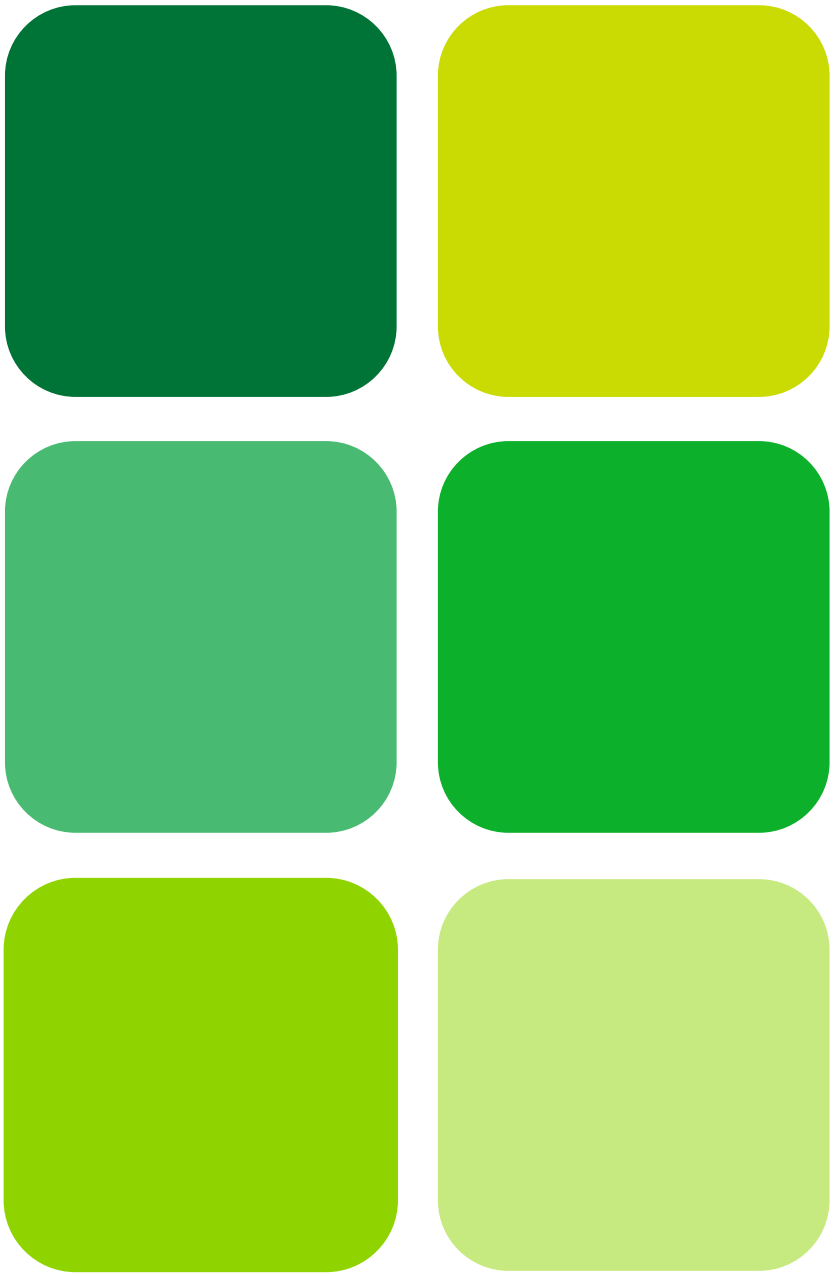


www.sapientia.uji.es | 120

UNIVERSITAT JAUME·I

Fonaments d'enginyeria del programari

Cristina Campos Sancho
Reyes Grangel Seguer
Victòria Nebot Romero

Fonaments d'enginyeria del programari

Cristina Campos Sancho
Reyes Grangel Seguer
Victòria Nebot Romero

Departament de Llenguatges i Sistemes Informàtics

Codi d'assignatura EI1023

Fonaments d'enginyeria del programari - UJI - DOI: http://dx.doi.org/10.6035/Sapientia120

Edita: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions
Campus del Riu Sec. Edifici Rectorat i Serveis Centrals. 12071 Castelló de la Plana
http://www.tenda.uji.es e-mail: publicacions@uji.es

Col·lecció Sapientia 120
www.sapientia.uji.es
Primera edició, 2017

ISBN: 978-84-16356-94-2

Publicacions de la Universitat Jaume I és una editorial membre de l'UNE, cosa que en garanteix la difusió de les obres en els àmbits nacional i internacional. www.une.es

Aquest llibre, de contingut científic, ha estat avaluat per persones expertes externes a la Universitat Jaume I, mitjançant el mètode denominat revisió per iguals, doble cec.

ÍNDEX

Pròleg

El terme enginyeria del programari es va introduir per primera vegada al segle passat, a finals dels anys 60, en una conferència realitzada per discutir el que es va denominar «la crisi del programari». Aquesta crisi ha sigut un tema recurrent en les últimes dècades del segle XX i encara ara, en el segle XXI, molts projectes i productes de programari es desenvolupen de manera incorrecta o provoquen errors per causes semblants a les identificades durant aquesta crisi. Per a millorar els resultats i evitar els errors es va justificar la necessitat de tenir una disciplina que guie els desenvolupadors de productes de programari en el procés que es du a terme amb l'objectiu d'aconseguir productes d'acord a uns estàndards i paràmetres de qualitat avaluables.

S'ha eixit de la crisi? A l'àmbit del desenvolupament de programari s'ha fet evident, i ja no és discutible, la necessitat d'usar i aplicar mètodes formals que donen suport al desenvolupament de productes d'alta qualitat, que no tinguen ni els provoquen en la seua execució, i a més a més, que es desenvolupen amb el temps i el cost adients a les necessitats per a les quals es desenvolupa el producte.

Però igual que les tecnologies canvien i evolucionen ràpidament incorporant nous avanços que són esperats pels usuaris i clients de productes de programari, cada vegada més experts, l'enginyeria de programari ha d'evolucionar. Els mètodes, tècniques i eines han de donar resposta per una part, als enginyers informàtics, perquè el seu treball siga més dinàmic i col·laboratiu amb els seus clients i, per una altra part, als clients o usuaris, que tenen capacitat i predisposició a participar de manera més activa, perquè el resultat siga el que ells esperen incorporant tot allò que les noves tecnologies al seu abast els proporcionen. En els últims anys els mètodes denominats àgils ajuden a proporcionar alternatives a mètodes tradicionals, però incloent-hi tècniques que asseguren que el producte final i el procés tinguen la qualitat que es requereix.

Aquesta obra recopila els resultats i l'experiència d'un professorat que porta molts anys impartint, aprenent i renovant els continguts i la pràctica en la docència d'enginyeria del programari, amb el repte de transmetre a l'alumnat la necessitat d'usar mètodes de treball que impliquen el fet de documentar, pensar abans de programar i revisar el que es fa més enllà de crear un codi i un programari.

A banda del primer capítol d'introducció dels conceptes d'enginyeria del programari, la resta de capítols estan ordenats de la mateixa manera que es durien a terme les fases de desenvolupament d'un projecte de programari que segueix un ordre seqüencial.

Cal destacar que per a completar el text s'inclou un exemple pràctic desenvolupat seguint els passos descrits de manera similar a com es duria a terme en el treball de pràctiques de l'assignatura.

L'assignatura inclou com a resultat de l'aprenentatge «Redactar en llengua anglesa documents tècnics d'enginyeria del programari», per això s'inclouen cites, definicions i alguns termes en llengua anglesa. En l'apartat del cas pràctic, que es proporciona com a exemple, s'annexa l'índex de continguts de la solució proposada en anglès.

CAPÍTOL 1

Introducció a l'enginyeria del programari

Software is embedded in systems of all kinds. As software importance has grown the software engineers continually attempt to develop technologies that will make it easier, faster and less expensive to build and maintain high quality computer programs.

(Pressman, 2010)

Objectius

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Comprendre el concepte d'enginyeria del programari, els seus orígens, com s'han establert les bases teòriques d'aquesta matèria i conèixer i reflexionar sobre algunes de les principals definicions proposades.
- Comprendre el que és el programari, les seues característiques i com forma part del sistemes informàtics. Conèixer diferents tipus definits de programari.
- Saber per què és necessari conèixer i utilitzar tècniques, procediments i ferramentes que donen suport a totes les activitats relacionades amb l'enginyeria del programari.
- Entendre que un sistema informàtic no és un objecte aïllat i que influeix en el funcionament de l'entorn on s'implanta, en la seua gestió i en les persones que l'utilitzen.

1. Introducció

En aquest primer tema del material de l'assignatura de Fonaments d'Enginyeria del Programari es pretén donar la base teòrica del que és l'enginyeria del programari i introduir l'alumnat en els conceptes bàsics de producte i procés de programari que es tractaran durant el curs. L'assignatura és una introducció a tots els continguts de la matèria, per tant, es tracten tots però sense aprofundir. En altres assignatures del grau es tractaran cadascun dels temes del curs amb més detall.

2. Enginyeria del programari

L'enginyeria del programari sorgeix com una disciplina al voltant de la creació del programari per a proporcionar ajuda a les persones que treballen en el seu desenvolupament, millorar la seua qualitat i permetre que el maquinari i el programari funcionen en concordança.

Per comprendre aquesta disciplina s'analitzen en aquest apartat diferents definicions i conceptes creats al seu voltant, incloent-hi aspectes relacionats amb el producte d'interès per a l'Enginyeria del Programari, i quines són les condicions que ha de complir un producte per ser correcte en aquest àmbit.

Des del punt de vista filològic i des del punt de vista tècnic, pot dir-se que l'enginyeria és una disciplina que pretén proporcionar mètodes robustos, tècniques adequades i ferramentes eficients per a crear solucions reals a problemes de l'àmbit en què es considere aquesta enginyeria. I per solucions reals, es considerarà aquelles que siguen factibles, és a dir, que puguen desenvolupar-se amb els recursos de què es disposa en un termini temporal acceptable.

Des de fa diverses dècades s'han estudiat i desenvolupat procediments, mètodes i tècniques basats en els principis generals de l'enginyeria i s'han traslladat i adaptat a l'àmbit del desenvolupament de programari i de sistemes informàtics. Partint d'aquesta idea, s'han donat diferents definicions d'Enginyeria del Programari. Dues de les definicions més acceptades per la comunitat de persones que es dedica al desenvolupament del programari van ser les proposades per Fritz Bauer[1] i per Ian Sommerville:[2]

> «*The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines*.»
>
> «L'establiment i ús de principis d'enginyeria robustos, orientats a obtenir programari econòmic que siga fiable i funcione de manera eficient sobre màquines reals.» (Bauer, 1969).
>
> «Disciplina de l'enginyeria que comprèn (inclou) tots aquells aspectes de la producció de programari des de les etapes inicials de l'especificació del sistema, fins el manteniment d'aquest després de la seua utilització.» (Sommerville, 2005).

1. Un dels primers investigadors en els temes del desenvolupament del programari.
2. Professor d'Enginyeria del Programari a Lancaster University i autor de nombrosos llibres sobre enginyeria del programari.

En general totes les definicions reforcen la necessitat d'una disciplina d'enginyeria per al desenvolupament del programari. La problemàtica del desenvolupament de grans sistemes és comparable amb els problemes que sorgeixen en qualsevol gran projecte d'enginyeria: ús de mètodes adequats per a desenvolupar el producte, control de costos, compliment de terminis establerts, gestió de personal i tasques, selecció de ferramentes, control de qualitat, etc.

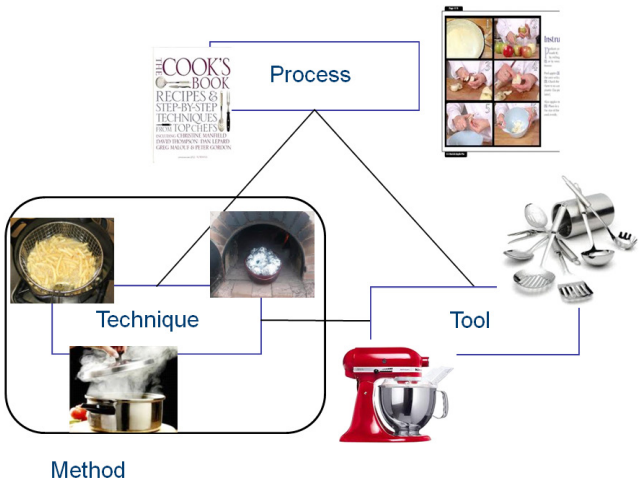


Figura 1.1. Procés, tècnica i eina

Com a conclusió a totes aquestes definicions es pot considerar que l'Enginyeria del Programari inclou (Pressman, 2010) (vegeu figura 1.1):

- **Mètodes i Tècniques (*Methods and Tecniques*)**: indiquen «com» s'ha de construir tècnicament el programari. Els mètodes i les tècniques inclouen normalment una sèrie de notacions, gràfics, normes i criteris per dur a terme alguna tasca relacionada amb la planificació o el desenvolupament de programari. Un mètode indica la manera d'efectuar un procés o conjunt d'activitats. Les tècniques donen suport a la realització d'una activitat. Per a la realització d'una activitat o procés es pot fer ús de diferents tècniques.
- **Eines (*Tools*)**: subministren un suport automàtic o semiautomàtic per a desenvolupar els mètodes. Les ferramentes més completes són les denominades ferramentes CASE (Computer-Aided Software Engineering/Enginyeria del Programari Assistida per Computador). Aquestes combinen maquinari, programari i bases de dades que contenen la informació sobre l'anàlisi, el disseny, la codificació i la prova per crear un entorn anàleg al que es gasta en altres disciplines de l'enginyeria com els productes CAD (Computer Aided Design/Disseny Assistit per Computador).
- **Procés (*Process*)**: és la unió entre els mètodes i les ferramentes, defineix la seqüència en què s'apliquen els mètodes, els documents que s'han de produir, els controls de qualitat i les directrius per desenvolupar el producte. A l'apartat 4 d'aquest tema s'estudia aquest concepte amb més profunditat.

L'objectiu dels enginyers és obtenir un mètode senzill per a la resolució de problemes complexos. Per a aconseguir aquest objectiu els enginyers han d'aplicar teories, mètodes i ferramentes seleccionant aquelles que consideren més convenients. A més, els enginyers han de buscar solucions de qualitat tenint en compte restriccions financeres i d'organització (temps i recursos) (vegeu figura 1.2).

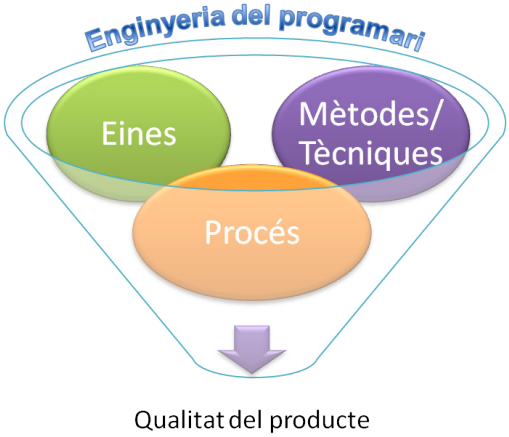


Figura 1.2. Mètodes, eines i procés en enginyeria del programari

«*Software engineering encompasses a process, a collection of methods and an array of tools that allow professionals to build high quality computer software.*» (Pressman, 2010)

3. El producte: programari i sistema informàtic

Des del punt de vista d'un enginyer de programari, el producte a desenvolupar és un conjunt de programes que s'executen de manera ordenada i processen continguts, normalment dades, proporcionant uns resultats determinats.

El programari és un producte i al mateix temps un mitjà per a gestionar i proveir altres productes. Per desenvolupar programari de qualitat cal entendre què és i què fa que un programa, com a part del programari, i tot el programari en si, siga millor que un altre. És a dir, quines són les característiques del producte i quines són les condicions que cal tenir en compte per desenvolupar un producte de programari de qualitat.

3.1. Característiques del programari

Per a justificar la necessitat de tècniques i mètodes adients es compara habitualment amb la construcció d'altres productes d'enginyeria, però es cert, que hi ha molts aspectes diferencials i característics del programari com a producte que justifiquen que la necessitat de tècniques i mètodes específics en aquest àmbit. Aquestes característiques específiques són (Pressmann, 2010):

- El programari es desenvolupa mentre que el maquinari es construeix i fabrica. *It is developed or engineered not manufactured*.

- No es trenca ni desgasta (*it doesn't wear out*). Les fallades es produeixen per omissions o per errors inadvertits durant la fase del seu desenvolupament, no existeixen en general peces de recanvi.[3]
- El programari és un producte lògic, no físic, necessita el suport de dispositius i altres elements per al seu funcionament.
- El producte final no és tangible per al seu destinatari, menys la part visual o física que proporcionen les interfícies. No és senzill d'imaginar per als destinataris del seu ús.
- El programari no pot funcionar de manera aïllada, necessita d'altres components que el complementen per a formar un sistema.

Tenint en compte aquestes característiques quan es considera el programari com a producte, es visualitza el seu ús i la visió de les interfícies, o dispositius que acciona, que constitueixen la part visible per a les persones que l'usen. El programari funciona quan és part d'un conjunt d'elements que donen suport i resposta a la seua execució. Aquest conjunt d'elements és el que en l'àmbit de l'enginyeria del programari es defineix com sistema informàtic (*Computer System*, en anglès).

Els sistemes informàtics poden estar involucrats en el processament de la informació (*Computer information systems*) o poden donar suport a la producció i processos industrials, controlar dispositius, etc. Pressman defineix **Computer System** o sistema basat en computadora com (Pressman, 2010):

> «Conjunt o disposició d'elements, que inclouen maquinari i programari i que estan organitzats per portar a terme un objectiu predefinit processant dades i informació.»

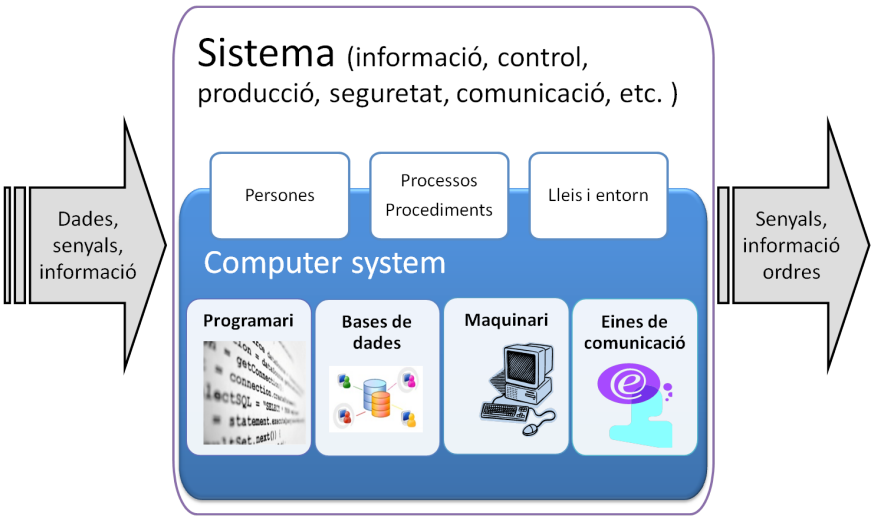


Figura 1.3. Components d'un sistema informàtic

En la figura 1.3 es mostra quins són els components habituals d'un sistema informàtic que pot tenir com a objectiu el processament d'informació o el control de processos de producció o bé donar suport a les comunicacions, entre d'altres. Aquest tipus de sistemes han de processar dades, senyals, imatges, etc. i convertir-les en informació, respostes, ordres o senyals de qualsevol tipus. Hi inclouen

3. L'ús de biblioteques i la reutilització del programari són alguns dels objectius actuals de la construcció del programari.

una part informatitzada que es denomina sistema informàtic (en anglès *computer system*). Es consideren elements bàsics d'aquest sistema les bases de dades o magatzems d'informació, el programari que executa la funcionalitat, el maquinari on estan implantats el programari i els magatzems de dades i per últim les comunicacions, que representen totes les eines que permeten que el sistema es comunique amb altres sistemes, dispositius o persones.

Es consideren part del sistema: les persones per a les quals és útil, els procediments que defineixen com s'ha d'usar, les lleis i les normatives que regulen l'execució del processos i l'ús de les dades i el programari. I tots en conjunt condicionen el disseny, desenvolupament i ús del sistema informàtic final.

Per tant, per a desenvolupar programari de qualitat, cal tenir en compte que no és un producte aïllat, que és necessari conèixer i prendre en consideració tots els altres components del sistema del que forma part a l'hora de decidir i avaluar alternatives de disseny i d'implementació.

3.2. Condicions per a desenvolupar programari de qualitat

Software must be fast to develop, right and cheap.

Com es pot distingir un programari ben realitzat d'un que no ho està? Per a obtenir un bon programari (com a part d'un sistema informàtic) no és únicament necessari que el producte realitze les tasques per a les quals s'ha creat. Els productes d'enginyeria s'han de crear d'acord amb unes condicions relacionades amb el cost, l'eficàcia i la qualitat. Aquestes condicions es reflecteixen en quatre requisits definits per Sommerville (2005), però que són encara característiques fonamentals que el programari de qualsevol sistema ha de complir.

- El programari ha de ser **mantenible**: els sistemes informàtics poden tenir una llarga durada i per tant estan subjectes a canvis (modificacions de lleis, costos, noves tecnologies, etc.). Els sistemes han d'estar documentats i realitzats de forma que possibles canvis puguen fer-se amb els mínims costos possibles. Els sistemes informàtics no són objectes estàtics. Es desenvolupen en un entorn que està subjecte a canvis, o fins i tot en un principi pot ocórrer que l'enginyer no haja arribat a comprendre bé l'entorn.
- El programari ha de ser **fiable**: el programari desenvolupat no ha de tenir errors, per això serà necessari realitzar les comprovacions i proves del programari que siguen necessàries. Per a aconseguir un programari fiable el desenvolupament ha de seguir unes pautes i metodologies correctes i utilitzar tècniques robustes que afavoreixen la construcció d'un sistema informàtic fiable. Els errors comesos durant les fases inicials (identificació de requisits i anàlisi) es veuen incrementats de forma exponencial en les etapes finals.
- El programari ha de ser **eficient**: és a dir, ha de complir les funcions requerides però amb l'ús d'un mínim de recursos del maquinari on s'executa. No

ha de malgastar els recursos del maquinari com per exemple, la memòria o els temps de procés. No obstant, aquesta eficiència pot fer que el programari siga més difícil de mantenir.

- Ha de proporcionar una **interfície apropiada amb l'usuari**: a vegades un programari no s'utilitza de forma completa, ni s'aprofiten tots els recursos i funcions que proporciona a causa únicament de la dificultat que troba l'usuari per a la seua manipulació. Per això és important un bon disseny que tinga en compte els futurs usuaris i facilite al màxim la comunicació de l'usuari amb el programari i la seua utilització.

Es pot dir, per tant, que el principal objectiu d'un enginyer informàtic és el desenvolupament d'un programari útil per a les tasques que l'usuari necessita, senzill de manipular i de construir, flexible per a possibles incorporacions de noves necessitats i viable (tecnològica i econòmicament). El problema és arribar a trobar un equilibri entre tots aquests atributs. Alguns d'aquests s'exclouen entre si, per exemple el cost i l'ús de recursos pot veure's en contraposició amb tenir una interfície tecnològicament més complexa però més simple per als usuaris. Depèn de les característiques del programari i dels usuaris als quals està destinat el fet de considerar un atribut primordial davant dels altres. Per exemple, l'eficiència i fiabilitat, en el cas de programari crític com el que controla avions o equips espacials, són les primeres consideracions que cal tenir en compte, ja que aquest programari ha d'usar pocs recursos, tenir resposta ràpida i per descomptat, no provocar errors. Però, com els seus usuaris estan preparats i formats de manera específica, les interfícies poden ser més bàsiques per a utilitzar menys recursos en l'execució. En sistemes dedicats a usuaris amb poca experiència o sense coneixements informàtics, una interfície senzilla és primordial.

3.3. Classificació del programari

Des de l'inici de la definició de l'enginyeria del programari s'ha intentat establir diferents classificacions del producte objecte d'aquesta disciplina. Però l'evolució de les tecnologies, de les xarxes de comunicacions i el progrés, en quasi tots els aspectes de la vida quotidiana, de dispositius mòbils, automatismes, complica aquesta classificació. En aquest apartat es proporcionen diferents classificacions del programari, però cal tenir en compte que un producte pot tenir característiques de diferents tipus o alguna no definida. Fins i tot la ràpida evolució fa que aquestes classificacions no incloguen els últims avanços, que potser no existeixen quan es defineix la classificació.

Depenent de l'ús o objectiu al qual es destina el programari es pot classificar en:

1. **Programari de sistemes**, desenvolupat per a gestionar altres productes de programari. El programari de sistemes ajuda a executar el maquinari i el sistema informàtic i proporciona a l'usuari i al programador interfícies adequades d'alt nivell, eines i utilitats de suport que permeten el seu manteniment. Es caracteritza per una forta vinculació amb el maquinari, operacions concurrents, ús per múltiples usuaris, gran varietat de mecanismes d'interacció, compartiment de recursos, gestió sofisticada. Els usuaris d'aquest programari habitualment són els professionals dels sistemes informàtics.

2. **Programari per a desenvolupar programari**, que es desenvolupa perquè altres puguen desenvolupar programari. Inclou gestors de bases de dades, editors, depuradors i compiladors de codi, generadors de codi, eines CASE (Computer-Aided Software Engineering), etc. Les característiques són semblants a les del programari de sistemes.
3. **Aplicacions** que són els conjunts de programari que proporcionen solucions per a determinades necessitats de negoci. El seu objectiu abasta un ampli espectre, encara que en general es podria dir que és facilitar i donar suport a totes les activitats humanes, tant des del punt de vista de negoci com quotidià. Permeten automatitzar tasques de gestió dels negocis, donar suport a la pressa de decisions, controlar processos de fabricació o de seguretat, etc.
4. **Programari científic** que és tot el programari, l'objectiu del qual és donar suport a la recerca i desenvolupament d'avanços científics. Es basa habitualment en un processament molt complex, tant des del punt de vista de les dades com dels algorismes i maquinari. S'aplica en àrees com l'astronomia, l'aeronàutica, la medicina, la biotecnologia, nanotecnologia, etc. Actualment inclouen, no només complexos algorismes numèrics, sinó també programari de simulació, productes de disseny assistit per computador (CAD) i programari en temps real que permet analitzar hipòtesis i resultats
5. **Programari encastat** (*embedded*) és aquell que està inclòs en un producte o sistema per a permetre-hi el seu control. Aquest programari pot desenvolupar funcions limitades i simples o altres de sondroll més complexes. Exemples són els dispositius de control d'electrodomèstics o els sistemes de control informatitzats dels vehicles.
6. **Software d'usuari o d'oficina** (*product-line software*), fa referència a programari que es distribueix de manera massiva, es pot distribuir a molts usuaris, i que té funcionalitat específica i limitada com són, per exemple, els fulls de càlcul o els processadors de textos.
7. **Aplicacions en tecnologies web** que inclouen un ampli ventall de productes que es caracteritzen per poder ser usats en la web. A més a més, a mida que les tecnologies Web 2.0 emergeixen, aquestes aplicacions són cada vegada més complexes i sofisticades, i inclouen continguts destinats a l'usuari final, funcions informàtiques, sofisticades bases de dades, etc.
8. **Programari d'intel·ligència artificial** és aquell que fa ús de complicats algorismes no numèrics, els productes dels quals són capaços de realitzar funcions complexes i autoaprendre del seu entorn i resultats. Exemples d'aquest programari es poden trobar en àrees com la robòtica, reconeixement de patrons (de veu o imatges), xarxes neuronals, teoria de jocs, etc.

Depenent de les característiques de la seua distribució, desenvolupament i portabilitat, es pot classificar en:

1. **Programari de fabricació estàndard** per a ús general, que desenvolupen les empreses productores de programari i que pot ser de qualsevol dels tipus descrits abans. En aquest tipus es pot considerar tant productes amb

una finalitat única i ben definida com els fulls de càlcul o processadors de texts, o be aplicacions de gestió empresarial integrada (*Enterprise Resources Planning ERP*), que en alguns casos requereixen un gran esforç d'adaptació i tenen un cost elevat.

2. ***Legacy software***, es denomina així el programari desenvolupat a mida fa dècades però que encara està en ús. Quan les necessitats d'informatització dels processos de negoci van començar, moltes empreses (normalment grans) van invertir importants quantitats de diners i recursos humans en productes de programari a mida. Aquest programari ha sigut actualitzat i modificat contínuament per a incloure nous requisits o adaptar-se a noves plataformes tecnològiques, però el seu nucli és obsolet. Les empreses mantenen els seus sistemes basats en aquest programari perquè gestionen els principals processos de negoci (*core business*) i els resulta complicat i costós, inicialment, substituir-los completament. Actualment encara és un àmbit on desenvolupen la seua activitat professional molts enginyers informàtics.
3. **Programari de codi obert** (*Open source* ≠ gratuït) és el programari que es distribueix amb el codi (obert) amb el qual s'ha desenvolupat. És una tendència creixent, que permet millorar els productes de programari amb l'aportació d'altres.
4. ***Open World Computing***: les comunicacions sense fil i els dispositius mòbils han provocat que els enginyers en informàtica es troben davant un nou repte que és desenvolupar programari que es puga usar des de diferents dispositius i en qualsevol localització.

4. El procés en enginyeria del programari

A partir dels conceptes descrits en els apartats anteriors es pot dir que l'enginyeria del programari és una disciplina que dóna suport al desenvolupament de programari de sistemes informàtics. Aquesta disciplina, proporciona procediments, tècniques i eines per donar suport a totes les activitats que cal portar a terme per a construir un sistema informàtic i, per tant, un programari de qualitat, fiable eficient, mantenible i adient per als usuaris i propòsit al qual està destinat.

Igual que en altres enginyeries, es considera que els mètodes i tècniques s'apliquen seguint uns determinats passos que s'organitzen en un procés. A continuació es descriu quin seria un procés genèric en aquesta disciplina i diferents models o paradigmes de processos més importants en enginyeria del programari. Un bon enginyer de programari ha de conèixer aquests models i saber aplicar-los de manera flexible, canviant o combinant-los, segons les característiques dels seus projectes i l'evolució del treball i resultats.

4.1. Procés genèric

Pressman (2010) defineix un procés genèric de desenvolupament de programari com un projecte que inclou cinc activitats fonamentals (vegeu figura 1.4) i que podrien servir com a punt de partida per a definir les d'un projecte en qualsevol àmbit de l'enginyeria:

- **Comunicació**: abans de començar el treball tècnic és imprescindible comunicar-se i establir una col·laboració entre els enginyers informàtics i tots els agents involucrats al sistema (*stakeholders*[4]) per tal d'establir els objectius principals, recopilar una primera versió de necessitats que permeten identificar les característiques i prioritats del producte a desenvolupar des d'un nivell alt. La comunicació pot ser una activitat molt complexa o molt senzilla (una simple telefonada per a concretar condicions d'un programa) depenent de la grandària del sistema.
- **Planificació**: defineix el pla del projecte que inclou les tasques tècniques que cal dur a terme, els recursos que seran necessaris, els productes i el calendari temporal del projecte.
- **Modelització**: com en qualsevol altra enginyeria és necessari utilitzar models gràfics, esquemes i documentació per a comprendre quin és el producte d'enginyeria que s'ha de construir, quines parts hi haurà i com s'hi acoblaran. El refinament del model i l'ús de models diferents per a representar diferents aspectes del producte permet tenir una comprensió millor del que es desenvoluparà.
- **Construcció**: en aquesta activitat s'inclou la generació de codi i les proves necessàries per a produir un producte final de qualitat, sense errors i que realitze la funcionalitat que ha sigut definida en la modelització.
- **Posada en marxa**: és el lliurament del producte a l'usuari, per a la qual cosa pot ser necessari realitzar tasques com formació, conversió, etc.

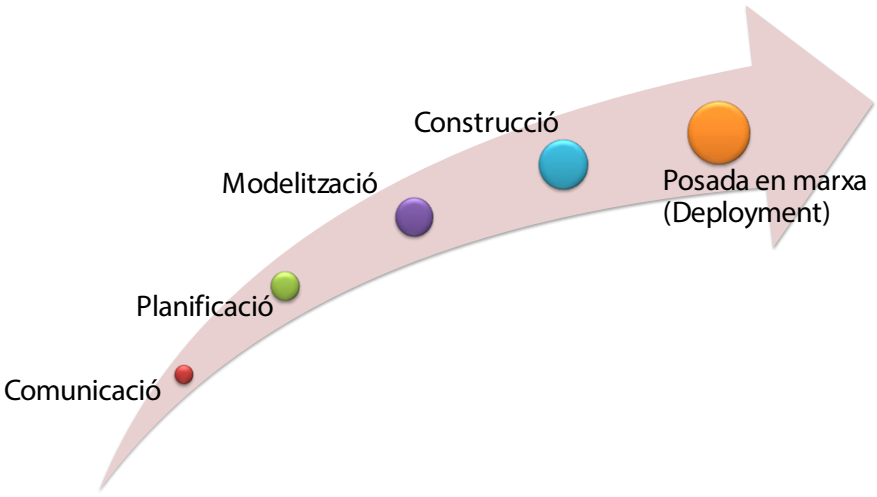


Figura 1.4. Procés genèric segons Pressman (2010)

4. Anyone who benefits in a direct or indirect way from the system which is being developed (Sommerville, 1997).

En aquest model genèric es considera que són necessàries a més a més un conjunt d'activitats de suport que s'han de dur a terme durant el procés de desenvolupament per a controlar el procés en si i assegurar la qualitat del producte final: seguiment i control del projecte, gestió de riscs, assegurament de la qualitat, gestió de configuracions, revisions tècniques, etc. Depenent de la complexitat i la grandària les activitats inclourien tasques més complexes o menys, es podrien agrupar o descomposar en altres més menudes.

Cadascuna d'aquestes fases fonamentals inclou subactivitats i tasques, utilitza diferents tècniques per portar a terme aquestes tasques i involucra persones de diferents nivells i perfils professionals. En aquest curs es pretén donar una visió generalitzada d'aquestes fases i activitats, però s'ha de tenir en compte que han d'adaptar-se de forma adequada a cada projecte particular de desenvolupament d'un sistema informàtic.

Per a organitzar els continguts de l'assignatura i facilitar la docència i l'aprenentatge es defineix, considerant els models estudiats en enginyeria del programari, les fases següents: **Inici i definició de requisits, Planificació**, **Anàlisi** i **Disseny**, ***Construcció i Posada en Marxa***.

En cada tema es detallaran les tasques i activitats a executar en cada fase i algunes de les tècniques que es poden usar. En la figura 1.5 es mostra aquesta organització de temes amb les fases que es descriuen breument a continuació.

4.1.1. Planificació

Com qualsevol altre producte d'enginyeria, abans de començar el desenvolupament de programari s'ha de valorar quant de temps és necessari per a realitzar cadascuna de les tasques i activitats del projecte i planificar-les correctament. És a dir, s'ha d'estimar l'esforç i organitzar les tasques temporalment per saber quan durarà el projecte complet. L'estimació permet considerar les necessitats de recursos econòmics, tecnològics i humans per a desenvolupar el sistema. Una vegada feta aquesta estimació de l'esforç en temps, l'enginyer planifica tot el projecte considerant, entre altres coses, les restriccions temporals i la disponibilitat de recursos.

A més a més, durant la planificació cal tenir en compte altres tasques que controlen la realització correcta i en els terminis establerts de tot el procés, que verifiquen la qualitat del treball i del producte que es vol obtenir, que gestionen el risc i en definitiva, que proporcionen seguretat pel que fa a la construcció del producte en el temps i l'ús de recursos estimats.

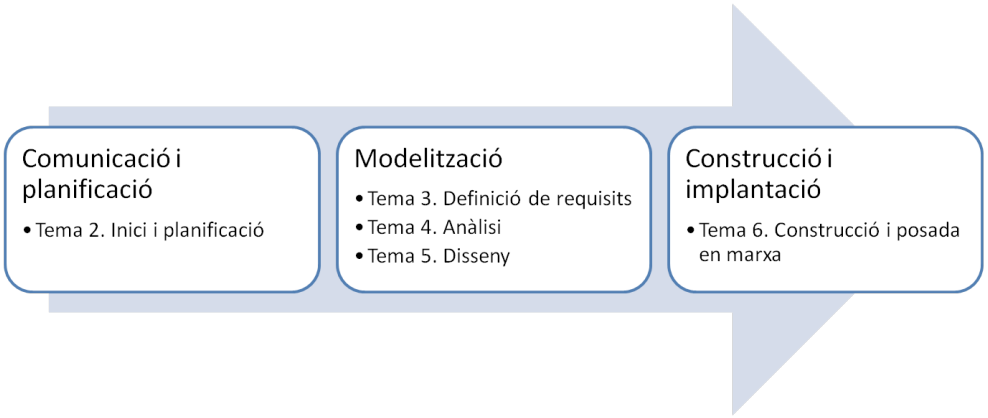


Figura 1.5. Fases del desenvolupament de sistemes informàtics i organització dels temes

4.1.2. Definició de requisits

Qualsevol projecte d'enginyeria comença perquè algú detecta una necessitat o s'ha planificat dintre d'un conjunt d'accions a desenvolupar. Per començar, cal establir quins són els objectius, quin ha de ser-ne el resultat i com s'ha de realitzar, és a dir, quins mètodes, activitats i eines s'usaran.

Dintre d'aquesta primera fase, que pot ser molt senzilla o pot necessitar reunions on participen nombroses persones, es considera la definició de requisits, que és el procés mitjançant el qual els futurs usuaris del sistema informàtic i els enginyers involucrats en el seu desenvolupament investiguen, descobreixen, revelen, especifiquen i comprenen les capacitats i condicions que necessiten per a resoldre un determinat problema o objectiu.

Per arribar a una definició específica i realista, inicialment s'han de definir l'abast i els objectius i s'han d'establir les restriccions que afecten el programari en particular, i el sistema informàtic en general, tenint en compte els recursos disponibles. Per a desenvolupar un programari correcte un enginyer ha de ser capaç d'identificar i definir els requisits que permetran aconseguir els objectius tenint en compte restriccions temporals, econòmiques, tecnològiques i de recursos humans.

Aquesta primera activitat és crítica per a aconseguir amb èxit l'objectiu final que és construir un programari de qualitat. Punts ambigus, necessitats definides vagament o no identificades, poden proporcionar un sistema final pobre.
La definició de requisits és part de l'activitat de comunicació, en aquest cas pot fer-se amb un nivell de detall alt per a establir els principals objectius del sistema i la funcionalitat principal del programari.

4.1.3. Anàlisi del sistema informàtic

L'objectiu de l'activitat d'anàlisi, també denominada a vegades anàlisi de requisits, és convertir el catàleg de requisits (una llista de condicions que el sistema ha d'assolir) en unes especificacions que permeten dissenyar el codi que s'ha de programar per tal que el sistema implemente aquests requisits. Per a ser completes, aquestes especificacions s'han de desenvolupar mitjançant models gràfics i descripcions complementàries.

4.1.4. Disseny del sistema informàtic

El disseny comporta la visió física del programari. S'especifica com i de quina forma es volen introduir les dades que necessita el programari per a funcionar correctament, com es mostra la informació als usuaris i en quins tipus de suports (documents, formularis, pantalles, etc.), quina serà l'arquitectura del programari i la distribució de les diferents capes en les quals s'implantarà, quines seran les tecnologies de comunicacions a usar, com i on s'emmagatzemarà la informació, etc. És a dir, l'anàlisi se centra en **què** és el que el sistema ha de realitzar, mentre que el disseny es preocupa en **com** funcionarà i on estarà.

D'una manera més concreta es pot dir que en aquesta activitat es dissenyaran, d'una banda, aquells components que formen l'aspecte extern del sistema, pantalles, informes i qualsevol altra interfície d'usuari i, d'una altra, la forma física en la qual s'implementarà el programari, els arxius i taules de la base de dades, l'estructura de maquinari i la xarxa de comunicacions, fins a arribar, en alguns casos, al nivell de les especificacions d'implementació.

4.1.5. Construcció i posada en marxa

La construcció i posada en marxa inclou totes les tasques que s'han de dur a terme per a desenvolupar programari correcte i de qualitat, i aquelles necessàries per poder lliurar el producte final a l'usuari de manera que puga ser usat amb èxit.

Per a desenvolupar el programari a més de codificar o programar, serà necessari la instal·lació i prova del maquinari (nou o de desenvolupament), formació de l'equip de treball, el disseny detallat dels programes, la prova individual (preparació de dades de prova), la creació física de la base de dades, etc. Habitualment el maquinari s'adquireix i només requereix activitats d'acoblament, configuració i prova.

Una vegada construït el programari, aquest s'instal·la en el maquinari i es configura físicament la xarxa de comunicacions. No obstant, abans de la posada en marxa definitiva és necessari realitzar una sèrie de tasques prèvies com són la prova integral del sistema, la formació dels usuaris, la preparació de les dades inicials del sistema i la conversió.

4.2. Models del procés

Un procés de desenvolupament de programari proporciona un marc per a definir les activitats, accions i tasques que es requereixen per a desenvolupar un programari de qualitat. L'enginyeria del programari a més a més del procés considera els mètodes i les eines que donen suport al procés. En l'enginyeria del programari s'han definit diferents propostes de procés que es denominen **model** o **paradigma**. En aquest apartat s'estudien algunes de les definicions de models de procés proposats a partir dels estudis i de l'experiència dels enginyers en aquesta disciplina. Entre els models que s'estudien s'inclouen alguns de més tradicionals i d'altres més innovadors i que van tenint una major acceptació entre els desenvolupadors per la seua agilitat i la millor adaptació per a construir productes de programari més actuals.

En qualsevol cas, en el món professional no se segueix un model de manera estricta, sinó que habitualment es defineix un procés que incorpora activitats i tasques de diferents models. Un bon enginyer del programari ha de conèixer aquests models que ha de saber adoptar i adaptar el que puga ser més efectiu considerant les característiques del producte i les demandes del mercat.

4.2.1. El model en cascada

El model en cascada, també denominat el cicle de vida clàssic del programari, és un dels primers que es van proposar i per la seua visió tradicional és qüestionat i considerat obsolet en el desenvolupament dels sistemes actuals. El punt crític d'aquest model és la necessitat de tenir una correcta definició de requisits a l'inici del projecte, la qual cosa a vegades no és possible.

Aquesta visió del model de desenvolupament del programari es basa a seguir un determinat nombre de passos que es realitzen un a continuació de l'altre. Els passos (figura 1.6) principals corresponents són: definició de requisits, anàlisi, disseny, codificació i prova individual, prova global del sistema.

Els principals problemes que presenta el paradigma del cicle de vida clàssic són:

- En els projectes reals els passos rarament segueixen un cicle seqüencial estricte, es produeixen iteracions i sorgeixen problemes en l'aplicació del paradigma.
- És difícil per al futur usuari establir des del principi tots els requisits, i açò provoca dificultats quan s'intenta afegir noves funcionalitats i proporciona alguns punts d'incertesa.
- Els resultats no són visibles fins a les últimes etapes del desenvolupament, i el futur usuari s'impacienta i els errors que es detecten quan el sistema es posa en funcionament poden ser molt difícils de solucionar.

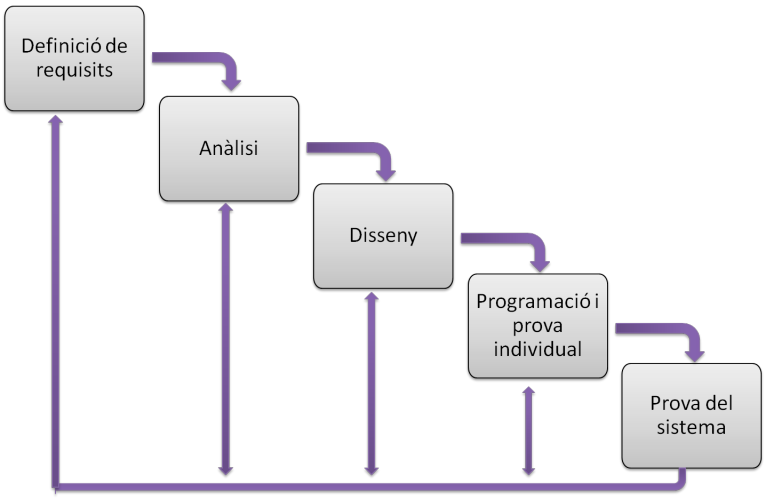


Figura 1.6. Model en cascada (*waterfall*)

Encara que tots aquests problemes són reals, el cicle de vida clàssic proporciona un esquema en el qual es poden emplaçar els mètodes per a l'anàlisi, disseny, codificació, prova i manteniment. Els seus passos són molt semblants als passos genèrics aplicables a tots els paradigmes.

4.2.2. Desenvolupament de prototips

Normalment, la dificultat major per al futur usuari és identificar de forma concreta i completa els requisits del futur sistema. També pot ocórrer que l'enginyer informàtic no estiga segur de l'eficiència d'un determinat algoritme, de l'adaptabilitat d'un determinat entorn, sistema operatiu o de la forma en què la interacció entre l'usuari i el sistema ha de realitzar-se.

El desenvolupament de prototips és un procés que facilita a l'enginyer informàtic i al programador la creació d'una versió experimental del programari que s'ha de desenvolupar. Aquest pot ser:

- Un prototip en paper o un model desenvolupat amb alguna ferramenta sobre PC que represente de forma gràfica o mecànica la interacció home-màquina, d'aquesta forma es facilita al futur usuari la comprensió de tot allò que el sistema li proporcionarà.
- Un model que implemente algunes parts de les funcions requerides o programes, per verificar el correcte funcionament dels algoritmes en particular i la seua adaptació a l'entorn de desenvolupament.
- Un programa existent que execute part de la funcionalitat desitjada però que tinga característiques que hagen de ser millorades.

En la figura 1.7 es poden observar les diferents fases d'aquest paradigma. Com tots els mètodes de desenvolupament de sistemes informàtics comença amb la **recopilació de requisits**; inicialment s'identifiquen tots els requisits coneguts i els

objectius globals, com també les àrees on és necessari desenvolupar una definició més detallada.

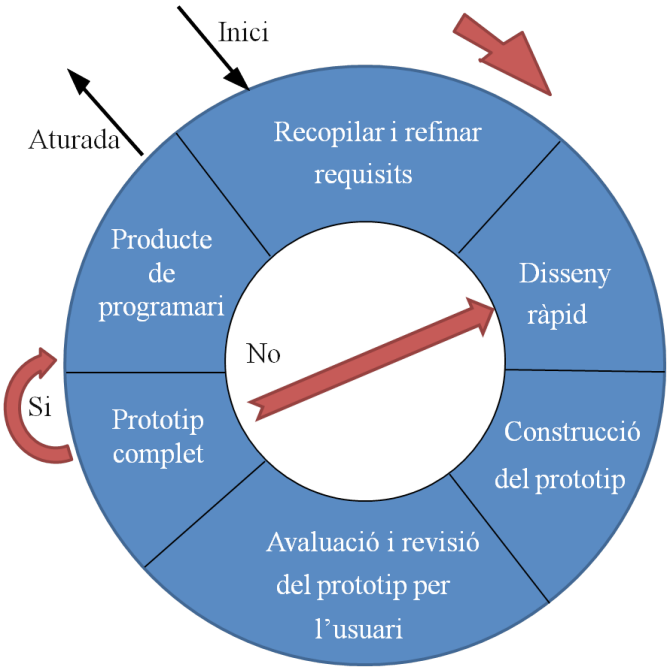


Figura 1.7. Desenvolupament de prototips

Els principals problemes que presenta el desenvolupament de prototips són:

- L'usuari veu en funcionament, en el cas de prototips generats sobre alguna plataforma informatitzada, una primera versió del programari. Cal observar que el prototip s'ha realitzat de forma ràpida i esquemàtica sense tenir en compte que durant el seu desenvolupament no s'han considerat qüestions de qualitat del programari, de fiabilitat o de manteniment a llarg termini. Quan s'informa que el producte ha de ser reconstruït l'usuari sol·licita que s'apliquen millores sobre el producte obtingut amb la idea d'estalviar temps i costos o simplement perquè pareix inútil desfer-se d'una cosa que en principi té un aspecte correcte.
- L'enginyer informàtic o programador que ha desenvolupat el prototip a vegades ha utilitzat sistemes operatius o ferramentes que encara que siguen ràpides en la implementació del prototip, poden no ser les més apropiades per al desenvolupament del sistema final. L'oblit de les raons per les quals s'ha utilitzat una determinada ferramenta fa que una elecció no del tot correcta siga la finalment utilitzada.

La construcció de prototips és un dels paradigmes més efectius si es tenen en compte aquestes consideracions i s'estableix des de l'inici que servirà únicament per a determinar els requisits. Posteriorment ha de ser rebutjat, almenys en part, i ha de desenvolupar-se un sistema mirant tots els aspectes referents a la qualitat, fiabilitat i manteniment del programari.

4.2.3. El model en espiral

El model en espiral es va desenvolupar per a unificar en un model les millors característiques del model del cicle de vida clàssic i del model de prototips, i afegir millores com són l'anàlisi de risc.

Es defineixen quatre activitats fonamentals:

- Planificació: on es determinen els objectius, restriccions i alternatives.
- Anàlisi de risc: per fer una anàlisi d'alternatives mitjançant l'avaluació dels riscos que cadascuna d'aquestes comporta.
- Enginyeria: desenvolupament del producte del següent nivell, o refinaments dels productes obtinguts anteriorment.
- Avaluació de l'usuari o client: que es correspon amb la valoració del producte obtingut en l'activitat anterior per l'usuari final.

Com es pot observar en la figura 1.8, amb cada iteració representada en el gràfic s'obtenen noves versions de programari o producte cada vegada més completes. En les primeres iteracions, en l'activitat d'enginyeria, es poden crear prototips per tal d'analitzar els objectius i avaluar els riscos. A mesura que el cicle es repeteix, en aquesta activitat d'enginyeria es pot seguir el paradigma de prototips o d'altres com el del cicle de vida clàssic. En cada fase de l'anàlisi de risc es planteja la qüestió de seguir o no, bé perquè el producte obtingut siga ja satisfactori o bé perquè els riscos, normalment en costos, no es poden assumir.

Aquest és actualment l'enfocament més realista de l'Enginyeria del Programari sobretot per al desenvolupament de sistemes informàtics a gran escala. Permet tant la utilització de prototips per a avaluar els riscos com la utilització de les fases del cicle de vida del programari d'una forma iterativa.

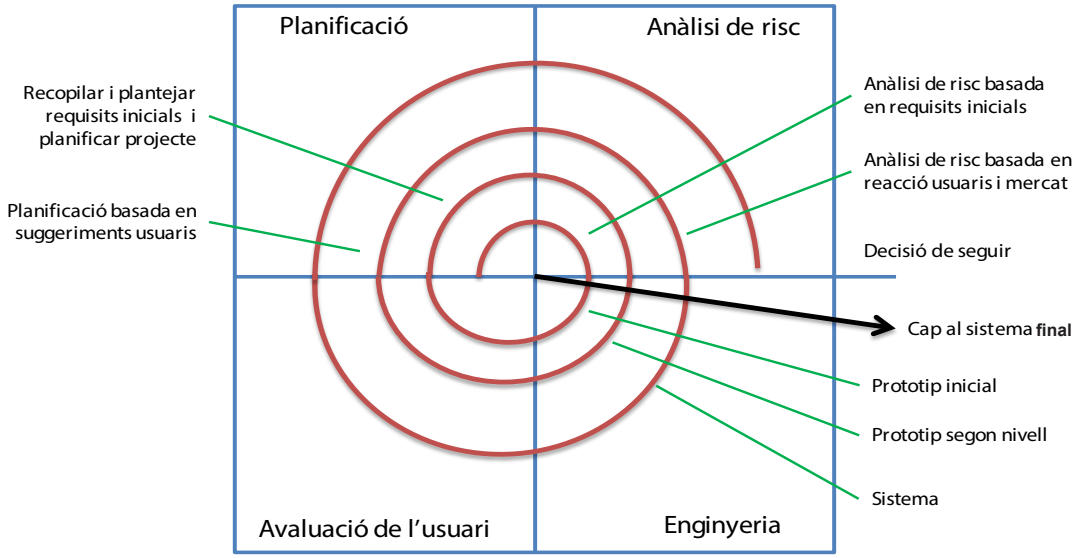


Figura 1.8. Model en espiral

Els principals problemes que presenta el model en espiral són:

- És molt difícil convèncer els futurs usuaris o responsables a nivell de gestió del sistema informàtic que aquest enfocament que evoluciona cap al sistema final és controlable.
- És necessari tenir habilitats i coneixements elevats per realitzar una correcta anàlisi de costos. És fonamental identificar els riscos, ja que si s'ometen després poden ser difícils de solucionar.
- És un model no tan altament utilitzat com els vistos anteriorment.

4.2.3. The Unified Process

UP defineix quatre fases, que corresponen amb les del procés genèric estudiat anteriorment i diferents fluxos de treball i de suport al projecte (vegeu figura 1.9). En cada fase es treballa en diferents fluxos, amb més o menys esforç en cadascun tenint en compte la fase, fins que s'arriba a un resultat satisfactori per a continuar en la següent fase.

L'aplicació del procés unificat va lligada a l'ús del llenguatge unificat de modelització (UML), que proposa diferents diagrames per a representar els artefactes del sistema. En aquesta assignatura es descriurà com usar i com realitzar alguns d'aquests diagrames en les fases que corresponga.

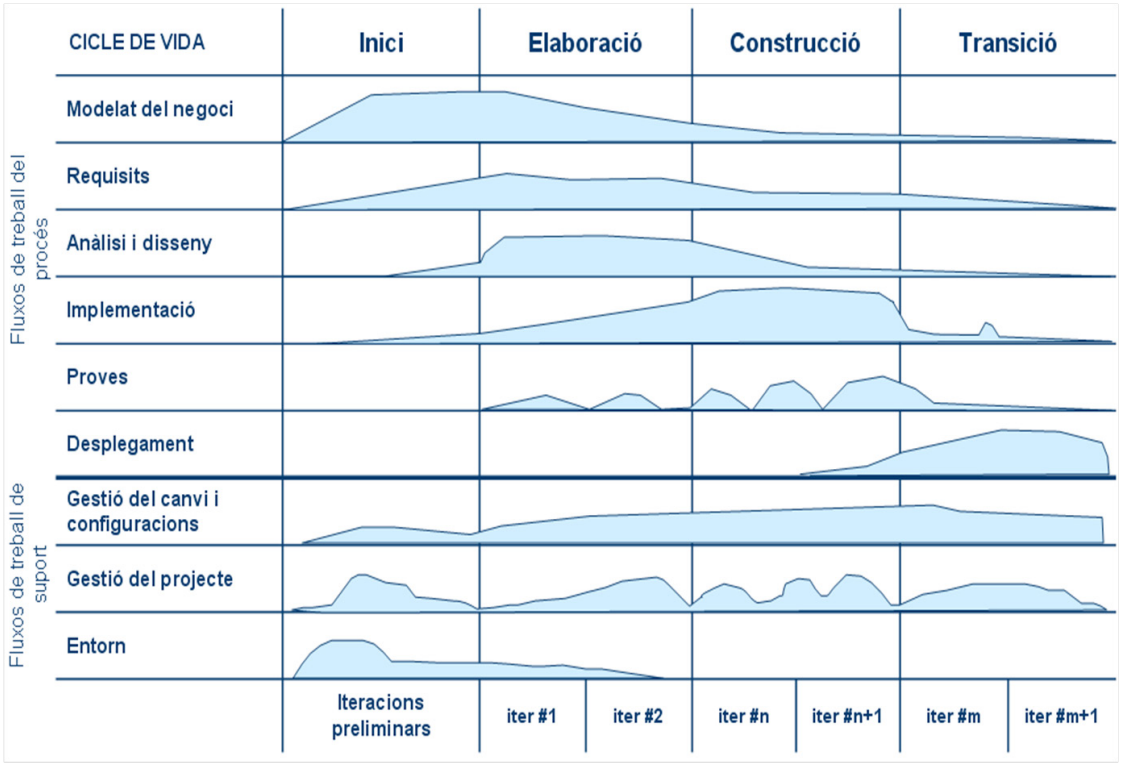


Figura 1.9. El procés unificat de desenvolupament de programari (UP)

4.2.4. Mètodes àgils

L'aparició de les metodologies àgils cal lligar-la al *Manifesto for Agile Software Development* (http://agilemanifesto.org/), com una millora en els processos de desenvolupament de programari que posa en valor:

- Individus i interaccions per damunt dels processos i ferramentes.
- Programari que funciona per damunt d'una extensa documentació
- Col·laboració amb el client per damunt de la negociació d'un contracte.
- Resposta al canvi per damunt del seguiment d'un pla.

La idea subjacent a aquest manifest no és que els elements de la dreta no tinguen valor, que el tenen, sinó que realment aquestes metodologies valoren de forma més significativa els elements de l'esquerra per damunt dels de la dreta.

Els processos de desenvolupament àgil basats en els principis àgils han sorgit com a resposta a la rigidesa d'alguns dels processos més tradicionals, com una alternativa que s'adapta millor a la forma de treballar en els desenvolupaments de projectes de programari actuals, i en entorns de més incertesa pel que fa als requisits. Sobretot donen resposta al perfil dels equips de treball autònoms, col·laboratius i dinàmics, allunyats de les estructures jeràrquiques tradicionals i amb una característica important, la cooperació activa del client en el procés de desenvolupament.

El terme ***àgil*** sorgeix amb l'ànim d'alleugerar el procés de desenvolupament i dotar-lo només de les activitats que suposen un valor afegit per al client. Aquestes guies reforcen el lliurament de producte front a l'anàlisi i disseny, encara que aquestes activitats no són menyspreades, i també propugnen una comunicació activa i contínua entre els usuaris/clients i els desenvolupadors.

Entre aquestes metodologies es pot destacar: eXtreme Programming (XP, 2016), Scrum (Scrum, 2016, Scrum Manager, 2016), Crystal (Version One, 2016). Per poder aplicar aquests processos amb èxit es requereix un equip que treballe de manera col·laborativa, amb respecte i autoorganitzat, de manera que els membres tinguen habilitats, coneixements i experiència adequats. És a dir, cal tenir coneixements, habilitat i experiència en diferents àmbits de l'Enginyeria del Programari, com ara la gestió de projectes i de l'equip humà, llenguatges de modelització, per a representar els diferents components del sistema i la seua evolució mentre es du a terme el projecte, tècniques de validació i verificació per a comprovar l'adequació del producte que es va desenvolupant, tècniques de programació que faciliten la incorporació de millores i l'evolució dels programari cap al producte final, etc.

En la figura 1.10 es mostra gràficament el procés que es du a terme quan es desenvolupa el producte de programari seguint la metodologia Scrum proposat en Vlaanderen (2010). Els requisits s'agrupen en una pila denominada *Product Backlog*; el responsable del projecte selecciona un subconjunt de requisits i els agrupa en el que es denomina *Sprint Backlog,* de manera que un *sprint* siga implementable en un període estipulat d'entre 2 i 6 setmanes; diàriament l'equip es

reuneix per a assignar tasques per implementar aquest *sprint*, o revisar l'avanç i actualitzar-lo; cada *sprint* finalitzat produeix un increment del producte que és avaluable pel propietari del producte.

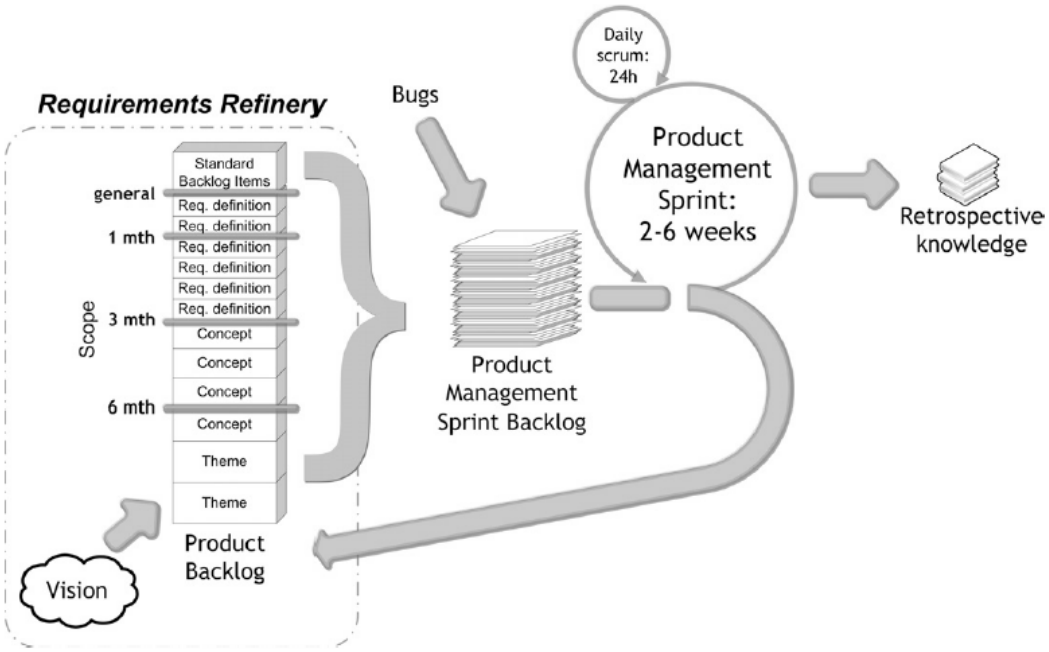


Figura 1.10. El procés de desenvolupament Àgil amb Scrum (Vlaanderen, 2010)

5. Resum

En aquest tema es pretén donar una idea generalitzada dels continguts i organització de l'assignatura. També s'intenta fer comprendre a l'alumne la importància de tenir uns coneixements sòlids per tal d'enfrontar-se amb la construcció de programari com a part dels sistemes informàtics, igual que en qualsevol altra enginyeria.

Es dóna una primera idea de quin és el producte que un enginyer informàtic ha de construir o millorar, com també una introducció a les fases, les activitats i les tasques que formen part del procés per a construir aquest producte.

Es descriuen diferents processos que poden ser emprats per a desenvolupar programari.

Activitats complementàries

1. Tenint en compte les assignatures cursades, feu una classificació de tècniques i mètodes que heu aprés, i per a cadascun identifiqueu quina seria la seua utilitat.

2. Feu una llista de tots el tipus de programari dels quals feu ús en un dia normal i classifiqueu-los.
3. Com es podrien classificar el programari d'oci o videojocs? I les xarxes socials?
4. Busqueu un exemple de programari de cadascun dels tipus identificats, que no heu trobat a l'exercici 1.
5. Considerant un producte de programari Open-source, identifiqueu quin seria el treball d'un enginyer en informàtica o quins beneficis podria obtenir treballant amb aquests productes.
6. Busqueu uns altres dos models o paradigmes de desenvolupament de programari, identifiqueu-ne les diferències amb els proposats en el tema i feu una taula en la qual relacioneu quins models serien adequats per a desenvolupar quin tipus de programari dels estudiats en la secció 2.
7. Estudieu el model de desenvolupament de programari de prototips, en quina fase i quan es podria fer la planificació del projecte?

CAPITOL 2

Projecte: Inici i planificació

«*Projects whithout clear goals will not achieve their goals clearly.*»
(Gilb, 1988)

Objectius

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Aprendre què és un projecte i quins són els tipus de projectes que es poden dur a terme en enginyeria del programari.
- Conèixer quins poden ser els punts de partida i les possibles causes que provoquen la decisió de començar un projecte de desenvolupament de programari.
- Saber etapes i activitats que s'han de dur a terme per a la gestió correcta de projectes.
- Conèixer els conceptes bàsics de planificació i estimació de projectes.
- Aprendre com es pot fer la planificació temporal d'un projecte de desenvolupament de programari, tenint en compte les activitats que cal realitzar, les seues relacions i els recursos.
- Tenir un coneixement bàsic de com es pot estimar el temps (esforç) que és necessari per a desenvolupar un projecte.
- Saber fer un diagrama de Gantt i un esquema de descomposició de treball.

1. Introducció

El desenvolupament de programari és una tasca complexa, que suposa l'execució i coordinació d'un nombre important d'activitats i recursos. Per tant, una bona manera de gestionar aquest desenvolupament és l'organització d'aquestes activitats en projectes. Definir projectes per a desenvolupar productes és típic de les disciplines d'enginyeria. En l'enginyeria informàtica també s'ha adoptat aquest enfocament, per tal d'aplicar tècniques que puguen ajudar a una gestió més eficient de les activitats de desenvolupament de programari o d'altres productes informàtics. En enginyeria informàtica es poden trobar projectes de diferents tipus, per a instal·lar una xarxa de comunicacions sense fil, per a desenvolupar una nova interfície de comunicació, per a desenvolupar programari a mida o per a adquirir i implantar un producte de programari desenvolupat per una tercera empresa, entre altres.

En aquest tema es descriu el que s'entén per un projecte de desenvolupament de programari i les diferents activitats que inclou en un projecte bàsic. El tema descriu amb més detall què és la planificació del projecte, i quines tasques inclou aquesta activitat. Finalment, es descriuen algunes tècniques de suport a la planificació i s'explica com fer una estimació pel que fa al cost temporal d'un projecte de desenvolupament d'un producte de programari.

La planificació que es desenvolupa com a exemple en aquest tema, i en el cas pràctic, corresponen al que seria un projecte de programari desenvolupat seguint un model seqüencial, perquè s'adapta millor a la seqüència temporal de l'aprenentatge dels temes de l'assignatura. Però la definició d'activitats dependrà en cada projecte del model de desenvolupament que se seguirà.

2. Projectes de programari

El procés de desenvolupar un producte de programari de certa grandària comporta un conjunt d'activitats, persones i recursos que s'han d'organitzar i gestionar de manera correcta per a assolir la consecució dels objectius marcats. L'èxit del projecte vindrà donat perquè el producte que es desenvolupa té la qualitat correcta, les característiques de la qual ja s'han vist en el tema 1; i a més a més, perquè el procés també acaba en temps i costos viables per a les parts involucrades.

Per poder aconseguir l'èxit del procés és necessari per una part, organitzar les activitats en un projecte i per una altra part, realitzar tasques de suport que s'emmarquen dintre del que es denomina gestió de projectes.

La guia *Project Management Body of Knowledge* PMBOK (publicada pel Project Management Institut PMI[1]) defineix un projecte com «un esforç temporal que es dut a terme per crear un únic producte o servei» (PMBOK, 2016). Es poden trobar altres definicions de projecte com ara «conjunt d'activitats i tasques que es duen a

1. http://www.pmi.org

terme per tal d'assolir un objectiu, de manera que implique un treball no immediat o en un termini relativament llarg» (Piattini, 2004).

Traslladant aquestes característiques a l'àmbit de l'enginyeria del programari, un projecte és un conjunt d'activitats organitzades en fases, tasques i subtasques agrupades amb diferents criteris, que segueixen mètodes i usen tècniques de desenvolupament de programari, amb l'objectiu de desenvolupar un producte de programari o proporcionar un servei relacionat amb els sistemes informàtics condicionat pel temps i els recursos humans i tecnològics a l'abast.

En general, des d'un punt de vista pràctic, un projecte es caracteritza per:

- Estar format per un conjunt d'activitats que es poden subdividir en tasques i subtasques cada vegades més xicotetes i que, agrupades amb diferents criteris, donen lloc al que s'anomena Paquets de Treball (*Work Packages*, WP).
- Tenir un objectiu final el qual es pot subdividir en diferents objectius parcials, que pot ser aconseguir el desenvolupament, la millora o la modificació d'un producte, un servei per donar solució a un problema determinat o cobrir un objectiu.
- Produir un resultat final i un o diversos resultats parcials (*deliverables*).
- Estar condicionat pel temps i els recursos (*restrictions*), amb una data d'inici, una data de fi i diverses dates límit o fites que es volen aconseguir.

En aquest apartat després d'analitzar què és un projecte en l'àmbit de l'enginyeria del programari es descriu com pot iniciar-se un projecte de programari, què són i com s'identifiquen els objectius, l'abast i les restriccions del projecte i del producte.

2.1. Inici del projecte

L'inici d'un projecte de desenvolupament d'un producte de programari comença amb una fase de comunicació. Algú dintre de l'empresa detecta una necessitat o algú oferta un producte per donar suport a un procés de negoci. Aquest fet propicia el contacte entre l'enginyer informàtic i les persones responsables de l'adquisició i ús del programari. Aquest primer contacte permet identificar quines són les necessitats que han generat la demanda, els objectius del producte i establir les primeres relacions contractuals si és necessari.

Existeixen diferents punts de partida pels quals una empresa o un grup d'usuaris detecta la necessitat de millorar un sistema informatitzat o de crear-ne un de nou, com poden ser:

- L'organització té un *pla de sistemes* previ on enginyers d'organització i de sistemes han definit necessitats a llarg i mitjà termini per tal de millorar l'estratègia de l'empresa, i que inclou la revisió i millora dels sistemes d'informació i de la seua part informatitzada.

- Es produeixen canvis en els procediments que utilitza l'organització, i aquests canvis necessiten nou programari o modificació dels que hi ha en funcionament.
- Les *àrees operatives* detecten necessitats que no estan cobertes pels sistemes o programari en funcionament.
- Influència o imposicions d'agents externs, legislacions, imposicions tecnològiques de clients o proveïdors que afecten a la funcionalitat del programari, noves col·laboracions que requereixen de la connectivitat de les tecnologies de l'empresa.
- Millores en les tecnologies de la informació, nou maquinari, nous sistemes operatius més ràpids, possibilitat de compartir informació entre diferents àrees, incloure comunicacions en xarxa per a diferents departaments, clients, proveïdors, etc.
- En definitiva, algú de l'organització detecta una necessitat, aquesta necessitat implica una millora o un canvi de programari i es decideix crear un equip de treball per tal de desenvolupar aquesta millora.

Totes aquestes activitats impliquen la realització de reunions on s'estableixen les diferents condicions que el producte de programari i el projecte han de complir, objectius i abast del projecte, es té un primer contacte amb els usuaris implicats, i es reconeixen quines persones o usuaris claus són els que han de tenir una participació més activa en cadascuna de les activitats del desenvolupament del sistema informàtic o del programari

Per tant l'inici del projecte inclou les següents activitats (vegeu figura 2.1):

- Definir objectius i abast del projecte i del producte.
- Identificar restriccions.
- Avaluar alternatives per al desenvolupament del producte de programari.

Figura 2.1. Inici del projecte: comunicació usuaris i enginyers (Pressman, 2010)

Tenint en compte les condicions inicials, objectius, abast, alternatives i millores, cal fer un estudi de viabilitat on s'estimen temps, costos i beneficis i es planifique el projecte. La planificació permet organitzar les activitats estimades en temps tenint en compte restriccions i els recursos humans i tecnològics. La planificació és desenvolupada habitualment per enginyers informàtics amb experiència en altres projectes similars i també pels directius de l'organització amb capacitat de decisió sobre el sistema del qual forma part el programari. La viabilitat avalua si els beneficis previsibles i la necessitat compensen en un temps assumible per l'empresa la inversió que es farà per desenvolupar el producte. L'estudi de viabilitat no es tracta en aquesta assignatura.

2.2. Definir objectius i abast del projecte i del producte

Com s'ha mencionat, el procés de desenvolupament d'un producte de programari comença quan s'identifica un problema o una necessitat i es considera que la forma de resoldre'l involucra uns elements de programari i maquinari. En aquesta primera activitat es decideix invertir econòmicament o amb recursos humans considerant que aquesta inversió proporcionarà beneficis a curt o llarg termini. És a dir, hi ha uns objectius que justifiquen la inversió que es produirà. **L'objectiu del projecte** de desenvolupament d'un producte de programari serà desenvolupar un producte que assolisca les necessitats identificades amb un temps i cost adequats.

L'abast del projecte identifica fins on s'arriba i, el què és més important, què s'inclou dintre de les tasques del projecte i què no. A vegades el projecte té una envergadura que sobrepassa certa grandària i és necessari definir projectes més menuts, de manera que l'abast se subdivideix en diferents subprojectes. L'abast del projecte també pot considerar part de les fases de desenvolupament del producte, si es decideix fer subprojectes per necessitats organitzatives o de viabilitat. Per exemple, un projecte pot incloure només la definició dels requisits d'un nou producte de programari i l'avaluació d'alternatives per al seu desenvolupament o compra.

Els objectius del producte de programari o del sistema o servei han de definir-se des del punt de vista operatiu (curt termini i baix detall), tàctic (mig termini) i estratègic (llarg termini, solucions per a directius executius), han de ser realistes i donar resposta a la pregunta: **per què** es desenvoluparà el programari?, en lloc de què farà el programari?

Objectius estratègics poden ser per exemple: posicionament en el mercat, obrir noves línies de negoci, tenir una imatge compromesa amb la societat o més actual, quant a l'ús de tecnologies. Objectius tàctics poden ser: millorar la gestió de clients, poder gestionar nous productes de manera adequada, millorar l'eficàcia i qualitat del treball dels empleats de l'empresa, etc. I objectius operatius serien aquells que estan relacionats amb aspectes de més baix nivell pel que a fa als processos de l'empresa, per exemple: reduir el temps de lliurament de productes als clients, poder tenir informació sobre les perfils dels clients i les compres que fan,

poder saber quins tipus d'espectacles els agrada més al públic d'una zona geogràfica, controlar les vendes i saber la disponibilitat de producte en temps real, etc.

L'abast del producte identifica quina funcionalitat ha de cobrir, fins on arribarà, àrees involucrades, les interaccions amb altres entitats u organitzacions o amb altres dispositius o sistemes informatitzats. Igual que a l'abast del projecte caldrà identificar clarament que es considera part del producte i el que és més important, que es queda fora i per tant no es programarà perquè seria més temps i cost afegit al projecte: funcionalitat que no és considera part del producte, no és viable o es considerarà en millores posteriors.

La complexitat de la determinació dels requisits augmenta quan el programari ha de mantenir connexions amb altres sistemes o ens externs. En aquests casos s'han d'estudiar i identificar estàndards, legislacions, formats preestablerts, etc., que s'han de tenir en compte durant la definició de requisits i tot el desenvolupament del programari.

Per tal de definir l'abast del projecte i l'abast de producte, es poden tenir en compte els següents aspectes:

- Abast des del punt de vista organitzatiu, quines àrees o departaments dins de l'organització faran ús del programari i quines organitzacions externes (bancs, administracions públiques, clients, proveïdors, etc.) interactuaran. Aquest punt és fonamental perquè d'interacció amb organitzacions externes implica la necessitat d'usar estàndards de formats d'intercanvi de dades, comunicacions i aspectes relacionats amb la seguretat i protecció de la informació.
- Abast del sistema des del punt de vista funcional, on s'identificaran les funcions (alt nivell) que cobrirà el programari tenint en compte els processos de negoci als quals donarà suport. Per exemple, el sistema servirà per millorar i agilitzar la gestió de les reserves de lloguer d'apartament als clients però no inclourà manteniment de clients.
- Abast des del punt de vista informàtic, és a dir, quins altres productes de programari actuals se substituiran o s'integraran, ja siga amb modificacions o no, de manera parcial o total amb el nou producte. La interacció amb altres sistemes implica la necessitat d'establir comunicacions, intercanviar dades, o connectar programes ja en funcionament que poden imposar aspectes relacionats amb el disseny i la implementació de programari que cal tenir en compte.

2.3. Identificar restriccions

Cada organització té uns recursos limitats que pot utilitzar per a aconseguir els seus objectius. Aquests poden ser de temps, econòmics, humans, d'espai i tots han de ser considerats de forma convenient per al correcte funcionament de l'organització. Les restriccions més usuals en el desenvolupament de productes de

programari normalment són les relacionades amb els costos, el temps i el personal disponible. Poden ser imposades pels responsables de l'organització per a controlar la utilització dels recursos o bé per altres factor externs (lleis, normatives, canvis de monedes, etc.). Altres restriccions poden estar condicionades per factors predefinits com són per exemple la necessitat de mantenir la compatibilitat amb el maquinari existent o evitar la necessitat de contractar més personal.

A vegades les restriccions poden fer variar l'abast del projecte identificat en un inici i en ocasions fer que la llista dels objectius «desitjats» varie substancialment o no siga factible.

En aquest punt, el treball de l'enginyer informàtic és discernir i aconsellar els usuaris entre les necessitats factibles i el que es vol, per tal d'obtenir uns objectius clars i específics de forma que els recursos del projecte permeten aconseguir-los.

2.4. Avaluar alternatives

A l'hora de desenvolupar un producte de programari els responsables de l'empresa i els del projecte poden considerar diferents alternatives, com per exemple d'adquirir un producte de mercat, o fer un producte a mida, o contractar el serveis d'una empresa externa o que el treball el desenvolupe personal de la mateixa empresa, o qualsevol combinació de totes elles. A més a més, les tecnologies actuals proporcionen altres alternatives com ara el *cloud computing* o el software lliure, que cada vegada són més usats com a alternatives als sistemes tradicionals implantats en cada empresa.

Els recursos dels quals es disposa, tant econòmics com de recursos humans, d'equipament o d'espai, influeixen en prendre una decisió entre diferents alternatives.

Per avaluar alternatives primer cal identificar aquelles que són adequades per al desenvolupament del projecte, i les característiques del producte, i valorar els beneficis, costos i riscs de cadascuna. L'enginyer informàtic ha de fer una proposta raonada completa, per tal que els usuaris responsables puguen decidir amb tota la informació possible al seu abast.

En aquesta activitat s'utilitzen tècniques, que corresponen a coneixements d'altres matèries, com ara estudis de costs i beneficis i càlculs del retornament de la inversió.

3. Gestió de projectes

La gestió d'un projecte inclou com a activitats fonamentals tres activitats, les quals es poden executar en una seqüència cíclica però estan actives durant tot el desenvolupament del projecte:

- **Direcció del projecte** que organitza, coordina i supervisa les activitats, assigna recursos, defineix estàndards de qualitat i de l'empresa per al procés i per al producte. El responsable és el cap del projecte. La tasca de direcció implica a més d'exercir de cap, les següents accions (Piattini, 2004): motivar els membres de l'equip de treball, afavorir la comunicació, coordinar les activitats que cal realitzar, valorar l'execució del projecte, resoldre els possibles conflictes que puguen sorgir i mantenir les polítiques d'empresa, entre altres.
- **Control del projecte** que supervisa la realització correcta del projecte, tenint en compte la planificació inicial, per tal de detectar possibles desviacions i corregir-les. Controlar l'execució d'un projecte de desenvolupament de programari suposa seguir, revisar i comparar els resultats davant les estimacions, els compromisos i el pla de projecte, i actualitzar-ho tot en funció dels resultats. El control implica les accions següents (Piattini, 2004): supervisar la realització del projecte, comparar l'execució actual i la desitjada o estimada i prendre accions correctives sobre les possibles desviacions detectades.
- **Planificació del projecte** que identifica les activitats que s'han de dur a terme en el projecte i estableix un calendari de realització d'aquestes en funció dels recursos i temps disponibles.

El cap del projecte és la persona que té la responsabilitat de planificar, dirigir i controlar les activitats del projecte, en funció dels recursos i temps disponibles. En aquest tema s'introduiran alguns conceptes i tècniques bàsics per a la planificació de projectes. La direcció i el control de projectes són activitats d'un nivell més alt dels plantejats als objectius de l'assignatura i és per això que no es tractaran amb més detall.

A més a més d'aquestes tres activitats bàsiques, n'hi ha d'altres que donen suport a l'execució el projecte i al control de la qualitat del procés com són: gestió del risc, gestió de la qualitat, gestió del canvi, i que s'estudien amb més detall en altres matèries.

Aquest tema se centra principalment en la planificació des d'un punt de vista bàsic. S'introdueixen els conceptes necessaris que permeten fer una estimació aproximada i una planificació d'un projecte d'enginyeria del programari de forma molt simplificada.

4. Planificació del projecte

La planificació defineix les necessitats econòmiques, de recursos humans i tecnològics, com també la durada temporal del projecte. La planificació és desenvolupada habitualment per enginyers informàtics o d'organització amb experiència en altres projectes similars, i també pels directius de l'organització amb capacitat de decisió sobre el producte a desenvolupar.

La planificació d'un projecte de desenvolupament de programari té com a objectiu establir les activitats que s'han de realitzar per tal d'aconseguir els objectius del projecte, organitzar-les temporalment tenint en compte els recursos, les dependències entre elles i altres restriccions. Per poder planificar un projecte cal estimar quan de temps i recursos són necessaris en cadascuna de les activitats identificades.

4.1. Activitats de la planificació del projecte

Existeixen nombroses propostes sobre com realitzar la planificació de projectes, fins i tot estàndards avalats per entitats internacionals com el PMI o la IEEE.[2] Tenint en compte aquestes propostes, en aquesta assignatura es defineixen els següents passos bàsics per dur a terme una planificació:

- **Definició dels objectius** del projecte, que han de ser realistes, quantificables i factibles.
- **Identificar** fases, activitats i **tasques** i organitzar-les de manera jeràrquica. El resultat es pot visualitzar en un diagrama de descomposició de treball (*Working Breakdown Structure*, WBS) figura 2.4.
- **Estimar** l'esforç en hores/persona per tasca/activitat. L'esforç en el desenvolupament de programari es mesura en hores, una activitat A dura 24 hores/persona. És a dir, si una persona treballa 8 hores al dia, A té una durada de 3 dies. I si s'assignen dues persones? I 24 persones? L'estimació es veurà amb més detall a l'apartat 4.2.
- **Establir relacions** entre tasques/activitats, indicant les predecessores. Una activitat A és predecessora d'una altra B si per poder iniciar B, A ha d'estar finalitzada.
- **Identificar i distribuir els recursos**, és necessari identificar l'equip de treball que constituirà els recursos humans i assignar-los a les activitats i tasques de manera adequada tenint en compte l'esforç estimat, el perfil dels components de l'equip de treball, la disponibilitat, etc.
- **Crear el calendari del projecte** amb les activitats, precedències, temps i recursos estimats fins a aquest moment. La tècnica que es pot utilitzar per tal de confeccionar el calendari és el diagrama de Gantt.

2. IEEE són les sigles de l'Institute of Electrical and Electronics Engineers, l'Institut d'Enginyers Elèctrics i Electrònics, una associació tècnico-professional mundial dedicada a l'estandardització, entre altres coses.

El resultat final de l'etapa de planificació d'un projecte de desenvolupament de programari ha de ser el pla de projecte. El projecte es pot planificar i controlar amb el suport d'eines informatitzades (com per exemple el Microsoft Project). Al pla de treball també s'ha d'incloure: els resultats, finals i parcials (*deliverable* o lliurable) i el cost que es calcula a partir dels recursos assignats. A vegades també es necessari detallar aspectes logístics com el lloc on s'ha de fer, de quins recursos físics, maquinari i programari es disposa per tal de treballar en condicions adequades i desenvolupar el treball.

4.2. Estimació de projectes de programari

L'objectiu de l'estimació de projecte es realitzar una quantificació aproximada de l'esforç o temps d'execució de cada activitat i del cost que suposa la seua realització. Per a estimar el temps o durada d'una tasca o activitat cal realitzar una mesura d'allò que s'ha de produir, és a dir, el programari. Com es pot mesurar el programari?

Per exemple, si s'analitza el cas d'un pintor al qual se li demana un pressupost per pintar una casa, aquest per fer un pressupost i planificar el treball primer mesura la superfície de la casa, per tal d'obtenir la quantitat de metres que ha de pintar i després realitza un càlcul del temps que li suposa pintar la casa tenint en compte els metres, les dificultats de la casa, el tipus de pintura que es vol usar i la seua experiència en aquest tipus de treball. Després, segons els recursos de personal de què disposa i les tècniques que usarà pot calcular el temps que durarà el treball. Per a calcular el cost final del treball haurà de tenir en compte les hores i el seu preu (per categories professionals) i el material que estime que ha d'utilitzar.

- La mètrica és m^2 de paret.
- La **mesura** és el resultat del mesurament, en aquest cas 100 **m^2 que s'han de pintar.**
- Segons la seua experiència en altres treballs previs, sap que pot pintar 25 m^2 per dia, tenint en compte un tipus de pintura determinat i unes parets normals. Els 25 m^2/dia és el **factor de productivitat**.
- **L'esforç** total (en dies-pintor) per realitzar el treball resulta de dividir els m^2 pel factor de productivitat, dóna 4 dies pintor o 32 hores-pintor.
- També sap quanta pintura s'usa per m^2.
- La planificació l'obté considerant quantes persones participaran en el treball, quantes hores treballaran al dia i quants dies. Haurà de tenir en compte que tal vegada hi ha tasques que s'han d'acabar abans de començar unes altres.
- El cost l'obté multiplicant hores per preu hora (segons categoria de les persones que han treballat) i sumant el cost del material que estima que usarà.

En enginyeria del programari s'usen els següents conceptes:

- **Mètrica**, que és la unitat de mesura que es defineix basada en les característiques d'un producte o activitat, per exemple línies de codi (LDC o KLDC) o punts de funció (PF) o punts de casos d'ús (PCU).

- **Mesura** que s'obté d'un mesurament o acció de mesurar. El resultat pot ser 100 línies de codi, per exemple.
- **Factor de productivitat**, és la dada que indica quantes unitats de les mesurades es poden desenvolupar en una determinada unitat temporal. Per exemple, 18 hores per 1.000 LDC. El factor de productivitat està condicionat per l'experiència de l'equip, les característiques de l'entorn i les característiques del programari.
- **L'esforç** en hores-persona, total d'hores que una persona necessita per a desenvolupar una tasca. És el resultat de multiplicar la mesura obtinguda pel factor de productivitat. L'esforç és un total en brut que s'ha de distribuir en el temps tenint en compte altres activitats i els membres de l'equip.

Per tant, per a estimar el temps i el cost i obtenir l'esforç, cal conèixer com es pot mesurar el programari. Per a mesurar-lo es poden definir mètriques com ara: interfícies d'usuari, programes, accessos a la base de dades, accessos al sistema i tradicionalment, LDC (línies de codi). Però el fet de mesurar un producte lògic, no físic com una paret, i basats en el que en un futur serà el sistema incorpora un punt més d'incertesa i inexactitud en el resultat del mesurament. Aquesta mesura de línies de codi no és molt aplicable en els entorns de programació actuals i n'han sorgit algunes alternatives. En aquest curs utilitzarem una mètrica basada en els *casos d'ús* que s'estudien en el tema 3, però es poden consultar altres exemples en el capítol 25 de Pressman (2010).

Després de mesurar el producte, s'ha de definir el factor de productivitat, és a dir, quantes hores es necessita per a desenvolupar una unitat. El factor de productivitat és un càlcul que s'obté a partir d'altres projectes semblants, igual que el pintor calcula quantes hores necessita per cada x metres quadrats. Quantes hores costa de programar una interfície? El factor de productivitat depèn de la complexitat de l'entorn de desenvolupament, de l'experiència dels membres de l'equip i d'altres factors externs o tecnològics que cal tenir en compte.

A partir de l'esforç (hores) es calcula el nombre de persones i el temps que han de treballar per tal de realitzar el projecte. S'ha de puntualitzar que la relació que existeix entre el nombre de persones que participen en un projecte i el temps de desenvolupament no és lineal, perquè cal tenir en compte que en el desenvolupament de programari la comunicació entre tots els integrants del projecte és fonamental per a la realització de les diferents activitats (Brooks, 1995). Una vegada s'ha determinat el nombre de recursos i el temps que s'ha d'invertir, ja es pot calcular quin és el cost del projecte i fer una planificació temporal.

En la taula 2.1 es mostren els càlculs d'un exemple molt simplificat. Es considera que es mesuren (es fa una previsió) 80 interfícies d'usuari, i un factor de productivitat de 15 hores per a desenvolupar una interfície, l'esforç per a aquesta activitat són 1200 hores-persona. En aquest cas, es considera que tenir un equip de tres persones requereix incrementar l'esforç un 5 %, per a cobrir tasques de coordinació, comunicació, revisió del treball etc. Si cada desenvolupador treballa 7 hores per dia, són 60 dies que corresponen a 12 setmanes.

Taula 2.1. Exemple de càlcul de durada de tasques.

Mètrica	Mesurament	Factor de productivitat Hores/unitat	Hores persona	Càlcul coordinació 5 %	Dies 3 persones 7 hores dia	Setmanes
Interfícies d'usuari	80	15	1200	60	60,00	12,00
Taules BBDD	20	10	200	10	10,00	2,00

L'estimació en projectes de desenvolupament de programari es caracteritza per:

- No ser una ciència exacta, l'estimació és sinònim de previsió, que no d'endevinació, es tracta d'una valoració amb algun marge d'error. Per tant, és necessari seguir una sèrie de passos sistemàtics que proporcionen un risc acceptable de la previsió realitzada.
- S'ha de realitzar a priori, però com més s'avança en el temps, més fiable és la predicció. Per tant, és possible seguir un procés continu de refinament.
- No és una estimació en unitats monetàries, sinó en termes de l'esforç que és necessari per a desenvolupar un determinat projecte. Aquest esforç es mesura en persones-mes, com una mesura del nombre de persones que haurien de treballar durant un mes per tal d'aconseguir l'objectiu desitjat. El cost d'un projecte de desenvolupament de programari està dominat pels recursos de personal.

Existeixen diferents **mètodes d'estimació** de l'esforç i del factor de productivitat en projectes de desenvolupament de programari:

- **Opinió d'experts:** estimació que es realitza basada en l'experiència personal de durades de projectes semblants.
- **Estimació per analogia:** variant més formal que l'anterior que consisteix a comparar projectes similars i també diferents, i a partir d'aquesta comparació obtenir l'esforç del nou projecte que es vol estimar.
- **Estimació per descomposició:** descompondre el projecte fins a un nivell de detall suficient que permeta estimar l'esforç de les unitats elementals en les quals s'ha descompost el projecte. L'esforç total del projecte és igual a la suma de l'esforç de cadascuna de les unitats elementals.
- **Models d'estimació:** models que presenten equacions o fórmules matemàtiques que relacionen diferents paràmetres del projecte amb el cost o l'esforç.
- Estimació basada en punts de funció o en unitats d'implementació com ara taules de la base de dades o interfícies.
- **Estimació basada en casos d'ús** (Karner, 1993).

4.3. Tècniques de suport a la planificació

La planificació temporal té com a objectiu generar un calendari amb les activitats, precedències, temps i recursos estimats fins a aquest moment. Per a representar la planificació d'un projecte s'estudia en aquest apartat el diagrama de Gantt i els diagrames de descomposició de treball (WBS). Encara que no es veuran en aquesta assignatura existeixen altres tècniques basades en grafs com els diagrames PERT (*Project Evaluation and Review Techniques*) o els grafs de Roy per a calcular la durada d'un projecte, identificar tasques que poden ser critiques, etc.

En la taula 2.2 es mostren les activitats definides per a un projecte de desenvolupament de programari que segueix l'estructura del model general de Pressman, desenvolupat amb l'eina Microsoft Project 2016. Cada activitat inclou unes determinades tasques que s'indiquen pels nivells d'indentació. Cada tasca de més baix nivell té una durada assignada a partir de l'estimació feta. La durada total del projecte i les durades de les activitats que agrupen tasques depenen de quines activitats es poden realitzar en paral·lel (el que s'ha de considerar per a assignar recursos) i quines han d'estar seqüenciades en el temps.

El diagrama de descomposició de treball (*Work Breakdown Structure*, WBS) visualitza de manera jeràrquica quina és l'estructura de les activitats. Habitualment un projecte es descompon en fases, les fases en activitats i les activitats en tasques, en la figura 2.2 es mostra el WBS per a l'exemple de projecte proposat. Aquesta estructura permet agrupar les tasques i activitats del projecte en paquets de treball (*Work Packages*, WP) i assignar-li millor els recursos.

El diagrama de Gantt és una tècnica que permet fer una planificació temporal i generar un calendari amb les activitats, precedències, temps i recursos estimats, es pot veure un exemple fet amb Microsoft Project 2016 en la figura 2.3. Els passos a seguir per a desenvolupar el diagrama de Gantt són:

- Indicar la data de començament del projecte i les seues característiques generals.
- Organitzar l'esquema lògic d'activitats.
- Determinar amb l'ús de tècniques d'estimació la durada de cadascuna de les activitats.
- Identificar les relacions i dependències entre les activitats assenyalant les tasques que són predecessores d'una altra tasca determinada.
- Identificar i assignar els recursos.

Taula 2.2. Exemple d'activitats i descomposició d'un projecte de programari.

Activitat/Tasca	Durada
Projecte	**55 dies**
Inici: Comunicació i proposta tècnica	**7 dies**
Reunions inicials	3 dies
Definició d'abast i objectius	1 dia
Planificació	4 dies
Presentació del pressupost i aprovació	0 dies
Modelització	**14 dies**
Anàlisi	**6 dies**
Modelitzar processos	5 dies
Modelitzar dades i components	6 dies
Disseny	**8 dies**
Dissenyar interfícies	6 dies
Dissenyar components	5 dies
Dissenyar base de dades	4 dies
Dissenyar plataforma física	2 dies
Validar model	0 dies
Construcció	**29 dies**
Creació entorn físic	5 dies
Construcció components	17 dies
Acoblar components	3 dies
Prova	4 dies
Posada en marxa	**5 dies**
Formació	5 dies
Conversió	2 dies
Lliurament	1 dia

Una vegada identificades les tasques i subtasques s'han de considerar les possibles restriccions per realitzar una activitat. Aquestes restriccions poden ser amb precedències, que indiquen que una activitat no pot començar fins que no acabe una altra, o bé temporals, indicant una data exacta en la qual pot començar una activitat.

Entre les tasques cal diferenciar:

- Les **predecessores** indiquen quines activitats han d'haver finalitzat per a poder iniciar una altra activitat. Per exemple, per a provar el programari és necessari haver generat almenys una part del codi, per a fer una posada en marxa han d'estar formats els usuaris i provat el programari, etc. A vegades cal definir una data, perquè és un dia crític en el funcionament del negoci o s'ha de complir alguna condició especificada per a una data.

- Les **fites** són activitats que marquen un punt en el temps en el qual no es pot seguir si no s'ha dut a terme adequadament, per exemple cal prendre una decisió valorant una proposta o un pressupost o un determinat producte, o cal tenir un maquinari instal·lat i a punt per a continuar, etc. En Microsoft Project 2016 s'ha de crear una tasca amb durada 0 per a representar una fita, i gràficament, per defecte es representen com un rombe.
- **Activitats crítiques**, són aquelles que si s'endarrereixen provoquen que tot el projecte finalitze fora del termini mínim establert.
- **Tasques de resum**, que agrupen altres tasques, aquestes tenen una durada condicionada per la durada de les subtasques i per com estan seqüenciades, és a dir, com es defineixen les precedències. Per exemple, si es poden realitzar totes en paral·lel la tasca durarà el que dure la subtasca més llarga, i si s'han d'executar una darrere de l'altra, la tasca durarà la suma de les durades de les subtasques.

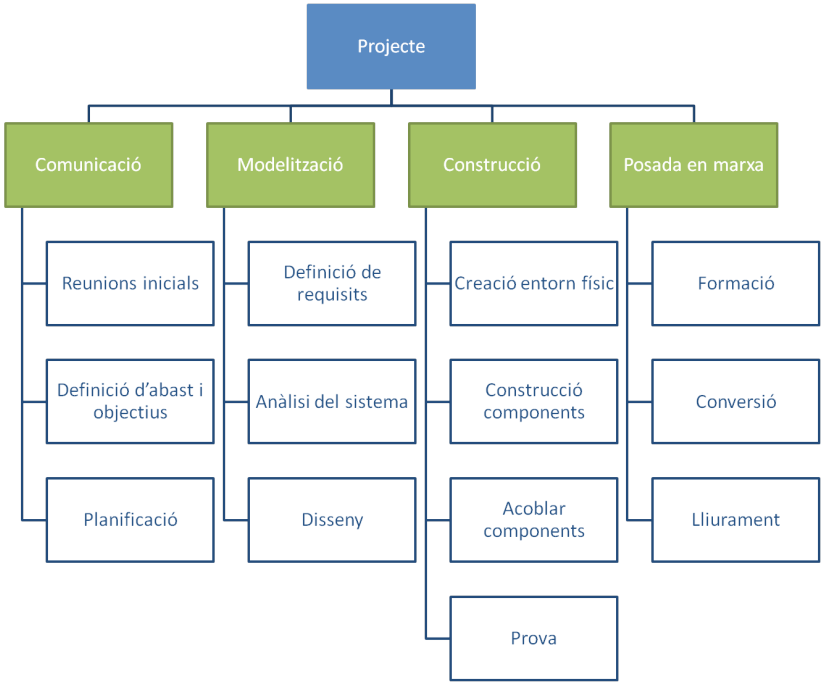


Figura 2.2. Esquema jeràrquic de la descomposició de treball (WBS)

En la figura 2.3 és mostra l'exemple de diagrama de Gantt fet amb l'eina Microsoft Project 2016 a partir de la definició de les durades i precedències de la taula 2.2. S'hi ha afegit una tasca que agrupa totes les altres, projecte. En la figura s'observa la representació dels diferents tipus d'activitats i les vinculacions en la part gràfica corresponents a les precedències.

- Les predecessores de cada tasca es detallen amb el número corresponent i gràficament apareixen vinculades amb una fletxa.
- La durada es mostra en dies, encara que es podria mostrar en hores.
- Les tasques de resum es representen amb una barra gràfica de color negre i el text en negreta.

- El diagrama gràfic mostra les tasques i les dates d'inici i fi amb el calendari i la barra temporal.
- Les fites es mostren amb un rombe i es defineixen donant-los durada 0.
- La durada de les tasques de resum es calcula automàticament activant l'opció de programació automàtica que proporciona l'eina.

L'últim pas és l'assignació de recursos, que s'haurà de fer tenint en compte les capacitats i competències dels membres de l'equip de treball i la disponibilitat. L'assignació de persones, com s'ha explicat abans, no redueix la durada de manera lineal. L'assignació de persones, com s'ha explicat abans, no redueix la durada de manera lineal. En la figura 2.3 es mostra el diagrama de Gantt realitzat amb Microsoft Project 2016.

Taula 2.3. Llegenda del diagrama de Gantt de la figura 2.3.

Tipus de tasca	Representació
Tasca de resum	
Fita	
Tasca simple	

En aquesta eina de suport a la planificació les predecessores s'assignen usant el número de la tasca que es genera de manera automàtica en la columna de l'esquerra i separats per comes si hi ha més d'una. Els tipus de tasques de resum o fites es poden representar de manera personalitzada, en aquest cas es mostren tal i com es descriu a la taula 2.3 i es mostren les inicials dels recursos a la dreta de les tasques.

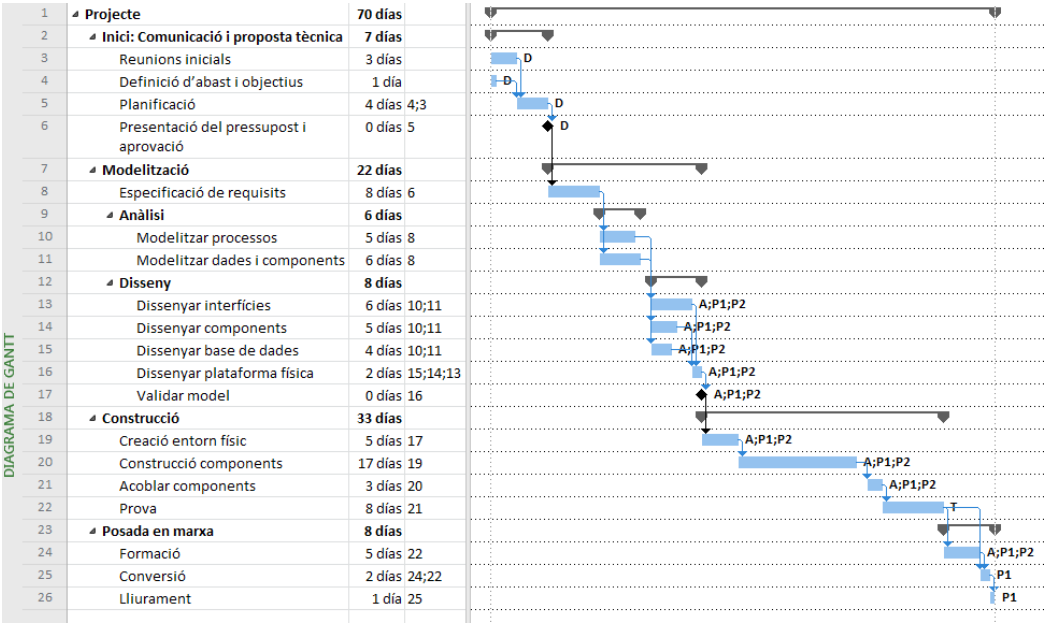


Figura 2.3. Imatge del diagrama de Gantt (Microsoft Project 2016)

5. Resum

Com en qualsevol àrea de l'enginyeria per a desenvolupar programari de certa grandària, es necessari definir i gestionar el projecte d'execució del producte. Per a gestionar el projecte es duen a terme tres activitats: planificació, direcció i control. Per a planificar un projecte és necessari definir les activitats, seqüenciar-les, estimar la seua durada, assignar recursos, entre altres tasques. Per a estimar l'esforç que s'ha d'invertir en realitzar una tasca cal definir mètriques. El diagrama de descomposició de treball i el diagrama de Gantt són dues tècniques que donen suport a la planificació i el control de projectes.

Activitats complementàries

1. Tenint en compte la definició de projecte, penseu un producte de programari que us agradaria desenvolupar:

   – Doneu un nom al projecte que hauríeu de definir per al seu desenvolupament i descriviu l'objectiu del producte i del projecte.
   – Penseu les activitats que hauríeu de dur a terme per al seu desenvolupament, seqüencieu-les indicant quines han d'haver acabat per a iniciar una altra. Per a cada activitat identifiqueu quin seria l'objectiu i el resultat.

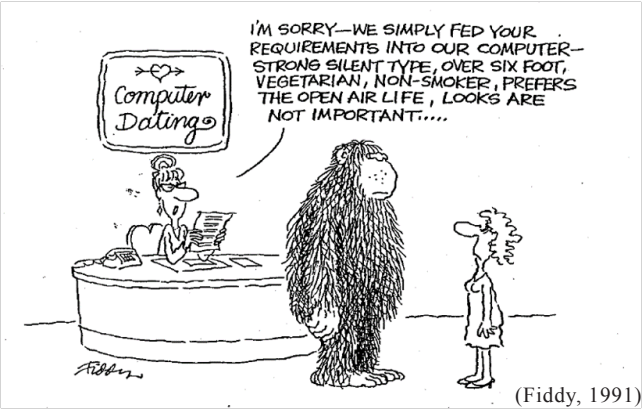
2. Donada la següent taula d'activitats d'un projecte de desenvolupament de programari:

   – Definiu tres activitats de resum i una per al projecte complet.
   – Organitzeu les tasques i dibuixeu el diagrama de descomposició de treball.
   – Representeu el diagrama de Gantt utilitzant el Microsoft Project.
   – Quina és la durada mínima del projecte? I la màxima?

Activitat	Activitat	Act. precedents	Durada
Estudi inicial	1	-	5d
Presentació del pressupost	2	1	3
Aprovació del pressupost i inici del projecte	3	2	4d
Reunions de control i seguiment del projecte	4	3	24d
Reunions de treball inicials	5	2	5d
Avaluació i selecció del programari	6	5	7d
Preparar plataforma de maquinari	7	5	5d
Parametrització del producte	8	6;7	10d
Desenvolupament de mòduls a mida	9	6;7	11d
Formació en el producte	10	7	12d
Migració de dades	11	8;9	2d
Posada en marxa	12	10;11	3d
Acceptació dels usuaris i pagament final	13	4;12	1d

CAPÍTOL 3

Definició de requisits



(Fiddy, 1991)

I know you think you understand what I said, but what you don´t understand is what I said isn´t what I meant.

(Young, 2001)

OBJECTIUS

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Aprendre què és i com es pot realitzar la definició de requisits.
- Saber què la definició de requisits és una activitat crítica de la qual depenen els bons resultats del producte final.
- Adquirir coneixements sobre les principals tècniques de comunicació i recopilació de dades i informació a fi d'estudiar la situació actual i definir què es desitja obtenir del sistema informatitzat.
- Saber documentar de manera adequada els requisits i quines condicions ha de complir aquesta documentació.
- Conèixer què són els diagrames de casos d'ús i aprendre a desenvolupar-los per a un cas pràctic.

1. Introducció

Com s'ha estudiat en els capítols anteriors a l'inici d'un projecte de desenvolupament de programari es produeix una fase de comunicació on es defineixen els objectius i l'abast del projecte, s'identifiquen les necessitats primordials que requereix el programari, es defineix i planifica el projecte i l'equip de treball.

En un projecte ideal els usuaris i els enginyers informàtics participen de manera col·laborativa en l'equip de treball. En aquest cas, investigar requisits es converteix en reunions de treball entre companys d'equip, com per exemple en els projectes que es desenvolupen seguint mètodes àgils. Altres vegades els usuaris estan físicament separats dels enginyers i és difícil treballar de manera col·laborativa. Per a reduir problemes cal tenir en compte els següents principis:

- Identificar agents participants (*stakeholders*),[1] que poden ser responsables de les àrees de negoci, *product manager*, clients, consultors, usuaris finals, enginyers informàtics, etc.
- Reconèixer diferents punts de vista, tenint en compte que poden haver-hi participants de diferents àrees, poden tenir objectius i expectatives diferents respecte al que els proporcionarà el producte de programari.
- Treballar cap a la col·laboració, en aquest aspecte l'enginyer de programari ha de saber reconèixer àrees on els participants poden coincidir i estar d'acord, i aspectes on poden haver-hi conflictes, identificant alternatives i prioritats. En definitiva, ha de saber desenvolupar una tasca d'intermediari entre necessitats contraposades.
- Primeres preguntes, en aquesta etapa es realitza el primer contacte i cal obtenir resposta a preguntes com: qui ha demanat aquest producte? qui en farà ús? quins beneficis econòmics o indirectes (socials, de qualitat, etc.) proporcionarà? La resposta a aquestes preguntes ajuden a comprendre la utilitat del producte i augmenta l'interès dels usuaris en la participació activa i, per tant, facilita l'èxit del projecte.

L'objectiu de la definició de requisits és identificar la funcionalitat i la informació que processarà el programari per a cobrir les necessitats dels usuaris.

> *Un requisit és una condició que el programari ha de complir o una característica que ha de tenir per tal d'assolir els objectius i abast definits, tenint en compte les restriccions.*

Les restriccions són limitacions físiques, econòmiques, organitzatives, temporals o tecnològiques que condicionen la implementació del programari, el seu abast i objectius. La identificació de requisits és un dels factor crítics en el desenvolupament de programari i tenir una idea clara a l'inici del projecte del que s'ha de construir, normalment no és fàcil. L'enginyer informàtic ha d'identificar quins són els usuaris implicats, ha d'investigar què volen els usuaris que el futur sistema

1. Anyone who benefits in a direct or indirect way from the system which is being developed (Sommerville, 1997).

realitze, identificar factors i lleis externes que afecten, documentar les seues necessitats i obtenir l'aprovació de l'equip sobre la documentació generada (vegeu figura 3.1).

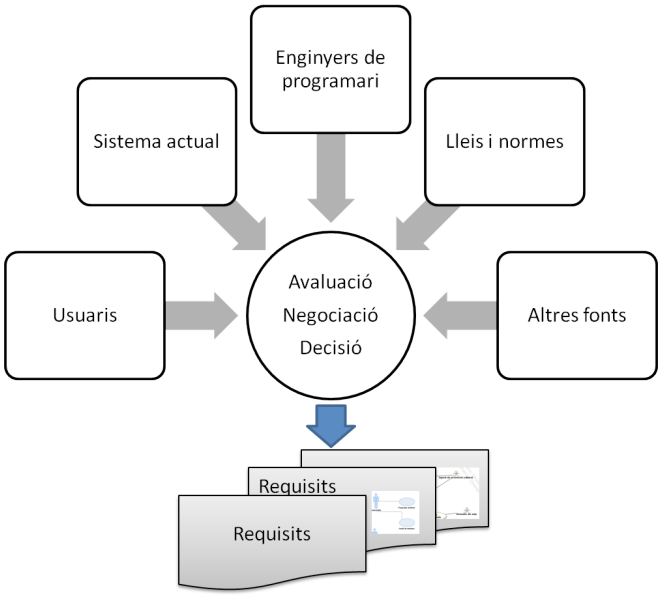


Figura 3.1. Participants i fonts de la definició de requisits

Per tant, la definició de requisits es desenvolupa en tres etapes fonamentals, que no se seqüències de manera estricta:

1. **Investigació o extracció**, durant aquesta primera etapa s'utilitzen tècniques apropiades de recerca per a investigar, buscar i recollir la informació, es fan reunions i es recullen idees que permetran determinar els requisits del sistema proposat.
2. **Anàlisi i documentació**, s'analitzen els fets i la informació identificats i s'organitzen i documenten. Per a documentar els requisits es poden usar tècniques d'especificació textual i de modelització com les que s'inclouen en aquest tema.
3. **Validació i verificació**, finalment es validen i verifiquen els requisits, en aquesta etapa es presenten els resultats als responsables de prendre les decisions sobre el sistema a desenvolupar per tal de validar la seua correcció, i es revisa la consistència i viabilitat per verificar que es defineixen de manera correcta.

Exemples de requisits amb una descripció textual:

- El sistema permet crear, modificar i esborrar informació dels empleats.
- El sistema ha de proporcionar funcionalitat per generar els informes mensuals de baixes per malaltia (format, informació que s'ha de mostrar).
- El sistema calcularà a partir de la informació dels contractes els imports de la nòmina per a cada empleat.

En aquest tema es detalla en què consisteixen aquestes tres etapes i es descriuen algunes de les tècniques que es poden utilitzar en cadascuna d'elles. Amb més detall s'inclou com desenvolupar els diagrames de casos d'ús per a documentar els requisits que corresponen al comportament del producte de programari; com documentar altres requisits de dades o de maquinari; i finalment, una xicoteta guia per a validar i verificar els requisits documentats.

2. Investigar els requisits del sistema

Investigar els requisits pot tenir un plantejament tan senzill com «preguntar als usuaris què volen que el producte de programari realitze», però aquest enunciat pot tenir complicacions degudes a diferents motius. Per exemple, factors que afecten a una correcta definició de requisits poden ser: l'experiència dels usuaris sobre què pot fer o no el programari o sistema; la complexitat o un abast que involucre moltes àrees o processos; la claredat en què estan definits els objectius; el coneixement sobre les tecnologies a emprar o l'ús de noves eines de les quals no es coneix el resultat per part de l'equip de treball; problemes de comprensió entre els usuaris i els enginyers informàtics respecte d'allò que es vol i que es pot fer o la idea que les dues parts tenen del producte final, etc. Aquests factors són clau per a l'èxit del projecte i s'han de tenir en compte dintre de la seua gestió, en particular en la gestió del risc. Per realitzar la investigació de requisits tenint en compte aquests factors es poden usar algunes tècniques bàsiques com ara:

- Realitzar reunions conjuntes codirigides per enginyers informàtics i usuaris.
- Preparar regles de participació i preparació de documents, agendes, etc.
- Nomenar moderadors de reunions.
- Definir mecanismes de gestió documental que faciliten la gestió de la informació que es recopila i del document o es redacten o especifiquen els requisits.

En aquest apartat s'estudia com es pot iniciar la investigació dels requisits amb la revisió del sistema en funcionament i algunes de les tècniques més usades per a investigar els requisits del futur sistema.

2.1. Revisar el sistema en funcionament

Per a investigar els requisits es poden utilitzar diferents tècniques. En ocasions un dels punts de partida fonamental és conèixer el sistema d'informació (automatitzat o no) en funcionament. Un bon enginyer informàtic ha de conèixer el negoci i context per al qual es desenvolupa el programari. Estudiar el sistema en funcionament proporciona, entre altres beneficis, una primera aproximació al coneixement dels processos de negoci dintre de l'abast del projecte. A més a més, aquest estudi també permet identificar quins punts forts o febles té el sistema actual, de manera que els requisits es definisquen tenint en compte els

primers, i millorant o eliminant aspectes deficitaris. Per a estudiar el sistema en funcionament es poden usar diferents tècniques, algunes de les quals es descriuen en els subapartats següents.

2.1.1. Recopilar documents existents

La recopilació de documents permet identificar dades d'entrada i d'eixida del sistema que poden formar part del requisits. Aquests documents poden recollir-se durant les reunions o amb anterioritat i cal fer una revisió del seu ús i validesa com també de la seua distribució.

En la taula 3.1 es proporciona una llista de documents, que es poden recollir i analitzar per tal d'obtenir informació sobre el sistema en funcionament.

Una vegada recopilats els documents caldrà analitzar el seu ús, per la qual cosa es pot fer un catàleg que assenyale quin document és útil, quines carències es detecten, quins documents o llistats no s'utilitzen, o quins altres són necessaris en el futur.

2.1.2. Observar el funcionament del sistema

Observar el funcionament del sistema i les operacions que es realitzen en el sistema existent és una tècnica molt habitual en sistemes de producció, i imprescindible quan es desenvolupa programari de control de producció o suport a processos industrials. També és útil per identificar algunes necessitats d'un sistema d'informació, per exemple quan existeix poca documentació o aquesta no està al dia o bé no és fiable, o quan hi ha problemes d'activitat que no són fàcils de detectar amb entrevistes i informació escrita, o perquè l'enginyer informàtic puga comprendre millor les activitats de l'empresa.

Aquesta tècnica s'ha de realitzar amb l'acceptació dels usuaris i deixant clar que l'objectiu no és controlar el treball que fa l'usuari, sinó entendre'l i estudiar-lo per tal de definir millores en el nou sistema.

Les observacions s'han de fer en moments d'activitat normal i en moments d'alta activitat o de realització de processos especials, per a detectar factors que poden influir en el bon funcionament del sistema, colls de botella, etc. No s'ha d'interferir en les tasques dels usuaris, observar sense alterar la seua activitat quotidiana dóna una visió més real de com es fa el procés.

Taula 3.1. Exemples de documents de sistemes d'informació.

Documents de sistemes d'informació	
Documents d'organització de l'empresa	Organigrames Normatives, normes de qualitat i estàndards Descripció de llocs de treball o perfils
Manuals d'instruccions tècniques	Documentació de procediments d'usuari i de negoci Models de processos de negoci Manuals de formació
Productes del sistema d'informació	Formats d'entrades i d'eixides Manuals d'usuari Exemples de documents i formularis Exemples d'informes
Documents tècnics del sistema actual	Documentació de la definició de requisits Diagrames de context Diagrames de flux de dades Models de dades Models d'arquitectura i de plataformes de comunicacions i tecnològiques

El resultat és un document os s'analitzen els aspectes positius del funcionament actual (punts forts) i aquells que requereixen una millora (punts febles), tant des del punt de vista de la seua informatització com a vegades d'organització. També es poden utilitzar models gràfics de processos com IDEF0, que no són objecte d'estudi en aquesta assignatura com IDEF0.

2.2. Tècniques d'investigació de requisits

Les tècniques que s'inclouen en aquest apartat serveixen fonamentalment per a recopilar informació. En el cas que es considera en aquest tema s'utilitzen per a definir necessitats que el programari i altres components del sistema han de cobrir, però també podrien utilitzar-se per a definir objectius inicials, conèixer el sistema en funcionament o millorar un procés de producció. La descripció que s'inclou de cada tècnica es fa amb poc detall, només per a donar una guia bàsica del seu ús. Es pot trobar més detall en la bibliografia i en publicacions exhaustives dedicades a algunes de les tècniques, que poden ser usades en un gran nombre d'àrees científiques i empresarials.

2.2.1. Entrevistes

Les entrevistes, juntament amb les reunions, són una de les tècniques més habituals i usades per a recopilar informació sobre les necessitats del futur sistema. L'objectiu de les entrevistes és obtenir informació que siga d'utilitat a l'enginyer informàtic o per a qualsevol altre membre de l'equip de desenvolupament.

Les entrevistes tenen com a avantatge, entre altres coses, el contacte personal amb els futurs usuaris, que proporcionen més informació i de més valor que altres mètodes, i poden modificar-se, en temps i contingut si es considera que l'entrevistat és de més o menys valor del que s'havia pensat. Però també s'han de tenir en compte alguns desavantatges com ara que consumeixen molt més temps que altres mètodes, l'avaluació de la informació obtinguda pot ser difícil, a vegades no es pot quantificar, etc.

A continuació es mostra una guia bàsica dels passos a seguir per a fer entrevistes adequades i productives:

- Identificar fonts d'informació i persones que s'han de entrevistar: informar-se sobre les diferents responsabilitats, les relacions existents i l'organització en departaments o àrees relacionades en principi amb el sistema.
- Identificar les activitats relacionades amb cadascuna de les persones: és important identificar prèviament fonts d'informació que permeten enfocar i centrar l'entrevista i planificar-la.
- Preparar les entrevistes: preparar un guió tenint en compte els diferents tipus d'usuaris que tindran relació amb el futur sistema. Organitzar les preguntes per temes i anar de preguntes més obertes a més concretes.
- Concertar l'entrevista, lloc, data i hora. Elegir un moment adequat per a l'usuari pot contribuir a un millor resultat.
- Realitzar l'entrevista: prendre notes, ser proactiu, professional i respectuós, saber escoltar, no perdre el control del tema de manera objectiva i productiva.
- Resumir les entrevistes en un termini curt des de la seua realització.

2.2.2. Qüestionaris

Els qüestionaris són documents que sol·liciten informació específica als seus destinataris. Són més impersonals que les entrevistes però més objectius i proporcionen una altra forma d'investigar la informació sobre el sistema que es vol desenvolupar. Els qüestionaris poden dissenyar-se i enfocar-se a problemes concrets, tenint en compte tant els nivells dels destinataris com els temes que s'han de tractar.

És convenient utilitzar-los quan es desitja obtenir informació general sobre algun aspecte del sistema o bé es desitja conèixer la resposta d'un gran nombre de persones, els seus interessos o actituds respecte a un determinat aspecte.

Els qüestionaris proporcionen una informació més objectiva que les entrevistes, encara que normalment també és menys completa. Però, el temps necessari per a obtenir informació d'un gran nombre d'usuaris és menor que si es fan entrevistes.

2.2.3. Fonts externes

Quan l'enginyer informàtic s'enfronta amb un entorn nou (empresa d'un sector on no s'ha treballat, nous sistemes o noves tecnologies), consultar textos, llibres, publicacions i formar-se és una de les seues obligacions per tal d'abordar el problema amb un coneixement total i poder realitzar cadascuna de les activitats amb correcció i qualitat.

Quan el sistema en funcionament té molts problemes, es vol realitzar un canvi total o bé el sistema s'implanta en una nova organització pot ser convenient consultar fonts externes. Aquestes fonts poden ser altres empreses del sector, o associacions relacionades o empreses consultores que aconsellen i recomanen sobre certs aspectes estratègics i operacionals del futur sistema.

2.2.4. Desenvolupament conjunt d'aplicacions

Aquesta tècnica es coneguda com JAD (*Joint Aplication Development*) i promou la cooperació i el treball en equip com a alternativa a les entrevistes individuals. Consisteix a organitzar i realitzar un conjunt de reunions de treball (*Workshops*) agrupades en el temps (tres o quatre dies) en les quals participen usuaris experimentats i enginyers informàtics. L'ús de tècniques audiovisuals, posades en comú, grups de discussió i documentació WYSIWYG,[2] poden proporcionar resultats més eficients i correctes.

Aquests tallers tenen diferents avantatges front a altres tècniques individuals:

- Els usuaris participants se senten més involucrats des del principi, ja que consideren el producte de programari com propi.
- Tot l'equip de treball participa i supervisa l'especificació de requisits, que són revisats per tots els usuaris de manera conjunta.
- Es fa un ús més efectiu del temps.

Malgrat aquests avantatges no s'usen gaire, principalment per falta de temps o perquè resulta complex organitzar aquestes reunions de tres o quatre dies (Piattini, 2004).

3. Documentar els requisits del sistema

Determinar i definir tots els requisits d'usuari és essencial perquè el document on s'hi arrepleguen és la base per al desenvolupament posterior del sistema. A més a més, el document serveix per a validar el programari i dur a terme l'acceptació del producte, és a dir, comparar els resultats obtinguts amb les especificacions del que es volia obtenir. Per a definir i documentar els requisits és necessari el treball conjunt dels clients/usuaris i de l'enginyer informàtic.

2. What you see is what you get.

Com en quasi totes les activitats cal generar un document que incloga tots els aspectes importants i tota la informació rellevant sobre l'activitat desenvolupada i que servisca de punt de partida per a les següents activitats. Existeixen diferents propostes i mètodes per proporcionar una documentació completa i verificable. En aquest apartat estudiarem algunes de les més usades que poden ser complementàries entre si. El document final és una mena de contracte entre els usuaris i els enginyers informàtics (contractats externament o empleats de l'empresa), on es defineixen els objectius i l'abast del producte a desenvolupar i quines condicions ha de complir. Per tant, és important que tinga uns continguts bàsics i unes característiques adequades.

El contingut del document on s'especifiquen els requisits del sistema ha de complir les següents condicions:

- Reflectir únicament els comportaments externs del sistema, és a dir, la visió que els futurs usuaris tenen del programari i del maquinari.
- Ha de ser fàcilment modificable.
- Ha de servir com a referència quan es realitzen manteniments del sistema.
- Ha de tenir-se present durant tot el cicle de vida del programari. Els analistes i programadors han de consultar-lo per a tenir en compte el que s'espera que el sistema realitze.
- Servir com a contracte per establir el que s'espera del producte que es vol construir, per tant, cal que el seu contingut tinga l'aprovació del usuaris i responsables de les àrees implicades.
- Ha d'incloure una taula de continguts, glossari de termes utilitzats, i una possible previsió de canvis en els requisits inicialment definits.
- No és un document de disseny, ha de mostrar què ha de fer el sistema sense especificar la forma o el suport en què es realitza. Però, poden tenir-se en compte les necessitats de maquinari genèrics per tal de donar suport al sistema o alguns processos específics. Per exemple: nous servidors, dispositius de comunicació, dispositius automatitzats d'introducció de dades, etc.
- Els requisits han de ser condicions precises sense ambigüitats ni contradiccions, han de cobrir els objectius i han de ser viables.

El document pot organitzar-se de diferents maneres depenent, en part, de les tècniques utilitzades, com es mostra en la taula 3.2.

Taula 3.2. Proposta d'índex de documentació de requisits.

Documentació de requisits	
Continguts	**Tècnica**
Títol, autors, data Controls de versions i canvis Índex de continguts	Tècniques de redacció i producció de documents escrits
Introducció Objectiu del document Com s'ha desenvolupat (qui ha participat i quines tasques s'han fet) Com està organitzat el document	Tècniques de redacció i producció de documents escrits
Estudi del sistema / processos actuals (si aporta valor) Descripció dels principals processos analitzats, models gràfics, punts fort i febles, documents i l'ús que se'n fa, etc. Diagnòstic del problema Proposta de millora	Models de processos de negocis (IDEF0) Diagrama de casos d'ús de processos (UML) Anàlisi de cost/benefici Reenginyeria de processos: AS-IS_TO-BE[3]
Catàleg de requisits Models de requisits / Requisits funcionals Requisits de dades / informació Altres requisits (de seguretat, d'accessibilitat, navegabilitat, etc.) Especificació de la plataforma de maquinari	Diagrama de casos d'ús de requisits (UML) Plantilles de definició de requisits (Durán, 1999)
Bibliografia, glossari, etc.	

En aquesta assignatura es veurà com realitzar casos d'ús per a especificar el funcionament del programari des del punt de vista dels requisits del comportament del producte (que ha de fer el programari) i plantilles que permeten completar la informació.

4. Diagrames de casos d'ús

UML (*Unified Modelling Language*) és un llenguatge definit com un estàndard de modelització per l'OMG® (Object Management Group®) que permet visualitzar, representar, construir i documentar els artefactes d'un producte de programari. UML proposa diferents models i notacions que permeten representar el sistema informàtic des de diferents punts de vista, diferents nivells de detall i amb diferents objectius. Aquests models evolucionen i es modifiquen a mesura que es duen a terme les diferents fases del desenvolupament del sistema i s'han d'usar durant tot aquest procés com a suport i guia per a obtenir, de manera eficient, un producte de qualitat i correcte. Els models en general, i en aquests cas els d'UML, són similars als diferents plànols (amb les seues vistes i detalls) que dissenyen els arquitectes i que s'usen com a base per a tot el procés de construcció d'un edifici (Maciaszek, 2007: cap. 3).

3. Aquestes tècniques no s'estudien en aquesta assignatura.

El diagrama de casos d'ús és un dels diagrames més representatius d'UML que s'usa per a modelar el comportament del sistema. Un model de comportament representa transaccions de negoci, operacions, algorismes sobre dades, és a dir, la vista dinàmica, allò que produeix transformacions sobre els objectes o les dades del sistema. Un model pot incloure un diagrama o més, en un model de comportament poden haver-hi diagrames de CU i altres diagrames d'UML, com ara els diagrames d'activitats o de seqüència.

En aquesta assignatura dintre del model de comportament del sistema s'estudien els diagrames de casos d'ús per a representar la funcionalitat del sistema informàtic i poder documentar i analitzar els requisits del sistema. L'objectiu és documentar el comportament del sistema informàtic des del punt de vista de l'usuari. És a dir, el que el sistema ha de fer com a resposta a un esdeveniment extern per a cobrir els requisits d'usuari.

Es considera un usuari del sistema qualsevol cosa aliena o entitat externa al mateix i que interactua directament amb ell, com per exemple una persona que realitza transaccions, un altre sistema d'informació, un dispositiu hardware, etc. Els diagrames de casos d'ús són fàcils de comprendre, intuïtius i senzills i en una primera versió bàsica proporciona una vista d'alt nivell del que ha de fer el sistema.

Poden fer-se models de casos d'ús de diferents nivells de detall, de manera que amb menys detall es captura el comportament del sistema de manera completa i, si es vol incloure més detall, s'haurà de fer representant diferents parts de la funcionalitat del sistema en diferents diagrames.

4.1. Elements dels diagrames de casos d'ús

Els elements o constructors bàsics que es poden utilitzar en un diagrama de casos d'ús són: **actors, casos d'ús i relacions.** El diagrama de casos d'ús és una representació visual d'actors i casos amb definicions i especificacions complementàries.

Un actor representa un rol que pot assumir un grup de persones, una persona o dispositius físics externs al sistema, que interactuen amb el producte de programari. Per exemple, un client en un sistema de compres per Internet. Un client, un venedor o un repartidor poden ser actors diferents, però també ser la mateixa persona. S'anomenen en singular i es representen amb una icona com es veu en la figura 3.2. Poden haver-hi altres representacions específiques per actors no humans.

Per a crear un cas d'ús el primer pas és identificar els actors, fent una llista de les persones que interactuen amb el sistema, i classificar-los segons el rol que hi juguen en cada moment.

Els actors d'un diagrama de cas d'ús són externs al sistema i han d'interactuar amb el sistema (consultar/modificar l'estat del sistema) directament.

Un cas d'ús representa una seqüència d'accions, incloent-hi les possibles variants, que executa el sistema per a produir un resultat observable, de valor i interessant per a un actor en particular (considerant l'objectiu del sistema). Un cas d'ús descriu quin serà el comportament del sistema (un servei) sota diverses condicions i com respon el sistema a una demanda dels actors. És a dir, mostra el que ha de fer el programari des del punt de vista dels usuaris, independentment de la seua implementació tecnològica, i modelitza un servei proporcionat pel sistema que aporta un valor afegit «notable» per a l'actor.

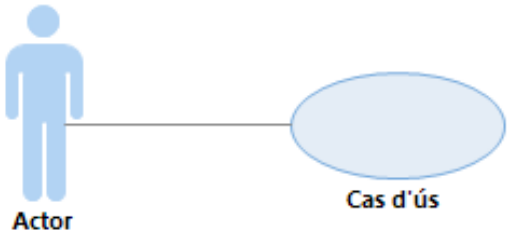


Figura 3.2. Exemple de representació d'un actor, un cas d'ús i una associació amb IBM RSA

Un **cas d'ús** representa diferent situacions depenent de les condicions de la interacció o del comportament específics dels actors, cada situació es denomina un ***escenari***.

Els casos d'ús s'anomenen amb un verb que identifica l'acció o funcionalitat que es podrà implementar en el sistema i l'objecte sobre el qual actua. L'objecte es veurà modificat perquè canvia d'estat o s'actualitza la seua informació. El nom de cada cas d'ús ha de ser únic.

Les característiques dels casos d'ús són:

- Representen funcionalitat del sistema.
- Denoten només els comportaments essencials del sistema i no han de ser ni massa genèrics (resultat observable) ni massa específics.
- Descriuen que fa un sistema, però no especifiquen com ho fa.

Per a definir els casos d'ús s'ha de considerar:

- Què volen fer els actors amb els sistema, principals funcions o requisits.
- Quin és l'objectiu dels actors i quins resultats volen obtenir.
- Sota quines condicions s'executaran els casos d'ús i quines exempcions hi pot haver.
- Quina informació del sistema pot obtenir, consultar o modificar un actor.
- Quins canvis ha de comunicar un actor al sistema.
- Quins canvis i quina informació vol un actor que el sistema li comunique.

Una **relació** representa la connexió que existeix entre l'actor i el cas d'ús. Pot ser navegable indicant en quin sentit flueix la informació. També poden haver-hi relacions entre casos d'ús per indicar que un cas d'ús pot activar o necessita d'un altre.[4]

4. La forma de connectar casos d'ús entre ells i quan es pot fer s'estudiarà en altres assignatures.

Un actor es relaciona amb un cas d'ús si l'actor interacciona amb el sistema per dur a terme la tasca que el cas representa i amb l'objectiu d'obtenir un resultat observable i de valor per a ell, és a dir, la interacció de l'actor és imprescindible per a la realització del cas d'ús. En la figura 3.3 es mostra un exemple de diagrama de casos d'ús complet.

4.2. Com fer el diagrama de casos d'ús

Els passos que cal seguir per a desenvolupar un diagrama de casos d'ús simple, són els següents:

1. Identificar els actors del cas i pensar el paper que representen (imaginar una instància).
2. Identificar els missatges necessaris entre els actors i el sistema informàtic a desenvolupar.
3. Identificar la funcionalitat (a alt nivell) que el sistema proporcionarà des del punt de vista dels usuaris i definir els casos d'ús (de manera general i clara).
4. Dibuixar el diagrama de casos d'ús.
5. Descriure els casos d'ús i els actors.
6. Validar amb els usuaris competents.

Exemple

Requisit	Actors	Cas d'ús
Una persona podrà registrar-se com a soci per Internet o bé sol·licitar que l'administratiu del club el done d'alta com a soci.	Soci Administratiu	Registrar soci
L'oferta d'activitats del club es podrà consultar obertament.	Persona/Públic	Consultar oferta d'activitats
El responsable del club podrà introduir la informació de les activitats que s'ofereixen als socis i fer la seua programació temporal. És a dir, detallar dies, horaris, preus, data d'inici data de fi, monitor i sales del club on s'imparteixen.	Responsable	Programar activitats
Un soci pot consultar i anul·lar les activitats en què està inscrit i inscriure's en noves activitats.	Soci	Mantenir inscripcions / Fer inscripcions a activitats
Cada mes es podrà realitzar un procés per a generar les factures a tots els socis i cobrar les activitats a les quals està inscrit cadascun d'ells.	Administratiu	Facturar
El responsable del club podrà mantenir la informació de monitors del club.	Responsable	Mantenir monitors

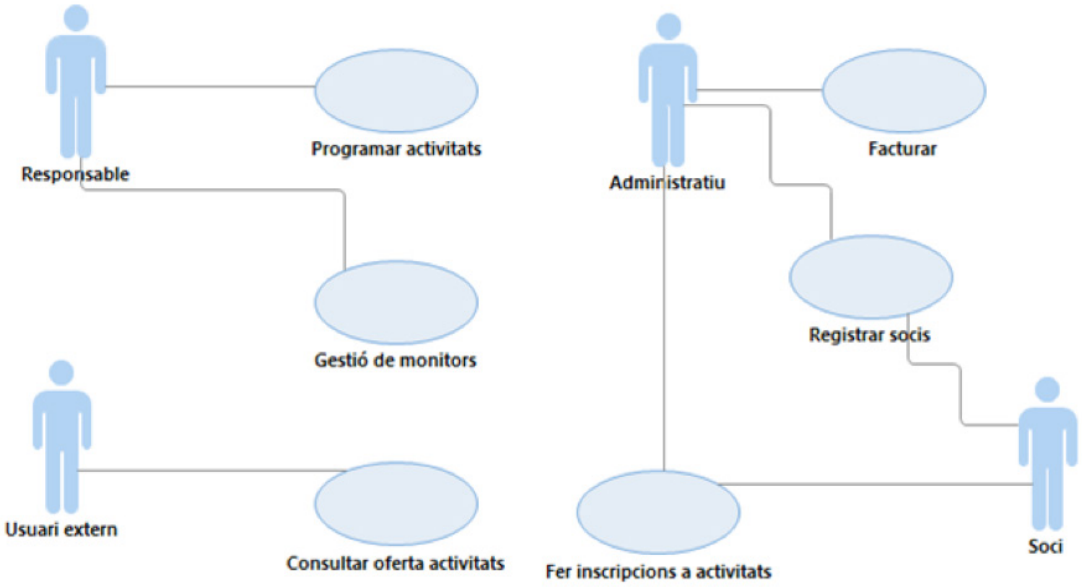


Figura 3.3. Exemple de diagrama de casos d'ús

4.3. Descripció dels casos d'ús

Existeixen diferents propostes per a documentar els casos d'ús, la majoria basades en plantilles més o menys completes. En aquesta assignatura es proposa una plantilla simplificada que permet donar informació bàsica per a comprendre el funcionament del cas d'ús.

És important tenir en compte que s'està descrivint el funcionament del sistema informàtic des del punt de vista dels usuaris. Per una part, s'han de descriure les accions que corresponguen a l'execució de programari (llegir, mostrar, comprovar, etc.), no accions que realitzaran els usuaris (entregar el producte al client, si no és el cas d'un dispositiu automàtic que ho fa, anar i veure, etc.). Per una altra part, no s'ha de donar un nivell de detall d'implementació del sistema ni aspectes que durant aquesta fase no són rellevants (no és correcte dir *s'accedeix a la taula de clients amb el codi per comprovar*, és millor dir *comprova si el client*).

Taula 3.3. Exemple de plantilla per a descriure casos d'ús.

Descripció del cas d'ús	
Identificador: CU01 *(exemple de numeració)* **Nom:** *Programar activitats (exemple d'acció/verb ha de ser informatitzable)* **Autor:** **Font:**	**Data creació:** **Data revisió:** **Data aprovació:** **Versió:**
Descripció: *L'usuari amb rol de responsable del club pot donar d'alta activitats esportives, assignar-los un calendari, una ubicació, consultar i modificar la programació d'activitats ja donades d'alta.*	
Passos seqüència normal *Alta nova activitat*: *El responsable selecciona donar d'alta una nova activitat* *El sistema proporciona informació sobre tipus d'activitats i deixa seleccionar el tipus* *El responsable introdueix les dates i l'horari de l'activitat i confirma* *El sistema mostra les sales disponibles en eixes hores* *El responsable selecciona sales i confirma* *El sistema registra l'activitat* **Exempcions** Si no hi ha sales disponibles en l'horari introduït en el pas 4, el sistema deixa tornar enrere i permet introduir altres horaris **Precondició**	*Accions que se segueixen per dur a terme la principal funcionalitat (o la més habitual)*
Comentaris	

Exercici: Modifiqueu el cas d'ús per a mostrar les hores disponibles abans d'introduir l'horari.

5. Plantilles de documentació de requisits

Per a completar l'especificació dels requisits és important tenir en compte la funcionalitat que, com s'ha descrit, es pot representar amb els diagrames de casos d'ús, i també aspectes específics de les dades (estructura, continguts i relacions) i altres condicions que el producte de programari ha de complir. En aquest apartat es proporcionen dues plantilles simples, (vegeu taules 3.4 i 3.5) obtingudes a partir de la metodologia proposada en Durán (2010):

1. **Taula 3.4. Definició de requisits de dades** on s'inclouen les condicions i detalls de dades d'entrada, dades que el programari gestionarà i actualitzarà i informació que mostrarà. No s'especifica en aquesta activitat del

projecte de desenvolupament de programari en format físic o tipus de dada (per exemple string, varchar, etc,), encara que en algun cas pot ser necessari fixar el format visual (per exemple es pot dir d'una dada: haurà de tenir com a mínim 30 caràcters o ser un número amb dos decimals).

2. **Taula 3.5. Altres requisits**: aquesta taula es pot adaptar i usar per a descriure requisits de diferent tipus referents a condicions que el programari ha de complir relacionades amb temes de seguretat i accessibilitat.

Els requisits de maquinari també es podrien especificar utilitzant plantilles semblants a aquestes.

Taula 3.4. Proposta de descripció de requisits de dades.

Requisit de dades	
Identificador: RD01 *(exemple de numeració)* **Nom:** Soci (exemple *objecte o bloc de dades o d'informació)* **Autor:** **Font:**	**Data creació:** **Data revisió:** **Data aprovació:** **Versió**:
Tipus: Dades a mantenir / Dades d'entrada / Dades eixida (*elegir-ne un*). **Dades específiques a mantenir:** nom, cognoms, domicili, DNI; telèfon, nom-usuari, pwd, data-alta, data-baixa, data-naixement, número de compte corrent	
Detall: **DNI:** pot estar en blanc per als menors, pot tenir formats diferents per a admetre socis de diferents països. **Telèfon:** haurà de donar la possibilitat de registrar dos números. **Número de compte corrent:** ha de ser un número amb format bancari correcte	
Comentaris	

Taula 3.5. Proposta de descripció de requisits de tipus nnn.

Altres requisits: nnn	
Identificador: RA01 (exemple de numeració*)* **Nom:** Seguretat d'accés **Autor:** **Font:**	**Data creació:** **Data revisió:** **Data aprovació:** **Versió**:
Descripció:	

6. Verificar i validar requisits (V&V)

A mida que s'avança en l'especificació del que el programari haurà de fer, i del projecte en general, s'han de revisar els productes i documents que s'obtenen per a controlar que no es produïsquen inconsistències, omissions o ambigüitats. És a dir, cal comprovar que:

- El que es defineix i es genera es fa de manera correcta, sense errors, ni inconsistències, i seguint els estàndards de qualitat en cada cas. Estem fent les coses bé?
- El que es defineix i es genera és el que els usuaris necessiten, i es valida per a tenir l'acceptació dels resultats per part dels usuaris. Estem fent el producte correcte?

6.1. Verificar

En la definició de requisits, **verificar** significa comprovar que la definició de requisits s'ha fet de manera correcta, és a dir, que se segueixen els estàndards de qualitat; per tant, entre altres aspectes cal comprovar si cada requisit:

- Es consistent amb els objectius generals i està dintre de l'abast del projecte.
- Es necessari, és a dir, representa una funcionalitat essencial per a aconseguir l'objectiu del producte de programari.
- S'ha especificat a un nivell d'abstracció adequat i equilibrat amb la resta.
- Està definit sense ambigüitat, és comprensible i determina una condició clara que es pot programar amb les tecnologies disponibles o que hi ha a l'abast del client.
- No entra en conflictes amb altres requisits, és a dir, no pot haver-hi requisits contraposats o que la implementació d'un impedisca la d'un altre.
- Es pot comprovar una vegada implementat.
- La documentació està desenvolupada de manera adequada pel que fa al nivell de detall tenint en compte el volum i la grandària del producte.

Taula 3.6. Verificació de consistència requisits de dades / casos d'ús.

	Casos d'ús					**Falta**
Requisits de dades	Registrar soci	Programar activitats	Gestió de Monitors	Fer Inscripcions	Facturar	
R1: Soci	Alta			Es consulta	Es consulta	Modificar
R2: Activitat		Alta i Modificació		Consulta i modifica	Consulta	Modificar
R3: Inscripció				Crea Modifica		
R4: Factura					Es crea	Consultar Innecessari

En la taula 3.6 es mostra un exemple d'una taula per a verificar si els casos d'ús i les taules de requisits de dades són consistents. Cada requisit de dades es crea quan hi ha una necessitat d'enregistrar, mostrar o consultar informació, per tant ha d'haver-hi un cas d'ús que cree, modifique o consulte eixa informació. Poden

haver-hi excepcions si la informació no es pròpia del producte de programari per desenvolupar, és a dir, es crea amb una altre programari.

Com a resultat d'aquesta taula es pot indicar entre altres coses que hi faltaria un requisit de dades relacionat amb el cas d'ús Gestió de Monitors, indicant que cal tenir informació de monitors, per poder-la donar d'alta i modificar en el CU Gestió de Monitors, així com consultar en l'assignació de monitors a activitats. També s'hauria d'analitzar per què els socis només es donen d'alta, i no hi cap cas d'ús per poder modificar les seues dades en cas de produir-se algun canvi.

Una vegada verificat que la documentació és consistent i completa s'ha de validar amb els usuaris o clients que el que s'ha documentat és el que ells volen del producte. Es pot demanar l'aprovació formal del document mitjançant una gestió documental formal o reunions. L'acceptació permet continuar amb les següents fases del desenvolupament del producte de programari.

6.2. Validar

Validar els requisits consisteix a comprovar que el document de requisits i els models desenvolupats reflecteixen el que els usuaris volen que el producte de programari duga a terme. Aquesta validació serà una acceptació per part dels usuaris del que farà el producte de programari. Per a validar es comprovarà que cada requisit, cas d'ús i necessitat documentada correspon a una necessitat detectada pels usuaris i que cada necessitat detectada pels usuaris està registrada de manera correcta en el document de requisits sempre que no entre en conflicte amb la verificació.

Per a validar els models de casos d'ús caldrà comprovar que cada necessitat que implique una funcionalitat en el sistema està representada per un o més casos d'ús i que cadascun correspon a una, o part d'una, de les necessitats de funcionalitat identificades.

Aquesta tasca de verificació i validació proporcionarà la seguretat de partir d'informació correcta per a seguir en les següents fases de modelització del programari. Per a validar i verificar els document de requisits es poden gastar matrius o taules en les quals es comprove la consistència dels models i de les necessitats del usaries des alt nivell de detall.

7. Resum

En aquest tema es proposen algunes de les tasques fonamentals que cal realitzar per a desenvolupar una correcta definició de requisits. Es dóna especial rellevància al fet d'aconseguir informació útil i correcta per tal d'identificar les necessitats de programari. S'estudien diferents tècniques d'ajuda a la recopilació i extracció

de requisits, i d'avaluació del sistema en funcionament. Es proposa una organització del document de definició de requisits.

Com a contingut fonamental es descriu què és i quins elements es representen en un diagrama de casos d'ús per a desenvolupar el model de requisits i es proporcionen exemples de plantilles per a documentar els casos d'ús, els requisits de dades i altres requisits.

Finalment, es proporciona una guia senzilla per validar la correcció dels requisits definits.

Activitats complementàries

1. Llegiu el capítol 3 de Maciaszek (2007), disponible en la biblioteca.
2. Busqueu almenys una altra tècnica que puga ser utilitzada per a recopilar informació per definir requisits d'un sistema informàtic. Feu un xicotet resum de com pot aplicar-s'hi i de la seua utilitat.
3. Feu una llista de qüestions a tractar en entrevistes fictícies als vostres usuaris el cas pràctic que es proposa al final del llibre.
4. Especifiqueu i feu un gràfic de la plataforma de maquinari que penseu que podria ser necessària per al cas pràctic proposat.

CAPÍTOL 4

Anàlisi

Objectius

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Entendre què significa el concepte d'anàlisi
- Conèixer i saber desenvolupar els diagrames d'activitats i de classes d'UML corresponents a l'anàlisi del programari.
- Saber documentar l'anàlisi d'un sistema amb UML.
- Saber validar i verificar els diferents models UML dissenyats per al mateix projecte de desenvolupament de programari.

1. Introducció

L'objectiu principal de l'anàlisi del sistema és transformar la definició dels requisits d'usuari en una especificació del programari, tenint en compte a més, els objectius definits, les restriccions i les necessitats particulars de l'entorn.

> El terme anàlisi, aplicat a sistemes, significa descompondre'l en els seus components, per a estudiar cadascun d'aquests, tant com un ens aïllat, com en interacció amb la resta. Per a ser útil, a l'anàlisi li ha de seguir la síntesi, que consisteix a unir els components del sistema per a veure com funcionen en conjunt. (Piattini, 2004)

Dins de les organitzacions i empreses, el terme anàlisi i disseny de sistemes es refereix al procés d'examinar la situació d'una empresa amb el propòsit de millorar-la amb mètodes i processos més adequats. L'anàlisi de sistemes és el procés d'interpretació dels fets, la identificació dels problemes i l'ús de la informació per a millorar el funcionament del sistema. El resultat de l'anàlisi és una especificació del programari, és a dir, una documentació completa i precisa de què ha de realitzar el sistema per a cobrir els requisits d'usuari.[1]

Per realitzar i documentar l'anàlisi de manera correcta és necessari utilitzar mètodes i tècniques que permetisquen representar el programari i la resta de components d'un sistema informatitzat. Aquests mètodes, com ja s'ha vist en temes anteriors, poden utilitzar tècniques gràfiques completades amb descripcions adequades que ajuden a definir com serà el producte final que es vol desenvolupar.

2. Model de l'anàlisi amb UML

En aquest apartat es descriu com es pot realitzar l'anàlisi en el procés de desenvolupament de programari utilitzant UML. Primer es descriuen dos dels diagrames que es poden utilitzar a l'anàlisi: el diagrama d'activitats per a mostrar el comportament del programari des d'un punt de vista alt i el diagrama de classes per a representar la informació que s'ha de poder emmagatzemar per a donar suport a la funcionalitat del programari. Després d'explicar aquests diagrames, es detallen quins passos es poden seguir desenvolupant a l'anàlisi. En aquesta assignatura s'expliquen dos diagrames per a l'anàlisi, però pot haver-hi projectes on siga convenient usar-ne altres, o bé que algun dels que es veu ací no siga necessari. Per exemple, un altre model que podria ser adient és el diagrama d'estats per a representar aspectes de comportament dels objectes d'una classe amb més detall.

La norma bàsica és dur a terme aquells models i diagrames que aporten valor afegit al projecte, bé perquè ajuden a la comprensió de la solució entre tots els participants del projecte, bé perquè proporcionen una visió que millora el desenvolupament del producte. En aquesta assignatura s'usaran per a l'anàlisi els diagrames de classes i el d'activitats d'UML. El diagrama de casos d'ús, ja descrit en el tema 3, es pot considerar també per a documentar l'anàlisi si en la definició de requisits s'ha fet amb poca profunditat i es completa en aquesta fase del projecte.

1. Part del treball d'anàlisi ja es du a terme en la fase prèvia de «Definició de requisits».

2.1. Diagrama d'activitats

El diagrama d'activitats és un dels diagrames de comportament d'UML2 que es pot utilitzar durant la fase d'anàlisi del sistema informàtic.

En la figura 4.1 es mostra un exemple de diagrama d'activitats, el qual especifica la lògica de negoci d'un cas que representa la inscripció d'un soci a activitats del Club Esportiu (s'ha fet a un nivell de detall molt baix per poder mostrar tots els constructors del diagrama). L'exemple mostra la major part dels elements bàsics d'un diagrama d'activitats.

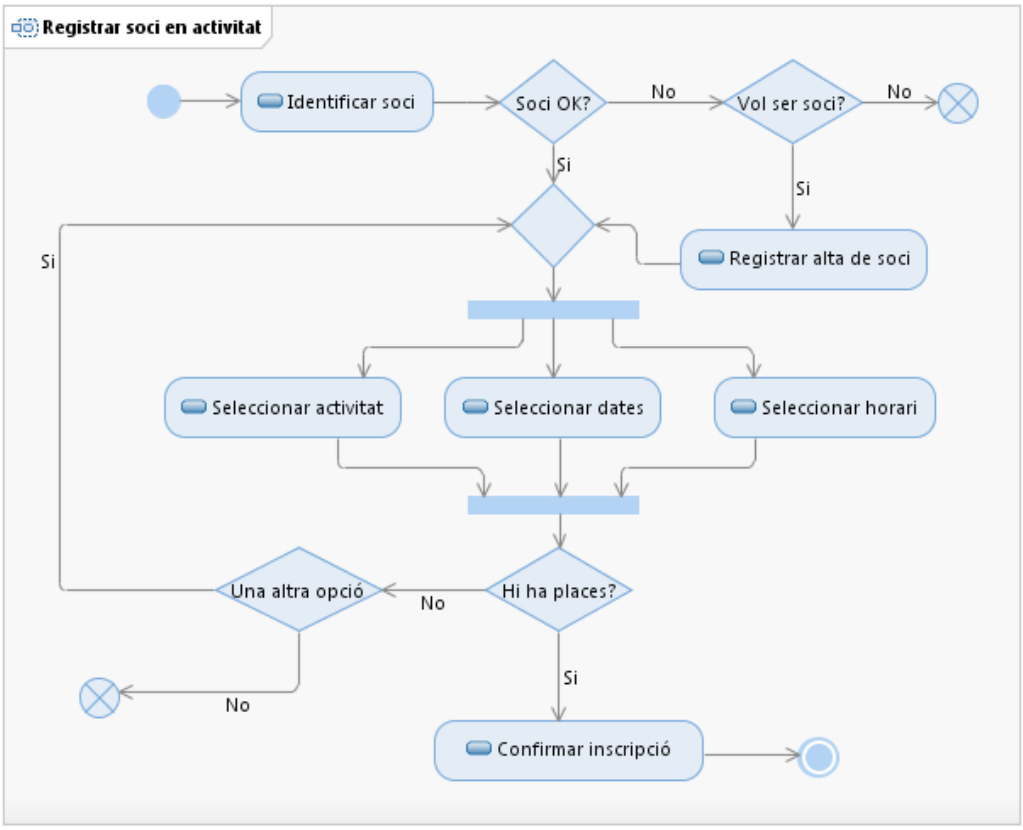


Figura 4.1. Exemple de diagrama d'activitat d'un procés del Club Esportiu

El principal objectiu d'aquest diagrama és mostrar les connexions entre accions amb la finalitat de comprendre comportaments complexos. De manera que és útil tant per a mostrar processos de negoci com per a detallar casos d'ús, i fins i tot especificar el flux d'un programa. En l'anàlisi se sol utilitzar aquest diagrama per tal de mostrar la lògica de negoci de l'aplicació informàtica que es desenvoluparà.

Els elements o constructors bàsics que es poden utilitzar en un diagrama d'activitats són les activitats, les transicions i els nodes de control.

Una **activitat** és un conjunt d'accions, de forma que una activitat sempre es pot dividir en altres activitats més menudes o en accions, les quals per definició són

aquelles execucions que ja no es poden descompondre més. Per tant, en un diagrama d'activitats a nivell de lògica de negoci no haurien d'aparèixer accions, al contrari del que passaria en un diagrama d'activitats que mostrara el flux d'un programa on la majoria d'execucions seria del nivell d'accions.

Transició. Una transició representa el pas d'una acció a un altra, automàticament el final d'una acció dispara la transició, la qual produeix el començament d'un altra activitat o acció.

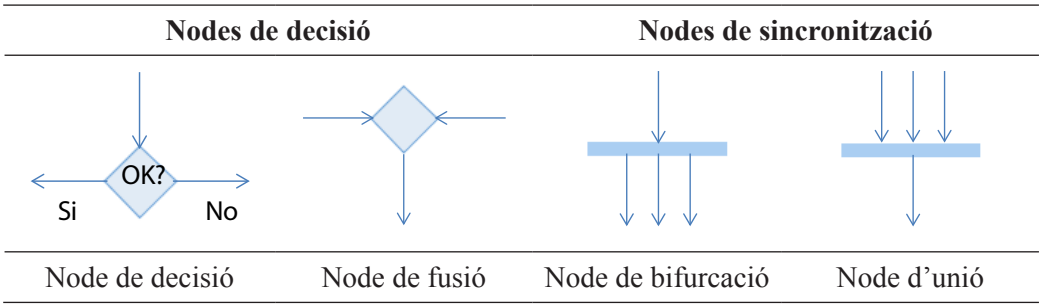


Figura 4.2. Representació dels nodes de control de decisió i de sincronització

Nodes de control. Un node de control representa la coordinació de fluxos entre nodes. En la figura 4.2 es mostren els diferents tipus de control que es poden utilitzar en un diagrama d'activitats d'UML2. Existeixen quatre tipus de nodes de control:

- **Node de decisió** (*decision node*) i **node de fusió** (*merge node*): el node de decisió s'empra per a obrir la condició (s'executa una de les opcions) i el de fusió per tal de tancar-la (qualsevol opció fa continuar el flux d'execució). En general, se'ls anomena nodes de decisió. Es representen amb un rombe com es mostra en les figures 4.2 i 4.3.
- **Node de bifurcació** (*fork*) i **node d'unió** (*join*): s'utilitzen per a indicar l'establiment de camins d'execució concurrent. El node de bifurcació s'empra per a obrir la concurrència (diferents accions que s'executen a la vegada o en paral·lel) i el node d'unió per a tancar l'execució d'accions (totes han d'arribar per a seguir el flux). En general, se'ls anomena nodes de sincronització.
- **Node inicial**: indica el començament de la seqüència d'accions i per tant del flux de treball.
- **Node final**: indica la fi de la seqüència d'accions i a la seua vegada pot indicar: Final d'activitat (quan realment acaba tot el flux de treball), o Final de flux, que significa que pot acabar part de la seqüència d'accions, sense que s'haja completat tot el flux de treball representat en el diagrama d'activitats.

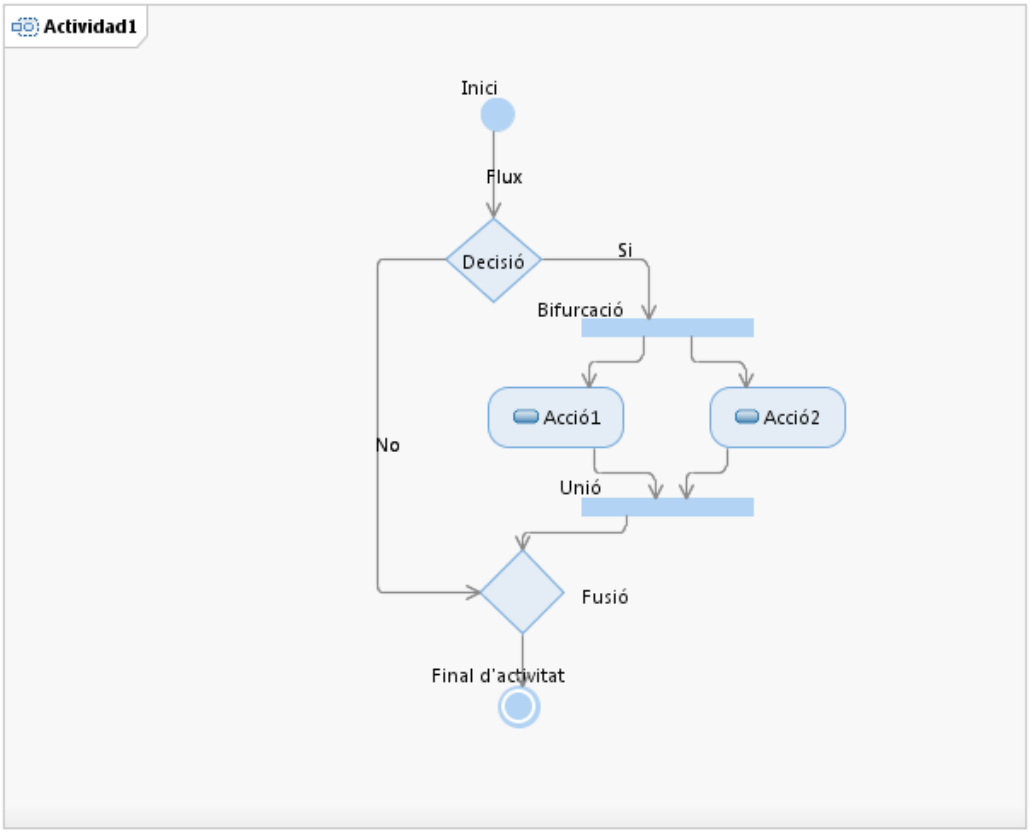


Figura 4.3. Constructors bàsics d'un diagrama d'activitats

Exemples del seu ús en un diagrama d'activitats es poden veure en la figura 4.3.

Dos fluxos de control que arriben a una activitat vol dir que són ambdós necessaris per a executar-la. Per una altra part dos fluxos de control que ixen d'una activitat vol dir que s'executen ambdós camins. Però, encara que en alguns casos els nodes de bifurcació o fusió poden obviar-se, és convenient fer ús d'ells quan la representació puga donar lloc a confusió.

2.2. Diagrama de classes

El diagrama de classes serveix tant per a mostrar l'estructura de la informació del sistema, com el comportament del mateix a través de les operacions de les classes. És un dels diagrames que més s'utilitza al llarg de totes les fases de desenvolupament de programari amb UML, ja que es va refinant en les successives iteracions d'aquest procés. Encara que en aquest tema es descriurà el diagrama al nivell de l'anàlisi, una vegada refinat es pot arribar a convertir en el conjunt de classes implementades en un determinat llenguatge de programació orientat a objectes, com per exemple Java, el qual constitueix el sistema informàtic com la solució del problema plantejat.

A nivell de l'anàlisi el diagrama de classes s'utilitza per a mostrar els principals elements del domini o l'esquema lògic de la base de dades i la representació de la funcionalitat representada en els casos d'ús mitjançant les operacions.

Els elements o constructors bàsics que es poden utilitzar en un diagrama de classes són les classes, amb atributs i operacions, i les relacions. En la figura 4.10 es mostra un exemple senzill d'un taller de reparacions.

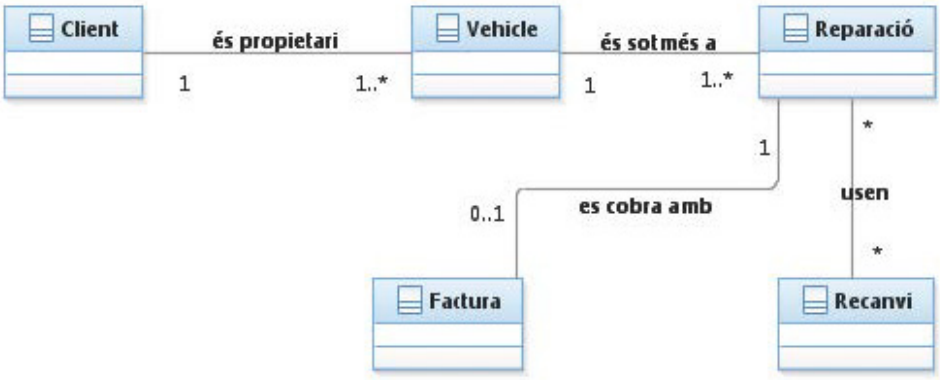


Figura 4.4. Exemple de diagrama de classes

Classe. Una classe és un conjunt d'elements o instàncies que tenen un rol equivalent en el sistema, és a dir, que comparteixen característiques (denominades atributs), operacions i semàntica (tenen un mateix significat per al sistema). A nivell d'instància, les instàncies de les classes s'anomenen **objectes.** La representació gràfica en UML2 es mostra en la figura 4.5.

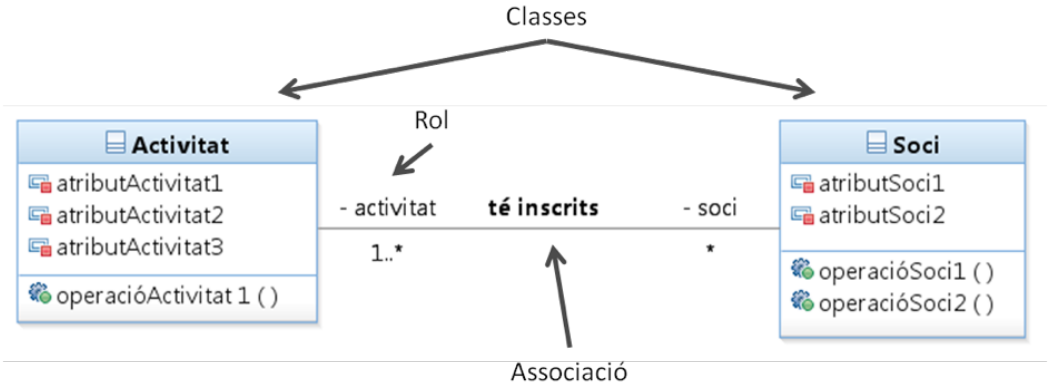


Figura 4.5. Constructors bàsics del diagrama de classes

El nom de la classe ha de ser significatiu i anar en lletres minúscules, excepte la primera lletra del nom que va en lletra majúscula. Si el nom de la classe és compost, cadascun dels noms comença en majúscula i van juntes sense espais ni guions.

Associació. Una associació és una relació entre dues o més classes, la qual té algun objectiu específic o és necessària per a la implementació del programari. Les associacions en UML2 s'anomenen amb un verb en tercera persona del singular.

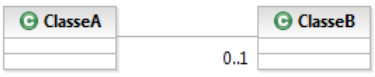
En la figura 4.5 es mostra un exemple d'associació entre les dues classes Soci i Activitat, aquesta associació representa que una activitat en particular pot tenir molts

(*) o cap soci inscrit i que un soci pot estar inscrit en una activitat com a mínim, i pot estar-ne en moltes activitats.

En cada associació s'ha de definir la **multiplicitat** de cada un dels extrems, que defineix quants objectes d'una classe es connecten amb quants objectes de l'altra. La multiplicitat indica si la connexió és obligatòria i si un objecte pot estar connectat amb més d'un de l'altra classe. Per dues classes A i B connectades, la multiplicitat de la classe A es detalla a l'extrem oposat on connecta amb la classe B. La multiplicitat s'expressa habitualment amb dos valors, el primer indica el mínim requerit i pot ser 1 o 0, i el segon que indica el màxim possible (habitualment 1 o més indicat amb *); en la taula 4.1 es mostra la representació de les multiplcitats més habituals. Però també es poden donar altres possibilitats. En resum, la multiplicitat ve donada per un rang que pot ser:

- Número exacte: 1, 2, 5, 10, etc.
- Rang de valors: 0..1, 1..1 (1), 1..10, 0..11, 1..25, etc.
- Número sense especificar: 1..*, * (0..*).

Taula 4.1. Representació de la multiplicitat amb UML.

Combinació de respostes	Representació multiplicitat
Un objecte de la classe A obligatòriament ha de tenir un objecte de la classe B connectat. Un objecte de la classe A pot estar connectat amb molts (més d'un) objectes de la classe B.	
Un objecte de la classe A pot no estar connectat amb cap objecte de la classe B (no és obligatori que un objecte de la classe A tinga connectat un en la B). Un objecte de la classe A pot estar connectat amb molts (més d'un) objectes de la classe B.	
Un objecte de la classe A obligatòriament ha de tenir connectat un objecte de la classe B. Un objecte de la classe A com a màxim només pot estar connectada amb un objecte de la classe B.	
Un objecte de la classe A pot no estar connectat a cap objecte de la classe B. Un objecte de la classe A com a màxim només pot estar connectada amb un objecte de la classe B.	

Atributs. Un atribut és qualsevol propietat d'interès per al sistema, comuna a tots els objectes de la classe, és a dir, una característica per a la qual cada objecte té valors (iguals o diferents). Poden haver classes sense atributs. S'anomenen en singular, la primera lletra en minúscula i, si es composta, l'inici de les següents paraules en majúscula (vegeu figura 4.6).

Per als atributs es pot indicar un tipus de dada, encara que açò se sol fer en la fase de disseny i no en l'anàlisi. Tenen una sèrie de propietats i es pot especificar el que s'anomena visibilitat, encara que no s'estudia en aquesta assignatura.

Operacions. Una operació permet implementar funcionalitat associada als objectes de la classe per donar resposta als requisits d'usuari representats en els casos d'ús.

Poden haver-hi classes sense operacions. S'anomenen en singular, la primera lletra en minúscula i si es composta l'inici de les següents paraules en majúscula (vegeu figura 4.6). Per a les operacions es pot indicar un tipus de dada per a cadascun dels paràmetres d'entrada i un tipus de dada per al retorn de l'operació, açò és el que s'anomena la **signatura de l'operació**. Encara que la signatura no se sol especificar en l'anàlisi, sinó en el disseny.

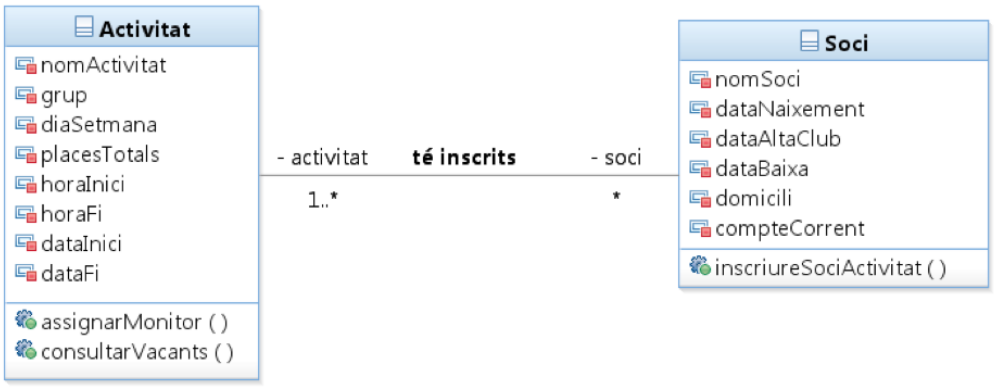


Figura 4.6. Exemple de la notació dels atributs i de les operacions en una classe

2.3. Passos de l'anàlisi amb UML

Durant l'anàlisi es realitza l'especificació del programari a desenvolupar tenint en compte els requisits i casos d'ús. No es detallen aspectes físics o d'implementació. Per realitzar l'anàlisi amb UML es poden utilitzar el digrama de classes, el diagrama d'activitats i el d'estats.

Per tal de mostrar l'estructura de la informació del domini s'utilitza un diagrama de classes. És convenient mostrar diferents vistes d'aquest diagrama, una primera en la qual es puguen observar amb claredat quines són les classes incloses i les seues relacions. En un altra vista es pot mostrar de forma detallada els principals atributs i operacions de cadascuna de les classes. Tant el tipus de dades dels atributs com la signatura de les operacions són dos elements que se solen ometre a nivell d'anàlisi, i es deixen per a posteriors refinaments durant el disseny. A més a més, cal remarcar que el diagrama de classes realitzat a nivell d'anàlisi s'ha de centrar només en la lògica de negoci, i que altres qüestions tècniques, de codificació, de presentació o interfície d'usuari, o de persistència s'han de deixar per a la fase de disseny.

Per tal de mostrar el comportament del sistema s'utilitzen:

- Diagrama d'activitats: per mostrar quin és el flux de treball de la lògica de negoci del sistema informàtic que s'està desenvolupant.
- Diagrama d'estats: només per representar la variació d'estats que poden tenir aquelles classes més rellevants o complexes de diagrama de classes. Per exemple, un diagrama d'estats amb tres o menys estats no és necessari perquè denota un comportament massa senzill.

Els passos que cal seguir per a desenvolupar aquest model són els següents:

1. A partir del diagrama de casos d'ús realitzat anteriorment, dibuixar un diagrama d'activitats per tal de mostrar la lògica de negoci. S'ha de centrar el diagrama en el cas d'ús de major valor afegit, intentant després completar tota la lògica de negoci associada.
2. Revisar el diagrama d'activitats desenvolupat per que siga equilibrat. És a dir, que totes les activitats són d'un nivell de complexitat semblant, no hi ha activitats molt grans sense descompondre i no hi ha accions de molt baix nivell amb altres més complexes. Cal recordar que és un model d'anàlisi, i per tant s'han d'evitar qüestions tècniques i de disseny.
3. Dibuixar un diagrama de classes per tal de representar tota la informació associada al diagrama d'activitats dibuixat i als casos d'ús. Proporcionar la vista del diagrama només amb les classes i les relacions.
4. Incloure els atributs i operacions de cadascuna de les classes del diagrama de classes per tal de proporcionar una vista més detallada.
5. Per a aquelles classes complexes pel que fa als diferents estats que poden assolir, realitzar un diagrama d'estats.

3. Verificar i validar el model de l'anàlisi

De la mateixa manera que s'ha fet en la fase anterior és necessari verificar que el model de l'anàlisi es fa de manera correcta, que els diferents diagrames són correctes i consistents entre ells i amb la definició de requisits, i també validar els models amb els usuaris que poden entendre que s'avança cap a la solució que ells esperen. La representació de la funcionalitat i de les dades del producte de programari permet validar abans d'implementar codi (tasca que comporta més temps i més cost) que el que entenen els desenvolupadors del que ha de ser el producte final és el mateix.

Per a verificar que es desenvolupen els diagrames d'UML correctes caldrà comprovar en primer lloc que la notació és la correcta i que els diagrames són complets. Es poden usar tècniques com matrius per comprovar les possibles inconsistències. En aquesta assignatura es demana que almenys es verifiquen les regles de les taules 4.2 i 4.3 per comprovar la consistència entre el diagrama de casos d'ús i el diagrama de classes:

Taula 4.2. Consistència bàsica entre el diagrama de casos d'ús i el diagrama de classes de l'anàlisi.

Diagrama de casos d'ús	Diagrama de classes
Cas d'ús (representen funcionalitat)	Operacions primitives que no es representen Operacions especificades en les classes
Actors	No hi ha cap correspondència en el diagrama de classes
Informació que s'emmagatzema o registra	Classes una o combinacions de classes associades
Consulta d'informació clau	Atributs bàsics o fonamentals en les classes

Taula 4.3. Consistència bàsica entre el diagrama de classes de l'anàlisi i el diagrama de casos d'ús.

Diagrama de classes	Diagrama de casos d'ús
Classes una o combinacions de classes associades Associacions	Ha d'haver-hi casos d'ús per registrar, consultar i modificar la informació de la classe i de l'associació
Operacions especificades en les classes	Hi ha casos d'ús que representen la funcionalitat
Atributs bàsics o fonamentals en les classes	Hi ha casos d'ús que necessiten eixa informació

A manera d'exemple en la taula 4.4 es mostra una matriu en la qual es verifiquen els casos d'us del diagrama de casos d'ús de l'exemple del Club Esportiu (tema 3 figura 3.3) i el diagrama de classes de la figura 4.7.

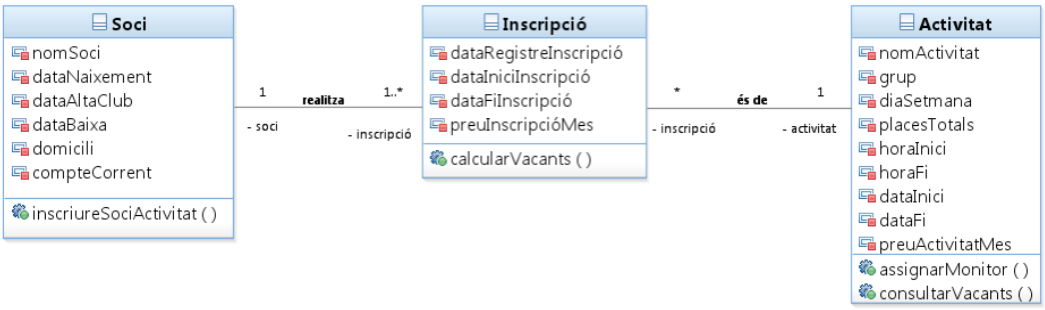


Figura 4.7. Part del diagrama de classes del cas del Club Esportiu

Taula 4.4. Verificació de consistència casos d'ús i diagrama de classes.

	Casos d'ús				
Classe	Registrar soci	Programar activitats	Gestió de monitors	Fer inscripcions	Facturar
Soci	Operacions i atributs complet				
Associació Soci Inscripció				Quan es crea una inscripció és necessari saber de quin soci és	

4. Annex: Tècniques estructurades

Existeixen altres tècniques, a més a més dels diagrames d'UML, per a documentar l'anàlisi. Per exemple, les que es denominen tècniques o mètodes estructurats que inclouen com a models més representatius el Model Conceptual (Entitat/Relació), estudiat en les assignatures de bases de dades, i el Diagrama de Flux de Dades (DFD). Com a annex al tema s'inclou la descripció del que és i com es desenvolupen aquests dos diagrames.

Les metodologies estructurades proposen tècniques que permeten documentar requisits i modelitzar el sistema tant des del punt de vista lògic a l'anàlisi, com des del punt de vista físic al disseny. Per definició les metodologies estructurades segueixen un model de desenvolupament en cascada (vegeu figura 4.8).

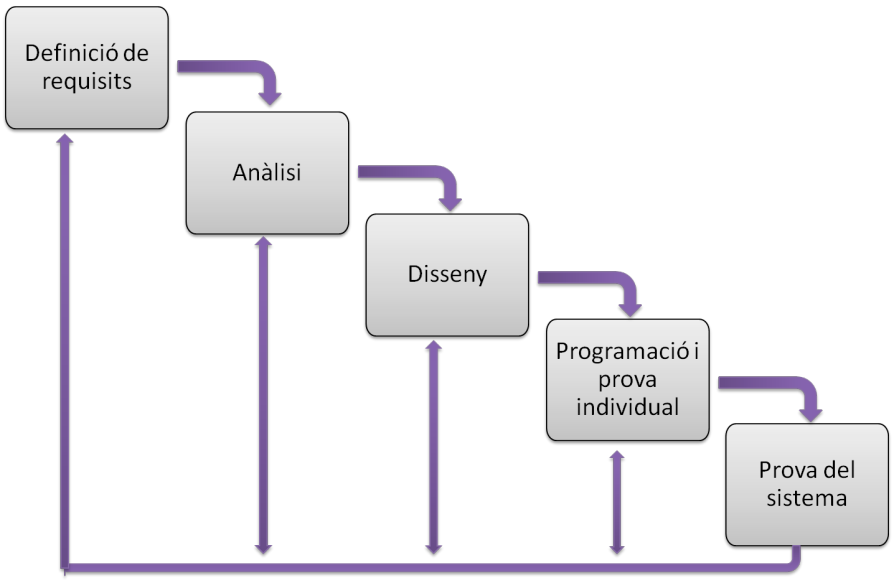


Figura 4.8. Model en cascada

Les principals tècniques que s'usen habitualment en l'anàlisi quan se segueix una metodologia estructurada són el Diagrama de Flux de Dades i el Model Conceptual de Dades o el Model Entitat-Relació.

4.1. El diagrama de flux de dades

El diagrama de flux de dades (DFD) és un llenguatge gràfic que descriu les transformacions de la informació i com aquesta es mou a través dels diferents mòduls del sistema. Inclou símbols per a representar les dades, la informació tant externa com interna al sistema, processos, magatzems de dades i el flux de la informació. El DFD és una tècnica que va ser proposada per Edwuard Yourdon (Yourdon, 1993) i Tomas de Marco, per això a vegades se'ls anomena diagrames de Yourdon, o Yourdon-De Marco.

Els diagrames de flux de dades mostren la visió lògica del flux de la informació i són una de les tècniques més comunament utilitzades sobretot en sistemes operacionals, encara que també són útils per a modelar sistemes en temps real. Poden utilitzar-se per a representar el programari a qualsevol nivell d'abstracció. Permeten realitzar de forma senzilla una abstracció del producte de programari independentment dels suports físics sobre els quals funcionarà. Aquest diagrama, encara que es considera obsolet, pot ser útil per representar la funcionalitat del sistema com l'alternativa als diagrames de casos d'ús i de classes.

EL DFD utilitza quatre elements bàsics:

- Els **processos** representen les transformacions de les dades i la informació. És a dir, mostren les accions que s'executaran amb el codi quan s'hi implemente. S'han d'anomenar amb una sola paraula, o una frase senzilla que indique l'acció (verb) i l'objecte sobre el qual es realitza l'acció. La representació gràfica pot variar segons l'eina CASE o la metodologia particular utilitzada, normalment és un rodolí o una bombolla, com a vegades s'anomenen.
- Els **fluxos de dades** que representen la informació i les dades en moviment. Hi ha una ampliació de la notació on es poden representar fluxos de control. S'han de designar amb un nom representatiu. La representació és en forma de fletxa.
- Els **magatzems** que representen la informació en repòs, és a dir, que ha d'estar guardada per poder ser reutilitzada pel programari per dur a terme la seua funcionalitat. Els magatzems es representen habitualment per dues ratlles paral·leles i s'anomenen amb un nom en plural. Un flux que entra en un magatzem el modifica. Un flux que ix d'un magatzem obté informació sense modificar-hi el seu contingut.
- Les **entitats externes** que representen els productors inicials i els receptors finals de la informació i les dades habitualment són persones (o agrupacions), altres sistemes informàtics o dispositius externs al sistema amb capacitat d'interactuar amb el programari. S'han d'anomenar amb un nom en singular i es representen amb un rectangle.

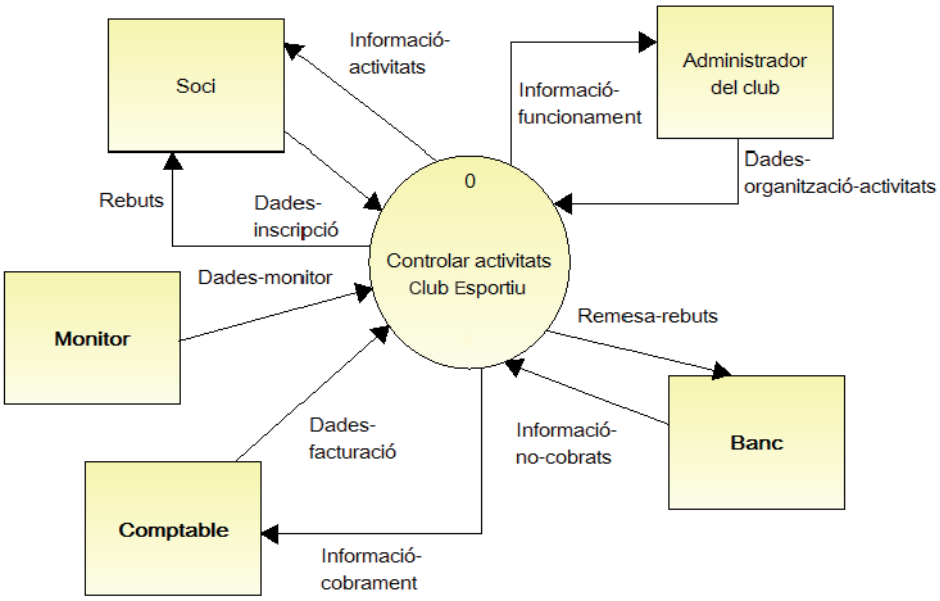


Figura 4.9. Diagrama de context del cas del Club Esportiu

Els diagrames de flux de dades s'organitzen per nivells per a comprendre el problema que cal resoldre anant de dalt (i menys detall) cap a baix (amb més detall). Com a exemple en la figura 4.9 es mostra el que seria un DFD de primer nivell, també anomenat diagrama de context amb sis entitats externes i cap magatzem. La figura 4.10 mostra un diagrama de segon nivell o explosió del procés del diagrama de context, on es manté la consistència amb les entitats externes i es representen els magatzems del sistema.

El desenvolupament del diagrama de flux de dades ha de complir una sèrie de regles d'integritat i consistència que permeten comprovar la correcció del seu ús. Algunes d'aquestes regles bàsiques són:

- Tots els processos han de tenir almenys un flux d'entrada i un flux d'eixida. No hi poden haver processos de generació espontània d'informació o processos que reben informació i no proporcionen res (forats negres).
- El flux d'informació ha de ser continu. No poden aparèixer fluxos d'informació en nivells intermedis que no havien aparegut abans o no estigueren inclosos en fluxos de nivell superiors. Tampoc poden desaparèixer sense proporcionar altres resultats.
- Qualsevol flux de dades que deixa un procés s'ha de basar en les dades d'entrada d'aquest. Només han d'entrar en cada procés els fluxos de dades necessàries.
- Tots els fluxos tenen almenys un dels seus extrems connectat amb un procés.
- Els magatzems han de tenir fluxos d'entrada i d'eixida.

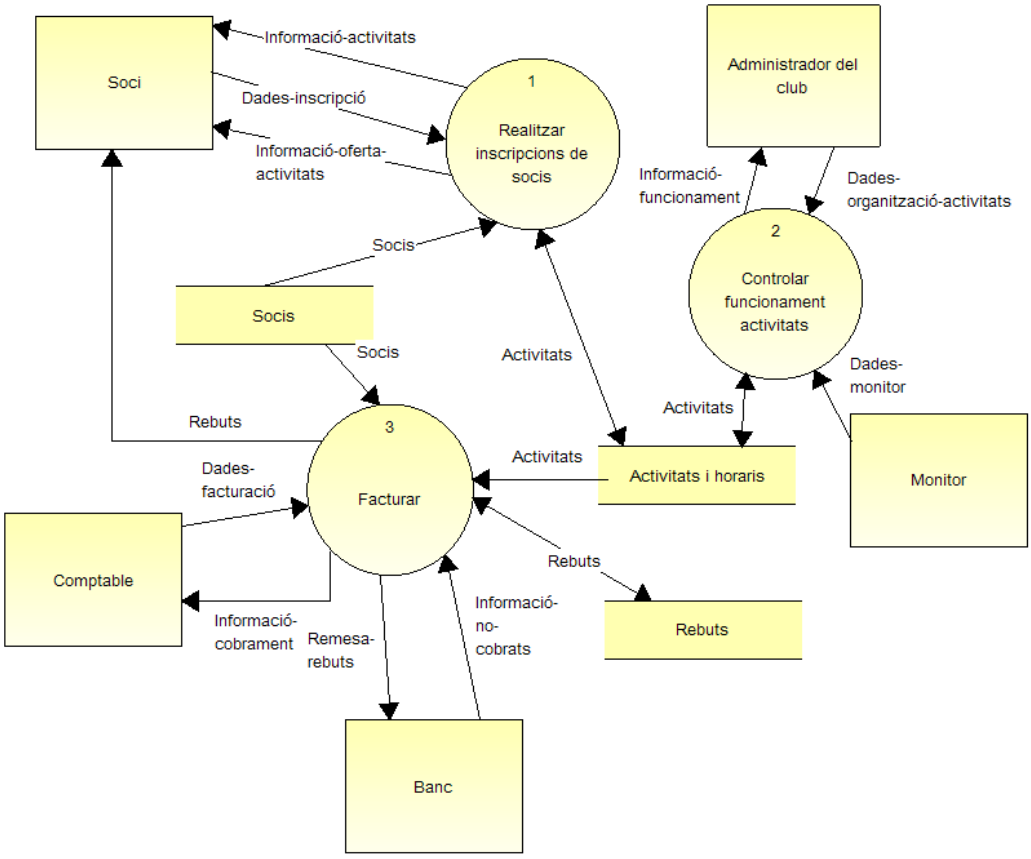


Figura 4.10. Exemple d'explosió del procés «Controlar Activitats Club Esportiu» amb DFD

De manera resumida els passos que cal seguir per a desenvolupar un DFD de dalt cap a baix són:

1. Crear el diagrama de context amb una única bombolla que represente tot el sistema com un únic procés, incloent-hi els fluxos de les entrades de dades i els fluxos d'eixida d'informació connectats amb les entitats externes.
2. Crear un segon digrama (primera explosió) en què apareguen els principals subprocessos i els magatzems representatius. Connectar les entrades i les eixides amb els subprocessos adequats mantenint la consistència i afegir els fluxos necessaris.
3. Repetir la subdivisió fins a tenir funcions d'un nivell adequat.

El model representat pels diagrames de flux de dades es completa amb documentació adequada que aporte la informació necessària per entendre els diferents diagrames. En particular, per als diagrames de flux de dades cal desenvolupar:

- El **diccionari de dades** que inclourà la descripció lògica de la informació que representen els magatzems i els fluxos que apareixen en el model gràfic. És un llistat organitzat de totes les dades del sistema, amb definicions precises i rigoroses.

- **La descripció o especificació de les funcions**, on es descriurà la funcionalitat representada per les bombolles representades en els diagrames. Corresponen a l'especificació del programari a codificar.
- Una descripció textual del que representa cada **entitat externa**.

4.2. El model E/R

El model conceptual de dades (MCD) o el Model Entitat/Relació (E/R) són tècniques que permeten representar la informació en repòs d'un sistema, des del punt de vista lògic (sense especificar suport o formats físics), i defineixen la seua estructura i les regles necessàries de funcionament. Aquests models proporcionaran la base per al correcte disseny de bases de dades relacionals. L'objectiu principal és identificar i representar la informació independentment de com s'hi accedirà i de com estarà informatitzada físicament.

Encara que són diferents en alguns conceptes bàsics, el resultat és molt semblant i constitueixen un pas previ fonamental per al correcte desenvolupament de bases de dades relacionals. En aquesta assignatura es descriu com desenvolupar el model E/R utilitzant la notació coneguda com *crow's foot*, que és un estàndard de facto per a les eines CASE. En la figura 4.11 es mostra el model E/R realitzat amb aquesta notació per al cas pràctic del Club Esportiu.

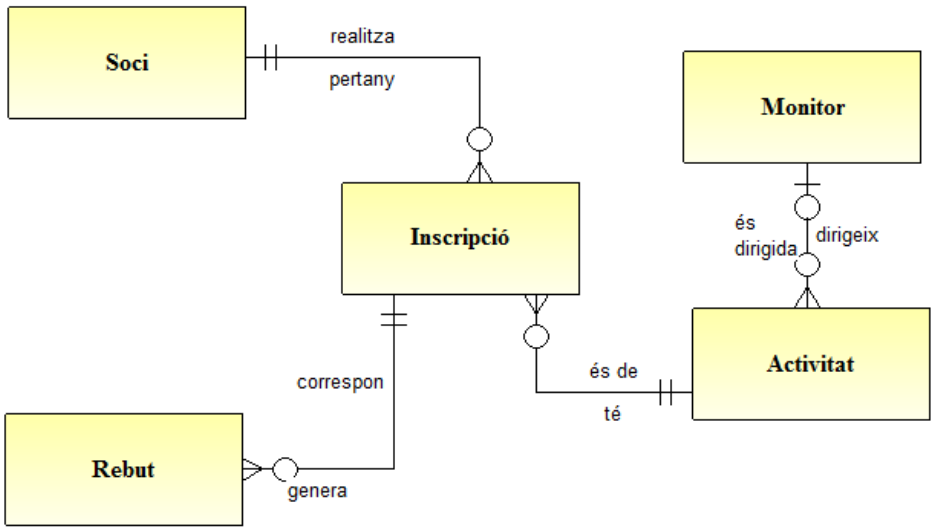


Figura 4.11. Model E/R inicial del cas del Club Esportiu

El model E/R es desenvolupa a partir dels requisits de dades i de les característiques d'aquestes dades definides pels requisits d'usuari. Per a obtenir un correcte resultat, s'han de plantejar qüestions com ara:

- Quina és la principal informació i quins són els objectes d'interès de cadascuna de les funcions/requisits/casos d'ús del sistema?

- Quins detalls caracteritzen aquests objectes i aquesta informació?
- Com estan relacionats aquests objectes i aquesta informació?

En un model E/R es representen entitats i relacions.

2.2.1. Entitats

Les entitats són els conceptes d'interès per al sistema, i sobre els quals el sistema ha de mantenir informació per a cobrir requisits dels usuaris. Una entitat representa un conjunt d'objectes, tangibles o abstractes, que existeixen amb unes característiques semblants i que poden distingir-se d'altres objectes en el sistema. Cadascun d'aquests elements individuals es denomina ocurrència.

Per exemple, en el cas pràctic del Club Esportiu una entitat seria «Soci» i una ocurrència d'aquesta entitat «El soci número 2 Manolo Pérez» encara que puga haver-se donat de baixa. Per a nomenar les entitats s'assignen noms en singular i es dibuixen amb un rectangle com es mostra en la figura 4.12.

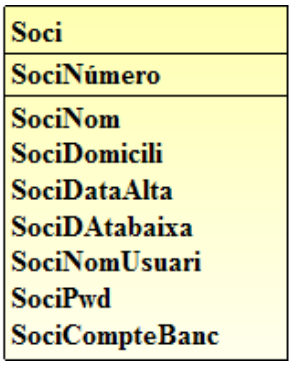


Figura 4.12. Exemple d'entitat Soci, clau primària SociNúmero i atributs

Per a cada entitat es defineixen diferent **atributs**, que són les característiques d'interès per al programari o sistema. En el model E/R per a cada entitat es defineix una **clau primària,** que és un atribut (o més d'un concatenats) que permet identificar cada ocurrència de manera única i inequívoca. Per tant, la clau primària pren un valor únic per a cada ocurrència. Per a l'entitat soci la clau primària seria «SociNúmero», que seria una dada a definir en la base de dades de manera que no poguera tenir el mateix valor per a dos socis diferents.

2.2.2. Relacions

Les relacions representen les connexions que són d'interès per al sistema entre les ocurrències de dues o més entitats. En general, es representen mitjançant una línia que uneix les entitats connectades. Però pot haver-hi diferents representacions gràfiques per a indicar característiques i valors de les relacions.

Les relacions han de tenir un nom que indique per què unes ocurrències d'una entitat estan connectades amb ocurrències d'una altra. A més a més, per a cada relació cal identificar el que s'anomena cardinalitat.

La cardinalitat reflexa si la connexió és obligatòria o no per a totes les ocurrències de cada entitat involucrada en la relació i el nombre d'ocurrències màxim i mínim que poden estar connectades de cada entitat.

Per exemple, en la figura 4.13 es mostra la relació entre l'entitat Soci i l'entitat Inscripció. Aquesta relació representa la connexió necessària entre un soci i totes les inscripcions que ha realitzat. En aquest cas es considera per una part, que pot haver-hi socis que no han fet cap inscripció i que un soci pot tenir moltes inscripcions; i per una altra part es considera que una inscripció obligatòriament ha de tenir un soci al qual pertany i que només pot pertànyer a un soci.

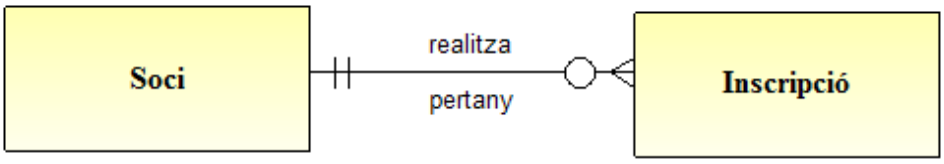
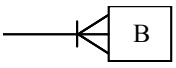
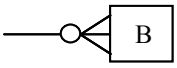
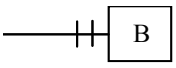
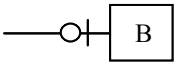


Figura 4.13. Exemple de relació i cardinalitat

En la taula 4.5 es mostra un resum de la representació de la cardinalitat amb aquesta notació.

Taula 4.5. Representació gràfica de la multiplicitat en el model de dades.

Combinació de respostes	Representació de la cardinalitat
Una ocurrència de l'entitat A obligatòriament ha d'estar connectada amb una ocurrència de l'entitat B però també pot estar connectada amb moltes.	
Una ocurrència de l'entitat A pot no estar connectada amb cap ocurrència de l'entitat B, o també pot estar connectada amb moltes, una o més d'una.	
Una ocurrència de l'entitat A obligatòriament ha d'estar connectada amb una ocurrència de l'entitat B i com a màxim només pot estar connectada amb una.	
Una ocurrència de l'entitat A pot no estar connectada amb cap ocurrència de l'entitat B i com a màxim només pot estar connectada amb una.	

5. Resum

En aquest tema es fa una breu revisió del que són les tècniques estructurades i les dues tècniques més usades en aquest paradigma, encara que l'objectiu principal és estudiar els diagrames UML que es poden fer servir per realitzar l'anàlisi d'un producte de programari.

Es detallen els elements bàsics de dos dels diagrames d'UML2 que es poden utilitzar en la fase de l'anàlisi durant el procés de desenvolupament d'un sistema informàtic. En primer lloc, s'analitza el diagrama d'activitats com un dels diagrames de comportament d'UML2 que durant l'anàlisi es pot utilitzar per tal de mostrar la lògica de negoci del sistema informàtic que es vol desenvolupar. En segon lloc, s'expliquen les principals característiques del diagrama de classes i la seua utilitat per representar tant l'estructura de la informació com el comportament del sistema al llarg de les diverses iteracions del procés de desenvolupament del sistema informàtic.

Activitats complementàries

1. Penseu en alguna recepta de cuina que conegueu. Dibuixeu un diagrama d'activitats per tal de representar les diferents activitats que es duen a terme a l'hora de realitzar aquesta recepta.
2. Tenint en compte algun dels programes que heu codificat en assignatures d'altres cursos, dibuixeu un diagrama d'activitats que represente la seua lògica d'execució.
3. Reviseu el diagrama de flux de dades de la figura 4.9 i proposeu un diagrama de casos d'ús i un diagrama de classes amb operacions i atributs.
4. Busqueu algun dels models E/R que heu fet en assignatures de base de dades i feu el diagrama de classes corresponent usant la notació adequada.
5. Llegiu el tema i feu una proposta de quatre preguntes de test.

CAPÍTOL 5

Disseny

What is design? It´s where you stand with a foot in two worlds- the world of technology and the world of people and human purposes- and you try to bring the two together.

M. Kapor, 1990 (Lotus 123 creator)

OBJECTIUS

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Comprendre i assimilar quines són les principals activitats que s'han de dur a terme en el disseny d'un producte de programari.
- Saber alguns aspectes que cal tenir en compte per fer un disseny correcte amb l'objectiu d'assolir les bases de la construcció d'un programari d'alta qualitat.
- Conèixer els diagrames d'UML que es desenvolupen per al disseny del programari.
- Aprendre algunes de les principals consideracions per a dissenyar unes interfícies d'usuari adequades, en particular pantalles i informes.

1. Introducció

El **disseny de programari** comporta l'especificació física del programari que s'ha documentat en l'anàlisi de manera lògica. Al disseny del programari es detalla com i amb quin suport s'han d'introduir les dades, els formats i suports físics en els quals es mostrarà la informació d'eixida, com s'emmagatzema físicament la informació (model físic de la base de dades), és a dir, es decideix l'arquitectura del producte de programari. Habitualment es parla del disseny del sistema, i aleshores a més a més s'inclou també l'arquitectura tecnològica o de maquinari, incloent-hi la definició dels dispositius de comunicació del programari amb els usuaris, i altres sistemes.

El disseny és una activitat crítica per a aconseguir un producte de programari de qualitat, és a dir eficient, fiable, mantenible i orientat a l'usuari**.** En aquesta fase es decideix, per exemple, com s'introduiran les dades, aquest aspecte pot ser la diferència entre tenir un producte fiable i eficient o no. Per exemple, si es dissenya un procés d'entrada de dades, caldrà decidir si se'n fa ús, o no, d'un lector òptic de codi de barres o d'un lector de RFID (*Radio Frequency Identification*) o simplement s'han de teclejar les dades. Per a prendre la decisió que aporte una solució més eficient caldrà valorar, entre altres aspectes, el cost i els beneficis que aporta l'ús dels dispositius, quantes vegades es fa aquesta introducció de dades, temps que s'estalvia en cada cas, com de crític és el procediment, on s'ubica i quin és el perfil de l'usuari que farà ús de la funcionalitat. En definitiva, durant el disseny cal identificar i avaluar alternatives i seleccionar aquella que millor assegure implementar un producte de qualitat tenint en compte tots els components que interactuen en el desenvolupament d'un producte de programari.

Per a aconseguir un disseny que assolisca aquests objectius s'han d'aplicar tècniques adequades (algunes de les quals s'estudien en aquest tema) i s'han de tenir en compte, entre altres, els següents criteris:

- El disseny d'un producte de programari ha de cobrir tots els seus components: interfícies, arquitectura i components del programari, dades i, disseny de la plataforma de maquinari si es considera part del producte.
- Prioritzar en el disseny solucions que conduisquen a desenvolupar un producte clar i senzill d'usar.
- Sobre la introducció de dades i la informació d'eixida:
  - S'han d'utilitzar mètodes automàtics d'entrada de dades sempre que siga possible. La qualitat i fiabilitat de la informació que es produeix és proporcional a la correcció i fiabilitat de les dades introduïdes.
  - Les dades s'han d'introduir i verificar on i quan es produeixen, sempre que siga possible, i una sola vegada. Els retards innecessaris poden produir errors i pèrdues d'informació.
  - La informació d'eixida ha d'estar ben presentada, ha de ser fàcil de comprendre i tenir un apropiat nivell de detall.

- Sobre el programari i el disseny de la base de dades:
  - S'han de detectar futures necessitats dels usuaris finals i dissenyar el producte perquè puga adaptar-se fàcilment per estalviar modificacions posteriors costoses del programari.
  - El disseny ha de ser modular, usar patrons i estils identificables. És a dir, aplicar principis de modularitat (alta cohesió) i independència (baix acoblament).

Moltes d'aquestes consideracions, a vegades, entren en **conflicte** unes amb les altres. És llavors quan l'enginyer informàtic ha de valorar les alternatives i quina pot ser la millor solució. Per exemple, desenvolupar un sistema senzill, des del punt de vista dels usuaris, fa que normalment la programació siga més complicada i difícil de mantenir; la qualitat i el cost també són punts que es contraposen.

2. Activitats del disseny

Abans de començar la fase de disseny cal tenir un coneixement complet i comprendre els requisits del sistema i la seua anàlisi. Per tant, la primera activitat del disseny és realitzar una revisió de l'anàlisi del sistema que s'ha desenvolupat abans. A vegades, els enginyers que participen en el disseny i els que han participat en la definició de requisits i l'anàlisi són persones diferents, la qual cosa fa necessari un estudi en profunditat per part del nou equip de treball.

La taula 5.1 mostra un resum de les principals activitats que es poden dur a terme en la fase de disseny d'un sistema informàtic que inclouria l'apartat de disseny de la plataforma de maquinari. En aquest tema es descriuen alguns dels diagrames d'UML que es poden gastar per documentar el disseny del programari i de la base de dades, i s'inclou un apartat bàsic sobre el disseny de les interfícies d'usuari.

Taula 5.1. Activitats de la fase de disseny.

TASCA	DESCRIPCIÓ-OBJECTIU	Tècnica
Disseny dels fitxers del sistema i **la BD**	Dissenyar com s'emmagatzema físicament la informació. Seleccionar el tipus de base de dades, si és necessària i la distribució.	Diagrama de classes nivell de disseny (UML) Disseny lògic
Disseny dels processos del sistema	**Dissenyar l'arquitectura del programari** i els procediments d'usuari.	Diagrama de seqüència Diagrama de components
Disseny d'interfícies: les entrades i eixides del sistema: • **Interfícies d'usuari** • Interfícies amb altres sistemes i dispositius	Determinar com s'han d'introduir les dades, dissenyar els suports per a la captura de dades, els registres d'entrada i les mesures de seguretat i validacions per a les dades. Dissenyar com s'han de proporcionar la informació o altres eixides a dispositius externs.	Disseny interfícies Prototipus
Disseny i/o selecció del maquinari del sistema	Avaluar les necessitats de la plataforma sobre la qual s'executarà el programari. Aquesta activitat forma part de disseny però no de l'enginyeria del programari.	
Especificar i presentar el disseny	Documentar el disseny utilitzant tècniques adequades com models i representacions gràfiques.	

No es detalla com realitzar el disseny d'interfícies amb altres sistemes o dispositius, ja que aquest és en realitat disseny de programari, atés que els sistemes informàtics es comuniquen amb programes o tecnologies específiques dels dispositius del que es tracte.

3. Disseny amb UML

El disseny d'interfícies d'usuari és un aspecte comú a qualsevol metodologia de desenvolupament de sistemes o de productes de programari. Però pel que fa al disseny d'altres components del producte de programari, així com s'ha vist en el tema de l'anàlisi existeixen diferents mètodes i models gràfics segons les metodologies i paradigmes de desenvolupament emprades.

En les metodologies basades en paradigmes estructurats per a desenvolupar el disseny de la base de dades es segueixen el mètodes usats per a bases de dades relacionals. En aquest cas es part del model conceptual i es refina completant-lo, afegint atributs i normalitzant fins a obtenir el model lògic de la base de dades que ja s'estudia en altres assignatures del grau. Per a dissenyar els mòduls o parts del programari en el paradigma estructurat es poden utilitzar tècniques com

el diagrama d'estructures entre altres, que no inclourem en aquesta assignatura que se centra en l'ús d'UML.

A continuació es descriuen els diagrames d'UML que s'usen habitualment per al disseny i que són: el digrama de classes (afegint detalls i refinant relacions), els diagrames d'interacció i el diagrama de desplegament.

3.1. Diagrama de classes de disseny

El diagrama de classes durant el disseny evoluciona incorporant detalls específics del disseny que no s'havien especificat en l'anàlisi.

Entre d'altres han d'afegir-se tipus de dades als atributs, valors predeterminats i la signatura de les operacions. La signatura de les operacions identifica els tipus de paràmetres d'entrada i el tipus de retorn que proporciona l'operació. Aquests aspectes es poden veure en les classes quan en les eines CASE s'activa l'opció de fer visible la signatura, tal com es veu en la figura 5.1.

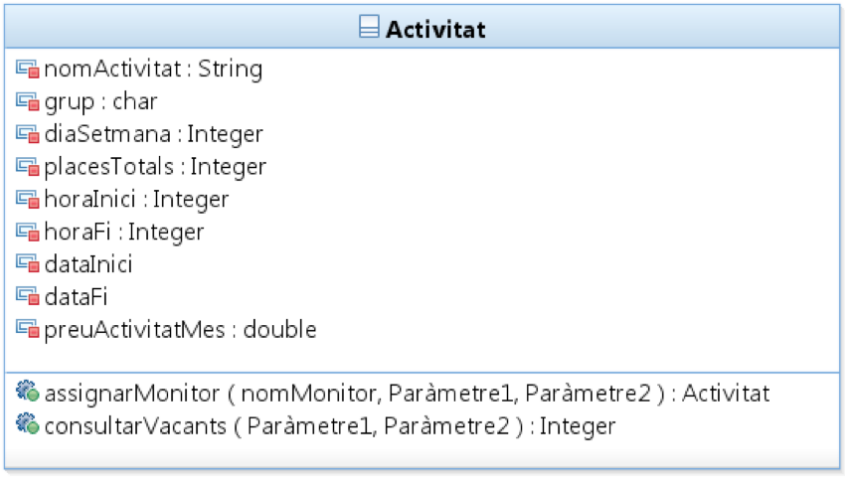


Figura 5.1. Exemple de classe amb signatura visible

S'afegeixen classes pròpies del disseny com poden ser (vegeu figura 5.2):

- Les que substitueixen atributs codificats. Per exemple, si un atribut d'una o més classes és província, es pot crear una classe província amb els atributs codi i nom. De manera que aquesta última classe es relaciona amb la resta de classes que tenien aquest atribut.
- Classes per a proporcionar detalls atributs de les relacions.
- Classes abstractes estereotipades com interfícies.
- Altres classes d'UML més avançat.

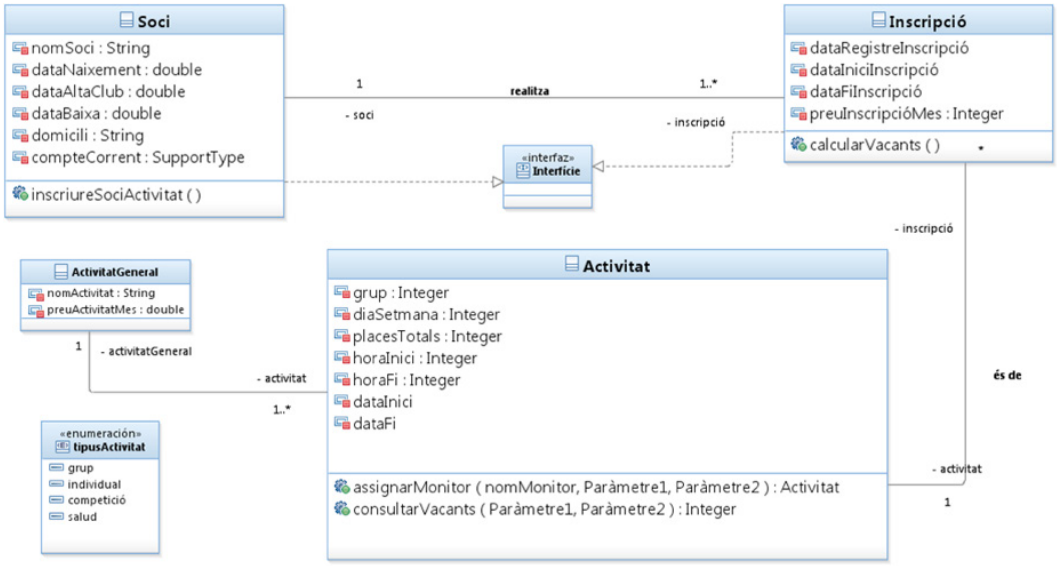


Figura 5.2. Exemple de diagrama amb classes de disseny

3.2. Diagrames d'interacció

Un diagrama d'interacció mostra un conjunt d'objectes (instàncies d'una classe) relacionats entre si per missatges que es poden enviar entre els objectes.

Els diagrames d'interacció s'usen per a modelitzar aspectes dinàmics d'un sistema. Hi ha dos tipus de diagrames d'interacció els diagrames de seqüència i els de comunicació. En el primer es destaca l'ordenació en el temps de com es produeixen els missatges, i en el segon, es destaca l'organització dels objectes que reben i envien missatges. En aquest tema estudiarem els diagrames de seqüència. Ambdós diagrames mostren semànticament (per significat) la mateixa informació, i es poden obtenir automàticament un de l'altre, encara que hi ha detalls d'informació diferents que s'haurien de completar en cadascun.

En els diagrames d'interacció es representen rols o **objectes**, **comunicacions** o enllaços i **missatges**.

En un diagrama de seqüència s'identifica la línia de vida d'un objecte que marca el temps que l'objecte existeix durant d'interacció. Els objectes poden persistir durant tota la interacció o bé crear-se i destruir-se. Aquests fets s'han de representar convenientment com es mostra en la figura 5.3.

Per a desenvolupar un diagrama d'interacció és necessari identificar en quin context es considera, que pot ser un sistema, una classe o bé un escenari d'un cas d'ús (una situació identificable amb un principi i un final). Per a crear el diagrama de seqüència se segueixen els següents passos:

- Identificar l'escenari i els objectes que hi participen col·locant-los en el diagrama d'esquerra a dreta en ordre cronològic a com s'activen. Habitualment s'inicia d'interacció corresponent a un escenari d'un cas d'ús amb una instància d'un actor que crida una operació sobre l'objecte.
- S'ha d'establir la línia de vida de cada objecte. Es representa per una línia vertical.
- Els missatges, que corresponen a instàncies de les operacions, es van collocant en les línies de vida. Es poden crear missatges (operacions) nous si no existeix el que siga necessari, el que proporciona més detall al diagrama de classes.
- Es poden establir detalls com intervals temporals i restriccions precondicions i postcondicions.

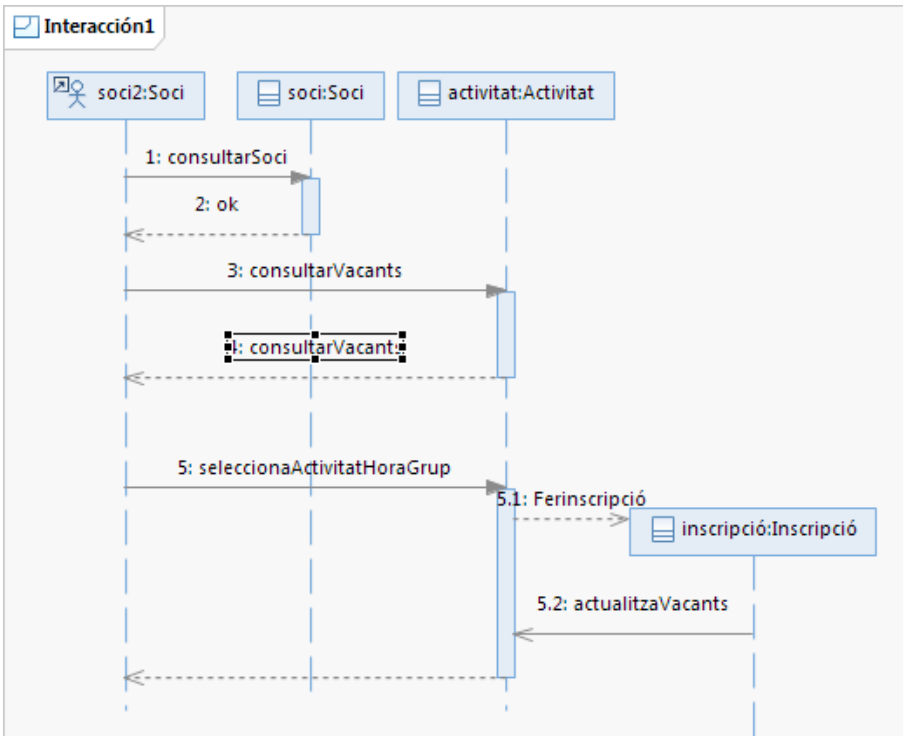


Figura 5.3. Exemple de diagrama de seqüència amb missatge de creació d'objecte

Un diagrama de seqüència únicament mostra un flux de control, és a dir, una execució en una situació determinada. Normalment per a una interacció complexa es desenvolupen diversos diagrames d'interacció, un en seria el principal i els altres els que mostrarien l'execució en condicions excepcionals.

3.3. Diagrama de desplegament

Els diagrames de desplegament se centren en l'estructura del programari i la seua distribució física. En la figura 5.4 es mostra un diagrama de desplegament on el programari s'executa en tres plataformes diferents.

Els diagrames de desplegament inclouen nodes que representen els elements computacionals. Es poden estereotipar o usar icones representatives per als processadors o dispositius físics que puguen aclarir la comprensió del model.

Per a especificar què s'ubica en cada node s'utilitzen els artefactes. Un artefacte representa les parts del programari que s'utilitzen per a acoblar i posar en producció el sistema. Poden ser llibreries reutilitzables o un codi fet a mida. S'ubica cada artefacte significatiu del sistema en un node.

Figura 5.4. Exemple de diagrama de desplegament bàsic

4. Disseny d'interfícies d'usuari

La **interfície d'usuari** és el mecanisme a través del qual s'estableix un diàleg entre el sistema informàtic i les persones. Per això, s'han de tenir en compte els **factors humans** per tal d'establir una comunicació fluida, obtenir una interfície d'usuari apropiada i desenvolupar el que es considera un entorn amigable. El tipus i grandària dels caràcters, la forma, el color i el moviment, per exemple, són factors fonamentals en aquests casos. A més de les característiques generals de l'habilitat humana, cal tenir en compte les capacitats particulars dels usuaris als quals s'adreça el sistema.[1] El nivell d'habilitat i de coneixements de l'usuari final influeix notablement en els resultats que el sistema proporciona. Cal conèixer el perfil de l'usuari típic, en cada cas, per poder adaptar la interfície a la seua capacitat.

El disseny de les interfícies d'usuari (en particular pantalles) és una de les activitats del disseny, els resultats de la qual, si són pobres, fan que el sistema final no siga del tot satisfactori per als usuaris. Menús difícils d'interpretar, textos d'error incorrectes o ambigus, documentació incorrecta o incompleta, poden donar una imatge de producte pobre encara que la programació siga excel·lent.

Aquesta activitat està directament relacionada amb **els avanços de la tecnologia** informàtica. Actualment es parla de *Grafical User Interface* (GUI) per fer referència a tot allò que proporciona una forma d'interactuar entre les persones i els productes de programari de forma gràfica. La tecnologia i els nous entorns multimèdia proporcionen mètodes de comunicació cada vegada més senzills i més adaptables als usuaris i, al mateix temps, més sofisticats pel que fa a la

1. L'accessibilitat del sistema pot ser un factor crític sobretot en entorns adreçats a usuaris finals no experts o amb capacitats reduïdes.

tecnologia que utilitzen. S'han de conèixer tots aquests aspectes tecnològics per tal de seleccionar els més adequats al perfil dels usuaris i a les tasques que s'han de desenvolupar.

Les **interfícies d'usuari** més habituals, que poden ser **tant d'entrada com d'eixida**, són les pantalles, tàctils o no, teclat i dispositius de selecció i lectura de dades. En el cas **d'interfícies d'eixida**, també s'utilitza per a obtenir la informació d'un sistema els informes impresos, i altres mitjans, com per exemple, la veu, la connexió d'interruptors o senyals digitals, etc.

4.1. Disseny de pantalles

Un adequat disseny de pantalles pot ser la diferència entre un producte de programari correcte i un que no ho és. Un correcte disseny de pantalles combinat amb teclats, dispositius de selecció i senyalització o dispositius de lectura és fonamental per al bon ús del sistema.

En aquest tema s'estudia el disseny de pantalles des d'un punt de vista genèric. Existeixen aspectes i consideracions que varien si es dissenya una pantalla per a un entorn web o dispositius tàctils o mòbils que no s'inclouen en els continguts del tema. Es pot trobar més detall sobre el disseny d'aplicacions web en el capítol 13 de Pressman (2010) i en Shneiderman (2006).

Alguns dels aspectes que s'han de tenir en compte per a dissenyar pantalles correctament són:

- Les **característiques humanes** generals, i les particulars de cadascun dels usuaris del producte de programari. Les pantalles han de ser agradables d'ús, que no cansen o resulten molestes a la vista, que eviten la introducció d'errors, etc. L'objectiu és proporcionar formats de pantalles que siguen simples, senzills de comprendre i clars pel que fa al seu ús. Per exemple, si està previst que l'intefície siga accessible per a persones amb discapacitats diverses caldrà preparar introduccions de dades alternatives, com per exemple alguns caixers bancaris que estan preparats per a ser usats per persones amb deficiències visuals.
- El **tipus d'entorn** i dispositius. Actualment l'ús d'entorns web per a interactuar des de qualsevol lloc i terminal i per a qualsevol tipus d'usuari fa plantejar-se nous models de pantalles i modes de comunicació amb els usuaris.

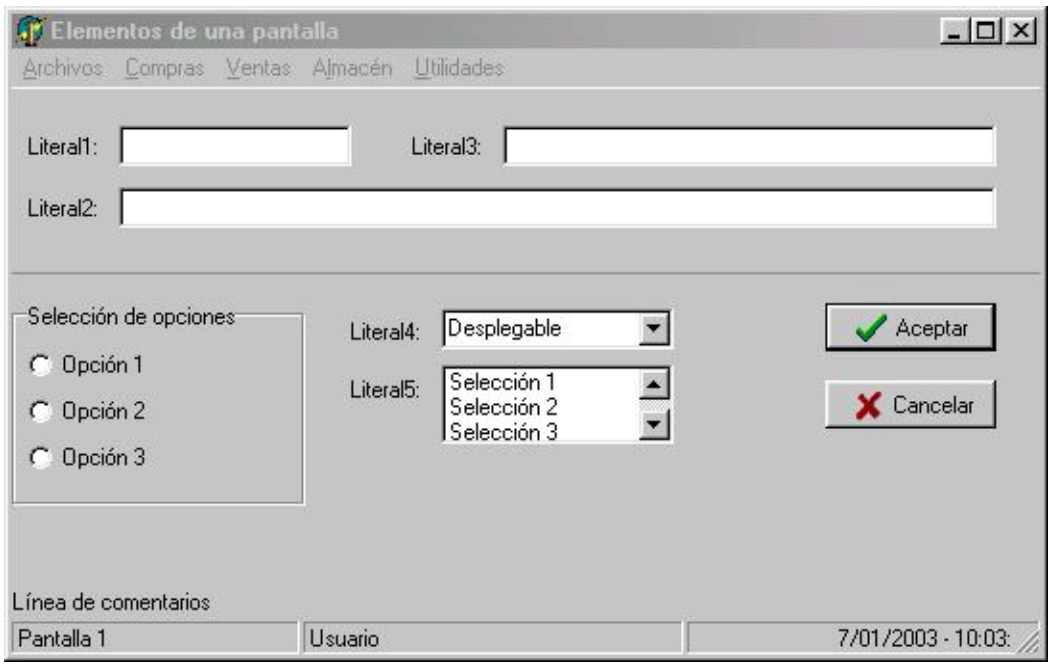


Figura 5.5. Exemple de finestra i elements habituals

- Els **elements** que han de mostrar-se. Habitualment les interfícies es desenvolupen tenint en compte la disposició que proporcionen les ferramentes de programació o determinats sistemes operatius. En qualsevol cas, és fonamental definir uns estàndards per als objectes comuns a totes les pantalles del mateix producte de programari, distribució, colors, fonts, missatges. Els elements més habituals són (vegeu figures 5.5 i 5.6):
  - Títol, normalment a la part de dalt perquè l'usuari puga ubicar-se.
  - Dades per a identificar l'usuari, la data, la sessió, etc., segons de quin programari es tracte.
  - Missatges, emergents o en llocs on no minven la visibilitat.
  - Barra de menús: normalment a la part superior de la pantalla davall del títol.
  - Botons, tecles de funció i comandaments: a la part inferior, o dreta però sempre de forma homogènia en totes les pantalles i finestres.
  - Cos de la pantalla o de la finestra: inclou dades capturades de forma automàtica per la pantalla, i la resta de dades que es presenten d'una forma lògica en sentit descendent. Les dades que l'usuari ha d'introduir se sol·liciten amb l'ús d'etiquetes o literals, camps de selecció, capçaleres de seccions, instruccions, etc. Cal establir la forma dels literals, grandàries, fonts i ubicació. Per exemple, els literals es poden situar sobre els espais que han de completar els usuaris o davant. Un altre aspecte important a l'hora de dissenyar el cos de la pantalla és l'alineació dels literals, per exemple, ajustats a l'esquerra, o ajustats a la grandària del seu text.

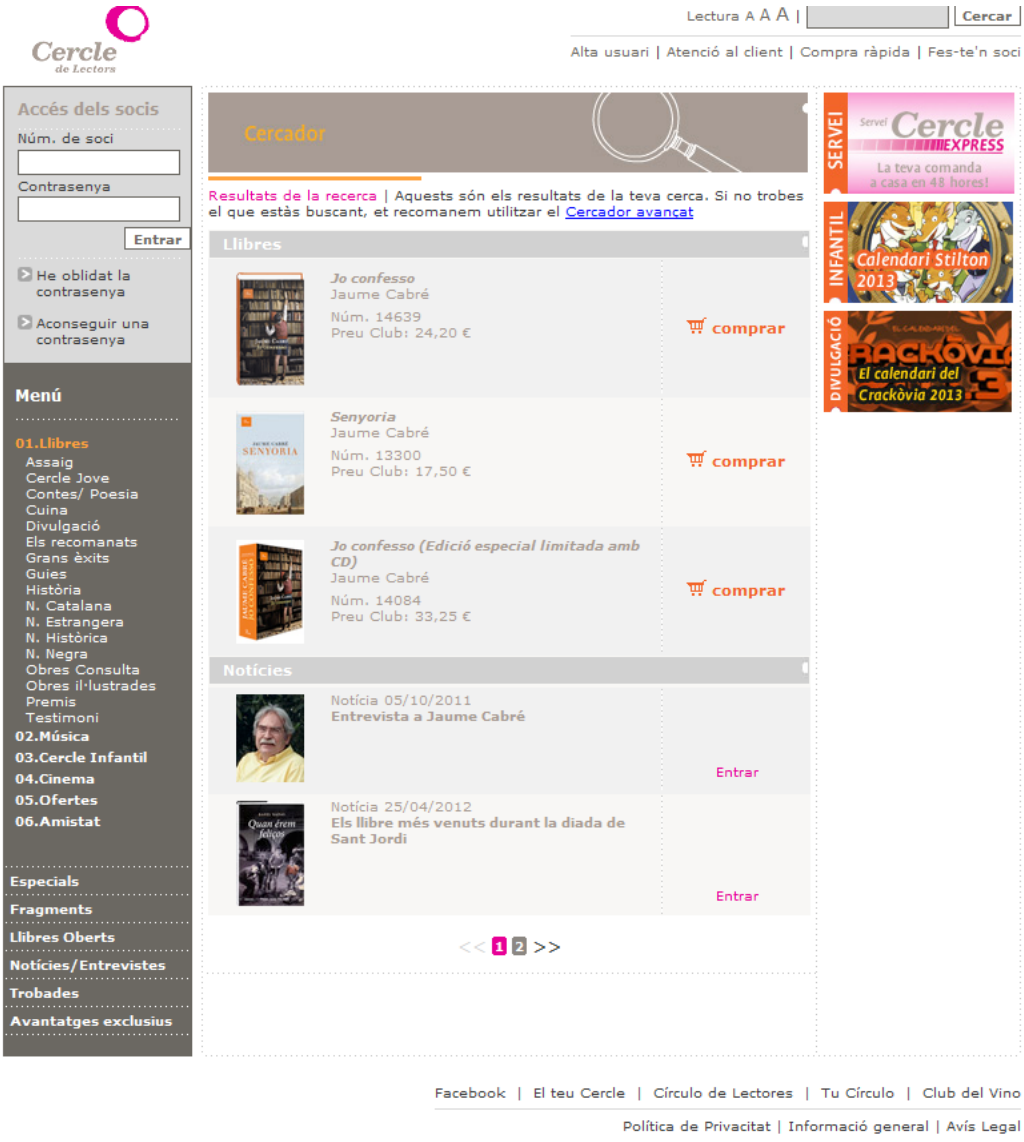


Figura 5.6. Imatge del portal de compra de Cercle.es

- **Ubicació** de la informació. L'homogeneïtat i la senzillesa en la representació de les dades, entre altres, són aspectes fonamentals a l'hora de dissenyar pantalles. Però, a més, cal considerar quina és la informació que es vol representar, i agrupar-la pel significat, ús o altres consideracions intrínseques que té. A continuació, es detallen algunes de les consideracions generals que es poden tenir en compte per a distribuir les dades i la informació del cos de la pantalla:
  - La **informació més rellevant** o necessària per dur a terme l'acció s'ubica a la part superior dreta de la pantalla.
  - Cal **reservar espais específics** per a determinats tipus d'informació: missatges d'error, comandaments, títols, dates, etc., i mantenir aquestes àrees de forma homogènia en totes les pantalles.
  - Una **distribució agradable i amb una certa estètica** és atractiva a la vista humana i subliminalment atrau l'atenció de l'usuari. Durant anys

s'han realitzat estudis sobre el que es pot considerar agradable o estèticament correcte en el disseny de pantalles. Com a resultat es poden considerar algunes característiques bàsiques que proporcionen aquest aspecte: equilibri, distribució de les dades de forma que proporcione estabilitat visual en la pantalla, regularitat o uniformitat en la distribució dels elements, simetria respecte a la pantalla, predicció, planificació de les dades de forma convencional i consistent, seqüencialitat, ordenació sistemàtica i lògica de les dades, l'agrupació d'elements a la pantalla es realitza tenint en compte l'organització visual i els significats de les dades. Les utilitats visuals de la pantalla poden ajudar a definir grups i donar-los visibilitat més atractiva (marcs, quadres, contrastos, pestanyes, etc.).

– S'ha de proporcionar **únicament informació** que siga **essencial** per realitzar les accions o prendre decisions i s'ha d'intentar proporcionar totes les dades relacionades amb una única tasca, en una única pantalla.
– S'han de mantenir uns determinats **nivells de densitat** de les dades en una pantalla. Una pantalla no ha d'estar saturada d'informació; un 25 % o 30 % és el percentatge aconsellable.
– És aconsellable crear les pantalles tenint en compte la **imatge** de l'empresa.

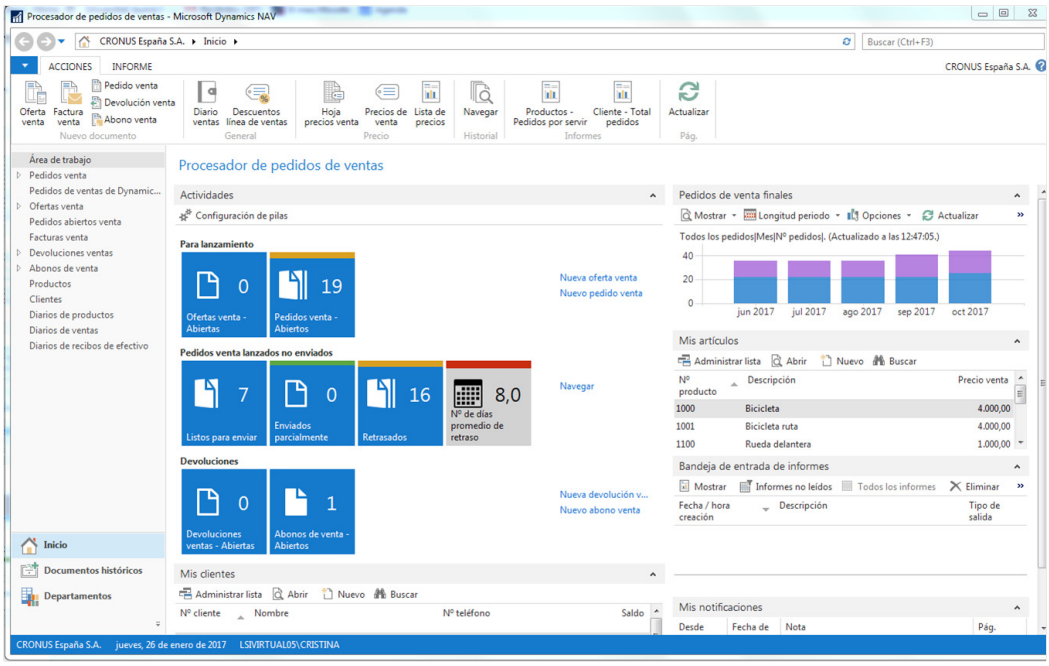


Figura 5.7. Exemple de pantalla de l'ERP Microsoft Dynamics NAV Classic

Passos per a dissenyar les pantalles, partint dels casos d'ús on hi ha actors humans, per cada cas d'ús:

- Identificar els tipus d'usuari partint els actors que són rols de persones: expert o inexpert i ús habitual o ús eventual, per exemple.
- Identificar els tipus d'interacció: dades d'entrada, diàleg amb recerca d'informació, seleccions que tinguen en compte els passos definits en el cas d'ús i les operacions de les classes involucrades.

- Identificar el tipus d'entorn i els dispositius: intranet, gestió interna (*backoffice*), web oberta, teclats, pantalles tàctils, etc.
- Crear una pantalla o plantilla estàndard per cada tipus d'entorn i usuari. Fer una llista de pantalles per cada tipus d'usuari i entorn i per a cada interacció.
- Organitzar-les en menús. Crear la jerarquia de pantalles (pot fer-se per usuari si l'abast és ampli).
- Per a cada pantalla s'ha de considerar, les dades que es demanen als usuaris i la informació que s'hi mostra. Quines dades poden ser desplegables perquè es poden seleccionar (com per exemple atributs codificats).

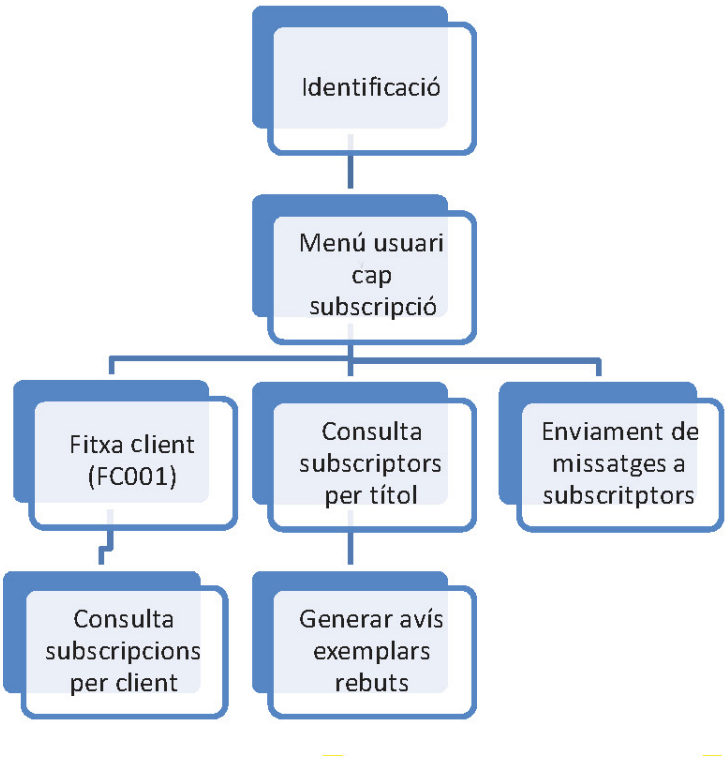


Figura 5.8. Exemple de jerarquia de pantalles per a les subscripcions

Per a documentar el disseny de pantalles:

- Un **document** dels **estàndards** que s'utilitzaran, comuns a totes les pantalles o per tipus d'entorn. Cal diferenciar com formats per a les pantalles d'introducció o de consulta de dades (pantalles principals) i com seran altres pantalles com les finestres emergents que mostren missatges d'error o ajudes per realitzar les transaccions.
- Un i**nventari de pantalles** on es mostra el codi de cada pantalla, el nom, tipus de pantalla o utilització a què està destinada, usuaris, etc.
- Un document on es mostre **la jerarquia** de les pantalles principals (vegeu figura 5.8), és a dir, com es pot accedir o navegar (mapa web) entre les diferents pantalles que componen aquest apartat de la interfície d'usuari.

4.2. Disseny d'informes

Com s'ha mencionat, els informes constitueixen un dels principals apartats per a proporcionar als usuaris informació del sistema, encara que siga en format electrònic tipus documents en PDF.

Bàsicament es poden considerar part de la interfície d'eixida d'un sistema; però, a més, a vegades s'han de considerar documents que els usuaris o ens externs al sistema han d'emplenar per portar a terme un determinat procediment d'entrada de dades. Per exemple, els formularis bancaris que el client emplena prèviament a la realització de certes transaccions. Encara que la tendència és **eliminar al màxim l'ús de paper imprès**, el disseny de formularis, impresos, llistats i qualsevol tipus de document és fonamental per a mostrar i introduir informació al sistema.

Cliente - Total pedidos 28. Noviembre 2012
CRONUS España S.A. Pág. 1
SUPER

			Importe pedidos pendientes					
Nº cliente	Nombre		...antes	27/01/11 28/02/11	27/02/11 28/03/11	27/03/11 28/04/11	después...	Total
10000	GDE Distribución S.A.		1.290,00	0,00	0,00	0,00	0,00	1.290,00
20000	Sellafio S.L.		7.850,16	1.717,91	0,00	0,00	0,00	9.568,08
30000	Seguros Bella Vista S.A.		2.860,00	7.342,36	0,00	0,00	0,00	10.202,36
40000	Reno Diseño gráfico		8.605,60	5.133,32	0,00	0,00	0,00	13.738,92
50000	Servicio de aguas Deco		883,98	0,00	0,00	0,00	0,00	883,98
60000	Marcolanc Sonido		12.520,00	0,00	0,00	0,00	0,00	12.520,00
61000	El camino del sonido		352,00	1.318,00	0,00	0,00	0,00	1.670,00
62000	La Tienda Aparatos		160,00	3.654,00	0,00	0,00	0,00	3.814,00
01454545	New Concepts Furniture	USD	0,00	1.346,50	0,00	0,00	0,00	1.346,50
38128456	MEMA Ljubljana d.o.o.	SIT	0,00	717.364,20	0,00	0,00	28.462.086,80	29.179.451,00
43687129	Designstudio Gmunden		3.053,19	0,00	0,00	0,00	141.000,00	144.053,19
49525252	Beef House		0,00	0,00	0,00	0,00	33.973,22	33.973,22
49633663	Autohaus Mielberg KG		9.795,60	11.727,20	0,00	0,00	79.295,78	100.818,58
Total (DL)			47.370,64	84.880,79	0,00	0,00	573.289,00	466.820,33

Figura 5.9. Exemple d'informe sense ornaments (obtingut de Microsoft Dynamics NAV Classic)

Els informes es poden classificar pel **contingut** i per la seua **distribució**, a saber:

- Informes de detall com per exemple: fitxa de client, document de matrícula d'un alumne, vendes d'un client, factura, tiquet de venda.
- Informes de resum: com llistats de vendes per clients o articles, etiquetes per a l'enviament de correu, estadístiques, etc.
- Informes interns: resultats de les vendes, informes de productivitat, etc.
- Informes externs. Tiquets de venda, factures per a clients, nòmines, etc. Necessiten tenir uns formats especials que, a vegades, estan regulats per normatives oficials (declaracions per a Hisenda, informes per a la Seguretat Social, factures, etc.). En general, s'han de dissenyar tenint en compte la

imatge externa de l'organització, és a dir, han de ser atractius, tenir una aparença oficial i professional, utilitzar el logotip de l'empresa corresponent i incloure la informació necessària segons el tipus de document (CIF de l'empresa, data, tipus de document, etc.).
- Informes confidencials com les nòmines.

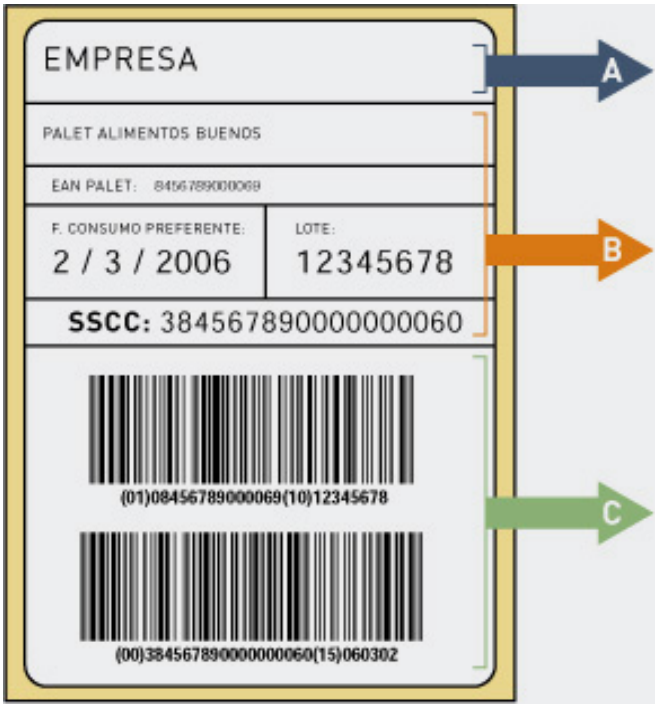


Figura 5.10. Exemple d'etiqueta amb codi de barres

L'informe ha de ser senzill d'utilitzar i ha de tenir un aspecte agradable. Les dades més importants han de ser les més fàcilment localitzables, en les primeres columnes o en les capçaleres. En general, en un informe es poden diferenciar tres parts: capçalera, cos i peu de pàgina. El disseny i contingut de cadascun d'aquests apartats depèn del tipus i de la destinació de l'informe. En general, en un informe es poden trobar els següents tipus de dades:

- **Dades identificadores**: excepte casos excepcionals, es mostra en el document la data, el número de pàgina i, en el cas d'un informe que s'obtinga més d'una vegada al dia, l'hora d'emissió. Alguns documents legals, com per exemple les factures han de portar informació fiscal de l'empresa impresa de forma adequada. Normalment s'ubiquen a la capçalera, als laterals de les pàgines o al peu.
- **Títols, capçaleres i logotips**: en qualsevol tipus de document s'ha de proporcionar informació perquè l'usuari o destinatari reconega el document del qual es tracta. Les capçaleres i els logotips són d'especial rellevància en documents de distribució externa. S'han de posar a la part superior i diferenciar-se clarament de la resta del document.
- **Files o resta del cos de l'informe**: molts documents mostren informació repetitiva que cal organitzar per files i columnes de forma adequada; per

exemple, factures, comandes, llistats de resultats, etc. En aquests casos, els noms de les columnes han de ser curts i significatius i estar alineats de manera uniforme. Els camps variables s'han d'alinear convenientment, els camps numèrics normalment s'han de justificar a la dreta, i els alfanumèrics a l'esquerra. Cal deixar algun espai entre les columnes, dos o tres espais són prou per poder distingir els valors i per poder seguir correctament les línies d'informació. L'ordre de les dades que es mostren en les línies de detall s'ha d'establir segons el seu significat, les dades més importants o identificadors a l'esquerra, i les dades referents a quantitats o imports, a la dreta.

Igual que en les pantalles, la forma de distribuir i d'ubicar la informació en els informes són fonamentals per a aconseguir l'acceptació i el seu ús correcte. Segons el tipus d'informe i l'objectiu al qual està destinat hi haurà consideracions particulars.

Passos per a dissenyar els informes bàsics, partint dels casos d'ús on hi ha actors humans que utilitzen documents (poden ser d'entrada o d'eixida), per cada cas d'ús:

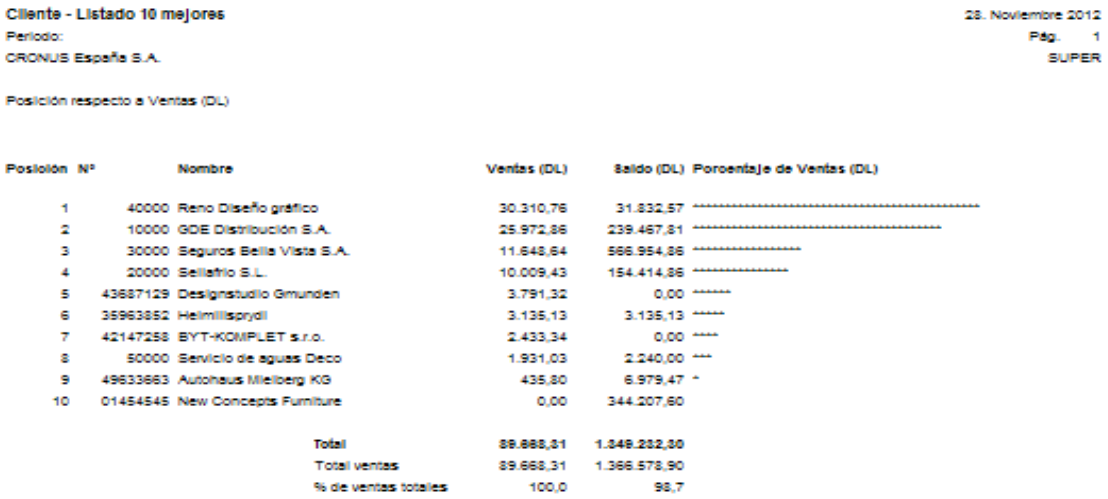
- Identificar els tipus d'informació i document que cal generar (també poden ser formularis per a introduir dades).
- Definir els tipus d'informes, objectiu, destinatari, finalitat legal, paper, distribució, etc.
- Definir una plantilla o estàndard per a cada tipus.
- Fer un llistat o taula (exemple taula 5.2) dels informes més importants i sobretot d'aquells que són d'ús especial.
- Valorar l'ús d'eines de generació d'informes o la creació de programari específic. Incloure en el disseny de pantalles les opcions necessàries per a generar els informes corresponents atenent el cas d'ús. S'ha de considerar la necessitat de crear operacions específiques en les classes.

Taula 5.2. Exemple de llistat d'informes.

Codi	Nom	Ús/objectiu	Usuari/destinatari	Tipus
IFC0001	Fitxa client	Dades de detall del client	Departament Comercial	Ús intern
ILC001	Llistat de nous clients mensual	Informació de nous clients	Departament Comercial	Ús intern

És important destacar que, en aquest apartat, només s'han de **dissenyar aquells informes que són fixos.** En qualsevol sistema d'informació és convenient l'existència d'un mòdul informatitzat que permeta els responsables de les àrees i de l'empresa obtenir informes en què ells mateixos puguen seleccionar les dades que volen obtenir, les comparacions que cal realitzar i el nivell de detall que desitgen visualitzar. Aquest tipus d'informes es generen normalment amb el suport de ferramentes

específiques de generació d'informes o ferramentes que proporcionen els sistemes de gestió de bases de dades per realitzar consultes, estadístiques, gràfics, etc.



Cliente - Listado 10 mejores 28. Noviembre 2012
Periodo: Pág. 1
CRONUS España S.A. SUPER

Posición respecto a Ventas (DL)

Posición	Nº	Nombre	Ventas (DL)	Saldo (DL)	Porcentaje de Ventas (DL)
1	40000	Reno Diseño gráfico	30.310,76	31.832,57	**********
2	10000	GDE Distribución S.A.	25.972,86	239.467,81	**********
3	30000	Seguros Bella Vista S.A.	11.648,64	566.954,86	**********
4	20000	Bellafilo S.L.	10.009,43	154.414,86	**********
5	43687129	Designstudio Gmunden	3.791,32	0,00	*******
6	35963852	Helmillsprydl	3.135,13	3.135,13	******
7	42147258	BYT-KOMPLET s.r.o.	2.433,34	0,00	*****
8	50000	Servicio de aguas Deco	1.931,03	2.240,00	****
9	49633663	Autohaus Mielberg KG	435,80	6.979,47	*
10	01454545	New Concepts Furniture	0,00	344.207,60	
		Total	**89.668,31**	**1.349.232,30**	
		Total ventas	89.668,31	1.366.578,90	
		% de ventas totales	100,0	98,7	

Figura 5.11. Exemple d'informe de resum

5. Resum

El disseny és una fase fonamental del desenvolupament d'un producte de programari. El seu objectiu és connectar la visió lògica desenvolupada en l'anàlisi amb l'activitat pròpia de la programació i altres tasques necessàries per posar el sistema en ús.

Segons els elements que constitueixen un sistema informàtic, el disseny s'encara a desenvolupar models que representen l'aspecte físic del sistema per a cadascun d'aquests elements: les interfícies (d'usuari i amb altres sistemes), els processos que ha de suportar el programari, els emmagatzematges o bases de dades i el maquinari (equips, xarxes o qualsevol altre dispositiu).

En aquest tema s'estudien aquestes activitats i, en particular, el disseny de pantalles i d'informes com a part del disseny d'interfícies d'usuari, els diagrames de seqüència per al disseny de processos i el refinament del diagrama de classes per a incloure-hi detalls de disseny.

El resultat del disseny genera tota la documentació necessària perquè l'equip d'enginyers en informàtica i programadors puga desenvolupar físicament els diferents components del sistema informàtic amb una proposta que assegura el desenvolupament d'un programari de qualitat i orientat a l'usuari.

Activitats complementàries

1. Busqueu en el diccionari els següents conceptes: interfície, pantalla, informe, mòdul, acoblament, cohesió, eficàcia, eficiència.
2. Dissenyeu una pantalla per a un procés que permet introduir el préstec d'una pel·lícula d'un sistema que dóna suport a un videoclub.
3. Dissenyeu un informe que proporcione la informació de tots els préstecs fets en un videoclub ordenats per títols de pel·lícula en el qual s'indique quantes vegades ha sigut prestat cada títol i quants dies en total. De quin tipus és aquest informe?
4. A partir d'alguna de les pantalles d'exemple del tema elaboreu un document descriptiu.
5. Busqueu diversos tipus d'interfícies d'aplicacions de gestió o del web per tal de comentar el seu disseny en funció dels paràmetres explicats en el tema.
6. Busqueu una factura de la llum, de telèfon, gas, etc., i feu una crítica del disseny del document des del punt de vista d'un usuari.

CAPÍTOL 6

Construcció i posada en marxa

Objectius

Els objectius específics d'aquest tema són que l'alumne siga capaç de:

- Comprendre i assimilar quines són les principals activitats que s'han de dur a terme en la fase final de desenvolupament d'un producte de programari, la seua construcció i posada en marxa.
- Saber la necessitat de tenir entorns independents de desenvolupament i proves, i d'explotació.
- Conèixer les diferents tasques que s'han de realitzar per a la conversió de les dades des de l'antic sistema informàtic fins al nou.
- Conèixer quines són les principals estratègies que es poden utilitzar per a l'entrega del producte a l'usuari.

1. Introducció

Aquest tema tracta la fase de construcció i posada en marxa d'un sistema informàtic. El propòsit d'aquesta fase és completar totes les activitats necessàries per a obtenir un programari fiable i adequat que puga ser instal·lat i usat per al propòsit definit inicialment. En aquest tema es parla d'implantació del sistema, i no únicament del programari, tenint en compte que la posada en marxa d'un producte de programari requereix que tots els components del sistema, del qual formarà part el programari, estiguen ubicats i preparats per a iniciar el seu funcionament.

Aquesta fase comporta el volum de treball més considerable de totes les que corresponen al desenvolupament d'un producte de programari pel que fa a l'esforç.[1] L'equip de treball d'aquesta fase està format habitualment per enginyers amb poca experiència (que participen com a programadors), programadors experts (depenent de la complexitat del desenvolupament), els caps de projecte i els usuaris que supervisen el resultat en l'activitat de les proves. Durant aquesta fase els caps del projecte supervisen, coordinen i validen tot allò que l'equip de programadors du a terme. Tenint en compte que és l'última fase i que té major durada temporal que la resta, el bon funcionament de l'equip i una correcta supervisió per part dels responsables del projecte és un factor crític per a finalitzar amb èxit.

Figura 6.1. Organització dels temes

A partir de les especificacions obtingudes en la fase d'anàlisi i de disseny del sistema, estudiades en els temes anteriors com es mostra en la figura 6.1, s'ha de completar el detalla de la lògica del programari, la codificació i les proves de les diferents unitats de programari que componen el producte final. Les activitats que es mostren en la taula 6.1 poden variar en forma i ordre depenent de quin paradigma o quines metodologies se seguisquen.

1. Aquest concepte i la proporció d'esforç de la fase de construcció es va estudiar en el tema 2.

Taula 6.1. Activitats de la construcció i posada en marxa del sistema.

ACTIVITATS I TASQUES	DESCRIPCIÓ-OBJECTIU
Revisió de la definició dels requisits, l'anàlisi i el disseny del sistema	Familiaritzar-se amb les especificacions del sistema, l'anàlisi i el disseny realitzats
Preparació de l'entorn de desenvolupament i preproducció Instal·lació del maquinari i programari necessaris per al desenvolupament Preparació de l'entorn de prova Definició dels procediments, operacions i estàndards de desenvolupament	Determinar els recursos de maquinari i programari necessaris per poder programar i provar els diferents mòduls del sistema informàtic per construir i instal·lar-los de forma adequada
Desenvolupament dels components de programari del sistema Disseny detallat (algoritmes, etc.) Programació Disseny i realització de les proves Preparació de la conversió	Completar el disseny a partir dels models i especificacions obtinguts en fases anteriors, generar el codi i les proves pertinents
Planificació i preparació de la conversió Planificar el projecte de conversió Identificar, recopilar i classificar dades a convertir Dissenyar processos i procediments de conversió Desenvolupar mòduls i programari necessari	Identificar, preparar i planificar com introduir en el nou sistema les dades i informació necessària per a començar a treballar que està en els sistema en funcionament
Desenvolupament dels procediments d'usuari i formació Preparació de manuals de formació i ajudes Desenvolupament del pla de formació dels usuaris Formació dels usuaris	Definir com s'han de desenvolupar els processos de l'empresa per part dels usuaris mitjançant el nou sistema i com s'ha de realitzar la formació dels usuaris
Posada en marxa del sistema Preparació de l'entorn d'explotació Realització de la Conversió Entrega del producte a l'usuari	Instal·lació del maquinari, programari i les bases de dades necessàries per a l'utilització del sistema per part dels usuaris Traspassar les dades necessàries de l'antic al nou sistema, per posar-lo en funcionament real i assegurar la seua acceptació per part de l'usuari Inciar l'ús del nou sistema

Al mateix temps es desenvolupen els procediments d'usuari, material per a la seua formació, manuals d'ús i ajudes, i es defineix un pla de formació. Cal identificar quina informació és necessària perquè el sistema puga començar a funcionar; aquesta informació haurà d'introduir-se abans de la posada en funcionament del nou

producte, mitjançant el que es denomina conversió o migració de dades. Encara que les activitats poden canviar o en part estar desenvolupades perquè s'ha seguit un model de desenvolupament de programari iteratiu com l'UP, en general s'identifiquen les activitats i tasques que es mostren en la taula 6.1 i que es descriuen en aquest tema.

Durant el desenvolupament del programari cal realitzar proves i per tant, tenir un entorn on els errors d'execució i les activitats pròpies de la programació no afecten els processos de negoci diaris, no modifiquen dades reals i no interferisquen en els treball dels usuaris. L'entorn per a desenvolupar programari ha d'incloure compiladors, depuradors i eines CASE entre d'altres, i ha de tenir programari i ubicacions físiques adients on els membres de l'equip puguen treballar de manera col·laborativa. Aquestes necessitats de configuració i resposta són molt diferents de les necessitats de l'entorn on s'implantarà el producte perquè siga utilitzat pels usuaris.

Una vegada el producte s'ha codificat, s'ha provat completament, els usuaris s'han format, s'ha preparat la plataforma de maquinari i programari on s'ubicarà i s'ha migrat la informació del sistema anterior necessària, el producte de programari està preparat per la seua utilització pels usuaris finals.

2. Preparació de l'entorn de desenvolupament i preproducció

Com s'ha descrit anteriorment, una de les primeres activitats és identificar les necessitats i definir els següents entorns:

- Un **entorn de desenvolupament**, format pel maquinari necessari per al treball de l'equip de desenvolupament i pel programari (editors i depuradors de codi, compiladors, programari gestor de projectes, ferramentes CASE, etc.) necessari per a la programació i prova dels diferents mòduls del sistema. Aquest entorn l'utilitzen els analistes i programadors i és totalment independent de l'entorn de treball habitual dels usuaris.
- Un **entorn de prova**, o **de preproducció**, que reproduïsca a escala reduïda el que serà el futur entorn d'explotació, per tal de provar el sistema en un entorn el més real possible i sense interferir en el treball i les dades reals dels usuaris.
- Un **entorn d'explotació**, que estiga format només pel maquinari i programari necessaris per al funcionament del sistema informàtic que utilitzaran els usuaris.

Actualment, els entorns de desenvolupament i de preproducció o prova són el mateix i se'n crea un de separat per a les proves de rendiment final.

Aquesta activitat de la fase de construcció i posada en marxa del sistema té l'objectiu de posar a disposició de l'equip de treball un entorn informàtic que permeta codificar i provar els mòduls del sistema informàtic en construcció independentment de l'entorn de treball habitual dels usuaris, a banda d'un estàndard de desenvolupament que unifique criteris a l'hora de programar. El resultat de l'activitat és l'entorn de desenvolupament i els procediments i estàndards de treball per a l'equip de desenvolupament. A continuació es mostren les diferents tasques que s'han de realitzar en aquesta activitat:

- Instal·lació del maquinari i programari necessaris per al desenvolupament: en el cas de desenvolupar un sistema informàtic que s'ha d'implementar i executar sobre un nou maquinari i/o programari de suport, no es pot començar a desenvolupar el sistema fins que el maquinari i el programari no hagen estat instal·lats i provats. La instal·lació del maquinari i programari de vegades es realitza per un personal especialitzat, però en qualsevol cas és necessari desenvolupar i provar totes les funcionalitats noves que el nou entorn proporciona. A més, a vegades és necessari instal·lar maquinari i/o programari específic per realitzar la codificació dels mòduls del sistema nou.
- Preparació de l'entorn de prova: en l'entorn de prova cal instal·lar el programari adequat per a mantenir la informació que permeta provar el sistema com si es tractara de l'entorn real d'explotació. Per tant, s'hi ha d'instal·lar el SGBD necessari, implantar una base de dades amb informació adient per a la prova, crear altres fitxers de dades necessaris per a les proves, etc.
- Definició dels procediments, operacions i estàndards de desenvolupament: els estàndards definits en aquesta activitat han d'assegurar que la documentació i els programes realitzats tinguen una certa uniformitat durant tot el desenvolupament del projecte. Açò permet obtenir un sistema més senzill d'operar i mantenir. En aquests estàndards cal establir el desenvolupament i ús de mòduls comuns, com per exemple anomenar els programes i variables, etc. Els noms dels camps ja tenen uns estàndards que han sigut assignats durant el disseny de la base de dades del sistema. Aquests estàndards han de ser utilitzats per tot l'equip de treball durant tot el desenvolupament del sistema.

3. Desenvolupament dels components de programari

En aquesta activitat la tasca a la qual es dedica la major part de l'esforç és la programació. En canvi, es presta poca atenció amb freqüència a la prova del sistema, la qual és una de les tasques principals per tal d'assegurar el desenvolupament de programari de qualitat. A continuació, es descriuen les principals tasques que es poden dur a terme en aquesta activitat.

3.1. Disseny de baix nivell dels components

El disseny de baix nivell comprén el disseny detallat del codi, a nivell de llenguatge algorítmic, i el disseny físic de la base de dades.

El disseny detallat té com a propòsit la definició de la lògica del codi de les diferents unitats de programació. Per a especificar la lògica dels programes procedurals es poden utilitzar diferents tècniques: algorismes, diagrames de Warnier, diagrames de flux de programes i diagrames d'activitats (UML), entre altres. En el paradigma orientat a objectes utilitzant el llenguatge UML, el programari a implementar està identificat en els mètodes de les classes. Normalment aquests mètodes són tan modulars que no requereixen d'especificacions detallades sobre la seua lògica. Aquesta fase i les tasques que inclou depenen del paradigma de programació emprat. En aquesta assignatura no s'aprofundeix més tenint en compte que els treballs per dur a terme es defineixen en nombroses assignatures d'algorítmica, estructures de dades, programació i paradigmes del programari. Com a diagrames d'UML es poden utilitzar el de classes amb els detalls d'implementació, el diagrama d'estats, el de desplegament (vegeu figura 6.2) i el de components (vegeu figura 6.3) per a l'arquitectura de maquinari i programari.

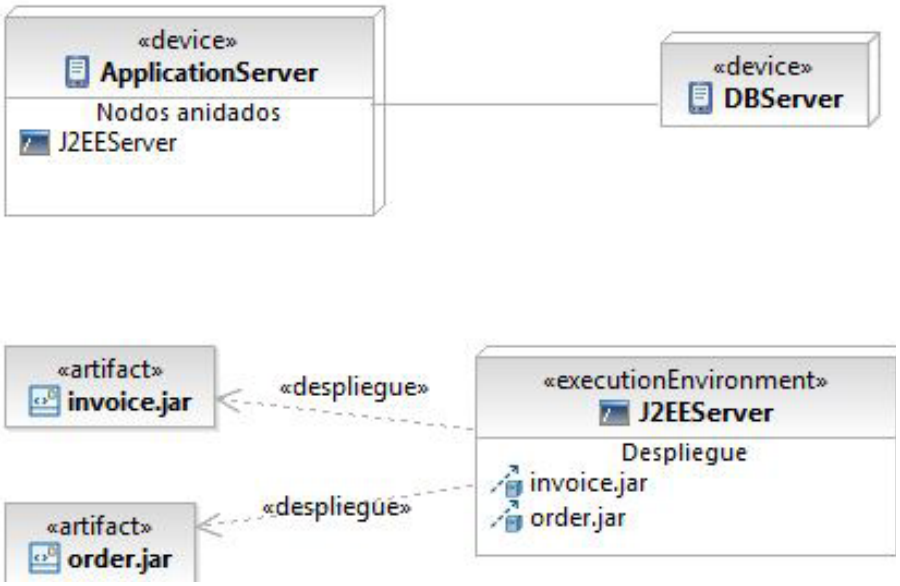


Figura 6.2. Exemple de diagrama de desplegament

A vegades, durant el disseny detallat dels diferents programes que formen part del sistema, s'identifiquen modificacions que és necessari incorporar a la documentació de les fases anteriors. Aquestes modificacions poden ser a causa, en part, de les característiques del maquinari i del programari necessari per al seu funcionament. Tenint en compte que es revisa i finalitza amb detall el disseny físic de la base de dades i fitxers auxiliars, potser que s'identifiquen requisits de dades nous que afecten la pròpia base de dades i també les interfícies. És important que qualsevol possible modificació siga incorporada abans de començar la programació, quan els canvis podrien suposar un cost afegit molt més elevat.

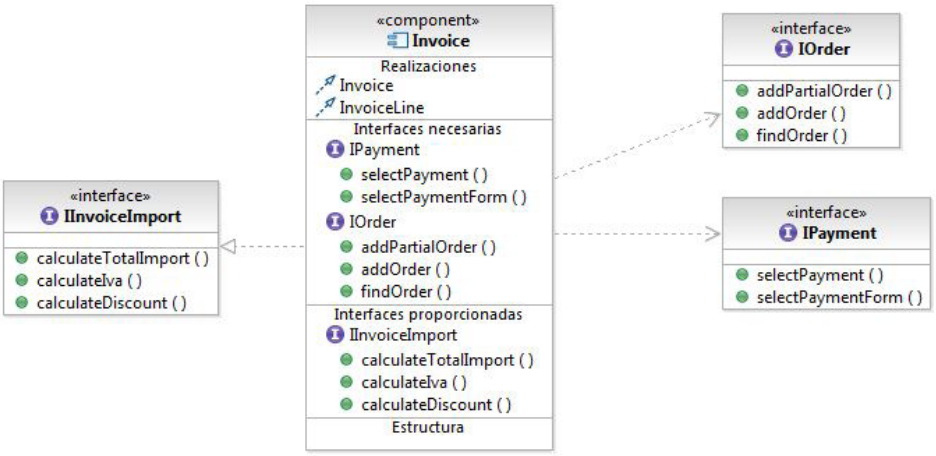


Figura 6.3. Exemple de diagrama de components

3.2. Programació

Habitualment és en aquesta tasca en la qual els enginyers informàtics amb poca experiència comencen a treballar encarregant-se de generar el codi font, els executables, la documentació del codi, realitzar les proves i documentar-les. Si el producte de programari té entre els seus components una base de dades, s'haurà d'implementar una versió adequada a l'entorn de desenvolupament i prova. S'haurà de crear un conjunt de dades inicials sobre les quals es puguen realitzar les proves unitàries i proposar mecanismes per a recuperar la informació cada vegada que la realització d'aquestes proves ho requerisca.

Per dur a terme aquesta tasca han de comptar amb les definicions dels programes que poden incloure-hi:

- El disseny detallat o especificació del codi definit anteriorment (lògica del programa).
- Les principals entrades i eixides (dades i informació de la base de dades), formats i suports físics.
- Les taules o fitxers de la base de dades que s'usen.
- El disseny de les interfícies.
- El disseny de les proves.

És important tenir definits mecanismes de coordinació i estàndards de programació i documentació perquè el treball d'equip siga efectiu el codi generat mantenible i puga integrar-s'hi sense problemes.

3.3. Disseny i realització de les proves

L'objectiu d'aquesta tasca és dissenyar un conjunt de dades de prova que permeta comprovar el correcte funcionament dels diferents programes realitzats. És important que aquestes siguen representatives de les dades reals, si ja hi ha informació mecanitzada pot fer-se una còpia sobre la qual s'han de desenvolupar les diferents proves. Si no hi ha informació mecanitzada s'han de crear fitxers de prova i introduir-hi informació significativa que siga d'utilitat per a la prova.

La programació està molt lligada a les proves. Durant la programació s'ha de provar cadascun dels mòduls individuals codificats. Després, tots els mòduls es proven de forma conjunta per a verificar que el sistema funciona correctament com un tot. Com s'ha descrit anteriorment, les proves han de dur-se a terme sobre dades semblants a les reals, però sense interferir en la informació real que pertany als usuaris, és a dir, en un entorn de preproducció.

4. Planificació i preparació de la conversió

La conversió consisteix en el conjunt de tasques necessàries per a traspassar o migrar aquelles dades que es troben a l'antic producte de programari (sistema d'informació automatitzat o no) i són necessàries per a iniciar el funcionament del nou. Aquestes dades poden estar en suport informàtic o no.

En aquesta tasca s'han d'especificar amb detall tots els fitxers, procediments i personal que seran necessaris per realitzar la conversió. A vegades la conversió es comença a realitzar amb anterioritat a l'activitat de preparació de l'entorn d'explotació, sobretot quan aquesta comporta un elevat esforç.

Les tasques que s'han d'efectuar durant aquesta preparació són:

- **Realitzar un pla de conversió**: el pla de conversió ha d'incloure totes les tasques que seran necessàries per al trasllat de les dades de l'antic al nou sistema. S'han d'estimar les dates oportunes per realitzar-lo, com també l'equip que cal utilitzar. Els analistes han de desenvolupar procediments que permeten verificar que la conversió es realitza de forma correcta com també accions que s'han de realitzar si es produeix alguna contingència, com per exemple documents o fitxers perduts, variació de formats, errors en la conversió de dades, omissió de passos, etc. Per tant, cal desenvolupar un document en què s'identifiquen tots els fitxers involucrats, una estimació del personal necessari i els procediments que cal desenvolupar per tal de completar la conversió. El document pot incloure per exemple:
  - Llistat de fitxers i arxius que s'han de convertir.
  - Llistat i disseny de processos específics que s'han de desenvolupar per a la conversió.
  - Identificació de les dades necessàries per a construir els arxius nous.
  - Llistat de documents nous i antics que s'ha d'utilitzar.

  - Identificació de tots els controls i programes per a verificar la informació de l'antic sistema i del nou.
  - Realització d'una estimació i un estudi de la periodicitat per a la conversió.
- **Desenvolupar procediments de conversió**: les activitats identificades en el pla de conversió s'han de desenvolupar mitjançant procediments que és necessari especificar i documentar. Aquests procediments són els que permetran a l'equip que realitze la conversió saber quines són les tasques en detall. Algunes vegades és necessari formar el personal que realitza la conversió.
- **Crear els fitxers de conversió**: aquests fitxers són els que contindran tota la informació que el sistema necessita per a la seua posada en marxa des d'un primer moment. La informació que s'ha d'introduir en aquests fitxers ha de ser validada i confirmada per a verificar que tots els fitxers contenen la informació correcta. A vegades no és possible crear en la preparació de la conversió tots els fitxers que necessaris posteriorment durant aquesta.

A més, existeixen altres tasques que és necessari tenir en compte per a la conversió com són **capturar i preparar dades** que fins al moment no estaven registrades en cap sistema, **actualitzar la informació** que es vol introduir, etc. Aquestes tasques depenen en part del mètode de conversió elegit i de la disponibilitat de la informació, com també de la importància de la informació que es vol tractar.

5. Desenvolupament dels procediments d'usuari i formació

L'objectiu d'aquesta activitat és dissenyar la documentació necessària que permetrà els futurs usuaris del sistema aprendre a usar-lo i a obtenir el major rendiment possible i formar als usuaris.

Durant l'anàlisi i el disseny es van identificar i es van dissenyar les funcions que satisfeien les necessitats dels usuaris per a cobrir els requisits del sistema. Els procediments d'usuari i la formació han d'informar l'usuari de com ha d'utilitzar les funcions que cobreixen els seus requisits. És a dir, han de constituir una interfície entre l'usuari i el sistema. Aquesta informació ha d'identificar qui, quan i com han d'usar la funcionalitat inclosa en el sistema A més, cal identificar el personal que ha de formar-se per a l'ús del futur sistema i establir períodes de temps i l'equip de treball per realitzar la formació.

Les especificacions funcionals són la principal informació d'entrada per al desenvolupament dels procediments d'usuari, els quals han d'incloure informació sobre quines funcions pot realitzar l'usuari i quines accions pot dur a terme en el cas que es produïsca algun error del sistema o de l'usuari. S'han d'incloure instruccions per a indicar com es pot recuperar la informació en cas de fallada, com s'han de fer còpies de seguretat, etc.

Les tasques que cal realitzar en aquesta activitat són:

- Desenvolupar procediments d'usuari: S'hi ha d'incloure la descripció dels processos que han de desenvolupar els usuaris tenint en compte l'estratègia i les normes de funcionament de l'empresa. Aquests procediments indiquen quines funcions del nou sistema informàtic s'han d'utilitzar en cada cas. Per documentar els processos de negoci o procediments es poden gastar tècniques de Modelització Empresarial com per exemple l'IDEF0 que es pot veure en la figura 6.2.
- Desenvolupar el pla de formació d'usuari: una vegada s'han identificat i desenvolupat els procediments d'usuari s'ha de desenvolupar un pla per a formar els futurs usuaris del sistema. Aquest pla ha de considerar el personal que estarà relacionat amb el sistema en un futur i la possibilitat que siga necessari formar personal específic per a algunes tasques com pot ser la conversió.
- Desenvolupar el material de formació d'usuari: s'han de desenvolupar exemples d'ús de les diferents funcions que poden efectuar-se amb el sistema perquè els usuaris puguen practicar durant el període de formació.
- Realitzar la formació d'usuaris i personal: finalment el personal que serà el futur usuari del sistema es forma. Aquestes sessions de formació es realitzen abans que el sistema estiga en funcionament. Cada grup d'usuaris es forma en la part específica del sistema que li correspon.

6. Posada en marxa del sistema

La posada en marxa inclou totes les tasques necessàries per a entregar el producte desenvolupat a l'usuari i que aquest puga utilitzar-lo amb totes les garanties de correcció i rendiment.

6.1. Preparació de l'entorn d'explotació

Aquesta activitat de la posada en marxa del sistema té l'objectiu de configurar el maquinari i programari de suport per tal que el sistema informàtic construït puga funcionar de forma adequada i ser utilitzat pels usuaris. El resultat de l'activitat és l'entorn d'explotació format pel maquinari i programari necessaris per als usuaris, el programari desenvolupat, la base de dades creada i els manuals d'usuari. Les tasques que s'han de dur a terme són les següents:

- Localització de l'espai físic.
- Instal·lació del maquinari (servidors, PC, impressores, dispositius de comunicació, etc.).
- Instal·lació del sistema operatiu i del SGDB. El sistema operatiu s'ha de provar i a vegades és necessari realitzar un període de formació d'aquest per als programadors i futurs usuaris, normalment personal tècnic.

- Implantació de la base de dades i dels fitxers de l'entorn d'explotació.
- Instal·lació del programari desenvolupat.
- Creació i entrega de manuals. Aquests manuals han de permetre els futurs usuaris obtenir tot el rendiment possible del sistema. Han de ser una guia d'ajuda en cas de dubte per a la realització d'alguna funció. Han de permetre identificar quines funcions pot realitzar cadascun dels usuaris i quins informes i pantalles de consulta proporciona el sistema. A vegades és necessari desenvolupar diferents manuals d'usuari, en el cas que hi haja diferents funcions lligades a diferents grups d'usuaris.

6.2. Conversió

Com s'ha comentat anteriorment, la conversió reuneix el conjunt de tasques que és necessari planificar i desenvolupar per a traslladar al nou sistema informatitzat tota la informació necessària de l'antic sistema d'informació en funcionament (vegeu figura 6.3.).

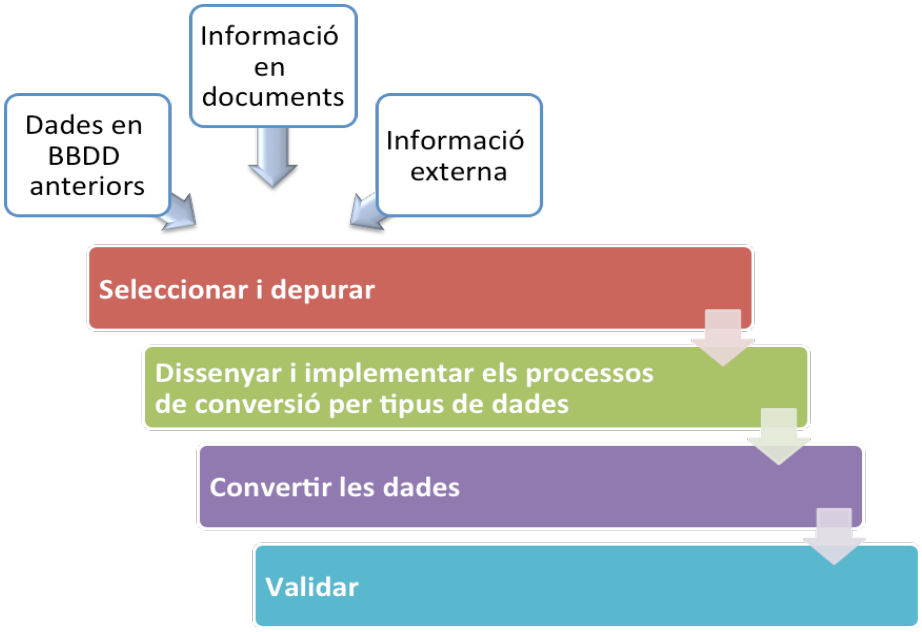


Figura 6.4. Esquema de la conversió

En aquesta tasca s'ha de seguir el pla de conversió dissenyat i preparat amb anterioritat. Els fitxers identificats i preparats anteriorment es modifiquen durant aquesta tasca, i s'han de crear els fitxers nous necessaris. La introducció d'informació normalment té un cost elevat en temps i a vegades requereix l'ús de programes específics. La informació recollida durant la conversió i introduïda en el sistema ha de correspondre's amb l'estat del sistema en el moment de posar-lo en funcionament.

6.3. Proves del sistema

L'objectiu de les proves és localitzar els errors no descoberts fins al moment i determinar fins a quin punt el sistema abasta els requisits especificats i la funcionalitat per a la qual ha sigut dissenyat. La prova no pot demostrar l'absència de defectes, només pot demostrar l'existència d'alguns defectes del sistema.

S'han de dissenyar proves que tinguen la màxima probabilitat de trobar errors amb el mínim esforç i temps, ja que és impossible realitzar una prova exhaustiva de cada programa. Per tant, es poden utilitzar **casos de prova** per a seleccionar les funcionalitats més representatives dels programes per tal de ser comprovades. Utilitzar casos de prova significa, en primer lloc, identificar conjunts de dades i condicions de prova que exerciten tots els requisits funcionals de cada programa i procés. I en segon lloc, realitzar una anàlisi dels resultats obtinguts per tal de determinar si eren els esperats o no, i si no ho són corregir els possibles errors que els han produït. Els errors que s'intenten localitzar amb els casos de prova són funcions incorrectes o absents:

- Errors d'interfícies.
- Errors en estructures de dades o bases de dades.
- Errors de rendiment.
- Errors d'inicialització o de fi.
- Resultats incorrectes. Per tal que els casos de prova siguen suficientment representatius s'han de considerar els següents aspectes:
  - Valors d'entrada que poden ser claus per al sistema, pels processos que s'han d'efectuar sobre aquests.
  - Volums de dades que suportarà el sistema.
  - Combinacions de dades i processos que poden afectar de forma específica el sistema.
  - Valors correctes i incorrectes de les dades d'entrada, els rangs i els valors límit dels rangs.

6.4. Entrega del producte a l'usuari

El lliurament del producte a l'usuari consisteix a entregar l'entorn d'explotació als usuaris perquè comencen a realitzar les seus tasques amb el programari desenvolupat i és, per tant, un moment crític.

Un producte que funcione correctament, perquè s'ha desenvolupat seguint mètodes robustos, per al quals els usuaris estan preparats gràcies a una formació adequada, i que pot entrar a ple rendiment d'ús perquè s'han inicialitzat les dades i s'ha preparat un entorn d'explotació complet proporciona a l'equip de treball la satisfacció d'haver fet un correcte projecte d'enginyeria informàtica i als usuaris la seguretat de tenir el producte que necessiten. Si alguns dels punts anteriors funciona de manera incorrecta el primer dia d'ús, els usuaris es decebran i serà difícil aconseguir amb èxit el funcionament final del producte.

Per a lliurar el producte als usuaris es poden seguir diferents estratègies. A continuació se'n descriuen algunes, encara que s'ha d'elegir aquella que millor s'adeque a les característiques del producte i de l'organització, combinant-les si es considera necessari:

- **Sistemes paral·lels**: quan ja hi ha un sistema en funcionament, el més segur és utilitzar durant un determinat període de temps aquests dos sistemes al mateix temps. Els usuaris segueixen operant amb el sistema anterior de la forma acostumada però comencen a utilitzar el sistema nou. Permet en el cas d'errors en el nou sistema o en el no funcionament d'algun procés, que la informació transferida fins a la data puga conservar-se en el sistema anterior. Els desavantatges d'aquest enfocament són, en primer lloc, que els costos en temps de processament de dades es dupliquen. En alguns casos és necessari contractar personal temporal per poder realitzar els processos en aquests dos sistemes. En algunes ocasions en què els usuaris saben que poden tornar al sistema antic, es produeix una certa resistència a la incorporació del sistema nou.
- **Directa**: es traspassa el funcionament del sistema anterior al nou de forma pràcticament immediata, a vegades en una nit o un cap de setmana. El sistema anterior deixa d'utilitzar-se en un moment planejat en el qual es realitza la conversió de la informació existent al nou sistema. Obliga els usuaris a treballar directament amb el sistema nou sense comptar amb l'antic per a res. Açò pot ser un desavantatge si sorgeixen problemes amb el nou sistema. Aquesta estratègia s'utilitza comunament per a introduir noves aplicacions. Necessita una preparació adequada i una aplicació ben dissenyada i provada per realitzar-se. És aconsellable tenir un equip de persones que estiguen preparades per a resoldre possibles problemes tant tècnics com d'usuari.
- **Enfocament pilot**: quan els sistemes nous impliquen canvis dràstics en l'organització és necessari plantejar el desenvolupament del sistema nou en una part de l'organització i implantar el sistema per àrees gradualment. Els usuaris de l'àrea en qüestió saben que estan provant un nou sistema i que poden realitzar-se canvis per a millorar o modificar algun procés. Quan el sistema està provat totalment, s'implanta en tota l'organització. El principal desavantatge és que si es produeixen massa errors els usuaris poden perdre la confiança en el nou sistema i objectar que el sistema proporciona més dificultats que avantatges.
- **Mètode per etapes**: aquest mètode s'utilitza quan no és possible instal·lar de colp un nou sistema. La conversió dels arxius, la capacitat del personal existent o la instal·lació del nou equip informàtic pot forçar que aquesta fase es realitze per períodes. Els llargs períodes de conversió poden crear dificultats tant si el sistema funciona correctament des del principi com si es produeixen errors. Els nous usuaris en aquests dos casos poden comunicar el seu entusiasme o desil·lusió als futurs de forma que la implantació final pot no ser un èxit. Quan els sistemes es desenvolupen per etapes han de treballar bé des de la primera conversió.

Després de totes aquestes tasques el sistema es considera operatiu i els usuaris tenen la capacitat i formació adequades per al seu ús complet.

7. Resum

En aquest tema es tracta la fase de construcció i posada en marxa del desenvolupament d'un sistema informàtic. Aquesta és la fase final d'un projecte de programari i habitualment la de més temps. L'objectiu és, a partir del resultats obtinguts en la fase anterior de disseny, obtenir un sistema producte de programari que puga usar-se amb ple rendiment. Per a això és necessari dur a terme diferents activitats: preparació de l'entorn de desenvolupament i d'explotació, desenvolupament dels components de programari i de procediments d'usuari, formació d'usuaris, migració de dades i, finalment, la posada en marxa del sistema.

Cal destacar que el moment del lliurament del producte als usuaris per al seu ús és crític. Els usuaris comencen a treballar amb el nou producte i l'èxit serà resultat, entre altres factors, de l'ús de mètodes robustos proporcionats per l'enginyeria del programari, de les capacitats i els coneixements de l'equip de desenvolupament.

Esborrar la imatge que es té del resultat d'un projecte d'enginyeria del programari, com es mostra en la figura 6.5, és un dels reptes de l'enginyeria del programari i dels professionals i docents d'aquesta matèria.

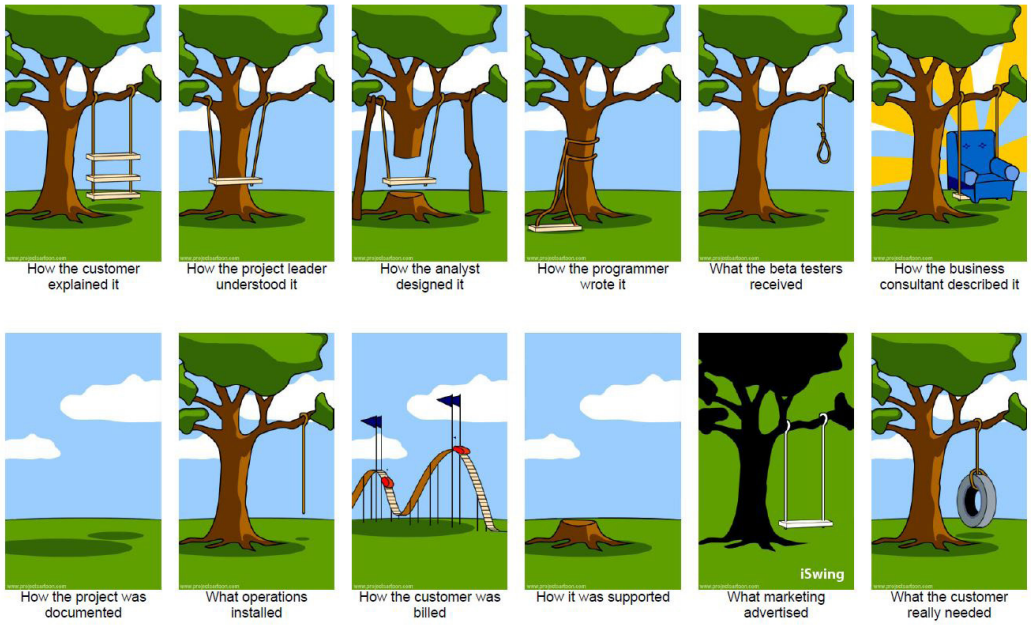


Figura 6.5. Com funciona realment el projecte. http://projectcartoon.com/cartoon

Bibliografia i referències

Agil Manifesto (2016). *Manifesto for Agil Software Development*, http://agilemanifesto.org

Bauer, F. L. (1972). «Software Engineering», Information Processing, 71.

Booch G., Rumbaugh, J., Jacobson, I. (2006). *El Lenguaje unificado de modelado: guía del usuario*, Madrid: Addison-Wesley, cop.

Brooks, Jr. F. P. (1995). *The Mythical man-month: essays on sotware engineering*, Addison-Wesley.

Durán Toro, A. Bernárdez Jiménez, B. (1999). *Metodología para el Análisis de Requisitos de Sistemas* Software Versión 2.2. http://www.lsi.us.es/~amador/index.php/Publicaciones_relevantes

Galitz, W. O. (1993). User-Interface Screen Design, John Wiley and Sons.

Gilb, T. (1988). *Principles of Software Engineering Managem*ent, Addison-Wesley.

IEEE STANDARD 1490-2011-IEEE Guide- *Adoption of the Project Management Institute(PMI(R))StandardAGuidetotheProjectManagementBodyofKnowledge* (PMBOK(R) Guide)--Fourth Edition IEEE. http://standards.ieee.org/findstds/standard/1490-2011.html

Jacobson, I., Booch, G., Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software*, Addison-Wesley, Madrid.

Kabor, M. (1990). *Software Desing Manifesto. Dr. Dobbs Journal.*

Karner, G. (1993). *Metrics for Objectory*, Sweden: University of Linköping.

Maciaszek, L.A. (2007). *Requeriments Analysis and System Desing*, Addison-Wesley. Chapter 2 and 3.

OMG The Object Management Group® (OMG®) http://www.omg.org

Piattini, M. G., Calvo-Manzano, J. A., Cervera, J., Fernández, L. (2004). *Análisis y diseño detallado de Aplicaciones Informáticas de Gestión*, Ra-Ma.

PMBOK® Guide and Standards http://www.pmi.org/pmbok-Guide-and-Standards.aspx.

Pressman, R. S. (2010). *Software Engineering. A Practitioner's Approach* (7a edició), Mc Graw-Hill.

Rumbaugh, J., Jacobson, I., Booch, G. (2000). *El Lenguaje Unificado de Modelado. Manual de Referencia*, Addison-Wesley, Madrid.

Shneiderman, B., Plaisant, C. (2006). *Diseño de Interfaces de Usuario*, Madrid, Addison-Wesley.

Sommerville, I. (2008). Sofware Engineering, 8a edició, Pearson Education.

Scrum Manager (2016). Scrum Manager® http://www.scrummanager.net/

Scrum.org (2016). The Scrum Guide. http://www.scrumguides.org/scrum-guide.html

UML Unified Modeling Language™ (UML®) http://www.omg.org/spec/UML/Current

XP (2016). *Extreme Programming: A gentle introduction* http://www.extremeprogramming.org/

Version One (2016). *What Is Agile Methodology?* https://www.versionone.com/agile-101/agile-methodologies/

Yourdon, E. (1993). *Análisis estructurado moderno*, Prentice-Hall Hispanoamericana, Mèxic.

Bibliografia complementària

Fiddy, R. (1991). *The Fanatic's Guide to Computers*. Exley Publications Ltd.

IDEF0 Function Modeling Method, http://www.idef.com/idef0.htm.

Karolak, D. W. (1996). Software engineering risk management / Dale Walter Karolak Los Alamitos, CA: IEEE Computer Society Press, cop.

Llamazares Redondo, F., Romero Roldán, J. R. (2010). *Planificación y control de proyectos con MS Project 2010*. Pozuelo de Alarcón, Madrid: ESIC, Dl.

Métrica V3 (2016). *Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información. Portal de la Adminsitración Electrónica* (PAE).

http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.VyddQHr-SyE

Vlaanderen, K., Jansen, S., Brinkkemper, S., Jaspers, E. (2011). *The agile requirements refinery: Applying scrum principles to software product management*, Information and Software Technology, Volume 53, Issue 1, January, pp. 58-70 (http://www.sciencedirect.com/science/article/pii/S0950584910001539).

Young, R. (2001). *Effective Requirements in Practice*, Addison-Wesley.

CAS PRÀCTIC: EasyRent

1. Enunciat del cas

El cas proposat consisteix a crear un portal web per a gestionar tant l'oferta com la demanda de lloguer d'allotjaments entre particulars. El promotor d'aquest portal és l'empresa EasyRent, que ja es dedica a aquest negoci i ho planteja per a poder ampliar el seu mercat, millorar el rendiment econòmic dels allotjaments, promocionar el lloguer entre particulars i fer arribar, tant l'oferta de lloguers com la demanda, a un públic més ampli.

La plataforma web ha de permetre als propietaris d'allotjaments oferir les seues propietats i als potencials inquilins consultar allotjaments disponibles i realitzar la reserva de lloguer.

Després de diferents reunions i recerca d'informació sobre sistemes actuals de gestió d'allotjaments entre particulars, s'ha desenvolupat una llista preliminar dels requisits del sistema. Aquestes necessitats s'han agrupat en quatre apartats que haurà de suportar el portal web: gestió d'allotjaments per part dels propietaris, consulta d'allotjaments per part d'inquilins potencials, gestió de les reserves de lloguers i gestió interna que duran a terme els responsables d'EasyRent.

1.1. Gestió d'allotjaments

Perquè la plataforma web siga àgil i col·laborativa, els propietaris seran els responsables de publicar les dades dels allotjaments disponibles amb les seues característiques i restriccions específiques.

En particular, el portal web ha de permetre als propietaris:

- Enregistrar-se la primera volta i accedir al portal amb un usuari i una contrasenya les vegades següents.
- Donar d'alta un nou allotjament amb les seues característiques tècniques, situació geogràfica, restriccions temporals de lloguer i preu. En aquesta primera versió del sistema, s'entén que es lloga l'allotjament complet, no es poden llogar habitacions individuals.
- Modificar les dades referents a un allotjament ja publicat.
- Eliminar la informació d'un allotjament, sempre que no tinga sol·licituds de lloguer pendents o ja realitzades.
- Cancel·lar la disponibilitat d'un allotjament, sempre que no tinga sol·licituds de lloguer pendents en el període cancel·lat.
- Consultar la informació sobre les reserves del seus allotjaments.

Les dades a mantenir quan un propietari es registre seran les següents: nom i cognom, NIF, correu electrònic, domicili, data d'alta en el sistema, telèfon, nom d'usuari i contrasenya.

Pel que fa a les característiques tècniques, de cada allotjament s'haurà de mantenir la següent informació: títol que ha d'aparèixer a l'anunci de promoció de l'allotjament, una xicoteta descripció en forma de text, tipus d'allotjament (apartament, adossat, casa, *loft*, bungalou, torre, xalet, etc.), capacitat total en nombre de persones, nombre d'habitacions, nombre de llits, nombre de banys, metres quadrats, ubicació de l'allotjament (carrer, número, planta, localitat), fotos, i serveis que ofereix (TV, aire condicionat, calefacció, cuina, si hi ha Internet, mascotes permeses, etc.). En aquesta primera implementació els propietaris introduiran un únic preu de lloguer per dia de reserva.

Pel que fa a les restriccions temporals de lloguer, el propietari podrà indicar quins dies està disponible per a llogar un allotjament. Podran ser dies solts, períodes complets entre dues dates o bé una data d'inici de disponibilitat sense límit. Aquesta disponibilitat serà utilitzada pel sistema a mode de filtre per a mostrar aquest allotjament només a inquilins que consulten dins d'aquest període de disponibilitat.

Si un propietari vol cancel·lar la disponibilitat temporal d'un allotjament, podrà introduir una data a partir de la qual l'allotjament no està disponible, però no s'esborrarà la informació.

1.2. Consulta d'allotjaments

Per tal de donar a conèixer l'oferta d'allotjaments, arribar a un públic més ampli, i facilitar la gestió del procés de lloguer, en el portal web els inquilins potencials podran consultar allotjaments d'una manera fàcil i flexible, així com sol·licitar-ne el lloguer mitjançant una reserva.

En particular, el portal web ha de permetre als inquilins potencials:

- Enregistrar-se la primera volta i accedir al portal amb un usuari i una contrasenya les següents vegades.
- Consultar la informació dels allotjaments disponibles segons les seues restriccions concretes sobre l'allotjament, nombre de persones, zona geogràfica i període temporal.
- Realitzar una reserva d'un allotjament seleccionat.

Les dades que caldrà mantenir quan un inquilí es registre seran les següents: nom i cognom, NIF, correu electrònic, domicili, data d'alta en el sistema, telèfon, nom d'usuari i contrasenya.

En les consultes de disponibilitat d'allotjaments es mostraran els períodes marcats pel propietari, incloent-hi com a no disponibles els que corresponguen a reserves confirmades.

1.3. Gestió de reserves

Per realitzar la reserva de lloguer de l'allotjament una vegada l'inquilí potencial haja seleccionat l'allotjament per reservar:

- L'inquilí haurà d'indicar quantes persones s'allotjaran i la data d'inici i fi de la reserva. El sistema haurà de calcular i mostrar el preu total.
- L'inquilí haurà de confirmar la realització de la reserva i se li demanaran les dades de la targeta de crèdit per a procedir al cobrament. El pagament es realitzarà mitjançant un sistema extern de pagament segur.
- El sistema generarà les dades perquè el servidor de correu electrònic (servei extern contractat per l'empresa) envie dos correus electrònics: un per a l'inquilí amb la informació de la reserva realitzada i pendent d'acceptar pel propietari, i un altre per al propietari informant-lo de la reserva que té pendent d'acceptació.
- El propietari podrà accedir al sistema per a confirmar o rebutjar les reserves pendents. Tant si el propietari accepta com si rebutja la reserva, l'inquilí rebrà un correu electrònic informant-lo de la situació.
- Quan es confirme una reserva per part del propietari es modificarà l'estat a acceptada, es generarà una factura (de vendes) per a l'inquilí que es registrarà al sistema i s'enviarà en format PDF amb un correu electrònic a l'inquilí.
- Quan els inquilins sol·liciten una reserva de lloguer el propietari tindrà 24 hores per poder acceptar-la o rebutjar-la. Passat aquest termini el sistema anul·larà la reserva i generarà l'ordre perquè s'envie un correu electrònic a l'inquilí i un altre al propietari informant-lo de la cancel·lació amb les dades de la reserva.

El correu enviat a l'inquilí quan aquest fa la reserva inclou el localitzador de la reserva, data i hora de reserva, característiques de l'allotjament reservat, dates d'entrada i eixida, dades personals tant del propietari com de l'inquilí i l'import total. Aquesta informació també es registrarà al sistema convenientment.

El correu enviat al propietari informant-lo de la reserva que té pendent de confirmació inclou la mateixa informació que el correu enviat l'inquilí.

La factura per a l'inquilí s'adjuntarà en format PDF al correu electrònic d'acceptació de la reserva. Les dades de la factura són un número únic i seqüencial, data de facturació, reserva a la qual correspon, inquilí al qual pertany, import, concepte i IVA.

En aquesta primera versió, no es contempla la gestió del pagament al propietari.

1.4. Gestió interna

El sistema ha de permetre al/s responsable/s de l'empresa EasyRent:

- Consultar dades referents a tots els aspectes de la plataforma (inquilins, propietaris, allotjaments, reserves).
- Donar de baixa tant a inquilins com a propietaris amb un comportament fraudulent.
- Donar de baixa ofertes d'allotjaments fraudulentes.
- Consultar informació sobre els pagaments a propietaris i cobraments d'inquilins.

2. Continguts del treball de pràctiques

Els continguts que es proposen han d'aparèixer tots en la memòria, encara que l'organització pot variar-la cada equip de treball atenent a l'ús dels generadors d'informes que proporcione l'eina CASE que s'use en pràctiques.

Taula 7.1. Índex dels continguts del treball pràctic.

Títol:EI1023 Memòria treball de pràctiques CASE: EasyRent Autors i data de lliurament 1. Introducció 2. Context i planificació del projecte 2.1. Descripció del cas 2.2. Objectius i abast 2.3. Equip de treball 2.4. Planificació inicial 3. Definició de requisits 3.1. Model de casos d'ús 3.1.1. Diagrama 3.1.2. Plantilles de descripció 3.2. Requisits de dades 3.3. Plataforma tecnològica 4. Anàlisi 4.1. Diagrama de classes 4.2. Documentació del model 5. Model de disseny 5.1. Interfície d'usuari gràfica 5.1.1. Classificació d'usuaris 5.1.2. Llistat de pantalles 5.1.3. Exemples de disseny gràfic de pantalles 5.2. Diagrama de classes de disseny (opcional) 6. Conclusions 7. Bibliografia	Title: EI1023 Practice work report CASE: EasyRent Authors and date 1. Introduction 2. Project context and planning 2.1. Case description 2.2. Goals and scope 2.3. Project team 2.4. Preliminary planning 3. Requirements definition 3.1. Use case model 3.1.1. Diagram 3.1.2. Use case templates 3.2. Data requirements 3.3. Technological proposal 4. Analysis 4.1. Class diagram 4.2. Class diagram documentation 5. Preliminary Design 5.1. Graphic User Interface 5.1.1. User classification 5.1.2. Listing of interfaces (screens) 5.1.3. Graphical screen design examples 5.2. Design Class diagram (optional) 6. Conclusions 7. Bibliography/References

3. Solució del cas

Es proporciona en aquest punt una proposta de solució dels principals apartats del treball de pràctiques. Encara que s'ha intentat ser el més pròxim al resultat que s'espera que els alumnes desenvolupen, hi ha punts, com per exemple la introducció del treball, que són diferents i que els alumnes hauran de desenvolupar perquè siga el que correspon al seu treball.

3.1. Objectius i abast del projecte i del producte

Objectius i abast del projecte

El projecte proposat té com a objectiu bàsic desenvolupar un sistema informàtic que permeta d'una manera flexible, àgil i eficient, la publicació, gestió i recerca de lloguers d'allotjaments per part de persones que ofereixen les seues propietats com d'inquilins potencials respectivament, mitjançant una plataforma web col·laborativa.

Aquest sistema informàtic inclou dos entorns: una plataforma web oberta per a la recerca, publicació i gestió de lloguers per part dels inquilins i propietaris, i un entorn web restringit que permetrà als empleats de l'empresa propietària del producte monitoritzar, controlar el funcionament correcte d'aquesta plataforma col·laborativa, així com avaluar i supervisar el seu negoci.

Pel que fa a l'abast del projecte en aquesta assignatura l'alumnat només arriba fins a la fase de disseny del sistema, la implementació del producte es farà en l'assignatura EI1027.

Objectius del producte

Els objectius operatius i tàctics, és a dir, a curt i mig termini, que es pretenen assolir amb la implantació d'aquest sistema són:

- Millorar la imatge de l'empresa, creant un portal que difonga les ofertes d'allotjaments de manera actual i usant les noves tecnologies.
- Arribar a més públic, tant potencials inquilins com propietaris, dotant aquests últims de la capacitat de publicar els seus propis anuncis d'allotjaments d'una manera àgil i flexible, així com de gestionar les reserves dels seus allotjaments.
- Incrementar la rendibilitat obtinguda dels lloguers, ja que el sistema permetrà una autogestió dels lloguers per part dels seus propietaris més àgil, la qual resultarà en un increment del volum de lloguers.
- Racionalitzar/reduir les necessitats de recursos humans de l'empresa, ja que el lloguer d'allotjaments a través d'aquesta plataforma web col·laborativa no requereix d'interacció directa amb el personal de l'empresa.
- Millorar i fer més àgil l'oferta d'allotjaments en lloguer, tenint en compte que es podrà consultar l'oferta i disponibilitat d'una manera actualitzada.

Abast del producte

L'abast del sistema inclou:

- Des del punt de vista organitzatiu, el sistema cobrirà la funcionalitat requerida per un agent immobiliari, i afectarà tant aquesta àrea com la de pagaments.
- Des del punt de vista funcional, el sistema ha de proporcionar la funcionalitat per al registre tant d'inquilins com de propietaris, la recerca d'allotjaments adequats a les necessitats de l'inquilí potencial, la publicació d'allotjaments disponibles per part dels propietaris i la part referent a la gestió dels lloguers (reserva del lloguer i pagament de l'inquilí). No s'inclou en aquesta primera versió la gestió dels pagaments als propietaris.
- Des del punt de vista informàtic, en aquesta primera versió el sistema s'haurà de connectar a un sistema extern de gestió de pagament segur amb targetes, per poder dur a terme el cobrament del lloguer als inquilins.

3.2. Planificació

En aquest apartat es fa una primera proposta de la planificació del projecte. Les tasques definides se seqüencien de manera temporal, ja que els alumnes desenvolupen els apartats a mida que s'avança en els continguts en les sessions de teoria de l'assignatura.

Les tasques definides i estructurades jeràrquicament es mostren en la figura 7.1, on es representa el diagrama de descomposició de treball (WBS).

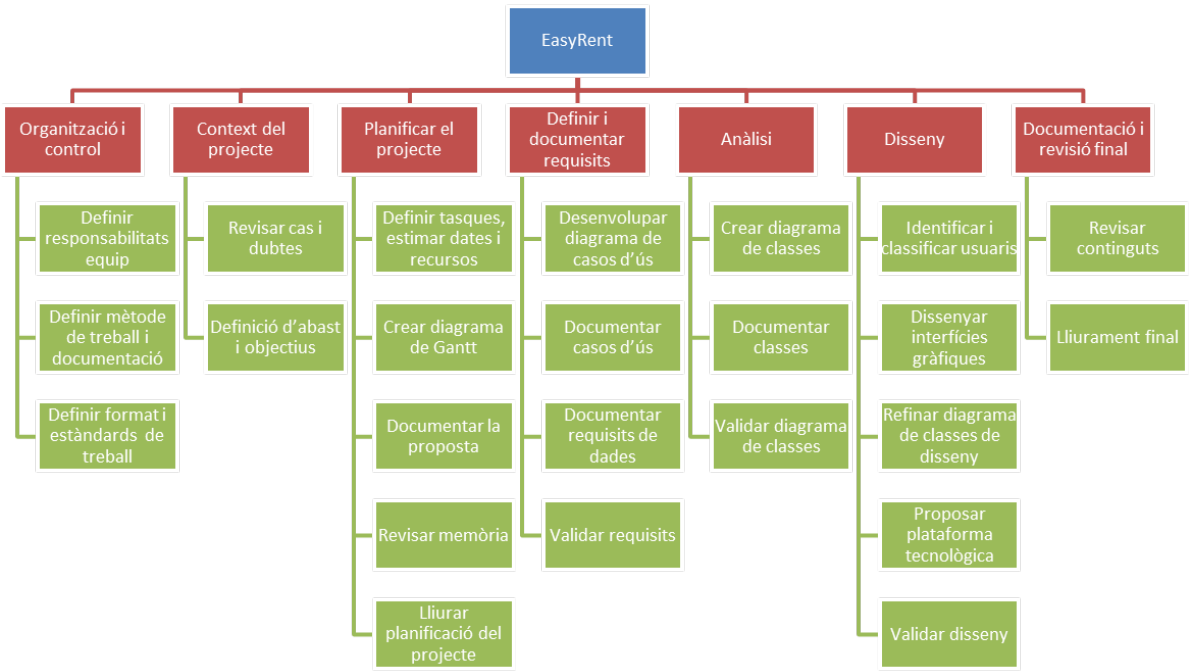


Figura 7.1. Diagrama de descomposició de treball per a planificar el projecte EasyRent

Una vegada identificades les tasques incloses en el projecte s'estima la seua durada, se seqüencien i s'organitzen temporalment amb la representanció del diagrama de Gantt que es mostra en la figura 7.2. No s'aplica cap mètode d'estimació de l'esforç, sinó que l'estimació de la durada de les activitats es basa en les dates de lliurament marcades per les restriccions temporals del projecte, que corresponen amb el calendari de les pràctiques de l'assignatura i els apartats del treball.

Es proposa com a recursos els rols que hi juguen els membres de l'equip de treball d'alumnes, i que són: Director de Projecte (PM), Responsable de Qualitat (QM) i Director Executiu (EM). A més a més, es considera que els diferents usuaris identificats en el cas també hi participen en les tasques.

Els passos seguits per a desenvolupar aquest diagrama de Gantt són:

- Identificar tasques a partir de la proposta de treball i dels lliuraments planificats.
- Identificar els lliuraments i revisions com a fites.
- Establir la durada per dies atenent la programació de les sessions dels laboratoris.
- Definir precedències i establir una seqüència temporal tenint en compte aquelles tasques que es poden fer en paral·lel. La majoria seran seqüencials, una darrere l'altra, ja que es duen a terme a mida que s'aprenen els conceptes en les sessions de teoria i en els laboratoris de l'assignatura.
- Ajustar la durada per poder complir amb els terminis marcats.
- Crear i assignar els recursos.

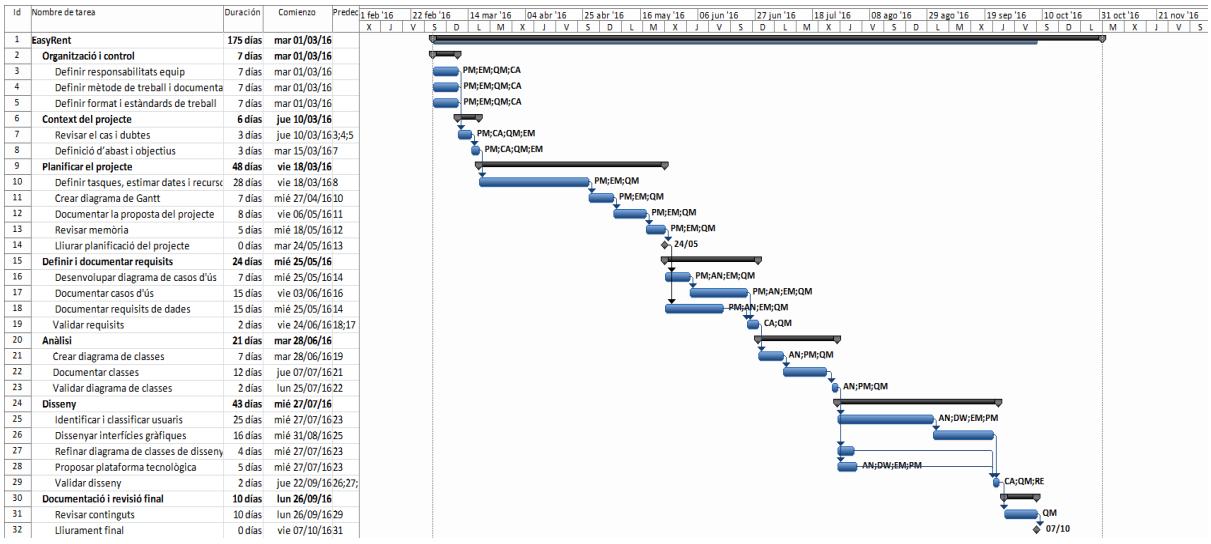


Figura 7.2. Diagrama de Gantt de la planificació del projecte EasyRent

3.3. Definició de requisits

Per a definir els requisits del sistema es parteix de l'enunciat del cas i es realitza una primera activitat de recerca d'informació de productes similars per poder tenir idees

de quines són les necessitats mínimes d'un sistema d'aquestes característiques. Es fan reunions de posada en comú i resolució de dubtes amb els promotors del producte.

3.3.1. Diagrama de casos d'ús

El diagrama de casos d'ús que es presenta en aquesta secció té com a objectiu documentar el comportament del sistema informàtic des del punt de vista de l'usuari. El primer pas és identificar els actors que hi apareixen.

Els actors identificats són:

- Promotor: representa les persones de l'empresa propietària del sistema. Tenen accés a tota la gestió interna del sistema i poden donar suport als propietaris i a possibles inquilins.
- Propietari: representa les persones externes a l'organització que registren alguna propietat en el sistema per ser oferta de lloguer.
- Inquilí: aquest actor representa tant les persones que només consulten informació de l'oferta d'allotjaments com les persones que es registren per realitzar una reserva de lloguer d'un allotjament.
- Sistema de Pagament Segur: per realitzar el cobrament de les reserves d'allotjaments el sistema es connectarà a una plataforma segura de pagament que està representada per aquest actor.
- Servidor de correu: representa el sistema de correu que usa l'empresa promotora al qual es connectarà el sistema de lloguer per a l'enviament de missatges en els processos que ho requerisquen.

Els casos d'ús s'identifiquen a partir de la funcionalitat que s'ha d'implementar descrita a l'enunciat i completada amb les activitats de recerca. Finalment, els actors i els casos d'ús es connecten mitjançant una associació, que representa que un actor fa ús directe del cas al qual està connectat. El diagrama es mostra en la figura 7.3.

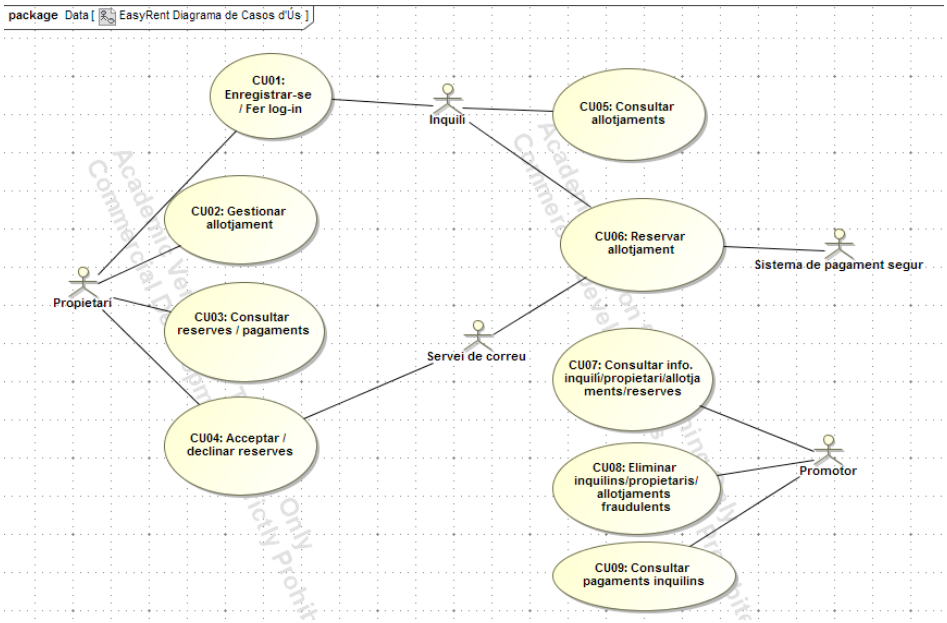


Figura 7.3. Diagrama de casos d'ús del projecte EasyRent

3.3.2. Especificació de casos d'ús

A continuació, es mostra l'especificació d'alguns casos d'ús usant la plantilla proposada en el tema 3. En cada plantilla es descriu un cas d'ús, així com la seqüència de passos de l'escenari principal.

Identificador: CU01 **Nom:** Enregistrar-se / Fer log-in **Autor:** VNR **Font:** Enunciat	Data creació: 26/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió: 0
Descripció: Permet tant al propietari d'un allotjament com a un potencial inquilí introduir les dades necessàries per a enregistrar-se en el portal web la primera vegada que s'accedeix, o accedir directament al seu compte si ja està enregistrat mitjançant un nom d'usuari i una contrasenya.	
Passos seqüència normal (Registre): El propietari/inquilí selecciona «Registre». El sistema mostra la interfície d'entrada de dades per al registre. El propietari/inquilí introdueix les seues dades. El sistema demana conformitat del registre. El propietari/inquilí confirma i queda registrat en el sistema. **Passos seqüència normal (Log in):** El propietari/inquilí selecciona «Log in». El sistema mostra la interfície per a autenticar-se. El propietari/inquilí introdueix el seu usuari i contrasenya. El sistema mostra el compte del propietari/inquilí. **Excepció** (Registre) Si alguna de les dades introduïdes no és correcta o no compleix les restriccions de seguretat (nom d'usuari no únic, contrasenya no vàlida, etc.) el sistema assenyala l'error i mostra de nou la interfície de «Registre». (Log in) Si el nom d'usuari o contrasenya no són correctes, el sistema mostra un missatge d'error i mostra la interfície de «Log in» de nou. **Precondició** Un propietari/inquilí només pot enregistrar-se si no ho ha fet mai prèviament. Un propietari/inquilí només pot fer «Log in» si s'ha enregistrat prèviament.	
Comentaris	

Identificador: CU02 **Nom:** Gestionar allotjament **Autor:** VNR **Font:** Enunciat	Data creació: 26/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió: 0

Descripció: Permet al propietari donar d'alta i modificar dades d'un allotjament. L'eliminació d'un allotjament només estarà permesa, si no hi ha sol·licituds de lloguer pendents o ja realitzades. En cas contrari, es podrà introduir una data a partir de la qual un allotjament no està disponible per a ser llogat sempre que no hi haja reserves durant aquest període.

Passos seqüència normal (Alta allotjament):

1. El propietari selecciona «Gestió d'allotjaments».
2. El sistema mostra la interfície de «Gestió d'allotjaments».
3. El propietari activa opció «Alta» i introdueix les dades de l'allotjament.
4. El sistema demana conformitat de l'alta.
5. El responsable confirma i es registra al sistema un nou allotjament.

Passos seqüència normal (Modificació allotjament):

1. El responsable selecciona «Gestió d'allotjaments».
2. El sistema mostra la interfície de «Gestió d'allotjaments».
3. El propietari selecciona un allotjament donat d'alta i activa l'opció «Editar».
4. El sistema mostra les dades de l'allotjament.
5. El propietari modifica les dades de l'allotjament.
6. El sistema demana conformitat de la modificació.
7. El propietari confirma.
8. El sistema guarda les noves dades.

Passos seqüència normal (Eliminació allotjament):

1. El responsable selecciona «Gestió d'allotjaments».
2. El sistema mostra la interfície de «Gestió d'allotjaments».
3. El propietari selecciona un allotjament i activa l'opció «Eliminar».
4. El sistema comprova que aquest allotjament no haja tingut mai sol·licituds de reserva.
5. El sistema elimina l'allotjament.

Excepció

Si l'allotjament ha tingut o té sol·licituds, el sistema informa el propietari que l'allotjament seleccionat no es pot eliminar.

Passos seqüència normal (Cancel·lació disponibilitat allotjament):

El responsable selecciona «Gestió d'allotjaments».
El sistema mostra la interfície de «Gestió d'allotjaments».
El propietari selecciona un allotjament donat d'alta i activa l'opció «Editar».
El sistema mostra les dades de l'allotjament.
El propietari modifica les dates de disponibilitat de l'allotjament.
El sistema comprova que no hi ha reserves pendents durant el període cancel·lat.
El sistema modifica les dades de disponibilitat.

Excepció
Si hi ha reserves pendents durant el període cancel·lat, el sistema informa el propietari que no es pot cancel·lar la disponibilitat.

Precondició
Només es pot modificar/eliminar/cancel·lar la disponibilitat d'allotjaments que han sigut donats d'alta.

Comentaris

Identificador: CU05 **Nom:** Consultar allotjaments **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió: 0

Descripció: Permet a qualsevol inquilí potencial consultar l'oferta d'allotjaments per a lloguer.

Passos seqüència normal:

L'inquilí potencial selecciona l'opció de «Consulta d'allotjaments».
El sistema mostra la interfície on es pot introduir el període temporal, zona geogràfica i nombre de persones.
L'inquilí introdueix els paràmetres de recerca.
El sistema mostra els resultats que coincideixen amb els paràmetres introduïts.
L'inquilí pot afinar més la seua recerca seleccionant altres paràmetres com el tipus d'allotjament, serveis que inclou, el rang de preu per dia, etc.
L'inquilí selecciona un allotjament.
El sistema dóna l'opció d'accedir al procés de reserva.

Excepció
Si no hi ha resultats que responguen als criteris, el sistema mostra un missatge i torna a la interfície de consulta.
Si algun paràmetre es deixa en blanc, el sistema mostra tots els resultats possibles.

Comentaris
Aquesta funcionalitat està disponible per a qualsevol persona sense necessitat de registrar-se en el sistema.

Identificador: CU06 **Nom:** Reservar allotjament **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió: 0
Descripció: Permet a qualsevol inquilí potencial realitzar la reserva de lloguer d'un allotjament.	
Passos seqüència normal: L'inquilí potencial selecciona l'opció de «Reserva» una vegada que ha seleccionat un allotjament. El sistema mostra el període temporal seleccionat, el nombre de persones i calcula el preu total. L'inquilí pot modificar alguna d'aquestes dades i el sistema recalcularà el preu total. Per a confirmar la realització de la reserva, el sistema demana a l'inquilí les dades de la targeta de crèdit per procedir al cobrament quan es confirme la reserva. L'inquilí confirma que accepta la transacció. El sistema mostra el missatge de la reserva feta amb èxit, envia un correu a l'inquilí amb la informació de la reserva de lloguer, un altre correu al propietari informant-lo que té una reserva pendent de confirmació i registra les dades de la reserva de lloguer. **Excepció** Si el sistema de pagament segur amb targeta informa que les dades de la targeta de crèdit no són correctes, es mostra un missatge al comprador i s'anul·la la reserva. **Precondició** Per poder reservar un allotjament, cal haver-lo seleccionat prèviament i l'usuari ha de haver fet «Log in».	
Comentaris Mentre la reserva de lloguer està pendent d'acceptació, la disponibilitat d'aquest allotjament durant el període reservat segueix estant disponible per a altres inquilins potencials.	

3.3.3. Requisits de dades

Identificador: RD01 **Nom:** Allotjament **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió:
Tipus: Dades a mantenir. **Descripció:** Propietats que es desitja llogar amb les seues característiques. **Dades específiques a mantenir:** Títol allotjament, descripció textual, tipus (apartament, adossat, casa, *loft*, bungalou, torre, xalet, etc.), capacitat total en nombre de persones, nombre d'habitacions, nombre de llits, nombre de banys, metres quadrats, ubicació (carrer, número, planta, localitat), fotos, serveis que ofereix (TV, aire condicionat, calefacció, cuina, si hi ha Internet, mascotes permeses, etc.), preu per dia, períodes disponibilitat.	

Comentaris:
Exemple:
Títol allotjament: Acollidor xalet a la vora de la mar.
Descripció textual: Es tracta d'un xalet situat a 100 metres de la mar amb totes les comoditats i de nova construcció.
Tipus: xalet
Capacitat total en nombre de persones: 4
Nombre d'habitacions: 2
Nombre de llits: 3
Nombre de banys: 1
Metres quadrats: 70
Ubicació: (carrer Mallorca, número 110, planta 2, porta 1, Benicàssim)
Fotos:
Serveis que ofereix (TV, aire condicionat, cuina, Internet, mascotes no permeses)
Preu per dia: 40 €
Períodes disponibilitat: 1/10 fins a l'1/12, 1/02 fins a l'1/04

Identificador: RD02 **Nom:** Propietari/Inquilí **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió:
Tipus: Dades a mantenir. **Descripció:** Dades sobre la persona que oferta l'allotjament o sobre la persona que sol·licita el lloguer. **Dades específiques a mantenir:** nom, cognoms, NIF, correu electrònic, domicili, data d'alta en el sistema, telèfon, nom d'usuari i contrasenya.	
Comentaris: Exemple: Nom: David Cognoms: García Fuster NIF: 54489302T Correu electrònic: dgarciaxxx@gmail.com Domicili: C/ Segòvia, núm, 34, 5é A, Conca Data d'alta en el sistema: 03/04/13 Telèfon: 604909321 Nom d'usuari: dagarcia Contrasenya: XXXXXXXX	

Identificador: RD03 **Nom:** Reserva lloguer **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió:
Tipus: Dades a mantenir / Dades d'eixida (PDF amb la reserva del lloguer). **Descripció:** Dades sobre la sol·licitud de lloguer que fa un inquilí potencial. **Dades específiques a mantenir:** localitzador de la reserva, data i hora de reserva, característiques de l'allotjament reservat, dates d'entrada i eixida, dades personals tant del propietari com de l'inquilí, import total, estat de la reserva.	

Comentaris:
A més de registrar la reserva al sistema, s'envia un correu electrònic tant al propietari com al potencial inquilí informant-los de la reserva pendent de confirmació.
Exemple:
Localitzador de la reserva: AXF949993
Data i hora de reserva: 22/07/2014 16:35
Característiques de l'allotjament reservat: ref. a un allotjament
Dates d'entrada i eixida: 16/08/2014 al 23/08/2014
Dades personals propietari: X
Dades personals inquilí: Y
Import total: 258 €
Estat de la reserva: pendent de confirmació

Identificador: RD04 **Nom:** Pagament inquilí **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió:

Tipus: Dades a mantenir / Dades d'eixida (factura en PDF).
Descripció: Dades sobre la factura de pagament a l'inquilí que es genera al acceptar una reserva de lloguer.
Dades específiques a mantenir: l'estat de la reserva canvia a «confirmada». A més a mes, es registra una factura amb les següents dades: número únic i seqüencial, data de facturació, reserva a la qual correspon, inquilí al qual pertany, import, concepte i IVA.

Comentaris:
Quan una reserva és confirmada per un propietari, a més de canviar el seu estat intern a «confirmada» i enregistrar-se la factura corresponent al sistema, s'envia un correu electrònic a l'inquilí amb la factura de pagament amb format PDF.
Exemple:
Número únic i seqüencial: A000001
Data de facturació: 23/07/2014
Reserva a la qual correspon: AXF949993
Inquilí al que pertany: X
Import: 258 €
Concepte: Z
IVA: Y

Identificador: RD05 **Nom:** Cancel·lació de reserva **Autor:** VNR **Font:** Enunciat	Data creació: 27/08/2015 Data revisió: 18/01/2016 Data aprovació: Versió:

Tipus: Dades a mantenir / Dades d'eixida.
Descripció: Dades que es modifiquen en el sistema i que es mostren quan, o bé un propietari cancel·la una reserva, o bé el propietari no atén la reserva durant les 24 h. següents a la seua generació i, per tant, el sistema la cancel·la automàticament.

Dades específiques a mantenir/mostrar: en el sistema, l'estat de la reserva canvia a «cancel·lada». Al propietari se li envia un correu electrònic amb les dades de la reserva que ha sigut cancel·lada: localitzador de la reserva, data i hora de reserva, característiques de l'allotjament reservat, dates d'entrada i eixida, dades personals tant del propietari com de l'inquilí i import total.
Comentaris: Exemple: Localitzador de la reserva: AXF949993 Data i hora de reserva: 22/07/2014 16:35 Característiques de l'allotjament reservat: ref. a un allotjament Dates d'entrada i eixida: 16/08/2014 al 23/08/2014 Dades personals propietari: X Dades personals inquilí: Y Import total: 258 € Estat de la reserva: Cancel·lada

3.4. Anàlisi

En aquest apartat es mostra una part de l'anàlisi del sistema amb el diagrama de classes bàsic desenvolupat a partir dels requisits de dades. No s'inclouen conceptes avançats d'aquest tipus de diagrames perquè no apareixen en les competències de l'assignatura.

El diagrama de classes d'UML2 amb atributs i operacions es representa en la figura 7.4.

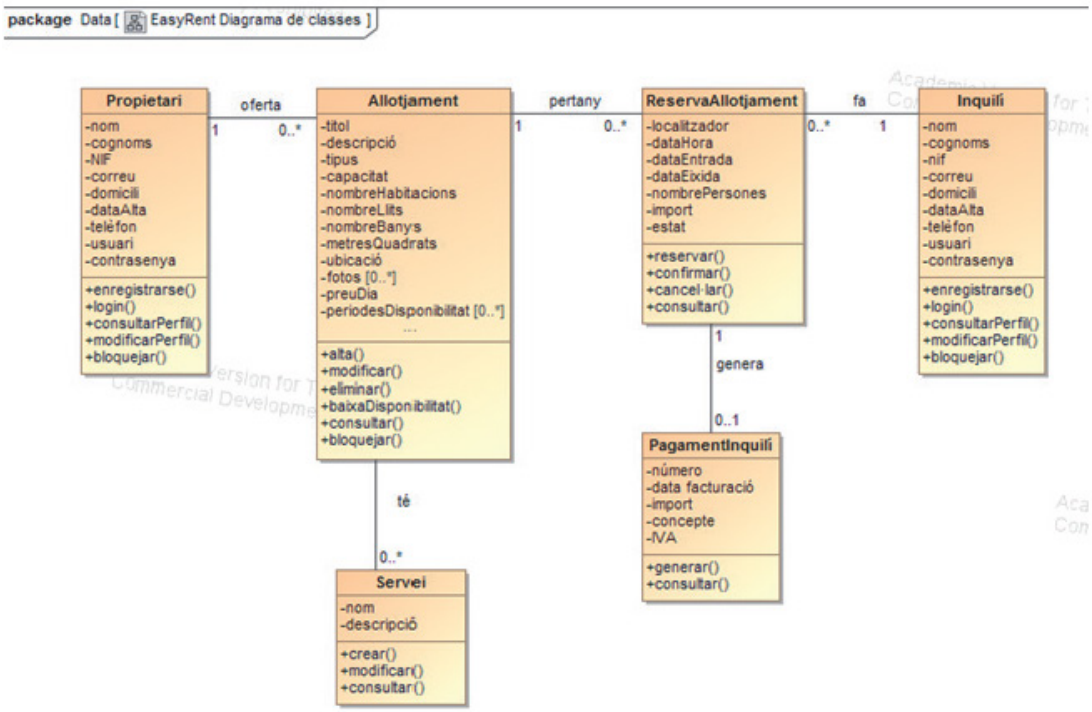


Figura 7.4. Diagrama de classes del projecte EasyRent

3.5. Disseny del sistema

En aquest apartat s'inclouen alguns dels continguts del model del disseny. El cas serà completat en altres assignatures. En concret, es proporcionen el disseny d'interfícies d'usuari i el diagrama de classes de disseny.

3.5.1. Diagrama de classes de disseny

Aquest diagrama (figura 7.5) mostra detalls de la funcionalitat afegint-hi operacions i signatures i detalls als atributs.

3.5.2. Disseny d'interfícies d'usuari GUI

Per dur a terme el disseny d'interfícies d'usuari el primer pas és identificar els grups d'usuaris, les seues necessitats, i la seua formació i capacitació sobre el sistema i l'ús d'interfícies. El resultat es mostra en la taula 7.2.

Taula 7.2. Usuaris i perfils identificats en el cas EasyRent.

Usuari	**Experiència**	**Freqüència d'ús**	**Casos d'ús**
Gerent/directiu de l'empresa promotora	Alta	Habitual	CU07: Consultar Info Inquilins, propietaris, apartaments i reserves CU08: Eliminar Info Inquilins, propietaris, apartaments i reserves CU09: Consultar Pagaments
Personal administratiu de l'empresa promotora	Alta	Habitual	CU07: Consultar Info Inquilins, propietaris, apartaments i reserves CU08: Eliminar Info Inquilins, propietaris, apartaments i reserves CU09: Consultar Pagaments
Propietari	Mitja	Mitja	CU01 Enregistrar-se / Fer log-in CU02 Gestionar allotjament CU03 Consultar reserves i pagaments CU04: Acceptar i declinar reserva
Inquilins potencials	Mitja	Eventual	CU01 Enregistrar-se / Fer log-in CU05 Consultar allotjaments
Inquilins amb reserves	Mitja	Mitja	CU01 Enregistrar-se / Fer log-in CU05 Consultar allotjaments CU06 Reservar allotjament

Una vegada identificats els usuaris i tenint en compte la funcionalitat representada en els casos d'ús, es crea una primera llista de pantalles o finestres. Es defineixen uns estàndards pel que fa a l'aspecte gràfic i a aquells elements comuns de les diferents pantalles i es creen els esbossos usant una eina de «mockup». En la taula 7.3 es mostren aquests esbossos realitzats per a les principals interfícies.

Taula 7.3. Llistat inicial de les interfícies gràfiques d'usuari.

Interfície	Tipus	Usuaris
IG001 Registrar-se/login com a inquilí o propietari	Finestra Conversació	Propietari/ Inquilí
IG002 Recuperar pwd.	Finestra Conversació	Propietari/ Inquilí
IG003 Missatge d'error d'accés	Finestra emergent	Propietari/ Inquilí
II001 Consultar/modificar info inquilins	Finestra Conversació	Administratiu i Gerent
II002 Llistat d'inquilins	Llista i selecció	Administratiu i Gerent
IP001 Consultar/modificar info propietaris	Finestra Conversació	Administratiu i Gerent Propietari
IP002 Llistat propietaris	Llista i selecció	Administratiu i Gerent
IA001 Consultar/modificar info allotjaments	Finestra Conversació	Administratiu i Gerent Propietari
IA002 Llistat allotjaments	Llista i selecció	Administratiu i Gerent Propietari
IR001 Consultar/modificar info reserves	Finestra Conversació	Administratiu i Gerent Propietari/ Inquilí
IL001 Consultar pagaments	Llista	Administratiu i Gerent Propietari
II001 Crear/modificar/anul·lar info inquilí	Finestra Conversació	Administratiu i Gerent Inquilí
IA002 Consultar info allotjaments per a inquilins	Llista o pantalla amb imatges	Inquilí
IR002 Fer reserva	Finestra Conversació	Inquilí
IR003 Consultar/modificar reserves pròpies	Finestra Conversació	Inquilí
IL002 Consultar pagaments propis	Llista	Inquilí
IP001 Consultar/modificar info propietaris	Finestra Conversació	Propietari

IA001 Consultar/modificar info allotjaments propis	Finestra Conversació	Propietari
IA002 Llistat allotjaments propis	Llista i selecció	Propietari
IR001 Consultar reserves dels seus allotjaments	Finestra Conversació	Propietari
IR004 Acceptar/declinar reserva	Finestra Conversació	Propietari
IL003 Consultar pagaments de les seues propietats	Finestra Conversació	Propietari
IG00n Finestras emergent de missatge d'error en cada cas	Llista i selecció	

Es mostren en les figures 7.5, 7.6 i 7.7 algunes de les pantalles generades en una eina de *mock up* com a primera versió del disseny final de les interfícies.[1]

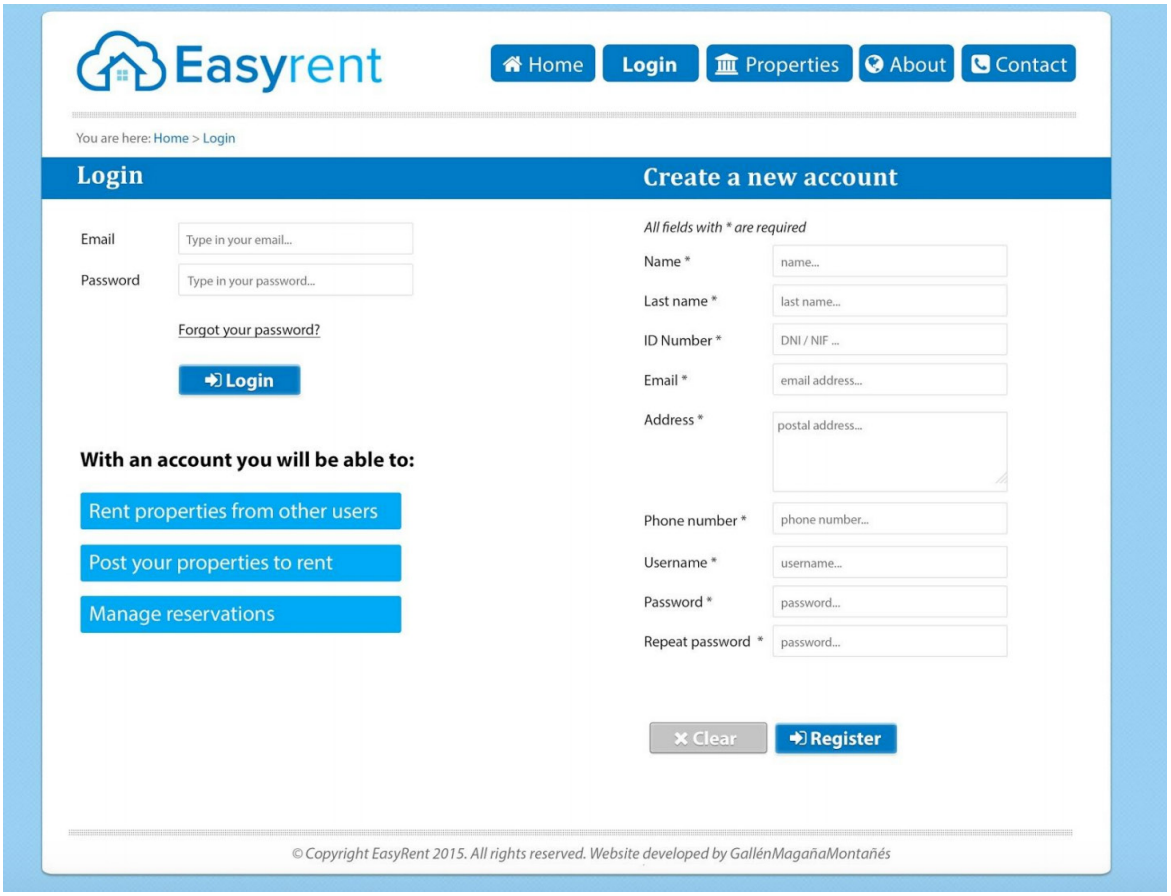


Figura 7.5. Interfície «Login Usuari»

1. Aquests dissenys d'interfícies han sigut creats per un dels grups d'alumnes de l'assignatura del curs 2015/2016. Es mostren les imatges amb el seu consentiment.

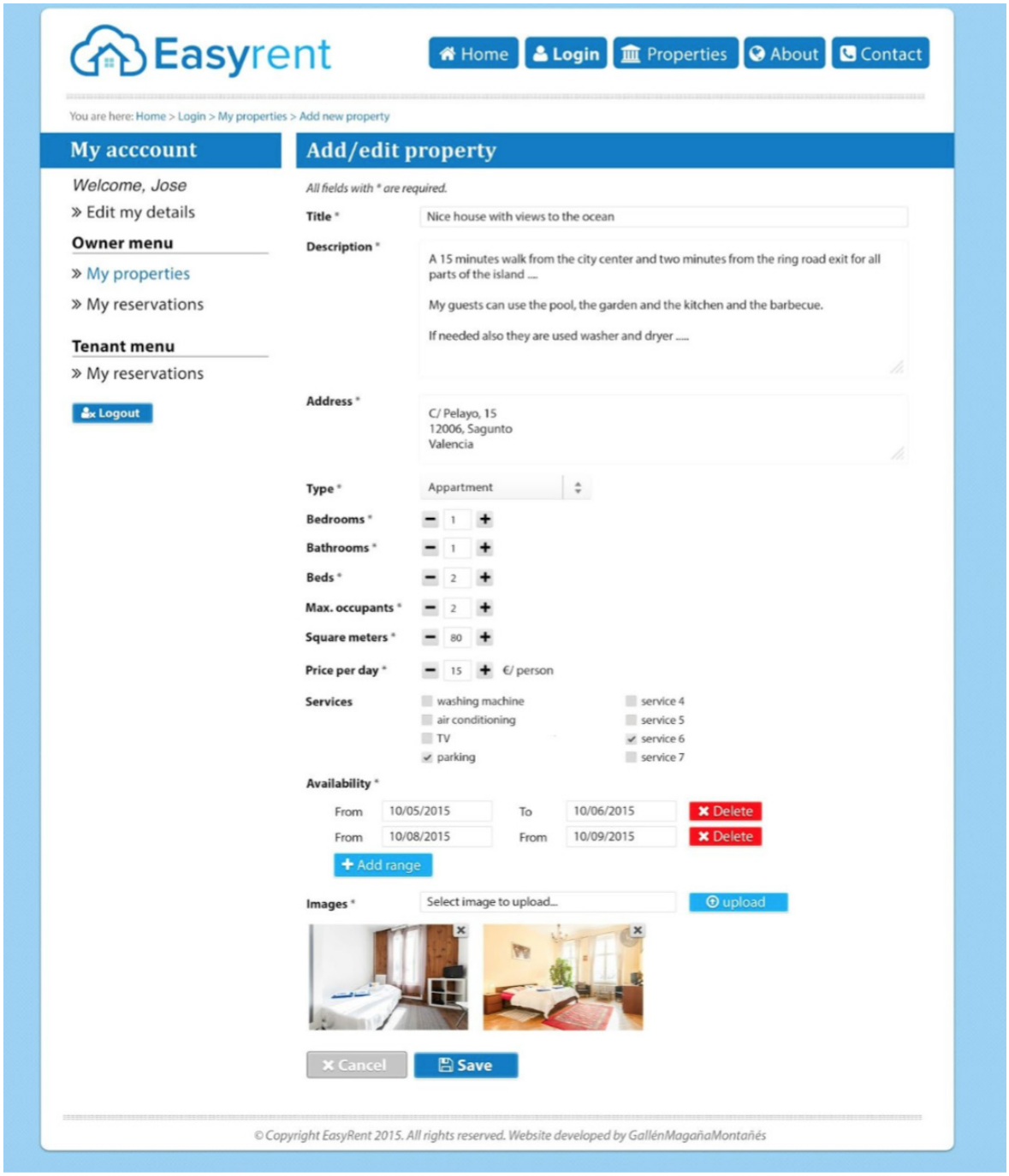


Figura 7.6. Interfície «Crear i modificar allotjaments»

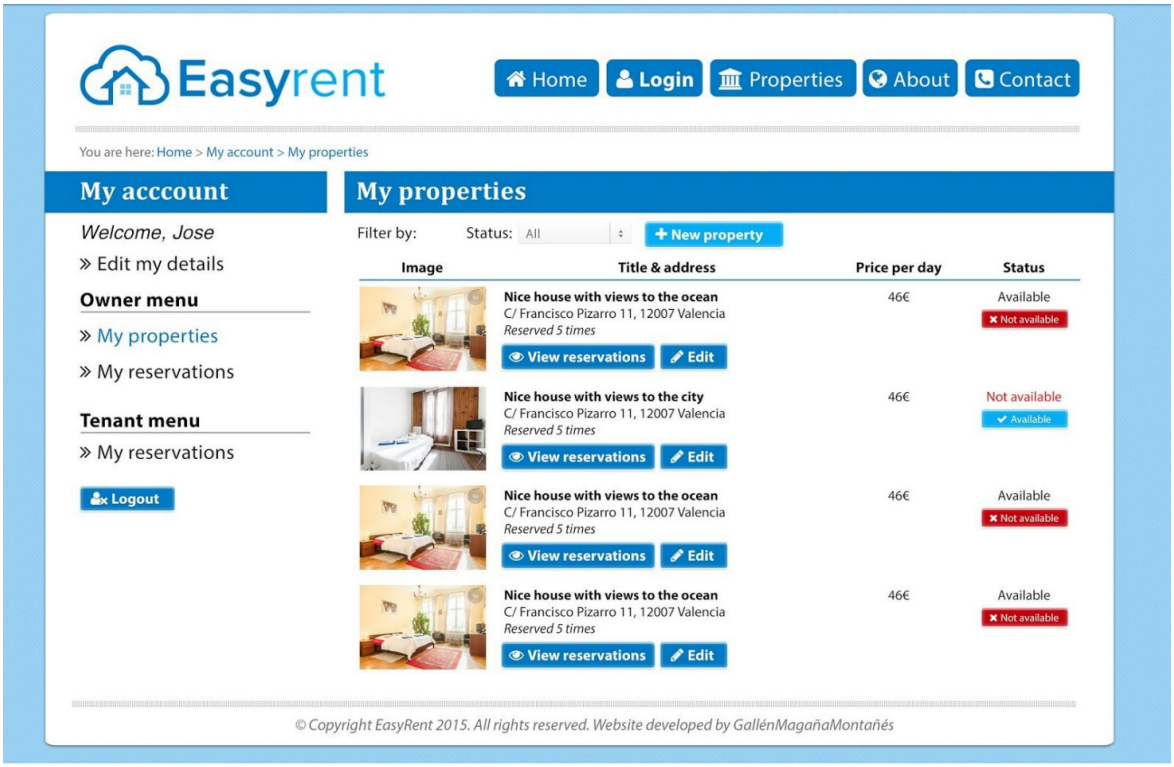


Figura 7.7. Interfície «Llistat d'allotjaments»

3.5.3. Disseny d'interfícies d'usuari: Informes

En aquest cas pràctic és necessari dissenyar els documents que, en format PDF, es generen per confirmar una reserva o facturar. Es mostra en la figura 7.8 un model dissenyat per a la factura d'un lloguer.

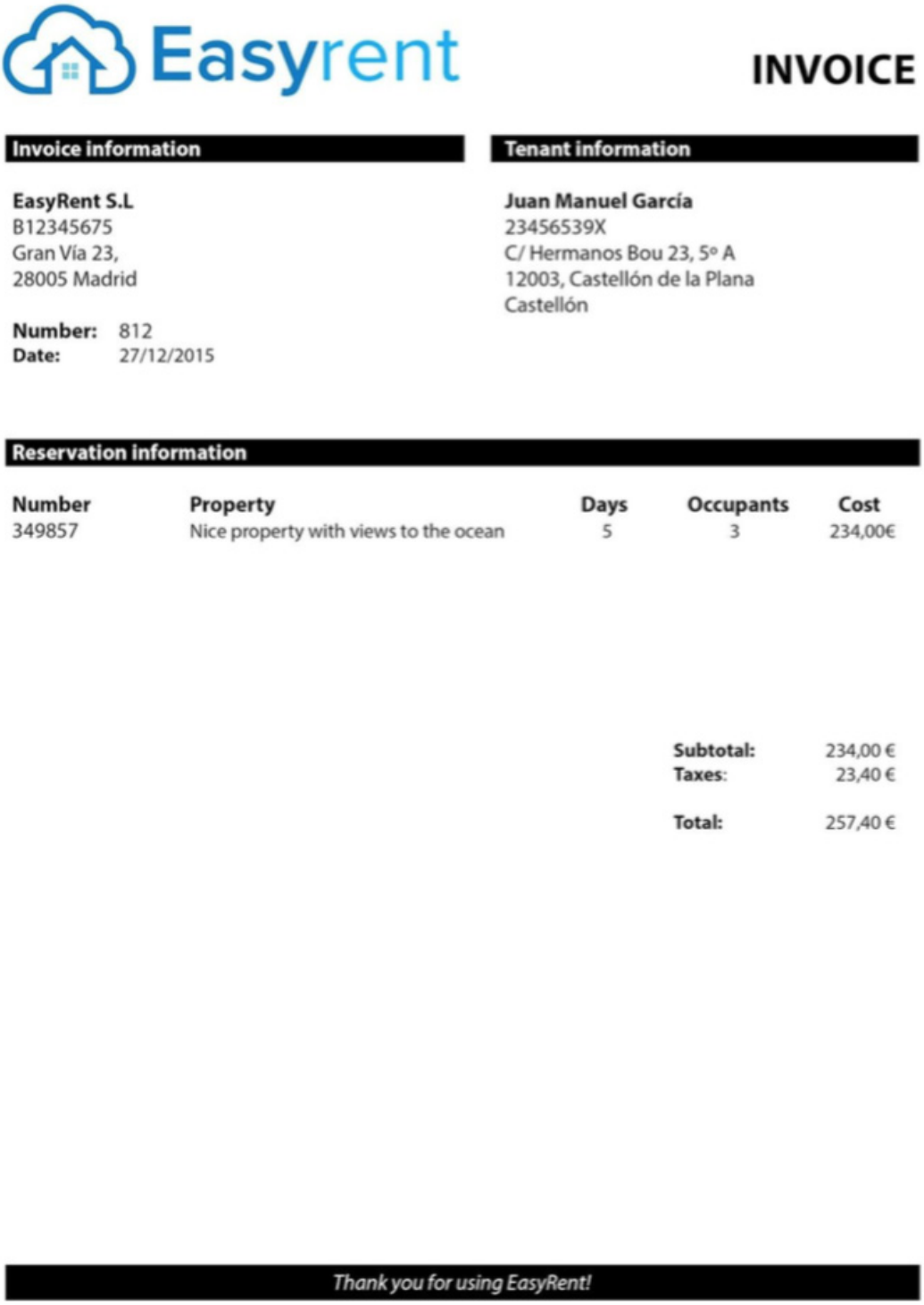


Figura 7.8. Disseny d'una factura

Índex de figures

Índex de taules