3 Anàlisi de requeriments

El requeriments son aquelles condicions, capacitats o necessitats que el sistema ha de cobrir en el seu disseny. Així doncs en aquest apartat delimitarem quines seran les prestacions del sistema i quines les seves fronteres.

Es poden diferenciar dos tipus de requeriments, els funcionals i els no funcionals.

- **Requeriments funcionals**: Son aquells que descriuen les funcionalitats que ha de donar-nos el sistema.

- **Requeriments no funcionals**: Son aquells que descriuen quines son les condicions generals que ha de complir el sistema en quant a qualitat, rendiment, velocitat, etc.

3.1 Requeriments funcionals

Les funcionalitats que ha d'oferir el sistema son les següents:

- **Comunicació amb els altres components LeanSim.**
- **Representació de la informació de debug.**
- **Control de l'execució del model de simulació.**
- **RollBack del sistema.**

3.1.1 Comunicació amb les altres components LeanSim

El sistema ha de permetre la comunicació amb els altres components LeanSim, tant, és a dir, el disseny ha d'incloure un mòdul de comunicació que segueixi l'especificació adoptada per LeanSim per comunicar els seus mòduls.

Aquest component ha de complir el requeriment que la comunicació sigui TCP/IP amb sockets tal i com defineix el seu disseny. Aquesta implementació es requerida degut a l'alt rendiment que necessita LeanSim per a l'execució dels seus models, donat que es generen un gran nombre de missatges entre els seus components per cada unitat de temps de simulació.

Aquesta funcionalitat té una part important de recerca perquè el missatge amb el que es comuniquen els clients de LeanSim no és estàndard per a tots i cal decidir quin serà el proper format. S'ha de tenir en compte quins clients juguen amb el motor de simulació quines necessitats en quant a informació necessiten e intentar minimitzar la mida del missatge per no saturar la xarxa.

3.1.2 Representació de la informació de debug

La informació de simulació es rep es rebuda al principi de la simulació, quan es rep el Model. I posteriorment amb cada missatge es van rebent els canvis que es van executant sobre el model deguts a l'execució de LeanGen.

Aquesta funcionalitat es molt important degut a que la principal característica d'una eina de debug es la representació de les dades, ja que es el seu principal objectiu donat que quan es volen trobar errors en un model el més important es saber quins valor pren en qualsevol instant de temps.

3.1.3 Control de l'execució del model de simulació

El model de simulació té tres estats d'execució: pausat, aturat, en execució.

- **Estat aturat**: es l'estat inicial del model, quan carreguem el model de simulació estarà en estat aturat, encara no s'ha executat cap acció i no han hagut esdeveniments. Cada vegada que s'aturi el model de simulació tornarem a l'estat inicial.

- **Estat pausat**: Aquest estat es quan han succeït uns quants esdeveniments i es notifica a LeanGen que no ha de continuar executant-ne de nous, no hi ha canvis en el sistema i aquest es l'estat ideal per a verificar l'estat actual del sistema, examinat els atributs, maquines, objectes i fer una anàlisi del model.

- **Estat en execució**: En aquest estat Leangen anirà enviant missatges amb tots els esdeveniments que modifiquin el model de simulació. Es aquí on haurà d'actuar el sistema de rollback emmagatzemant tots els missatges per poder anar enrere en cas que l'usuari vulgui tornar a un estat de temps anterior per visualitzar l'estat de les objectes i els seus valors.

3.1.4 RollBack del sistema

La funcionalitat de Rollback es la que li dona potència a l'eina. Aquesta funcionalitat li permet a un usuari final donat un instant de simulació poder moure's cap enrere en el temps de simulació i tornat a un estat anterior per veure els canvis que hi ha hagut en el model.

Aquesta funcionalitat es la diferencia entre una eina de debug de un llenguatge de programació i una el de un motor de simulació.

Un programador de models de simulació necessita verificar que el seu model te un comportament correcte en diferents estats i per això necessita veure l'evolució dels esdeveniments. Si s'arriba a un punt on es veu clarament que la simulació no es correcte, el programador té la necessitat de tornar enrere en el temps fins veure quin és el detonant d'aquest mal funcionament.

Com a desenvolupador de models de simulació he patit aquesta carència, així quan teníem una execució que podia durar fins a 24h, una vegada finalitzada, la única manera que teníem de fer un seguiment era suposar fins on havia anat

bé la simulació, fer una pausa en aquest instant i a partir d'aquí, anar executant poc a poc la simulació fins a trobar l'error.

3.2 Requeriments no funcionals

El principal requeriment no funcional es que l'eina sigui adaptable als canvis en el model de dades, és a dir, si s'afegeixen nous objectes de simulació,a tributs, valors etc... que aquesta s'adapti fàcilment a aquests canvis.

Un altre requeriment no funcional es la no saturació del sistema. Això s'explica perquè en una simulació podem tenir milions d'esdeveniments i el sistema a desenvolupar ha de ser capaç de tractar-los tots i ser capaç de prevenir el col•lapse d'informació mitjançant tècniques de finestra. No es disposa de recursos il•limitats i la simulació de sistemes reals es propensa a saturar el sistema amb milions de missatges.

Per últim un requeriment no funcional e inherent a tota eina de debug es un control total de tots els esdeveniments del sistema, on no quedi cap acció per part del motor de simulació no reflexada en el sistema de debug.

4 Especificació

Per el procés d'especificació i disseny es farà servir la notació UML (Unified Modeling Language). Aquesta es la metodologia impartida avui en dia a la FIB i es considerada avui en dia com una notació estàndard per a dur a terme el procés de desenvolupament del software.

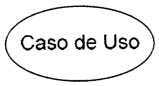
El procés d'especificació es divideix en dos parts:

- **Model de casos d'ús.**
- **Model conceptual.**

4.1 Model de casos d'ús

El diagrama de casos d'ús descriu quins actors hi ha al sistema i quines tasques realitzen.

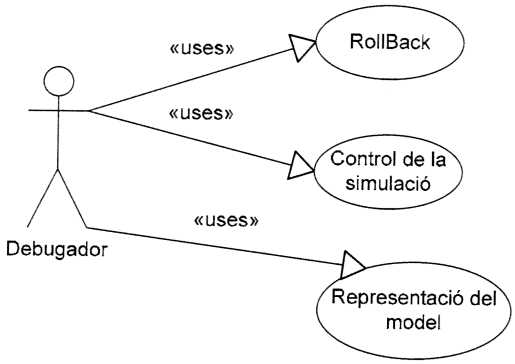
Un cas d'ús descriu una seqüència de esdeveniments que realitza un actor (agent extern) que fa servir el sistema per portar a terme un procés que té algun valor per a ell.



Un actor es una entitat externa al sistema que participa en la historia del cas d'ús. Pot ser una persona, un conjunt de persones, un sistema hardware, un sistema software o un rellotge. L'actor que inicia la seqüència d'esdeveniments rep el nom d'iniciador, mentre que la resta d'actor que intervenen reben el nom de participants.

D'aquesta manera es tindran totes les tipologies d'usuaris que participen i podrem saber quines tasques realitza cada usuari i per quin o quins usuaris es realitzada cada tasca.

4.2 Diagrama de casos d'ús d'alt nivell



ha una única tipologia d'usuari, l'usuari debugador, això es degut a que tots els usuaris han de tenir control total sobre l'eina de debug per a poder trobar els errors en el codi.

Per el moment la opció de separar les funcionalitats en dos actor i que nomes un dels dos tingui les funcionalitats relatives al control de la simulació queda descartat. El debug no serà paral•lel, nomes un usuari farà debug a la vegada. Però està la possibilitat de separar aquesta funcionalitat i que un usuari sigui qui controli la simulació i la resta que facin debug, així quan algú detectés comportament anòmals podria aturar la simulació i la resta d'actors podrien analitzar el model i es repartirien la feina per trobar més ràpidament els errors.

S'ha fet un anàlisi en profunditat de un debug sobre LeanSim i no es pot limitar a l'eina LeanDebug, tots els components han de participar en el debug, es tant important una representació de les dades del model com una visió virtual mitjançant VRML, així com una possible interacció de LeanTraining amb un usuari expert o la recopilació d'estadístics per validar el model i trobar possibles anomalies.

Així doncs aquest model de casos d'ús s'hauria de complementar amb els models de casos d'ús del altres components que integren LeanSim.

4.3 Model conceptual

El model conceptual es la representació dels conceptes significatius en el domini del problema.

Es tindran els següents components:

- **Classes d'objectes.**
- **Atributs de classe.**
- **Associacions entre classes.**

4.4 Clients LeanSim

El primer model que s'exposarà es el model conceptual dels clients de LeanSim.

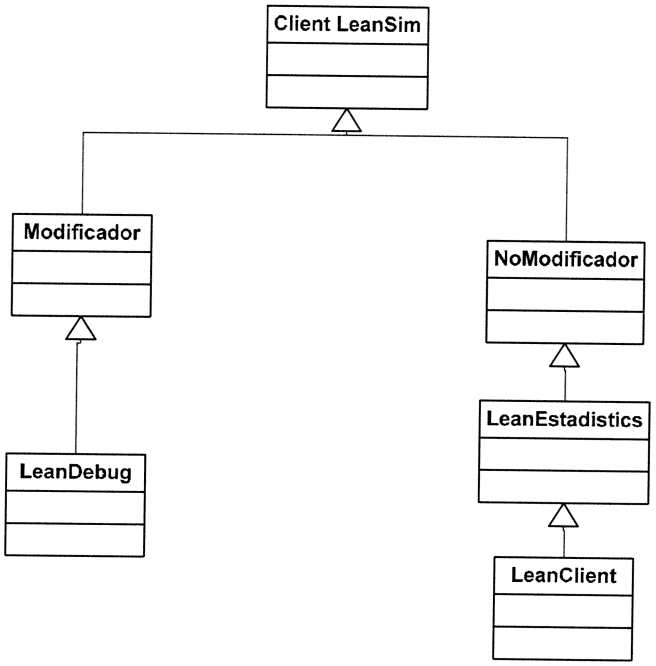
Es important aquest model ja que LeanSim va néixer com un motor de simulació i un client de representació, però en els últims anys han sorgit nous clients de LeanSim, LeanTraining, LeanStadistics i ara LeanDebug.

Tant LeanTraining com LeanStadistics van començar a ser dissenyats simultàniament i potser es per aquest motiu que no van compartir un model conceptual.

LeanClient només tenia un objectiu, la representació tridimensional de l'execució del model.

Donat que ja son quatre el clients de LeanSim s'ha decidit fer un model conceptual que pretén organitzar els clients i les seves funcionalitats per tal de no desenvolupar més d'una vegada les mateixes classes i no malaguanyar temps en el seu manteniment.

El model conceptual es el següent:



S'ha decidit dividir els clients segons si modifiquen el model o no el modifiquen, LeanDebug al tenir les funcionalitats de RollBack pot modificar el model de dades i LeanTraining a l'interactuar amb l'execució també estén d'aquesta classe.

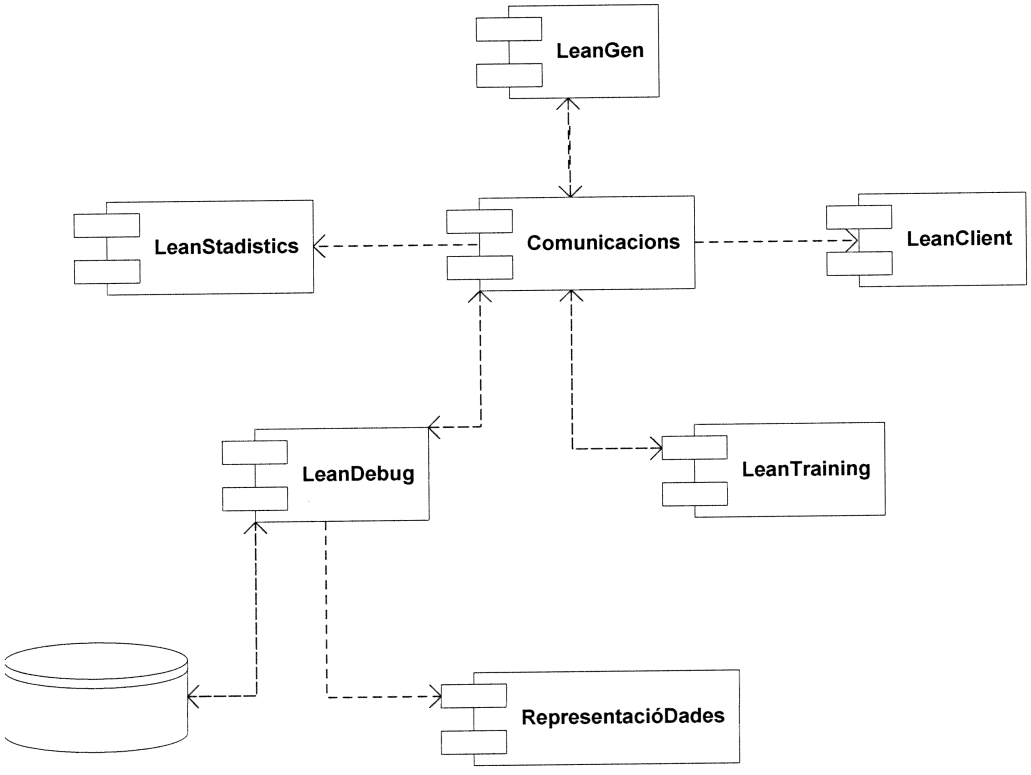
LeanClient només rep missatges de representació i realitza una representació en VRML i LeanStadistics rep missatges d'estadístics i realitza càlculs amb aquests però no interactua amb el motor, per tant entren els dos en la categoria de Clients de LeanSim No Modificadors.

S'ha parlat amb els desenvolupadors de LeanSim i probablement en un futur s'hauria de desenvolupar un únic client de LeanSim que permetés una instal•lació segons les funcionalitats que farà servir l'usuari.

D'aquesta manera es podria aprofitar el mateix codi, tots els clients tindrien el mateix model de dades i comunicacions i s'evitaria multiplicar el cost de manteniment del producte.

4.5 Interacció LeanDebug

El següent model conceptual detalla La interacció de LeanDebug amb el motor de simulació i descriu els elements amb els que treballa.



Al gràfic anterior es pot veure el diagrama de components de LeanSim i com interaccionen, en el gràfic s'ha volgut destacar la interacció dels components nous i de la modificació del component de comunicacions.

A l'esquema es pot veure com tots els client i el servidor de simulacions es comuniquen tots amb el mateix mòdul de comunicacions reduint els problemes originats per disposar d'un protocol de comunicacions diferent per cada client.

Tant LeanStadistics com LeanClient rebrien informació de LeanGen però no enviarien, a la realitat si que envien informació però només de connexió i el que es vol plasmar al diagrama serien els missatges que provoquen canvis a LeanGen.

LeanDebug i LeanTraining tenen una comunicació bidireccional donat que els dos interaccionen amb el model i realitzen canvis sobre aquest.

LeanDebug disposa de dos mòduls addiciona'ls, el primer es el gestió de base de dades que s'encarrega de emmagatzemar tots els missatges que rep del motor de simulació per un posterior RollBack, d'aquest mòdul es parla en detall més endavant.

Per últim cal destacar el mòdul de representació de les dades que mostrarà les dades amb un component visual en format d'arbre i permetrà la cerca d'elements per tal de poder inspeccionar tots els valors dels objectes del model i verificar a cada instant de temps quins valor va prenent.

4.6 Especificacions d'avaluació de les tecnologies

A continuació es presenten alguns dels diagrames de classe que s'han fet servir per testejar l'aplicació. Cal dir que encara que no s'hagi implementat completament l'eina hi ha hagut una gran tasca de programació per fer proves de les tecnologies que fa servir LeanSim, a més d'aquests mòdul també n'hi ha de molts altres que s'han rebutjat per canvis en el disseny o perquè s'ha vist que la solució no era prou eficient o no complia les especificacions.

Han sigut moltes les tecnologies a estudiar i moltes les proves realitzades doncs hom pensa que per molt que es llegeixi sobre una tecnologia fins que no es veu en pràctica no s'acaba de entendre completament. Així s'ha hagut d'experimentar amb : XPath, SQL Server 2005, XQuery, DTD, XML, C#, Els clients de LeanSim i el motor LeanGen.

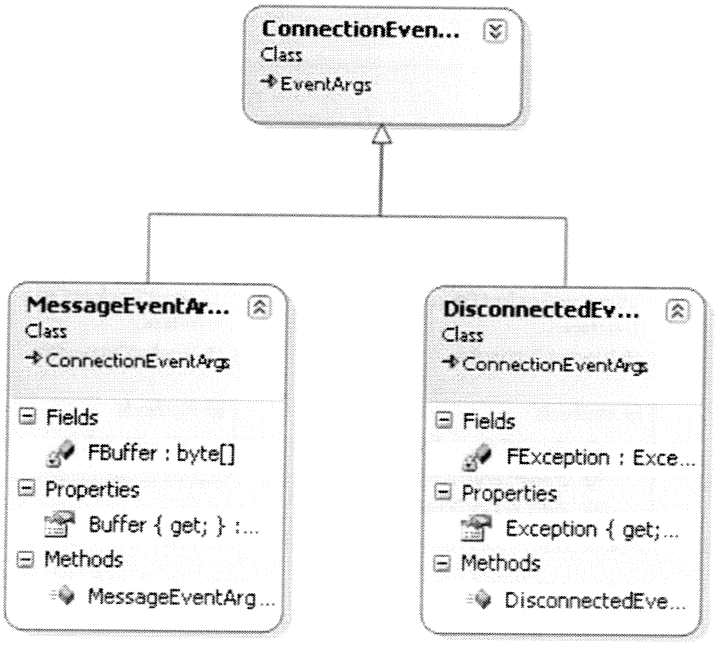
4.7 Model de comunicacions

A continuació es descriurà el model de comunicacions del sistema.

4.7.1 Esdeveniments de connexió de missatges

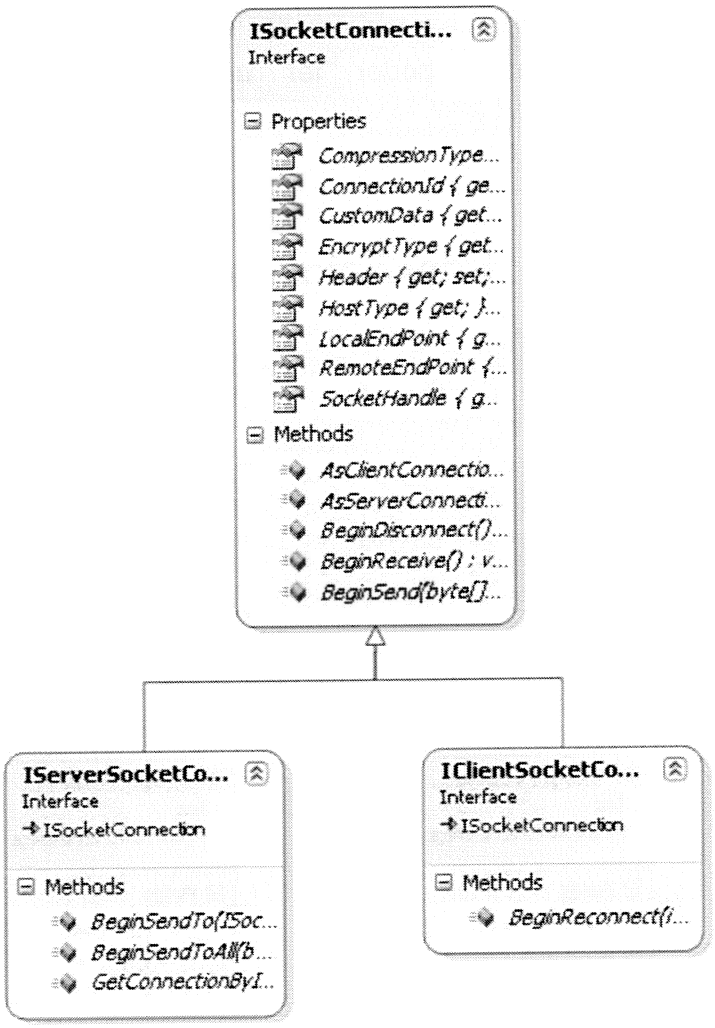
Es va dissenyar una classe per captar un nou missatge d'arribada al sistema com un esdeveniment i poder fer una programació orientada als esdeveniments.

4.7.2 Interfícies de sockets



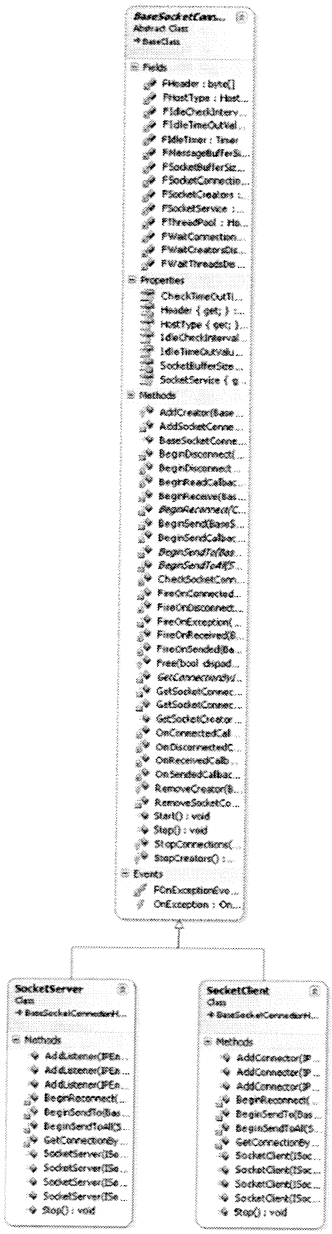
Es van definir les interfícies de sockets a implementar amb els mètodes usuals per enviar i rebre informació.

4.7.3 Dades de connexió



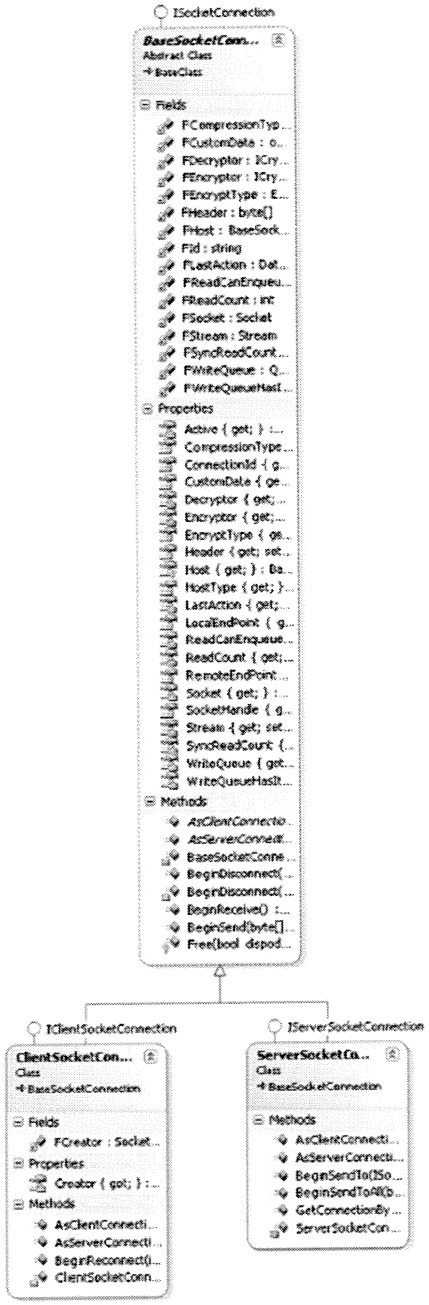
Aquí tenim les dades de connexió amb les característiques de xifrat i compressió de les dades tant per un connector de socket com per el Listener.

4.7.4 Client i servidor



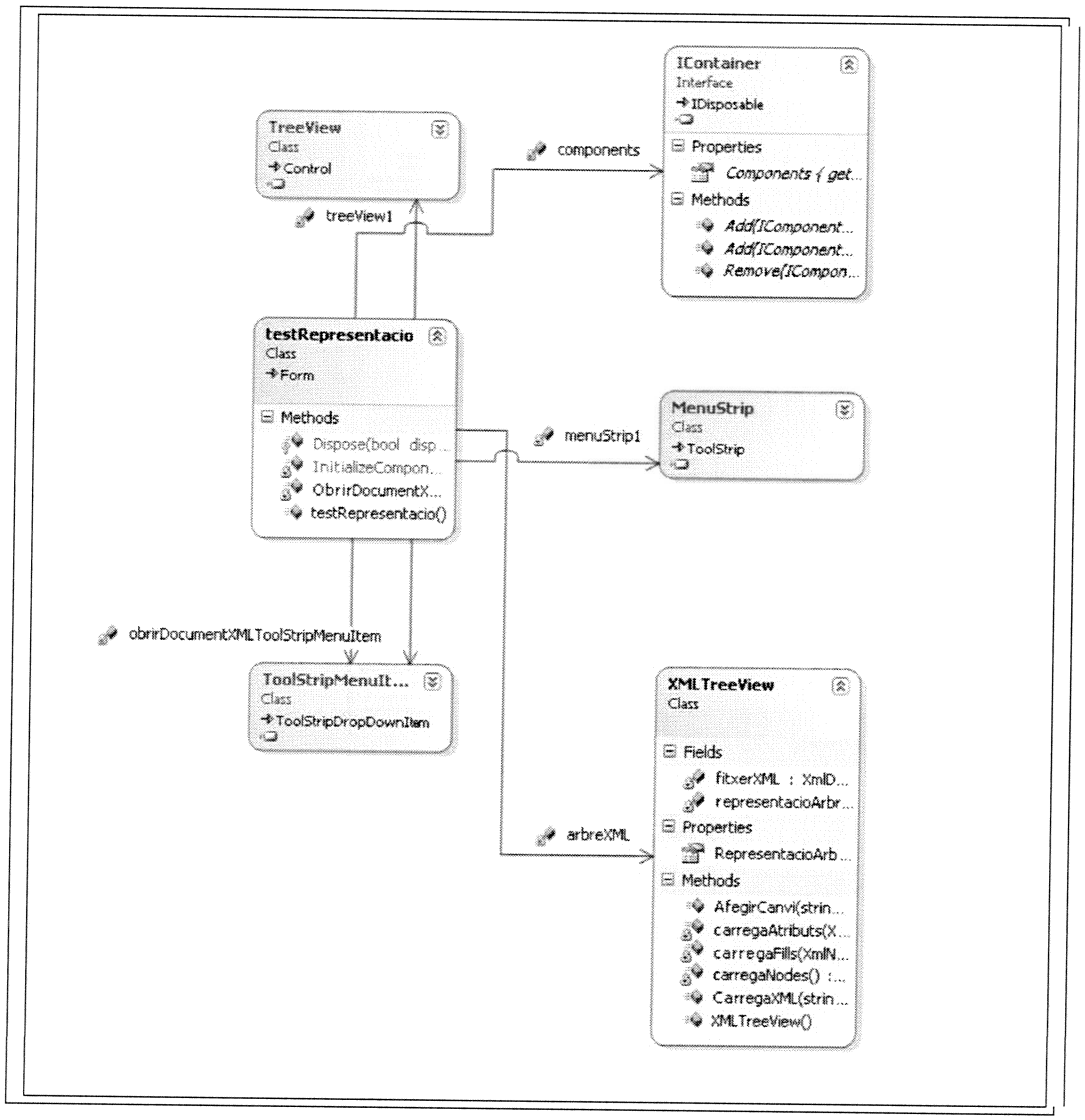
Implementació bàsica d'un client i servidor de missatges per testejar l'enviament d'informació i el posterior processament a la base de dades.

4.7.5 Implementació sockets



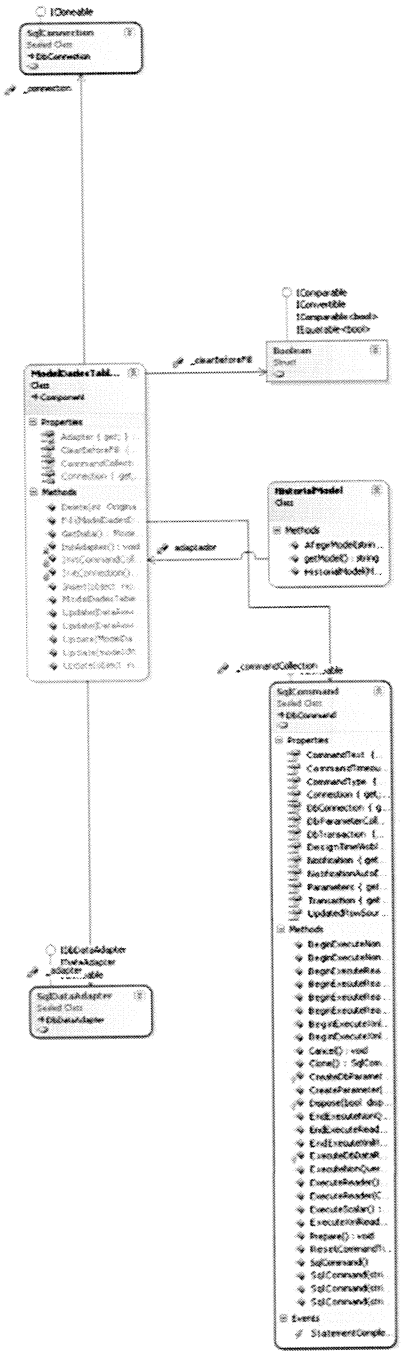
Implementació de la interfície anterior de sockets.

4.7.6 Model de representació de dades



Amb aquest disseny s'ha provat una representació de la informació XML en un objecte del tipus TreeView de .NET, també es varem implementar proves en quant anar canviant els valors dels nodes i que aquest vagi mostrant els canvis i la seva navegabilitat.

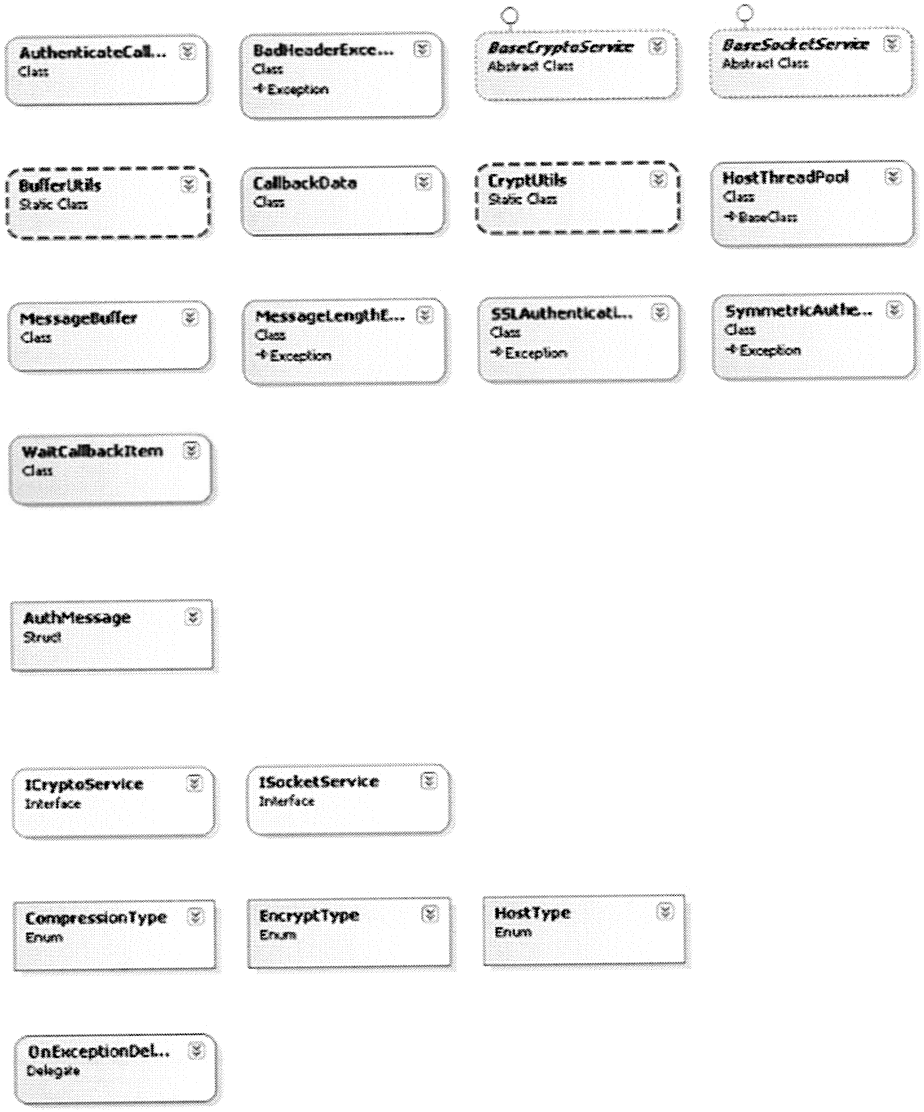
4.8 Accés a la Base de dades



Es van dissenyar les funcionalitats per accedir a la base de dades i fer test de rendiment i provar tota la funcionalitat del nou tipus de dades XML que aporta SQL Server, es varen fer moltes proves amb aquest tipus de dades i també

amb les noves sentencies XQuery que aporta SQL Server aportant grans millores de rendiment per al tractament de dades XML

4.9 Resta de classes



Aquestes classes han sigut testejades per provar funcionalitats de autenticació, xifrat de la informació que viatgi per la xarxa i compressió de les dades.

Son una sèrie d'utilitats extra però no per això menys importants ja que la solució descrita a l'arquitectura en parla d'elles i s'ha volgut provar la seva factibilitat.