LLENGUATGES DE MODELITZACIÓ

4

4 LLENGUATGES DE MODELITZACIÓ

4.1 UNIFIED MODELING LANGUAGE (UML)

ENTITATS I RELACIONS : UNA CONCEPTUALITZACIÓ DEL MÓN REAL

Els sistemes informàtics han de servir per resoldre problemes d'un determinat domini. La modelització conceptual té com a principal missió representar, a través d'un llenguatge, la realitat d'aquest domini.

Aquesta representació requereix un procés d'abstracció que en permeti la seva modelització. Una aproximació acceptada àmpliament es basa en la representació del món real com un conjunt de conceptes identificables i relacionats entre sí, que poden interactuar per donar vida al sistema. Existeixen altres llenguatges i tecnologies que segueixen aquesta aproximació, coneguda amb el nom d'*Orientació a Objectes*.

EL LLENGUATGE UML

El *Llenguatge Unificat de Modelatge (UML)* és un llenguatge dissenyat per a la modelització de sistemes d'informació. L'UML ofereix un conjunt de models amb una sèrie d'elements que permeten representar tant la part estructural d'un sistema, com la part de comportament.

L'UML és el llenguatge estàndard de modelització adoptat per l'*Object Management Group* (OMG), un organisme internacional fundat l'any 1989 i integrat per organitzacions diverses, desenvolupadors, usuaris i empreses de software. El seu principal objectiu és la promoció de tecnologies orientades a objectes i la integració d'estàndards en l'establiment d'un marc comú en el desenvolupament de software.

L'UML proposa un conjunt de models que representen, a través de diagrames, diferents perspectives d'un sistema. Els diagrames de casos d'ús i els diagrames de classes i d'objectes són els més usats en la modelització de sistemes d'informació, juntament amb els

diagrames de transició d'estats i els *statecharts*. Altres diagrames proposats, com els diagrames d'implementació, queden fora de l'àmbit de la modelització conceptual.

4.2 OBJECT CONSTRAINT LANGUAGE (OCL)

MODELS UML I RESTRICCIONS

Els models UML permeten la representació gràfica d'entitats, relacions i algunes restriccions del domini. No obstant això, en molts casos apareix la necessitat d'especificar restriccions que no poden ser expressades en UML.

Una opció per expressar aquestes restriccions és la utilització del llenguatge natural. Som conscients però, que el llenguatge que els humans utilitzem com a mitjà de comunicació és una font d'ambigüitats. Precisament per aquest motiu, sorgeixen llenguatges formals com l'OCL, amb l'objectiu de cobrir la necessitat d'especificar restriccions de forma no ambigua.

EL LLENGUATGE OCL

L'*Object Constraint Language (OCL)* ha estat adoptat per l'OMG com a llenguatge d'especificació de regles aplicades a models UML.

En aquest marc, l'OCL actua com a complement de l'UML, per tal d'expressar aquelles restriccions i operacions que no poden ser representades gràficament.

L'OCL no és un llenguatge de programació, sinó un llenguatge declaratiu de modelització que no pot ser directament executat.

L'OCL és un llenguatge d'especificació pur: quan s'avalua una expressió, aquesta retorna un únic valor. A més, és un llenguatge de consulta i per tant, l'avaluació de les expressions OCL no altera l'estat del sistema.

Així doncs, el llenguatge OCL pot ser utilitzat, entre d'altres, per tal de representar invariants, regles de derivació i precondicions i postcondicions d'operacions per tal de fer més precisos els models UML.

4.3 EXTENSIONS

Un metamodel es pot veure com un "model d'un model". Els elements estàndard dels llenguatges UML i OCL estan especificats a través del seus respectius metamodels [OMG07], [OMG06].

Les aplicacions d'aquests llenguatges són diverses i el propi UML ofereix un mecanisme, els *profiles,* per extendre la seva expressivitat a través d'estereotips aplicats a les classes del metamodel.

Quan aquests mecanismes d'extensió no són suficients, s'opta per ampliar el metamodel incorporant nous elements de modelització, adaptats a un àmbit concret i fora de l'estàndard.

En l'àmbit específic de la modelització conceptual existeixen diferents propostes que amplien l'expressivitat dels models especificats en UML i OCL.

L'esquema conceptual de l'*osCommerce* presentat en aquest PFC, utilitza extensions dels models UML proposades en el llibre *Conceptual Modeling of Information Systems*, d'Antoni Olivé [Oli07], tals com :

- Representació d'entitats i relacions constants i permanents a través dels estereotips *<<Constant>>* i *<<Permanent>>*.

- Representació gràfica d'esdeveniments del sistema a través d'entitats amb els estereotips *<<Event>>*, *<<DomainEvent>>* i *<<Query>>*. Les precondicions i postcondicions dels efectes d'aquests esdeveniments són especificats en OCL i les seves restriccions (abans d'executar-se) són especificades com a operacions amb l'estereotip *<<InilC>>*.

- Especificació dels atributs derivats com a operacions OCL. Això permet aprofitar les propietats de les operacions en la definició d'atributs derivats (herència, redefinició, etc.)

- Especificació de les restriccions d'integritat com a operacions OCL. Com en el cas anterior, això permet aprofitar les propietats específiques de les operacions en l'especificació de restriccions d'integritat.

ESTRUCTURA DE L'ESQUEMA CONCEPTUAL

5

5 ESTRUCTURA DE L'ESQUEMA CONCEPTUAL

5.1 VISIÓ GENERAL

L' esquema conceptual de l'*osCommerce* està estructurat en dues parts: l'esquema estructural i l'esquema de comportament.

L'esquema estructural representa el coneixement estàtic del sistema a través de diagrames de classes i de les seves corresponents restriccions i regles de derivació.

D'altra banda, l'esquema de comportament ofereix una visió de les funcionalitats del sistema a través del diagrama de casos d'ús i la seva especificació. Els casos d'ús estan especificats en llenguatge natural i contenen enllaços als principals esdeveniments del sistema, ordenats alfabèticament i representats per classes UML. Els efectes i restriccions dels esdeveniments estan especificats en OCL.

5.2 ESQUEMA ESTRUCTURAL

L'esquema estructural de l'*osCommerce*, per les dimensions del propi sistema, està format per una important quantitat de tipus d'entitat, relacions, regles de derivació i restriccions.

Les dimensions del sistema fan necessària una fragmentació de l'esquema estructural en subesquemes que agrupen conceptes relacionats i que en conjunt, conformen l'esquema estructural complet.

Inicialment, per tal de donar una visió global de tots aquests subesquemes estructurals, es presenta un diagrama de classes amb les entitats més importants del sistema.

Seguidament, es mostren els subesquemes que conformen l'esquema estructural complet.

Així doncs, l'esquema estructural es presenta com un conjunt de diagrames interrelacionats que, de forma conjunta, representen el coneixement sobre el domini analitzat.

Amb l'objectiu de facilitar la comprensió del lector, cada subesquema es presenta seguint el format de la següent plantilla :

Nom del subesquema

■ **Overview**

Breu explicació sobre el coneixement del domini representat pel subesquema.

■ **Structural Schema**

Representació del coneixement del domini representat pel subesquema mitjançant un diagrama de classes UML. Les entitats que formen part d'altres subesquemes tenen els atributs ocults i contenen referències al subesquema on estan completament especificades.

■ **Operations**

Operacions auxiliars usades a les restriccions i a les regles de derivació.

■ **Derivation Rules**

Regles de derivació. Per criteris d'uniformitat estan especificades amb operacions OCL.

■ **Constraints**

Restriccions d"integritat no expressades gràficament. Per criteris d'uniformitat estan totes especificades a través d'operacions booleanes en OCL.

■ **Description**

Descripció detallada en llenguatge textual del subesquema estructural.

■ **Example**

Exemple per millorar la comprensió a través d'un diagrama d'instàncies.

Seguidament, s'expliquen més detalladament els diferents conceptes que hi estan implicats.

DIAGRAMA DE CLASSES

Un diagrama de classes modelitza l'estructura estàtica dels conceptes del domini i de les seves interrelacions. És un dels diagrames més importants proposats per l'UML, en tant que la identificació dels conceptes d'un domini és una de les activitats clau de la modelització conceptual i de l'especificació d'un sistema.

Els elements més comuns que formen part del diagrama de classes i que utilitzarem a nivell conceptual són :

- **Classes**:

  Les classes són l'element UML que permet representar allò que conceptualment coneixem com a **tipus d'entitats**. Cada tipus d'entitat representa un concepte rellevant del sistema d'informació especificat.

- **Atributs**:

  Les classes conceptuals del sistema poden tenir atributs que caracteritzen les seves propietats.

- **Associacions (relacions):**

  Les classes representen conceptes i els conceptes d'un domini estan relacionats entre sí. Les associacions representen aquestes relacions. Els extrems de cada associació tenen una cardinalitat i poden tenir un nom de rol que els identifica.

- **Agregacions:**

  Són un tipus genèric d'associació binària. Representen relacions *part-tot*, de tal manera que una entitat es veu com a suma de les seves parts (que hi estan relacionades a través d'agregacions).

- **Composicions:**

  És un tipus específic d'agregació que requereix que una part estigui sempre inclosa en almenys un tot. Gràficament es representen com una agregació amb el rombe pintat.

- **Generalitzacions/Especialitzacions:**

  Un tipus d'entitat és una especialització d'un altre tipus d'entitat si la més específica és del tipus d'entitat general però té més propietats. D'altra banda, les propietats del tipus d'entitat general ho són també de totes les entitats específiques.

- **Constant i permanent:**

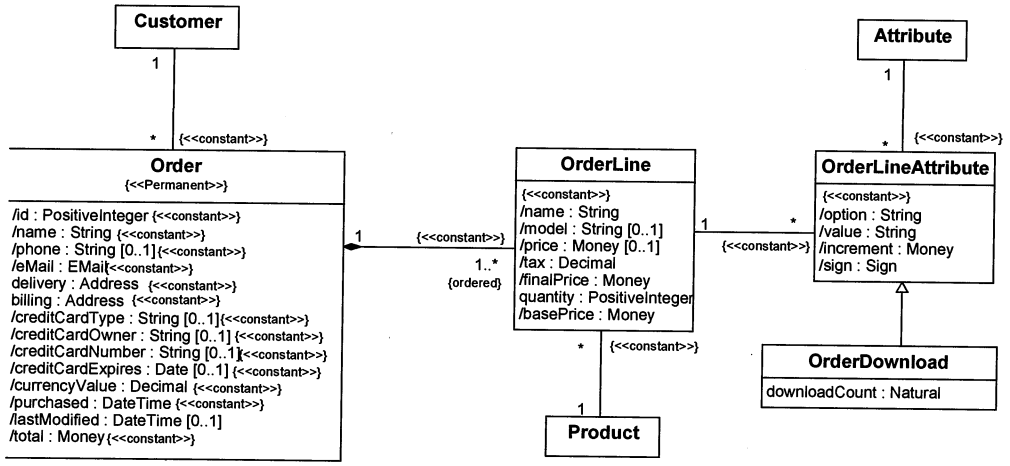
  Un tipus d'entitat és constant si les seves entitats són les mateixes sempre.

  Un tipus d'entitat és permanent si les seves entitats mai deixen de ser-ho. A diferència dels tipus d'entitat constants, poden haver-hi noves entitats.

  Un tipus de relació és constant respecte a un participant si les instàncies en les que aquest participa són sempre les mateixes. Si una relació és constant respecte a tots els participants, es diu que la relació és constant.

  Un tipus de relació és permanent respecte a un participant si les instàncies en les que aquest participa mai deixen d'existir. Si una relació és permanent respecte a tots els participants, es diu que la relació és constant.

El següent fragment d'esquema representa gràficament bona part dels conceptes explicats anteriorment:



La classe *Order* té atributs constants (un atribut es pot veure com una associació) i d'altres que no ho són. Està relacionada amb la classe *Customer*, especificada en un altre subesquema. La relació entre *Customer* i *Order* és constant respecte *Order*. La relació entre *Order* i *OrderLine* és del tipus composició. Una *OrderLine* està relacionada amb un producte i a la vegada amb més d'una *OrderLineAttribute*. Un tipus específic d'*OrderLineAttribute* és *l'OrderDonwload*. *OrderLineAttribute* està relacionat amb un *Attribute* i *OrderDownload* també.

OPERACIONS AUXILIARS

La versió 2.0 del llenguatge OCL permet la definició d'operacions en el context d'una classe, que poden ser usades a la resta d'expressions OCL. Aquestes operacions tenen la següent sintaxi:

context *NomTipusEntitatContext* **def:**
nomOperació(paràmetre:Tipus,...) : TipusRetorn = Expressió_OCL

REGLES DE DERIVACIÓ

Els tipus d'entitat i relació poden ser derivats. Un tipus derivat és aquell pel qual es pot saber la seva població en qualsevol moment a través d'una regla de derivació. Les regles de derivació poden ser especificades en OCL, a través d'una sintaxi específica:

context *NomTipusEntitatContext :: NomAtributDerivat : Tipus*
derive: *Expressió_OCL*

Una altra opció és aprofitar les operacions OCL i les seves propietats associades (herència, redefinició, etc.). Aquesta ha estat l'opció escollida uniformement per especificar totes les regles de derivació:

context *NomTipusEntitatContext:: NomAtributDerivat():Tipus*
body : *Expressió_OCL*

RESTRICCIONS D'INTEGRITAT

Les restriccions d'integritat aplicades a un esquema conceptual s'interpreten com a condicions que asseguren la seva integritat (validesa i completesa).

Algunes restriccions d'integritat (cardinalitat de les associacions, tipus constants i permanents, característiques pròpies de les agregacions i composicions, etc.) es poden representades gràficament en UML.

D'altres, són especificades en OCL. Aquest llenguatge contempla una sintaxi específica que permet expressar restriccions d'integritat com a invariants, és a dir condicions que s'han de complir contínuament:

context *NomTipusEntitatContext* **inv** *NomRestricció* :
Expressió_OCL_Booleana

Una altra opció és aprofitar les operacions OCL i les seves propietats associades (herència, redefinició, etc.). Per uniformitat, a l'igual que amb les regles de derivació, aquesta ha estat l'opció escollida per definir les restriccions d'integritat:

context *NomTipusEntitatContext*:: *NomRestricció*():Boolean
body : *Expressió_OCL_Booleana*

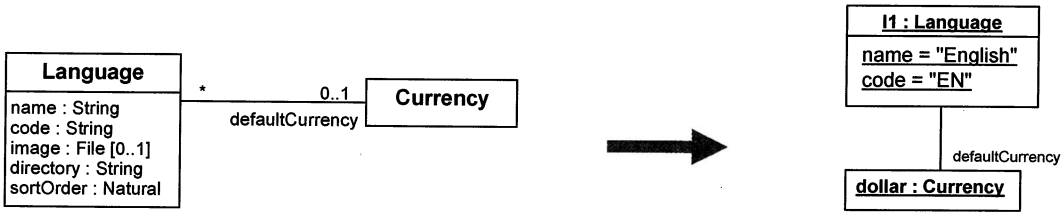
DIAGRAMA D'INSTÀNCIES

Un diagrama d'instàncies està format per classes que no representen tipus d'entitat i de relació, sinó les seves instàncies (relacions i entitats). Es poden veure com una fotografia parcial o completa de l'estat d'un sistema d'informació en un moment concret.

Cada classe del diagrama d'instàncies s'anomena *Instance Specification* i els valors concrets d'atributs i relacions per a cada instància, *Slots*. Si els *Slots* són instàncies de relacions, reben el nom de *Links*.

Els diagrames d'instància són utilitzats, a l'esquema conceptual de l'*osCommerce,* per tal de mostrar instanciacions d'exemple per millorar la comprensió de l'esquema conceptual.

Les següents il·lustracions mostren un exemple d'aquest tipus de diagrama: El primer representa el subesquema conceptual relacionat amb els llenguatges del sistema. El segon representa gràficament una possible instanciació d'exemple.



5.3 ESQUEMA DE COMPORTAMENT

L'esquema de comportament presenta inicialment un **diagrama de casos d'ús** que mostra de forma agrupada les funcionalitats del sistema, relacionades amb els actors principals que hi participen.

Seguidament, es detalla l'especificació de cada cas d'ús de forma textual, seguint el format de la següent plantilla :

Use case
Nom Cas d'ús
Primary Actor: *Actor principal del sistema.* **Precondition:** *Condicions que s'han de satisfer abans d'executar el cas d'ús.* **Trigger:** *Acció d'inici del cas d'ús.*
Main Success Scenario:
L'escenari principal especifica, en forma de seqüència numerada d'activitats, la interacció entre el sistema i els actors del cas d'ús.
Extensions:
Les extensions són interaccions sistema-usuari alternatives a l'escenari principal del cas d'ús. El primer nombre de la seva numeració indica el pas de l'escenari principal on s'inicia l'escenari alternatiu.

L'especificació dels casos d'ús invoca de forma textual els principals esdeveniments del sistema. La sintaxi utilitzada és la següent:

[→ *NomEsdeveniment*]

Els esdeveniments de sistema es presenten alfabèticament, per tal de facilitar la seva cerca, i s'especifiquen prenent com a base la següent plantilla :

Event

Nom Esdeveniment

■ Event diagram

Els esdeveniments de sistema es modelitzen com a classes UML amb l'estereotip <<Event>> i amb una operació que representa l'efecte de la seva execució sobre la base d'informació.

■ Initial Integrity Constraints

Restriccions que s'han de complir abans d'iniciar l'efecte de l'esdeveniment. S'expressen en OCL.

■ Effect

Efecte de l'execució de l'esdeveniment de sistema. S' especifica en forma d'operació OCL.

CAS D'ÚS

Un cas d'ús representa una funcionalitat del sistema d'informació.

Els actors són aquells rols d'usuari que participen en els casos d'ús.

L'esquema conceptual de l'*osCommerce* especifica els casos d'ús de forma textual, com una seqüència d'interaccions entre el sistema i els actors que hi participen.

Existeixen diferents tipus de cas d'ús. A nivell conceptual, utilitzarem els casos d'ús essencials, vistos com un caixa negra del sistema. El seu objectiu és descriure què fa el sistema independentment de la tecnologia emprada i de la interfície gràfica utilitzada.

Constantine i Lockwood [CoL99] defineixen la idea de cas d'ús essencial com:

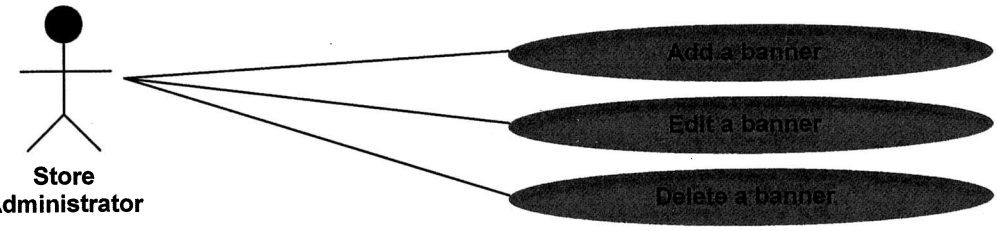
" Un cas d'ús essencial [...] és un cas d'ús simplificat, abstracte i general que captura la intensió de l'usuari d'una manera independent a la tecnologia i la implementació. [...] Un cas d'ús essencial és una narració estructurada, expressada en el llenguatge d'aplicació del domini i els seus usuaris i que descriu, sense fer esment a detalls tecnològics ni d'implementació, una tasca o interacció".

Els casos d'ús concrets, en canvi, expliciten la interacció amb la interfície i la tecnologia utilitzada i tot i que poden ser d'utilitat en la fase de disseny, queden fora de l'àmbit de la modelització conceptual.

DIAGRAMA DE CASOS D'ÚS

El diagrama de casos d'ús representa gràficament els diferents casos d'ús del sistema i els actors principals que hi participen.

El següent diagrama mostra un fragment del diagrama de casos d'ús, on es mostren 3 casos d'ús iniciats pels usuaris que tenen assignat el rol d'*Store Administrator*: *Add a banner*, *Edit a banner* i *Delete a banner*.



ESDEVENIMENTS DEL SISTEMA

Existeixen diferents tipus d'esdeveniments de sistema. Els utilitzats en l'esquema conceptual de l'*osCommerce* són els següents :

- **Esdeveniments de domini**: Són un conjunt d'esdeveniments estructurals (canvis simples de l'estat de la base d'informació) que es perceben com un sol canvi a nivell conceptual.

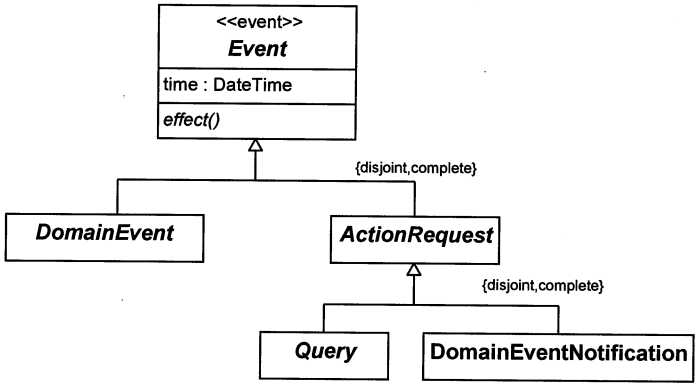
- ***Action Request Events* (ARE)**: Un sistema d'informació realitza accions que poden suposar la modificació de la base d'informació o una comunicació d'informació. Aquestes accions es representen a través d'*Action Request Events*.

  Així mateix, hi ha diversos tipus d'AREs. Els més significatius són els següents:

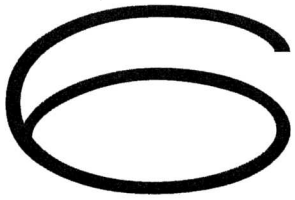
- ***Domain Event Notification***: És una acció del sistema que només té com a efecte un canvi a la base d'informació, és a dir, l'efecte d'un esdeveniment de domini. Per tal de simplificar l'esquema conceptual assumim que cada esdeveniment de domini té un *Domain Event Notification* associat.

- ***Query***: Una *query* és un tipus específic d'ARE que té com a efecte donar una informació. No implica cap canvi a la base d'informació.

El següent diagrama de classes mostra la taxonomia dels esdeveniments utilitzada a l'esquema conceptual de l'*osCommerce*:



EINES DE SUPORT



6 EINES DE SUPORT

6.1 MAGIC DRAW

Les eines *CASE* (*Computer Aided Software Engineering* o Enginyeria del software assistida per ordinador) són eines dirigides a millorar la productivitat en els processos de desenvolupament de software.

Magic Draw [MaD07] és una eina CASE que té l'objectiu de facilitar la modelització gràfica en UML. Abans de l'inici del projecte es van analitzar altres eines de modelització en UML (*Poseidon*, *ArgoUML* o *Rational Rose*). Amb tot, però, l'eina *MagicDraw* és la que s'ajusta més a les necessitats del projecte. La majoria d'eines utilitzen el metamodel de l'UML 1.4 amb funcionalitats afegides per representar alguns elements del metamodel de l'UML 2.0.

La versió *MagicDraw* 12.0 ofereix facilitats interessants d'exportació de diagrames i està implementada prenent com a base el metamodel de l'UML 2.0 i per tant, permet representar la majoria d'elements que necessitem per elaborar la part gràfica de l'esquema conceptual (associacions ternàries, classes associatives, diagrames d'instàncies,...).

MagicDraw és una eina comercial. No obstant això, ofereix una versió gratuïta, amb funcionalitats més reduïdes que la versió de pagament (tot i que suficients per a l'elaboració de diagrames de classes) dirigida a desenvolupadors que treballen en projectes no comercials.

6.2 USE

L'eina *USE* (*UML Based Specification Environment*) [USE07] és un entorn per a l'especificació i validació de sistemes d'informació especificats en UML i OCL.

L'eina ha estat desenvolupada per la Universitat de Bremen i permet la instanciació d'esquemes conceptuals representats a través del subconjunt implementat d'elements de l'UML i l'OCL.

El principal avantatge d'aquesta eina és la possibilitat de validar la sintaxi de les expressions OCL i la seva consistència amb l'esquema UML. L'eina també permet la creació d'objectes que representen instàncies de l'esquema conceptual i l'avaluació d'expressions OCL en funció d'aquestes instàncies.

La possibilitat de validar expressions OCL (restriccions, regles de derivació, operacions addicionals...) ha estat una funcionalitat de màxima utilitat en el marc d'aquest projecte.

6.3 MS VISIO

Microsoft Visio [Mic07] és un software comercial per a l'elaboració de diagrames empresarials i tècnics a partir de formes gràfiques ja definides. El programa s'emmarca dins el paquet Office de l'empresa *Microsoft*, tot i que és utilitzable gratuïtament per als estudiants de la Universitat Politècnica de Catalunya a través de l'acord MSDNAA (*Microsoft Developer Network Academic Alliance*) subscrit entre la universitat i l'empresa.

La seva utilització s'ha centrat en la creació d'esquemes de representació de l'estructura de la base de dades.

6.4 EASYPHP

La instal·lació del sistema *osCommerce* requereix l'execució de codi PHP, la seva connexió amb una base de dades *MySQL* i la publicació del sistema en un servidor web.

EasyPHP [FAT03] és un paquet de software que facilita la instal·lació, la interconnexió i la gestió de tots aquests elements i permet una fàcil instal·lació local del sistema *osCommerce* en un ordinador personal. A més, el paquet disposa de la utilitat *PHPMyAdmin* que permet la realització de consultes i modificacions de la base de dades *MySQL* a través d'una interfície web.

Així doncs, la utilització d'aquest paquet no només ha servit per a la instal·lació i l'experimentació amb el sistema, sinó que ha resultat una eina d'utilitat per a l'anàlisi de la base de dades.

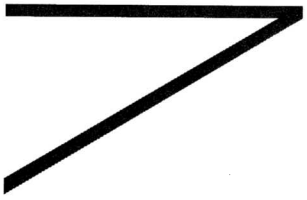
6.5 MICROOLAP

El software *MicroOLAP* [MiO07] es distribueix en modalitat *Shareware,* és a dir, és possible provar el sistema durant un període limitat de temps de forma gratuïta. En cas d'exhaurir el temps de prova s'ha d'adquirir el producte si es vol continuar usant-lo.

La funcionalitat més important del producte per al nostre projecte és la possibilitat d'obtenir un esquema conceptual gràfic d'una base de dades *MySQL*. Aquesta funcionalitat ha estat usada per a l'obtenció d'un esquema complet de la base de dades de l'*osCommerce* per tal de disposar d'un element més intuïtiu d'anàlisi a part de les consultes i l'exploració directa de la base de dades.

CONCEPTUAL SCHEMA

SCHEMA
PRESENTATION



7 SCHEMA PRESENTATION

7.1 THE *osCommerce* SYSTEM

E-commerce allows people to exchange goods and services with no barriers of time or distance. Surfing the net, you can easily find online shops where you can buy almost anything you need, at any time and without leaving home.

Electronic commerce changes the way of businesses operate. Nowadays, it is possible to start up an online store open 24 hours with lower running costs than a traditional establishment. The idea of small local shops makes no sense to e-commerce: online stores have been internationally accessible since the moment they started operating.

osCommerce is an e-commerce solution available as free software under the GNU General Public License. The *osCommerce* project was started in March 2000 in Germany and since then it has become the base of thousands of online stores around the world.

7.2 CONCEPTUAL MODELING

An information system is useful for people who work in a domain if it knows information about it. Conceptual modeling is the activity that elicits the general knowledge that an information system needs to know [Oli07]. The conceptual schema is the specification of that knowledge.

To develop an information system it is necessary to first define its conceptual schema. In some cases the conceptual schema is well documented and accessible and sometimes it is only known by the software developers.

7.3 REVERSE ENGINEERING

In a context of continuous changes, the purpose of software engineering is to improve the productivity of the development process, focusing on quality and the unavoidable necessity of maintenance.

This is why it is so important to decide what an information system must do and know before implementing it. Documenting this process and its results is essential in order to improve quality, communication between developers (which can change over time) and cost of maintenance.

The concept of reverse engineering involves inverting the usual order of engineering. Chikofsky and Cross [ChC90] define reverse engineering as follows:

"Reverse engineering is the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction".

That is, reverse engineering can be seen as the inverse process of the traditional life cycle of software engineering.

"The purpose of reverse engineering is to understand a software system in order to facilitate enhancement, correction, documentation and redesign" [ChC90].

7.4 THE *osCommerce* CONCEPTUAL SCHEMA

The main purpose of this paper is to present a conceptual schema of the *osCommerce* system as a result of a reverse engineering process. This conceptual schema is specified using standard UML and OCL with some extensions to improve expressivity (proposed in [Oli07]).

The osCommerce conceptual schema is presented in chapters 2 and 3. It was created using the public documentation of the system (sometimes limited and imprecise), by experimenting as a user with the current version and by analyzing the database schema.

It also contains conceptual improvements on the current knowledge and behaviour of the system. Differences between the conceptual schema and the *osCommerce* implementation are described in Chapter 4. Finally, Chapter 5 gives a list of improvements and solutions.

We publish this work and make it accessible for the community. It can serve as a complementary documentation of the system and provide detailed specification of its knowledge and behaviour. It is useful for engineers who are going to develop online stores based on *osCommerce* and for anybody interested in learning more about it.

STRUCTURAL SCHEMA