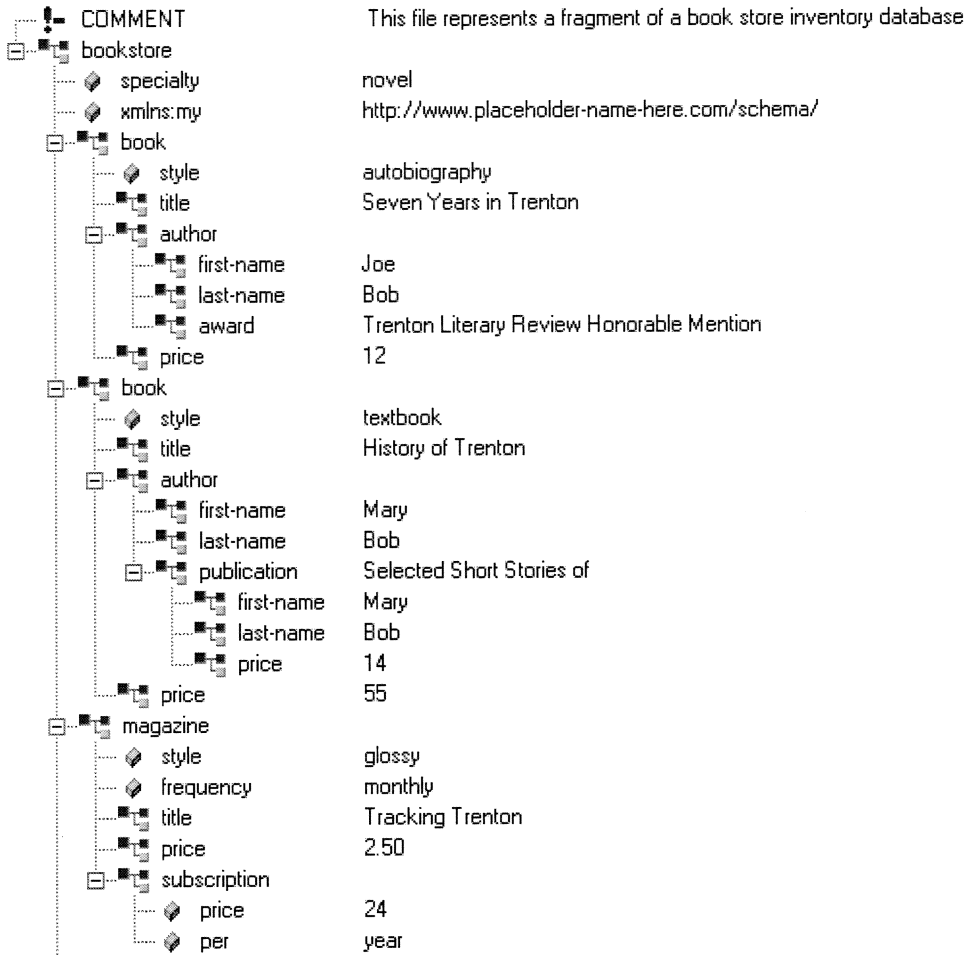
La informació a representar a l'arbre es tot el model de dades, l'estructura de l'arbre resultant és exactament la mateixa que l'estructura del document XML.

Les branques de l'arbre es poden marcar per destacar certs objectes per exemple es podrien marcar totes les que siguin entitats d'un color i les màquines d'un altre o el text de certs atributs més gran o eliminar informació aplicant una màscara. Aquestes marques s'han de programar sobre el mòdul de representació i donat que l'eina de debug no depèn del format del model de dades no es te a priori cap mascara dissenyada, dependria de necessitats específiques per a un tipus de debug.

Per exemple si només es volgués examinar una part del model com per exemple les entitats s'hauria de crear una màscara que només representés els nodes que tinguessin informació sobre les entitats per tan d'eliminar informació innecessària del la representació de les dades i trobar abans els errors.

Es varen avaluar diverses opcions de representació, entre elles cal destacar dues, XSLT i TreeView de .NET.

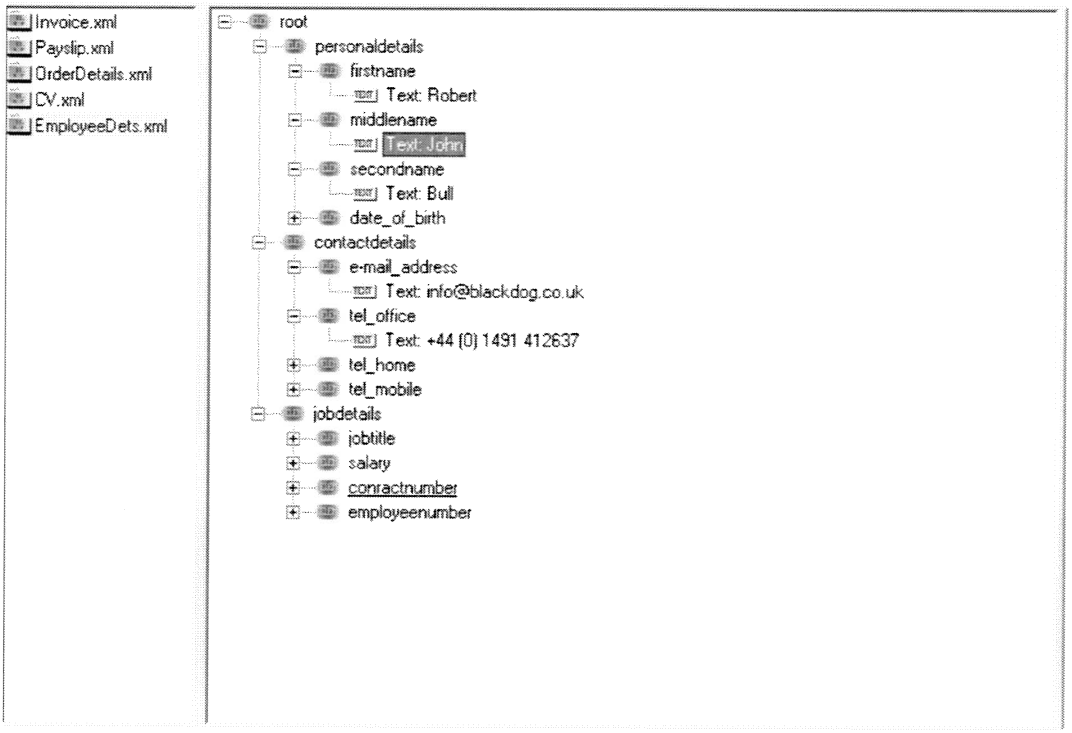
La representació mitjançat XSLT consisteix en agafar el model de dades XML aplicar una fulla XSLT de transformacions sobre l'XML i obtenir com a resultant un document HTML amb la informació en format d'arbre.



Aquesta opció es molt bona doncs dona grans avantatges de portabilitat donat que si es fa amb javascript i html es portable a tots els sistemes que implementin aquestes especificacions.

La pega es el rendiment doncs si tenim un model molt extens el navegador no pot mostrar tota la informació sense consumir molts recursos a més de haver de carregar tota la representació a memòria.

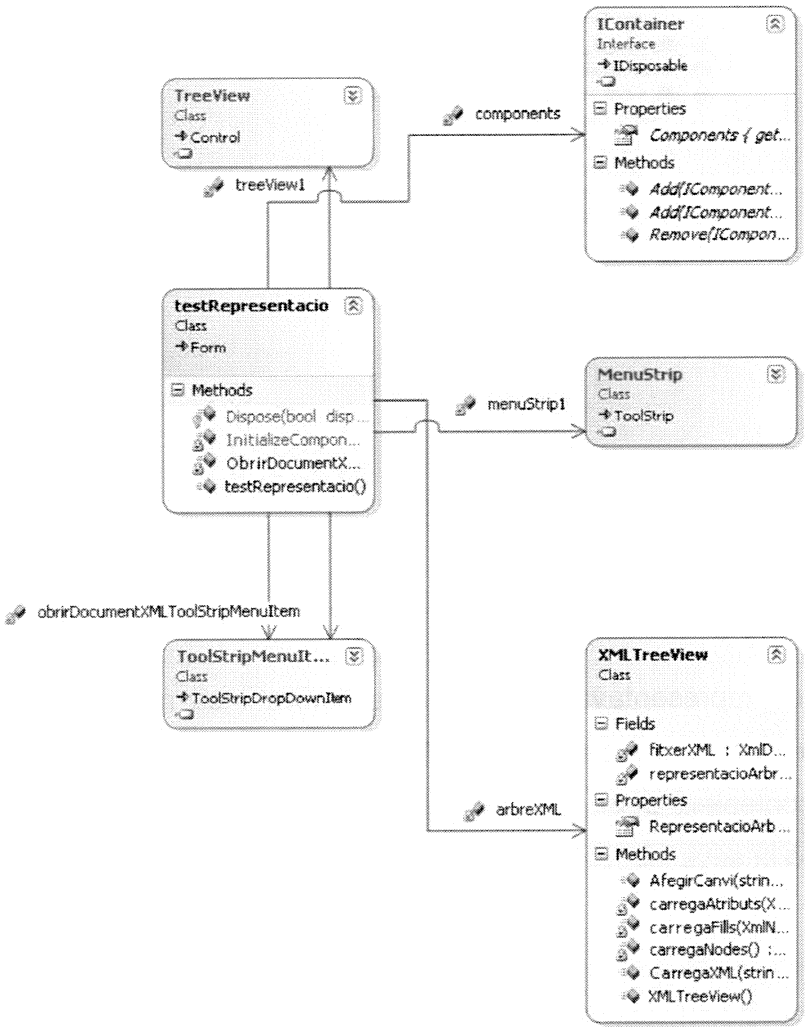
L'altre opció es l'objecte per representar arbres que aporta .NET, el TreeView. Aquest objecte consumeix molts menys recursos, es molt ràpid i a més hi ha algunes implementacions que poden carregar els nodes dinàmicament, es a dir, permet carregar els fills d'un node sense carregar-ne els successors, només carregar-los a memòria si es desplega el node.



La informació es representava ràpidament a l'arbre i es molt fàcil realitzar cerques dins de l'arbre, a més es va implementar una petita cerca per trobar quins nodes contenien una paraula concreta, de manera que quedessin desplegats per a la seva inspecció.

Es va pensar en introduir icones que denotessin la tipologia del node però donat que un arbre XML pot ser extremadament llarg es va descartar per l'ús de recursos de memòria innecessaris.

A continuació es mostra la implementació feta per testejar l'objecte TreeView:



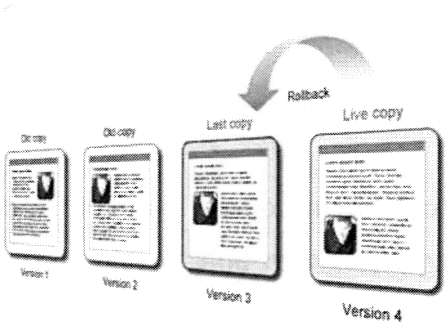
Aquest component no pot ser gaire més específic donat que es vol desvincular de quin sigui el format de les dades, així un canvi en el model de dades o la addició o supressió d'informació en la estructura estàtica del model no suposaria cap canvi en el component.

El seu objectiu es mostrar tota la informació, que sigui fàcilment accessible i donar facilitats per trobar-la

.

5.3 Arquitectura RollBack

Denotem per RollBack la capacitat del sistema per tornar a un estat anterior. Aquesta funcionalitat es fonamental a l'hora de debugar un model de simulació per poder aïllar comportaments del model en estats concrets, retrocedir en el temps de simulació i poder inspeccionar com i perquè evoluciona d'aquesta manera.



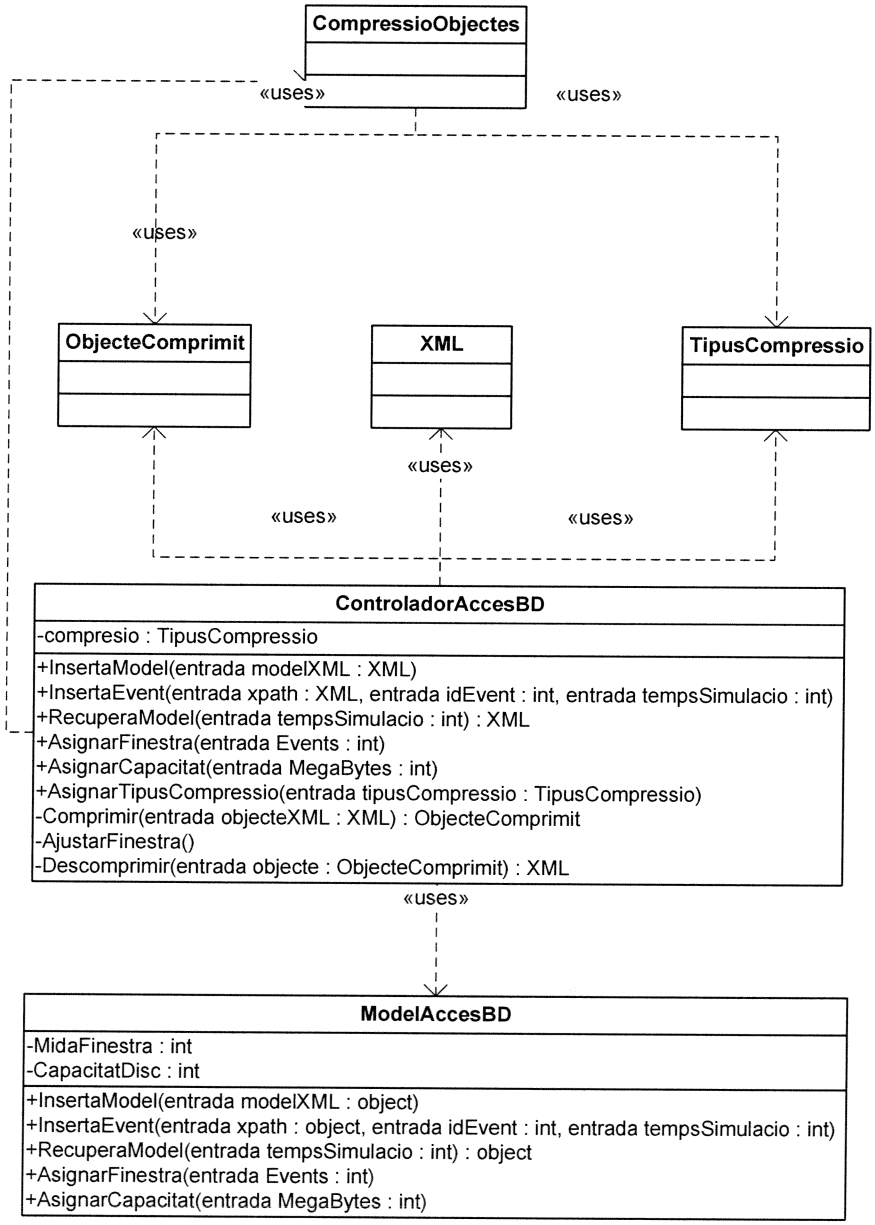
La solució adoptada ha de ser molt flexible, doncs amb una solució estàtica ens arrisquem a que no sigui vàlida per un model concret de simulació o per un hardware determinat.

La informàtica, les telecomunicacions i l'electrònica evolucionen molt ràpidament i si fa uns pocs anys els ordinadors disposaven de poca capacitat de disc, de memòria i les xarxes no superaven els 10Mb/s actualment aquestes capacitats i velocitats són més de 100 vegades més ràpides.

El problema amb els models de simulació es que les seves execucions poden generar molts missatges degut a que s'executen molts esdeveniments. Com més missatges s'emmagatzemen més capacitat de disc es necessita i això pot suposar un problema si se supera aquesta capacitat.

5.3.1 Diagrama RollBack

A continuació es mostra el diagrama de classes dissenyat per suportar l'arquitectura de RollBack:



5.3.1.1 ObjecteComprimit

Aquesta classe representa un objecte comprimit en aquest cas es tracta de un model XML o una sentència XQuery, en qualsevol dels dos casos es tracta del format XML.

El disseny d'aquest mòdul queda fora de l'abast d'aquest projecte, existeixen moltes implementacions d'objectes comprimits en codi lliure, tant en c# com java o altres llenguatges de programació.

L'objecte seria el resultat de la compressió de l'XML amb el tipus de compressió seleccionada, per ser emmagatzemat en la base de dades.

5.3.1.2 XML

Aquest mòdul encapsula una representació del llenguatge XML, no s'ha fet un disseny en profunditat perquè es faria servir qualsevol de les representacions que tenim a qualsevol llenguatge de programació d'aquest objecte.

5.3.1.3 TipusCompressio

Aquest objecte encapsularà els tipus de compressió definits per emmagatzemar la informació a la base de dades i estalviar espai.

Es important fer un estudi del cost de procés de la compressió i de l'espai de disc estalviat doncs es podrien disposar de diversos tipus de compressió i escollir el més adient en funció de les capacitats dels recursos informàtics disponibles.

També està la possibilitat de generar una compressió pròpia donat que la majoria dels missatges tindran moltes parts comunes i es repetiran sovint, a més el nombre tipologies d'esdeveniments generables es coneguda i reduïda.

5.3.1.4 ControladorAccesBD

Aquesta classe es el controlador de model de dades i encapsularà totes les funcionalitats d'accés a la Base de dades de manera que si canviés el gestor de base de dades o si hi haguessin nous mètodes de compressió només s'hauria de modificar el controlador i els canvis serien transparents.

La informació principal que gestiona la base de dades es el model de dades XML de simulació i tots els missatges de modificació sobre aquest model que es van guarden per poder realitzar posteriorment el RollBack del sistema. Es imprescindible disposar d'una base de dades doncs aquesta ja aporta un tipus de dades XML i a més pot executar directament les sentències XQuery sobre les columnes de tipus XML obtenint com a resultat el RollBack del sistema. La simulació genera milions de missatges i sense el gestor de base de dades e índex sobre el missatges guardats la recuperació de la informació seria massa costosa.

5.3.1.4.1 Atributs

5.3.1.4.1.1 compresio

Aquest atribut es l'indicador del tipus de compressió si es vol un rendiment màxim aquí s'haurà d'indicar que no es vol cap tipus de compressió, per defecte el seu valor indicarà que no hi ha compressió.

5.3.1.4.2 Mètodes

5.3.1.4.2.1 InsertaModel

Aquest mètode rep el model XML comprovarà si es vol cap tipus de compressió i s'encarregarà de comprimir el model si s'escau i cridarà al mètode del model de dades encarregat de guardar el model de simulació a la base de dades.

5.3.1.4.2.2 InsertaEvent

Amb aquesta funcionalitat es podran emmagatzemar els canvis generats per un esdeveniment en format XQuery, tal i com fa el mètode anterior haurà de comprovar si requereix compressió i cridar al mètode corresponent del model de dades per emmagatzemar la sentència XQuery a la base de dades.

5.3.1.4.2.3 RecuperaModel

Aquesta funcionalitat serà l'encarregada de recuperar el model de dades, una vegada s'hagi recollit la informació del model de dades s'haurà de recuperar el model i executar totes les sentències XQuery posteriors.

Si hi havia algun tipus de compressió s'han d'executar les operacions necessàries per obtenir les dades en format XML.

5.3.1.4.2.4 AsignarFinestra

S'assigna la mida de la finestra de dades, és a dir, marcarem quin es la distància màxima entre el model de dades XML i l'últim esdeveniment de la simulació.

5.3.1.4.2.5 AsignarCapacitat

Marcarà la capacitat màxima de la base de dades que conté el model de simulació. Hi haurà la opció de deixar la capacitat il•limitada a risc de l'usuari. Si es sobrepassa aquesta capacitat hi haurien dos alternatives d'acció:

- Reduir la mida de la finestra.

- Tornar a calcular el nou model de dades XML fins que es tingui una capacitat d'un cert percentatge i encara hi hagi una finestra de dades amb la que poder jugar.

5.3.1.4.2.6 AsignarTipusCompressio

Es marcarà quin es el tipus de compressió desitjat tenint en compte el rendiment del sistema i la capacitat de disc.

5.3.1.4.2.7 Comprimir

La seva funció es comprimir les dades XML, la implementació i disseny de les rutines de compressió queden fora de l'abast d'aquest projecte, actualment hi ha molts patrons de compressió de dades i també s'ha de tenir en compte fer-ne un de propi donat que els esdeveniments del sistema son limitats i hi ha molt de codi en els missatges que es repeteix.

5.3.1.4.2.8 AjustarFinestra

Aquesta funcionalitat s'encarregarà d'ajustar la mida de la finestra quan es detecti que es sobrepassarà.

El seu funcionament consisteix en obtenir l'últim model XMI, recuperar els esdeveniments posteriors i executar totes les sentencies XQuery contra el model per obtenir la nova versió del model i així reduir temps de procés posterior en un RollBack.

5.3.1.4.2.9 Descomprimir

La funció descomprimir tal i com indica el seu nom té la tasca de donat un objecte comprimit i revisant quina es la compressió seleccionada descomprimir l'objecte i extreure'n l'objecte XML.

5.3.1.5 ModelAccesBD

Aquesta classe es la que s'encarrega dels accessos directes contra el gestor de bases de dades, si hi hagués algun canvi en el gestor de bases de dades seria aquí on s'haurien de fer els canvis.

Aportarà totes les funcionalitats bàsiques d'emmagatzemament per a que les dades tingui un repositori i es puguin recuperar per executar el RollBack.

5.3.1.5.1 Atributs

5.3.1.5.1.1 MidaFinestra

Aquest atribut marcarà quina es la mida de la finestra d'esdeveniments que s'ha d'enmagatzemar a la base de dades.

5.3.1.5.1.2 CapacitatDisc

Aquest atribut marca`r quina es la capacitat màxima que es farà servir per la base de dades de RollBack. La mida serà introduïda en MegaBytes, una vegada modificat aquest valor no es podrà tornar a modificar fins que no finalitzi la simulació.

5.3.1.5.2 Mètodes

5.3.1.5.2.1 InsertaModel

Aquesta funcionalitat s'encarregarà d'atacar a la base de dades per per introduir el model de simulació inicial al sistema. També introduirà el model cada vegada que es necessiti degut a la mida de la finestra.

5.3.1.5.2.2 InsertaEvent

Realitza la inserció d'un esdeveniment a la base de dades.

5.3.1.5.2.3 RecuperaModel

Aquesta funcionalitat recupera el model de dades donat un temps de simulació. S'haurà de recuperar el model guardat del temps de simulació sol•licitat o bé el més proper dels anteriors i tots els esdeveniments posteriors que no sobrepassin el temps de simulació per tal que es pugui fer la reconstrucció del model executant les sentències XQuery.

Una vegada recuperat el model XML les execucions de les sentències XQuery es faran contra la base de dades ja que aquesta aporta totes les funcionalitats d'execució de sentències XQuery contra el tipus de dades xml.

5.3.1.5.2.4 AsignarFinestra

Assigna el valor de la finestra de esdeveniments.

5.3.1.5.2.5 AsignarCapacitat

Assigna la capacitat màxima de disc que pot ocupar la base de dades en MegaBytes.

5.4 Arquitectura missatgeria

Com ja s'ha comentat en capítols anteriors un dels objectius de LeanDebug es dissenyar l'arquitectura dels missatges que s'envien als diferents clients.

Es podrien dividir els missatges en les següents categories:

- **Connexió.(inici, final,...)**
- **Control de la simulació.(aturar, reiniciar...)**
- **Modificacions sobre el model.**

Entre totes les tipologies de missatges ens centrarem sobretot en els que modifiquen el model doncs aquests seran els que es re-dissenyaran per poder integrar LeanDebug.

5.4.1 Connexió

Aquesta categoria de missatges poden ser no modificats ja que no suposa grans avantatges sobre el missatge de connexió actual. Poder lo més interessant seria fer un protocol de missatgeria de connexió homogeni per a tots els clients.

Actualment LeanStadistics te un protocol de connexió i LeanClient i LeanTraining tenen un altre de diferent.

Els missatges de connexió haurien de contenir la següent informació:

- **Protocol inicial de connexió per sockets.**
- **Autentificació de l'usuari.**
- **Tipologia del client (LeanClient, LeanDebug, LeanStadistics, LeanTraining).**
- **Model de simulació (només pels clients que ho necessitin).**

5.4.2 Control de la simulació

Aquests missatges tampoc cal que canviïn gaire, això es degut a que només LeanClient i per tant LeanTraining tenen accés a aquest tipus de control. LeanDebug també ha de tenir accés al control de la simulació.

L'única modificació es l'adició del missatge de RollBack que s'explica a continuació.

Els missatges bàsics que ha de tenir aquesta tipologia son:

- **Engegar simulació.**

Aquest missatge donaria la ordre per a que LeanGen comencés a fer la simulació del model i enviar tots els missatges que suposin un canvi sobre el model de simulació.

- **Parada de la simulació.**

Aquest missatge atura la simulació definitivament i torna a l'estat inicial.

- **Pausa de la simulació.**

Amb aquest missatge s'aconsegueix que LeanGen pari de simular i es aqui on podem revisar l'estat dels objectes de sistema i veure quins valors prenen per tal de fer debug.

- **RollBack.**

Aquesta comanda només es posible que s'executi si l'estat de la simulació es en pausa ja que hauria d'indicar aquest missatge el model recuperat a carregar per LeanGen i els seus Clients i quin es el temps de simulació recuperat.

5.4.3 Modificacions sobre el model

Després de valorar les diferents implementacions que s'han fet sobre la informació enviada des de LeanGen als seus clients la solució escollida serà la de enviar les modificacions directes sobre el model XML amb sentencies XQuery.

L'elecció d'aquesta configuració dels missatges es degut que mitjançant sentencies XQuery es tenen exactament quines son les modificacions realitzades sobre el model XML i es pot saber en tot moment quin es l'estat exacte de tots els objectes del sistema.

A més es reduiran el nombre de missatges enviats i la mida dels missatges guanyant en capacitat de disc i de processament.

5.4.4 Format dels missatges

Un dels punt clau es el format dels missatges ja que afectarà a tots els components. Com ja s'ha comentat anteriorment un dels punts crítics a tota simulació es la quantitat d'informació que es genera. Per aquest motiu dos punts clau en el format dels missatges son:

- **Minimització del nombre de missatges.**
- **Minimització de la mida del missatge.**

A diferencia de la metodologia que hi havia fins ara on el motor deia a cada client quin era el resultat amb el que havia de treballar el client, en el nou format només s'informaran sobre els canvis sobre el model XML i serà el client qui els haurà d'interpretar.

Així si fins ara el motor li deia al client de representació quines eren les accions a pintar ara li dirà quin es el canvi i serà el client qui decidirà que pinta i com ho pinta.

El client d'estadístics hauria de ser l'encarregat de recollir els estadístics i no el motor de elaborar-los i enviar-li.

Sembla que hi hagi molta feina nova però es tracta de moure funcionalitats del motor al client i així també aconseguim un millor rendiment al motor de simulació descarregant-lo de feina.

La informació mínima que es necessita es un identificador de missatge , ja que els missatges han d'arribar sempre per ordre, la sentència XQuery que hauria d'executar el client per veure els canvis i el temps de simulació on s'executa aquest canvi.

D'aquesta manera també minimitzem molt el nombre de missatges, ja que ara si un objecte provoca canvis a 100 objectes amb un sol esdeveniment no s'han d'enviar 100 missatges si no que amb un de sol que indiqui a qui se li ha de fer la modificació i quina és, cada client interpretaria els canvis i executaria l'acció pertinent.

El format del missatge seria doncs:

```
< id_missatge, temps_simulacio, XQuery >
```

Les llibreries actuals de XML tenen mètodes per esbrinar quines son les branques de l'arbre modificades després d'executar una sentencia XQuery així doncs no hi haurà gaires complicacions a l'hora d'esbrinar els canvis.

Els canvis que suposa aquesta nova implementació de la missatgeria son bastant crítics i suposa un canvi en la metodologia de treballar dels clients però no en el paradigma de treball de LeanSim ja que s'ha fet un esforç molt gran per distribuir la feina a tots els clients i considero que aportarà grans millores.

A més es deslligarà al motor de molta feina i si s'han de fer canvis com per exemple, que es vulgui mostrar visualment amb VRML un nou canvi no haurà de ser el motor qui generi el canvi i enviï un nou missatge de pintat, les modificacions es remeteren als clients.

Aquest canvi en l'arquitectura de la missatgeria aportarà que el motor de simulacio no hagi d'implementar més canvis cada vegada que hi ha un client nou o que algun d'aquest necessita canvis en els missatges i d'aquesta manera es més facil tancar la seva implementació.

La descripció del format dels missatges en terminologia DTD es la següent:

```
<!ELEMENT Missatge (id_missatge,temps_simulacio,XQuery)>
<!ELEMENT temps_simulacio (#PCDATA)>
<!ELEMENT XQuery (#PCDATA)>
<!ELEMENT id_missatge (#PCDATA)>
```

5.4.5 Exemple de missatgeria

Suposem el següent següent model XML:

```
<Root>
<Location LocationID="10"
            LaborHours="1.1"
            MachineHours=".2" >Manufacturing steps are described
here.
<step>Manufacturing step 1 at this work center</step>
<step>Manufacturing step 2 at this work center</step>
</Location>
</Root>

<Maquina TipusMaquina="Generica" NumeroInstancies="1"
Nom="Servidor4" Categoria="Generica" IndexCategoria="1"
Movible="true" id="3" MultiInstancia="false"
NomUsuari="Servidor4">
<Instancies>
<Generic Numero="0" Tipus="1" id="4">
  <Calendari Valor="Simulator.sch" />
<Representacio>
  <VRML
Valor="C:\LCFIB\LeanSim\LeanEditor\Environment\Mesh\demorarobotw
rl.WRL" />
  <Posicio3D PosX="47.0183838627854" Offset="0" PosY="0"
PosZ="52.5850465032009" Nivell="String" />
  <Orientacio y="0" z="0" x="0" />
  <Escalat y="0" z="1" x="1" />
  <Velocitat Valor="0" />
<Vectorial>
  <Punt PosX="15" PosZ="165" />
  <Punt PosX="15" PosZ="120" />
  <Punt PosX="30" PosZ="120" />
  <Punt PosX="45" PosZ="105" />
  <Punt PosX="255" PosZ="105" />
  <Punt PosX="270" PosZ="120" />
  <Punt PosX="285" PosZ="120" />
  <Punt PosX="285" PosZ="165" />
  <Punt PosX="270" PosZ="165" />
  <Punt PosX="255" PosZ="180" />
  <Punt PosX="45" PosZ="180" />
  <Punt PosX="30" PosZ="165" />
  <Punt PosX="15" PosZ="165" />
  <Color>00FF00</Color>
  </Vectorial>
  </Representacio>
<EstatisticsEstat Valor="true">
  <Descripcio>String</Descripcio>
  </EstatisticsEstat>
<Caracteristiques>
```

```
  <EntitatsForaServei Valor="false" />
  <EntitatsEnAveria Valor="true" />
  <PerdEntitatReparacio Valor="true" />
  <PerdEntitatBloqueig Valor="false" />
  <AturarEnBloqueig Valor="false" />
  <SenyalCuaEntrada Valor="false" />
  <SenyalCuaSortida Valor="false" />
  <SenyalForceSetup Valor="false" />
  <Avaries />
  <Atributs />
  <Estadistics LongCues="true" TempsCues="true" />
  <PoliticaCues Politica="0" PerLots="false" />
<CuesEntrada>
<Cua Nom="Buffer Entrada">
  <Capacitat Valor="10" />
  <Lot Tamany="0" />
  <Politica Valor="0" />
  <Prioritat Valor="0" />
  <PerdEntitat Valor="1" />
  </Cua>
  </CuesEntrada>
<CuaSortida Nom="Buffer Salida">
  <Capacitat Valor="0" />
  <Lot Tamany="0" />
  <Politica Valor="0" />
  <Prioritat Valor="0" />
  <PerdEntitat Valor="false" />
  </CuaSortida>
  <PulledEntity />
  <Transicions />
<Indicadors>
<BufferIn tipus="0">
<Entrada id="Buffer Entrada" visible="0">
  <Rectangle top="-0.5" left="0.5" bottom="0" right="0" />
  </Entrada>
  </BufferIn>
  </Indicadors>
  </Caracteristiques>
  </Generic>
  </Instancies>
<Operacions>
----------------  </Maquina>
```

Si es generes un event que modifiques el valor de l'atribut LaborHours de Location a 100.0 el el temps de simulació 1 i aquest fos el primer missatge el missatge resultant seria:

```
<Missatge>
<id_missatge>1</id_missatge>
<temps_simulacio>1</temps_simulacio>
```

```
        <XQurey>replace value of (/Root/Location/@LaborHours)[1]
with "100.0"          </XQuery>
</Missatge>
```

Si el següent missatge fos a l'instant 2 i el missatge fes una inserció dins el node location el missatge resultant seria el següent:

```
<Missatge>
<id_missatge>2</id_missatge>
<temps_simulacio>2</temps_simulacio>
        <XQurey>insert <test></test> into (/Root/Location)[1]
           </XQuery>
</Missatge>
```

Per últim si encara en el segon instant de simulació es generés un nou event que elimines el primer node <step> del node location el missatge seria el següent:

```
<Missatge>
<id_missatge>3</id_missatge>
<temps_simulacio>2</temps_simulacio>
        <XQurey>delete /Root/Location/step[2]
           </XQuery>
</Missatge>
```

5.4.6 Canvis en el model de dades

Per poder tenir un control total de la simulació cal fer alguns canvis sobre el model de dades actual per tal que el document XML contingui exactament tot el que està succeint en la simulació.

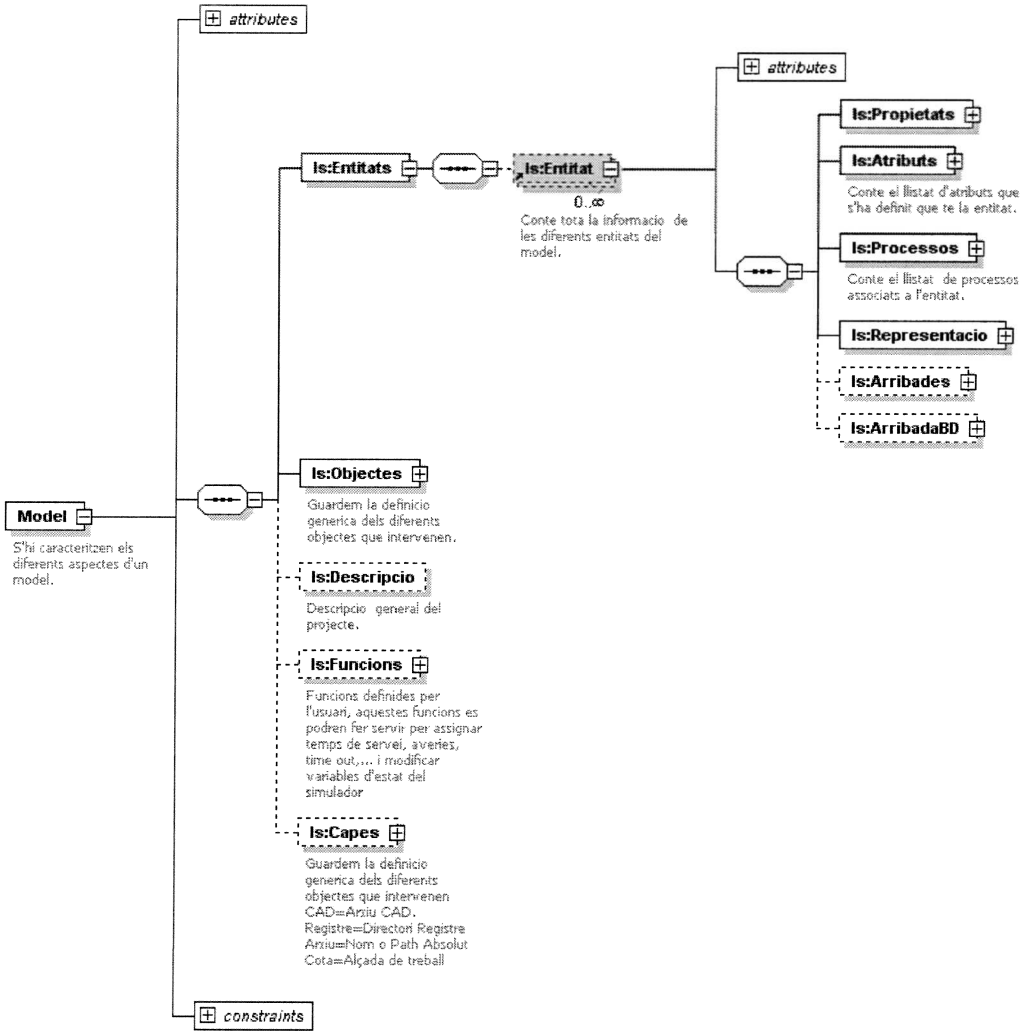
Actualment no calen aquests canvis donat que LeanGen treballa amb aquestes dades però les emmagatzema a memòria i no reflecteix els canvis en el document XMl però com cap client en feia ús no es tenia la necessitat. Amb l'aparició de LeanDebug cal que LeanGen generi una replica exacta de l'estat de la simulació en tot moment sobre el document XML.

Els canvis necessaris són els següents:

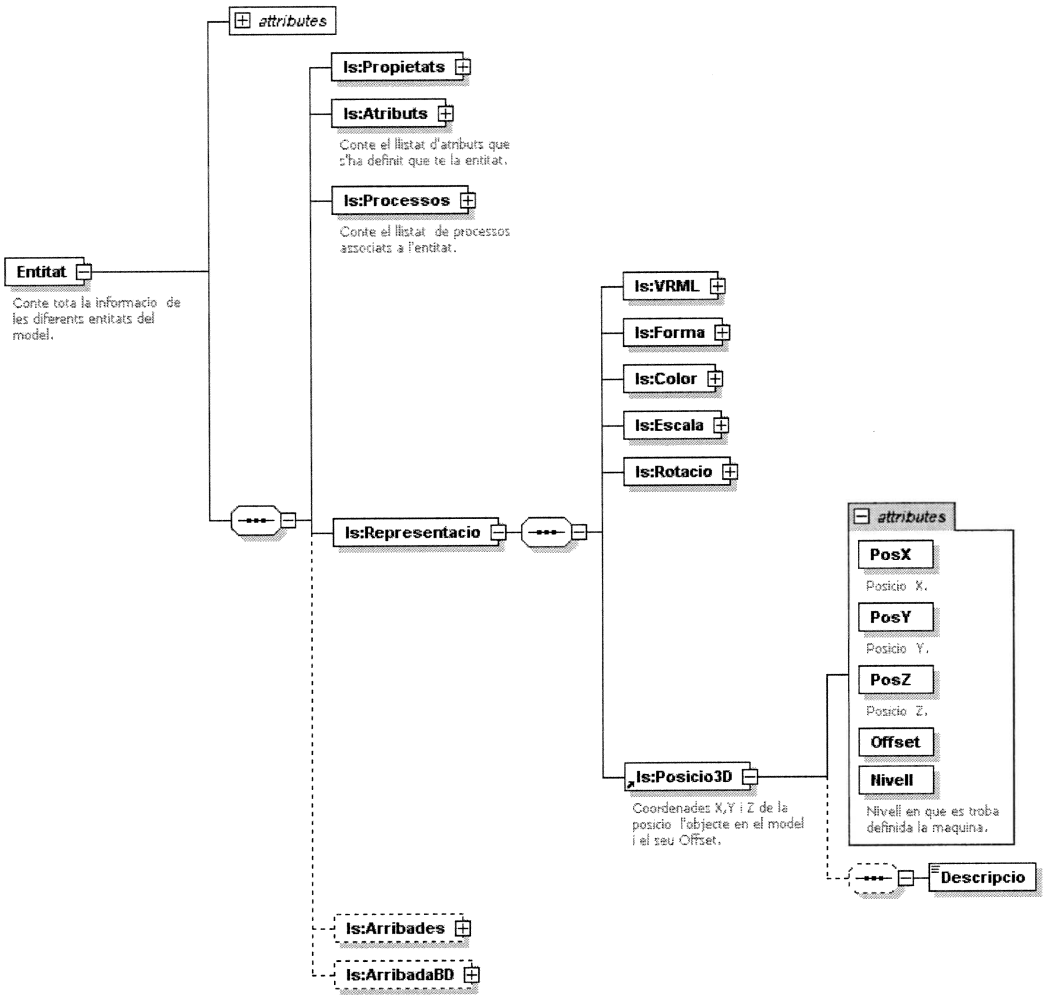
- **Permetre guardar totes les entitats del model.**
- **Posició de les entitats.**
- **Estats de les màquines.**
- **Entitats servint-se a cada màquina.**
- **Temps de servei de cada entitat.**
- **Temps transcorregut de l'entitat a la màquina.**
- **Quantitat d'elements en una cua.**
- **Ordre de les cues.**
- **Llista d'esdeveniments.**

A continuació es mostren els dissenys de l'arxiu XSD desenvolupats per tal de soportar aquests canvis:

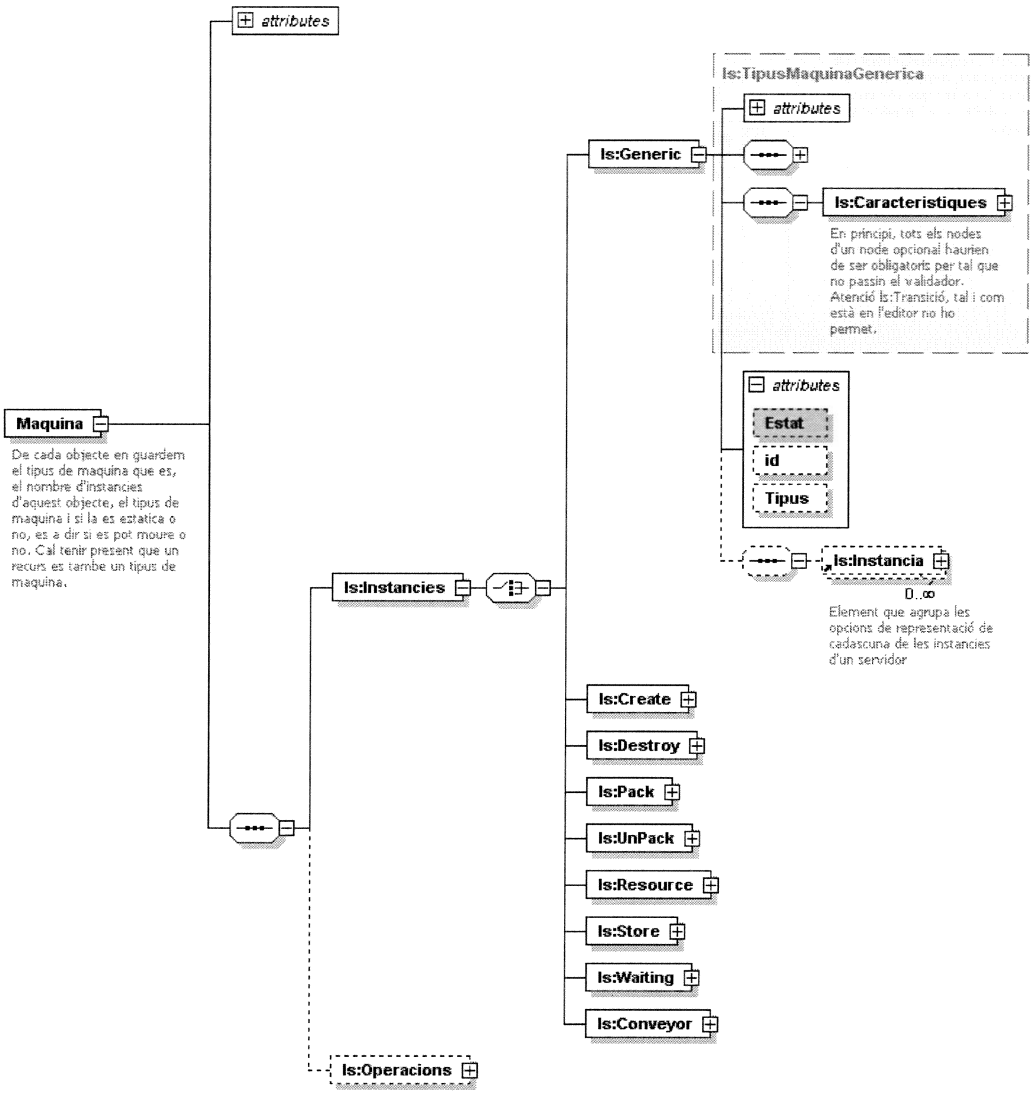
- **Entitats en el model**



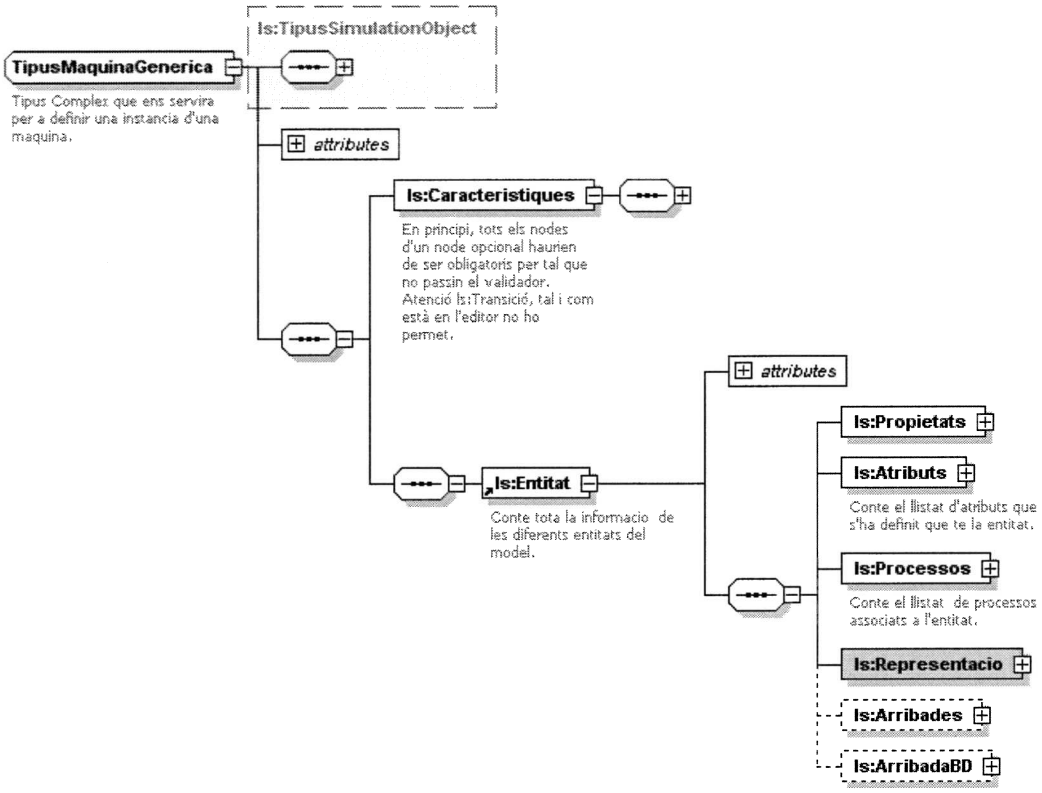
- **Posició de les entitats**



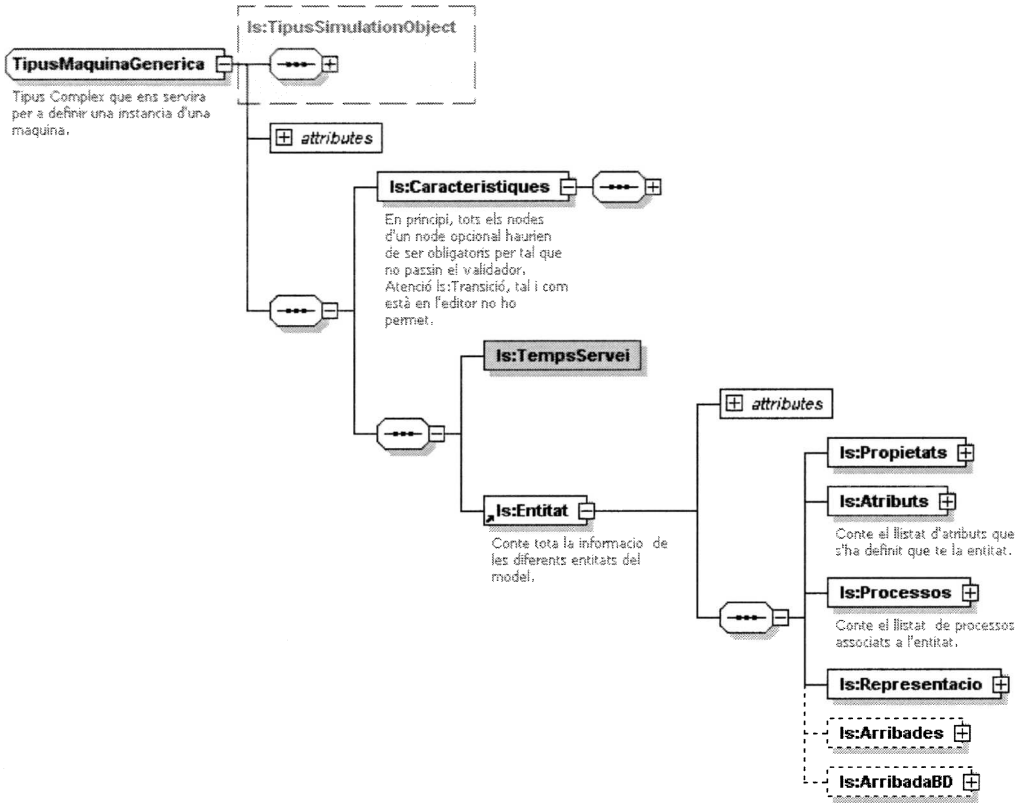
- **Estats de les màquines.**



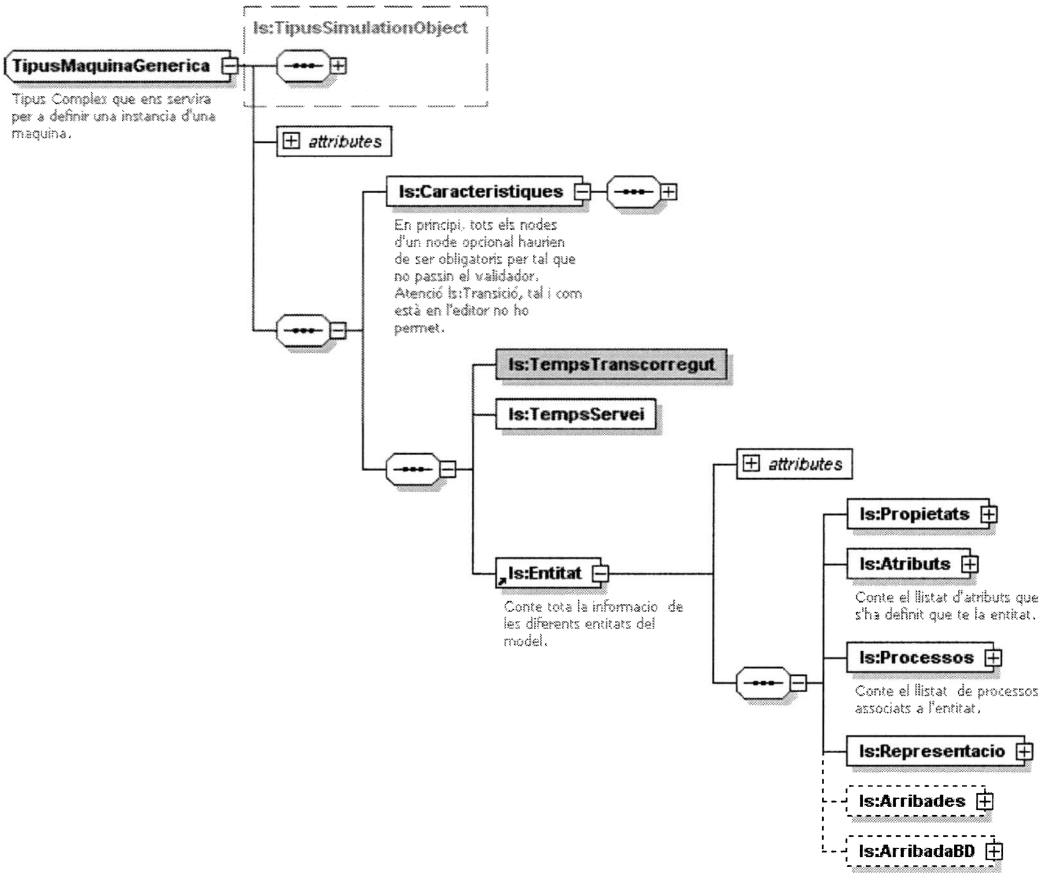
- **Entitats servint-se a cada màquina.**



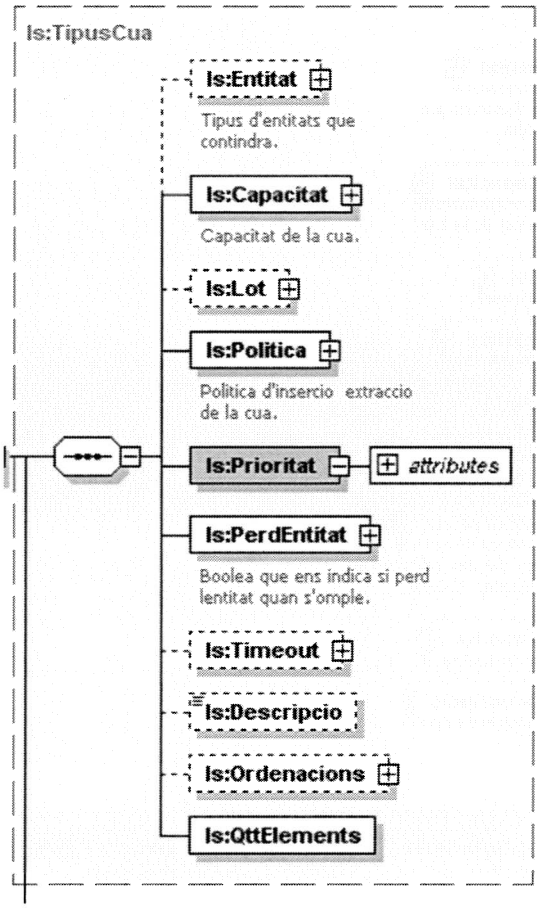
- **Temps de servei de cada entitat.**



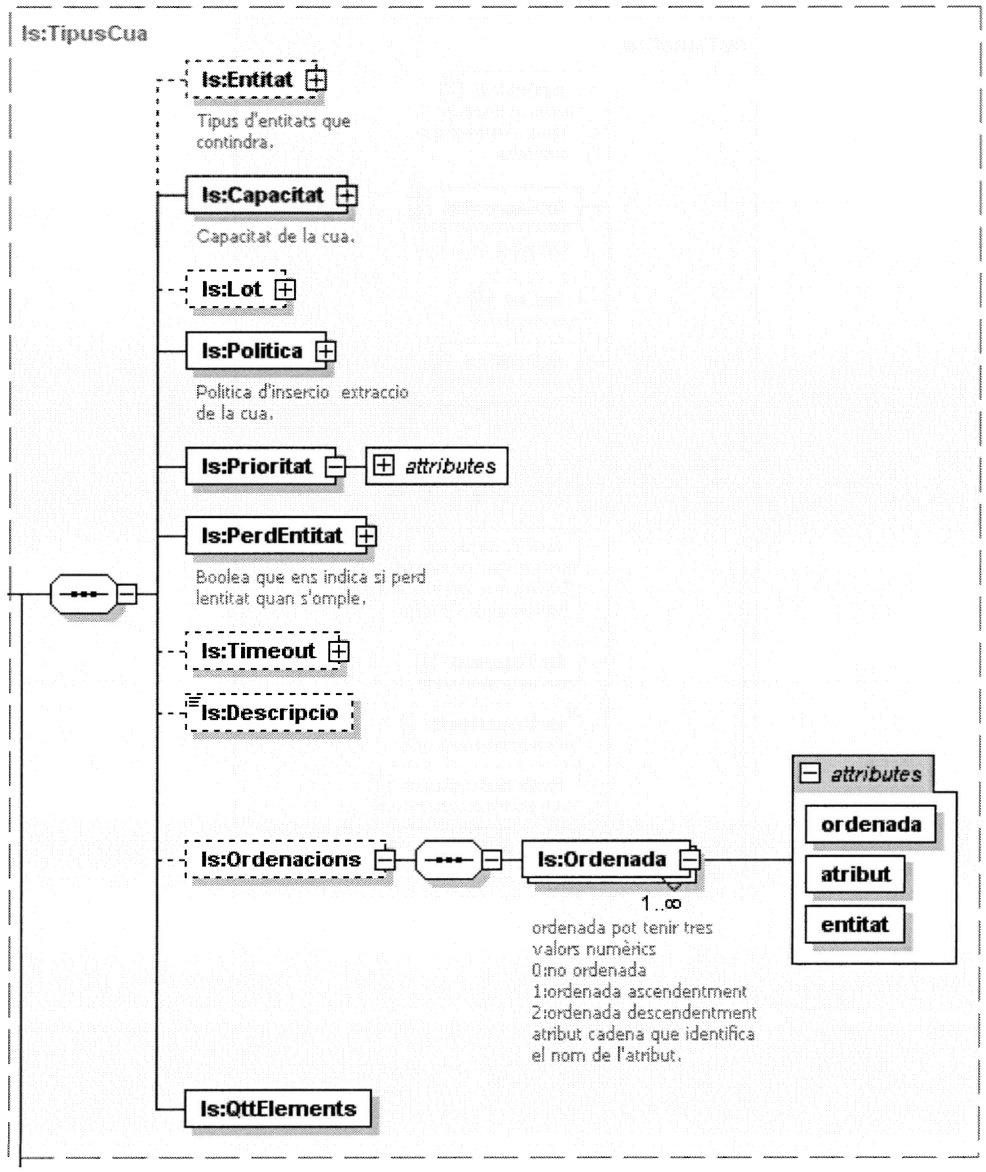
- **Temps transcorregut de l'entitat a la màquina.**



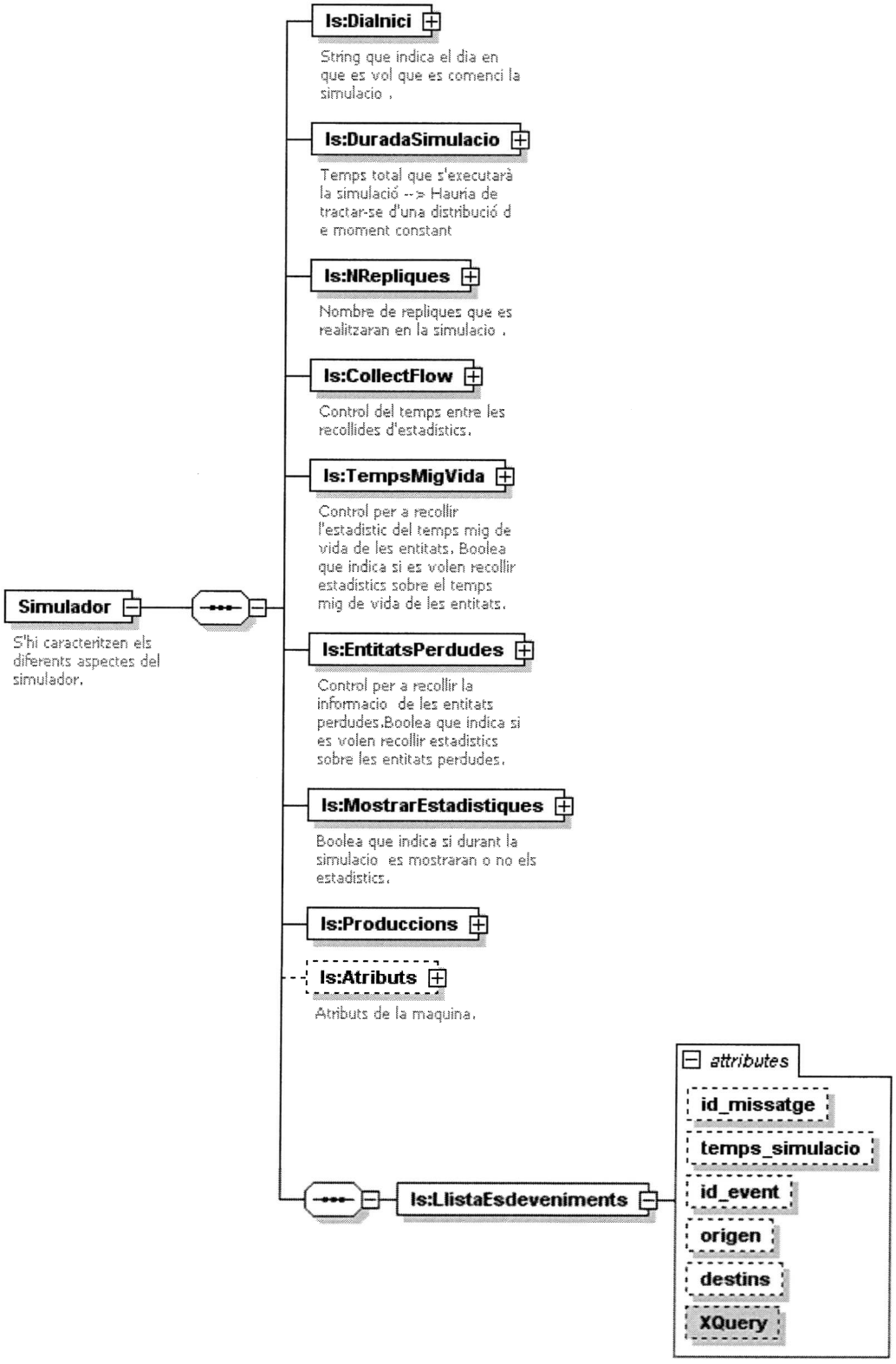
- **Quantitat d'elements en una cua.**



- **Ordre de les cues.**



- **Llista d'esdeveniments.**



6 Implementació: Modificació a LeanStadistics per depurar

Per tal de poder gaudir de una primera aproximació de la funcionalitat de debug s'han fet una sèrie de modificacions a l'eina LeanStadistics per tal de fer servir l'eina per poder ser utilitzada a mode de debug.

S'ha decidit fer la modificació sobre aquesta eina ja que incorpora resultats estadístics que poden ser de gran ajuda a l'hora de trobar possibles bugs al modelat.

Les funcionalitats que s'han incorporat son una inspecció de tots els missatges del sistema, control sobre la simulació, edició dels missatges i rollback.

A continuació es detallen els canvis que s'han fet sobre el sistema per tal de soportar aquestes funcionalitats:

6.1 Inspecció dels missatges del sistema

Per tal de poder revisar els missatges del sistema el primer que s'ha fet es una modificació sobre LeanGen per tal que enviï tots els missatges del sistema per tal de poder ser inspeccionats per l'eina de debug.

Per tal de diferenciar els missatges se'ls ha donat un identificar per a que els clients que no vulguin revisar aquesta informació la puguin descartar.
El següent punt es modificar el mètode que rep missatges TCP/IP de LeanStadistics per tal de detectar els missatges de debug i fer un tractament especial.

El tractament consisteix en guardar en una base de dades tota la informació de debug per poder explotar-la.

Una vegada hem pausat la simulació o ha finalitzat podrem veure un nombre de registres de la base de dades de debug polsant sobre el node pare de l'arbre que disposa LeanStadistics. Això es gracies a una modificació sobre la vista (LeftView) i aprofitant el codi que ens permet mostrar la informació en una taula sobre el marc dret.

Una vegada examinats podem moure'ns endavant i enrere en els diferents registres amb la finestra que ja ens proporciona LeanStadistics per a tal fi.

6.2 Control de la simulació

S'han afegit nous botons per tal de poder pausar, iniciar i aturar la simulació i així poder fer revisar les dades.

El missatges son els mateixos que envia LeanClient al motor per controlar la simulació.

Una vegada pausada la simulació LeanStadistics ha d'esperar un temps fins que LeanGen li envia un missatge conforme s'ha pausat ja que des de que premem el botó fins que para pot passar cert temps.

6.3 Edició dels missatges

Per poder editar els missatges s'ha fet una modificació sobre la vista (LeanStatisticsView) per detectar quan es polsa sobre la graella de dades i s'obri una finestra d'edició de les dades que hi consten.

Una vegada editades es recuperaren les dades anteriors i el missatge actual modificat.

6.4 Rollback

Poder aquesta part fora la més complicada donat que per poder fer un rollback "real" es necessari fer grans canvis a LeanGen i LeanEditor, donat que en el model de simulació no hi consten totes les dades i n'hi ha moltes que només hi consten a la memòria de LeanGen mentre esta realitzant la simulació com es el cas de les cues amb entitats.

Per tal de no interferir en el funcionament actual del motor de simulació s'han generat unes classes d'accés a la base de dades i s'ha creat una taula amb columnes per guardar tota la informació de debug.

La informació rebuda es tota emmagatzemada i en cas que es vulgui fer rollback es recuperen tots els missatges de debug fins el missatges indicat i s'envia la modificació realitzada, llavors el motor tornarà a executar la simulació i executarà el nou esdeveniment arribant al punt indicat amb les modificacions desitjades.

El problema es que una simulació molt llarga si fos sol•licitada a rollback tindria un cost molt gran tornar a executar-la sencera. Però la idea es tenir estats intermedis de la simulació una vegada LeanGen sigui modificat per tal de guardar tot el model en un instant de simulació concret i així nomes recuperar l'estat del model en un instant i executar uns pocs esdeveniment per tal d'assolir la simulació amb la modificació sol•licitada.