INTRODUCCIÓ

L'UML no defineix una metodologia concreta de desenvolupament del programari orientat a objectes. Entre les metodologies orientades a objectes basades en UML més conegudes hi ha: UP, ICONIX, FDD, ... El procés prescrit per ICONIX:

- és una simplificació del procés que cal seguir amb UP;
- complementa alguns dels aspectes que amb UP no acaben de quedar prou clars.

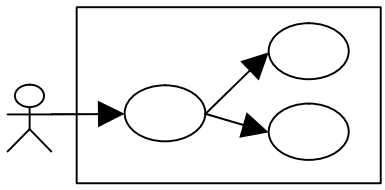
A la primera part d'aquest tema, dedicat a donar una visió general de la definició de requeriments, l'anàlisi i el disseny orientats a objectes fent servir UML, donarem una descripció d'un possible procés a seguir que és semblant al prescrit per ICONIX. A la segona part del tema comentarem algunes qüestions relatives a la definició de requeriments i a l'anàlisi orientats a objectes no tractades a la primera part.

REQUERIMENTS

Durant la definició de requeriments (funcionals) del sistema es genera el ***model de casos d'ús*** que reflecteix les *responsabilitats* del sistema a construir. Cada cas d'ús s'especifica per mitjà d'una fitxa en la que es descriuen l'escenari principal i els escenaris alternatius, si cal amb l'ajuda de diagrames d'activitat. Els diagrames de casos d'ús visualitzen els actors, els casos d'ús i les seves relacions.

Durant la definició de requeriments també s'acostuma a fer prototipus de les ***interfícies d'usuari***.

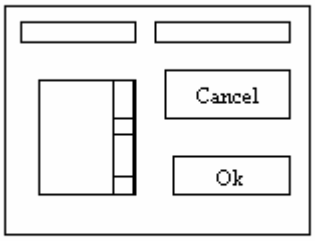
- **Modelat de Casos d'Ús**
  - Identifcar els casos d'ús.
  - Escriure les fitxes dels casos d'ús: descriure l'escenari principal i els escenaris alternatius.
  - Organizar els casos d'ús en paquets.
  - Dibuixar els diagrames de casos d'ús.



Per aplicacions de grandària mitjana/gran és convenient organitzar els casos d'ús en paquets de casos d'ús

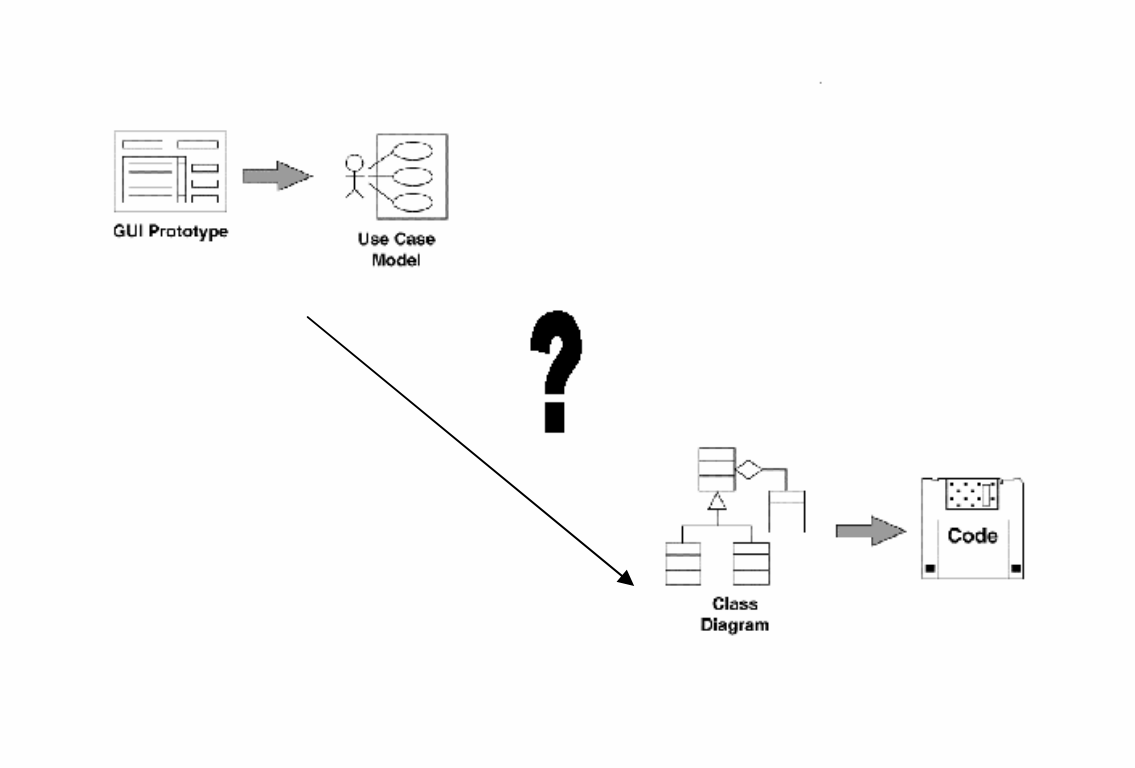
- **Prototipatge d'Interfícies d'usuari**

  - Si és possible, fer el prototipatge de les principals interfícies d'usuari del sistema a construir.



Abans de començar a escriure el codi de l'aplicació a construir necessitem disposar del disseny del conjunt complet de les classes de l'aplicació, amb els seus atributs i operacions. Aquesta informació es representa per mitjà del diagrames de ***classes de disseny***.

El problema que se'ns planteja és com anar des dels casos d'ús i els prototipus d'interfície d'usuari fins als diagrames de classes de disseny i el codi.



L'Anàlisi i el Disseny ens ajudaran a resoldre el problema que tenim plantejat.

ANÀLISI

L'*anàlisi* de sistemes s'ocupa de la investigació del problema a resoldre (*que*), però no s'ocupa de trobar una solució al problema (*com*). Durant l'anàlisi es busquen i descriuen els objectes del domini del problema (atributs i associacions) a partir dels casos d'ús especificats – refinats si es consideri necessari - en el document de definició de requeriments del sistema a construir. El resultat és un conjunt de diagrames que mostren les ***classes d'anàlisi*** que conformen l'anomenat ***model del domini***.

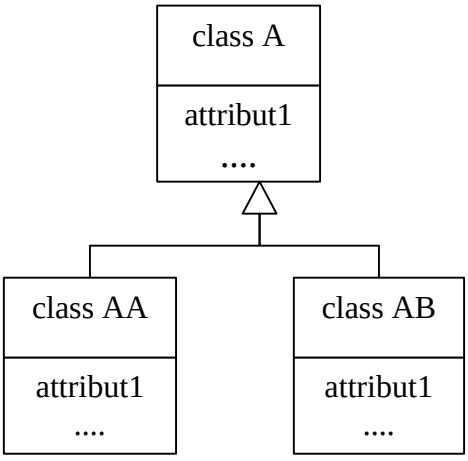
Com a resultat de l'anàlisi es mostra, per mitjà de ***diagrames de col·laboració simplificats***, com interactuen els objectes de les classes d'anàlisi per tal d'assolir el propòsit dels diferents casos d'ús.

Durant l'anàlisi també es refina la descripció de les interfícies d'usuari obtinguda durant la definició de requeriments del sistema a construir.

La primera part de l'anàlisi es centra en el Modelat del Domini del problema**.**

- ***Modelat del Domini***

  Durant aquesta fase s'identifiquen els objectes del mon real i les generalitzacions i associacions entre aquests objectes. El model del domini es representa per mitjà de **diagrames de classes.**



La segona part de l'anàlisi es centra en la determinació dels objectes del sistema necessaris per a la realització de cada cas d'ús i en el comportament d'aquests objectes.

- ***Anàlisi dels objectes.***

  - S'identifiquen els objectes que participaran a cada cas d'ús.
  - Es determinen les classes d'anàlisi (de frontera, de control i d'entitat) que són necessàries per a realitzar cada cas d'ús.
  - El model del domini es transforma en el ***diagrama de classes d'anàlisi***.

Les classes d'anàlisi, més detallades que les utilitzades en el model del domini, constitueixen l'esquelet de les classes de disseny.

- ***Anàlisi del comportament.***

  - Cal conèixer quines seran les accions que realitzarà el sistema com a resultat de l'interacció amb l'usuari.

  - Es distribueix el comportament de cada cas d'ús entre els objectes associats a les classes d'anàlisi.

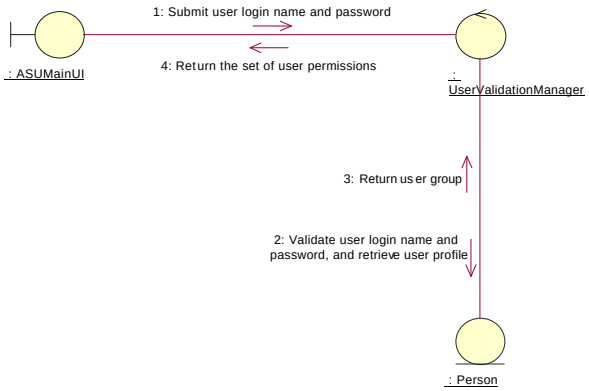
  - El sistema es modela com una "caixa negra": només es modelen les funcionalitats externes per mitjà de ***diagrames de col·laboració simplificats***.

    - Per a cada Cas d'Ús:

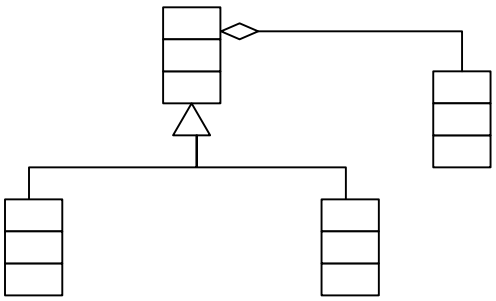
      - Identificar els objectes d'anàlisi (de frontera, de control i d'entitat) necessaris per a realitzar els diferents escenaris del cas d'ús.

      - Proposar un conjunt d'objectes que per mitjà de col·laboracions resolguin cadascun dels escenaris del cas d'ús.

      - Fer un diagrama d'iteració (generalment de col·laboració) simplificat.

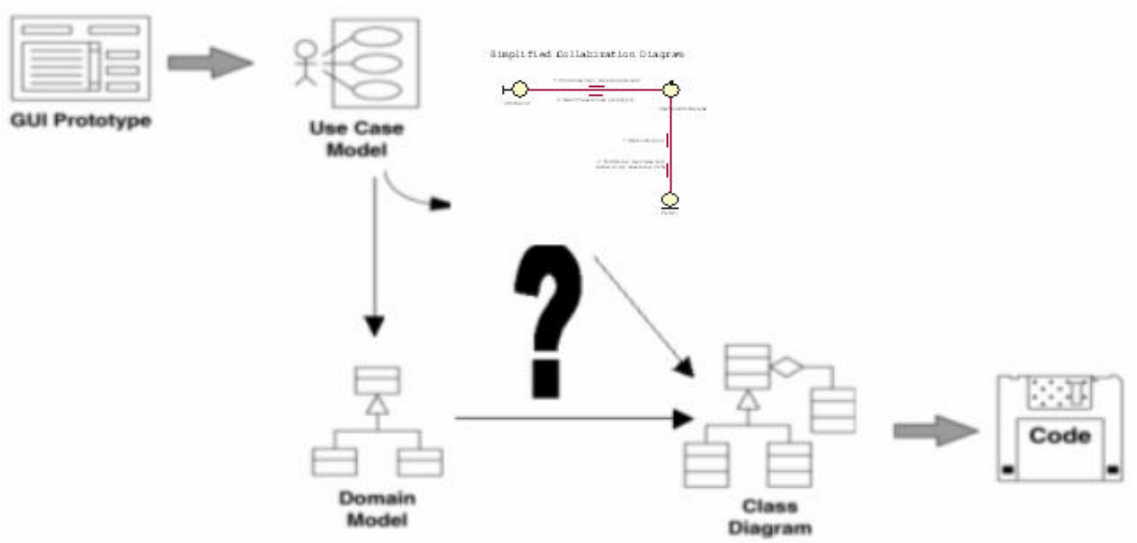


    - Crear el ***diagrama de classes d'anàlisi*** a partir del model del domini, afegint els nous objectes, atributs i associacions a mesura que es van descobrint.



És possible que al passar a la fase de disseny, algunes de les classes de control desapareixin i acabin convertinse en mètodes de classes de frontera o d'entitat.

Queda per resoldre com passar dels diagrames de col·laboració simplificats i del diagrama de classes d'anàlisi al diagrama de classes de disseny i al codi.



DISSENY

A partir dels resultats de l'anàlisi, el *disseny* s'ocupa de crear una solució conceptual que permeti implementar els requeriments del sistema a construir. El disseny es l'etapa anterior a la implementació del sistema.

Durant el disseny es defineix la forma com col·laboren els objectes de disseny (directament implementables) per tal d'aconseguir el requeriments del sistema a construir.

Per a representar aquesta visió *dinàmica* de les col·laboracions entre objectes es fan servir *diagrames de seqüència* que mostren el flux de missatges entre objectes, i per tant la invocació de operacions.

Durant el disseny orientat a objectes també es crea una vista *estàtica* de les classes per mitjà del *diagrama de classes de disseny*.

En el diagrama de classes de disseny es mantenen els atributs que existeixen en el diagrama de classes d'anàlisi, s'afegeixen totes les operacions que apareixen en els diagrames de seqüència i es refinen les associacions: multiplicitat, navegabilitat, agregacions, etc.

El disseny s'ocupa dels objectes tal com seran codificats. El diagrama de classes de disseny amplia les classes d'anàlisi amb operacions, afegeix les classes necessàries per a poder fer la implementació final i incorpora les associacions entre classes prèviament no identificades. La informació necessària per a decidir quines classes han de ser responsables de quines operacions s'obté durant el procés de creació dels diagrames de seqüència.

Es dibuixa un diagrama de seqüència per a cada cas d'ús i durant la seva creació s'assignem les operacions a les diferents classes. A partir dels diagrames de seqüència es poden veure les responsabilitats de les diferents classes i la seva interacció.

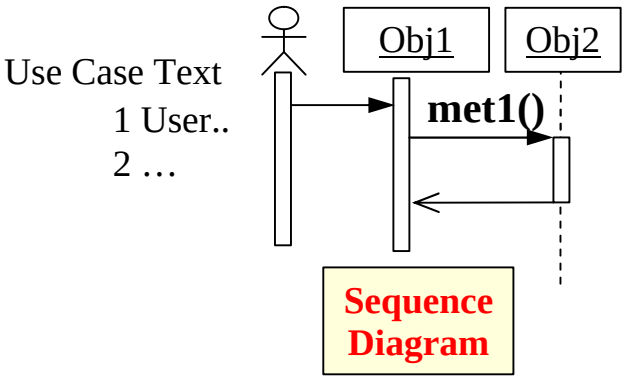
Es dibuixen diagrames d'estat pels objectes de classes amb comportament complex.

- ***Modelat de la interacció***

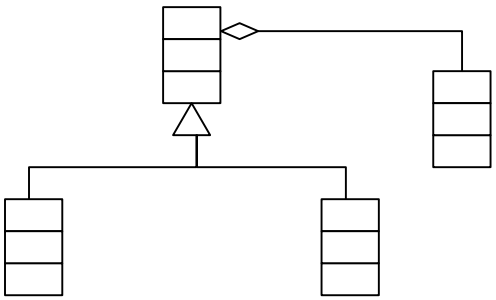
  - Per a cada cas d'ús:

    - Identificar els missatges que s'han de passar entre objectes, els objectes i les operacions involucrades.

    - Dibuixar un diagrama de seqüència que mostri el flux de missatges entre els objectes que interactuen per a resoldre el cas d'ús.

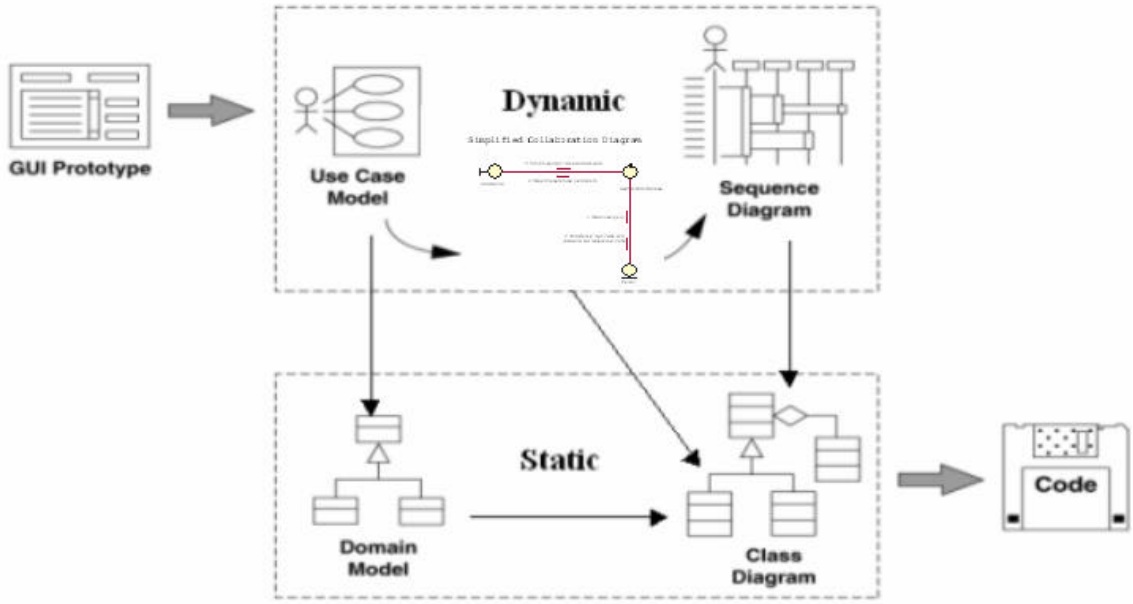


  - Crear el ***diagrama de classes de disseny*** a partir del diagrama de classes d'anàlisi, afegint atributs, operacions i associacions a mesura que es van trobant mentre es dibuixen els diagrames de seqüència.



Les classes de disseny han de ser completes: amb tots els atributs i mètodes.

VISIÓ GLOBAL DEL PROCÉS



Desenvolupament iteratiu i incremental

Convé insistir en que els processos orientats a objectes són iteratius i incrementals, per tant la definició de requeriments, l'anàlisi i el disseny no queden completats fins que comença la fase de implementació. En general és difícil marcar la frontera entre anàlisi i disseny.

- Cada cicle (iteració) consisteix d'un o diversos processos (requeriments, anàlisi,...).

  Normalment només s'implementa a cada cicle una part (un increment) de les funcionalitats requerides.

- Durant tot el procés es tenen presents els requeriments: els objectes es disseny per a satisfer els casos d'ús (traçabilitat).

- A cada cicle els requeriments es verifiquen i validen.

La resta d'aquest tema la dedicarem a comentar qüestions relatives a la definició de requeriments i a l'anàlisi orientats a objectes no tractades a la primera part.

REQUERIMENTS DEL SISTEMA

Els *requeriments* del sistema són el conjunt d'idees que el client té sobre què ha de ser el software a desenvolupar. Un requeriment és una capacitat que un sistema ha d'assolir. Els requeriments delimiten les fronteres i les prestacions del sistema.

Els requeriments són dinàmics, canvien durant la vida d'un projecte. Hi ha estudis que demostren que la principal font de fracàs dels projectes és la naturalesa canviant dels requeriments i la falta d'habilitat per manegar aquesta circumstància.

La gestió dels requeriments consisteix en descobrir, organitzar i documentar les necessitats reals que porten a un clients a sol·licitar el desenvolupament d'un sistema, així com mantenir un acord continu sobre els requeriments que canvien.

La gestió de requeriments comporta les següents avantatges:

- Els requeriments poden tenir una prioritat i un seguiment;
- Les inconsistències en els requeriments es poden detectar més fàcilment;
- Faciliten el seguiment de l'estat del projecte.

En una aplicació hi ha bàsicament dos tipus de requeriments:

- *Funcionals*. Descriuen bàsicament quins serveis ha d'oferir l'aplicació, independentment de la implementació. Cada requeriment funcional expressa una relació entre les entrades i sortides del sistema, és a dir, especifica les sortides que s'han de produir a partir d'unes determinades entrades i les operacions necessàries per aconseguir això. També han d'especificar com s'ha de comportar el sistema davant de situacions anormals (entrades invàlides, errors, etc.).

- *No funcionals*. Descriuen restriccions, normalment quantificables, que venen imposades pel client o pel mateix problema. Les restriccions més habituals es refereixen a:

  - terminis, pressupost;
  - característiques d'execució: nombre de terminals, usuaris simultanis, etc;
  - comportament d'execució del sistema: temps de resposta, etc.;
  - format dels informes, procediments d'assignació de comptes, auditories, etc.;
  - disponibilitat de recursos: tipus de màquina, S.O., llenguatge, SGBD sobre el que haurà de córrer, capacitat d'emmagatzematge;
  - recuperació i fiabilitat davant errors;
  - seguretat: control d'accés a dades, condicions d'accés de diferents usuaris, etc;
  - interfícies externes: interacció amb usuaris, hardware i altres mòduls de software;
  - objectius de disseny o requeriments de qualitat: fàcil d'utilitzar, de mantenir, ampliable.

El més important i difícil és *analitzar* els problemes a resoldre per l'aplicació a construir, sobre tot si són complexos, per tal de poder *definir* quines són les funcionalitats que el sistema haurà de proporcionar, i per conèixer en quines condicions haurà de funcionar.

L'anàlisi i definició dels requeriments funcionals i no funcionals es recullen en el *document de requeriments del sistema.*

El document de requeriments acostuma a incorporar també els requeriments de les interfícies d'usuari. Els casos més habituals d'interfícies d'usuari són: entrades mitjançant teclat i pantalla, sortides per pantalla i impressora, entrades i sortides sonores i entrades per escàner.

Un enginyer del programari no només ha de tenir coneixements tècnics, sinó que també ha de saber comunicar-se (oralment i per escrit) i tenir nocions de gestió de projectes. Un gran problema que afecta a molts informàtics és la seva incapacitat d'entendre's amb els clients que encarreguen les aplicacions, i això és l'origen de molts malentesos i situacions desagradables.

Les tasques lligades a la generació del document de requeriments del software són:

- **Reconeixement del problema.** Comprendre el software dins del context;
- **Anàlisi i síntesi del problema.** Flux, estructura i contingut de la informació. Definir funcions. Entendre el comportament;
- **Modelització.** Fer models del sistema;
- **Definició.** Representació formal del *domini del problema*;
- **Revisió.** Comprovar que tant client com analista han "entès" el problema.

Les propietats desitjables d'un document de requeriments del software són:

- Que especifiqui el comportament extern, no la implementació;
- Que sigui comprensible;
- Que no tingui ambigüitats;
- Que sigui complet;
- Que sigui verificable;
- Que sigui consistent;
- Que sigui modificable;
- Que sigui seguible;
- Que sigui utilitzable també per fer el manteniment.

Algunes de les dificultats per elaborar un bon document de requeriments són:

- Cal desenvolupar una aplicació completament nova, sense tenir-ne cap referent;
- No saber a qui demanar la informació per comprendre el problema;
- Existència de diversos usuaris del nou sistema, probablement amb requeriments i prioritats diferents;
- Els usuaris finals no sempre són els que encarreguen el software, i hi poden haver col·lisions entre els requeriments desitjats i el pressupost assignat pel desenvolupament del sistema.
- La percepció dels objectius del sistema canvia en el temps;
- A mesura que creix la mida del problema, creix la complexitat de la tasca d'anàlisi de requeriments;
- El client acostuma a no valorar l'etapa d'anàlisi de requeriments.

En una primera fase es definiran els requeriments a alt nivell, sense entrar en massa detall. Aquestes definicions es faran en llenguatge natural, per tal que puguin ser enteses per tothom. En una segona fase, aquestes definicions es refinen per obtenir un conjunt d'especificacions de requeriments. Aquestes especificacions ja no cal que siguin comprensibles per tothom, ja que van adreçades al personal tècnic.

Tècniques de recollida de requeriments:

- Entrevistes;
- Observació;
- Estudi de documentació;
- Qüestionaris;
- Brainstorming;
- Prototipatge;
- Sistemes semblants.

Tot i que les fonts d'informació per als requeriments de funcionalitat i d'interfície d'usuari són les mateixes, la informació necessària no és la mateixa. Al descriure les funcionalitats es posa èmfasi en allò que ha de fer el programari. Quan es descriuen les interfícies d'usuari, el que interessa més són les accions que farà l'usuari.

El document de requeriments del sistema tindrà una doble funció:

- Farà de *contracte* que descriu com ha de ser l'aplicació. Cal posar tot l'èmfasi en descriure què ha de fer el sistema, deixant per més endavant el com s'ha de construir;

- Base de treball dels analistes.

Els destinataris del document són:

Clients: comproven que el que s'ha escrit s'adiu amb el que ells pretenien obtenir, i proposen canvis si cal.

Gestors: l'utilitzaran per fer els pressupostos i planificar el desenvolupament

Desenvolupadors: l'usen per entendre i conèixer el sistema que han de desenvolupar

Encarregats de les proves: dissenyen les proves de validació en funció dels requeriments.

Encarregats del manteniment: prendre les consideracions oportunes per facilitar el futur manteniment.

Caldrà validar i tenir en compte la futura evolució del document de definició de requeriments:

- **Validació**. Abans de donar per finalitzat el document de requeriments del sistema cal valorar-ne la seva qualitat a dos nivells:

  - Macroscòpic: comprovar si l'especificació és completa, consistent i precisa;

  - Detallat: precisar termes concrets.

- **Evolució**. No és realista pensar que, en projectes relativament llargs, els requeriments que s'havien especificat inicialment es mantindran invariables fins al final del projecte. Cal, per tant, preveure aquests possibles canvis i organitzar-se de forma que la documentació que es genera sobre l'especificació de requeriments sigui fàcilment modificable. En cas que no ho sigui, els canvis s'incorporaran directament en el disseny, i la documentació dels requeriments quedarà obsoleta i, per tant, sense utilitat.

DOCUMENTACIÓ DELS REQUERIMENTS DEL SISTEMA

Caldrà:

- identificar (a nivell d'usuari del sistema):
  - guions;
  - actors;
  - activitats;
  - casos d'us;
  - diagrames de casos d'us;
  - diagrames d'activitat;
  - glossari.
- analitzar i especificar les interfícies d'usuari que haurà de tenir el sistema. Es poden construir els camins de navegació amb l'ajuda dels diagrames d'estat.

Els guions

Difícilment els usuaris identifiquen els casos d'ús d'una manera explícita i sistemàtica. Generalment descriuen les operacions - *guions* - més freqüents que necessiten fer en el seu treball. Els guions expressen les necessitats dels usuaris tal com les veuen ells. Cada guió ha de descriure de manera precisa i completa la interacció entre l'usuari i el programari: precondicions, casos particulars, excepcions, ... Els guions són sessions que un actor duu a terme amb relació al programari i serveixen per extreure'n el casos d'ús.

El Glossari

És una llista de termes rellevants que s'extreu del model del domini i dels guions. És útil per a unificar terminologia.

Com identificar actors

- Quin usuari està interessat en un requeriment concret ?
- A quins dominis de l'organització s'usarà el sistema ?
- Qui serà beneficiari de la nova funcionalitat ?
- Qui proveirà, usarà i/o retirarà informació ?
- Qui donarà suport i administrarà el sistema ?
- El sistema, farà servir un recurs extern ?
- Un usuari actuarà amb diferents rols ?
- Diferents usuaris actuaran amb un mateix rol ?
- Interaccionarà el nou sistema amb un sistema antic ?

Com identificar casos d'ús

- Quines són les tasques i responsabilitats de cada actor ?
- Algun actor crearà, emmagatzemarà, canviarà, esborrarà o llegirà informació del sistema ?
- Quins casos d'ús crearan, emmagatzemaran, canviaran, esborraran o llegiran aquesta informació ?
- És necessari que un actor informi al sistema sobre canvis externs ?
- És necessari que un actor sigui informat sobre certes incidències del sistema ?
- Quins casos d'ús donaran suport i mantindran el sistema ?
- Poden ser realitzats pels casos d'ús tots els requeriments funcionals documentats ?

Observacions sobre els casos d'ús

- Els casos d'ús descriuen activitats visibles per l'usuari i, per tant, no descriuen moltes de les activitats (amagades) del sistema;
- Els casos d'ús s'escriuen: els detalls de cada cas d'ús s'han de reflectir a la corresponent fitxa d'especificació del cas d'ús;
- Els casos d'ús han de ser tan senzills com sigui possible i tant sols han de reflectir els requeriments del sistema des del punt de vista de l'usuari.
- Els diagrames d'activitat ajuden a descriure els casos d'ús;
- És útil fer un diagrama de casos d'ús que fixi el context del sistema a desenvolupar;
- És convenient descompondre els casos d'ús complexos en diversos escenaris (instàncies del cas d'ús);
- Un error comú és el fet de preocupar-se més del necessari pel dibuix dels diagrames de cas d'ús;
- Només si és necessari, es poden refinar alguns diagrames de cas d'ús amb nous diagrames de casos d'ús que mostrin relacions de generalització, inclusió i extensió;
- La descripció d'us escenari convé fer-la amb l'ajuda d'un diagrama de seqüència.

EXEMPLE D'ESTRUCTURA D'UN DOCUMENT DE REQUERIMENTS

A continuació donem un exemple d'estructura de document de requeriments fent servir orientació a objectes:

1. Introducció
2. Sistema actual (descripció textual)
3. Sistema proposat (descripció textual)
    - 3.1 Visió general
    - 3.2 Requeriments funcionals
    - 3.3 Requeriments no funcionals
    - 3.4 Models del sistema (descripció gràfica i textual)
        - 3.4.1 Guions
        - 3.4.2 Model de casos d'us
        - 3.4.3 Diagrames d'activitat
        - 3.4.4 Interfícies d'usuari
4. Glossari

ANÀLISI

EL MODEL DEL DOMINI

El **modelat del domini** és una representació del mon real que, mitjançant diagrames de classe, mostra:

- Les classes (conceptuals) del sistema a construir;
- Els atributs de les classes;
- Les associacions entre classes;
- Les relacions de generalització/especialització entre classes.

Un bon model del domini ha de capturar, en el context dels requeriments del problema a resoldre, la informació esencial necessària per a entendre el domini.

Classes conceptuals

Una **classe conceptual** és una idea, lloc, cosa o persona important dins del sistema d'informació que volem construir. La identificació de les classes conceptuals és la part central del procés de modelat del domini. Durant el modelat del domini no es modelen les classes disenyades des de la perspectiva de la implementació final, per tant no cal preocupar-se d'operacions de classe, bases de dades, ...

Per a identificar les classes conceptuals es poden seguir diverses estratègies:

1. Crear, a partir de les fitxes dels casos d'ús, una llista per categories de les classes conceptuals candidates. Algunes de les categories que s'acostumen a considerar són: objectes tangibles, noms abstractes, especificació de conceptes, descripció de coses, sistemes informàtics o electro-mecànics, llocs, organitzacions, transaccions, esdeveniments, rols de persones, regles i normes, contenidors de coses, catàlegs, coses dins d'un contenidor, manuals, llibres, registres financers, treballs, contractes, qüestions legals, instruments finacers, serveis, ...

2. Identificar els substantius que apareixen a les frases de les fitxes dels casos d'ús.

   Una fablesa d'aquesta estrategia és la imprecisió del llenguatge natural: diferents substantius poden representar la mateixa classe conceptual o atribut.

3. Fer servir patrons d'anàlisi documentats, que són models de domini parcials referits a diferents àmbits d'aplicació creats per experts.

A la fase inicial d'identificació acostumen a quedar sense determinar algunes classes conceptuals que s'identifiquen posteriorment, mentre s'estant buscant atributs i associacions o fins i tot durant la fase de disseny, i finalment s'afegeixen al model.

Els atributs de les classes

Un atribut emmagatzema informació d'un objecte d'una classe conceptual. Un cop completada la llista de classes conceptuals cal determinar-ne els seus atributs. Caldrà incloure en el model del domini els atributs que, a partir dels requeriments recollits en les fitxes dels casos d'ús, es dedueixi que seran necessaris per a emmagatzemar informació. Una regla que cal tenir en compte és que un adjectiu generalment acostuma a representar un atribut de classe. Un error comú és agafar com a atribut un concepte que hauria de correspondre a una classe. En cas de dubte és millor definir una nova classe en comptes d'un atribut.

Les associacions entre classes

Les associacions descriuen relacions entre classes. Una associació indica que els objectes d'una classe necessiten accedir a informació dels objectes de l'altra classe. Les associacions poden ser bidireccionals. Hi pot haver més d'una associació entre dues classes. Cadascun dels dos extrems d'una associació s'anomena rol. Els rols opcionalment poden tenir: nom, multiplicitat i sentit de navegació.

Una vegada identicades les classes del model del domini cal determinar-ne les associacions. Les associacions més comunes són del tipus:

- És una part de
- Línia de detall de
- Contingut a
- És membre de
- És decisió de
- És el següent de
- Fa servir
- Es comunica amb
- Es relaciona amb

Les relacions de generalització/especialització entre classes

Les relacions de generalització i especialització s'obtenen a partir de la identificació de les superclasses i subclasses rellevants en el domini del problema. Alguns dels motius que justifiquen la partició d'una classe en subclasses especialitzades són:

- la subclasse té atributs addicionals;
- la subclasse té associacions addicionals;

Convé crear una superclasse generalitzada quan:

- les subclasses potencials representen variacions d'un concepte similar;

- totes les subclasses tenen un conjunt d'atributs comuns que es poden traslladar a la superclasse;

- totes les subclasses tenen una mateixa associació, que es pot traslladar a una associació de la superclasse.

Tot i que les jerarquies representades en el model del domini no tenen perquè quedar necessàriament recollides en el model del disseny, normalmernt són la base de la jerarquia de classes que finalment s'acabarà implementant, i un bon disseny d'aquestes classes permet reduir la duplicació de codi i la generació de software de qualitat.

Les classes d'anàlisi

Inicialment el diagrama de classes d'anàlisi coincideix amb el diagrama del model del domini. Al anar dibuixant els diagrames de col·laboració simplificats dels diversos casos d'ús caldrà anar fent modificacions del diagrama de classes del domini del problema, que d'aquesta forma s'anirà transformant en el diagrama de classes d'anàlisi.

A la fase d'anàlisi no cal preocupar-se de com s'emmagatzemaran en una BD els objectes de l'aplicació. L'emmagatzemament a BD es tracta a la fase de Disseny. A la fase d'anàlisi s'assumeix que tots els objectes estan en memòria. Aquest fet obliga a que per tal de poder accedir a multiobjectes, en alguns casos calgui crear una classe *catàleg* d'objectes. En Java la classe catàleg tindrà un atribut que serà una llista d'objectes del tipus del multiobjecte a accedir. A la fase de disseny la classe catàleg ens transforma de forma convenient per tal que es pugui accedir a la corresponent BD.

ANÀLISI DEL COMPORTAMENT

Diagrames de col·laboració simplificats

Els diagrames de col·laboració que es dibuixen durant aquesta fase s'anomenen "simplificats", entre d'altres motius, perquè:

- Les operacions no estan completament definides. No cal posar: paràmetres, valors de retorn, ...

- No s'han reassignat, suposant que calgui fer-ho, les operacions de les classes de control a les classes d'entitat i de frontera.

- No es preveu l'accés a bases de dades.

Paquets de classes d'anàlisi

Per tal de facilitar l'organització del model del domini les classes d'anàlisi s'agrupen en paquets. Les classes d'un paquet generalment participen en un mateix grup de casos d'ús. Convé dissenyar els paquets de forma que es maximitzi la interrelació entre les classes d'un mateix paquet i es minimitzin les dependències entre paquets. Els paquets i les seves dependències es mostren per mitjà de ***diagrames de paquets.***

DISSENY ORIENTAT A OBJECTES

Durant l'anàlisi s'han determinat les classes des del punt de vista del problema a resoldre. A la fase de disseny l'especificació de les classes es completa de forma que siguin directament implementables.

El diagrama de classes de disseny s'obté, bàsicament, transformant el diagrama de classes d'anàlisi. Durant el procés de transformació:

- una classe d'analisi es pot transformar en cap, una o vàries classes de disseny;
- cal completar els atributs de les classes i les seves descripcions: tipus, visibiltat i opcionalemt valors per defetcte.
- s'han d'especificar, amb el màxim detall, totes les operacions de les classes amb: paràmetres, tipus de retorn i visibiltat;
- cal refinar les associacions del diagrama de classes d'anàlisi: determinació de relacions de composició, agregació, ... ;
- les classes de disseny incorporen elements que estan condicionats per la solució: llibreries d'interfícies, bases de dades a utilitzar, ...

Interacció i col·laboració entre objectes

Un objecte "posseeix" informació, emmagatzemada als atributs, i "té" comportament, definit per mitjà d'operacions/mètodes. Un objecte no coneix directament la informació emmagatzemada per un altre objecte, però la pot obtenir per mitjà d'un intercanvi de missatges amb aquest altre objecte. Els objectes col·laboren enviant-se missatges, és a dir fent crides a operacions de l'altre objecte.

Responsabilitats

Una responsabilitat és una obligació d'una classe en quant al seu comportament. Una responsabilitat reflecteix la informació disponible per una classe, tant si està emmagatzemada als seus atributs com si es pot obtenir per mitjà de col·laboracions amb altres classes.

Cal identificar quines són les responsabilitats de cada classe i determinar quines són les altres classes amb les quals ha de col·laborar per a complir amb les seves responsabilitats.

Sovint les responsabilitats comencen estant determinades de forma vaga i no es van definint clarament fins que se avança en la fase disseny.

Bàsicament les responsabilitat són de dos tipus:

- De *coneixement* (Knowing):
  - de dades privades
  - d'objectes relacionats
  - de resultats que es poden calcular o derivar
- D'*acció* (Doing):
  - Realitzada pel mateix objecte
  - Invocar, controlar i coordinar activitats en altres objectes

Una responsabilitat no és el mateix que una operació de classe, però les operacions s'implementen per tal que les classes puguin complir amb les seves responsabilitats.

Les responsabilitats s'assignen als objectes d'una classe a la fase de disseny per mitjà de diagrames d'interacció. A l'hora de construir els diagrames d'interacció el més important és distribuir les operacions entre les classes.

ESPECIFICACIÓ D'INTERFICIES D'USUARI

Per a cada actor, cal identificar la interficie requerida en termes de:

- un esboç de la pantalla, d'acord amb les preferències de l'usuari;
- la descripció dels possibles diàlegs;
- les dades a entrar per l'usuari amb el seu format (enter, decimal, string , ...);
- les dades de sortida i el seu format;
- el format dels llistats amb el format de les dades.

DISSENY DE SUBSISTEMES

A la fase de disseny, els paquets d'anàlisi es descomposen en un o diversos subsistemes que es fan servir per a separar les diferents components del sistema. A més a més s'introdueixin nous subsistemes relatius a: bases de dades, comunicacions, ...

Cal asegurar-se que cada subsistema:

- és altament independent d'altres subsistemes i de les seves interficies;
- proporciona les interficies apropiades;
- ofereix les operacions suportades per les interficies de forma correcta.

MODEL DEL DESPLEGAMENT

El model del desplegament descriu la distribució física del sistema en construcció, en termes de la forma com les funcionalitats es distribueixen en els diferents nodes. Es modela per mitjà de *diagrames de components* i *diagrames de desplegament.*