PFC-J2EE

Característiques dels Generadors de Codi

Estudiant: Jordi Forner Ferreres
Estudis: Enginyeria Informàtica

Consultor: Jordi Ceballos Villach
Data: 15/01/2007

A Isabel i als meus fills: Marta, Alex i Pol.

Característiques dels generadors de codi.

El projecte està relacionat amb coneixement d'un dels estàndards de desenvolupament de programari dirigit per models (MDD), en particular la implementació MDA (Arquitectura dirigida per models). Aquest paradigma de desenvolupament tracta sobre la generació automàtica de codi a partir del model d'un sistema. Els models són representats gràficament mitjançant UML (Llenguatge unificat de modelatge) amb l'ajut d'eines de modelatge. Posteriorment aquests models poden ser transformats, emmagatzemats i intercanviats en un format estàndard com és el XMI (Intercanvi de Metadades en XML). A partir d'aquests documents XMI una aplicació generadora de codi pot transformar el fitxer XMI en codi executable com per exemple per una aplicació basada en l'arquitectura J2EE.

El primer objectiu del projecte serà el de realitzar un estudi comparatiu de les característiques d'una sèrie de generadors de codi disponibles al mercat. Aquest estudi permetrà elaborar un informe on s'especificaran les característiques anunciades pels generadors de codi.
Un altre objectiu serà el verificar aquestes característiques ofertes pels generadors. Per això es realitzaran unes proves amb els generadors de codi més destacats en l'estudi comparatiu per tal de validar les característiques anunciades. Aquesta prova consistirà en dissenyar un model d'una aplicació, exportar-la en format XMI i transformar-la en codi executable mitjançant els generadors de codi.

Una vegada dut a terme el projecte s'espera:

- Obtenir uns coneixements de les tecnologies MDA i XMI en general i en particular l'estat de l'art, així com les característiques, de la generació de codi a partir de models representats en format XMI.
- Conèixer amb més detall les característiques reals d'alguns dels generadors de codi escollits.

L'estudi permet tenir una visió de les possibilitats reals dels generadors de codi, les seves característiques més destacades i les seves mancances. Finalment s'inclou una proposta de millora per incorporar als generadors de codi.

Índex

Índex de figures.

1. Capítol 1: Introducció.

Els documents XMI (subconjunt del llenguatge XML) són un estàndard per la representació de models. Les eines de modelatge (amb UML) poden exportar els seus models en format XMI. Els generadors de codi transformen aquests documents XMI en codi executable per una plataforma determinada.

Encara que aquest codi no proporciona tota la funcionalitat del sistema sí que és un bon punt de partida. Aquest codi generat automàticament pot integrar-se amb una codificació manual per tal d'obtenir el sistema final amb plena funcionalitat.

Aquests generadors de codi juntament amb les eines de modelatge amb UML són els representants del paradigma del desenvolupament orientat per models en particular el MDA (Model Driven Achitecture) proposat per OMG (Object Management Group).

Una eina de modelatge pot representar amb UML un model independent de la plataforma (PIM) i exportar-lo en format XMI. Aquest model pot ser intercanviat amb altres eines de modelatge i també amb generadors de codi. Aquest generador de codi transforma el model independent (PIM) en un altre d'específic (PSM) per una plataforma i llenguatge determinats.

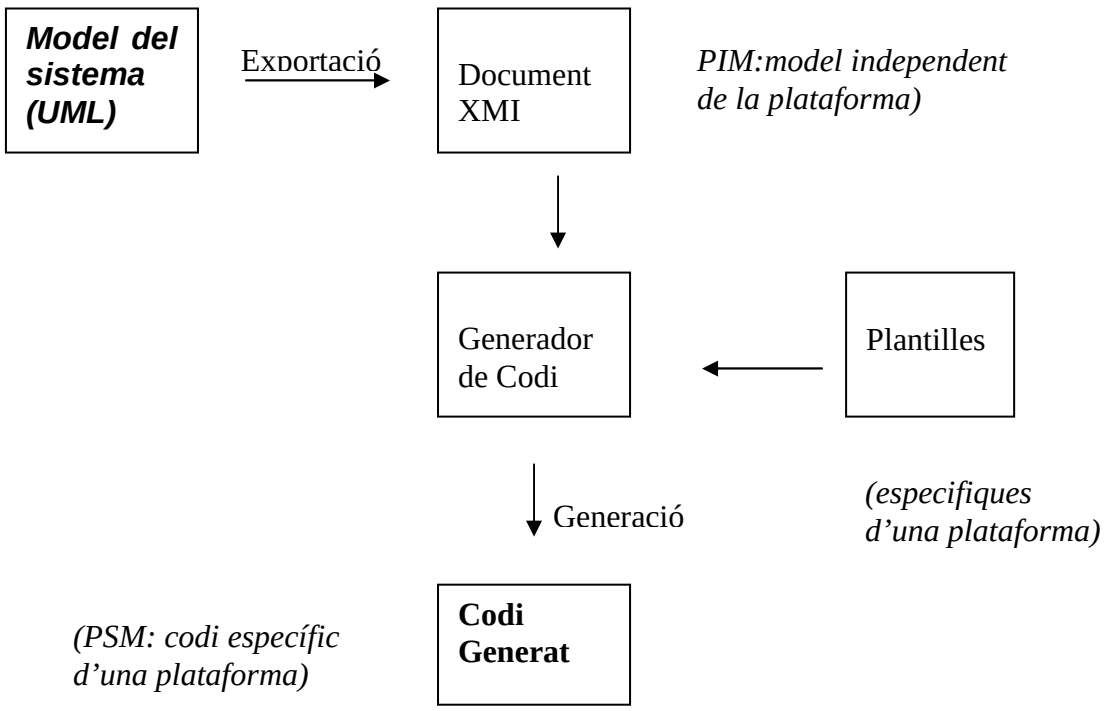


Fig.1 : La transformació del model independent de la plataforma (PIM) dona lloc a un model específic de la plataforma (PSM).

Els objectius del projecte són:

- per una part identificar les característiques que serveixen per catalogar i comparar les diferents eines generadores de codi i
- per altra part, una vegada comparats aquests generadors de codi, seleccionar-ne els més destacats i avaluar-ne les seves característiques reals.

Primerament s'ha realitzat l'estudi de les característiques de quatre generadors seleccionats. Aquesta relació de característiques ha servit per poder comparar entre els generadors i escollir-ne un parell per avaluar amb més detall les seves característiques.

Per realitzar aquesta avaluació s'han realitzat una sèrie de proves amb els generadors de codi per tal de verificar si aquelles característiques que anuncia el producte es compleixen realment o no, i si ho fan, en quin grau ho aconsegueixen.

Les proves consisteixen en realitzar, amb l'ajut d'una eina de modelatge, un model prototip i un model d'una aplicació. Exportar aquests models en format XMI i que cada generador generi el codi corresponent.

L'aplicació, inspirada en un exemple facilitat per el consultor, és el típic sistema de gestió per realitzar el manteniment d'un catàleg de productes. L'arquitectura del sistema és una arquitectura distribuïda per components on tenim els següents elements:

- El component client que realitza les peticions i rep les respostes del sistema.
- El component de presentació encarregar de traslladar les peticions a la capa de negoci i realitzar les presentacions de resultats per retornar-los al client.
- El component de negoci que respon a les peticions del client i conte la lògica del sistema.
- Els components d'entitat que gestionen l'accés a la BBDD.

L'anàlisi del codi generat a partir dels documents XMI ha servit per realitzar un estudi del funcionament real de les eines generadores de codi i veure fins qui punt compleixen les expectatives anunciades.

1.1 Descripció del Projecte

1.1.1 Objectius

- Dintre del context del desenvolupament d'aplicacions dirigides per models (MDA) i la representació d'aquests models amb format XMI realitzar un estudi dels generadors de codi i les seves característiques.
- Estudi de les característiques reals dels generadors de codi més destacats de l'estudi preliminar.

1.1.2 Resultats

- Coneixement de l'estat de l'art de les aplicacions generadores de codi.
- Coneixement a fons de les característiques d'alguns dels generadors de codi més destacats.

1.2. Abast del Projecte

Es considera dintre de l'abast d'aquest projecte:

I. Comparativa de les aplicacions generadores de codi java a partir de fitxers en format XMI.
II. Verificació de les característiques reals dels generadors de codi destacats en l'estudi preliminar.
III. Memòria i Presentació Virtual.

I. Comparativa d'aplicacions generadores de codi.

S'ha de realitzar una comparativa entre diverses aplicacions generadores de codi java. Aquesta comparativa s'ha de basar en una sèrie de característiques que permetin avaluar les diferencies entre els diversos generadors. Una vegada avaluada un primer generador s'ha de realitzar un lliurament amb totes les característiques objectes d'estudi per tal de validar-les. A continuació es procedirà a l'estudi de la resta d'aplicacions generadores.

Es tracta de generar un informe que reflecteixi els diferents graus de compliment d'una sèrie de característiques que permetin esbrinar quines aplicacions generadores de codi són les més destacades.

II. Verificació de les característiques reals dels generadors de codi.

La verificació de les característiques reals dels generadors destacats en l'estudi anterior és duu a terme realitzant un model prototip i un model d'una aplicació, exportant els models a format XMI, generant codi i finalment validant si es compleixen les característiques previstes.

III. Memòria i Presentació Virtual.

Elaboració de la memòria del projecte i de la presentació virtual

1.3. Organització del Projecte

1.3.1 Activitats

- Estudi previ de característiques d'un generador de codi

  Objectius Recull de característiques d'un generador de codi
  Lliurables Document Preliminar
  Durada Estimada 7 dies.
  Esforç Estimat 21 hores

- Estudi comparatiu de generadors de codi java-XMI

  Objectius Comparativa de les aplicacions generadores de codi java a partir de fitxers XMI
  Lliurables Document Comparatiu
  Durada Estimada 21 dies.
  Esforç Estimat 63 hores

- Generació de Codi a partir de XMI

  Objectius Estudi de les característiques reals dels generadors de codi seleccionats.
  Lliurables Document de l'Estudi
  Durada Estimada 45 dies.
  Esforç Estimat 135 hores

- Final del Projecte

  Objectius Tancament del Projecte
  Lliurables Memòria, Presentació Virtual
  Durada Estimada 26 dies.
  Esforç Estimat 78 hores

1.3.2 Fites i Lliurables

- Dia 17/10/2006: Document previ de característiques d'un generador de codi.
- Dia 07/11/2006: Document comparatiu de les aplicacions generadores de codi.
- Dia 22/12/2006: Document de l'estudi de les característiques reals dels generadors.
- Dia 15/01/2006: Memòria del Projecte i Presentació Virtual del Projecte

Capítol 2: Els generadors de codi.

Inicialment s'ha realitzat un estudi sobre la generació de codi i les tecnologies implicades en particular XMI. La informació bàsica sobre generació de codi s'ha obtingut a partir del cercador Google.

Podríem resumir que els quatre beneficis fonamentals de la generació de codi són:

- **Qualitat**: La qualitat depèn de la qualitat de les plantilles que generen el codi per tant assegurar la qualitat de les plantilles (corregir els errors) garanteix un codi de qualitat en cada generació cosa que no es pot assegurar en una codificació manual.

- **Consistència**: El benefici de l'automatització del codi és el que tots els mètodes, classes i noms d'atributs segueixen unes convencions establertes.

- **Productivitat**: A més a més de generar més ràpidament el codi que amb codificació manual, la generació automàtica allibera a l'enginyer de fer tasques reiteratives i pot ocupar el seu temps en tasques més creatives.

- **Abstracció**: Les plantilles dels generadors ens permeten codificar amb independència dels detalls d'implementació i del llenguatge de codificació.

El XMI és un document XML que segueix un esquema dissenyat per tal de representar models (també meta-models o meta-data en general) descrits en llenguatge MOF i UML en particular. L'avantatge de la codificació amb XMI és que es tracta d'un estàndard. Per tant pot servir per intercanviar models entre les diferents eines de modelatge amb UML. També els diferents generadors de codi poden generar codi executable a partir d'aquests models representats en el document XMI.

Una vegada realitzat l'estudi sobre XMI s'ha procedit a realitzar una selecció dels generadors disponibles a internet. A partir de suggestions fetes pel consultor s'han localitzat 4 generadors candidats per l'estudi:

- Xcoder
- Axgen
- CodeGenie
- IQgen

Aquests generadors han estat instal·lats i provats per tal de veure el seu funcionament. S'ha procedit a revisar la documentació associada a cada eina i s'ha buscat informació addicional a partir de les pàgines web dels productes.

Amb tota aquesta documentació s'ha realitzat una relació inicial de les característiques més destacades dels productes. Aquestes característiques ens ajudaran a avaluar els diferents generadors objectes de l'estudi. Aquesta relació de característiques ha estat complimentada amb els valors particulars del generador iQgen i s'ha documentat en forma de quadre.

Per tal de validar aquesta llista de característiques s'ha lliurat aquest quadre al consultor. Una vegada validada la relació inicial s'ha elaborat el quadre definitiu amb les característiques a avaluar.

Les característiques dels generadors de codi s'han agrupat en tres categories:

1. **Generals**: Característiques referents a l'ús de l'eina i a la seva distribució i disponibilitat.

    - ***URL***: Adreça electrònica del proveïdor/producte que dóna suport a l'eina.

    - ***Ultima Versió***: Número de versió disponible i data d'edició de l'ultima versió de l'eina.

    - ***Codi font disponible***: Indicador de si el codi font de l'eina està disponible (Open Source) i amb quines condicions.

    - ***Preu***: Cost en euros de la versió comercial.

    - ***Interfícies d'usuari***: Relació de les diferents interfícies que disposa l'eina per poder accedir a la funcionalitat.

    - ***Proporciona extensions de l'eina***: Indica si l'eina disposa d'una extensió de l'aplicació que pot integrar-se amb una altra eina, típicament un IDE.

    - ***Disposa d'API***: Indica si l'eina disposa d'una interfície programàtica per poder accedir a la funcionalitat.

    - ***Sistema Operatiu***: Sistemes Operatius pels que està disponible l'eina.

2. **Característiques de l'Entrada (Model del sistema)** : Totes aquelles característiques que fan referència al model del sistema, des de l'eina que el genera fins a les versions del llenguatge de modelatge (UML) o del llenguatge de representació (XMI).

    - ***Eines de modelatge en UML suportades***: Relació de les eines que s'usen per modelar un sistema i creen el document XMI que el generador de codi reconeix i és capaç de transformar en codi executable.

    - ***Exportador del document XMI***: Normalment és l'eina de modelatge la que exporta el document XMI a partir del model. Hi ha generadors de codi que poden exportar el XMI accedint directament al model emmagatzemat per l'eina de modelatge.

    - ***Versió UML*** : Número de la versió del llenguatge UML que reconeix el generador.

    - ***UML Estàtic***: La relació del tipus de diagrames estàtics o d'estructura que l'eina sap interpretar i generar codi. Concretament són els diagrames de classes i els d'interfície.

    - ***UML Dinàmic***: Relació de tipus de diagrames de comportament que l'eina sap interpretar i generar codi. Típicament són els d'activitat, d'estats i de seqüència.

    - ***Restriccions OCL***: Indicador de si l'eina sap interpretar i generar codi per les restriccions escrites en el llenguatge OCL.

    - ***Versions XMI suportades***: Relació de les versions del llenguatge XMI que reconeix o interpreta l'eina.

3. **Característiques de la Sortida (Generació del codi)** : Les característiques relacionades amb la generació del codi, des del mètode de transformació fins al tipus de codi i llenguatge generat.

    - ***Mètode de Transformació***: Tècnica utilitzada per transformar el document XMI en codi executable be a través de plantilles com codi.

    - ***Transformació configurable***: Indicador de la possibilitat de configurar o modifica/crear les plantilles o codi per tal d'ajustar la generació del codi a les necessitats particulars.

    - ***Permet Codi d'usuari***: Indica si l'eina permet la integració del codi generat amb altre codi i com és pot realitzar.

    - ***Llenguatges generats***: Relació de llenguatges i varietats que és capaç de generar l'eina.

    - ***Suport Persistència***: Tipus de suport a la persistència en BBDD. Típicament si genera SQL o altre tipus de llenguatge relacionat amb la persistència i/o recuperació de la informació.

2.1 Característiques dels generadors de codi.

iQgen és un generador que dóna suport al desenvolupament orientat a models (MDA) utilitza el XMI com llenguatge per representar els models i el JSP com llenguatge per configurar les transformacions del model al codi executable. Permet la generació de llenguatges com el Java, C, C++, C#, C, XML, SQL i codi per plataformes com J2EE i .NET. Reconeix el XMI de les principals eines de modelatge.

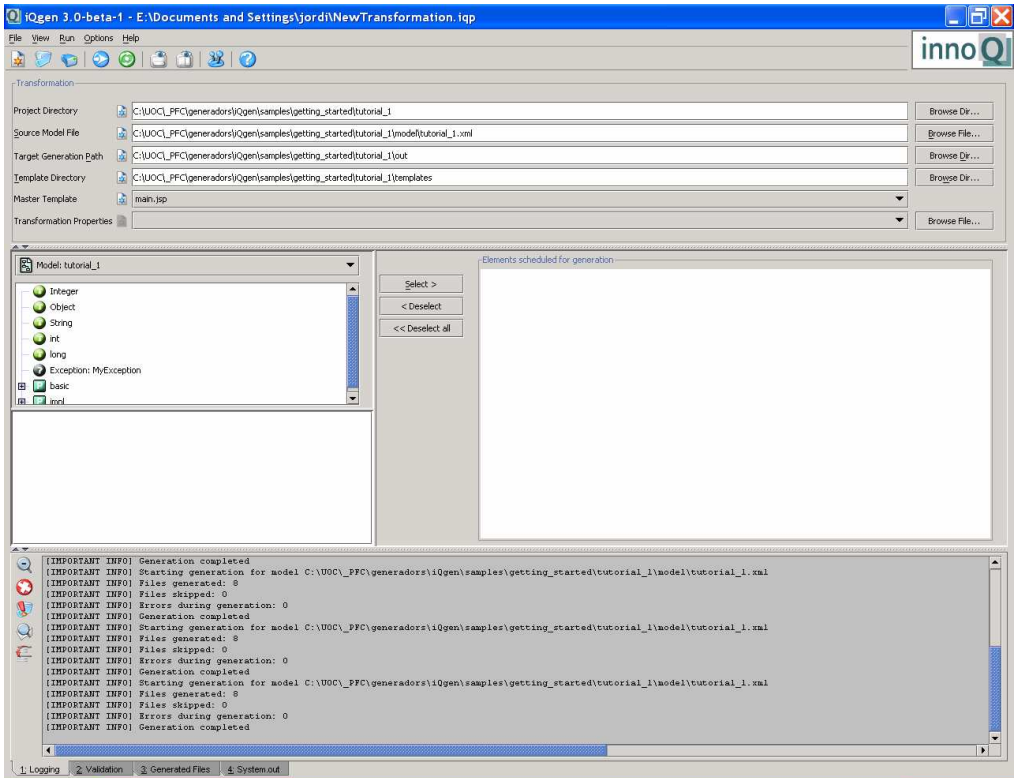


Fig. 2: Aquesta és la pantalla principal de l'aplicació iQgen. A la part superior es pot indicar la ubicació dels recursos com per exemple: el directori del projecte, el nom del XMI, el directori de sortida, etc. La part central de la pantalla permet seleccionar les parts del model per les que es generarà el codi. La part inferior de la pantalla permet visualitzar les diferents sortides de la generació.

Característiques	Valors pel generador iQgen
Generals	**Iqgen**
URL	http://www.innoq.com
Ultima Versió	3.0 Beta (abr-2006)
Codi font disponible	No (codi Java)
Preu	
Interfícies d'usuari	Disposa d'una interfície gràfica (GUI) per especificar els directoris on es troben els recursos. Les tasques es poden automatitzar i integrar mitjançant l'eina Ant
Proporciona Extensions de l'eina	Disposa d'una extensió (plug-in) integrable a l'IDE Eclipse
Disposa d'API	Sí, compta amb llibreries per escriure els JSP i accedir al model (classes i interfícies)
Sistema Operatiu	El generador està disponible per tots el S.O que disposen d'una maquina virtual Java 1.3 o superior, per exemple: Windows,Linux,Mac
Característiques de l'Entrada (Model del Sistema)	
Eines de modelatge UML suportades	El generador és capaç d'interpretar i generar codi a partir del XMI generat per les següents eines de modelatge: Rational Rose, MID Innovator, Microtool ObjectiF, Together,ObjectDomain i ArgoUML
Exportador del XMI	L'eina de modelatge exporta el document XMI. El generador de codi també por exportar el XMI a traves de les llibreries de Novosoft
Versió UML	1.0,1.1
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies
UML Dinàmic	El generador és capaç de generar les classes per un diagrama d'Activitat. En l'exemple de l'aplicació a partir d'un diagrama d'activitat genera el codi .html corresponent a la navegació entre les diferents pantalles de l'aplicació.
Restriccions OCL	No les considera
Versions XMI suportades	1.0,1.1,1.2,1.3
Característiques de la Sortida (Generació de codi)	
Mètode de Transformació	L'eina utilitza Plantilles JavaScript per transformar el codi XMI en codi executable
Transformació configurable	L'eina permet la modificació de les plantilles i la creació de noves per adaptar la transformació a les pròpies necessitats
Permet Codi d'usuari	Disposa d'àrees protegides per incorporar codi d'usuari. Aquest codi es manté entre una generació i una altra.
Llenguatges generats	Java (J2EE), C++, C# (.NET), C.,XML(per descriptors de desplegament, ant,..)
Suport Persistència	Dóna suport a la persistència generant sentencies SQL

Xcoder és un generador de codi amb una senzilla interfície d'usuari. No utilitza plantilles per generar codi. Reconeix els XMI generats per Rational Rose i Magic Draw i tan sols és capaç de generar codi per diagrames de classes e interfícies.

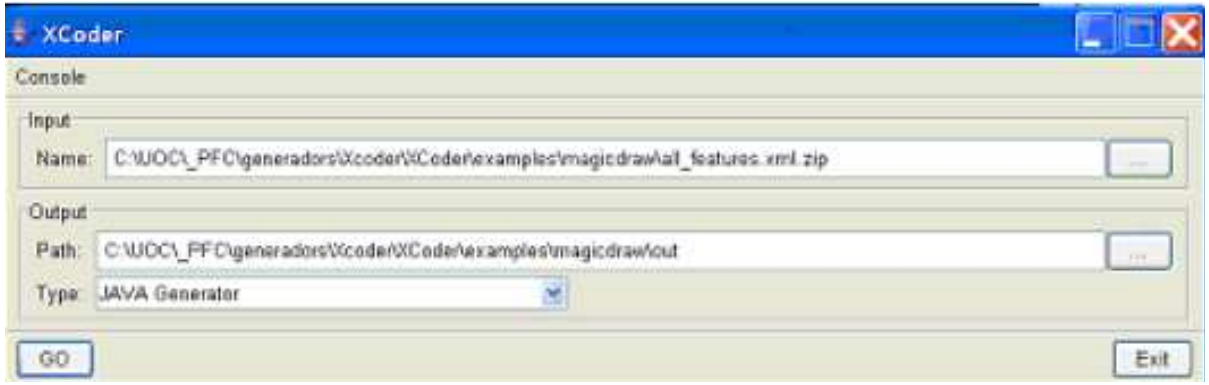


Fig. 3: L'eina disposa d'aquesta interfície d'usuari on especificar la ubicació del fitxer XMI (o la versió empaquetada .zip) i el directori on s'ubicaran els fitxers de codi generats. També pot escollir-se el tipus de generació: o be Java o bé J2EE.

Característiques	Valors pel generador Xcoder
Generals	**Xcoder**
URL	http://www.liantis.com
Ultima Versió	1.1 (gen-2004)
Codi font disponible (llenguatge codificació)	Sí (codi Java)
Preu	Gratuït, es pot usar sota llicència GPL
Interfícies d'usuari	Disposa d'una senzilla interfície gràfica (GUI) per especificar els directoris on es troben els recursos. Les tasques es poden automatitzar i integrar mitjançant l'eina Ant
Proporciona Extensions de l'eina	Disposa duna extensió (plug-in) integrable a Rational Rose
Disposa d'API	Disposa d'un ampli conjunt de paquets i classes per accedir al model i generar el codi
Sistema Operatiu	El generador està disponible per tots el S.O que disposen d'una maquina virtual Java 1.4 o superior, per exemple: Windows,Linux,Mac
Característiques de l'Entrada (Model del Sistema)	
Eines de modelatge UML suportades	El generador és capaç d'interpretar i generar codi a partir del XMI generat per les següents eines de modelatge: Rational Rose, Magic Draw
Exportador del XMI	L'eina de modelatge exporta el document XMI.
Versió UML	1.4
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies
UML Dinàmic	No
Restriccions OCL	No les considera
Versions XMI suportades	1.1
Característiques de la Sortida (Generació de codi)	
Mètode de Transformació	No disposa de plantilles. El codi el genera l'aplicació amb les seves pròpies llibreries.
Transformació configurable	Es pot modificar el codi per tal d'adaptar la generació a les pròpies necessitats.
Permet Codi d'usuari	Disposa d'àrees protegides per incorporar codi d'usuari. Aquest codi es manté entre una generació i una altra.
Llenguatges generats	Java (J2EE), C++
Suport Persistència	Dóna suport a la persistència generant sentencies SQL

CodeGenie : La principal característica d'aquesta eina és que mitjançant documents XMI s'obté tant el model de l'aplicació com el model de l'arquitectura. Aquests models han estat creats amb una eina de modelatge com Rational Rose. Disposa de la generació de codi per Java encara que es pot sol·licitar també per C++ i les extensions de J2EE i .NET.

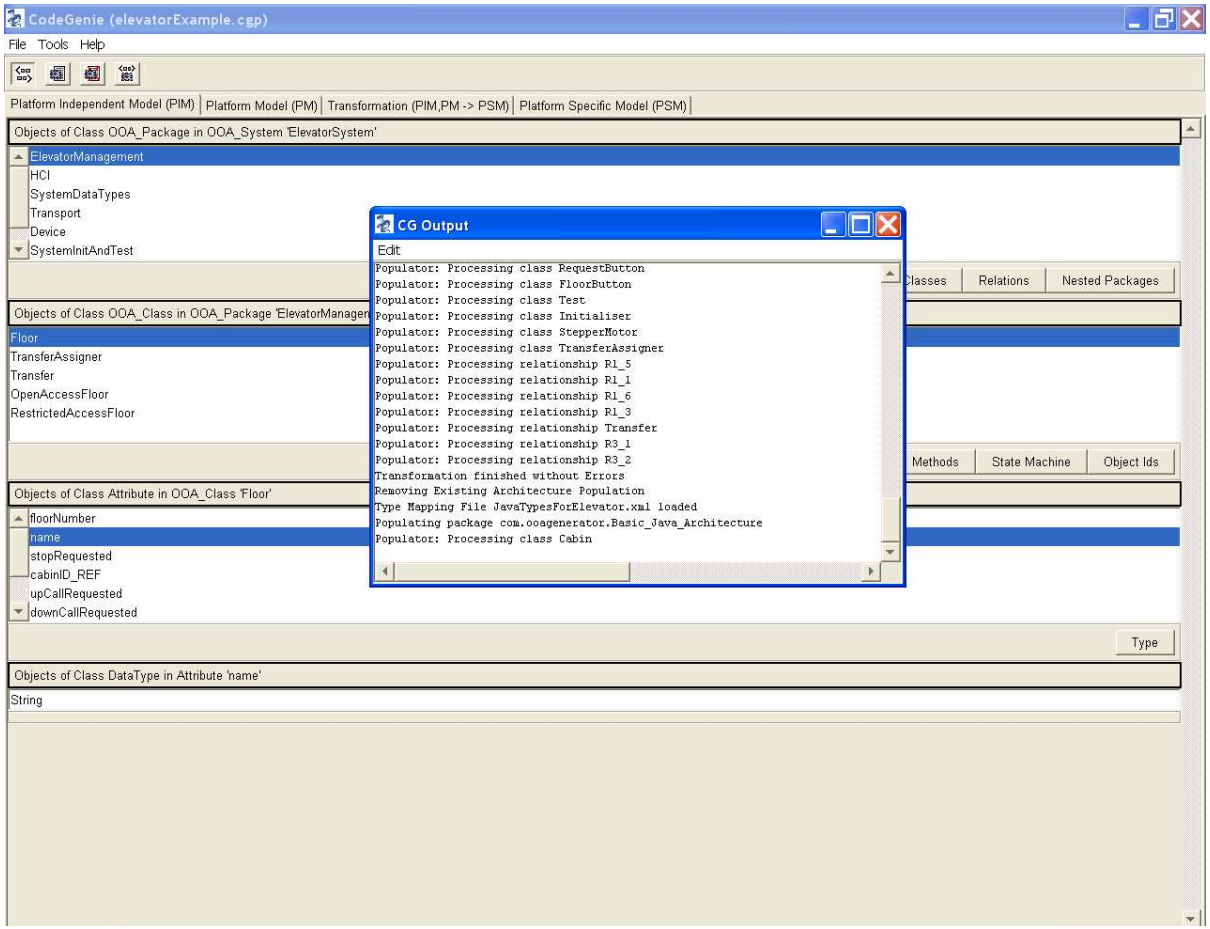


Fig. 4: La pantalla principal de l'eina està articulada en pestanyes. La primera pestanya que és la visible al fons de la imatge permet visualitzar la composició del model. Les altres pestanyes són per configurar el model de la plataforma i les transformacions del model. Una vegada arrenca la generació del codi apareix aquesta pantalla on es mostra la sortida (traça del procés) de la generació.

Característiques	Valors pel generador CodeGenie
Generals	**CodeGenie**
URL	http://www.ooagenerator.com
Ultima Versió	4.3 (jun-2005)
Codi font disponible (llenguatge codificació)	No (codi Java)
Preu	
Interfícies d'usuari	Disposa d'una interfície gràfica (GUI) per especificar els directoris on es troben els recursos i veure la composició del model.
Proporciona Extensions de l'eina	No
Disposa d'API	No
Sistema Operatiu	El generador està disponible per tots el S.O que disposen d'una maquina virtual Java 1.4 o superior, per exemple: Windows,Linux,Mac
Característiques de l'Entrada (Model del Sistema)	
Eines de modelatge UML suportades	El generador és capaç d'interpretar i generar codi a partir del XMI generat per les següents eines de modelatge: Rational Rose, Together
Exportador del XMI	L'eina de modelatge exporta el document XMI.
Versió UML	1.3,1.4
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies
UML Dinàmic	El generador és capaç de generar les classes per diagrames d'Activitat i Estats. En l'exemple de l'aplicació, per cada classe amb estats es genera una classe addicional que modela els diferents estats de la classe. Els diferents estats de la classe contenen per cada esdeveniment el seu resultat per tal de modelar el comportament de la classe.
Restriccions OCL	No les considera
Versions XMI suportades	1.0
Característiques de la Sortida (Generació de codi)	
Mètode de Transformació	L'aplicació obté el model independent de la plataforma (PIM) a traves d'un document XMI exportat per una eina de modelatge. Per una altra part obté un model de la plataforma (PM) , en aquest cas específic per java, també a través d'un document XMI també exportat per una eina de modelatge. Amb l'ajut un codi transformador específic (per generar codi java) i unes plantilles Java Archetipe (JARC) aquest model independent de la plataforma (PIM) es transforma en codi per un model específic (PSM) per una plataforma determinada (PM) en aquest cas java.
Transformació configurable	Pot modificar-se el model de la plataforma (PM) amb l'eina de modelat i també les plantilles JARC poden modificar-se per adaptar el codi generat a les nostres necessitats.
Permet Codi d'usuari	No explícitament però modificant les plantilles seria possible incorporar codi no generat.
Llenguatges generats	Java (J2EE), C++
Suport Persistència	No ho fa explícit se suposa que genera sentencies SQL però no està verificat.

Axgen és una eina que destaca per la seva simplicitat d'us. Com a trets a destacar tindríem l'ús de plantilles Velocity, el suport a la majoria d'eines de modelatge i el suport a la persistència a traves d'eines de mapeig objecte/relacional. Axgen pot llegir un document XMI exportat per una eina o exportar-lo ell mateix. Aquest fet li dóna independència de les versions UML i/o XMI emprades per l'eina de modelatge.

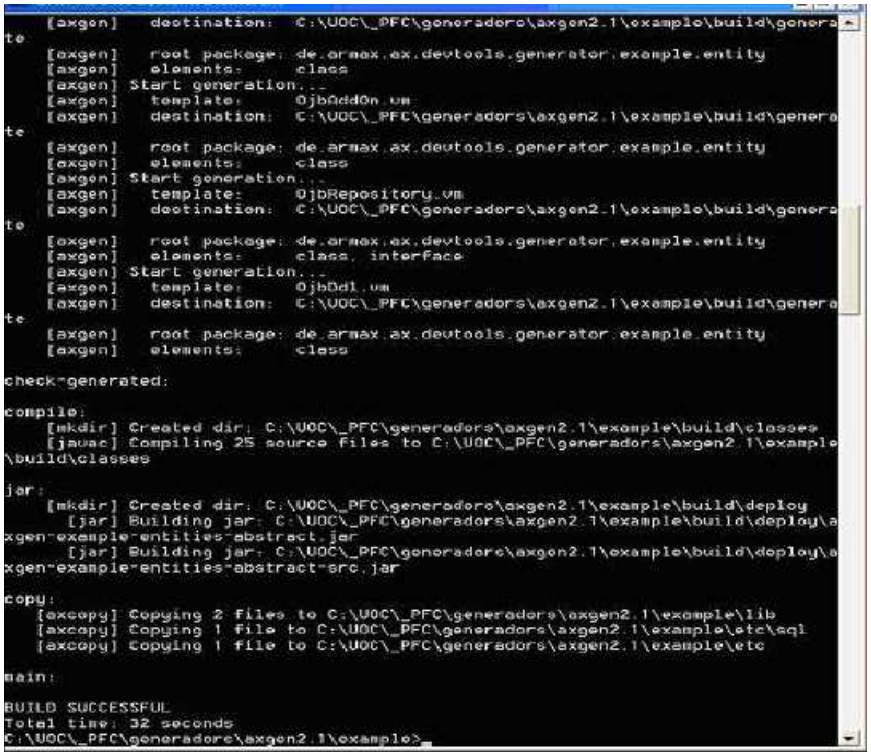


Fig. 5: Axgen no disposa d'una interfície d'usuari si no que utilitza l'eina Ant per generar el codi. Al fitxer buid.xml s'indica quin és el model i on s'ubicarà el codi generat. Una vegada configurat el fitxer por arrencar-se la generació de codi executant la instrucció Ant des de la línia de comandes.

Característiques	Valors pel generador Axgen
Generals	**Axgen**
URL	http://axgen.sourceforge.net
Ultima Versió	2.1 (data no disponible)
Codi font disponible (llenguatge codificació)	Sí (codi Java)
Preu	
Interfícies d'usuari	No disposa d'una interfície gràfica. Les tasques es poden automatitzar i integrar mitjançant l'eina Ant
Proporciona Extensions de l'eina	No
Disposa d'API	Disposa d'un ampli conjunt de paquets i classes per accedir al model i generar el codi
Sistema Operatiu	El generador està disponible per tots el S.O que disposen d'una maquina virtual Java 1.4 o superior, per exemple: Windows,Linux,Mac
Característiques de l'Entrada (Model del Sistema)	
Eines de modelatge UML suportades	Teòricament dóna suport a totes les eines ja que la mateixa eina crea el XMI a partir del repositori de l'eina de modelatge a partir de la seva API i altres llibreries incorporades (NetBeans MDR i NsUml)
Exportador del XMI	L'eina de modelatge exporta el document XMI. També por exportar el XMI a traves de les llibreries pròpies i d'altres, com: NetBean MDR o de NsUml.
Versió UML	1.4.
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies
UML Dinàmic	No
Restriccions OCL	No les considera
Versions XMI suportades	1.1.
Característiques de la Sortida (Generació de codi)	
Mètode de Transformació	Plantilles Velocity són plantilles de substitució on es defineix el codi que es generarà.
Transformació configurable	Aquestes plantilles poden modificar-se per adaptar el codi generat a les nostres necessitats.
Permet Codi d'usuari	Sí, poden incorporar-se a les plantilles crides a procediments propis.
Llenguatges generats	Java (J2EE)
Suport Persistència	Genera codi per eines de mapeig objecte/relacional com OJB (Object/Relational Bridge).

2.2 Quadre Comparatiu

Característiques	Generadors			
Generals	**iQgen**	**Xcoder**	**CodiGenie**	**Axgen**
URL	http://www.innoq.com	http://www.liantis.com	http://www.ooagenerator.com	http://axgen.sourceforge.net
Ultima Versió	3.0 Beta (abr-2006)	1.1 (gen-2004)	4.3 (jun-2005)	2.1 (data no disponible)
Codi font disponible (llenguatge codificació)	No (codi Java)	Sí (codi Java)	No (codi Java)	Sí (codi Java)
Preu		Gratuït amb llicència GPL		
Interfícies d'usuari	GUI,Ant	GUI,Ant	GUI	Ant
Proporciona Extensions de l'eina	Eclipse	Rose	No	No
Disposa d'API	Sí	Sí	No	Sí
Sistema Operatiu	S.O. amb JVM 1.3 o superior	S.O. amb JVM 1.4 o superior	S.O. amb JVM 1.4 o superior	S.O. amb JVM 1.4 o superior
Característiques de l'Entrada (Model del Sistema)				
Eines de modelatge UML suportades	Rational Rose, MID Innovator, Microtool ObjectiF, Together,ObjectDomain i ArgoUML	Magic Draw, Rational Rose	Rational Rose	Llibreries NetBeans MDR i NsUml
Exportador del XMI	*Document XMI exportat per l'eina de modelatge *Document XMI exportat pel generador	Document XMI exportat per l'eina de modelatge	Document XMI exportat per l'eina de modelatge	* Document XMI exportat pel generador * Document XMI exportat per l'eina de modelatge
Versió UML	1.4	1.4	1.4	1.4
UML Estàtic	Classe, Interfícies	Classe, I Interfícies	Classe, Interfícies	Classe, Interfícies
UML Dinàmic	Activitat	No	Activitat i Estats	No
Restriccions OCL	No les considera	No les considera	No les considera	No les considera
Versions XMI suportades	1.0,1.1,1.2	1.1	1.0	1.1
Característiques de la Sortida (Generació de codi)				
Mètode de Transformació	Plantilles JavaScript	Codi Java	Plantilles JARC	Plantilles Velocity
Transformació configurable	Modificació/creació de Plantilles	Generador personalitzat	Modificació de Plantilles	Modificació/Creació de plantilles
Permet Codi d'usuari	Disposa d'àrees protegides per incorporar codi d'usuari	Disposa d'àrees protegides per incorporar codi d'usuari	No	Sí
Llenguatges generats	Java (J2EE), C++, C# (.NET), C	Java (J2EE), C++	Java (J2EE), C++	Java (J2EE)
Suport Persistència	Generació de SQL	Generació de SQL		Generació de codi per OJB

2.3 Selecció dels generadors

Les característiques comunes a tots els generadors són:

- **Generació de UML estàtic**: sembla ser que totes les eines aconsegueixen generar les classes i les interfícies. Algunes poden generar els diagrames d'activitat i altres també els d'estats. Però cap de les eines avaluades és capaç per exemple de generar codi per diagrames de seqüència.

- **Codi Java**: Totes les eines tenen com llenguatge generat el Java encara que algunes també generen C++ i/o C.

- **Tasques de generació**: l'automatització de les eines de generació de codi i compilació del codi font es duen a terme a partir de Ant.

- **Llenguatge de desenvolupament de l'eina**: els propis generadors estan creats amb llenguatge Java això permet executar l'eina en altres plataformes.

Com a característiques més destacades tenim:

- **Restriccions OCL**: cap de les eines estudiades és capaç d'interpretar les restriccions OCL. És un gran inconvenient no poder generar codi a partir de les restriccions ja que això ens obliga a implementar-les manualment.

- **Disposar d'API**: per realitzar extensions de l'aplicació i fer usos de les funcionalitats exposades per l'API.

- **Disposar de les fonts**: permet el poder evolucionar l'eina i fer usos més adequats a les nostres necessitats.

- **Generar codi per J2EE**: totes les eines generen codi per aquesta plataforma que juntament amb .NET són les més populars pel desenvolupament de sistemes distribuïts.

- **Usos de plantilles de generació**: Quasi totes les eines fan usos de plantilles per la generació de codi. L'ús aquestes plantilles dóna flexibilitat ja que permet modificar-les a voluntat i adaptar-les a les necessitats de l'aplicació.

- **Integració amb codi d'usuari**: Es disposa d'unes àrees de codi protegit on incloure el codi d'usuari. Aquestes àrees es mantenen entre una generació i l'altra i això facilita la integració de codi.

Els criteris per escollir els generadors de codi seleccionats en aquest estudi han segut:

- Suport a diferents eines de modelatge.
- Suport a diagrames de classes i d'activitat.
- Disposar d'API i extensions de l'aplicació.
- Disposar de plantilles modificables.
- Permetre integrar codi d'usuari
- Altres característiques remarcables.

Els motius pels que s'han seleccionat els generadors iQgen i Axgen són:

- Donen suport a una gran varietat d'eines de modelatge.
- IQgen genera codi a partir dels diagrames d'activitat.
- Ambdues eines disposen d'API i iQgen disposa d'una extensió per Eclipse.
- Ambdues disposen de plantilles modificables.
- Ambdues eines permeten integrar codi d'usuari amb la generació automàtica
- IQgen disposa d'una bona interfície gràfica
- Axgen disposa de suport addicional a la persistència

Els motius pels que no s'han seleccionat CodeGenie i Xcoder són:

- Donen un suport limitat a una o dues eines de modelatge.
- Xcoder no dona suport als diagrames d'activitat
- CodeGenie no disposa d'API ni d'extensions de l'aplicació
- CodeGenie no permet la integració del codi d'usuari amb el codi generat automàticament.

Per les raons relacionades anteriorment els generadors escollits per l'estudi de les característiques reals dels generadors han estat:

- **IQgen**
- **Axgen**

Capítol 3: Anàlisi dels generadors de codi.

Per tal d'analitzar el funcionament real dels generadors de codi seleccionats efectuarem el seguen procés d'avaluació:

- Escollir l'eina de modelatge
- Dissenyar el model
- Exportar el model a format XMI
- Generar codi executable
- Analitzar el codi generat

Els models a provar seran:

- Prototip de diagrama de classes amb:
  - tot tipus de visibilitat d'atributs i operacions
  - tot tipus de relacions entre classes
- Aplicació exemple amb:
  - Diagrama de classes
  - Diagrama d'activitat

L'aplicació exemple a modelar es un sistema de gestió de catàlegs de productes. Les funcionalitats a desenvolupar seran:

- Llistar categories
- Llistar productes
- Cercar productes
- Gestionar productes
- Entrar al sistema

L'aplicació s'implantarà en un entorn distribuït per components i s'utilitzarà la tecnologia J2EE, les capes on residiran els components de l'aplicació seran:

- La capa client, on el component s'allotjarà en el navegador.
- La capa de presentació on els components residiran en un servidor de pàgines.
- La capa de negoci on els components s'allotjaran en un servidor d'aplicacions
- La capa de persistència on uns components residiran en el servidor d'aplicacions i els altres en el servidor de dades.

El procés d'avaluació s'aplicarà als generadors de codi:

- iQgen
- Axgen

Els resultats de l'avaluació s'inclouran en taules on es reflexarà el funcionament real dels generadors de codi.

3.1 Prototip: Diagrama de classes

S'ha realitzat un prototip de model de classes amb:

- Atributs públics, privats i protegits
- Operacions publiques, privades i protegides
- Relacions d'associació
- Relacions de composició
- Relacions d'agregació
- Relacions d'herència
- Classes abstractes

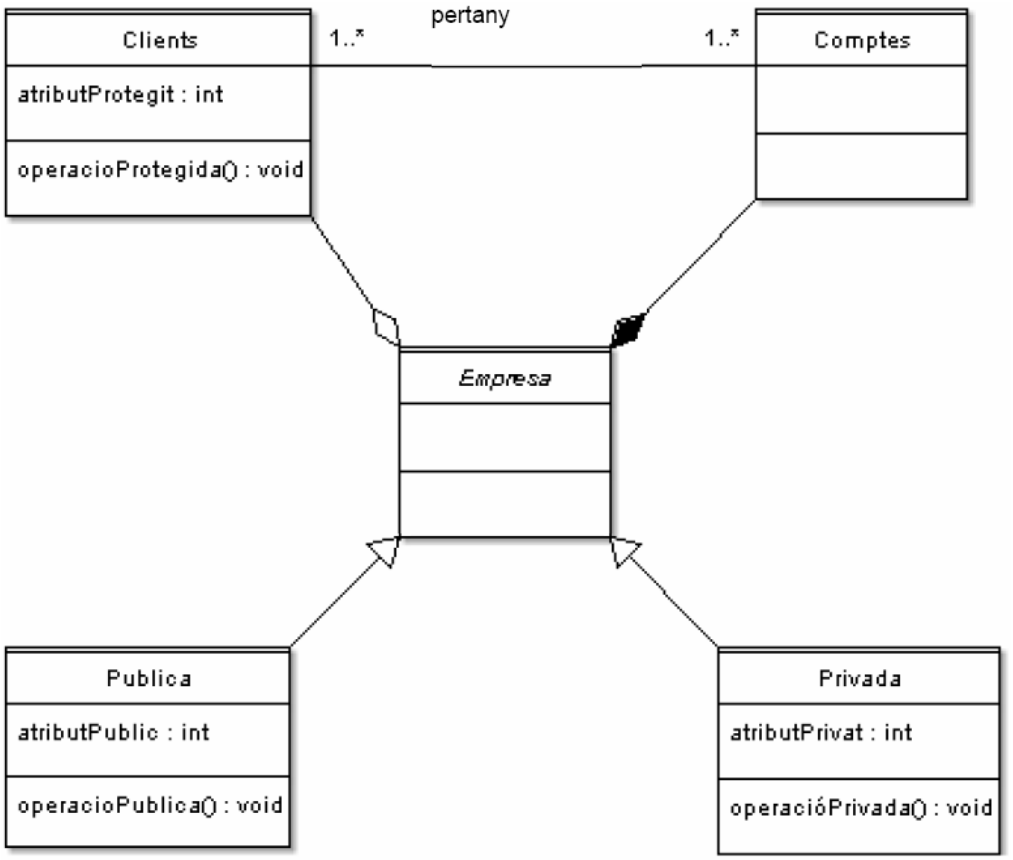


Fig. 6: Aquest diagrama de classes representa un prototip on es poden veure una representació dels diferents elements que poden aparèixer en un diagrama de classes. Hi ha atributs i operacions privats, protegits i públics. També hi ha relacions d'associació, d'agregació, de composició i d'herència. La classe Empresa està definida com abstracta.

3.2 Aplicació: Catàleg de Productes

L'aplicació escollida per realitzar les proves de generació automàtica de codi és una aplicació de gestió d'un catàleg de productes. Aquesta aplicació és la típica de comerç electrònic per una empresa.

L'aplicació permet gestionar (crear, modificar, esborrar i consultar) un catàleg de productes d'una determinada categoria. També és podran llistar totes les categories del catàleg de productes així com tots els productes que hi ha dintre d'una categoria determinada.

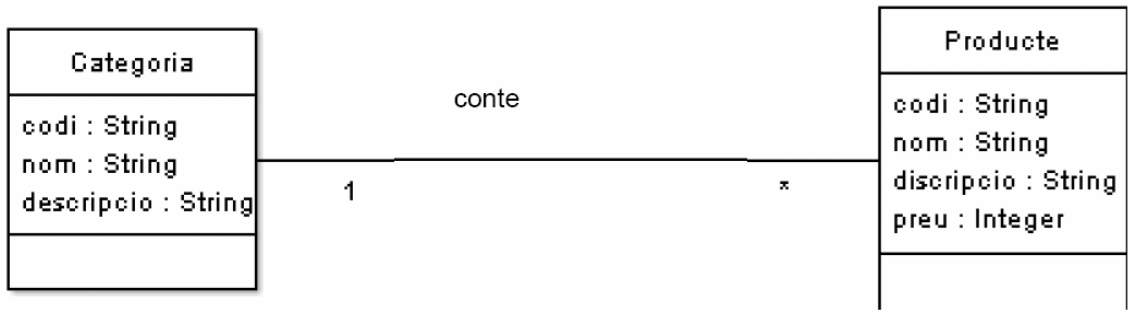


Fig. 7: El diagrama de classes de concepte mostra els conceptes de negoci que apareixen en l'aplicació i les seves relacions

Per fer ús de l'aplicació l'usuari s'haurà d'identificar i el sistema validarà si es tracta d'un usuari autoritzat o no per fer ús de l'aplicació. Una vegada autoritzat l'usuari podrà enregistrar un nou producte d'una determinada categoria, modificar les característiques del producte inclús esborrar-lo. Els productes es podran llistar per categoria, cercar per nom de producte i consultar-se. Les categories d'un producte també podran mantenir-se: fent altes, baixes, modificacions i consultes d'una categoria.

3.2.1 Anàlisi

Diagrama de Casos d'us

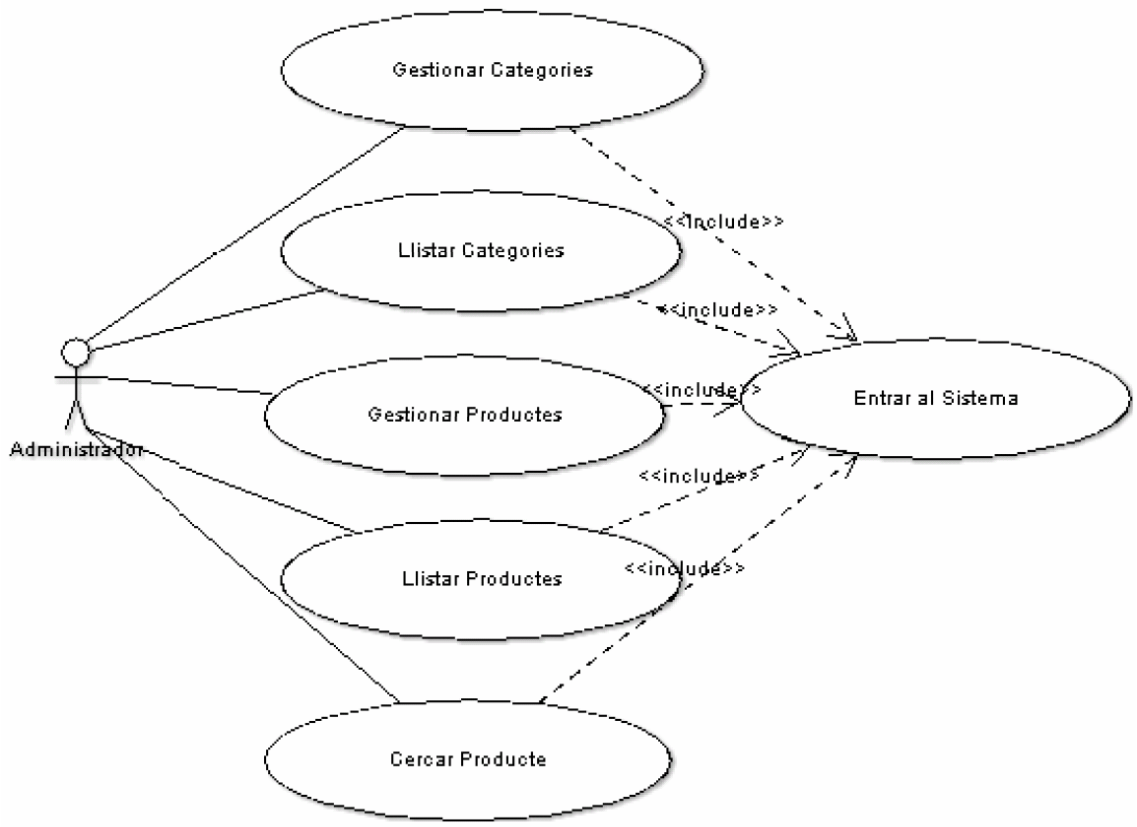


Fig.8: En el diagrama de casos d'us podem veure que fa el sistema i amb qui interactua. L'únic actor del sistema és l'administrador.

Actors:

- Tenim un sol actor que pot interactuar amb el sistema: l'administrador

Casos d'us:

- ***Entrar al sistema***: Permet als usuaris amb rol d'administrador identificar-se al sistema.
- ***Llistar Categories***: Permet als usuaris llista les categories que hi ha al catàleg de productes.
- ***Llistar Productes***: Permet als usuaris llistar tots els productes d'una categoria determinada.
- ***Cercar Productes***: Permet la cerca dels productes del catàleg a partir del nom.
- ***Gestionar Productes***: Permet ala creació, consulta, actualització i esborrat dels productes del catàleg.

Diagrama de Classes

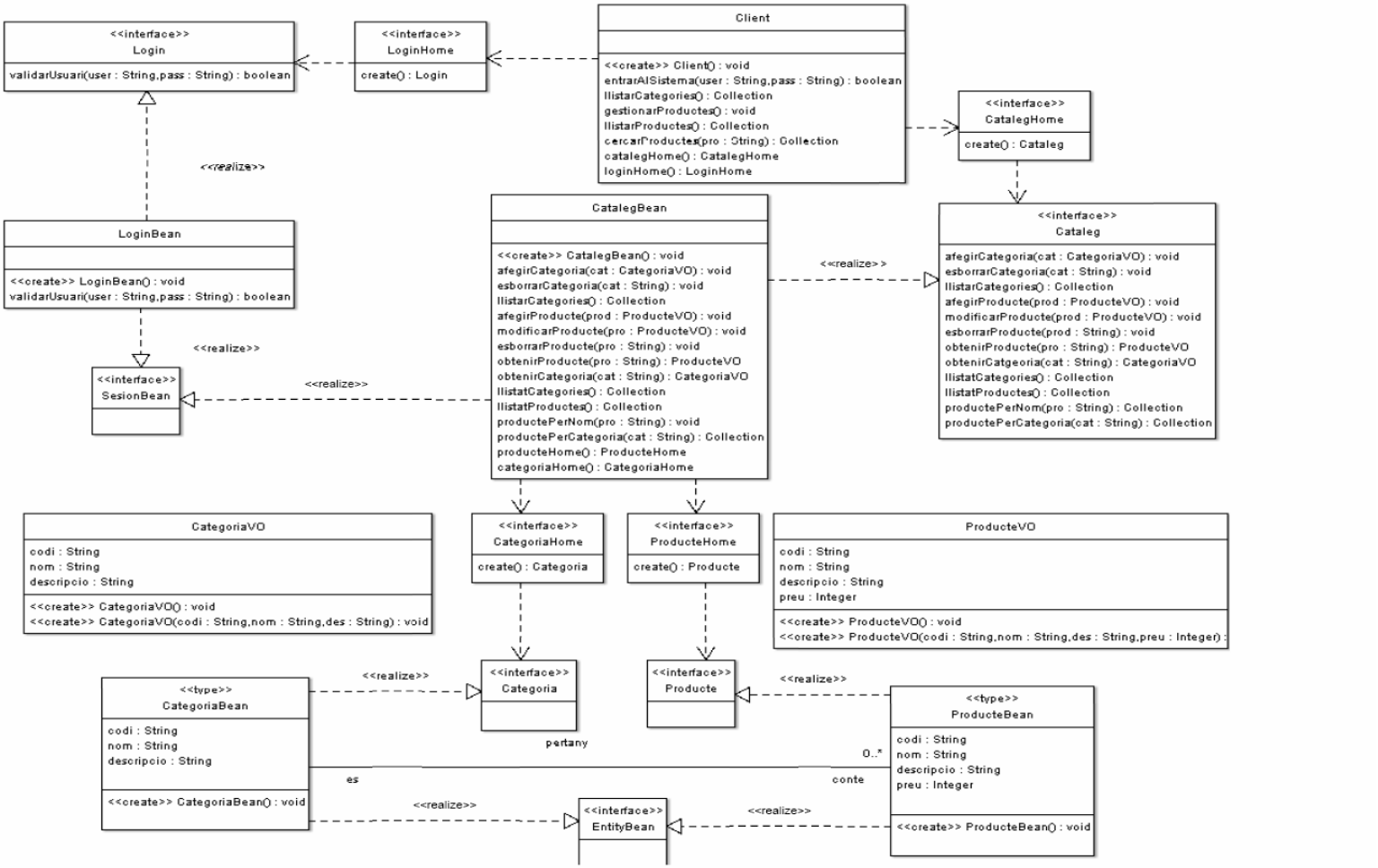


Fig. 9: Aquesta és un diagrama de classes on es pot veure les classes del model i les seves relacions. La classe Client realitza la validació de l'usuari amb la classe LoginBean a traves de la seva interfície. La lògica de negoci està suportada per la classe CatàlegBean. Aquesta classe al seu temps sol.licita els serveis de persistència a les classes ProducteBean i CategoriaBean.

3.2.2 Disseny i Arquitectura

Diagrama de Components

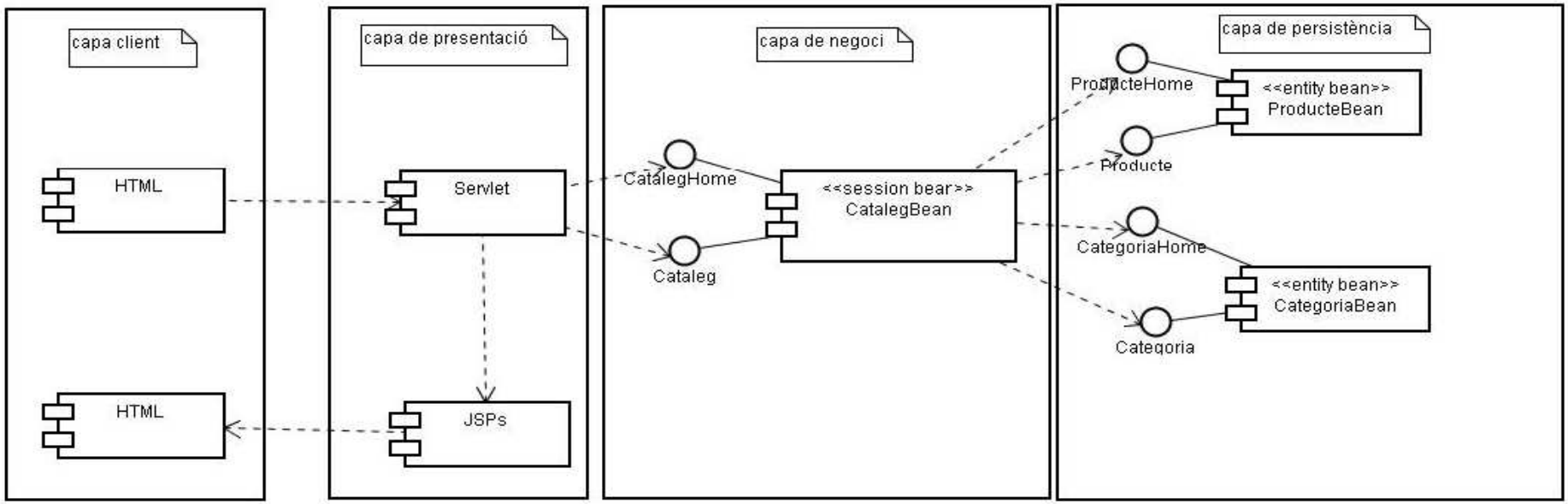


Fig. 10: Els clients interactuaran amb l'aplicació mitjançant planes html. Les peticions les adreçaran al servlet de la capa de presentació. Aquest servlet al seu temps sol.licitarà els serveis a la capa de negoci. El component CatalegBean serà l'encarregat d'atendre aquestes peticions que les resoldrà realitzant crides als components ProducteBean i CategoriaBean. Aquests components són els encarregats de gestionar la persistència i efectuaran accessos a la BBDD per resoldre el servei.

Diagrama de Desplegament

L'arquitectura de l'aplicació estarà dividida en 4 capes lògiques i 3 de físiques:

- Client (Maquina 1 amb un navegador)
- Presentació (Maquina 2 amb un servidor de pàgines web)
- Negoci (Maquina 2 amb un servidor d'aplicacions)
- Dades (Maquina 3 amb servidor de dades)

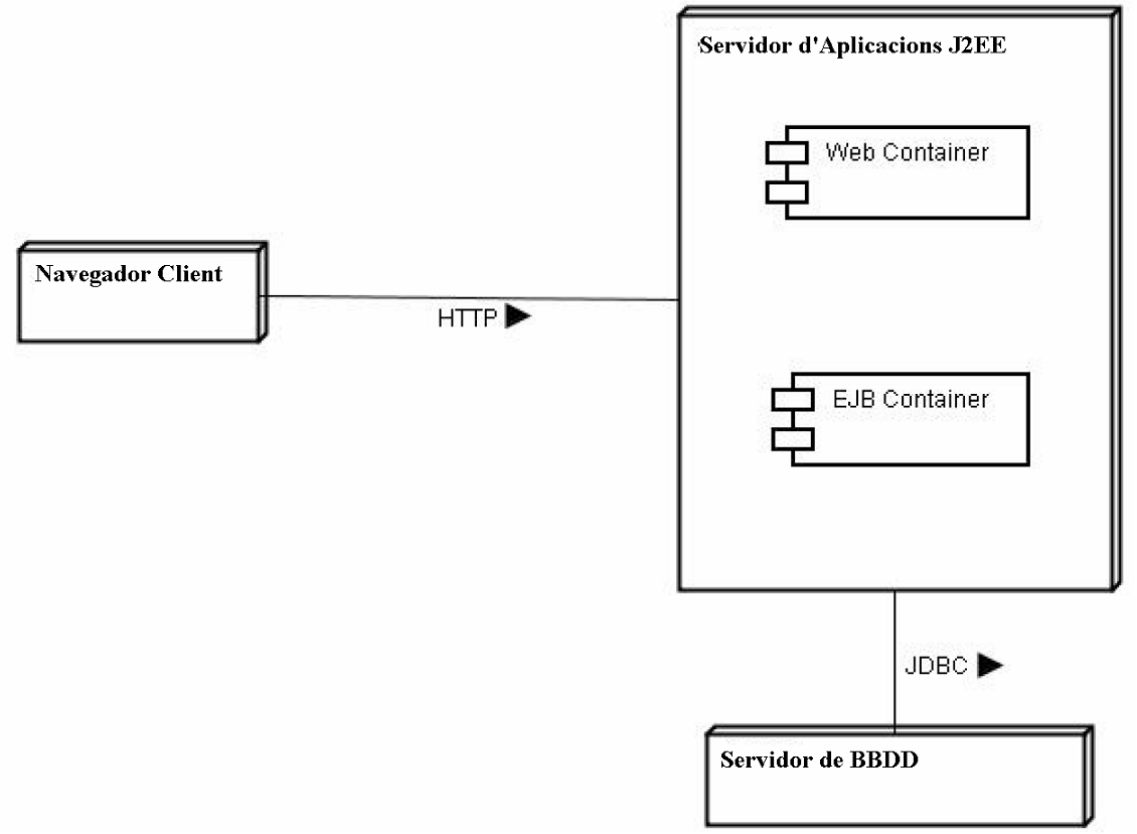


Fig. 11: Aquest diagrama és el diagrama de desplegament que mostra com se situen físicament els components. L'arquitectura distribuïda per components J2EE està dividida en dos contenidors, un realitza tasques de presentació (Web Container) i l'altre les tasques de negoci (EJB Container). El navegador realitza les peticions al servidor d'aplicacions i recupera les dades del servidor de dades

3.3 Procés d'avaluació de generadors.

Per poder realitzar l'avaluació de les eines generadores de codi s'ha seguit una sèrie de passos per tal d'obtenir el codi generat per les diferents eines a partir d'un mateix model.

La seqüència de passos és la següent

3.3.1 Eina de modelatge.
3.3.2 Disseny del model.
3.3.3 Exportació del model.
3.3.4 Generació de codi executable.
3.3.5 Anàlisi de les característiques dels generadors.

3.3.1 Eina de modelatge.

Inicialment s'ha triat una eina de modelatge per realitzar el model del sistema. Aquesta eina és la que permetrà especificar, emmagatzemar i finalment realitzar l'exportació del model en format XMI.

Els generadors de codi escollits tenen com característica comuna acceptar la versió 1.1 de XMI. Aquesta característica ha segut determinant a l'hora d'escollir l'eina de modelatge. Com a primera opció s'ha escollit Magic Draw per ser l'eina més completa, però finalment s'ha descartat per exportar els models en la versió 2.0 de XMI. Finalment s'ha escollit l'eina ArgoUML que genera documents XMI en versió 1.2, compatible amb les eines generadores de codi.

Bàsicament l'eina de modelatge ens servirà per dissenyar els models del sistema, concretament el diagrama de classes i el diagrama d'activitats.
Una vegada realitzats els diagrames es podran exportar ambdós per tal de generar els documents XMI corresponents.

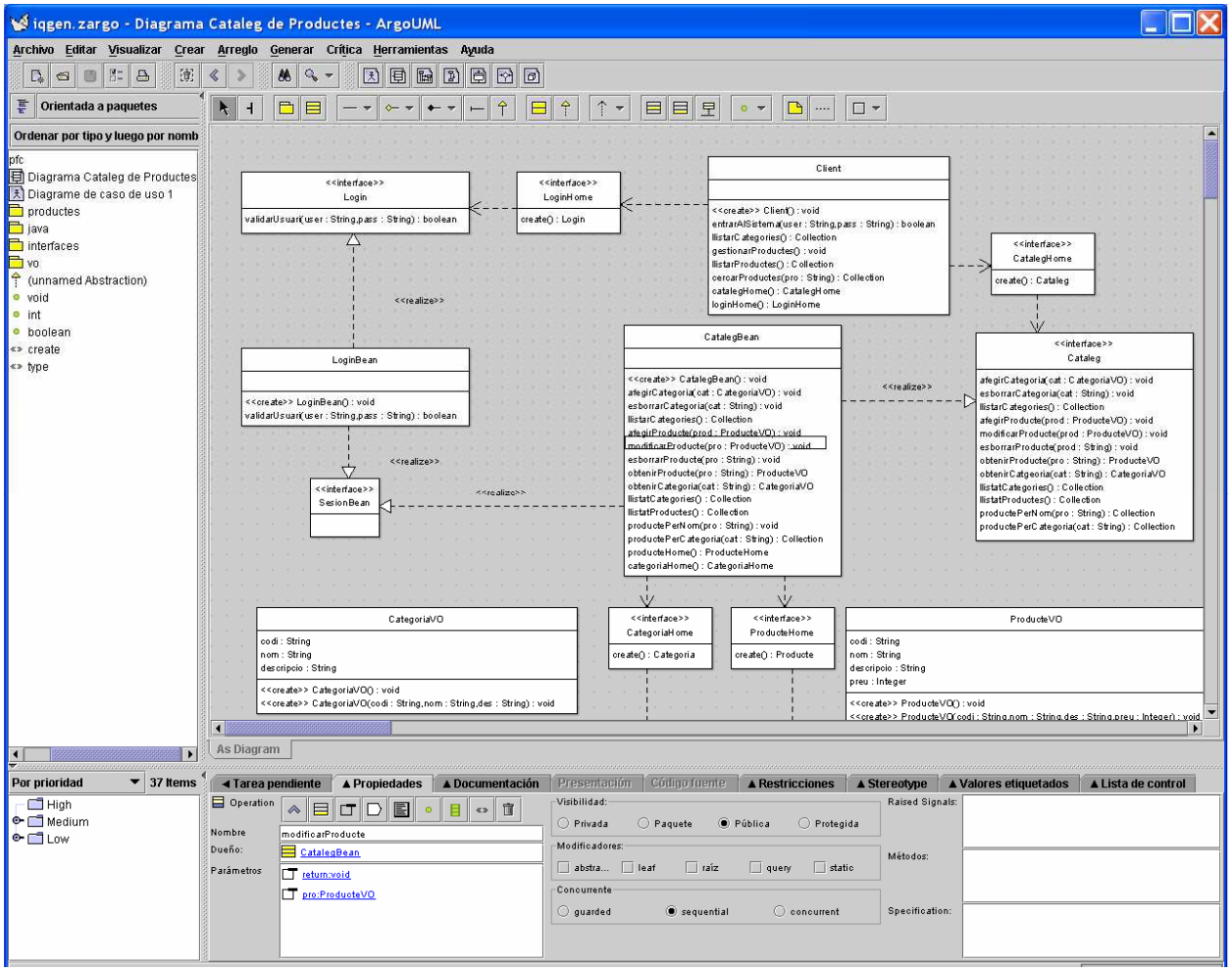


Fig. 12: Aquesta és la pantalla principal de l'aplicació ArqoUML. A la part superior es pot seleccionar els tipus de diagrames a visualitzar. La part central de la pantalla permet visualitza el diagrama de treball, en aquest cas un diagrama de classes.
La part inferior de la pantalla permet visualitzar i modificar les opcions de detall del diagrama. La part esquerra de l'aplicació és un navegador on es pot accedir i seleccionar per la seva edició qualssevol component del model.

3.3.2. Disseny del model.

S'ha dissenyat un model basat en un exemple proposat pel consultor. Aquest exemple s'ha reproduït amb l'eina ArgoUML tal com mostra la figura anterior.

El model representa una aplicació de gestió (alta, baixa, modificació i consulta) de productes d'una determinada categoria. L'arquitectura del sistema està distribuïda en 4 capes lògiques:

- La capa del client,
- La capa dels serveis de presentació
- La capa dels serveis de negoci
- La capa dels serveis de persistència de dades.

- **Diagrama de classes.**

Les principals classes del diagrama de classes són:

- Client: suport als serveis de presentació.
- LoginBean: suporta el component de seguretat de l'aplicació.
- CatalegBean: serveis de negoci
- ProducteVO: classe contenidora de la informació d'un producte.
- CategoriaVO: classe contenidora de la informació d'una categoria.
- ProducteBean: suport a la persistència de productes.
- CategoriaBean: suport a la persistència de categories.

Totes les classes tenen també les seves interfícies, a més a més les classes que donen suport a un "Bean" o component J2EE tenen també les interfícies relacionades amb la creació, localització i accés, pròpies dels servidors d'aplicacions J2EE.

Degut a les diferencies entre el codi generat pels dos generadors ens veurem obligats a crear dos models diferents del mateix sistema. Les diferencies es deuen a què el generador Axgen genera automàticament interfícies per cada classe del model: això ens obliga crear un nou model sense les interfícies del diagrama de classes.

- **Diagrama d'activitat.**

També s'ha realitzat un diagrama d'activitat que representa el flux de treball per identificar-se al sistema i poder accedir a les funcionalitats de l'aplicació. Les principals activitats són:

- Petició d'identificació de l'usuari.
- Introducció de dades.
- Validació de l'usuari pel sistema.
- Avis d'usuari no autoritzat.
- Autorització de l'usuari.

3.3.3 Exportació del model.

Una vegada definides les classes i les interfícies s'ha realitzat una exportació del model en format XMI (versió 1.2) amb una versió de UML 1.4.

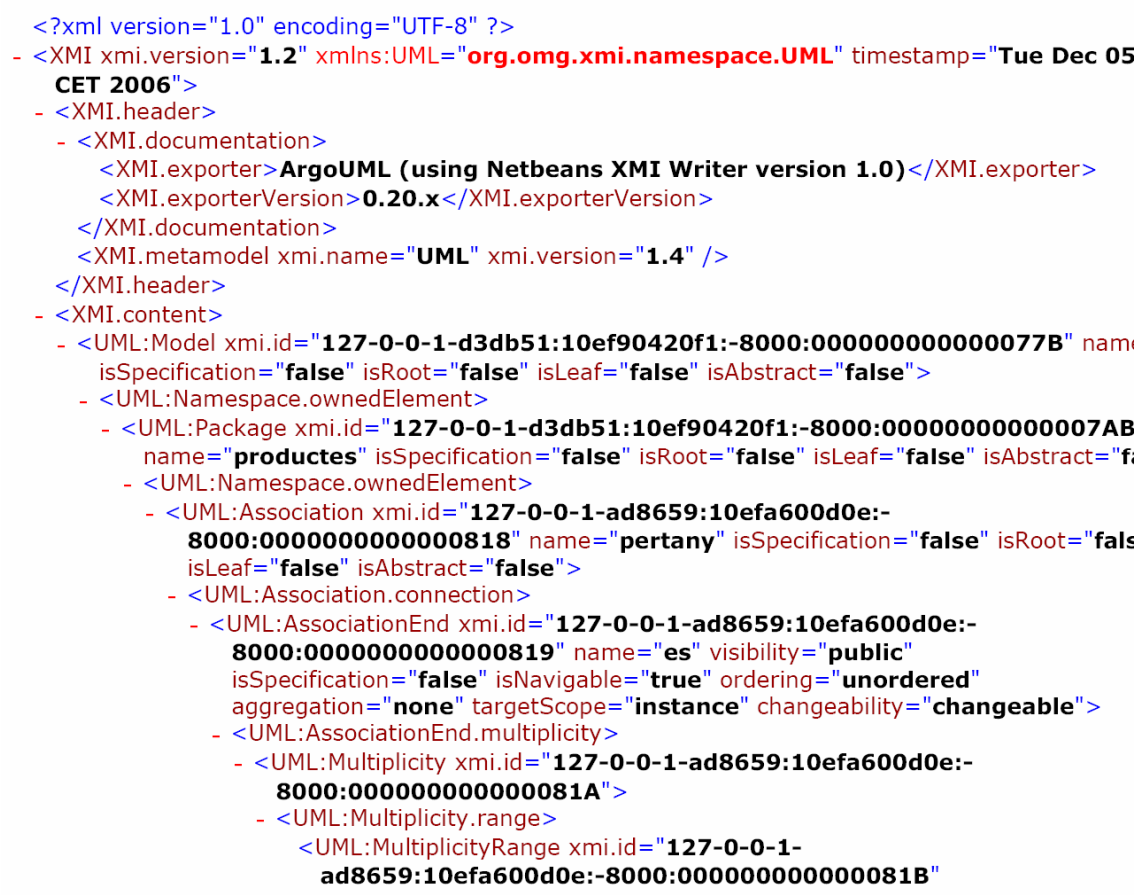


Fig. 13: Aquest fitxer XMI ha estat generat a partir del diagrama de classes. A la capçalera del fitxer es pot veure que utilitza la versió 1.2 de XMI i la versió 1.4 de UML. A continuació descriu el contingut del model i la seva composició. Podem veure con especifica primer el model anomenat pfc i a continuació el paquet producte i les seves parts.

A conseqüència dels diferents models suportats per cada eina, cadascun dels models generarà un document XMI diferent per cadascun dels generadors de codi.

A més a més hi haurà un altre document XMI amb l'exportació del diagrama d'activitats. Aquests documents XMI són la representació del model del sistema interpretable per un programa: la transformació d'aquest model donarà lloc al codi executable per una determinada plataforma.

3.3.4 Generació del codi.

Les diferents eines generadores de codi llegeixen el document XMI que representa el model del sistema i a partir de la transformació que tinguin definida en les seves plantilles converteixen el model amb codi executable.

```
 * (#)ProducteBean.java
 * Stereotype: type
 *
 * Generated with iQgen template /class.jsp

 */
package com.innoq.tutorial_1.productes;

//==> Begin Protected Area imports
// ...add additional imports here
//==> End Protected Area imports

/**
 * Class ProducteBean
 *
 *
 * Last generation: Sat Dec 02 18:24:20 CET 2006
 */
public class ProducteBean {


    //==> Begin Protected Area global
    // ...add your own attributes and methods here
    //==> End Protected Area global

    // Membervariable of attribute codi
    private String codi;

    // Membervariable of attribute nom
    private String nom;

    // Membervariable of attribute descripcio
    private String descripcio;

    // Membervariable of attribute preu
    private Integer preu;
```

Fig. 14: Aquest fitxer correspon al codi generat per l'aplicació iQgen a partir de la classe ProducteBean. Com es pot observar el generador reserva uns espais protegits per afegir el codi d'usuari. Aquest codi es conserva entre generació i generació. Això facilita la integració del codi generat en un procés de desenvolupament de l'aplicació.

Els dos generadors escollits generen codi ben diferent fins al punt de tenir que partir de models diferents per representar el mateix sistema. El codi està organitzat per paquets, s'han creat tres paquets: un paquet per contenir totes les interfícies, un altre paquet que conte els Value

Objects (ProducteVO.java i CategoriaVO.java) i finalment un paquet que sota el nom de producte conté la resta de classes.

3.3.5. Anàlisis i dels generadors.

Les diferents eines generadores de codi llegeixen el document XMI i generen les classes corresponents al model. El codi generat per les dues eines és ben diferent. A continuació realitzarem una anàlisi detallada del procés de generació de codi per les dues eines.

3.4 Avaluació dels generadors.

3.4.1 iQgen

Disseny del model

Amb l'eina ArgoUML s'han realitzat els diferents models corresponents al:

- Diagrama de classes del prototip
- Diagrama de classes de l'aplicació
- Diagrama d'activitat de l'aplicació

- **Diagrama de classes del Prototip**

S'ha realitzat el model de la pàgina anterior que correspon a un prototip on es mostra tota la casuística d'elements que poden aparèixer en un diagrama de classes i les seves relacions.

- **Diagrama de classes de l'Aplicació**

El model que es mostra en la següent pàgina correspon a una aplicació de gestió de productes d'una determinada categoria. Seguin l'arquitectura distribuïda per components tenim que les diferents classes e interfícies es poden agrupar en quatre capes:

1. La capa del client que no està representada en el diagrama de classes.
2. La capa de presentació que correspon a la classe Client, aquesta és la que realitza les peticions a la capa de negoci, rep les respostes i les envia a l'usuari.
3. La capa de Negoci que correspon a la resta de classes on podem destacar la Classe CatalegBean (una Sesion Bean) que concentra la lògica de negoci
4. La capa de persistència està representada per les classes CategoriesBean i ProductesBean que són components tipus Entity Bean i gestionen l'accés a la BBDD.

- **Diagrama d'activitat de l'Aplicació**

S'ha realitzat del diagrama d'activitat que es mostra a continuació del de classes. Aquest diagrama d'activitat representa el flux de treball entre les pantalles d'una connexió a l'aplicació (login).

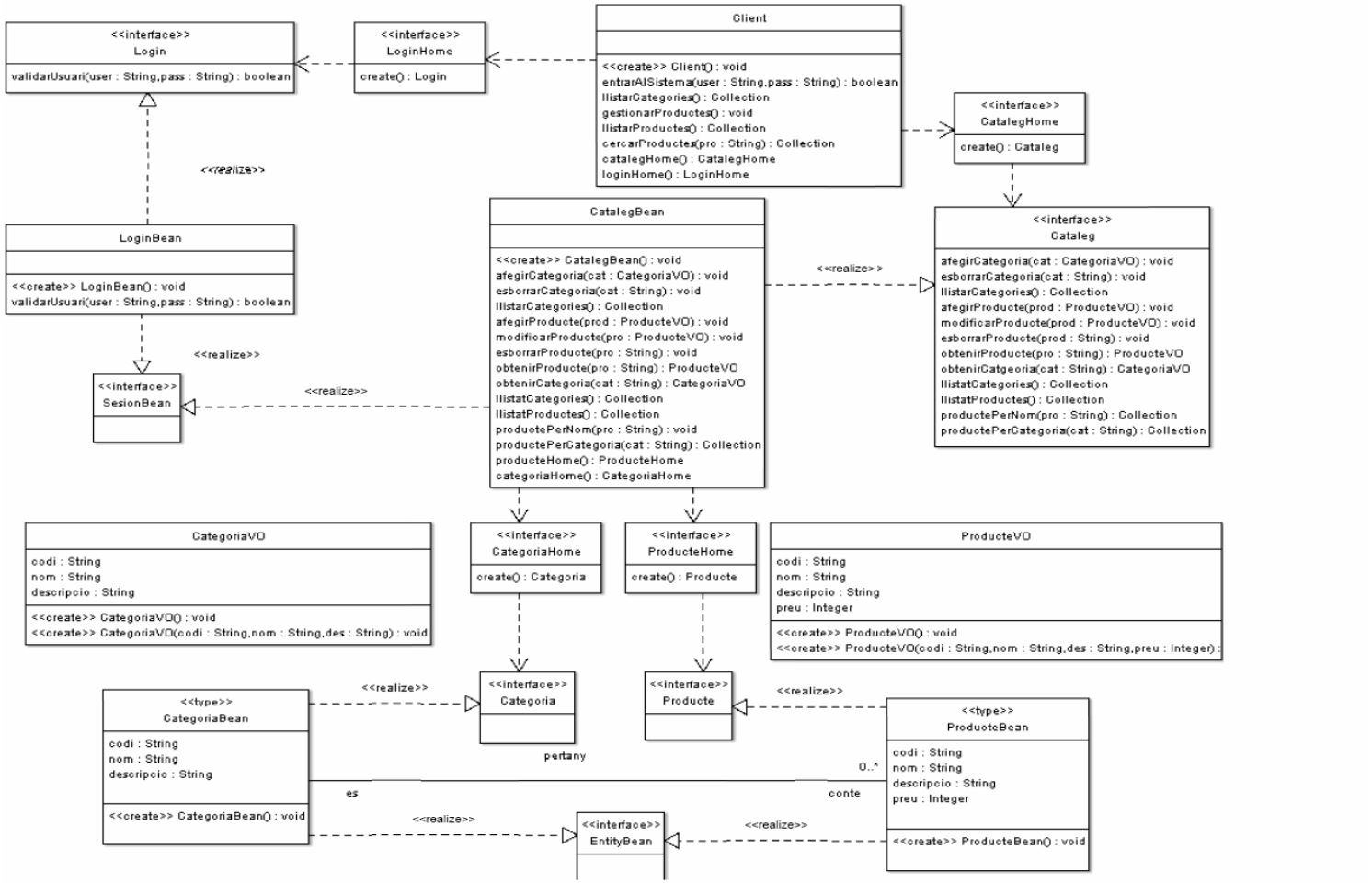


Fig. 15: Aquest diagrama de classes representa el model de l'aplicació del catàleg de productes. La classe principal CatalegBean és un component tipus Sessió que suporta la lògica de negoci i atén les peticions realitzades pel client El client abans de poder accedir als serveis de l'aplicació s'haurà de validar. Les classes CatalegBean i ProducteBean són components de tipus Entity i gestionen la persistència d'aquestes classes.

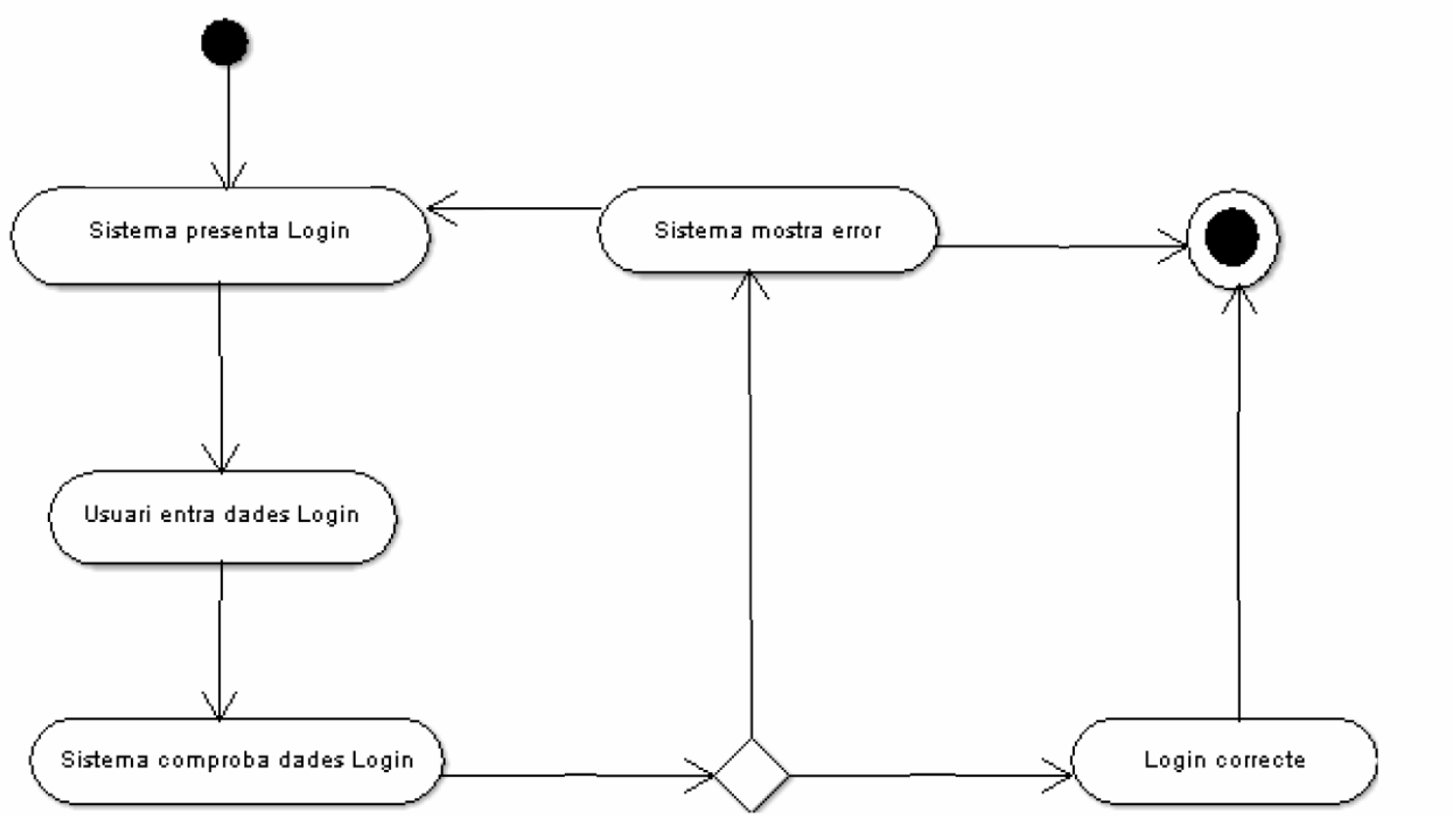


Fig. 16: Aquest diagrama d'activat representa els diferents estat pels que passa una validació d'un usuari que vol connectar-se a l'aplicació. En primer lloc el sistema mostra la pantalla d'entrada de les dades. A continuació l'usuari introdueix el nom i password per que el sistema el validi. En funció del resultat de validació l'usuari podrà connectar-se a l'aplicació o el sistema mostrarà un missatge d'error i tornarà a preguntar l'usuari i el password.

Exportació del model

L'eina ArgoUML ha realitzat l'exportació del model amb tots tres casos.

- **Diagrama de classes del Prototip**

El document XMI generat està disponible al fitxer: "iQgen XMI prototip.zip"

- **Diagrama de classes de l'Aplicació**

El document XMI generat està disponible al fitxer: "iQgen XMI classes.zip"

- **Diagrama d'activitat de l'Aplicació**

El document XMI generat està disponible al fitxer: "iQgen XMI activitat.zip"

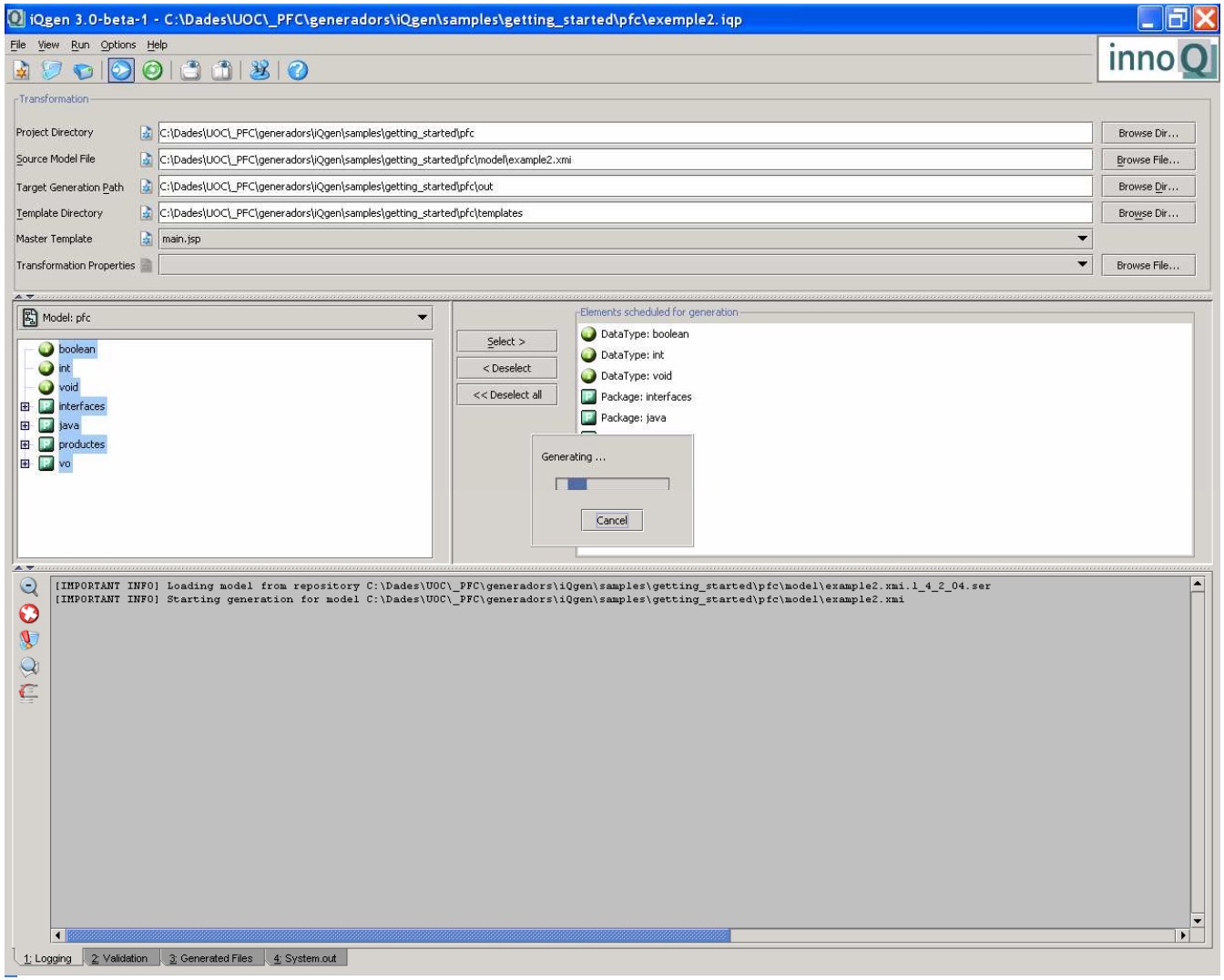


Fig. 17: Aquesta és la pantalla principal de l'aplicació iQgen. A la part superior es pot indicar la ubicació dels recursos com per exemple: el directori del projecte, el nom del XMI, el directori de sortida, etc. La part central de la pantalla permet seleccionar les parts del model per les que es generarà el codi. La part inferior de la pantalla permet visualitzar les diferents sortides de la generació: traçat, errors de validació, fitxer generats, etc.

Generació del codi

- **Diagrama de classes del Prototip**

Una vegada exportat el model a un document XMI es procedeix a configurar el generador de codi per tal que generi codi a partir d'aquest document. L'eina iQgen permet mantenir diferents generacions en forma de projectes. Per això creem un nou projecte iQgen i el configurem per què generi codi a partir del model prototip representat pel document XMI: ModelClasses.xmi.
El codi generat per l'eina iQgen està disponible al document "iQgen codi prototip.zip"
El codi sql generat per l'eina iQgen està disponible al document "iQgen sql prototip.rar"

- **Diagrama de classes de l'Aplicació**

S'exporta el model a un document XMI es procedeix a crear un nou projecte iQgen i el configurem per què generi codi a partir del model del sistema representat pel document XMI: iQgen.xmi.
El codi generat per l'eina iQgen està disponible al document "iQgen codi classes.zip"
El codi sql generat per l'eina iQgen està disponible al document "iQgen sql classes.zip"

IQgen genera codi tant per les classes com per les interfícies representades en el model. Pels atributs de les classes genera els seus mètodes d'accés (getters i setters) i per les operacions tant de les interfícies i les classes genera la signatura (paràmetres i tipus) tal i com està representada al model.
Els mètodes generats contenen una zona delimitada per marques d'inici i fi on es pot incloure codi d'usuari que es conserva (àrea protegida) entre una generació de codi i l'altra.
Manipulant mínimament les plantilles de generació es poden generar les sentencies SQL per esborrar i crear les taules corresponents a aquelles entitats declarades amb l'estereotip de persistents.

- **Diagrama d'activitat de l'Aplicació**

El generador iQgen és capaç de generar codi a partir d'un diagrama d'activitat. Concretament a partir d'aquest diagrama d'activitat que representa el flux de treball entre les pantalles d'una connexió a l'aplicació (login), l'eina genera les pàgines .html amb els enllaços de forma que es pot navegar entre les pantalles tal i com mostra el diagrama de la pàgina següent.

El codi generat per l'eina iQgen està disponible al document "iQgen Pantalles.zip":

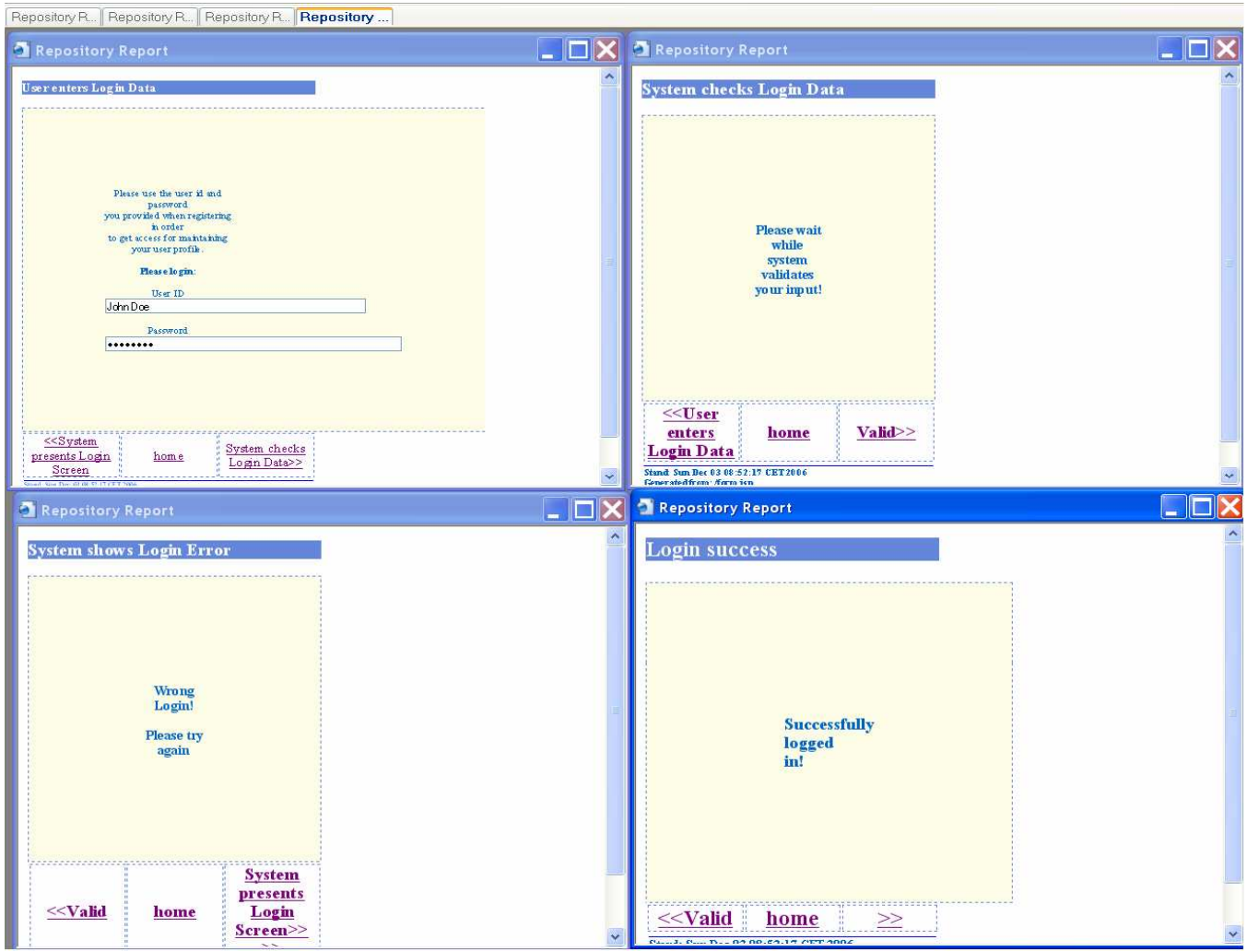


Fig. 18: Aquestes pantalles corresponen a les pàgines .html generades per iQgen a partir del diagrama d'activitats. Són respectivament les pantalles d'introducció de l'usuari i password, la pantalla corresponen a la validació del sistema, la pantalla d'usuari incorrecte i finalment la de connexió satisfactòria.

Característiques

- **Generals**

iQgen és un generador que ofereix una bona usabilitat i unes prestacions suficients ja que dona suport bàsic a la generació de codi d'interfícies, classes, mètodes i atributs així com diagrames d'activitat i suport a la persistència. Les proves realitzades confirmen les prestacions previstes amb uns resultats acords.

Característiques	Valors previstos	Valors provats
Característiques de l'Entrada (Model del Sistema)		
Eines de modelatge UML suportades	El generador és capaç d'interpretar i generar codi a partir del XMI generat per les següents eines de modelatge: Rational Rose, MID Innovator, Microtool ObjectiF, Together,ObjectDomain i ArgoUML	Tan sol s'ha provat amb l'eina ArgoUML
Exportador del XMI	L'eina de modelatge exporta el document XMI. El generador de codi també por exportar el XMI a traves de les llibreries de Novosoft	Tan sols s'ha provat l'exportació del XMI a partir de ArgoUML
Versió UML	1.0,1.1	1.4
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies	El generador genera codi per les Classes i les Interfícies
UML Dinàmic	El generador és capaç de generar les classes per un diagrama d'Activitat. En l'exemple de l'aplicació a partir d'un diagrama d'activitat genera el codi .html corresponent a la navegació entre les diferents pantalles de l'aplicació.	S'ha provat un exemple on a partir d'un diagrama d'activitat l'aplicació genera una sèrie de pantalles .html que permeten la navegació entre elles
Restriccions OCL	No les considera	
Versions XMI suportades	1.0,1.1,1.2,1.3	1.2
Característiques de la Sortida (Generació de codi)		
Mètode de Transformació	L'eina utilitza Plantilles JavaScript per transformar el codi XMI en codi executable	L'eina utilitza Plantilles JavaScript per transformar el codi XMI en codi executable
Transformació configurable	L'eina permet la modificació de les plantilles i la creació de noves per adaptar la transformació a les pròpies necessitats	S'ha modificat les plantilles per adequar la generació de codi SQL a les característiques de ArgoUML.
Permet Codi d'usuari	Disposa d'àrees protegides per incorporar codi d'usuari. Aquest codi es manté entre una generació i una altra.	Disposa d'àrees protegides per incorporar codi d'usuari. Aquest codi es manté entre una generació i una altra.
Llenguatges generats	Java (J2EE), C++, C# (.NET), C.,XML(per descriptors de desplegament, ant,..)	Java
Suport Persistència	Dóna suport a la persistència generant sentencies SQL	Genera codi SQL per l'esborrat i creació de les entitats declarades com persistents.

- **Transformació del model**

Classes	Visibilitat	Públic	Privat	Protegit
	Atributs	SI, transformació correcta	NO, els “geters” i “setter” són públics	NO, els “geters” i “setter” són públics
	Operacions	SI, transformació correcta	SI, transformació correcta	SI, transformació correcta

Classes	Tipus	
	Abstracta	NO ho ha tingut en compte

Relacions	Tipus	
	Associació	NO ho ha tingut en compte
	Composició	NO ho ha tingut en compte
	Agregació	NO ho ha tingut en compte
	Herència	SI

3.4.2 Axgen

Disseny del model

Amb l'eina ArgoUML s'han realitzat els diferents models corresponents al:

- Diagrama de classes del prototip
- Diagrama de classes de l'aplicació

- **Diagrama de classes del Prototip**

S'ha utilitzat el mateix model que amb l'eina iQgen que correspon a un prototip on es mostren tota la casuística d'elements que poden aparèixer en un diagrama de classes i les seves relacions.

- **Diagrama de classes de l'Aplicació**

En aquest model s'ha prescindit d'incloure les interfícies ja que el generador Axgen genera automàticament les interfícies per cadascuna de les classes.
Altra particularitat (o defecte) és que no reconeix els paràmetres ni els seus tipus en la signatura de les operacions i tampoc els tipus del resultat.

Tenim la classe client que representarà la capa de presentació de l'aplicació i que és la que realitzarà les peticions a la classe de negoci.
La capa de negoci la formen la resta de classes com CatalegBean (Bean de sessió) que conte la lògica de negoci i les classes CategoriaBean (Bean d'Entitat) que realitzen les peticions d'accés a BBDD. La informació viatja entre les classes a traves de les classes (patró Value Object) CategoriaVO i ProducteVO.

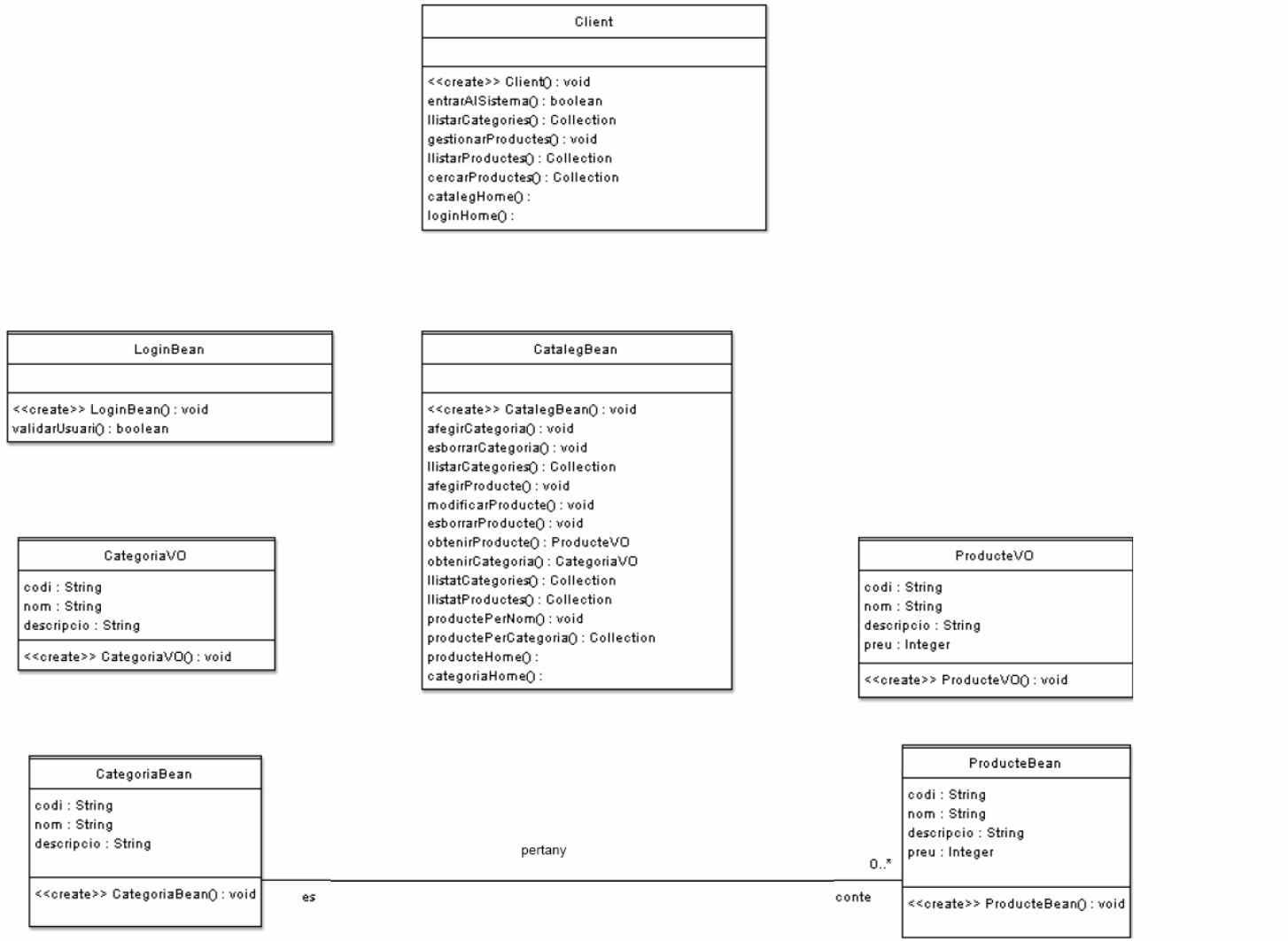


Fig. 19: Aquest diagrama de classes és el mateix que el realitzat pel generador iQgen però sense interfícies. Una característica del generador Axgen és que genera automàticament les interfícies per totes les classes del model. Per cada classe del model es generen cuatre classes java: una interfícies, una classe d'implementació, una classe 'AbstractAddOn' i una classe 'AddOn'.

Exportació del model

L'eina ArgoUML ha realitzat l'exportació del model ambdós casos.

- **Diagrama de classes del Prototip**

El document XMI generat per l'eina està disponible al document "Axgen prototip.zip"

- **Diagrama de classes de l'Aplicació**

El document XMI generat per l'eina està disponible al document "Axgen classes.zip"

Generació del codi

Una vegada exportat el model a un document XMI es procedeix a configurar el generador de codi per tal generi codi a partir d'aquest document. L'eina Axgen no disposa d'interfície d'usuari però permet configurar els projectes a traves del fitxer axgen.propieties i el fitxer buid.xml on s'especifiquen les tasques que ha de realitzar l'eina ant.

- **Diagrama de classes del Prototip**

El codi generat per l'eina Axgen està disponible al document "Axgen codi prototip.zip"
El codi SQL generat per l'eina Axgen està disponible al document "Axgen sql prototip.zip"

- **Diagrama de classes de l'Aplicació**

El codi generat per l'eina Axgen està disponible al document "Axgen codi classes.zip"
El codi SQL generat per l'eina Axgen està disponible al document "Axgen sql classes.zip"

L'eina genera varies classes i una interfície per cada classe del model. Si es declaren interfícies en el model, dona problemes a l'hora de compilar el codi generat. Pels atributs de les classes genera els seus mètodes d'accés (getters i setters). Les operacions (sense paràmetres ni retorn) es generen automàticament com mètodes a les classes i a les interfícies.
La persistència es genera automàticament per cada classe declarada en el model, malauradament es desconeix la forma d'activar/desactivar la persistència en una classe, això vol dir que es generen taules per totes les taules.
A més de generar codi l'eina Axgen també compila les classes i les empaqueta en fitxers .jar.

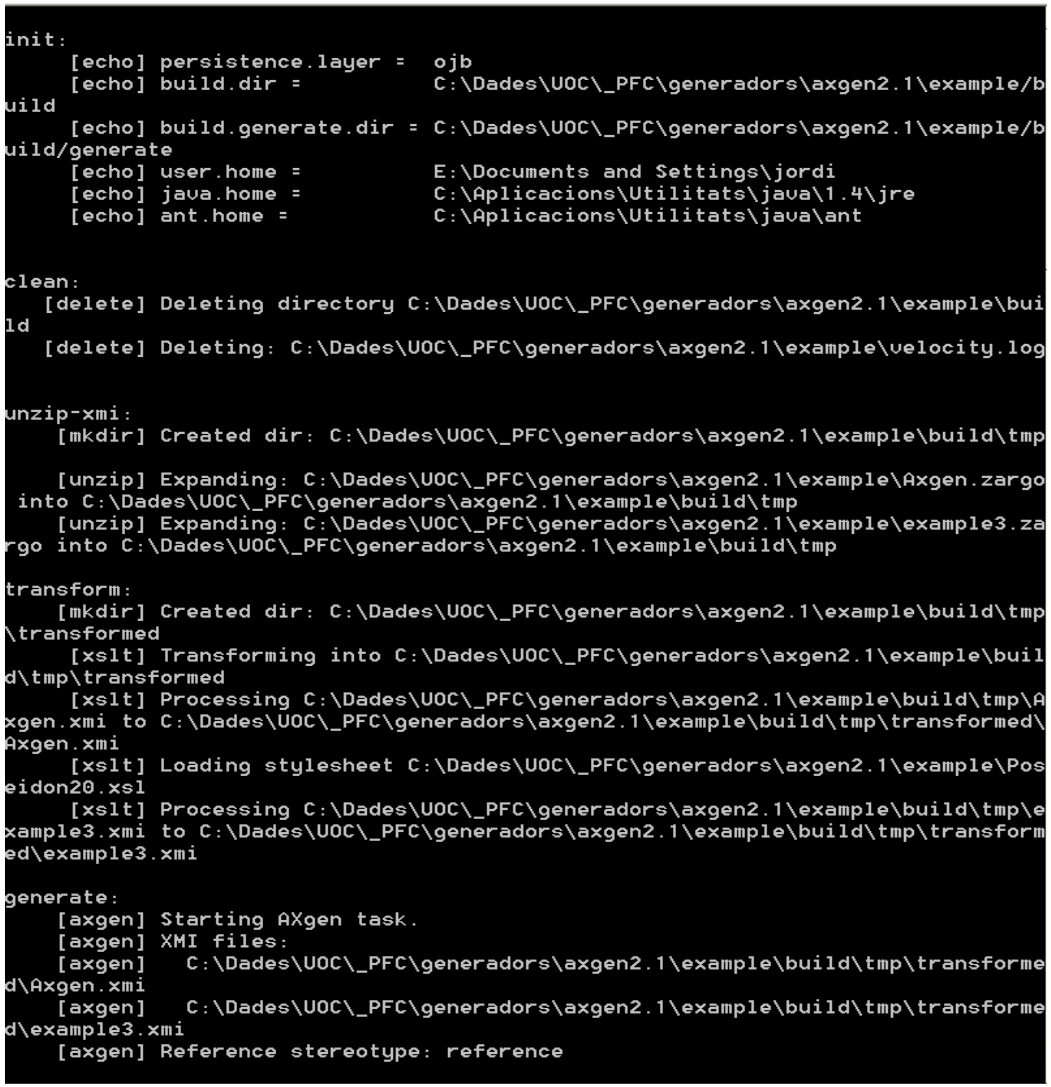


Fig. 20: Aquesta és la pantalla de l'aplicació Axgen, com es pot comprovar no té cap interfície d'usuari. L'aplicació es configurarà a traves de fitxers buil.xml corresponent a tasques ant. La transformació es realitza executant l'eina ant.

Característiques

- **Generals**

Axgen és un generador d'ús senzill però flexible. Té particularitats amb la generació de codi així com alguna limitació com no acceptar paràmetres en els mètodes o no donar suport als diagrames d'activitat. Les proves realitzades confirmen algunes de les característiques previstes.

Característiques	Valors previstos	Valors provats
Característiques de l'Entrada (Model del Sistema)		
Eines de modelatge UML suportades	Teòricament dóna suport a totes les eines ja que la mateixa eina crea el XMI a partir del repositori de l'eina de modelatge a partir de la seva API i altres llibreries incorporades (NetBeans MDR i NsUml)	Tan sol s'ha provat amb l'eina ArgoUML
Exportador del XMI	L'eina de modelatge exporta el document XMI. També por exportar el XMI a traves de les llibreries pròpies i d'altres, com: NetBean MDR o de NsUml.	Tan sols s'ha provat l'exportació del XMI a partir de ArgoUML
Versió UML	1.4.	1.4
UML Estàtic	El generador és capaç de generar codi per les Classes i les Interfícies	El generador genera les interfícies automàticament per cada classe del model. El fet que el model contingui interfícies dona problemes.
UML Dinàmic	No	No
Restriccions OCL	No les considera	
Versions XMI suportades	1.1.	1.2
Característiques de la Sortida (Generació de codi)		
Mètode de Transformació	Plantilles Velocity són plantilles de substitució on es defineix el codi que es generarà.	Plantilles Velocity
Transformació configurable	Aquestes plantilles poden modificar-se per adaptar el codi generat a les nostres necessitats.	No s'ha provat de modificar les plantilles
Permet Codi d'usuari	Sí, poden incorporar-se a les plantilles crides a procediments propis.	No s'ha provat
Llenguatges generats	Java (J2EE)	Java
Suport Persistència	Genera codi per eines de mapeig objecte/relacional com OJB (Object/Relational Bridge).	Genera les sentències SQL per crear les taules a partir de les classes del model

- **Transformació del model**

Classes	Visibilitat	Públic	Privat	Protegit
	Atributs	SI, transformació correcta	NO, els “geters” i “setter” són públics	NO, els “geters” i “setter” són públics
	Operacions	SI, transformació correcta	NO, els “geters” i “setter” són públics	NO, els “geters” i “setter” són públics

Classes	Tipus	
	Abstracta	NO ho ha tingut en compte

Relacions	Tipus	
	Associació	NO ho ha tingut en compte
	Composició	NO ho ha tingut en compte
	Agregació	NO ho ha tingut en compte
	Herència	SI, transformació correcta

3.5 Proposta de millora

Una millora a introduir en els generadors de codi és la transformació de les restriccions expressades en el model amb el llenguatge OCL en codi executable.

El OCL (Object Constraint Language) és un llenguatge formal usat per descriure expressions en models UML. Aquestes expressions serveixen per representar restriccions respecte a les classes o relacions.

Les restriccions que podem trobar en un diagrama són les següents:

- Restriccions a nivell de classificadors
    - **Definicions**: expressió que serveix per permetre la reutilització de variables o operacions en múltiples expressions.
    - **Invariants**: expressió que ha de ser certa per totes les instàncies d'una classe en qualssevol moments.
- Restriccions a nivell d'operacions d'un classificador
    - **PreCondicions**: expressió que ha d'ésser certa abans que comenci l'operació.
    - **Cos de l'Operació**: expressió per indicar el resultat d'una operació de consulta.
    - **PostCondicions**: expressió que ha d'ésser certa al final de l'operació.
- Restriccions a nivell d'atributs o el Final d'una Associació
    - **Valors Inicials**: expressió que s'utilitza per indicar un valor inicial d'un atribut o el valor inicial d'un Final d'una associació.
    - **Valors Derivats**: expressió que s'utilitza per indicar un valor derivat d'un atribut o el valor derivat d'un Final d'una associació.

3.5.1 Transformació de definicions

Una possible implementació de les definicions en Java seria declarant atribut privat (o protegit) de la classe i un mètode privat (o protegit) de la classe per cadascuna de les definicions. Aquests mètodes tindrien els mateixos paràmetres i resultats que les definicions i realitzarien les mateixes operacions. Els resultats d'aquests mètodes s'assignarien mitjançant un mètode "setter" i es consultarien amb el mètode "getter" corresponent per obtenir el seu valor des d'altres llocs de la classe o del paquet.

Exemple OCL:

```
context Persona
def: ingressos : Integer = self.treball.sou->sum();
```

Transformació Java:

```
Int ingressos;

private int sumIngressos() {
      Iterator it = new Iterator();
      Int ingress = 0;
      while (it.hasNext()) {
           ingress=+this.treball.sou();
      }
      return ingress;
}
```

```
public int getIngressos() {
      return ingressos;
}

public setIngressos(int valor) {
      ingresos=valor;
}
.....
setIngressos(sumIngressos());
```

3.5.2 Transformació d'Invariants

Un invariant ha de complir les condicions que expressa per totes les instàncies de la classe. Una manera de comprovar això podria ser validant aquesta expressió cada vegada que pot modificar-se la classe: al moment de crear la classe, al moment de modificar alguns dels seus atributs o al moment de realitzar alguna de les operacions de la classe. Aquesta comprovació es realitzaria invocant a un mètode auxiliar que llances una excepció si no es compleix l'invariant. Per qüestions d'eficiència, aquesta comprovació tan sols s'hauria de realitzar en aquells llocs on alguns dels elements que formen l'expressió poden variar, per això mateix aquest invariant s'haurien de comprovar al final de les operacions implicades. La implementació d'aquesta restricció podria realitzar-se mitjançant un mètode privat o protegit on es tradueixen les expressions en codi executable.

Exemple OCL:

```
context Companyia inv: self.nombreEmpleats()>50;
```

Transformació Java:

```
private boolean Invariant(){
      return this.nombreEmpleats()>50;
}
.......

public afegirEmpleat(Empleat emp) throws InvariantException{
      if not Invariant() thow InvariantException;
.......
}
```

3.5.3 Transformació de Precondicions

Una precondició és una restricció que ha de ser certa abans que comenci una operació. Per tant el lloc lògic on realitzar aquesta validació és abans de realitzar l'operació afectada. La implementació d'aquesta restricció podria realitzar-se mitjançant un mètode privat o protegit on es tradueixen les expressions en codi executable. A l'invocar aquest mètode, si no es compleix la restricció es llança una excepció.

Exemple OCL:

```
context Classe::nomOperacio(param1 : Integer,...) : TipusRetorn
pre: param1 > 10;
```

Transformació Java:

```
private boolean preOperació(integer param1) {
      return param1>10
}
....

public nomOperacio(int param1) throws PreconditionException {
      if not preOperació(param1) thow PreconditionException;
.....
}
```

3.5.4 Transformació del Cos de l'Operació

Un cos d'Operació és una restricció que expressa el resultat d'una operació de consulta. Per tant el lloc lògic on realitzar aquesta operació és després de realitzar la precondició. La implementació d'aquesta restricció podria realitzar-se mitjançant un mètode privat o protegit on es tradueixen les expressions en codi executable.

Exemple OCL:

```
context Persona::esposaActual() : Persona
pre: self.estaCasada=cert;
body: self.bodes->select(b|b.acabades=fals).esposa
```

Transformació Java:

```
public Collection esposes(Persona per) {
       Collection llista = new Collection();
       Iterator it = llista.iterator();
       while (it.hasNext()) {
              if (!per.bodes.acabades())
                     llista.add(persona.bodes);
       }
       return llista;
}

public Persona esposaActual() {
       return (Persona) (esposes(this).next());
}
```

3.5.5 Transformació de PostCondicions

Una postcondició és una restricció que ha de ser certa després que acabi una operació. Per tant el lloc lògic on realitzar aquesta validació és després de realitzar l'operació afectada. La implementació d'aquesta restricció podria realitzar-se mitjançant un mètode privat o protegit on es tradueixen les expressions en codi executable. A l'invocar aquest mètode, si no es compleix la restricció es llança una excepció. Moltes vegades la pròpia operació pot expressar-se en forma de postcondició.

Exemple OCL:

```
context Persona::ingresos(d : Data) : Integer
post: result=5000;
```

Transformació Java:

```
private boolean postIngressos(int ingress) {
       return ingress=5000;
}
....

public ingressos(Date d) throws PostconditionException {
       if not postIngressos(5000) thow PostconditionException;
.....
}
```

3.5.6 Transformació de Valors Inicials

Un valor inicial és una restricció que s'ha de complir per un atribut o per un final d'una associació. Una forma d'implementar aquesta restricció seria durant la creació de la classe que conté l'atribut o durant la classe que representa el final d'associació.

Exemple OCL:

```
context Persona::ingressos() : Integer
init: self.pares.ingressos->sum() * 1 %;
```

Transformació Java:

```
private int sumIgressos(Collection pares) {
       Iterator it = pares.iterator();
       Int ingress = 0;
       while (it.hasNext()) {
            ingress=+((Persona)it.next()).ingressos;
       }
       Return ingress;
}
....

setIngressos(sumIngressos(this.pares)*0,1);
```

3.5.7 Transformació de Valors Derivats

Un valor derivat és una restricció que s'ha de complir per un atribut o per un final d'una associació. Una forma d'implementar aquesta restricció seria durant la creació de la classe que conté l'atribut o durant la classe que representa el final d'associació.

Exemple OCL:

```
context Persona::ingresos() : Integer
derived: if menorEdat  then self.pares.ingresos->sum() * 1 %
       else treball.sou endif
```

Transformació Java:

```
Private ingressosDerivat() {
       Int ingress ;
       If this.menorEdat then ingress= sumIngressos(this.pares)*0,1
       Else ingress=this.treball.sou;
       Return ingress;
}
....

setIngressos(ingressosDerivat());
```

3.6 Conclusions

Els generadors de codi en general i les eines avaluades en particular aporten un valor afegit en el procés de desenvolupament de programari: una codificació automàtica (però no completa) per una plataforma determinada a partir d'un model independent d'aquesta plataforma. Aquest desenvolupament suposa un augment de la productivitat, una millora de la qualitat del codi així com una independència dels detalls d'implementació respecte a la plataforma.

Cap dels dos generadors realitza satisfactòriament totes les transformacions del diagrama de classes. Hi ha relacions con la composició i l'agregació que no es tenen en compte o es generen incorrectament com les associacions n a n. Tots dos generadors tenen molt a millorar en quant a la generació automàtica de codi. Podem destacar que el generador iQgen realitza mes tasques que Axgen i que aquest genera un codi mes sofisticat que iQgen encara que té carències més bàsiques.

L'assignatura pendent de la generació automàtica de codi és la transformació de les restriccions expressades en el model amb el llenguatge OCL en una implementació real.

És evident que estem encara en els inicis d'una prometedora generació automàtica de codi basada en models.

4 Glossari

J2EE : Implementació comercial de l'arquitectura distribuïda per components
Model: Representació abstracta d'un sistema.
Plataforma: Entorn on opera una aplicació.
MDD: Model Driven Development. Desenvolupament dirigit per models.
MDA: Model Driven Architecture. Desenvolupament dirigit per models promulgat per OMG.
OMG: Object Management Group, organisme que promulga estàndards de desenvolupament.
MOF: Meta Object Facility. Llenguatge per representar models i metamodels
UML: Unified Model Language. Llenguatge unificat de modelatge
XML: Extensible Markup Language. Llenguatge de marques utilitzat per representar i transferir dades.
XMI: XML MetaData Interchange. Subconjunt del XML utilitzat per intercanviar models i metamodels.
PIM: Platform Independent Model. Model independent de la plataforma
PSM: Platform Specific Model. Model específic de la plataforma
OCL: Object Constaint Language. Llenguatge d'expressió de restriccions.
API: Application Programming Interface. Llibreria de funcions d'una aplicació accessible per programació.

5 Referències

[1] Mastering XMI Java Programming With XMI, XML, and UML
[2] OMG. “UML 2.0 OCL Specifitication.”
[3] http://www.codegeneration.net
[4] http://www.oreillynet.com
[5] http://xdoclet.sourceforge.net
[6] ArgoUML v. 0.22, http://argouml.tigris.org
[7] iQgen v.3.0 beta, http://www.innoq.com
[8] Xcoder v.1.1, http://www.liantis.com
[9] CodeGenie v.4.3, http://www.oogenerator.com
[10] Axgen v.3.0 beta, http://axgen.sourceforge.net
[11] OMG. “Unified Modeling Language Specifitication. Version 1.3”
[12] OMG. “MDA Guide Version 1.0.1”

6 Annex A. El llenguatge XMI.

El llenguatge XMI (XML Metadata Interchange) és una especificació que s'ha dissenyat per intercanviar models o metadades en general. Inicialment va ser escrita per intercanviar models escrits en UML entre diferents eines de modelat actualment és un estàndard per la representació de la informació orientada a objecte en XML. La proposta inicial de dissenyar aquest llenguatge va ser feta per OMG (Object Management Group) i ha segut adoptada per gran part de la industria. Des de la seva versió inicial 1.0 al febrer de 1999 i la 1.1 que incorporava el suport als espais de noms un any després fins la versió actual la 2.0 que dona suport als esquemes XML aquest llenguatge ha esdevingut un estàndard.

Com a llenguatge XML que és (XMI és un subconjunt del XML) consta d'esquemes que diuen com han d'estar formats els documents.
El XML representa les dades usant elements XML, aquests consten de les següents parts:

- Una etiqueta inicial. Per exemple: <etiqueta>
    - Opcionalment una sèrie d'atributs XML, cadascun dels quals te
        - un nom i
        - uns valors
- Contingut de l'element que consisteix amb
    - Text
    - Un element
    - Una combinació dels dos
- Per últim una etiqueta final que coincideix amb l'inicial: per exemple </etiqueta>

Per exemple elements legals en XML seria:

```
<especificació titol=”Manual XMI”>
      <autor nom=”Pere”/>
      <autor nom=”Joan”/>
</especificació>
<comentari>Breu resum</comentari>
```

On hi han 4 elements on el element "especificació" amb el atribut "titol" consta de dos elements "autor" amb el atribut "nom". El element "comentari" te com contingut un text.

Per tal de distingir etiquetes amb el mateix nom pero amb significats diferents s'utilitzen els espais de noms:

```
<document xmlns:INSECTES="http://buglovers.org"
          xmlns:PROGRAMES="http://software">
      <INSECTES:bug nom="Mosca"/>
      <PROGRAMES:bug nom="D10589"/>
</document>
```

En aquest exemple hi ha dos espais de noms amb els prefixes INSECTES i PROGRAMES que serveixen per distingir els dos elements "bug".
També es poden fer referències a altres elements a traves del seu identificador (atribut id).

```
<estudiant id="E1"/>
<estudiant id="E2"/>
<estudiant id="E3"/>
<classe nom="BD" estudiants="E1 E2 E3"/>
<classe nom="PFC" estudiants="E1"/>
```

L'avantatge d'utilitzar XMI és que estandarditza la representació dels models i permet un intercanvi efectiu entre diferents eines de modelatge i altres eines com els generadors de codi.

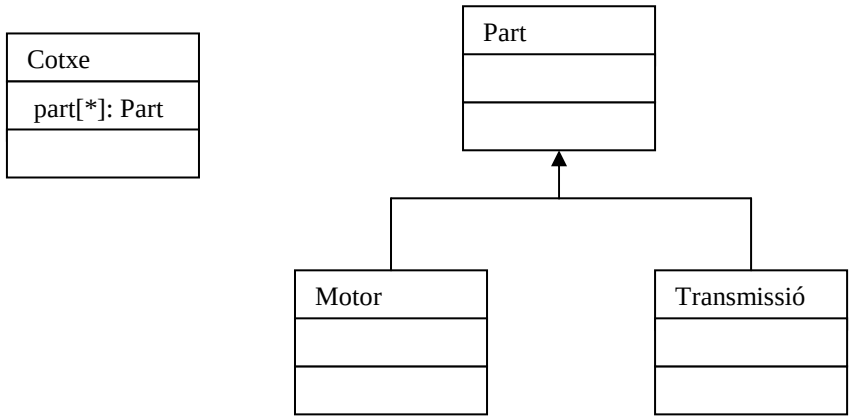
Amb el XMI podem representar models escrits en UML que consten de:

- Classes
- Instàncies
- Atributs
- Relacions
    - Composició
    - Agregació
    - Associació
- Herència
- Paquets
- Tipus

Per exemple per representar tres objectes (instàncies) de la classe cotxe en XMI:

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"/>
       <Cotxe xmi:id="_1" xmi:uuid="CA 9ABC123" xmi:label="seat"/>
       <Cotxe xmi:id="_2" xmi:label="seat"/>
       <Cotxe xmi:id="_3"/>
</xmi:XMI>
```

Per exemple aquest model representat amb UML:



Pot representar-se amb XMI:

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
                           xmlns:COTXE="http://cotxes"/>
       <COTXE:Cotxe>
              <part xmi:type="CAR:Motor"/>
       </COTXE:Cotxe>
</xmi:XMI>
```

Fig.21 : Dues representacions d'un model: gràfica en UML i textual en XMI