Índex

1 INTRODUCCIÓ 1

1.1 DESCRIPCIÓ DEL PROJECTE 1
1.2 MOTIVACIÓ 4
1.3 OBJECTIUS 5
1.4 ABAST 7

2 CONCEPTES PREVIS 9

2.1 RESUM HISTORIC DE LES EINES DE DEBUG 9
2.1 SIMULACIÓ 12
2.1.1 SISTEMES 13
2.1.2 MODELS 14
2.1.3 AVANTATGES I DESAVANTGES DE LA SIMULACIÓ 15
2.2 LEANSIM 17
2.2.1 FONAMENTS DE LEANSIM 17
2.2.1.1 Event Scheduling 18
2.2.2 ARQUITECTURA DE LEANSIM 20
2.2.3 LEANEDITOR 20
2.2.4 LEANGEN 21
2.2.5 LEANCLIENT 22
2.3 LÒGICA DEL MODEL 23
2.3.1 ENTITATS 23
2.3.2 MÀQUINES 24
2.3.3 OPERACIONS 25
2.3.4 PROCÉS I ACCIONS 27

3 ANÀLISI DE REQUERIMENTS 29

3.1 REQUERIMENTS FUNCIONALS 30
3.1.1 COMUNICACIÓ AMB LES ALTRES COMPONENTS LEANSIM 30
3.1.2 REPRESENTACIÓ DE LA INFORMACIÓ DE DEBUG 31
3.1.3 CONTROL DE L'EXECUCIÓ DEL MODEL DE SIMULACIÓ 31
3.1.4 ROLLBACK DEL SISTEMA 32

3.2 REQUERIMENTS NO FUNCIONALS 34

4 ESPECIFICACIÓ 35

4.1 MODEL DE CASOS D'ÚS 36
4.2 DIAGRAMA DE CASOS D'ÚS D'ALT NIVELL 37
4.3 MODEL CONCEPTUAL 39
4.4 CLIENTS LEANSIM 40
4.5 INTERACCIÓ LEANDEBUG 42
4.6 ESPECIFICACIONS D'AVALUACIÓ DE LES TECNOLOGIES 44
4.7 MODEL DE COMUNICACIONS 45
4.7.1 ESDEVENIMENTS DE CONNEXIÓ DE MISSATGES 45
4.7.2 INTERFÍCIES DE SOCKETS 45
4.7.3 DADES DE CONNEXIÓ 46
4.7.4 CLIENT I SERVIDOR 47
4.7.5 IMPLEMENTACIÓ SOCKETS 48
4.7.6 MODEL DE REPRESENTACIÓ DE DADES 49
4.8 ACCÉS A LA BASE DE DADES 50
4.9 RESTA DE CLASSES 52

5 DISSENY 53

5.1 ARQUITECTURA LEANDEBUG 54
5.2 ARQUITECTURA REPRESENTACIÓ DE LES DADES 57
5.3 ARQUITECTURA ROLLBACK 61
5.3.1 DIAGRAMA ROLLBACK 64
5.3.1.1 ObjecteComprimit 65
5.3.1.2 XML 65
5.3.1.3 TipusCompressio 65
5.3.1.4 ControladorAccesBD 66
5.3.1.4.1 Atributs 66
5.3.1.4.2 Mètodes 67
5.3.1.5 ModelAccesBD 69
5.3.1.5.1 Atributs 69
5.3.1.5.2 Mètodes 70
5.4 ARQUITECTURA MISSATGERIA 72
5.4.1 CONNEXIÓ 72

5.4.2 CONTROL DE LA SIMULACIÓ 73
5.4.3 MODIFICACIONS SOBRE EL MODEL 74
5.4.4 FORMAT DELS MISSATGES 75
5.4.5 EXEMPLE DE MISSATGERIA 77
5.4.6 CANVIS EN EL MODEL DE DADES 79

6 IMPLEMENTACIÓ: MODIFICACIÓ A LEANSTADISTICS PER DEPURAR 91

6.1 INSPECCIÓ DELS MISSATGES DEL SISTEMA 92
6.2 CONTROL DE LA SIMULACIÓ 93
6.3 EDICIÓ DELS MISSATGES 94
6.4 ROLLBACK 95

7 TECNOLOGIA UTILITZADA 97

7.1 C# 98
7.2 XML 100
7.3 DTD Y XML SCHEMAS 102
7.4 XPATH 105
7.5 DTD 109
7.6 XQUERY 113
7.6.1 ANTECEDENTS 113
7.6.2 XPATH 114
7.6.2.1 XQuery amb el tipus de dades XML 116

8 PROVES D'AVALUACIÓ DE LES TECNOLOGIES 119

8.1 CAIXA NEGRA 121
8.2 CAIXA BLANCA 122
8.3 PROVES REALITZADES 123

9 PLANIFICACIÓ DEL PROJECTE 125

9.1 PLANIFICACIÓ 126
9.1.1 FASES DEL PROJECTE 126
9.1.2 CALENDARI DE TREBALL 127
9.1.3 ESTUDI ECONÒMIC 130

10 CONCLUSIONS 133

10.1 CONSECUCIÓ D'OBJECTIUS 134
10.2 LÍNIES DE FUTUR 136

11 AGRAÏMENTS 137

12 BIBLIOGRAFÍA 139

12.1 SIMULACIÓN 139
12.2 LEANSIM 139
12.3 ENGINYERIA DEL SOFTWARE 139
12.4 WEBGRAFÍA 140

1 Introducció

1.1 Descripció del projecte

L'objectiu del present projecte es l'anàlisi i disseny dels components de programari principals d'una aplicació de depuració d'errors per a l'eina de simulació anomenada LeanSim(r), així com desenvolupar una primera implementació de la mateixa.

Entenent una aplicació de depuració d'errors (en la seva terminologia anglesa debug) com una eina de recolzament al desenvolupador per tal de trobar i reparar els errors (bugs) que pot haver-hi en el seu codi.

Durant tota la memòria del projecte es fan referències al termes debug, debugar, i derivats. S'ha decidit fer servir aquest anglicisme degut a que aquest terme està molt extés i acceptat a l'àmbit de la informàtica.

Des de sempre els programadors han fet servir les eines que els donava el seu programari de programació i quan aquestes no arribaven a les expectatives desitjades aquests s'havien de fer amb el seu enginy per trobar els errors dins una infinitat de línees de codi. Feien (i fan) servir missatges a la consola del sistema amb valors que es recullen en punts estratègics del programa, pop-ups que alerten quan alguna acció és sospitosa, etc...

De totes maneres l'eina més utilitzada a l'hora de debugar una aplicació és la famosa funció emprada per pràcticament tots els programadors: "println". És tan simple com útil, la seva funcionalitat consisteix en treure per pantalla un missatge descrit pel programador amb la possibilitat d'afegir el valor d'alguna variable.

La gran avantatge és que es implementada per la majoria de llenguatges de programació i et permet saber quin es el flux del programa i quins valors prenen les

variables. La seva desavantatge es que has d'introduir una infinitat de línies de codi amb aquesta funció i re-compilar cada vegada.

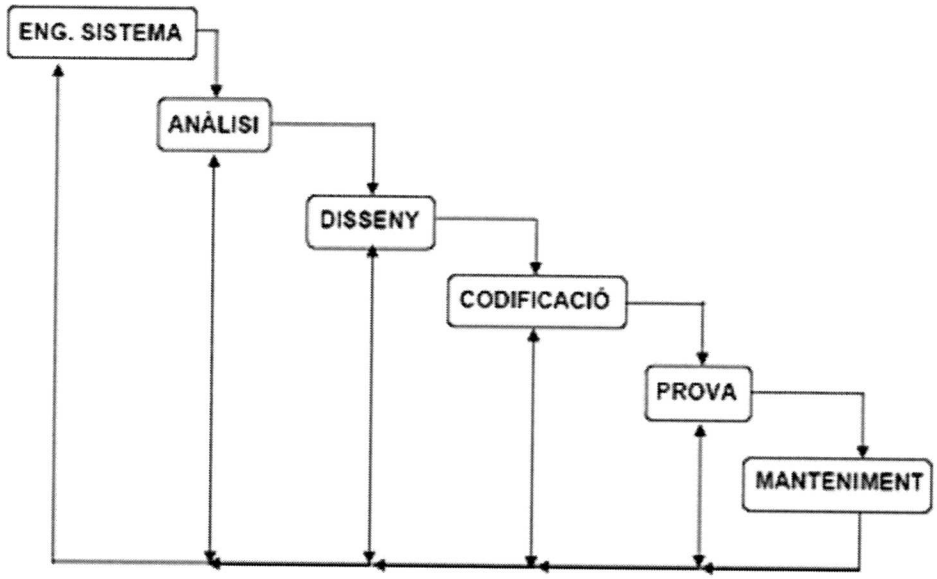
Les eines actuals que podem trobar al mercat estan fortament lligades a les aplicacions destinades al desenvolupament de programari. Fora de l'àmbit dels programadors pràcticament no es troba cap aplicació que en faci ús donat que la majoria de les aplicacions destinades a un usuari final tenen fortes restriccions. Les ordres que poden executar estan molt limitades ja que es fan mitjançant icones, botons, opcions de menú, és a dir, seleccions dins un àmbit restringit d'opcions.

El ràpid avanç que està vivint el món tecnològic fa que fer servir les tradicionals eines de debug dels llenguatges de programació on només es tracten variables, posicions de memòria, tipologies i valors queda completament desfasat.

La eina que es vol dissenyar es un debug que permeti la depuració de models de simulació. A un enginyer que esta desenvolupant un model de simulació i vol saber si el disseny es correspon amb les especificacions no se li pot parlar de posicions de memòria, variables o de si un "cast" de int a long és o no possible. El diàleg ha de ser coherent amb la llenguatge emprat en la descripció del model.

En aquest document també s'entrarà en detall de quins components formen LeanSim i com es complementa LeanDebug amb els altres components així com afecta a la resta de mòduls la introducció d'aquest nou component en l'entorn LeanSim.

Aquest mateix problema no el tenen només els programaris de simulació, es poden trobar una infinitat de productes que pateixen del mateix problema, per exemple, en el desenvolupament de videojocs, dissenyadors de components mecànics, etc...



Com es pot veure a la figura anterior l'etapa de prova del programari pren un paper fonamental en el procés de l'enginyeria del software.

Sent aquesta la última etapa abans d'entregar a un client final un programari i prèviament a entrar en un etapa de manteniment. Si no es realitza correctament es podria tenir un programari que no compleix amb les especificacions i repercutir en pèrdues en hores de programació, econòmiques o fins i tot si es tractés d'una simulació destinada a l'entrenament de persones aquestes rebrien un entrenament incorrecte i no sabrien reaccionar correctament davant les situacions per a les que s'han entrenat.

1.2 Motivació

La principal motivació per la realització del projecte es aportar el meu granet de sorra al desenvolupament de LeanSim i continuar treballant amb un projecte que coneixia bé conjuntament amb els meus companys de simulació del laboratori de càlcul.

La idea del projecte em va sorgir mentre treballava al laboratori de càlcul desenvolupant un model de simulació de la tercera pista d'aterratge de l'aeroport de Barcelona, el programari emprat era Witness i no disposava de cap eina de debug i com a programador vaig patir aquesta carència i ens havien d'enginyar metodologies i "trucs" per trobar els errors de disseny o desenvolupament.

Van ser molt dies i moltes nits buscant perquè per una cua no passava cap entitat o perquè la capacitat d'un pàrquing sempre es superava o el comportament de quina entitat estava mal implementat o dissenyat.

El desenvolupament d'un model de simulació que no s'ha pogut depurar amb certes garanties de qualitat es propens a errors i resultats equívocs, a més d'incrementar considerablement el cost de desenvolupament.

1.3 Objectius

L'objectiu del projecte és l'anàlisi i disseny d'uns mòduls bàsics d'una eina de debug per al projecte LeanSim.

Es realitzarà tot l'anàlisi i disseny de l'eina, en quant a la implementació, s'implementaran les principals funcionalitats per tal que un programador de LeanSim pugui veure algunes de les opcions de les que podrà gaudir amb l'eina de debug.

Els diferents subobjectius per a la realització del projecte son els següent:

- **Estat de l'art de les eines de debug.**

Revisió breu de quines eines es disposen actualment, quines funcionalitats aporten i cap on apunta el futur de les eines de debug.

- **Anàlisi dels requeriment de l'eina.**

Quines son els requeriments que hauran de complir els components dissenyat i els requeriments no funcionals imposats per LeanSim.

- **Anàlisi d'un mòdul de representació del model de simulació.**

Anàlisi d'un mòdul que permeti visualitzar la informació extreta del model.

- **Anàlisi d'un mòdul de comunicació entre LeanSim i LeanDebug.**

Anàlisi d'un mòdul que comuniqui tant el motor de simulació com els altres clients per tal de enviar i rebre missatges i establir els protocols de comunicació.

- **Anàlisi d'un mòdul d'emmagatzemament de l'historia'l de canvis.**

Anàlisi d'un mòdul que gestioni tota la informació de l'execució, i mantingui un historial de canvis per poder fer rollback, i poder desplaçar-se en el temps de la simulació.

- **Implementació de les principals funcionalitats de l'aplicació.**

S'implementaran els principals mòduls de l'aplicació per donar suport de debug a LeanSim.

1.4 Abast

Evidentment, dintre del marc del present projecte, implementar una eina de debug fins a l'últim detall i comercialment competitiva amb el mercat d'avui en dia és totalment impossible donats els recursos disponibles.

Actualment les eines de debug són desenvolupades per desenes o centenars de programadors i estan fortament lligades a l'eina on aporten aquesta funcionalitat.

LeanDebug pretén aportar les funcionalitats més útils que necessitaria una eina de debug per LeanSim però procurant no tancar-se a una implementació concreta de LeanSim de manera que encara que el programari evolucioni es pugui adaptar fàcilment als canvis.

Per això l'objectiu d'aquest projecte final de carrera consisteix en enllestir unes funcionalitats bàsiques que permetin un seguiment d'una simulació bàsica amb la possibilitat de tirar enrere en el temps i visualitzar l'estat del model de simulació.

Els funcionalitats bàsiques haurien de ser:

- **RollBack**

Tirar enrere en el temps de simulació per tornar a un estat anterior i fer un estudi en profunditat d'alguna casuística.

Representació visual

Mostrar l'estat actual del model de simulació per veure quina informació es pot extreure del model i analitzant les dades prendre les decisions oportunes.

- **Comunicació**

Un mòdul per a que el component pugui enviar i rebre missatges del motor de simulació.

2 Conceptes previs

2.1 Resum històric de les eines de debug

L'origen del terme debug prové de Admiral Grace Hopper cap al 1940 quan estava treballant amb el seu computador i un insecte va entrar en dins la circuiteria i va produir un curtcircuit provocant un canvi en la lògica del programa que estava fent servir.

Però fins l'any 1976 no apareix el terme "debugging" en el llibre de Glenford J Myers "Software Reliability: Principles and Practices"

Durant els últims 20 anys han sorgit un munt d'eines de debug, la majoria molt específiques per una tasca concreta o dissenyades per estrictament veure el contingut de la memòria i modificar-la, algun exemples son:

DEBUG: Comanda de MS-DOS/Windows per veure interactivament la memòria, fer-hi canvis i executar selectivament processos.

Oxygen: Eina per debugar documents XSL i XSLT i XQuery. Aporta un anàlisi sintàctic i cerca nodes malformats.

DDT: Allinea Distributed Debugging tool esta pensat per fer debug de eines amb gran escalabilitat i amb multithreading d'execució en paral·lel.

CodeVlew: El primer debug de visual C++ 1.0 que aportava l'integració de l'eina de debug amb l'eina de programació.

Dbx: Va ser desenvolupat a Berkley i està inclòs al paquet de Sun Studio, va introduir les parades a codi i la manipulació de variables i avaluació de sentencies.

Etnus TotalView: Aporta la funcionalitat de sincronitzar threads en un mateix breakpoint.

GDB: El debug de GNU, permet fer debug remotament però manca d'un entorn gràfic.

MacsBug: Sistema de debug de MAC, es a baix nivell i pot ser cridat directament mitjançant la tecla d'interrupció.

Purify: Es una eina de debug especialitzada en veure errors d'accés a memòria especialment en programes escrits en C/C++.

SoftIce: Eina de windows per debugar el kernel dissenyat per la construcció de drivers.

WinDbg: Dissenyat per Microsoft per analitzar els errors de la famosa "Blue Screen of Death" de windows98 e introdueix el concepte de postmortem debug per analitzar el volcat de pila.

Visual Studio Debugger: probablement la eina més potent de debug, disposa de tot un ventall d'avantatges i aporta pràcticament totes les millores que s'han anat introduint al llarg dels anys a més incorpora l'opció de "edit & continue" que permet modificar el codi font mentre s'està en mode debug i continuar debugant sense haver de parar, re-compilar i tornar a engegar, sempre amb algunes limitacions.

En quant a les eines de debug de programari de simulació cal destacar ARENA i Witness. Arena disposa d'un objecte TRACE que et reporta una traça amb els resultats de la simulació, conté molta informació però una traça d'una simulació

molt llarga pot fer perdre moltes hores en la seva revisió es molt difícil trobar errors dins una traça.

Witness disposa també d'una traça però a més de una funcionalitat WATCH per poder examinar els valor de les objectes del sistema una vegada s'ha pausat la simulació i veure quin es l'estat de la simulació.

La traça de Witness té algunes mancances i no emmagatzema tota la informació, això implica que quan es vol reconstruir una simulació a partir d'una traça no es tinguin tots els resultats esperats.

No s'ha pogut fer un estudi molt més extens sobre les eines de debug que tenen Witness i ARENA degut a que les dos son eines de pagament amb un cost molt elevat, tot el que he pogut esbrinar es remet a la meva experiència treballant amb Witness i ARENA..

2.1 Simulació

La simulació es defineix com una tècnica en la qual s'utilitzen computadors per reproduir, o simular, els processos que tenen lloc en el món real.

Es pot complementar aquesta definició amb la que dona Robert E. Shannon en 1975, on parlar de la simulació com "el procés de dissenyar un model d'un sistema real i portar a terme experiències amb ell, amb la finalitat d'entendre el comportament del sistema o avaluar diferents estratègies per al funcionament del sistema".

Ambdós definicions parteixen de la necessitat d'estudiar el comportament d'un conjunt de processos del món real, nomenat sistema. Per això es necessari crear un conjunt d'abstraccions, amb forma de relacions matemàtiques o lògiques, que donen lloc a un model conceptual.

Serà sobre aquest model sobre el que es realitzi l'anàlisi, però la simulació no és la única tècnica disponible. Si el model és senzill, de forma que les relacions que la composen son conegudes i poden ser definides de forma relativament senzilla, es poden utilitzar mètodes analítics (teoria de cues, teoria d'inventaris, etc.) per obtenir respostes exactes.

Els processos del món acostumen a ser massa complexes per ésser estudiats mitjançant tècniques analítiques, sent llavors la simulació l'eina adequada per realitzar l'estudi. En aquests casos s'implementa el model sobre programari específic de simulació (Witness, Arena, Vensim, etc.) amb el què s'executarà i s'obtindran els resultats. Tot i així s'ha de tenir en compte que a diferència dels mètodes analítics, on la solució era precisa, en el cas de la simulació el resultat és una estimació dels valors reals, degut als factors aleatoris que intervenen en el model. Per aquest motiu es important analitzar i validar el model i els resultats obtinguts, mitjançant l'ús de tècniques estadístiques (repeticions independents, tècniques de reducció de la variància, etc.

2.1.1 SISTEMES

Fins el moments s'ha parlat de la simulació com a eina per estudiar el comportament d'un sistema. Però què és un sistema?

Un sistema es defineix com un conjunt d'entitats, u objectes, que actuen i interactuen entre si per poder aconseguir un propòsit ben definit . La definició del sistema és un procés normalment complex, i que en general depèn del estudi concret que es vulgui fer, donat que els components del sistema observats poden variar d'un estudi a un altre. Aquest components son considerats les variables del sistema.

Una vegada definit el sistema es pot representar el seu estat. Per estat d'un sistema s'entén el conjunt de variables necessàries per descriure el sistema en un moment donat.

Amb aquesta primera abstracció del món real els sistemes es poden dividir en dos grups:

- **Sistemes continus**: Les variables d'estat canvien contínuament respecte al temps. La velocitat d'un vehicle o la quantitat d'aigua que passa per un tub son exemples de variables contínues.

- **Sistemes discrets**: Les variables d'estat canvien en instants de temps separats per períodes de temps en els que no passa res. Un exemple de variable es podria trobar en el nombre de persones d'una sala d'espera.

Tot i que hi ha casos en què la diferència entre discret i continu és molt clara en general no hi ha sistemes discrets purs ni sistemes continus purs. Sistemes que a priori podrien ser classificats com a discrets, com podria ser la població

d'animals en un parc natural, poden ser també estudiats com sistemes continus, utilitzant funcions matemàtiques per la representació dels seus estats.

De la mateixa manera, sistemes continus poden ser tractats com discrets. Donada la complexitat d'estudi dels sistemes continus, fonamentats en equacions diferencials, difícil de plantejar, validar i resoldre- s'acostuma a realitzar una discretització i tractar aquests sistemes com si es tractés de sistemes discrets.

2.1.2 MODELS

La necessitat de modelar neix de la dificultat i poca flexibilitat que ofereix un sistema real per realitzar experiments sobre ell i provar el efectes de certs canvis.

En canvi el procés de construcció d'un model és dificultós i complex, però una vegada construït permet la realització d'experiments amb un cost més baix que el què suposaria realitzar la prova sobre el sistema real.

El models es poden classificar segons tres punts de vista. Una primera classificació seria en funció de l'efecte del temps sobre el model:

- **Models estàtics**: Representen un sistema en un moment determinat, o sistemes en el quals el temps no afecta a l'estat.

- **Models dinàmics**: Evolucionen al llarga de temps.

Una segona classificació es troba a partir de l'efecte de l'aleatorietat sobre el model:

- **Models deterministes**: Aquests models no contenen cap component aleatori. Donat una entrada al sistema, la sortida està perfectament definida.

- **Models estocàstics**: En aquests models intervenen variables aleatòries jugant un paper important en les variacions del estat del sistema.

Per últim els models es poden classificar en funció del tipus de sistema que representen:

- **Models continus**
- **Models discrets**

2.1.3 AVANTATGES I DESAVANTGES DE LA SIMULACIÓ

La simulació és una tècnica utilitzada des de fa dècades, però ha estat en els últims anys quan ha començat a ser una eina per l'ajuda en la presa de decisions a les grans companyies. Aquest augment en l'ús de la simulació ha donat peu no sols per l'augment de la capacitat de computació i de visualització, o per gran disponibilitat en el mercat d'entorns de simulació, si no també gràcies als avantatges que ofereix la simulació en d'anàlisis, que es detallen a continuació:

El procés de desenvolupament d'un model de simulació proporciona un millor coneixement del sistema real. Aquest coneixement potser utilitzat per proposar millores en el seu rendiment, més enllà del propi estudi de simulació.

L'observació dels resultats obtinguts mitjançant la simulació ajuda a localitzar les variables més influents dintre del sistema, així com la sensibilitat del sistema a canvis en les entrades o paràmetres del mateix.

La simulació permet analitzar el impacte de certs canvis dintre de la configuració del sistema real actiu, sense necessitat d'interrompre la seva activitat.

Permet respondre a preguntes del estil " Què passaria si realitzem aquest canvi...", amb lo qual la converteix en una eina molt útil dins la pressa de decisions.

També vinculat a la presa de decisions, el temps d'execució d'un model de simulació és inferior al temps que triga en produir-se el mateix procés en el sistema real (llevat excepcions on el què es desitja és precisament l'efecte contrari, analitzar un procés més ampli de sistemes que el què potser estudiat mitjançant tècniques no estocàstiques).

Pot ser utilitzada com eina pedagògica, com ajuda a la comprensió i/o utilització de sistemes reals, en especial en combinació amb tècniques de representació virtuals.

Tot i així, es deuen tenir en compte possibles problemes derivats d'un ús incorrecte de la simulació:

Si un model no està correctament verificat i validat, no és possible traslladar els resultats obtinguts del seu anàlisi al sistema real, doncs el model podria estar descrivint un sistema diferent al desitjat.

Es precisament aquí on entra LeanDebug a recolzar aquesta difícil etapa de validació i verificació, es aquest punt el més crític doncs una mala validació d'un bon model de simulació pot portar resultats no desitjats. Així es podria tenir un bon modelat d'un sistema real amb molta feina i hores dedicades en el seu desenvolupament però que per problemes de verificació i validació donar uns resultats erronis i semblar-li a un usuari final que no s'ha fet bé la feina.

2.2 LeanSim

Abans de descriure que és LeanDebug és important un primer contacte amb el paquet LeanSim, entorn de simulació integrat que es complementa amb el component software desenvolupat, LeanDebug.

L'entorn de simulació LeanSim ha estat desenvolupat per l'equip de simulació del Laboratori de Càlcul de la facultat d'Informàtica de Barcelona (LCFIB) y el Departament d'Estadística i Investigació Operativa (EIO) de la Universitat Politécnica de Catalunya (UPC), i dirigit pel doctor Josep Casanovas García.

Per poder mostra aquesta eina seguirem els punts següents:

- **Fonaments de LeanSim**
- **Arquitectura de LeanSim**
- **Lògica del model**

2.2.1 Fonaments de LeanSim

LeanSim neix amb la necessitat de crear models de simulació a mida en un entorn que permet als usuaris sense coneixements específics poder interpretar els resultats obtinguts i facilitar la reutilització de components en projectes de transferència tecnològica. Per poder aconseguir això l'entorn es fonamenta la simulació ajustada i la representació virtual.

Basat en la simulació ajustada, LeanSim que està creat com una eina de simulació genèrica , permet partint de modificacions mínimes en el simulador definir nous objectes de simulació amb els que aconseguir el nivell de detall necessària pel projecte. El fet de ser una eina pròpia, desenvolupada en la

seva totalitat per l'equip de simulació , permet que aquestes modificacions siguin senzilles, donat que existeix un profund coneixement de l'eina.

La facilitat d'ús i interpretació ve donada per la representació virtual del model i els seus processos, amb la qual s'eviten les simplificacions i abstraccions a la representació que, en ocasions, podien induir a interpretacions errònies per part de l'usuari final.

Referent al motor de simulació LeanSim està fonamentat en dos dels principals paradigmes de simulació, ja que per una part s'utilitza una sèrie de processos per definir el comportament de les entitats dintre del sistema simulat – "Oriented Process"- i per un altre part està orientat a esdeveniments de simulació, seguint el paradigma "Event-Scheduling", que a continuació s'explica.

2.2.1.1 Event Scheduling

Els esdeveniments son accions instantànies que poden canviar l'estat del model. Cadascú d'aquest esdeveniments tenen una marca temporal que indica el temps en què ha d'executar-se pel que és important mantenir una llista on es contenen els esdeveniments pendents de forma endreçada. Aquesta llista rep el nom de "Llista d'esdeveniments futurs" (LEF).

A partir d'aquest esdeveniments que el motor de simulació realitza la simulació, tractant-los de forma endreçada segons en el pseudocodi de l'algoritme de Event Scheduling:

```
Arribada del primer esdeveniment a la LEF
Inici bucle de simulació
Mentrestant el rellotge no ha arribat el temps-final
          Executar el codi associat a l'esdeveniment extret
          Extreure l'esdeveniment més imminent de la LEF
          Avançar el rellotge fins el temps de l'esdeveniment
extret
Final mentrestant
Final bucle de simulació
```

On el codi associat a un esdeveniment pot provocar nous esdeveniments, tal i com es descriu en el pseudocodi:

```
Inici codi d'esdeveniment
     Actualitzar l'estat del model
     Actualitzar els comptadors estadístics
     Generar futurs esdeveniments i incloure'ls en la LEF
Final Codi d'esdeveniments
```

La singularitat d'aquesta tècnica és que el temps de simulació no manté cap relació amb el temps real, perquè l'execució dels esdeveniments és independent al temps que existeix entre ells, tenint tan sols en compte el seu ordre (al finalitzar de tractar un esdeveniment salta directament a la següent).

LeanSim incorpora algunes modificacions aquest algoritme per poder permetre la simulació en temps real, aspecte fonamental per donar realisme a un sistema d'entrenament.

El pseudocodi d'una simulació en temps real fonamentada en EventScheduling seria com a continuació:

```
Definir cicle de rellotge
Arribada del primer esdeveniment a la LEF
Iniciar rellotge real
Inici bucle de simulació
Mentrestant el rellotge no ha arribat en temps final
Extreure l'esdeveniment més imminent de la LEF
Si temps d'esdeveniment = temps actual
          Executar el codi associat a l'esdeveniment
Sinó
Reintroduir l'esdeveniment en la LEF
Esperar fins el següent cicle del rellotge_real
Final mentrestant
Final Bucle de simulació
```

En realitat el codi és molt més complexa, perquè s'ha de corregir possibles desviacions provocades per esdeveniments que tarden més que un cicle del rellotge real en processar-se, raó per la que també s'ha de realitzar una gestió d'esdeveniments lo més optimitzada possible.

2.2.2 Arquitectura de LeanSim

El fet de disposar d'una representació virtual en un model de simulació han canviat el requisits de la infraestructura necessària per executar-los. Ara els recursos no venen donats per la complexitat del model, sinó que la visualització del mateix augmenta els requisits necessaris per l'execució.

Per evitar aquest inconvenient LeanSim està dissenyat separant les tasques de computació de la simulació i de la representació virtual entre diferents components, de forma que poden esser executades en diferents màquines. Amb aquesta representació distribuïda s'aconsegueix disminuir els requisits dels recursos necessàries.

Com a resultat d'aquesta distribució LeanSim s'estructura en tres components que s'integren entre sí mitjançant l'utilització de fitxers XML i comunicació (LeanEditor), el motor de simulació (LeanGen) i el client de representació virtual (LeanClient).

2.2.3 LeanEditor

Aquest component proporciona les funcionalitats de creació i edició de models. Amb ell es poden definir les entitats, les màquines, els recursos i els processo que componen el model.

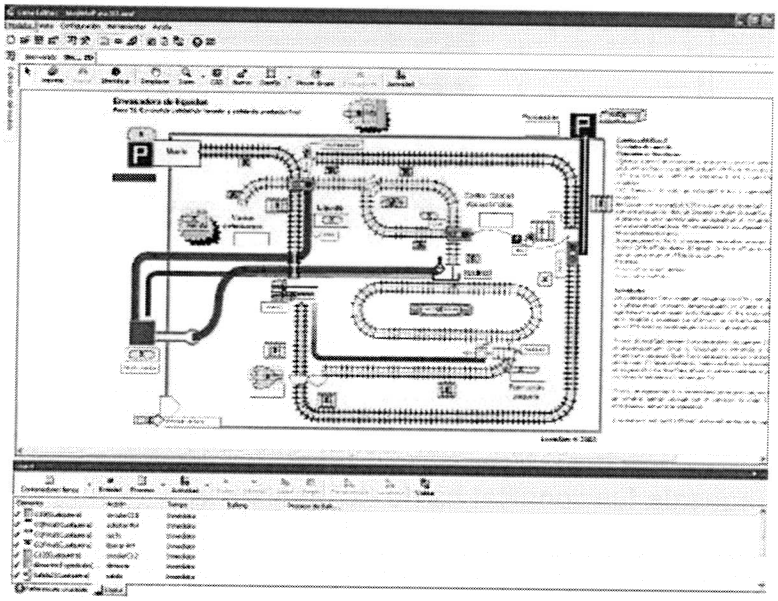
Tota aquesta informació es guarda en fitxers de text pla, fonamentats en el format dels fitxers. XML. Aquest XML, seran la base de dades dels projectes de simulació que es crearan en el LeanGen.

La interfície de LeanEditor es divideix en tres parts principals, tal i com es pot observar a il•lustració:

- Explorador del model, que permet la visualització estructurada i jeràrquica dels diferents elements (entitats, processos, objectes, etc.) que s'ha afegit al model.

- Visio 2D, en la que es pot contemplar una representació plana del model, sent la visió habitual dels simuladors comercials.

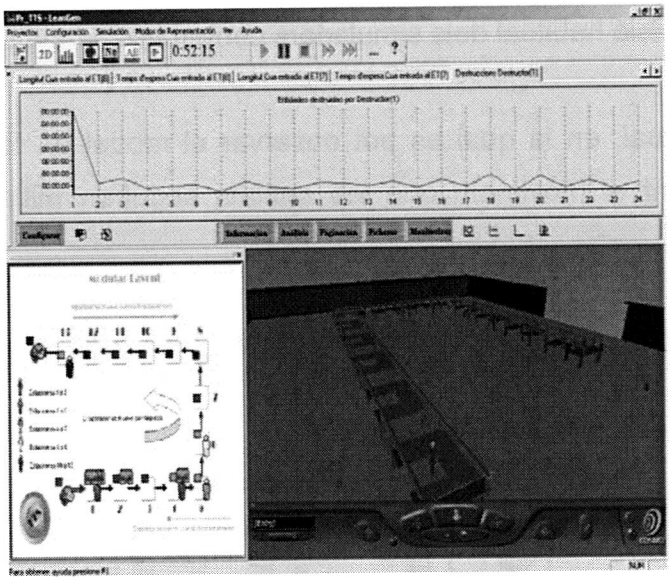
- Visió virtual, en la qual es pot observar el model en format de realitat virtual, apreciant el moviment que les entitats seguiran mitjançant les rutes definides.



2.2.4 LeanGen

Aquest component conté el motor de simulació, així com dos motors de representació (un de representació 2D i un altre per la representació virtual) i un recol•lector d'estadístics tal i com es mostra a la il•lustració.

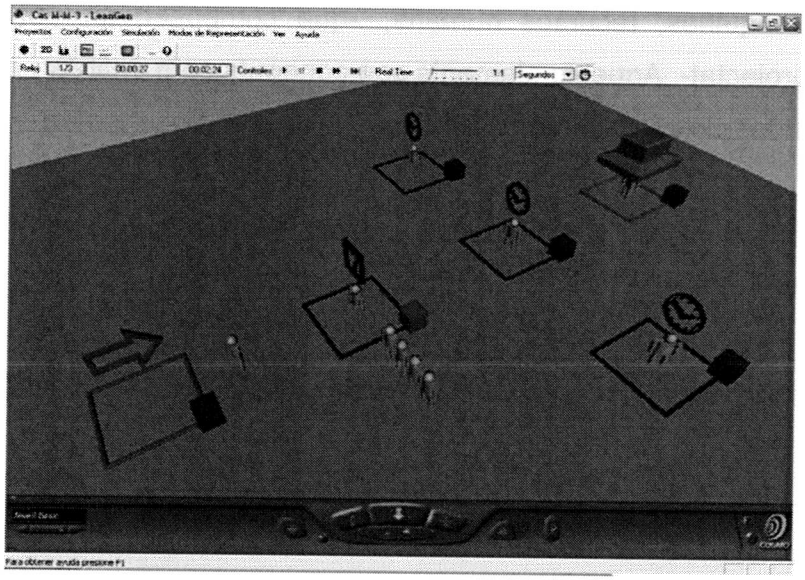
Partint d'un model construït con LeanEditor (els XML comentats en el punt anterior), LeanGen permet configurar cadascun dels objectes de simulació que integren el model, els temps de servei, el número de rèpliques, la recollida d'estadístics, etc., per així adaptar la simulació als diferents escenaris possibles.



Actualment LeanGen encara disposa d'un motor gràfic degut a que les primeres versions no disposava de LeanClient com a per a fer les representacions VRML, en un futur donat que LeanGen es un servidor de simulacions aquesta funcionalitat ha de desapareixer.

2.2.5 LeanClient

Aquesta aplicació del paquet LeamSim permet representar en una màquina remota una simulació de forma virtual, descarregant considerablement el nivell de càlcul en la màquina on s'executa la simulació.



2.3 Lògica del model

El procés de construcció i experimentació d'un model de simulació en l'entorn LeanSim clou en últim terme en l'ús d'un conjunt d'objectes, coneguts con elements lògics, que permeten definir e implementar la lògica del model. Aquests elements es classifiquen en quatre tipus diferents:

- **Entitats**
- **Màquines**
- **Operacions**
- **Processos i Accions**

2.3.1 Entitats

La majoria dels models de simulació el centre d'interès és l'observació de determinades elements –peces, matèries primes, ordres de fabricació...- que son processats –servits, transformats, analitzats,...- en un conjunt d'elements –

servidors, màquines, recursos humans,...-que composen un sistema real – existent o projectat-. Aquests elements son coneguts com entitats i en funció de la seva composició es cataloguen en:

- Entitats senzilles o simplement entitats. No es composen de cap altre tipus d'entitat.

- Entitats paquet. Es composen d'altres entitats senzilles o d'altres entitats paquets.

2.3.2 Màquines

Les màquines, o elements de simulació son aquells elements –servidors, recursos humans, magatzems, etc – que realitzen alguna activitat o transformació sobre les entitats del model.

- **Generador**: Element que emula l'arribada de les entitats al sistema virtual.

- **Transformador**: Permet transformar una entitat de determinat tipus en un altre. Aquest procés és d'utilitat, per exemple, en simuladors de processos industrials on les matèries primes es transformen en productes elaborats.

- **Empaquetador**: Element que permet l'agrupació d'entitats senzilles en paquets d'entitats.

- **Desempaquetador**: Element que permet la disgregació d'una entitat paquet en les seves entitats, senzilles.

- **Recurs**: Element pel qual competeixen els diferents servidors d'un model per realitzar els seus serveis o activitats.

- **Magatzem**: Permet l'estància de diferents entitats que son alliberades a petició d'altres entitats des d'altres màquines.

- **Controladors**: Elements que permeten gestionar els fluxos que segueixen les entitats pel model. Son fonamentalment bifurcadors, selectors, enrutadors i senyalitzadors.

- **Destructor**: Element que extrau les entitats del sistema.

- **Màquina genèrica**: Element que permet l'execució d'operacions definides per l'usuari.

Tots els elements de simulació, es caracteritzen per tenir associats: un conjunt de propietats o característiques i un estat de simulació – lliure, servei, avariat,...- que condiciona la progressió de la simulació y varien segons la lògica del model, dues cues (un d'entrada i una de sortida) que utilitzen les entitats quan una màquina a utilitzar ja està ocupada, etc.

LeanSim permet definir múltiples instàncies d'un element de simulació. Totes elles amb la mateixa configuració però diferent posicionament dins el model gràfic.

2.3.3 Operacions

Una operació s'entén com una abstracció d'una activitat del món real, que té lloc a un lloc a les màquines explicades en el punt anterior.

- **Abandonar**: L'entitat abandona el sistema.

- **Emmagatzemar**: Emmagatzemar l'entitat al magatzem on es realitza l'acció.

- **Bifurcar**: L'entitat pot canviar de procés en funció d'una condició.
- **Desempaquetar**: Realitza el desempaquetat del contingut d'una entitat.
- **Destruir**: Destrueix l'entitat i recull els estadístics.
- **Empaquetar**: Realitza el empaquetat d'una o més entitats d'un paquet.
- **Enviar senyal**: Envia una senyal de sincronisme i/o comunicació.
- **Enrutament**: Canvis de procés en funció d'una distribució acumulativa.
- **Entrar**: Un entitat entra en un sistema.
- **Generar**: Genera una entitat o conjunt d'entitats i les introdueix al model.
- **Alliberar recurs**: L'element allibera el recurs especificat en l'acció.
- **Modificar**: Modifica un atribut d'una entitat o d'un element.
- **Moure**: Mou un element de simulació a una posició especifica.
- **Selecció element**: Sol•licita una instancia d'un conjunt d'elements.
- **Sol•licitar recurs**: Petició d'un o més recursos.
- **Transformar entitat**: Transformar l'entitat en un conjunt de noves entitats.
- **Executar Script**: Execució d'un script de programació de programació.
- **Bifurcar Script**: L'entitat canvia de procés segons un script de programació.

- **Route Script**: Combinació de les dues operacions anteriors (executar i bifurcar)

- **Usuari**: Accions definides per l'usuari, en les quals l'entitat es demora en el temps.

2.3.4 Procés i Accions

Els processos es defineixen per cadascun dels diferents tipus d'entitats i defineixen el comportament concret d'aquestes. Un procés està composat per un conjunt d'accions que realitza l'entitat de forma seqüencial.

Una acció s'entén com l'execució concreta d'una operació d'una màquina determinada. Les accions s'agrupen formant els procés que defineixen el comportament de les entitats .