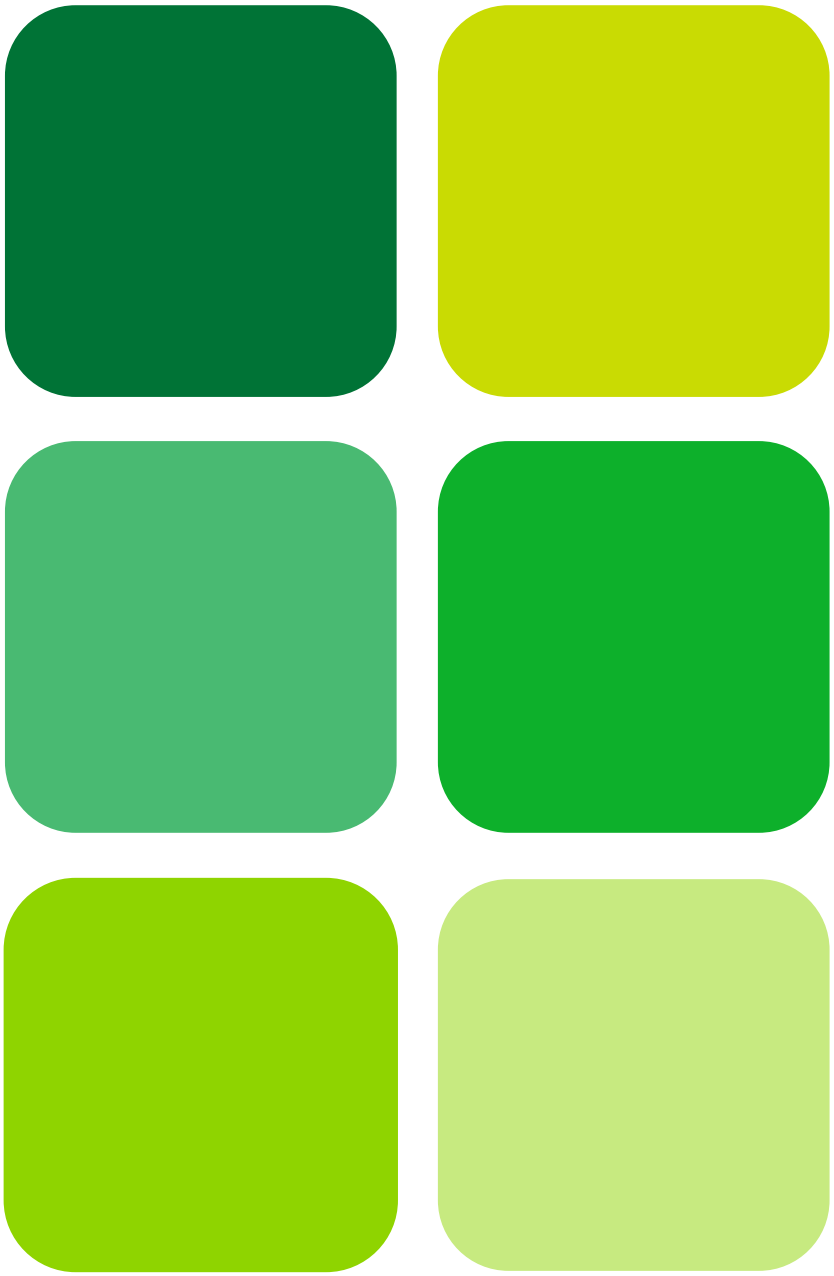


www.sapientia.uji.es | 50

UNIVERSITAT JAUME·I

Apunts d'Enginyeria del Programari de Gestió I

Cristina Campos Sancho
Reyes Grangel Seguer
Vicente Verde Peleato

Apunts d'Enginyeria del Programari de Gestió I

Cristina Campos Sancho
Reyes Grangel Seguer
Vicente Verde Peleato

UNIVERSITAT JAUME·I

Departament de Llenguatges i Sistemes Informàtics

■ Codi d'assignatura IG16

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, 50
www.sapientia.uji.es
Primera edició, 2011

ISBN: 978-84-694-2870-2

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

Índex

Índex de taules

Índex de figures

Resum

Aquest text recull els materials docents que s'utilitzen en l'assignatura IG16, Enginyeria del programari de gestió I, la qual s'imparteix en la titulació d'Enginyeria Tècnica en Informàtica de Gestió de la Universitat Jaume I. El llibre pretén ser, en primer lloc, un document de suport a l'estudi dels alumnes que es complementa amb els textos de referència. En segon lloc, proporciona diferents exemples d'aplicació de les tècniques que es descriuen en cadascun del temes, per a la qual cosa es proposa un cas pràctic, sobre el qual es va desenvolupant cadascun dels passos de la teoria descrita en el capítol corresponent. Tot en conjunt, permet tenir un exemple complet d'un projecte d'enginyeria de programari.

L'obra comença, al capítol 1, introduint els conceptes de procés i producte a l'àmbit de l'Enginyeria del programari. El procés està suportat per les diferents fases, activitats i tècniques que es descriuen al llarg dels capítols. Per analitzar i entendre millor quin és el producte a proveir com a solució dels problemes que es plantegen a l'àmbit de l'Enginyeria del programari, es defineix el concepte de sistema informàtic, i s'aprofundeix en els seus components i les característiques que ha de complir per a ser correcte i de qualitat.

Des del capítol 2 fins al 6 s'aborda el conjunt de fases o activitats que es poden considerar en el desenvolupament d'un projecte *software* o sistema, com es pot veure en la figura 1.

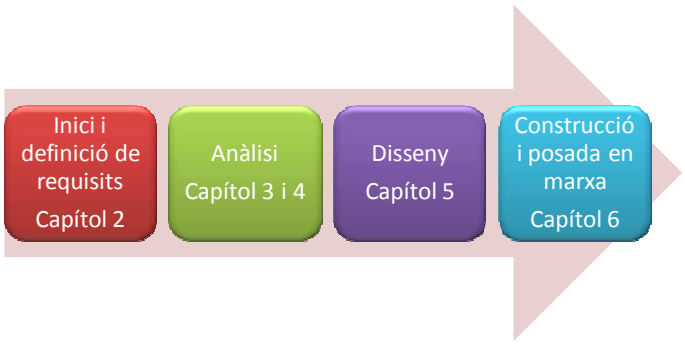


Figura 1. Organització del capítols i processos d'un projecte d'enginyeria del programari

Així, al capítol 2 es descriuen les situacions que originen la necessitat i el plantejament d'iniciar un projecte d'enginyeria del programari, i aquelles activitats que s'han de dur a terme per establir el marc de desenvolupament d'aquest projecte. Els capítols 3 i 4 es corresponen a l'activitat de l'anàlisi que es centra en la modelització del sistema, per a la qual cosa s'introdueixen tècniques de modelització de processos i de modelització de dades.

Al capítol 5 s'aborda la fase de disseny del sistema donant més contingut al disseny d'interfícies i el disseny de mòduls del sistema. Altres matèries i assignatures relacionades amb les bases de dades o

entorns d'usuari complementen els conceptes d'aquest capítol. Com a continuació de les activitats del desenvolupament de sistemes informàtics, es pretén donar una visió general de construcció i posada en marxa del sistema al capítol 6. No es tracta d'aprofundir en tot el conjunt de les tasques, ja que en alguns casos aquestes ja són objecte d'estudi en altres matèries de la titulació. Així l'objectiu d'aquest capítol és fer comprendre la necessitat de dur a terme activitats que asseguren la qualitat del programari desenvolupat i en faciliten l'ús.

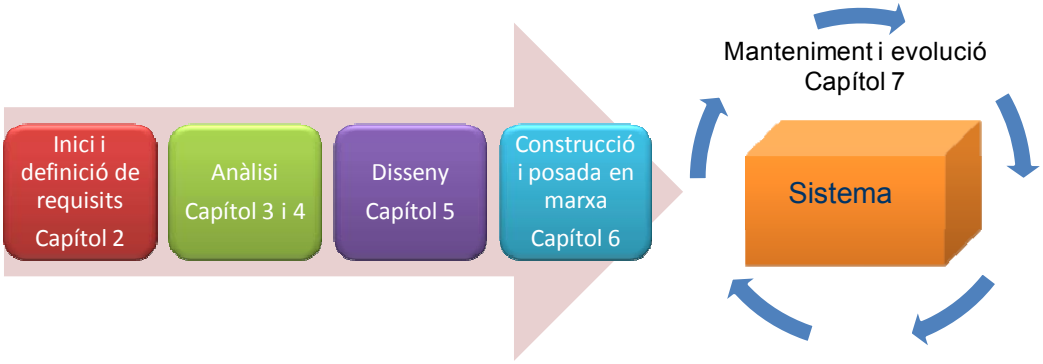


Figura 2. Organització del capítols amb el cicle de vida d'un producte de programari

Finalment, s'emfatitza el concepte de cicle de vida del programari, des d'un major nivell d'abstracció i reforçant una visió global i evolutiva del projecte d'enginyeria informàtica (de gestió). Considerant aquesta visió, s'inclou el capítol 7 dedicat al manteniment del programari, com una fase ineludible del cicle de vida posterior a la seua construcció i posada en marxa, tal com es mostra en la figura 2.

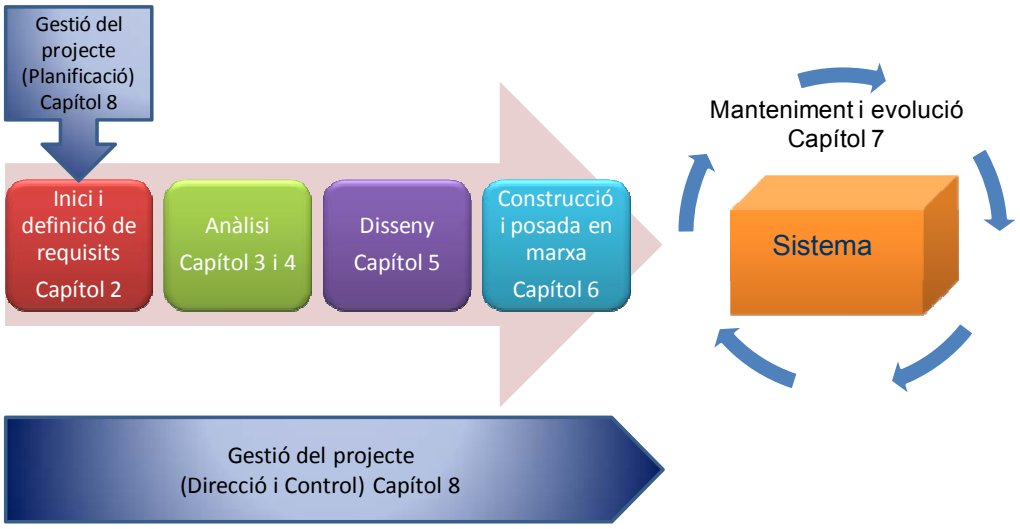


Figura 3. Organització dels capítols

Al capítol 8 es detallen les activitats de la gestió de projectes. Aquestes activitats inclouen la planificació del projecte, que s'ha de desenvolupar a activitat de l'inici del projecte.

La planificació, com a part de la gestió del projecte, s'inclou al capítol 8, perquè s'ha de tenir en compte que per a poder planificar cal saber quines són les activitats i tasques a dur a terme. Així, en

aquest capítol es detallen diferents aspectes i tècniques generals de gestió i planificació de projectes, que permeten entre altres aspectes estimar, planificar i controlar les activitats, temps, recursos i costos dels processos d'un projecte de desenvolupament de programari que s'han descrit en els capítols anteriors. En resum tots els capítols s'organitzen tal com es mostra a la figura 3.

Finalment, al capítol 9 es descriuen diferents propostes i alternatives del procés de desenvolupament d'un producte de programari. També s'inclou informació sobre algunes metodologies formals que donen suport al desenvolupament de sistemes informàtics i són usades en diferents àmbits.

Capítol 1

Introducció a l'Enginyeria del programari

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 concepte de sistema, sistema d'informació i sistema informàtic. Conèixer alguns dels diferents tipus de sistemes informàtics, com també els elements o components fonamentals que és necessari considerar per al correcte desenvolupament d'aquests.
- Saber perquè 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.

1.1. Introducció

El terme enginyeria del programari es va introduir per primera vegada a finals dels anys 60 en una conferència realitzada per a discutir el que es va denominar «la crisi del programari». Aquesta crisi va

ser conseqüència de la ràpida evolució del maquinari, enfront de la generació un tant anàrquica del programari. Durant els primers anys el programari es dissenyava a mida per a cada necessitat i no tenia una distribució àmplia. El mateix grup de persones que el desenvolupava era el que el programava i l'utilitzava, i si es produïa qualsevol problema, el corregia i el depurava. La mobilitat d'aquest grup de persones era baixa i les empreses confiaven en la seua disponibilitat en tot moment. La creació de programari era un procés totalment personalitzat del qual normalment no existia documentació (Pressman, 1997).

Gradualment es van introduir nous conceptes que van millorar la resposta i el funcionament dels sistemes informatitzats. A més, es va començar a construir el programari com un producte que podia ser distribuït en un mercat ampli i a diversitat d'usuaris.

L'àmplia distribució d'aplicacions, així com l'ampli nombre d'àmbits on s'estenia la utilització dels sistemes informatitzats va produir que els errors, deguts al mal funcionament del suport informàtic, es convertiren en un fet molt comú. Els professionals dedicaven més temps a mantenir antics sistemes i a solucionar errors, que a generar programari nou.

Arran d'aquests problemes, la comunitat de professionals que es dedicaven a la investigació i al desenvolupament de programari van començar a estudiar i analitzar els errors. Al mateix temps que estudiaven i proposaven tècniques, mètodes i ferramentes que els permetien treballar d'una forma més adequada, es va començar a utilitzar i a difondre el concepte d'enginyeria del programari.

1.2. Conceptes generals

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 aspectes relacionats amb el producte d'interès per l'Enginyeria del programari, i quines són les condicions que ha de complir un producte per ser correcte en aquest àmbit.

1.2.1. Enginyeria del programari

Què és l'enginyeria? Després d'analitzar diferents definicions, 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 proble-mes de

l'àmbit en què es considere aquesta enginyeria. I per a solucions reals, ha de comprendre's que siguen viables, 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. Segons aquesta idea s'han donat diferents definicions d'enginyeria del programari, entre les quals cal destacar les proposades per Fritz Bauer i per Ian Sommerville (Pressman, 2010):

> *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, 1977).*
>
> *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).*

S'han donat moltes altres definicions, però en general totes reforcen la necessitat d'una disciplina d'enginyeria per al desenvolupament del programari, és a dir, assenyalen uns conceptes comuns que permeten establir una relació entre els conceptes d'enginyeria i la construcció del programari. Aquesta relació està justificada perquè 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.

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

- **Mètodes:** indiquen «com» s'ha de construir tècnicament el programari. Abasten un ampli nombre de tasques corresponents a la planificació i desenvolupament d'un sistema informàtic. Aquests mètodes inclouen normalment una sèrie de notacions especials, gràfics i criteris per al desenvolupament d'un programari de qualitat (tècniques).
- **Ferramentes:** subministren un suport automàtic o semiautomàtic per a desenvolupar els mètodes. Les ferramentes més completes són les denominades ferramentes CASE (*Computer Aided Software Engineering*). Aquestes combinen maquinari, programari i bases de dades que contenen la informació sobre l'anàlisi, el disseny, la codificació i la prova per a crear un entorn anàleg al disseny assistit per ordinador.

- **Processos:** són la unió entre els mètodes i ferramentes, defineixen la seqüència en què s'apliquen els mètodes, els documents que s'han de produir, els controls de qualitat i les directrius per a avaluar el procés.

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

1.2.2. Sistema, sistema d'informació i sistema informàtic

Sistema és un terme utilitzat per a molts i diferents propòsits. Es pot parlar de sistema educatiu, sistema bancari, sistema de circulació, etc. Tenint en compte la definició de la RAE un sistema és:

> *Conjunt d'elements (objectes, normes, procediments, dispositius, etc.) relacionats entre si que funcionen de forma conjunta com un tot per tal de realitzar un objectiu, o una determinada tasca.*

Per exemple, el sistema de préstecs de llibres d'una biblioteca, que proporciona com a resultats: controlar els préstecs que es fan, obtenir una estadística dels llibres prestats i controlar els llibres que no han sigut tornats pels socis de la biblioteca. Per a obtenir aquest resultat, el sistema de Gestió de Préstecs està compost per diferents procediments, com són: enregistrar la informació dels nous llibres adquirits per la biblioteca, donar d'alta nous socis, enregistrar els préstecs i les devolucions, calcular els imports a cobrar, etc. A més a més, per tal de realitzar aquests processos es fa ús de diferents dispositius, participen diferents persones i els processos es realitzen segons unes normes i un determinat ordre.

Les dades i la informació que una empresa utilitza i necessita per al seu correcte funcionament s'organitzen al voltant del que es denomina sistema d'informació (*SI*). Un sistema d'informació és un conjunt d'elements o components organitzats, que actuen sobre un conjunt de dades i les recopilen, processen i subministren com a informació allí on siga necessària per a l'activitat empresarial, tal com es mostra a la figura 1.1.

El sistemes d'informació que inclouen entre els seus components dispositius de maquinari i programari (equips, xarxes comunicacions, etc.) es denominen sistemes d'informació automatitzats (*SIA)* o basats en computadora. Encara que el més habitual és que els sistemes d'informació utilitzen tecnologies informàtiques per al seu funcionament, es poden trobar alguns casos de sistemes d'informació no automatitzat.

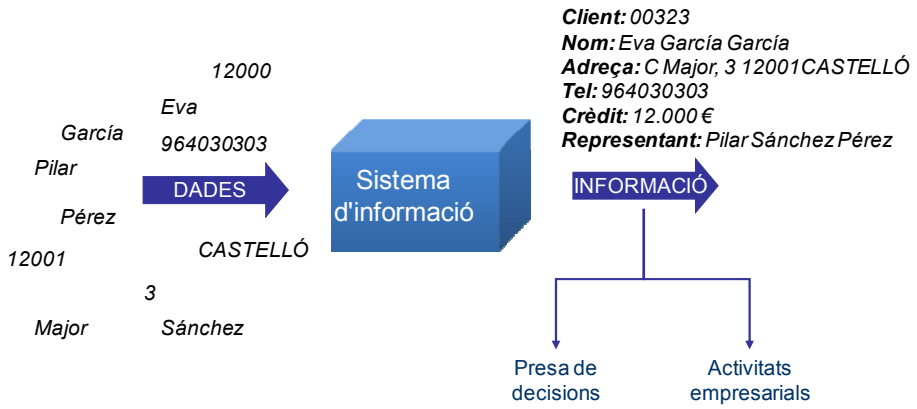


Figura 1.1 Sistemes d'informació

Quan un sistema inclou entre els seu elements maquinari i programari, es diu que és un sistema informàtic**.** Per tant els sistemes d'informació automatitzats són un tipus de sistema informàtic.

D'altra banda, hi pot haver sistemes a les empreses que inclouen maquinari i programari, i que donen suport a àrees de producció (cadena de muntatge de cotxes) o que controlen processos (forns ceràmics) i dispositius (transports filo guiats, programari encastat en electrodomèstics, etc.) que no estan relacionats directament amb el processament de dades i d'informació, és a dir són sistemes informàtics encara que no es consideren sistemes d'informació.

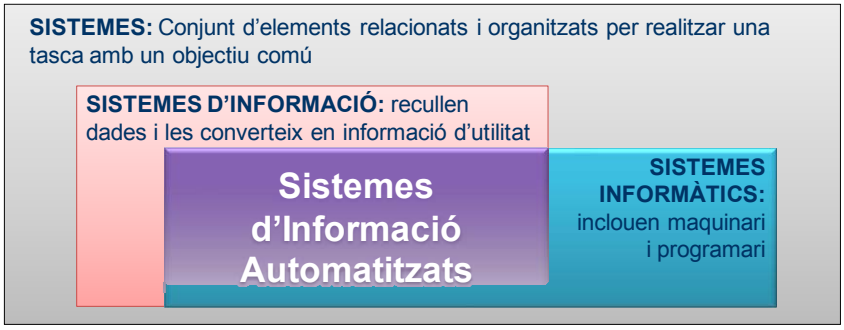


Figura 1.2. Sistemes d'informació i sistemes informàtics

A la figura 1.2 es representen els sistemes d'informació i els sistemes informàtics com a subconjunt de tots els possibles sistemes d'una empresa. La intersecció d'aquests conjunts, representada en color morat, correspon als sistemes d'informació automatitzats.

Tenint en els objectius de l'assignatura, el producte d'interès són els sistemes d'informació automatitzats i s'anomenaran de manera abreviada sistemes informàtics.

Encara que, com s'ha vist prèviament, no tots els sistemes informàtics són sistemes d'informació, alguns autors fan equivalents aquestes definicions, així Pressman defineix sistema informàtic o sistema basat en computadora com (Pressman, 2010):

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

Com es pot entendre, aquesta definició fa referència només a aquells sistemes informàtics que donen suport a sistemes d'informació, és a dir a sistemes d'informació automatitzats.

1.3. Components dels sistemes informàtics

En termes generals, tots els sistemes d'informació basats en computadora poden seguir l'esquema mostrat a la figura 1.3.

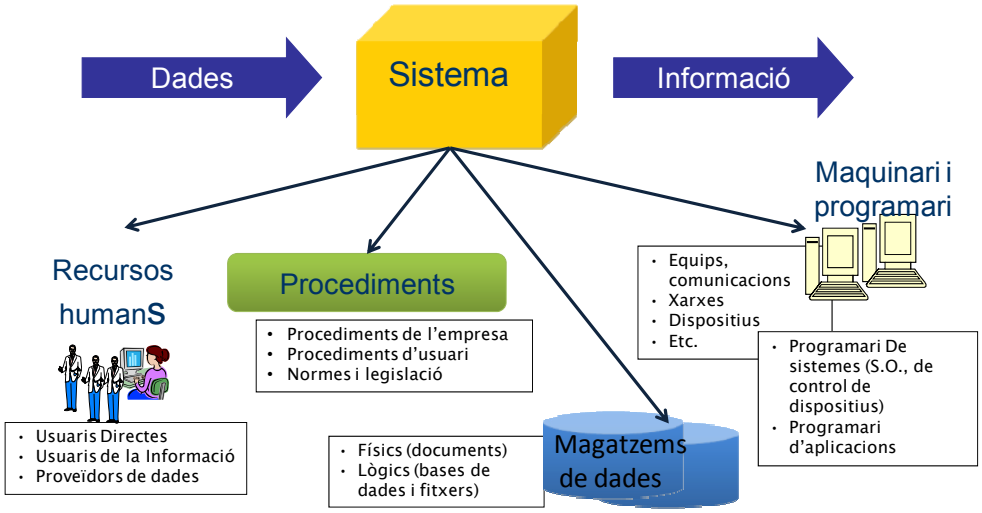


Figura 1.3. Components d'un sistema d'informació automatitzat

Les entrades dels sistemes es denominen dades i l'eixida informació. Les dades són el producte o matèria primera proporcionat per les activitats diàries de l'empresa. La informació es defineix com a les dades processades per a utilitat de les persones; representa els resultats finals que s'obtenen del sistema. L'objectiu del sistema és validar les dades d'entrada i convertir-les en informació veraç i vàlida en un període de temps apropiat.

Tenint en compte els elements que intervenen en els processos, es pot ampliar l'esquema anterior i considerar:

- **Persones:** la principal raó per a l'existència de sistemes d'informació és proporcionar informació a les persones que la requereixen. Es considera en aquest component tant les persones que exerceixen com a usuaris directes com els usuaris finals, que reben informes i resultats de forma indirecta. L'èxit o fracàs d'un sistema d'informació depèn en gran part de com estiguen de satisfets els usuaris finals, amb els resultats que el sistema els proporciona.

- **Procediments:** es descriuen normalment de forma escrita, s'indica com han de realitzar-se els processos que inclou el sistema d'informació. En el cas d'un sistema informatitzat inclouen manuals d'usuari i documents que descriuen les tasques a realitzar per cada persona involucrada en el sistema.
- **Maquinari:** aquest component consisteix en tot l'equipament utilitzat pel sistema d'informació: ordinadors, terminals, impressores. En ocasions es considera també equip no informàtic, com per exemple màquines d'escriure i subministraments de materials fungibles.
- **Programari:** el programari consisteix tant en el programari del sistema, que controla el funcionament del maquinari (sistemes operatius, programari de comunicacions, utilitats, etc.), com en el programari de l'aplicació, que consisteix en tots els programes directament relacionats amb els processos de dades del sistema d'informació que estem considerant. El programari d'aplicacions pot ser fet a mida o desenvolupat per companyies que el comercialitzen a diferents clients (fulls de càlcul, bases de dades, sistemes de nòmines, sistemes de gestió integrats, etc.).

1.4. Característiques i condicions del programari

Per tal de construir un bon programari, com a part d'un sistema informàtic, cal comprendre algunes característiques que hi són pròpies. El programari és un producte lògic, no físic, no es trenca, les fallades es produeixen per omissions o per errors inadvertits durant la fase del seu desenvolupament, no existeixen en general peces de recanvi, el producte final no és tangible per al seu destinatari, menys la part visual o física que proporcionen les interfícies. El programari es desenvolupa mentre que el maquinari es construeix i fabrica.

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 ha de preocupar únicament que realitze les tasques que l'usuari necessita. Com qualsevol altra enginyeria, l'Enginyeria del programari no consisteix únicament a desenvolupar productes, sinó que aquests s'han de crear d'acord amb unes condicions relacionades amb el cost, l'eficàcia i la qualitat.

Aquestes condicions es reflecteixen en quatre atributs que el programari de qualsevol sistema basat en computadora, desenvolupat correctament, ha de posseir (Sommerville, 2010):

- **El programari ha de ser mantenible:** els sistemes informàtics poden tenir una llarga duració 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 realitzar-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. A mesura que aquest entorn canvia o l'enginyer arriba a una millor comprensió de les necessitats del programari, aquest ha de ser fàcilment adaptable per a incorporar aquests canvis i ser cada vegada més útil a les necessitats per a les quals va ser dissenyat. En cas contrari, el programari es converteix en inútil i obsolet.

Quan el programari es desenvolupa per a ser utilitzat en diverses instal·lacions, (paquets estàndard d'alta difusió), entre altres condicions ha de ser senzill per a la incorporació de les particularitats de cadascun dels clients, de noves versions, incloure facilitats que permeten establir les característiques específiques de cada instal·lació (impressores, llocs de treball, etc.).

- **El programari ha de ser fiable:** el programari desenvolupat ha de ser segur, per això serà necessari realitzar les comprovacions i proves del programari que siguen necessàries. Un desenvolupament adequat que segueix unes pautes i metodologies correctes i que utilitza tècniques robustes que afavoreixen la construcció d'un sistema informàtic fiable.

  En alguns sistemes un error pot tenir un cost, tant en vides humanes com econòmic, que supera fins i tot el valor del mateix programari. Cada vegada s'incorporen més i més programaris en entorns on un error pot ser irreparable (avions, sistemes de seguretat de centrals tèrmiques i nuclears, etc.).

  La pèrdua d'informació, la distribució dels errors puntuals en tot el programari i la ineficàcia són alguns dels punts que s'han de tenir en compte a l'hora de dissenyar un programari. Un error en el disseny o en la construcció pot produir un mal funcionament del programari i provocar errors irreparables. 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 degut únicament a la dificultat que troba l'usuari per a la seua manipulació. Per això és important que un bon disseny tinga en compte els futurs usuaris i facilite al màxim la comunicació de l'usuari amb el programari i la utilització d'aquest últim.

Per tant, el principal objectiu que un informàtic s'ha de plantejar és el desenvolupament d'un programari útil per a les tasques que l'usuari necessita, senzill de manipular i de construir i flexible per a possibles incorporacions de noves necessitats.

El problema amb el qual ha d'enfrontar-se un enginyer que ha de desenvolupar un sistema informàtic és arribar a trobar un equilibri entre tots aquests atributs. Alguns d'aquests s'exclouen entre si, per exemple l'eficiència del programari pot veure's afectada per tenir una bona interfície amb l'usuari. A més, cal tenir en compte la relació entre el cost i les millores que proporciona cadascun d'aquests atributs. Depèn de les característiques del programari que un dels atributs es considere primordial davant dels altres. Per exemple, l'eficiència i fiabilitat, en el cas dels ordinadors d'avions i equips espacials són les primeres consideracions que cal tenir en compte, ja que aquest programari ha d'ocupar poc d'espai i utilitzar pocs recursos i per descomptat no provocar errors. En sistemes dedicats a usuaris amb poca experiència o sense coneixements informàtics, una interfície senzilla és primordial.

A més dels coneixements informàtics, un bon enginyer ha de ser capaç de comunicar-se tant de forma oral com escrita i conèixer els termes en què un possible interlocutor pot expressar-se. L'experiència, la capacitat d'assimilar conceptes aliens a la programació, la comunicació amb els usuaris, entre altres, són aspectes fonamentals que cal adquirir i que difícilment s'aprenen únicament en els llibres o a l'aula.

Com a reflexió final, cal destacar un aspecte que diferencia el programari com a producte de la resta de productes que es construeixen en altres àrees o disciplines de l'enginyeria. Els edificis, ponts o màquines són elements imaginables per les persones a les quals estan destinats. El producte al qual estan dirigits els principis de l'Enginyeria del programari, encara que cada vegada està més introduït en totes les àrees de la vida quotidiana, no és una cosa senzilla d'imaginar per als destinataris del seu ús. El programari és un objecte lògic que només per si mateix no pot funcionar, necessita recolzar-se en un suport físic.

1.5. Tipus de sistemes informàtics

Els tipus de sistemes informàtics poden classificar-se de diferents maneres segons l'ús a què es destinen, el nivell de l'empresa al qual donen suport, les tecnologies de les quals fan ús, la integració amb altres sistemes externs a l'empresa, etc. Tenint en compte els diferents nivells d'actuació i gestió d'una empresa mostrats a la figura 1.4 es pot fer la següent classificació de sistemes informàtics:

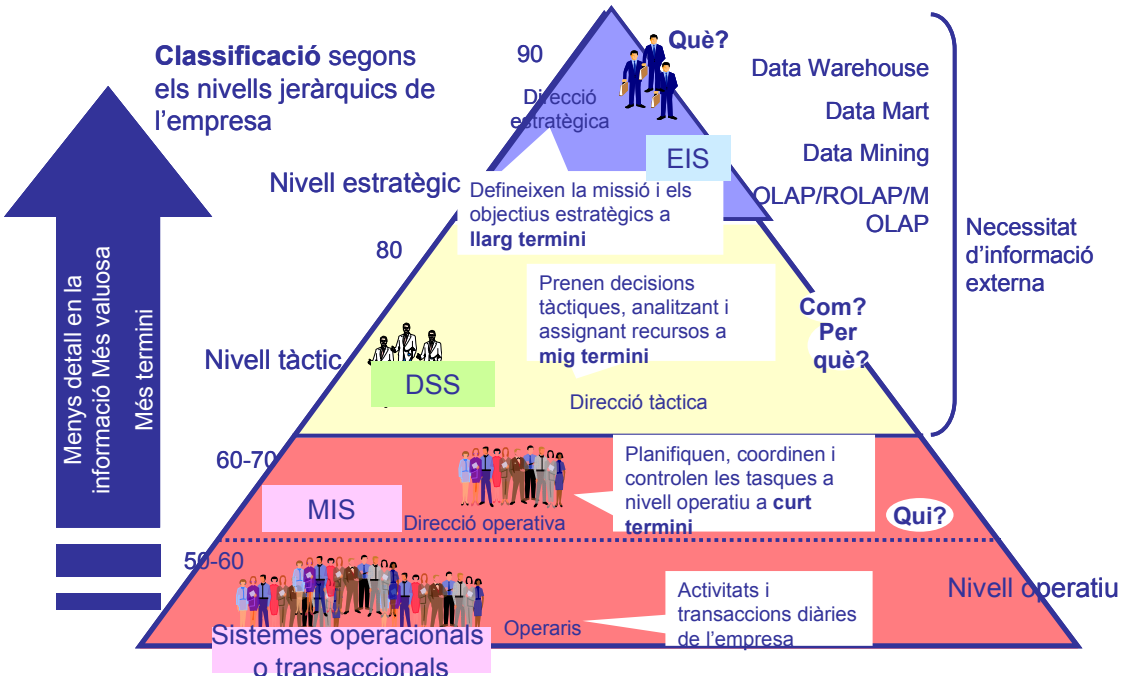


Figura 1.4. Piràmide organitzativa de l'empresa

- **Sistemes operacionals o transaccionals:** es designen així els sistemes que permeten desenvolupar les tasques generades per les transaccions diàries d'una empresa: sistemes de comptabilitat, de facturació, de control d'inventari, de compres, de vendes, de producció, etc. Inicialment, l'objectiu d'informatitzar les empreses era automatitzar procediments manuals tediosos i repetitius.
- **Sistemes per a la gerència o per a la direcció** (*Management Information Systems*, MIS): a mesura que els gerents d'empreses s'adonen de la necessitat d'obtenir informació diària correcta sobre el funcionament dels seus negocis, es comencen a desenvolupar aplicacions amb altres fins. Els gerents s'adonen que els sistemes informatitzats es poden utilitzar per a alguna cosa més que per a les transaccions diàries. La facilitat de càlcul dels ordinadors per a obtenir estadístiques, anàlisi comparativa de costos, etc., permet realitzar sistemes la informació d'eixida dels quals està dirigida a alts i mitjans directius d'empresa**.**
- **Sistemes de suport a la presa de decisions** (*Decisional Support Systems*, DSS): freqüentment els gerents d'empreses necessiten informació que no es proporciona de forma general pels sistemes per a la gerència; sorgeix la necessitat d'informació que responga a qüestions com ara: què passaria si el preu de venda d'un producte s'incrementa un determinat percentatge. Per a proporcionar aquest tipus d'informació s'han desenvolupat sistemes de suport o ajuda a la presa de decisions. Un sistema de suport a la presa de decisions és un sistema que proporciona informació a partir de dades comparatives tant internes com externes a l'empresa, i que permet als directius simular situacions per tal d'avaluar possibilitats de negoci. Aquests sistemes permeten entre altres

funcionalitats, generar llistats personalitzats, obtenir, combinar i imprimir informació específica de les bases de dades, i generar estadístiques i gràfiques.

- **Sistemes informàtics integrats:** fins ara s'informatitzava independentment cada àrea o departament de l'empresa amb la utilització de diferents solucions, la qual cosa feia molt complicada la comunicació entre les diferents aplicacions informàtiques i l'ús d'informació comuna entre diferents departaments. En els anys 90 apareixen els ERP (*Enterprise Resources Planning*): aplicacions comercials que informatitzen un conjunt de departaments, o fins i tot tots els departaments de l'empresa, de forma que la informació flueix i es comparteix per tots aquests departaments.
- **Sistemes informàtics interempresarials:** actualment la tendència és informatitzar els processos entre clients i proveïdors, la qual cosa es denomina B2B, B2C (*Business to Business*, *Business to Consumer*). Les comunicacions i altres noves tecnologies permeten que clients i proveïdors puguen estar comunicats als sistemes i milloren les gestions de les empreses.

1.6. Procés de desenvolupament dels sistemes informàtics

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, com s'ha dit, proporciona procediments, tècniques i eines per a de donar suport a totes les activitats que cal portar a terme per a construir un sistema informàtic i, per tant, un programari adient, de qualitat i amb les condicions descrites en l'apartat 1.2.3.

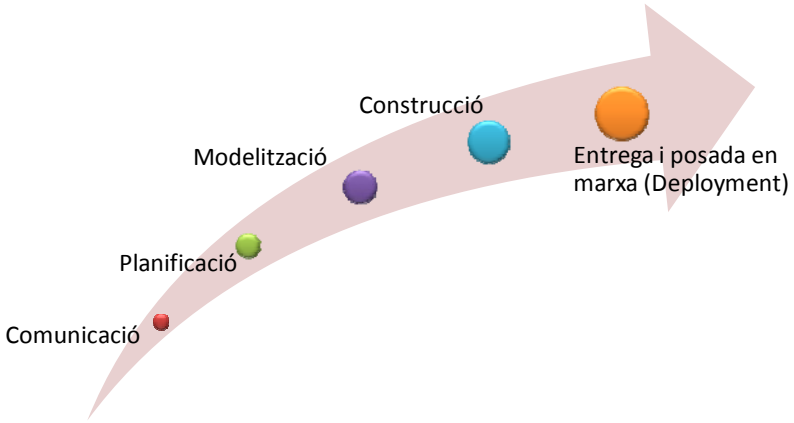


Figura 1.5. Marc de processos genèric proposat en (Pressman, 2010)

Es defineix procés de desenvolupament de programari (Pressman, 2010) com un marc on s'inclouen les activitats, accions i tasques que són necessàries per a construir un programari de qualitat. Els enginyers de programari han de tenir els coneixements i la capacitat per a saber adaptar un procés

madur de programari de manera que s'adeqüe a les característiques particulars del producte que ells van a desenvolupar.

S'han proposat diferents organitzacions del procés de desenvolupament de programari, de manera similar al procés de desenvolupament de productes d'altres enginyeries. Com a exemple, més actual, generalista i pràctic, Pressman defineix un marc de processos genèrics d'enginyeria del software que inclou les activitats: comunicació, planificació, modelització i construcció i entrega a l'usuari, que es mostren a la figura 1.5 (Pressman, 2010). A més a més defineix una sèrie d'activitats de suport com per exemple: gestió del risc, assegurament de la qualitat, gestió de configuracions. Cadascuna d'aquestes activitats en tasques, utilitza diferents tècniques per a portar a terme aquestes tasques i involucra persones de diferents nivells i perfils professionals.

Com en qualsevol producte d'enginyeria un sistema informàtic no comença a construir-se sense un estudi previ. Aquest estudi previ que inclou l'activitat de comunicació, s'inicia a partir d'activitats estratègiques que no són objecte d'aquesta assignatura, com per exemple: definició dels objectius estratègics de l'empresa, avaluació dels sistemes d'informació i necessitats del pla estratègic, planificació de sistemes a llarg termini, estudi de costos i beneficis, estudi d'alternatives, etc.

En aquest curs es pretén donar una visió generalitzada del procés de desenvolupament de sistemes informàtics, però sustentada en processos seqüencials i en mètodes estructurats. En particular es defineixen els processos inici, anàlisi, construcció i posada en marxa, i gestió del projecte, tal com es mostra en la figura 1.6.

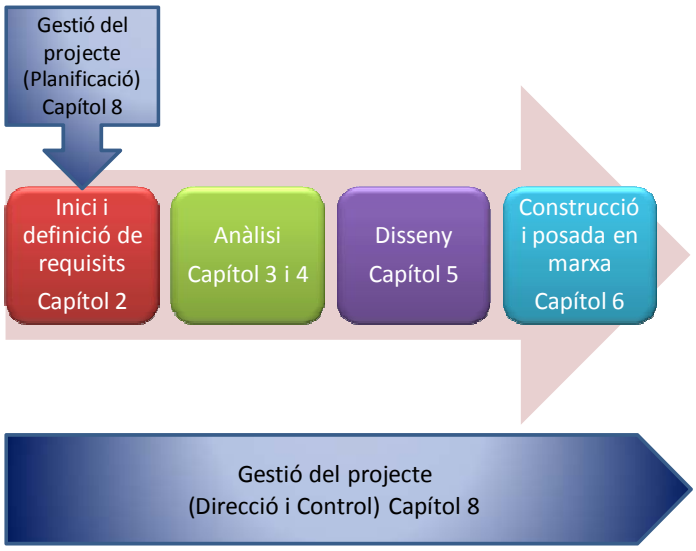


Figura 1.6. Procés de programari i organització dels capítols

Comparant els processos d'aquest procés i les activitats del marc de la figura 1.5, l'inici i definició de requisits correspondrien a l'activitat de comunicació incloent la planificació, l'anàlisi i el disseny a l'activitat de modelització, la construcció i posada en marxa a les activitats del mateix nom.

Finalment, al capítol 9 s'estudiaran altres organitzacions del procés de desenvolupament de programari. L'ús d'un procés o un altre dependrà del les característiques del producte (sistema informàtic), de l'equip i del projecte en si. És important tenir en compte que cada proposta de procés de desenvolupament de programari pot ser adequada a diferents projectes.

En els següents apartats es descriu breument cadascuna de les fases definides a la figura 1.6.

1.6.1. Inici, planificació del projecte i definició de requisits

Per a començar un projecte de desenvolupament de programari, és necessari crear una comunicació entre les persones que requereixen del producte de programari i l'equip tècnic que el desenvoluparà. A l'inici del projecte es duen a terme reunions i contactes que estableixen condicions comercials o laborals entre els clients i els desenvolupadors, i que permeten establir els objectius i abast del projecte. L'èxit del projecte ve condicionat per la comunicació i comprensió mútua que es produeix sobre el que es vol aconseguir.

Com en qualsevol producte d'enginyeria, abans de començar la construcció del sistema informàtic és necessari estimar i planificar les tasques que s'han de desenvolupar. L'estimació permet considerar les necessitats de recursos econòmics, tecnològics i humans per desenvolupar el sistema. Una vegada estimades les necessitats, l'enginyer planifica en el temps cadascuna de les tasques segons les necessitats estimades, restriccions temporals o de qualsevol altre tipus i la disponibilitat de recursos. La planificació del projecte s'ha de dur a terme a l'inici, per tal de poder concretar costos, assignació de recursos i establir les condicions contractuals del projecte.

La definició de requisits 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 sistema informàtic, tenint en compte els recursos disponibles. Per a desenvolupar un sistema informàtic correcte, un enginyer ha de ser capaç d'identificar i definir els requisits del futur sistema, per a aconseguir els objectius amb la consideració de les restriccions existents.

Aquesta primera activitat és crítica per a aconseguir amb èxit l'objectiu final que és construir un sistema informàtic de qualitat. Punts ambigus, necessitats definides vagament o no identificades poden proporcionar un sistema final pobre.

1.6.2. 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 de complir) en unes especificacions, que permeten dissenyar el codi que s'ha de programar per tal que el sistema complisca aquests requisits. Per a ser completes, aquestes especificacions s'han de desenvolupar per mitjà de models gràfics i descripcions complementàries.

És a dir, l'anàlisi de requisits consisteix en l'especificació del sistema informàtic amb la utilització de tècniques gràfiques i de mètodes complementaris, a partir d'un raonament realitzat sobre els requisits obtinguts en l'etapa de definició, amb la detecció i solució de possibles inconsistències i conflictes.

Tal com s'ha descrit, l'anàlisi d'un sistema informàtic defineix les relacions lògiques o conceptuals entre els components del sistema, les dades d'entrada necessàries, la informació d'eixida, els processos que han de dur-se a terme per tal de satisfer els requisits, quines restriccions han de tenir-se en compte, els usuaris i les necessitats de comunicació, independentment de com s'han de realitzar físicament.

1.6.3. Disseny del sistema informàtic

El disseny comporta la visió física dels processos, com i quin format ha de tenir físicament la informació d'eixida, de quina forma es volen introduir les dades que necessiten els processos dissenyats, sobre quins equips es volen implementar aquests processos, quin tipus de xarxa de comunicacions es vol implantar, l'emmagatzematge de les dades, els formats exactes dels informes, 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* es volen satisfer aquests processos.

En definitiva, 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'implementaran els processos, els arxius i taules de la base de dades, l'estructura de maquinari i la xarxa de comunicacions, fins arribar, en alguns casos, al nivell de les especificacions d'implementació.

1.6.4. Construcció i posada en marxa del sistema informàtic

La construcció del sistema informàtic agrupa totes les tasques que s'han de dur a terme per a desenvolupar el programari, com per exemple 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. També s'avalua i estudia la xarxa de comunicacions que caldrà instal·lar, per tal de configurar-la i provar-la en la fase de construcció i de posada en marxa.

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.

La posada en marxa del sistema inclou com a activitat principal el lliurament d'aquest a l'usuari. És a dir, consisteix a fer totes aquelles tasques que permetran que l'usuari faça servir el nou sistema. Hi ha diferents mètodes per a realitzar la posada en marxa d'un nou sistema. El canvi d'un sistema en funcionament, al qual els usuaris estan habituats, a un altre sistema que coneixen únicament per la formació rebuda, es converteix en un dels punts més delicats i importants del desenvolupament de sistemes informàtics i, en general, de qualsevol tipus de producte d'enginyeria.

1.6.5. Gestió del projecte de desenvolupament d'un sistema informàtic

A mesura que es desenvolupen les activitats per tal de construir el sistema informàtic, cal tenir en compte unes altres tasques que controlen la qualitat del treball i del producte que es vol obtenir, gestionen el risc i en definitiva proporcionen seguretat pel que fa a la construcció del producte en el temps i l'ús de recursos estimats.

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 sistemes informàtics.

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 activitats i les tasques que formen part del procés per a construir aquest producte.

Activitats complementàries

1. Consulteu adreces electròniques sobre grans errors del desenvolupament de sistemes informàtics.
2. Busqueu en els llibres que es proporcionen com a bibliografia bàsica unes altres classificacions del programari, segons l'ús al qual es destina.
3. Representeu de forma gràfica les activitats que haurien de desenvolupar-se en el cas que es decidisca adquirir un producte de programari de mercat, per tal de substituir un sistema d'informació obsolet.
4. A partir dels continguts vistos en aquest tema contesteu les següents qüestions:
   - Quin és el producte (o productes) d'interès per a un enginyer informàtic?
   - Quins apartats d'aquest tema estan relacionats amb les característiques d'aquest producte i quins tenen a veure amb el procés de desenvolupament del producte?
5. Confeccioneu 10 preguntes d'examen per a aquest tema i redacteu les respostes.

Cas pràctic: Taller de reparació de vehicles

El següent cas s'utilitza per a aplicar els diferents conceptes teòrics i pràctics que s'estudien en cadascun dels temes. En aquest primer tema només s'utilitza com a exemple dels conceptes relacionats amb els sistemes d'informació i els sistemes informàtics.

«La Bugia de Manolo» era un taller caòtic i poc rendible, el propietari del qual, encara que era un mecànic experimentat, resultava un desastre com a organitzador del treball. Ell personalment dirigia totes les reparacions amb crits desmesurats, sense que cap operari poguera fer cap altra cosa que obeir ordres estrictes per a cada xicoteta manipulació. Els cotxes que arribaven es reparaven sense ordre definit, perquè el cap botava d'un vehicle a un altre segons l'humor o com resultaren de persuasius els clients en les freqüents queixes pels retards.

A l'hora de cobrar les reparacions, Manolo decidia el preu de cap, sobre la marxa. Com que era molt barat, part de la clientela seguia acudint tot i el mal servei.

Quan Manolo es va morir, va heretar el negoci un fill sense cap idea de mecànica. No obstant això, es va adonar que els operaris estaven completament desorientats: acostumats a treballar sense organització, es veien envoltats de cotxes a mig reparar, sense saber quin arreglar, qui s'ocupava de cada cosa, on demanar peces de recanvi, etc.

El nou gerent els va distribuir en tres grups especialitzats: un encarregat de la xapa, un altre de l'electricitat i un altre de les avaries mecàniques. En cada grup hi havia un responsable que dirigia les reparacions i s'ocupava de procurar-se els recanvis.

A l'operari més experimentat el va fer responsable de rebre el client, fer un diagnòstic del cotxe i donar un termini de reparació. Aquest operari assignava i programava les tasques als tres grups, i tractava d'equilibrar la càrrega de treball i de complir els terminis.

SISTEMA: hi ha un procediment de recepció, diagnòstic i reparació dels cotxes.

L'eficiència d'aquest sistema va permetre augmentar el volum del negoci. Ara el treball tècnic estava més organitzat, però sorgien més problemes. Amb tant de cotxe al taller, el responsable de diagnòstic no era capaç de tenir en el cap totes les reparacions en curs i la càrrega de treball que ja havia sigut assignada, per això era difícil donar termini als clients o programar adequadament els grups, que estaven sempre desbordats i demanaven més recursos.

El gerent, per a assumir el risc de fer aquestes inversions, també necessitava tenir informació de costos de les reparacions, per a calcular el marge de benefici de les tasques, de cadascun dels grups de treballadors, etc.

Tota la informació demanada no estava disponible perquè, tot i la bona organització del treball de taller ningú escrivia cap dada, ni s'arxivava cap informació. Per això, van contractar un administratiu que s'encarregava de documentar l'historial de cada cotxe des que entrava. Aquest administratiu obria una fitxa amb les dades del vehicle i del client. En les línies de la fitxa el responsable de diagnòstic indicava, una per una, les tasques que considerava necessàries, quin grup havia de fer-les, el temps estimat, etc. Amb aquestes dades calculava un pressupost al client, després adheria al parabrisa una còpia de la fitxa i, tal com es desenvolupaven les reparacions, el responsable de cada grup escrivia les tasques realitzades, amb el temps real invertit, recanvis consumits, etc. Al final dels treballs, l'administratiu calculava la factura segons una taula de preus.

SISTEMA D'INFORMACIÓ: registre de dades segons procediments organitzats, dels quals es pot obtenir informació que serveix per a gestionar el taller.

Ja es disposava d'una gran quantitat de dades, però per a traure'n partit i obtenir informació útil per als responsables de taller i per al gerent es necessitaven molts administratius. Encara que van introduir l'ús de PC amb fulls Excel, no arribaven a aconseguir informació actualitzada, i cada vegada el cost administratiu era més alt. Llavors algú va comentar que potser feia falta un... **SISTEMA INFORMÀTIC**. Per què?

Capítol 2

Inici del projecte i definició de requisits

Objectius

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

- 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 com s'han de recopilar i interpretar els fets que es produeixen en un entorn determinat de l'organització, i com estan relacionats amb el sistema informàtic.
- 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 que la definició de requisits és una activitat crítica, de la qual depenen els bons resultats del producte final.

- Saber plasmar els requisits del sistema en una documentació útil i adequada per a les futures activitats del desenvolupament del sistema.

2.1. Introducció

Com s'ha explicat al capítol 1, l'inici d'un projecte de desenvolupament d'un sistema informàtic comença establint el contacte entre el personal tècnic encarregat de desenvolupar el producte i les persones responsables de l'adquisició i ús d'aquest. Aquest primer contacte permet identificar quines són les necessitats que han generat la demanda, els objectius del producte i establir les primeres relacions contractual si és necessari.

Figura 2.1. Com s'inicia el desenvolupament d'un sistema informàtic

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 el suport de nous sistemes informàtics o modificació dels que estan en funcionament.
- Les àrees operatives detecten necessitats que no estan cobertes pels sistemes en funcionament.
- Influència o imposicions d'agents externs, legislacions, imposicions tecnològiques de clients o proveïdors.
- 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 la revisió dels sistemes i la creació o millora d'un sistema informàtic i es decideix crear un equip de treball per tal de desenvolupar aquesta millora.

2.2. Inici del projecte

Una vegada s'ha detectat una necessitat de millora en l'organització que implica un sistema informàtic, han d'establir-se les condicions inicials, objectius, abast i restriccions. Hi ha empreses que tenen departaments específics de desenvolupament de sistemes informàtics que es fan càrrec d'aquestes activitats. En aquest cas les persones involucrades tenen un millor coneixement de l'empresa i el seu funcionament, però poden no detectar millores que des d'un punt de vista aliè a l'empresa es podrien suggerir. Unes altres organitzacions contracten empreses externes de serveis, que poden desenvolupar les millores que es detecten i aporten la seua experiència i punts de vista en altres organitzacions i sectors.

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 defineix les necessitats econòmiques, de recursos humans i tecnològics, com també la durada temporal del projecte. La planificació es 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 a desenvolupar.

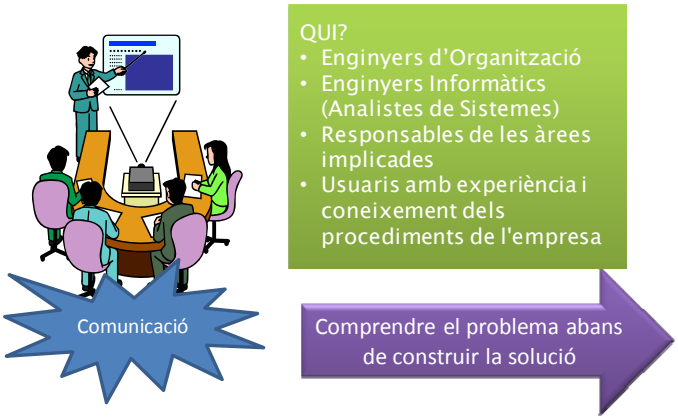


Figura 2.2. Inici del projecte: comunicació usuaris i enginyers (Pressman, 2010)

Totes aquestes activitats impliquen la realització de reunions on s'estableixen les diferents condicions que el sistema 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.

Per tant l'inici del projecte inclou les següents activitats:

- Definir objectius i abast del projecte,
- Identificar restriccions
- Avaluar alternatives
- Planificar el projecte

2.2.1. Definir objectius i abast

Com s'ha mencionat, el procés de desenvolupament d'un sistema informàtic 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. Els objectius generals del sistema han de considerar objectius operatius, tàctics i estratègics. A més a més, han de ser realistes i han de donar resposta a la pregunta sobre per què es desenvolupa el sistema informàtic.

Una vegada identificats els objectius, cal definir l'abast del futur sistema per identificar quins punts ha de cobrir, fins on arribarà, àrees involucrades, les interaccions amb altres entitats o organitzacions o amb uns altres sistemes informatitzats.

La complexitat de la determinació dels requisits augmenta quan el sistema 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 sistema.

L'abast del sistema defineix la funcionalitat que ha de complir. De vegades, quan es plantegen sistemes que afecten diferents àrees i d'una grandària determinada, és necessari establir l'abast o dividir el desenvolupament en diferents parts o subprojectes. En general, a l'hora de determinar l'abast del sistema, es pot tenir en compte tres aspectes:

- **Organitzatiu**: quines àrees o departaments interactuaran amb el sistema informàtic dins de l'organització i quines organitzacions externes (bancs, administracions públiques, clients, proveïdors, etc.) interactuaran amb el sistema.
- **Funcional**: quines funcions (d'alt nivell) cobrirà el sistema, tenint en compte els processos de negoci als quals donarà suport.
- **Informàtic**: quins sistemes actuals es substituiran o s'integraran amb el nou sistema, ja siga amb modificacions o no, de manera parcial o total.

2.2.2. 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 sistemes informàtics, normalment, són les relacionades amb pressupostos, planificació en temps i personal qualificat disponible. Poden ser imposades pels responsables de l'organització per a controlar la utilització dels recursos o bé per altres factors externs (lleis, normatives, canvis de monedes, etc.). Unes altres restriccions poden estar condicionades per factors predefinits com són, per exemple, la necessitat de mantenir la compatibilitat amb el maquinari existent o d'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- poden 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.

2.2.3. Avaluar alternatives

A l'hora de desenvolupar un sistema informàtic, els responsables de l'empresa i el del projecte poden considerar diferents alternatives, des 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.

Unes altres vegades les alternatives suggerides depenen de la grandària del projecte, dels costos i els requisits temporals d'algun dels apartats del sistema. Algunes àrees involucrades en el nou sistema tenen una prioritat major pel que fa a la data de posada en marxa, i els responsables del projecte han de decidir per on començar i quins apartats del sistema cal posar en marxa abans que altres.

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

Per tant en aquest apartat cal avaluar les necessitats, els recursos disponibles o que s'han d'adquirir i els costos que s'han d'assumir. L'enginyer informàtic ha de proposar les alternatives possibles, avaluar els costos i beneficis de cadascuna, per tal que els usuaris responsables puguen decidir amb tota la informació possible al seu abast.

2.2.4. Planificar el projecte

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, i també estimar el calendari que cal seguir i els recursos que cal assignar a cadascuna de les activitats. La planificació suposa realitzar les següents accions (Piattini, 2004):

- Fer previsions i estimar el temps que durarà cadascuna de les activitats i els costos associats.
- Definir els objectius del projecte a curt i llarg termini.
- Definir polítiques organitzatives que guien la realització del projecte.
- Determinar els processos estructurals.
- Analitzar les activitats que s'han de realitzar i la possible divisió en tasques més menudes.
- Obtenir i distribuir els recursos per a dur a terme les diferents activitats del projecte.

L'activitat *Planificar el projecte* s'inclou dintre les considerades en la gestió del projecte, que es descriuen al capítol 8.

2.3. Identificar i definir requisits

Es pot dir que un requisit és una condició que el sistema ha de complir o una característica que ha de tenir. Els requisits han de descriure de manera exhaustiva i detallada tot allò que ha d'arribar a proporcionar el sistema per tal d'assolir els objectius dintre l'abast definit, i tenint en compte les restriccions.

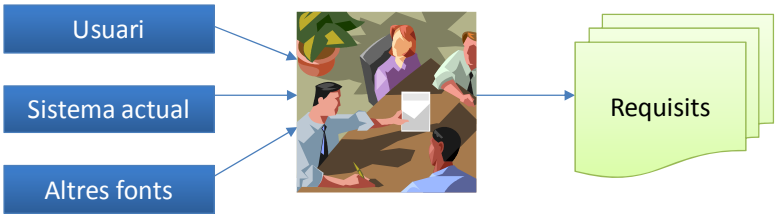


Figura 2.3 Definició de requisits

En ocasions la definició de requisits i la posterior anàlisi del sistema poden ser complexes, perquè involucren sistemes de gran envergadura, difícils de definir i, a més, subjectes a modificacions. Durant la definició de requisits és necessari treballar de forma conjunta amb els usuaris finals, és necessari comprendre i integrar tots els requisits dels diferents usuaris, que a vegades poden tenir objectius distints o que poden donar lloc a conflictes. L'enginyer informàtic, en aquests casos, ha de desenvolupar una tasca d'intermediari entre aquestes necessitats contraposades.

Per tal de desenvolupar una definició de requisits correcta i consistent, el primer pas que haurà de portar a terme un enginyer informàtic es conèixer el sistema en funcionament i quins són els aspectes que es poden millorar o quins s'han d'incloure en el nou. La tasca de sintetitzar les visions que tenen els usuaris del futur sistema i definir el sistema en funcionament, detectar problemes i proposar millores són algunes de les demandes fonamentals d'un enginyer informàtic, en la primera activitat del desenvolupament de sistemes informàtics.

En la fase de definició de requisits l'enginyer informàtic ha d'obtenir resposta a les qüestions que es mostren en la taula 2.1:

QUI?	QUÈ?	ON?	QUAN?	COM?
• Qui porta a terme cadascun dels procediments relacionats amb el sistema? • Pot realitzar una altra persona aquestes tasques? • Qui hauria de fer-ho?	• Què és el que s'està fent actualment? Quins processos es porten a terme i en quin ordre? • Per què es porten a terme aquests procediments? Què hauria de fer-se? • S'hauria de modificar o eliminar?	• On es porten a terme cada un dels processos identificats ? • Per què es porten a terme en eixos llocs? • On podrien desenvolupar-se? • On hauria de fer-se?	• Quan es realitza un determinat procés? • Per què? • És el millor moment per a desenvolupar-lo? • Quan hauria de fer-se?	• Com es porta a terme un procediment? • Podria fer-se millor, amb menor cost, o d'una forma més eficaç si es modifica? • Com hauria de fer-se ?

Taula 2.1. Qüestions per a extraure requisits

La resposta a aquestes preguntes proporciona la informació per tal de comprendre el sistema en funcionament i quines són les necessitats del futur sistema. Al mateix temps es detecten quins són aquells aspectes del funcionament actual que o bé no funcionen de manera adequada, o bé ni tan sols haurien de fer-se.

Comprendre completament els requisits del sistema és vital per a desenvolupar un sistema d'alta qualitat. Aquesta primera fase és la base sobre la qual s'ha de construir el sistema; punts ambigus, necessitats definides vagament, o que poden canviar amb facilitat, proporcionen un sistema final pobre.

A continuació es descriuen les principals activitats o tasques de la identificació i definició de requisits que són: revisar els sistema en funcionament i investigar els requisits del nou sistema. A l'apartat 2.6. es descriu com documentar els requisits obtinguts en aquesta activitat.

2.3.1. Revisar el sistema en funcionament

Un dels passos fonamentals en l'activitat de definició de requisits és l'estudi del sistema en funcionament, és a dir, què es fa. Per a fer aquesta activitat cal que existisca un sistema en funcionament comparable amb el sistema que es vol implantar, bé en la mateixa empresa o organització o en una organització semblant. El sistema en funcionament pot ser un sistema poc eficient, o pot tenir problemes, o una funcionalitat pobra respecte del sistema proposat, o simplement pot ser un sistema no informatitzat.

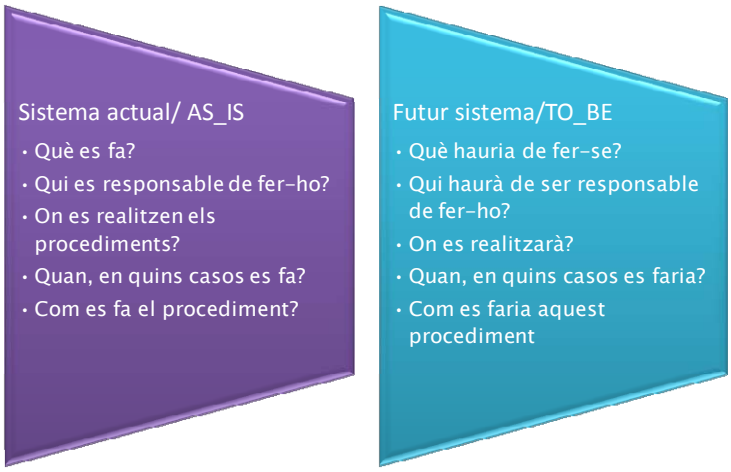


Figura 2.4 Revisió del sistema actual

El temps que és necessari invertir en l'estudi del sistema existent depèn de la grandària, l'abast i les àrees funcionals involucrades. És convenient establir un termini aproximat i utilitzar únicament el temps suficient per a comprendre els requisits principals.

Una vegada avaluat el sistema en funcionament s'han de documentar i determinar quins són els punts forts que el sistema proporciona respecte al seu funcionament, i que per tant s'han de mantenir o incloure en el futur sistema, i quins són els punts febles, per a esmenar-los i substituir-los. Per donar una informació més completa es poden fer diagnòstics i propostes de millora. A més de l'informe corresponent,on s'han d'identificar aquests punts forts i febles i l'avaluació del sistema, es pot completar l'estudi mitjançant models de processos on es representa el funcionament del sistema, independentment de la seua implementació.

2.3.2. Investigar els requisits del sistema

La definició de requisits es desenvolupa en tres etapes fonamentals:

1. **Extracció:** durant aquesta primera etapa s'utilitzen tècniques apropiades de recerca, per a extraure la informació que permetrà determinar els requisits del sistema proposat.

2. **Anàlisi i Documentació:** en la segona etapa s'analitzen els fets i la informació identificats i s'organitzen i documenten.
3. **Validació:** finalment se'n validen els requisits. En aquesta etapa es presenten els resultats als responsables de prendre les decisions sobre el sistema a desenvolupar, per tal de comprovar la seua correcció i consistència.

L'enginyer informàtic ha d'identificar quins són els usuaris involucrats, ha d'investigar què volen els usuaris que realitze el futur sistema, documentar les seues necessitats i obtenir la seua aprovació sobre la documentació generada.

Per a validar si un requisit és necessari i correcte s'ha de tenir en compte si està dintre de l'abast definit, no crea conflictes amb les restriccions i, el més important, si aporta valor al sistema (Pressman, 2010).

Exemples de requisits:

- El sistema permet crear, modificar i esborrar informació dels treballadors.
- 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 treballador.

Els requisits del futur sistema han de donar resposta a les qüestions plantejades en la Taula 2.1 i per tal de ser complerts han de:

- Descriure les condicions de l'entorn i els diferents tipus de maquinari necessaris.
- Enunciar les funcions que el sistema ha de realitzar de forma general.
- Descriure les entrades i eixides i determinar les dades que s'han d'emmagatzemar.
- Incloure altres requisits com per exemple temps de resposta, seguretat d'accés a les dades, verificació, etc.

Per tal d'investigar els requisits i completar tota aquesta informació es realitzen diferents tasques i es pot fer ús de diferents tècniques, algunes de les quals es descriuen a l'apartat 2.4 d'aquest tema. Els resultats s'han d'analitzar i documentar de manera adequada, com s'indica a l'apartat 2.5. Documentació de requisits.

2.3.3. Documentar els requisits del sistema

Determinar i definir tots els requisits d'usuari és essencial, perquè el document on s'arrepleguen és la base per al desenvolupament posterior del sistema. A més a més, el document corresponent als requisits pot servir com a base per a comparar els resultats obtinguts i determinar la seua acceptació.

La definició de requisits pot fer-se utilitzant el llenguatge natural per a definir les tasques que ha de realitzar que el sistema. Un exemple d'un requisit pot ser: «El sistema ha de calcular les notes mitjanes dels alumnes d'un curs per cada assignatura, i imprimir un diagrama de barres per alumnes i notes».

Per a especificar els requisits és necessari el treball conjunt dels clients/usuaris i de l'enginyer informàtic. A partir de les primeres nocions del problema i de tot el treball d'investigació realitzat, per a arribar a aquesta tasca d'especificació l'enginyer informàtic i el futur usuari del sistema han d'introduir-se en els detalls que defineixen les necessitats que el futur sistema ha de complir.

En l'apartat 2.6. es descriu com es poden documentar els requisits investigats i analitzats. Durant totes les activitats prèvies l'enginyer ha anat documentant els resultats, finalment tota la informació recopilada s'haurà d'organitzar per tal de proporcionar un resultat adequat.

2.4. Tècniques per a investigar i definir requisits

Des d'un punt de vista molt simple podem dir que per a determinar els requisits, un enginyer informàtic ha de preguntar els usuaris què volen que realitze el sistema, documentar les seues necessitats i obtenir l'aprovació del document on es redacten aquestes necessitats. Però realment no és una cosa tan simple, hi ha problemes generals, que poden trobar-se en la definició de requisits i l'anàlisi de qualsevol sistema, i altres específics, que és necessari resoldre a partir de l'experiència i coneixements de l'enginyer informàtic. Utilitzar una estratègia o una altra per a determinar els requisits depèn de la informació i de la seua disponibilitat. En aquest apartat es descriuen algunes de les aplicades, però el més adequat és utilitzar diverses d'aquestes estratègies, considerant l'entorn, la disponibilitat dels usuaris i les característiques del sistema.

2.4.1. Entrevistes

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 avantatges el contacte personal amb els futurs usuaris, 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 considerar alguns desavantatges com són que consumeixen molt més temps que uns altres mètodes, l'avaluació de la informació obtinguda pot ser difícil, a vegades no es pot quantificar -a diferència dels qüestionaris- la informació que proporcionen els entrevistats pot no ser imparcial o tenir una visió personal que s'allunya dels objectius del sistema i l'entrevistador pot no detectar la informació poc fiable.

A continuació es mostren els passos fonamentals que s'han de tenir en compte per a fer entrevistes adequades i productives:

- **Identificar fonts d'informació i persones a entrevistar:** les entrevistes comporten introduir-se en l'organització de l'empresa i comunicar-se amb persones que desenvolupen unes determinades tasques dins de l'empresa. Per això és important inicialment informar-se sobre les diferents responsabilitats, les relacions existents i l'organització en departaments o àrees relacionades en principi amb el sistema. En la majoria de les empreses hi ha esquemes o gràfics on es representa l'estructura de la seua organització; si no és així és necessari, a partir de la persona amb què s'ha establert el primer contacte, obtenir l'esmentada informació. És a dir, el primer pas per al desenvolupament de les entrevistes consisteix a conèixer com funciona l'empresa i quines són les responsabilitats i deures de cadascuna de les persones involucrades. Per realitzar les entrevistes amb èxit el suport que la direcció de l'empresa o de les persones que exerceixen la presa de decisions és de vital importància. Normalment es necessita entrevistar dos tipus de futurs usuaris: els directors o responsables de les àrees funcionals implicades i un representant o un grup representatiu dels usuaris operacionals finals. És convenient centrar-se en una de les àrees funcionals i no començar les entrevistes d'altres àrees fins haver finalitzat amb la primera, i també començar amb els usuaris que utilitzaran el sistema dia a dia i posteriorment resumir els requisits i revisar-los amb el responsable de l'àrea funcional.
- **Identificar les activitats relacionades amb cadascuna de les persones:** una vegada identificades les persones i les seues responsabilitats, és necessari revisar les tasques que porten a terme i com les realitzen. Per a això es pot analitzar la documentació existent (formularis, informes), el sistema en funcionament, manuals de política de l'empresa, manuals de mètodes i procediments existents, etc. Com ja s'ha comentat, a l'inici del desenvolupament del projecte normalment es realitzen reunions o entrevistes on participen els enginyers informàtics (responsables del projecte) i els responsables o gerents de l'organització amb responsabilitat sobre l'àrea que afecta el sistema

informàtic. A partir d'aquestes reunions inicials s'obté, d'una banda, l'aprovació de la direcció per al desenvolupament del sistema i, d'una altra, la identificació de les persones que estaran directament relacionades amb el sistema a nivell operacional. En qualsevol cas, és important identificar prèviament fonts d'informació que permeten enfocar i centrar l'entrevista i planificar-la.

- **Preparar les entrevistes:** en aquest punt una de les tasques fonamentals és establir l'hora i el lloc on s'ha de realitzar. Al mateix temps que es concreta l'entrevista és convenient informar l'entrevistat dels objectius i temes a tractar perquè aquest prepare, si és necessari, documentació o revise punts que puguen ser interessants. Per a realitzar les entrevistes als diferents usuaris, a més de tenir un coneixement previ de quin és l'entorn i la informació que s'ha d'utilitzar, és convenient preparar una sèrie de qüestions bàsiques que puguen interessar l'enginyer informàtic i adoptar una postura oberta a suggeriments. És important que la conversa es mantinga sempre a nivell de cooperació entre ambdues parts. L'enginyer ha de preparar un guió de les entrevistes, tenint en compte els diferents tipus d'usuaris que tindran relació amb el futur sistema.
- **Realitzar l'entrevista:** és aconsellable realitzar l'entrevista en el mateix lloc de treball de l'entrevistat, perquè ell se sent més còmode en un lloc que coneix i perquè la informació necessària, documents, informes, etc., està més a l'abast. No obstant, hi ha l'inconvenient que l'usuari pot veure's interromput durant l'entrevista per telefonades o pel funcionament del seu treball habitual. Establir un horari que permeta l'usuari dedicar tota l'atenció a l'entrevista pot ser una solució. També és necessari plantejar-se a vegades entrevistar a diverses persones al mateix temps però, en principi és més convenient realitzar les entrevistes d'un en un i després contrastar la informació. En cas d'inconsistències o diferents punts de vista les reunions globals poden ajudar a aclarir idees i unir objectius però cal evitar enfrontar els usuaris amb idees diferents.
- **Resumir i fer actes de les entrevistes: u**na vegada finalitzada l'entrevista és convenient començar immediatament el resum i l'obtenció de conclusions. Durant el primer període de temps la informació es manté en la ment i fàcilment es poden completar les notes preses durant les entrevistes. És convenient realitzar una acta formal que registre de manera adequada aquells fets i decisions rellevants, per a identificar les condicions del sistema a desenvolupar. Aquestes actes es poden revisar i aprovar en reunions amb els usuaris, de manera que es puga tenir la seua aprovació.

Regles per a realitzar una entrevista correcta i profitosa:

- **Saber escoltar:** l'entrevistador és qui menys ha de parlar, ha de saber escoltar i enfocar el tema.

- **Prendre notes**: durant l'entrevista no s'ha d'anotar tot allò que l'entrevistat diu, s'ha de prendre notes per a aclarir punts importants, resums, etc. Quan acabe l'entrevista, és convenient fer un resum per completar aquestes notes i incorporar tot allò que es recorde.
- **Tipus de preguntes:** és convenient començar amb preguntes obertes que permeten a l'entrevistat explicar el que ha preparat, no s'ha de realitzar preguntes que tallen el curs dels seus pensaments, ni canviar de tema bruscament o interrompre'l. Es poden fer preguntes de comprovació. Per canviar de tema és necessari indicar-ho clarament. S'ha de tenir en compte que l'usuari és qui proporciona la informació, en cap cas s'ha d'entrar en discussió ni fer comentaris que mostren la nostra visió particular. L'objectiu és obtenir informació fiable, rellevant, profitosa i el més objectiva possible.
- **Control de l'entrevista**: és convenient mantenir en tot moment el control de l'entrevista i moderar-la de forma que l'entrevistat puga parlar i s'assegure al mateix temps que el que diu és rellevant per a l'obtenció de la informació. Hi ha unes altres consideracions que cal tenir en compte, com ara presentar-se, acomiadar-se i agrair el temps dedicat, que permeten establir una relació de cooperació entre els desenvolupadors del sistema informàtic i els usuaris.

2.4.2. 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 es poden dissenyar i enfocar a problemes concrets, tenint en compte tant els nivells dels destinataris com els temes que s'han de tractar.

Les preguntes que cal incloure en un qüestionari han de dissenyar-se acuradament. És aconsellable que siguen curtes, que admeten respostes de vertader o fals, o bé que proporcionen una llista de possibilitats entre les quals es puga elegir. Aquest tipus de qüestionaris són més fàcils de formalitzar per les persones que han de respondre.

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.

Per a desenvolupar un qüestionari de forma que el resultat de l'enquesta siga el més adequat i útil possible, és necessari preparar les qüestions de forma que no es produïsquen ambigüitats, decidir quin tipus d'informació es vol recopilar mitjançant els qüestionaris i quina quantitat de persones han de contestar.

És convenient organitzar les preguntes en un ordre adequat, per temes i importància, elegir el format de les respostes, que pot ser una llista de possibilitats a elegir, o donar una valoració o un ordre d'importància de les respostes, o simplement vertader o fals. No cal incloure preguntes innecessàries o que puguen confondre. De vegades és convenient realitzar un esborrany i comprovar els resultats amb un grup xicotet de persones per a identificar inconsistències, preguntes confuses o falta d'informació.

Ja que les respostes han de ser avaluades és necessari verificar que el qüestionari pot quantificar-se de forma adequada. En el cas de realitzar qüestionaris per a ser contestats de forma multitudinària, la lectura es pot realitzar de forma automàtica. En aquest cas és necessari dissenyar adequadament el full de respostes i comprovar-lo abans de la seua edició i distribució.

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.

2.4.3. 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 entrevistes o amb anterioritat i cal fer una revisió del seu ús i validesa com també de la seua distribució.

En la taula 2.2 es proporciona una llista de documents, que es poden recollir i analitzar per tal d'obtenir informació sobre el sistema en funcionament.

Qualsevol tipus de sistema	Sistema Informatitzat
• Documentació de procediments • Manuals d'instruccions tècniques • Manuals de formació • Exemples d'informes • Exemples de documents i formularis • Descripció de llocs de treball o perfils • Documents d'organització	• Documentació de la definició de requisits • Diagrames de context • Formats d'entrades i d'eixides • Manuals d'usuari • Diagrames de flux de dades • Documents d'organització • Models de dades

Taula 2.2. Exemples de documents de sistemes d'informació i sistema informatitzat

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

2.4.4. 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 pot ser útil per identificar algunes necessitats d'un sistema informàtic. 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.

2.4.5. Utilitzar fonts externes de documentació

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 unes altres empreses del sector, o associacions relacionades o empreses consultores, que aconsellen i recomanen sobre certs aspectes estratègics i operacionals del futur sistema.

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.

2.4.6. Modelització de processos

Els diagrames de processos són una tècnica de modelització que és útil per a descriure la situació actual, informatitzada o no, i faciliten que l'enginyer informàtic i l'usuari comprenguen el sistema actual i com afectarà el nou sistema als processos que l'empresa desenvolupa per dur a terme els seus objectius.

Un exemple d'aquest tipus de models són els diagrames IDEF0, que permeten representar processos de producció industrials, d'organització o de qualsevol àmbit d'una empresa, (IDEF0, 2009), com l'exemple que es mostra en la figura 2.5.

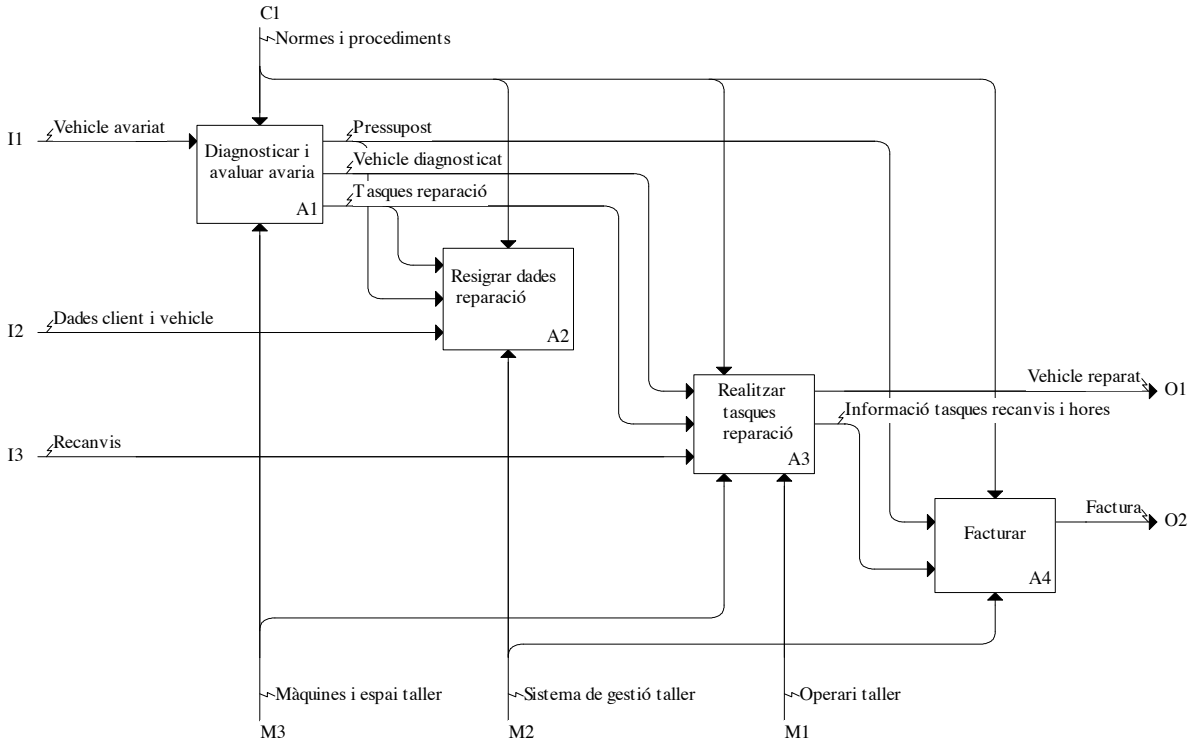


Figura 2.5. Exemple de diagrama IDEF0 per al procés del cas del Taller: *Reparar vehicles i facturar*

2.5. Documentació dels requisits

Com en qualsevol document professional sempre s'inclourà un títol, un índex i una introducció. Existeixen diferents propostes per organitzar un document de definició de requisits i propostes, per utilitzar plantilles i formats més estrictes que ajuden a descriure els requisits d'una manera completa i consistent. Un exemple d'una metodologia completa es pot trobar a *Metodología para la Elicitación de Requisitos de Sistemas Software* (Duran, 2002).

2.5.1. Condicions de la documentació de requisits

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 sistema.

- Ha d'incloure possibles restriccions en el funcionament del sistema i que és necessari tenir en compte per al seu disseny.
- 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 dels usuaris i responsables de les àrees implicades.
- Ha d'incloure una taula de continguts, un 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 realitzar 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.

2.5.2. Proposta per organitzar el document de definició de requisits

En aquest apartat es proposa una organització que pot modificar-se segons les característiques particulars de cada projecte o de cada sistema. La idea és documentar de manera organitzada tota la informació que s'ha obtingut en les activitats que es duen a terme, per tal d'investigar els requisits i redactar un document formal que servisca de guia en totes les fases del desenvolupament del sistema. Es proposen els següents apartats:

- **Índex**, en tota la documentació que acompanya qualsevol projecte o treball, és necessari incloure un índex que facilite la lectura i la possible recerca d'un tema determinat (índex per matèries, alfabètic, etc.).
- **Introducció**, on es descriu per què existeix la necessitat del sistema i els principals punts que aquest ha de cobrir, detalls sobre el context on el futur sistema estarà ubicat com també els possibles aspectes en els quals l'organització es veurà afectada (possibles millores o canvis necessaris).
- **Descripció** organitzada de l'àrea o àrees involucrades, on es descriuen les persones i departaments relacionats amb el sistema i les seues responsabilitats.
- **Objectius**, detallant quines són les metes i beneficis tàctics i estratègics que es volen assolir amb el sistema o millora a desenvolupar.
- **Estudi del sistema en funcionament**, que pot incloure:

- Identificació de problemes i propostes de millores.
- Models del sistema.
- Procediments.
- Llistat de documents actuals (ús, adequació, etc.).

- **Requisits** i necessitats del futur sistema:
  - **Requisits funcional**s: les funcions que aquest sistema ha de proporcionar a l'usuari.
  - **Requisits de dades:** necessitats d'informació, i les possibles relacions o estructures existents des d'un punt de vista de l'usuari.
  - **Requisits no funcionals**: restriccions sota les quals el programari ha de funcionar, condicions de seguretat de la informació i de nivells d'accés necessaris segons els perfils d'usuaris, i requisits de control i temporals.
  - **Requisits de maquinari**: descripció de la plataforma de maquinari necessària per al funcionament del sistema, connexions entre el sistema i altres futurs o en funcionament, requisits dels dispositius d'entrada i d'eixida, capacitats del maquinari, etc.
- **Glossari**[1], on s'identifiquen els termes tècnics específics utilitzats per a la definició de requisits. Aquests termes són útils tant per a la comprensió dels usuaris del document com per als futurs programadors en el cas de termes específics de l'empresa. Per a redactar aquest glossari l'autor ha d'assumir que el possible lector de la definició dels requisits no té cap coneixement sobre el tema.

Per a documentar cadascun dels requisits es proposa seguir un mètode semiformal en el qual es creen plantilles adients, per tal d'aconseguir homogeneïtat en la definició de cada requisit i tenir documentats aspectes formals que poden ser útils durant el desenvolupament del projecte, adaptades de les proposades en *La Metodología para la elicitación de requisitos* (Duran, 2002).

Resum

En aquest tema es proposen algunes de les tasques fonamentals que cal realitzar per tal de desenvolupar una correcta definició de requisits. Es dóna especial rellevància al fet d'aconseguir informació útil i veraç, per tal d'identificar les necessitats del futur sistema informàtic. S'estudien diferents tècniques d'ajuda a la recopilació i extracció de requisits, i d'avaluació del sistema en funcionament. Finalment es proposa una organització del document de definició de requisits.

[1] Els glossaris són una eina útil i senzilla per obtenir informació del domini del problema i compartir-la amb tots els participants en el projecte.

Activitats complementàries

1. Busqueu al diccionari els següents conceptes: risc, prototip, glossari, requisit.
2. Considerant l'entorn d'un gran centre comercial, feu la llista de requisits que hauria de complir el subsistema informàtic que funciona en una de les caixes enregistradores de les eixides que permeten cobrar als clients. Identifiqueu objectius, restriccions, entrades, eixides, i comunicació amb altres suposats sistemes.
3. Recolliu d'alguna empresa que conegueu exemples de documents que utilitzen en els seus sistemes d'informació (automatitzats o no). Per exemple albarans, factures, tiquets de compra o de venda, etc. Analitzeu cadascuna de les dades que hi apareixen, l'aspecte del document, etc. Feu per a cadascun dels documents una llista de totes les dades que apareixen i indiqueu què representen, si les ompli alguna persona de l'empresa o algú extern a aquesta.
4. Feu un document de definició de requisits amb el format de les plantilles de definició de requisits proposades per al cas del *Taller de Reparacions de Vehicles*.
5. Confeccioneu 10 preguntes d'examen per a aquest tema.

Cas pràctic: Taller de reparació de vehicles

Objectius

Els objectius generals del futur sistema de gestió de reparacions poden classificar-se com a objectius operatius i estratègics.

Per una part els objectius operatius són gestionar de manera més àgil i efectiva tota la informació que es genera en el taller sobre les reparacions i millorar d'aquesta manera l'organització del treball i el servei a clients.

Els objectius estratègics i tàctics se centren en una reestructuració de la política d'assignació de costos i recursos humans i materials. Per a la qual cosa es necessari obtenir informació sobre resultats del taller a més llarg termini, estadístiques de reparacions i temps de finalització de reparacions, rendiments del equips de treball, etc.

Abast organitzatiu i informàtic

Aquest sistema cobrirà la funcionalitat de l'àrea de reparacions i de manteniment de documentació i informació de clients i tindrà interaccions amb les àrees de compres, magatzem, recursos humans i comptabilitat.

Es pretén que el nou sistema es comunique amb el mòdul estàndard de gestió de compres que s'acaba d'adquirir per a la gestió de les compres de recanvis i del magatzem, de manera que:

Els consums de recanvis registrats en les reparacions han de donar lloc a l'actualització automàtica de l'estoc.

El sistema ha de permetre consultar informació del mòdul de compres sobre els recanvis (existències en magatzem).

El sistema també haurà d'accedir la informació de treballadors que es gestiona amb el subsistema de nòmines, de manera que quan es registre un codi de treballador en la fitxa de reparació s'obtinga el cost imputable per hora que correspon a la seua categoria professional.

Des del punt de vista de relacions externes amb altres organitzacions, s'haurà de tenir en compte l'ampliació prevista per tal que els clients puguen realitzar consultes via Internet sobre les reparacions dels seus vehicles.

Abast funcional

Gestió de reparacions del taller per a millorar la informació que es té sobre clients, vehicles i reparacions.

Donar suport al treball de facturació i la gestió del magatzem controlant els recanvis utilitzats en les reparacions.

Proporcionar funcions de recerca i consulta sobre les reparacions, clients, facturació, etc., i en general sobre qualsevol tipus d'informació enregistrada en el sistema.

Obtenir estadístiques comparatives per període, tipus de reparació, clients, etc.

Generar informes i consultes configurades segons la necessitat del responsable de l'àrea o del taller.

Proporcionar un control de les reparacions finalitzades i facilitar el treball de facturació.

Descripció de subàrees involucrades

Àrea de recepció de clients (exemple de descripció del procediment)

Hi ha una persona responsable de rebre al client en el moment d'accedir al taller. Aquesta persona comprovarà si el client ja ha estat al taller i/o si el vehicle ha estat reparat abans o no. En cas que el client o el vehicle siga nou pren nota de les dades necessàries i pregunta sobre quina és l'avaria detectada al vehicle o la necessitat de la reparació. Si es considera oportú el cap de reparacions avalua l'avaria per tal d'informar el client sobre la durada i el preu aproximat de la reparació. El client signa un document de conformitat de la reparació i el vehicle pot quedar-se en aquest moment o tornar-hi un altre dia que es pacte amb el client.

Definició de requisits

A l'igual del cas de la descripció de les àrees, es proporcionen només una part dels requisits a manera d'exemples, encara que no són tots els necessaris per tenir completament definides les necessitats del sistema. Es deixa com a exercici per als alumnes completar aquesta definició de requisits.

Es proposa la codificació següent per la definició dels requisits:

RF-RCn	Requisit Funcional de l'àrea de Recepció de Clients número n.
RE-RCn	Requisit Dades d'entrada de l'àrea de Recepció de Clients número n.
RI-RCn	Requisit Informació d'eixida de l'àrea de Recepció de Clients número n.
RH-RCn	Requisit Hardware de l'àrea de Recepció de Clients número n.
Rx-RPn	S'utilitzarà per als requisits de l'àrea de Reparacions.
Rx-FCn	S'utilitzarà per als requisits de l'àrea de Facturació i Cobrament.
Rx-ICn	S'utilitzarà per als per als requisits d' Informes i Consultes generals.

Àrea de recepció de clients

RF-RC1: Els sistema proporcionarà funcions de recerca i consulta de dades de clients i vehicles, i permetrà alternativament recerques amb el DNI del client, cognoms, o matrícula vehicle.

RF-RC2: El sistema permetrà donar d'alta nous clients i vehicles, i controlarà que no es dupliquen dades identificatives (matrícula vehicle i DNI client).

RF-RC3: Els sistema permetrà modificar les dades d'un client o d'un vehicle. També es permetrà modificar la relació de propietat entre un client i un vehicle, quan aquest es venga a altre client del taller.

RF-RC4: Baixes. Només es permetrà esborrar definitivament dades de clients i vehicles si no s'ha fet cap reparació relacionada amb el client o el vehicle. Si fa més de 5 anys des de l'última vegada que el client o el vehicle van fer ús del servei del taller es podran esborrar les dades del sistema, però se'n generaran registres adequats en dispositius d'emmagatzematge històric.

RE-RC1: Dades d'entrada del client: nom, DNI, adreça, telèfon.

RE-RC2: Dades d'entrada del vehicle: matrícula, any de fabricació, marca, model, color.

Àrea de gestió de reparacions

RF-RP1: El sistema permetrà donar d'alta una nova reparació lligada a un vehicle ja existent. Es podrà introduir informació sobre l'avaria, diagnòstic, tasques previsibles, altres observacions del client. La data de reparació serà per defecte la del dia del sistema, encara es que podrà modificar.

RF-RP2: El full de reparacions pot imprimir-se en qualsevol moment si es considera necessari, per tal de revisar les tasques fetes fins al moment o afegir-hi algun canvi.

RF-RP3: Es proporcionaran funcions de consulta sobre les reparacions en marxa, les donades d'alta no iniciades, etc.

RF-RP4: El sistema permetrà introduir la informació sobre les tasques fetes a una reparació, es podrà detallar les tasques i hores fetes i els recanvis utilitzats.

RF-RP5: Per introduir els recanvis el sistema proporcionarà mecanisme de recerca i selecció de la informació dels recanvis del subsistema de magatzem. Una vegada seleccionat el recanvi

es demanarà la quantitat utilitzada i el sistema actualitzarà l'estoc del subsistema de magatzem.

RF-RP6: S'hi crearà una funció per marcar aquelles reparacions que s'han finalitzat i estan preparades per a ser facturades. Es posarà data de fi com la data del sistema encara que serà modificable per l'usuari. Una vegada fet aquest canvi no es podran afegir més tasques o recanvis a la reparació.

RE-RP1: Dades entrada alta reparació: data (sistema per defecte), descripció avaria, diagnòstic, data prevista de fi.

RE-RP2: Dades entrada tasques reparació: data, descripció de la tasca, hores, operari (nom o categoria).

RE-RP3: Dades entrada ús de recanvis, codi del recanvi (o nom, descripció, o selecció de la informació del subsistema de magatzem), quantitat.

Facturació

RF-FC1: Els sistema permetrà seleccionar una reparació finalitzada, per nom de client o matricula de vehicle per generar la factura si se sol·licita. En aquest cas el sistema farà tots el càlculs necessaris (hores per preu i preu de recanvis per quantitat, totals i IVA). El sistema enregistrarà la nova factura en el sistema i permetrà indicar si està pagada o no.

RF-FC2: La factura es podrà imprimir en qualsevol moment una vegada generada.

RF-FC3: Es crearà una funció per poder generar factures en bloc a partir de reparacions finalitzades. Es proporcionarà mecanisme de recerca i selecció o opció de fer les factures de totes les reparacions finalitzades. El sistema calcularà i generarà totes les factures automàtica-ment o una seleccionada.

RI-FC1: Es donaran opcions per imprimir totes les factures generades o una existent. En la factura impresa s'inclouran les dades del client, del vehicle i el detall de les reparacions.

Capítol 3

Anàlisi. Modelització de processos

Objectius

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

- Aprendre què significa l'abstracció i la síntesi, i com amb l'aplicació d'aquests conceptes es pot obtenir el que ha de realitzar un sistema informàtic, a partir de la definició de requisits desenvolupada prèviament.
- Comprendre què és l'anàlisi en l'àmbit de l'Enginyeria del programari i quins són els punts fonamentals que s'han de tenir en compte per a desenvolupar aquesta activitat.
- Saber que modelitzar els processos gràficament proporciona una visió del sistema útil, completa i fàcil de comprendre per als usuaris i els analistes.
- Conèixer la dificultat que comporta la realització d'una anàlisi d'un sistema, i la necessitat d'obtenir el resultat final a partir de revisions i refinaments successius.

- Conèixer la tècnica dels diagrames de flux de dades, els seus components i diferents especificacions[2] del programari, és a dir, la documentació completa i precisa del què ha de realitzar el sistema per a cobrir els requisits d'usuari.

3.1. Introducció

Per a realitzar i documentar l'anàlisi i que el resultat siga complet, és necessari utilitzar mètodes que permeten modelitzar el sistema que ha de cobrir els requisits. Aquests mètodes, igual que en altres disciplines, utilitzen tècniques gràfiques completades amb descripcions adequades, que ajuden a sintetitzar i estudiar el producte final que es vol desenvolupar.

Tot i que els mètodes proporcionen uns passos que s'han de seguir, dos analistes amb les mateixes especificacions i que utilitzen el mateix mètode no han de generar necessàriament el mateix model del sistema. La causa és que qualsevol sistema sempre té en la seua modelització i disseny una gran part de la creativitat i dels coneixements de l'enginyer informàtic.

Cada mètode d'anàlisi té una notació i punts de vista diferents, però tots els mètodes d'anàlisi estan relacionats per uns principis fonamentals comuns. Aquests principis han sorgit a partir d'investigacions, l'experiència i altres mètodes aplicats en diferents àrees d'enginyeria. Aquests principis són: el domini de la informació, la partició i jerarquia, la modelització, la representació de diferents vistes.

El domini de la informació proporciona la comprensió del que s'ha de representar i comprendre. Els sistemes informàtics estan dissenyats per a processar dades (o fets que generen algun control del sistema), i per a transformar-les d'una forma o d'una altra. Per això el seu model ha de contenir tres visions diferents de les dades que es processen, i que completen la seua comprensió:

- El flux de la informació que representa com canvien les dades i el control a mesura que es mouen dins d'un sistema.
- El contingut de la informació que representa els objectes individuals de dades i de control que formen part d'un conjunt major d'informació que transforma el sistema informàtic.
- L'estructura de la informació que representa l'organització interna de les dades i les seues relacions.

[2] Especificació: és un document que defineix de forma completa, precisa i verificable els requisits, el disseny, el comportament o altres característiques d'un sistema o component d'aquest.

La partició i jerarquia proporciona capacitat per a representar el problema per parts i d'una manera estructurada. Quan el sistema inclou un gran nombre de funcions els problemes són massa grans per a ser compresos com un tot. Per aquesta raó és imprescindible partir-los i dividir-los en mòduls que siguen més fàcils de comprendre i solucionar. És necessari establir quines són les connexions o interfícies entre els diferents mòduls en els quals s'ha subdividit el problema. Si els mòduls no estan totalment diferenciats i són independents entre si, és convenient realitzar una partició jeràrquica del problema i partir d'un element superior a un altre inferior amb increment del detall.

La modelització permet representar gràficament el problema. Els models es creen per tal de comprendre millor la realitat que s'ha de construir. Quan el producte real és un objecte físic, com una casa o un avió, el model és un reflex de la realitat a una determinada escala i amb un determinat nivell de detall. En el cas del sistemes informàtics els models han de representar la informació, els processos i les transformacions que afecten la informació. El model gràfic ha de completar-se amb informació escrita que proporcione una millor comprensió. L'activitat de modelització es fonamental per a obtenir una anàlisi correcta, encara que es poden utilitzar diferents mètodes de modelització.

La representació de diferents vistes del problema per tal d'entendre i representar la visió lògica o essencial i la visió física d'implementació del programari. La visió lògica dels requisits del sistema representa les funcions que han de realitzar-se i la informació que ha de processar-se, independentment dels detalls d'implementació. Per exemple, llegir dades d'un codi de barres no fa referència a la forma física com s'han de llegir les esmentades dades.

L'anàlisi del sistema és una tasca que permet l'enginyer informàtic especificar les funcions del sistema, la informació (domini) que s'ha de processar, com han de funcionar els programes, indicar les possibles connexions o interfícies entre els diferents mòduls del sistema i establir els requisits de disseny que els programes han de complir. Proporciona a l'enginyer la representació de la informació i les funcions.

3.2. Diagrama de Flux de Dades (DFD)

Els components fonamentals d'un sistema informàtic són les dades i els processos. Una anàlisi d'un sistema no cobriria tots els seus requisits sense considerar aquests dos de forma conjunta. Per a dissenyar un sistema s'han de tenir en compte les següents qüestions:

- Quines funcions ha d'exercir el sistema? Com estan connectades aquestes funcions?
- Quines transformacions es fan en el sistema? Quines dades d'entrada es transformen i quina informació d'eixida es produeix?

- Qui proporciona les dades d'entrada i on? A qui va destinada la informació d'eixida?

La principal funció dels sistemes informàtics és modificar la informació. Aquesta es transforma i es mou com un flux a través del sistema. Els diagrames de flux de dades estudien on s'origina la informació, com s'utilitza o modifica i quina és la seua destinació, incloent-hi també els processos que sofreix fins que l'aconsegueix.

Els diagrames de flux de dades mostren la visió lògica del flux de la informació i són una de les ferramentes més comunament utilitzades, sobretot en sistemes operacionals que donen suport a la gestió de les empreses on les funcions del sistema són de gran importància. També són útils per a modelitzar sistemes en temps real, encara que en aquests casos existeixen unes altres ferramentes més apropiades.

Els diagrames de flux de dades poden utilitzar-se per a representar un sistema a qualsevol nivell d'abstracció. Permeten realitzar de forma senzilla una abstracció del problema, independentment dels suports utilitzats per a realitzar els processos; permeten representar activitats que s'executen de forma paral·lela, ja que en els sistemes informàtics hi ha processos que a vegades es realitzen al mateix temps.

Alguns dels avantatges que aporta el seu ús són:

- Són fàcils de comprendre tant per a l'usuari com per als analistes i programadors.
- Permeten aïllar les diferents àrees de treball o de processos on ha d'actuar el sistema.
- Proporcionen mètodes per a obtenir diferents nivells de detall que faciliten la revisió del sistema a qualsevol nivell.

3.3. Components dels DFDs

El diagrama de flux de dades és una ferramenta gràfica que descriu les transformacions de la informació i com aquesta es mou a través dels diferents mòduls del sistema. Inclou símbols per a representar la informació tant externa com interna al sistema, processos, magatzems de dades i el flux de la informació. Aquesta representació es realitza independentment dels components físics del sistema.

3.3.1. Els processos

Els processos representen la part del sistema que transforma les dades i la informació. L'entrada d'un procés pot ser de dades que es proporcionen al sistema informàtic des de l'exterior, o bé d'informació

que ja ha sigut introduïda i processada abans al sistema informàtic. L'eixida és informació processada. Per tant, els processos sempre representen una acció que es realitza sobre les dades o informació d'entrada per a produir la informació d'eixida. S'han d'anomenar amb una sola paraula, o una frase senzilla que indique l'acció i l'objecte sobre el qual es realitza l'acció.

En diferents metodologies es representen els processos de diferents maneres com exemple es mostren a la figura 3.1 les notacions utilitzades per Yourdon (Yourdon, 1996), Gane-Sarson i MÉTRICA (Métrica, 2010).

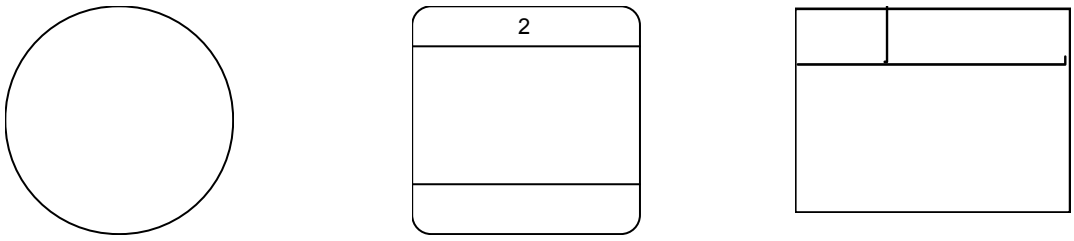


Figura 3.1. Representació de processos

3.3.2. Els fluxos de dades

Els fluxos de dades representen blocs d'informació o de dades que es desplacen d'un procés del sistema a un altre objecte, representat en el diagrama de flux de dades. S'han de designar amb un nom representatiu de les dades o informació que representen. Un mateix bloc de dades pot rebre un nom diferent per exemple: informació de client vàlid, informació de client no vàlid. En aquest cas la representació és única, com s'observa a la figura 3.2.

Figura 3.2. Representació de fluxos

3.3.3. Els magatzems

Els magatzems de dades s'utilitzen per a representar la informació en repòs. S'han d'anomenar mitjançant un nom en plural. A la figura 3.3 es mostren diferents representacions segons la notació utilitzada.

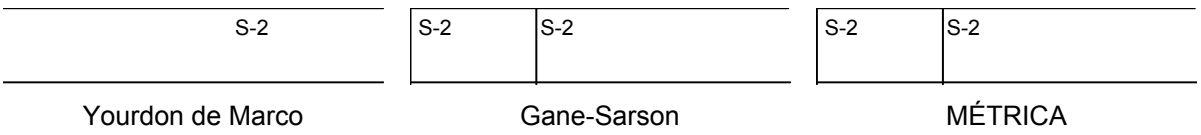


Figura 3.3. Representació de magatzems

3.3.4. Les entitats externes

Les entitats externes són objectes, persones o altres ens generadors o receptors d'informació amb els quals el sistema es comunica. Comunament són persones o agrupacions de persones, però pot donar-se el cas que siguen altres sistemes informàtics. Per ser externs al sistema, no és possible modificar el seu contingut, ni el concepte que representen. S'han d'anomenar amb un nom en singular. Es representen de diferents maneres, come s veu en la figura 3.4, segons la metodologia o notació.

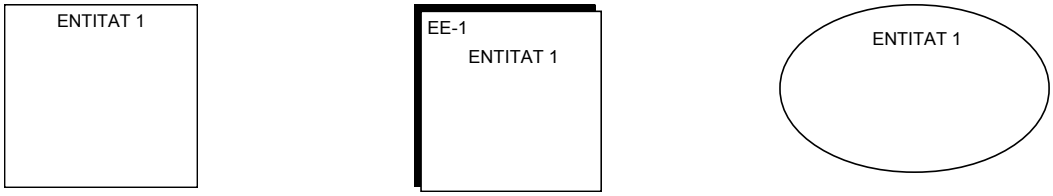


Figura 3.4. Representació de les entitats externes

3.3.5. Exemples

A continuació es proporcionen exemples dels processos bàsics d'alta, modificació, baixa i consulta que s'utilitzen habitualment en qualsevol DFD. Encara que cada cas s'ha de considerar particularment per tal de tenir en compte totes les seues necessitats a l'hora de desenvolupar el DFD corresponent.

Com a consideració important, cal tenir en compte que un flux d'eixida d'un magatzem només pot proporcionar informació sobre el que hi ha al magatzem, sense canviar el seu contingut. Quan un procés modifica un magatzem (alta modificació o baixa) es representa mitjançant la utilització d'un flux d'entrada.

Exemple 1: Donar d'alta informació. En aquest cas, pot ser necessari consultar informació d'altres magatzems, per a completar o comprovar les dades proporcionades per l'entitat o entitats externes (veure Figura 3.5).

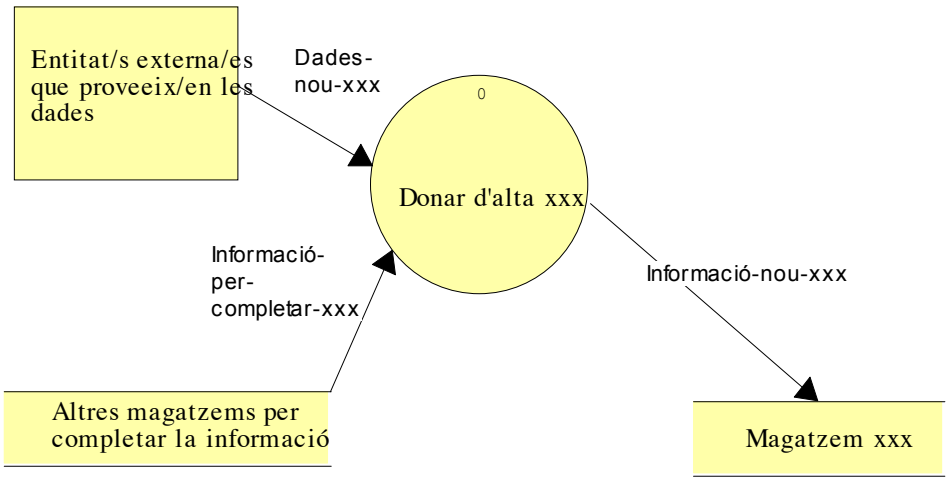


Figura 3.5. Exemple d'un procés d'alta d'informació

Exemple 2: Modificar informació que ja està en un magatzem. El procés ha de llegir dades per tal d'identificar la informació que s'ha de modificar, o bé es podria proporcionar a l'usuari algun mecanisme per tal de buscar i seleccionar aquesta informació (veure Figura 3.6). La informació s'ha de visualitzar, es llegeixen les dades que s'han de canviar i es pot demanar a l'usuari confirmació dels canvis. Sempre que es modifica informació de qualsevol magatzem és necessari identificar-la i llegir-la abans.

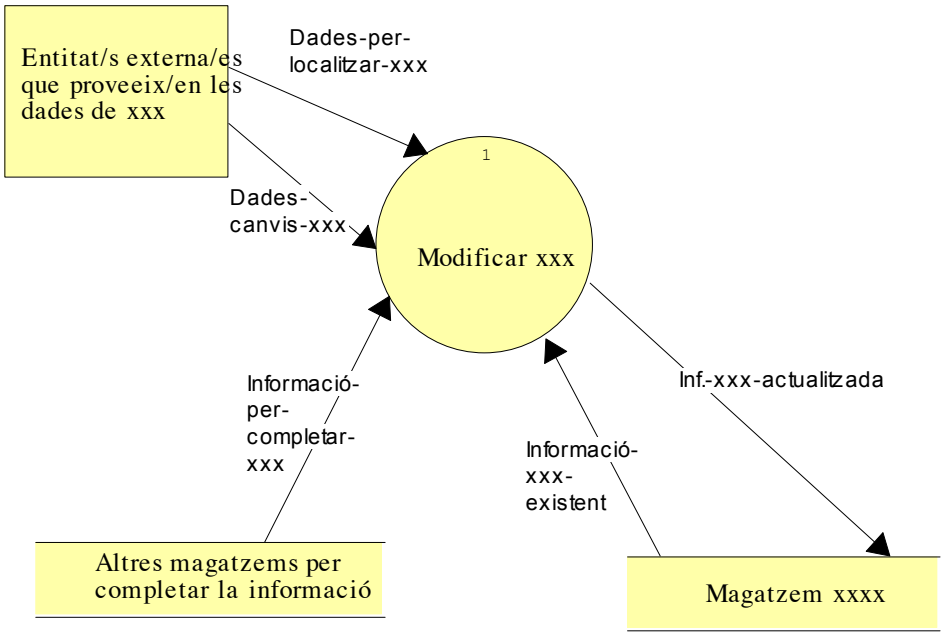


Figura 3.6. Exemple d'un procés de modificació d'informació

Exemple 3: **Donar de baixa informació.** Aquest cas és molt semblant a l'anterior, ja que l'eliminació d'informació modifica el magatzem corresponent. Habitualment en els sistemes de gestió no s'esborren les dades, sinó que es dóna de baixa informació mitjançant l'afegiment d'una data de baixa,

o alguna dada complementària, i en aquest cas el procés és una modificació (veure Figura 3.7). Sempre que es dóna de baixa informació de qualsevol magatzem és convenient mostrar la informació als usuaris, per tant s'ha d'identificar i buscar en el magatzem on s'allotja abans.

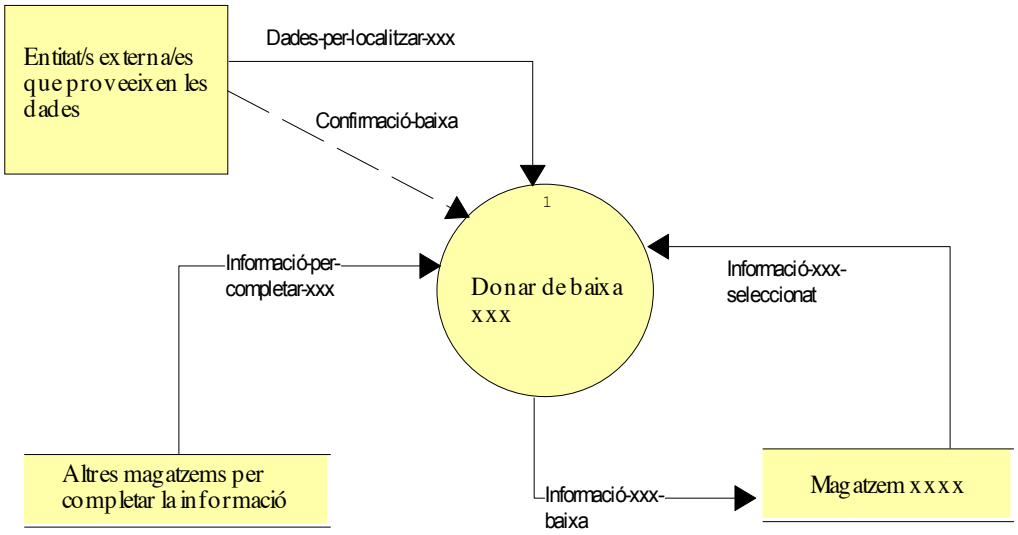


Figura 3.7. Exemple d'un procés de baixa

Exemple 4: Consultar informació d'un magatzem. (veure Figura 3.8) En aquest cas només cal llegir dades per a localitzar la informació a visualitzar, extraure aquesta informació del/s magatzem/s i visualitzar-la. No es modifica cap magatzem.

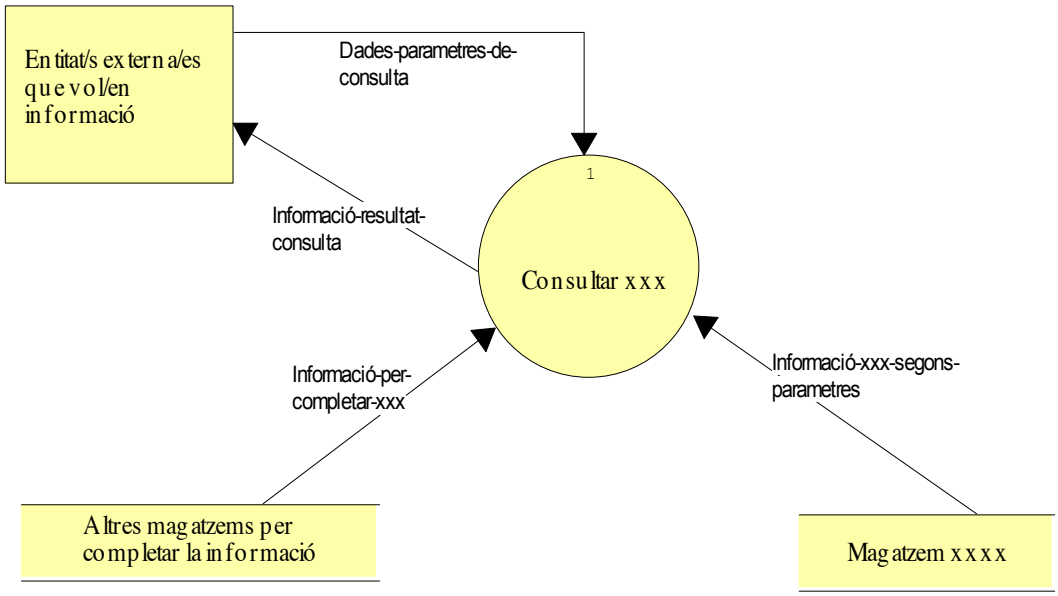


Figura 3.8. Exemple d'un procés de consulta d'informació

Exemple 5 :Primer nivell. anomenat **diagrama de context** (veure Figura 3.9). En el diagrama de context no es representen magatzems, excepte aquells que no són propis del sistema o subsistema objecte de l'anàlisi. En aquest cas el magatzem d'articles es manté des d'un subsistema de fabricació, i

aquest subsistema llegeix informació referent als articles que l'empresa comercialitza i modifica les dades d'estocs corresponents a comandes servides i per servir.

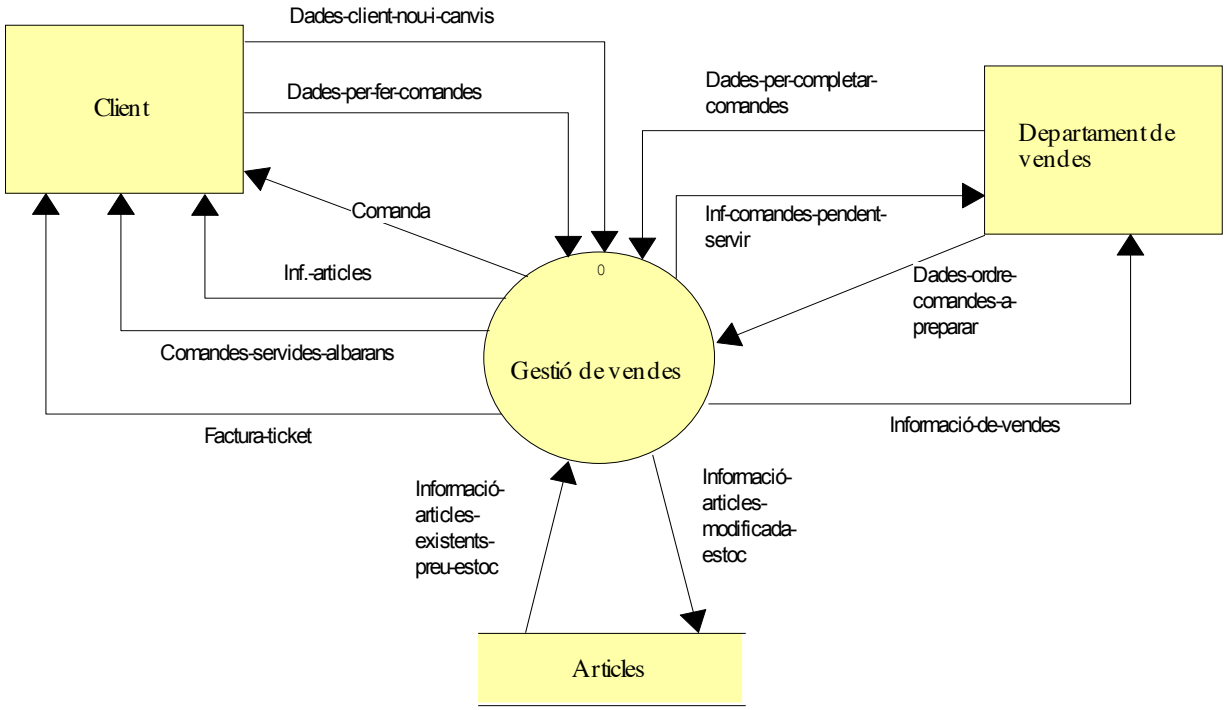


Figura 3.9. Exemple de diagrama de context amb un magatzem extern (Articles)

Exemple 6: Segon nivell. Es representen les principals funcions del sistema, tal com es mostra a l'exemple en la figura 3.10. Per a definir els processos en els quals es descompon inicialment la funcionalitat del sistema es pot partir de:

- L'abast funcional identificat del projecte, agrupant la funcionalitat del sistema en aquests apartats.
- L'abast organitzacional, considerant les àrees de negoci o departaments de l'empresa involucrats i que han de fer ús dels sistema.
- Agrupacions lògiques de les funcions que donen suport als requisits.
- Identificar quin seria el primer menú on els usuaris seleccionarien l'àrea de treball o la gestió que volen dur a terme.

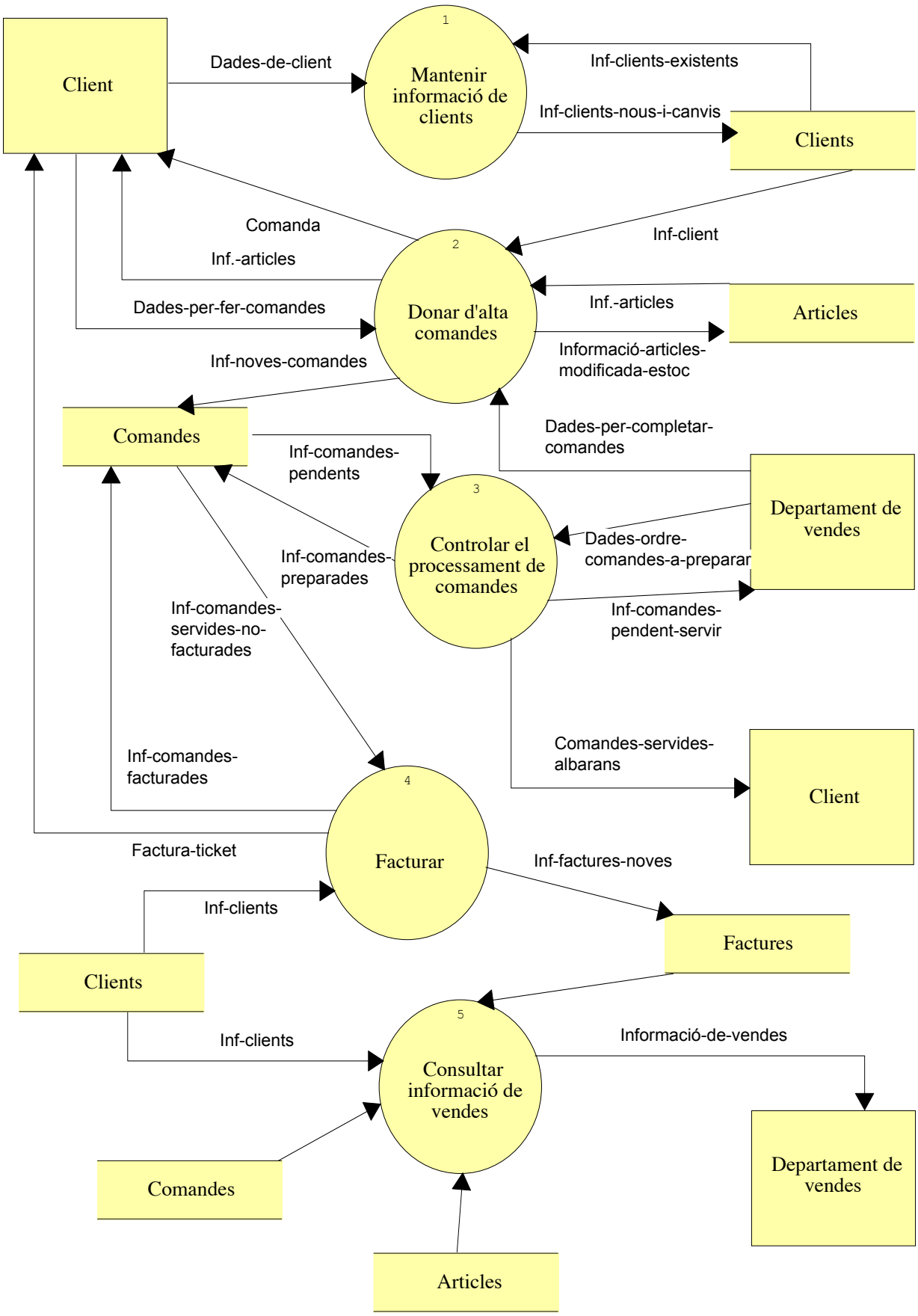


Figura 3.10. Primera explosió del model

3.4 Com dibuixar el DFD

A continuació es descriuen tres mètodes per desenvolupar un dels diagrames de flux de dades d'un sistema. En la pràctica s'apliquen de manera complementària.

A partir de la definició dels requisits, s'han d'identificar quines són les funcions que és necessari informatitzar per a donar suport al sistema d'informació. És necessari tenir en compte tots els aspectes identificats en la definició de requisits com són les restriccions, els recursos i objectius fonamentals del sistema, per tal de ser coherents durant tot el desenvolupament del producte. A partir de tota la documentació generada en la definició de requisits l'enginyer informàtic ha d'identificar:

- Quins processos han d'integrar el sistema.
- Quines dades captura cada procés del sistema.
- Quines dades s'emmagatzemen en cada procés.
- Quines dades s'introdueixen i extrauen de cada procés.

Inicialment existeixen dos enfocaments per a desenvolupar els diagrames de flux de dades:

1. Partint de la funcionalitat completa definida pels requisits, es crea el diagrama de context i es va refinant; és el que es denomina de dalt cap a baix (Top-down).
2. Partint de la funcionalitat de més baix nivell, es creen diferents bombolles per a cadascun dels requisits i s'agrupen cap a dalt; és el que es denomina de baix cap a dalt (Botton-Up).

Com a estratègia de suport per a desenvolupar els DFDs, es pot analitzar i agrupar els requisits en el seu context empresarial considerant els processos de negoci dintre l'abast del sistema. Una vegada definits els processos de negoci, es relaciona cada requisit amb un d'ells i es pot aplicar, segons el detall, una combinació dels dos enfocaments anteriors, tal com es descriu a l'apartat següent.

3.4.1. A partir dels processos de negoci

Els processos de negoci són activitats que du a terme l'empresa per arribar a aconseguir els seus objectius. Tenint en compte aquells que estan suportats pel sistema informàtic, configuren una llista de successos que es produeixen externament al sistema i que provoquen que aquest haja de donar suport mitjançant funcions, que l'enginyer informàtic ha de dissenyar i posteriorment implementar.

Identificar els processos de negoci permet tenir una visió del sistema que ajuda a reconèixer les entitats, processos informàtics, magatzems, fluxos i continguts d'aquests necessaris per a començar a realitzar els diagrames de flux de dades; a més a més permeten revisar i valorar els requisits definits prèviament.

En la taula 3.1 es mostra com aplicar esta estratègia pera a dos processos de negoci:

- Un client fa una comanda
- Un client anul·la una comanda

Procés de Negoci Funció/requisit	Quines dades o informació es proporciona	Qui/què proporciona (EE/magatzem)	Quina informació es produeix	On/ a qui va (EE/magatzem)
Procés de Negoci: Un client fa una comanda				
Verificar dades client	Número i/o nom client	Client		
Extraure inf. client	Informació de client	Magatzem de clients	Informació de client	Procés Generar comanda
Introduir dades client	Dades de client	Client	Informació de client	Magatzem de clients
Generar comanda	Dades comanda Informació d'articles Informació de client	Client Magatzem d'articles Verificar dades/alta de client	Comanda inf. comanda	Client Magatzem de comandes
Procés de Negoci: Un client anul·la una comanda:				
Extraure dades comanda	Número de comanda Informació de comanda i client	Client Magatzem de comandes Magatzem de clients	Informació de comanda anul·lada Informació d'articles actualitzada	Confirmar baixa
Confirmar baixa	Informació de comanda Informació d'articles	Extraure dades	Informació de comanda anul·lada Informació d'articles actualitzada	Magatzem de comandes Magatzem d'articles

Taula 3.1. Taula de processos de negoci corresponents a un àrea de vendes

Els passos a seguir per a desenvolupar la taula són:

1. S'identifiquen els processos de negoci dintre l'abast del sistema
2. Es relacionen els requisits funcionals o les funcions informàtiques corresponents per a cadascun del processos; i es detallen en la primera columna.
3. Per a cada funció/requisit, en la segona columna, s'identifiquen quines són les dades d'entrada (proveïdes per persones, dispositius o altres sistemes), i/o la informació d'entrada (obtinguda dels magatzems de dades existents).
4. En la tercera columna es relacionen els proveïdors de les dades (persones, dispositius o altres sistemes), que seran entitats externes i els magatzems, on prové la informació.
5. En la quarta columna s'identifica la informació que es proporciona com resultat de processar les dades i la informació d'entrada (seran els fluxos d'eixida).

6. Finalment en l'última columna es detalla quina és al destinació de la informació d'eixida, que poden ser magatzems (on es modifica la informació), i/o entitats externes que reben informació.

Aquesta taula permetrà identificar les bombolles del DFD, els fluxos, les entitats externes i els magatzems, i proporcionarà una base per organitzar les bombolles en els diferents nivells del diagrama.

3.4.2. Dibuixar el DFD de dalt cap a baix (*Top-down*)

Aquest procediment planteja el desenvolupament del DFD partint del diagrama de context i expandint-lo successivament, per anar creant nous diagrames en els quals s'introdueix més detall cada vegada.

En sistemes grans un sol procés pot estendre's moltes vegades, fins que s'obté el nivell de detall que permet l'enginyer informàtic comprendre cadascun dels subprocessos i el sistema complet. Els processos de més baix nivell i que no es descomponen més es corresponen amb funcions informàtiques.

Descomposició de processos

Consideracions a tenir en compte quan es subdivideix un procés (també es diu expandir o explosionar):

- Refinament gradual. No és convenient dividir un procés en més de set subprocessos i no es pot fer una expansió en la qual només hi apareguen dos subprocessos. Crear molts processos en un nivell fa que el diagrama siga difícil de manipular i comprendre, i a més a més això significa que no s'ha fet una descomposició gradual. Per una altra banda, crear una descomposició amb només dos subprocessos pot assenyalar que potser no siga necessària o que s'està fent amb poc detall.
- Numeració dels processos jeràrquicament: el procés del diagrama de context es numera amb 0, els processos de la primera explosió amb 1, 2, 3, etc.; els de nivells inferiors de manera jeràrquica. Per exemple els obtinguts de l'expansió del procés 1 es numeren com a: 1.1, 1.2, 1.3, 1.4 i així successivament.
- Cal mantenir la consistència entre processos**,** l'expansió en nivells inferiors serveix per a identificar més detalls relacionats amb els processos interns. La informació que necessita com a entrada/eixida (fluxos) un procés ha de ser la mateixa que l'utilitzada en la seua descomposició.
- Els magatzems de dades que són només significatius en un cert nivell de detall no cal representar-los en nivells superior. Per exemple un magatzem que únicament es comunica amb fluxos d'entrada i d'eixida a un procés és significatiu només en l'expansió d'aquest procés.

- S'afegeixen controls només en els diagrames de baix nivell. No es representen en els diagrames de flux de dades processos sobre possibles errors de seguretat del sistema. Encara que aquesta informació és necessària per al disseny final no ho és per a identificar les funcions del sistema. Poden afegir-se quan s'aconsegueix un nivell de detall que el fa necessari.
- Tampoc s'especifica l'existència de còpies d'informació, ni condicions especials per a realitzar un determinat procés o un altre.

Repetir la subdivisió

Fins a quin nivell de detall és convenient subdividir els processos?

- Quan l'especificació de la funció puga desenvolupar-se de forma adequada i amb un nivell de detall convenient al model. L'especificació de les funcions es descriu en l'apartat 3.7.
- Quan existeixen pocs fluxos d'entrada i d'eixida.
- Quan, si es descompon, es perd el significat i s'obtenen processos excessivament senzills que no són representatius.

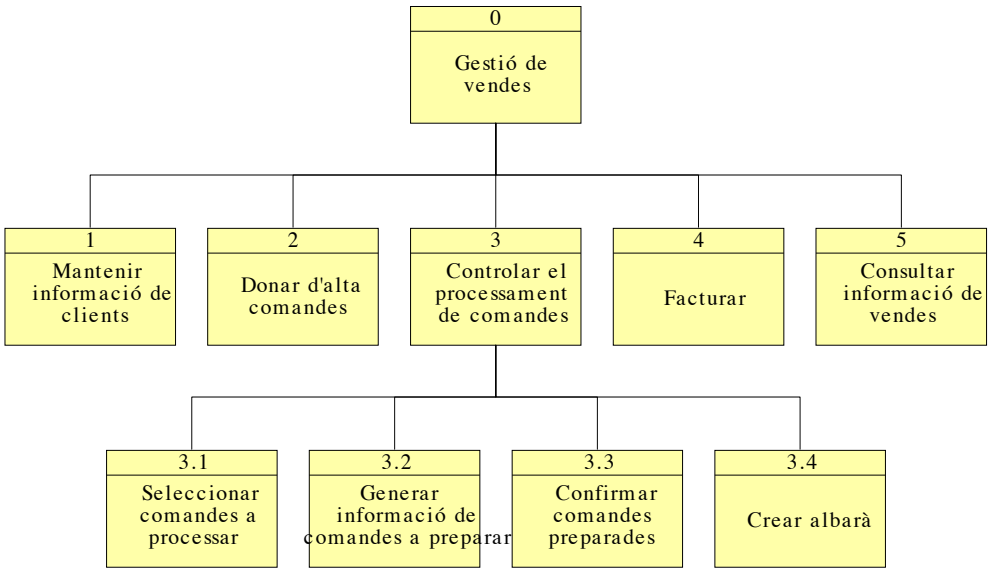


Figura 3.11. Representació jeràrquica dels processos d'un DFD

El resultat final podria representar-se com un arbre de processos, on el primer procés correspon al del diagrama de context i els últims són funcions que no es descomponen més. Cal recomanar que aquest arbre, sempre que siga possible, estiga equilibrat, tenint en compte sempre l'agrupació de processos per afinitat i contingut. En la figura 3.11 es mostra la representació jeràrquica d'un arbre de processos generat a partir d'un DFD.

Com a resum en la taula 3.2 es mostren els passos que cal seguir per a desenvolupar un DFD de dalt cap a baix.

Crear un diagrama que represente tot els sistema amb un únic procés, totes les entrades, totes les eixides i les entitats externes.	Diagrama de Context i Entitats externes
Subdividir el procés en subprocessos de manera gradual. Connectar les entrades i les eixides amb els subprocessos adequats mantenint la consistència	Nivells intermedis Fluxos
Identificar els magatzems i dibuixar-los on siguen representatius	Magatzems
Repetir la subdivisió fins tenir funcions d'un nivell adequat	

Taula 3.2. Passos que s'han de seguir per a desenvolupar un DFD de dalt cap a baix

Les ferramentes CASE (Computer Aided Software Engineering) són eines informatitzades que donen suport a les tècniques utilitzades en l'Enginyeria del programari. L'ús d'aquestes eines ajuda a crear DFDs i altres models de qualitat, ja que donen suport a les regles de consistència descrita i altres específiques de cada tècnica de modelització.

3.4.3. Dibuixar el DFD de baix cap a dalt (*Botton-up*)

A vegades l'estudi dels requisits i dels processos de negoci es fa a un nivell molt baix, per tal de comprendre millor funcions o alguns detalls del sistema. En aquest cas els processos que s'obtenen es corresponen amb els nivells més baixos del diagrama de flux de dades, i és necessari reagrupar-los per tal d'obtenir els nivells intermedis i el diagrama de context del sistema, si s'escau.

De manera similar al que es fa per subdividir els processos de nivell alt, també cal identificar processos, entrades, eixides, i dades a emmagatzemar, per a finalment agrupar els processos mantenint la consistència. En aquest cas, la decisió d'agrupar uns processos en uns altres depèn de l'abast del sistema, de l'afinitat entre les activitats a les quals dóna suport cadascuna de les xicotetes bombolles de més baix nivell (funcions), i del nombre de processos que s'agrupen en cadascuna de les bombolles de nivell superior.

Identificar transformacions de dades de baix nivell	Bombolles de més baix nivell
Identificar la informació d'entrada i d'eixida	Fluxos
Identificar la informació que s'ha d'emmagatzemar	Magatzems
Identificar els productors i/o receptors d'informació	Entitats externes
Agrupar els processos en altres que els contenen	Bombolles de nivell més alt
Mantenir la consistència	

Taula 3.3. Passos que s'han de seguir per a desenvolupar un DFD de baix cap a dalt

Com a resum, en la taula 3.3 es mostren els passos que cal seguir per a desenvolupar un DFD de baix cap a dalt.

Quan es desenvolupa un DFD normalment s'identifiquen processos d'un nivell intermedi. Per tant, per una banda caldrà aplicar la subdivisió de processos cap a baix, per tal d'expandir aquells que ho requereixen, i per un altra banda, caldrà agrupar de baix cap a dalt els processos intermedis obtinguts fins a arribar al diagrama de context.

3.5. Avaluació del DFD

És fonamental avaluar els diagrames de flux de dades per a verificar si són correctes, sobre aquest punt s'han de tenir en compte dos aspectes:

- Validar: s'ha dissenyat el sistema correcte?
- Verificar: s'ha dissenyat el sistema de forma correcta?

Els errors, omissions o inconsistències en ocasions permeten identificar deficiències en el disseny que es realitza, o bé en la comprensió inicial de la problemàtica del sistema. Per això l'enginyer informàtic ha de comprovar que cadascun dels diagrames de fluxos compleixen les condicions descrites anteriorment, i que reflecteixen els processos que ens permeten obtenir un sistema amb la funcionalitat desitjada.

Per a validar la correcció de l'anàlisi cal contrastar amb els usuaris i amb la documentació de definició de requisits que el model desenvolupat dóna suport a les necessitats identificades tenint en compte les restriccions i els objectius definits. Entre altres aspectes caldrà avaluar la traçabilitat, considerant si tots els requisits d'usuari estan representats per la funcionalitat del sistema i a l'inrevés.

Per a verificar la correcció del model s'ha de comprovar que la tècnica s'utilitza de forma correcta i que no hi ha inconsistències entre la informació gràfica i la documentació que acompanya el model.

Les condicions que ha de complir el model són:

- Tots els processos, fluxos, magatzems i entitats externes han d'estar etiquetats amb noms únics i amb significat. S'han d'assignar noms o etiquetes significatius, per a permetre els usuaris i els enginyers saber què es produeix en cada procés i quina informació es transfereix en cada flux.
- 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 i no 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 processos que els actualitzen i processos que obtinguen informació d'aquests (excepte magatzems externs al procés).
- Tots els processos de més baix nivell (funcions) han d'estar descrits convenientment, tal com s'estableix en l'apartat 3.7.

Altres recomanacions

Per a etiquetar el processos cal seleccionar noms que indiquen l'acció que es porta a terme, el més correcte és triar un verb i un objecte que rep la informació del verb. El nom ha de descriure completament el procés. També és recomanable seleccionar noms per als processos que relacionen els fluxos d'entrada i els d'eixida. I s'han d'evitar noms indefinits o genèrics (processar, revisar, organitzar) que puguen ocasionar confusió.

L'eixida dels processos pot ser:

- Flux de dades amb informació afegida.
- Una resposta o un canvi en la forma de les dades.
- Un canvi o condició d'estat.
- Canvi de contingut o canvi d'organització.

Cal tenir en compte consideracions gràfiques sobre la presentació dels DFD: bona presentació, evitar que els fluxos s'encreuen.

3.6. Diccionari de dades

El diccionari de dades conté la descripció lògica de la informació que representen els magatzems d'informació i cadascun dels fluxos que apareixen en el model gràfic. És un llistat organitzat de totes les dades del sistema, amb definicions precises i rigoroses, perquè tant l'enginyer informàtic com l'usuari tinguen una entesa comuna de totes les entrades, eixides, magatzems de dades i càlculs intermedis. Defineix el significat dels fluxos i magatzems, la composició de blocs d'informació complexos o compostos i especifica valors rellevants.

3.6.1. Per què és necessari el diccionari de dades?

Durant el disseny dels diagrames de fluxos de dades s'ha considerat l'aspecte del domini de la informació que correspon amb com es transforma la informació en el sistema. El diccionari de dades proporciona el mecanisme per a definir el contingut i l'estructura de cadascuna de les dades representades als diagrames i permet definir el contingut representat per cada fletxa i cada magatzem de dades.

Al diccionari de dades s'inclouen a més detalls, com per exemple la longitud de les dades, el tipus o format visual, i quins altres noms pot tenir una dada dins de l'organització. Es comença a desenvolupar durant el disseny dels diagrames de fluxos de dades i ajuda els enginyers en informàtica a identificar requisits funcionals i de dades que no estaven especificats inicialment. Alguns d'aquests detalls poden haver-se obtingut en la definició dels requisits d'usuari i altres per mitjà d'investigacions posteriors més detallades, al mateix temps que es desenvolupa l'anàlisi.

Els diccionaris són de gran utilitat en sistemes on es manipulen grans volums d'informació. Fins i tot per a sistemes xicotets un enginyer informàtic no pot mantenir tota la informació sobre les dades que ha de manipular, sense anotar-la i especificar les característiques en cap document. Hi ha diccionaris de dades automatitzades, i les ferramentes CASE permeten realitzar validacions automàtiques per a comprovar la consistència del diccionari i els diagrames de fluxos de dades.

Finalment, els diccionaris de dades permeten descriure en detall el contingut de la informació d'entrada i d'eixida del sistema: formats d'informes, com es volen processar les dades, estructures d'arxius i necessitats de capacitat del sistema. Aquesta informació permet avaluar l'anàlisi realitzada i, fins i tot, estimar la factibilitat o utilitat dels processos.

3.6.2. Notació i organització del diccionari de dades

Existeixen dos tipus de dades que s'han de descriure en el diccionari, les dades compostes i les elementals. Les dades elementals, també denominades elements de dades, són la unitat més xicoteta que té sentit per a l'enginyer informàtic. Són els blocs d'informació bàsics i el concepte és equivalent als camps en arxius. Per a aquestes dades elementals es proporciona una definició més explícita en funció dels valors que poden assumir. Les dades compostes són grups de dades elementals relacionades entre si. La seua definició es dóna en funció de les dades elementals que els componen.

Les eines CASE proporcionen mecanismes per a crear i mantenir el diccionari de dades[3] i proporcionen funcionalitat per obtenir els llistats corresponents, continguts i organització. En qualsevol cas és convenient definir normes per tal d'anomenar les dades elementals i les dades compostes. Un diccionari de dades per a ser complet ha d'incloure com a mínim els següents quatre llistats, amb la informació adient en cadascun dels casos:

1. Llistat de magatzems.
2. Llistat de fluxos.
3. Llistat de dades compostes.
4. Llistat de dades elementals.

Dades elementals

S'han d'identificar les dades elementals per mitjà de la següent informació:

- **Nom:** s'assigna a cada dada elemental un nom significatiu i únic. És important establir alguns estàndards per a la nomenclatura d'aquests, en ocasions semblants als utilitzats en programació (menys de 30 caràcters, sense espais en blanc, etc.). Aquest nom és el que s'utilitza per a fer referència a cada element durant tot el desenvolupament del sistema. Habitualment s'utilitzarà el nom de l'objecte al qual pertany davant del nom de la dada, per diferenciar-la i reconèixer-la clarament. Per exemple, «Factura número» és la dada elemental que correspon amb el número seqüencial d'una factura.
- **Descripció:** s'ha de proporcionar una descripció concisa i clara que indique de manera breu el que aquesta representa. Per a descriure aquesta informació es considera que el possible lector no té coneixements previs respecte al funcionament del sistema i s'ha d'utilitzar paraules comprensibles. Hi ha termes que es defineixen tots sols i que l'enginyer informàtic pot considerar que no cal especificar-hi una descripció, per exemple: data de naixement, estatura actual, etc. Per exemple, la dada simple número de factura pot definir-se com a «número únic, seqüencial que s'assigna a cada factura com a document per a identificar-la tant a nivell intern com extern».
- **Àlies:** a vegades una mateixa dada elemental és coneguda amb noms diferents, segons l'usuari o el departament que la utilitze; aquest punt serveix per a incloure tots aquests possibles noms que se'ls pot assignar a una mateixa dada elemental. Per exemple, una factura, també anomenada nota de pagament, etc. En alguns casos i organitzacions pot ser convenient indicar noms abreviats i fins i tot sigles que s'utilitzen per a anomenar algunes dades.
- **Longitud:** s'indica el nombre d'espais que l'element ocupa, sense tenir en compte com estan emmagatzemats internament. També és convenient assenyalar si són caràcters numèrics o no.

[3] En les eines CASE la informació sobre les dades dels models s'inclouen en el que s'anomena "Repository".

- **Valors de les dades:** per a algunes dades i processos només es permeten determinats valors. En aquest cas és necessari registrar els valors possibles en el diccionari de dades.

És convenient establir un format per a la descripció de cada dada elemental que incloga tots aquests conceptes.

Dades compostes

Les dades compostes poden estar formades per una seqüència de dades, una repetició o una selecció de dades elementals. Per a diferenciar les dades compostes en aquest text s'anomenen amb DC davant, i les paraules que les formen unides amb un guionet. Les dades elementals s'anomenen amb el nom del posseïdor, amb majúscula la primera lletra, i separant amb un espai es dona el nom de la dada elemental amb minúscules. Si aquesta dada elemental té un nom compost, les paraules s'uneixen amb guionets. Per exemple, l'any en què va nàixer un alumne serà «Alumne any-naixement».

- **Relació seqüencial:** estan constituïdes per una concatenació de dues o més dades; aquestes al mateix temps poden ser dades elementals o no. Per exemple, les dades de l'estudiant poden incloure: Nom (nom, cognom i cognom2), DNI, adreça, CP, telèfon, etc.:

DC-Estudiant = Estudiant nom + Estudiant DNI + Estudiant adreça + Estudiant CP + Estudiant telèfon

- **Relació de selecció:** una dada pot tenir diferents alternatives de construcció. Per exemple, el codi d'un client pot ser el seu DNI, o un codi del país més el número de passaport. No fa referència al fet que el valor de la dada puga ser un o un altre, sinó a l'estructura o l'element dada. Exemple:

DC-Client = [Client DNI, Client codi-país + Client número-passaport]

- **Relació d'iteració o repetició:** una dada pot estar formada per la repetició d'un nombre determinat de vegades d'una altra dada. Per exemple, les dades d'una factura inclouen un bloc repetitiu corresponent als articles i unitats cobrades, i la seua composició seria:

DC-Factura = Factura número + Factura data + Client codi + 1{Article codi + Factura quantitat-article}n

- **Dades opcionals:** hi ha dades en una estructura que són opcionals, és a dir que poden prendre o no valors per a una determinada idea de l'element de dades. Per exemple:

DC-Client = Client domicili-enviament + (Client domicili-facturació)

Llistat de dades compostes		
Nom	Descripció	Estructura
DC-Dades-Client	Dades personals o de l'empresa que cada client proporciona	= Client nom + Client NIF + [Client cognom1 + Client cognom2 + Client nom, Client nom-empresa] + Client adreça + Client telèfon
DC-Client	Informació del client que es genera a partir de les dades que aquest proporciona i que es registra en el sistema	= Client número + Client nom + Client import-deutor + Client tipus + Client descompte + Client data-alta + Client data-anul·lació

Taula 3.4. Exemple de documentació de les dades compostes

En al taula 3.4 es proporcionen exemples d'organització del llistat de dades compostes. I en la taula 3.5 es proposa una estructura per organitzar el llistat de dades elementals.

Llistat de dades elementals				
Nom	Descripció	Àlies	Tipus/Long.	Valors
Client número	Número únic i exclusiu que s'assigna de forma seqüencial a cada client quan es dóna d'alta	Núm.	6 dígits	Sencer

Taula 3.5. Exemple de documentació de dades elementals

Com s'han de documentar els fluxos i magatzems

En general, a cada flux i magatzem de dades se li assigna una dada composta. En el cas de magatzem de dades és important registrar el volum previst que és necessari tenir en compte per a definir la capacitat dels arxius del sistema.

Magatzems
Nom del magatzem: Clients
Descripció: Informació de les persones amb les quals l'empresa manté una relació comercial.
Volum: (Mitjà) 5.000
Dada composta: DC-Client

Taula 3.6. Exemple de documentació de magatzems

Llistat de fluxos					
Nom	Id.	Origen	Destinació	Freqüència	Dada composta
Dades Client Nou	F-10	E. E. Client	P. Gestionar vendes	(Vegades per dia o setmana)	DC-Dades-Client
Inf. Client Nou	F-13	P. Alta client	M. Clients		DC-Client

Taula 3.7. Exemple de documentació de fluxos

A la taula 3.6 es mostra una proposta de documentació per als magatzems i a la taula 3.7 es mostra una proposta de continguts del llistat de fluxos.

3.6.3. Validació i desenvolupament del diccionari de dades

El diccionari de dades és un document que genera l'enginyer informàtic durant el desenvolupament del model del sistema, però l'usuari ha de ser capaç de llegir-lo i d'entendre'l per a poder verificar l'anàlisi desenvolupada. A vegades, aconseguir l'acceptació de l'usuari de la notació utilitzada pot parèixer complex, i en ocasions el més probable és que l'usuari no estiga disposat a llegir tot el diccionari. L'usuari pot verificar-lo en conjunt amb els diagrames de flux de dades; l'enginyer informàtic té la responsabilitat de verificar que tots els fluxos i magatzems de dades que apareixen en els DFD estan definits en el diccionari de dades de forma única, que s'ha utilitzat la notació correcta en cada cas, i que no hi ha dades al diccionari que no s'utilitzen en els diagrames de flux de dades.

En un sistema mitjà o gran, el diccionari de dades pot incloure un volum elevat d'informació. Per això es fa cada vegada més necessari utilitzar alguna ferramenta informatitzada per a desenvolupar el diccionari de dades. Alguns sistemes de bases de dades proporcionen ajudes prèvies per a desenvolupar els diccionaris de dades, però en aquests casos s'imposen les limitacions de nomenclatura de la base de dades utilitzada. Com s'ha mencionat anteriorment, les eines CASE són les que proporcionen utilitats més potents per a l'especificació del diccionari de dades, permeten validar la consistència entre els diagrames de flux i el diccionari, i obtenir informes de gran utilitat per a l'enginyer informàtic. Si no es disposa d'aquestes ajudes és convenient almenys utilitzar un processador de textos.

Construir un diccionari de dades és una de les tasques més tedioses i llargues de l'anàlisi del sistema, no obstant, és també una de les més importants, sense un diccionari formal que definisca el significat dels termes no es poden obtenir resultats precisos.

A mesura que es desenvolupa el diccionari de dades poden produir-se canvis en el model gràfic, o el model pot sofrir modificacions i actualitzacions posteriors. En qualsevol cas cal comprovar que tots els fluxos i magatzems estan convenientment documentats en el diccionari de dades i que tot el que hi ha documentat en el diccionari de dades es correspon amb la representació gràfica d'un d'aquests dos elements.

3.7. Descripció de funcions

La descripció de funcions permet completar el DFD en l'apartat de les transformacions que pateixen les dades i la informació. Tot sol, el model pot donar una idea aproximada del programari que s'ha dissenyat, però cal afegir informació adequada que proporcione als enginyers i programadors més detalls sobre els programes i funcions que s'han de desenvolupar.

Per a cadascun dels processos de més baix nivell, denominats també funcions primitives, representats en els diagrames de flux de dades, és necessari donar una descripció que permeta obtenir una completa comprensió del sistema i de les seues funcionalitats. Per a aquestes descripcions pot utilitzar-se el llenguatge natural o bé establir determinats estàndards, semblants a algorismes d'alt nivell. Es detalla la seqüencialitat dels passos o accions que han de realitzar-se per a realitzar una determinada funció, així com també les validacions que el sistema realitza per als processos, els passos que un possible usuari ha de realitzar, etc.

L'especificació del procés ha de realitzar-se de manera que es puga verificar tant per l'enginyer informàtic com per l'usuari. En general la tècnica més estesa és utilitzar llenguatge narratiu però si no s'expressa correctament pot donar lloc a ambigüitats i no és molt útil per a descriure determinades situacions, com ara accions alternatives o repetitives.

D'altra banda, si s'opta per utilitzar especificacions semblants a llenguatges de programació o algorismes d'alt nivell, la comunitat de futurs usuaris implicada pot tenir una certa actitud poc inclinada a revisar aquestes especificacions.

Utilitzar una ferramenta o una altra depèn dels mitjans a l'abast de l'enginyer informàtic (ferramentes automatitzades), de les preferències de l'usuari i de la naturalesa dels processos. A més, una bona ferramenta d'especificacions de processos no ha d'influir sobre les decisions de disseny o implantació del programari. Una especificació del procés desenvolupada en llenguatge algorísmic supedita el desenvolupament posterior del programador que podria elaborar un algorisme més eficaç en un altre cas.

lEs poden utilitzar diferents tècniques per a descriure les funcions, les més conegudes són: el Llenguatge narratiu, Llenguatge estructurat, Pre/postcondicions i Taules de decisió. En qualsevol dels casos ha de tenir-se en compte que els destinataris de l'especificació poden tenir diferents perfils respecte al sistema, poden ser des d'usuaris directes i indirectes fins a programadors. Optar per un mètode o un altre depèn de la naturalesa i complexitat dels processos que s'han de descriure, i de l'equip de treball que ha de participar en les activitats de disseny i construcció del sistema.

3.7.1. Llenguatge narratiu

El llenguatge narratiu fa referència al llenguatge natural que s'utilitza habitualment per a descriure i desenvolupar tot tipus de documents. En aquest cas el vocabulari no està restringit, fet que fa possible la utilització de termes que no estan al diccionari de dades i que, per tant, poden no tenir un significat

concret. Amb el llenguatge narratiu és molt difícil expressar les accions alternatives i repetitives i les decisions, a més és molt ambigu.

En el cas d'utilitzar llenguatge narratiu s'ha de procurar restringir el llenguatge a la informació detallada al diccionari de dades, als verbs que indiquen accions i en cap cas estendre'ns en una especificació única.

3.7.2. Llenguatge estructurat

El llenguatge estructurat és un llenguatge natural al qual se li afegeix una determinada estructura. És a dir, un subconjunt de l'idioma al qual se li apliquen restriccions sobre les expressions i frases que poden utilitzar-se. El propòsit és obtenir una ferramenta intermèdia entre la precisió del llenguatge de programació i la informalitat i llegibilitat del llenguatge comú. Utilitza frases imperatives senzilles formades per verbs i objectes, i frases que es prenen de la programació estructurada per a poder expressar processos de repeticions i seleccions. En la taula 3.8 es mostren alguns exemples.

Calcular la dada Factura import-total com Factura import + IVA	Donar a Article preu-unitari el valor fix de 500 euros
	Multiplicar Article preu-unitari per Factura quantitat
Fer Factura import-final = Factura import – Client descompte	

Taula 3.8. Exemples de definicions/sentències en llenguatge estructurat

Components del llenguatge estructurat:

- **Verbs:** els verbs que es poden utilitzar han de correspondre amb accions que un sistema informàtic pot realitzar, com per exemple: *aconseguir*, *mostrar, escriure, sumar, restar, buscar, trobar, esborrar, moure, substituir, ordenar, afegir,* etc. Són suficients entre 40 i 50 verbs.
- **Objectes:** només poden utilitzar-se noms d'objectes definits en el diccionari de dades (dades elementals o compostes) i termes coneguts per la seua utilització en el sistema.
- **Expressions:** les expressions s'hereten de la programació estructurada i es corresponen amb accions de repetició i condició, tal com es mostra en la taula 3.9.

El principals avantatges del llenguatge estructurat són:

- Permet desenvolupar descripcions complexes de forma estricta pel que fa al vocabulari i l'organització.
- Serveix per a fixar l'algorisme, quan el procés ha de seguir uns passos o càlculs determinats que estan clars en la fase de l'anàlisi.

- Els usuaris després d'una breu explicació poden ser capaços de llegir i entendre aquest tipus d'especificacions.

FER MENTRE Client codi siga el mateix en DC-Inf-Comanda Incrementar Factura import amb Comanda import **FI MENTRE**	**SI** Factura import > 100 € Aplicar Client Descompte 10% **FI SI**	**REPETEIX** Comanda import = Línia import + Comanda import (anterior) **FINS A** Línies s'acaben

Taula 3.9. Exemple d'expressions en llenguatge estructurat

Com a desavantatges, es pot dir que de vegades el nom de llenguatge estructurat pot fer que els usuaris se senten reticents a revisar el seu contingut; en aquest cas és convenient no donar al llenguatge utilitzat el nom de llenguatge estructurat sinó, per exemple, descripcions formals del funcionament del sistema. A vegades pot ocórrer que, si es dóna massa nivell de detall, es fixe l'algorisme abans d'hora, o no es deixe al programador la capacitat de decidir quina és la millor estratègia d'implementació.

En qualsevol cas, és la tècnica més convenient sempre que s'utilitze seguint una sèrie de regles:

- Restringir el llenguatge estructurat que descriu un determinat procés a una sola pàgina.
- No utilitzar més de tres nivells d'imbricació dels bucles, en aquest cas és millor utilitzar altres mètodes.
- Utilitzar sagnats per a les diferents estructures.

3.7.3. Pre/postcondicions

Aquesta tècnica permet descriure les funcions sense especificar l'algorisme que s'ha d'utilitzar. És útil quan l'usuari tendeix a expressar un determinat procés en funció d'un algorisme que ha utilitzat durant molt de temps, quan puguen existir molts algorismes distints per a desenvolupar la funció i quan l'enginyer informàtic desitja que siga el programador qui prenga la decisió sobre quin algorisme cal utilitzar.

Les precondicions indiquen les dades o entrades que han d'estar disponibles perquè s'active el procés. No tots el fluxos d'entrada són precondició per a portar a terme el procés. Poden indicar també la relació entre els diferents fluxos d'entrada a una bombolla. Per exemple, si és necessari que el valor d'un component del flux estiga entre determinats valors. Relacions entre fluxos i magatzems de dades, per exemple si una determinada entrada del flux s'ha de correspondre amb alguna dada del magatzem de dades com es mostra en la taula 3.10.

Les postcondicions indiquen què s'ha de proporcionar quan el procés finalitza. És a dir, quina eixida produeix el procés, les relacions entre els valors d'entrada i els valors d'eixida, entre els valors d'entrada i els magatzems on es dirigeixen els fluxos d'eixida i/o les modificacions que afecten

aquests magatzems d'eixida. S'han de redactar tantes precondicions i postcondicions com situacions diferents puguen donar-se per a portar a terme el procés. S'han d'incloure tant les situacions normals com les que puguen donar-se en cas d'error.

```
SI Client DNI (llegit del flux d'entrada) = Client DNI de la DC-Factures del magatzem Factures
        Proporcionar Import deutor
SINO
        Proporcionar missatge "Aquest client no té factures pendents"
FI SI
```

Taula 3.10. Exemple de precondició i postcondició

Exemple**:** un client inicia el procés de vendes i comunica que està donat d'alta com a client. Aleshores es busca el seu número de client al magatzem de clients. Si la informació del client no està marcada com a cancel·lada llavors es carrega el material adquirit pel client al seu compte, amb el número de la compra i la quantitat. En un altre cas no es permet realitzar la compra.

- **Precondició 1:** el client té un número de client que es correspon amb el número del magatzem de CLIENTS i té un estat de client vàlid.
- **Postcondició 1:** es genera una factura amb el número de client i l'import de la venda.
- **Precondició 2:** la precondició 1 no es compleix (no existeix el número de client o l'estat del client no és vàlid).
- **Postcondició 2:** es produeix un missatge d'error.

En ocasions aquest enfocament pot no ser apropiat. No es detallen els passos intermedis entre les entrades i les eixides dels processos, les relacions complexes haurien d'especificar-se amb la utilització d'un llenguatge estructurat.

3.7.4. Taules de decisió

En ocasions els enfocaments detallats anteriorment no són útils per a descriure determinades situacions. Sobretot quan les informacions d'eixida depenen de decisions complexes en les quals intervenen massa condicions. Una taula de decisió permet representar totes les combinacions possibles de les condicions a tenir en compte per a portar a terme les accions.

Passos per a crear una taula de decisió:

- Identificar totes les condicions i tots els valors que aquestes poden prendre.
- Calcular el nombre de combinacions, si n'hi ha *n* condicions amb valors binaris són 2^n.
- Identificar totes les accions possibles.

- Crear una taula i situar totes les condicions i les accions en la vertical, i numerar les columnes amb tants números com possibles combinacions s'hagen calculat.
- Per a cada columna de combinacions de condicions indicar l'acció a portar a terme.
- Revisar amb l'usuari.

	1	2	3	4	5	6	7	8
Article en oferta	S	S	S	S	N	N	N	N
Facturació últims tres mesos > x €	S	S	N	N	S	S	N	N
Pagament al comptat	S	N	S	N	S	N	S	N
Descompte 10% a l'article	X	X						
Descompte 10% a la factura			X				X	
Descompte 20% a la factura	X				X			

Taula 3.11. Taula de decisió

Un exemple de taula de decisió es mostra en la taula 3.11, on es calcula el descompte que s'aplicarà a un client considerant combinacions de tres condicions, representades en la primera columna. En aquest cas les condicions d'entrada només tenen valors binaris però podria donar-se el cas que no fóra així.

Les taules de decisió són un mètode eficaç com a complement a l'especificació dels processos, desenvolupada amb altres mètodes (Pressman, 2010).

3.8. Documentació d'entitats externes

Finalment per a completar el diagrama de flux de dades cal donar una descripció o definició adequada sobre què representa cadascuna de les entitats externes. Aquesta definició ha de fer referència a les persones, departaments, o altres sistemes que es pretenen representar mitjançant les entitats externes. No es descriu què fan, sinó què són, com es mostra a la taula 3.12.

Llistat d'Entitats Externes			
Nom	Descripció	Fluxos d'entrada.	Fluxos d'eixida
Client	Persones o empreses que sol·liciten el subministrament de productes i estableixen una relació comercial amb el departament de vendes.	F5 Inf. articles F13 Factura	F1 Dades client F2 Dades comanda

Taula 3.12. Exemple de llistat d'entitats externes

Resum

L'anàlisi del sistema és una activitat que consisteix a transformar l'especificació dels requisits en models que representen les dades i les transformacions que realitza el sistema sobre aquests.

Una de les tècniques de modelització de processos més utilitzada en les metodologies estructurades són els diagrames de flux de dades. Els DFD representen les transformacions que sofreixen les dades, les dades que es transfereixen entre els processos del sistema, la informació que s'emmagatzema i els productors o receptors d'informació.

La representació gràfica de la funcionalitat del sistema per mitjà dels DFD ha de completar-se mitjançant la utilització de documents descriptius desenvolupats adequadament, que proporcionen informació sobre les dades, els magatzems, les especificacions dels processos i les entitats externes. Un DFD complet s'ha d'acompanyar dels següents llistats:

- Especificació de processos, preferiblement en llenguatge estructurat.
- Llistat d'entitats externes.
- Llistat de magatzems.
- Llistat de fluxos.
- Llistat de dades compostes.
- Llistat de dades elementals.

Activitats complementàries

1. Busqueu al diccionari les següents paraules: anàlisi, especificació, precís, concís, programari (*software*).
2. Identifiqueu en el cas del taller de reparacions de vehicles les possibles entitats externes. Doneu-les un nom i descriviu amb les vostres paraules què representa cada una.
3. Completeu convenientment la taula 3.1.
4. Confeccioneu 10 preguntes d'examen per a aquest tema.

Cas pràctic: Taller de reparació de vehicles

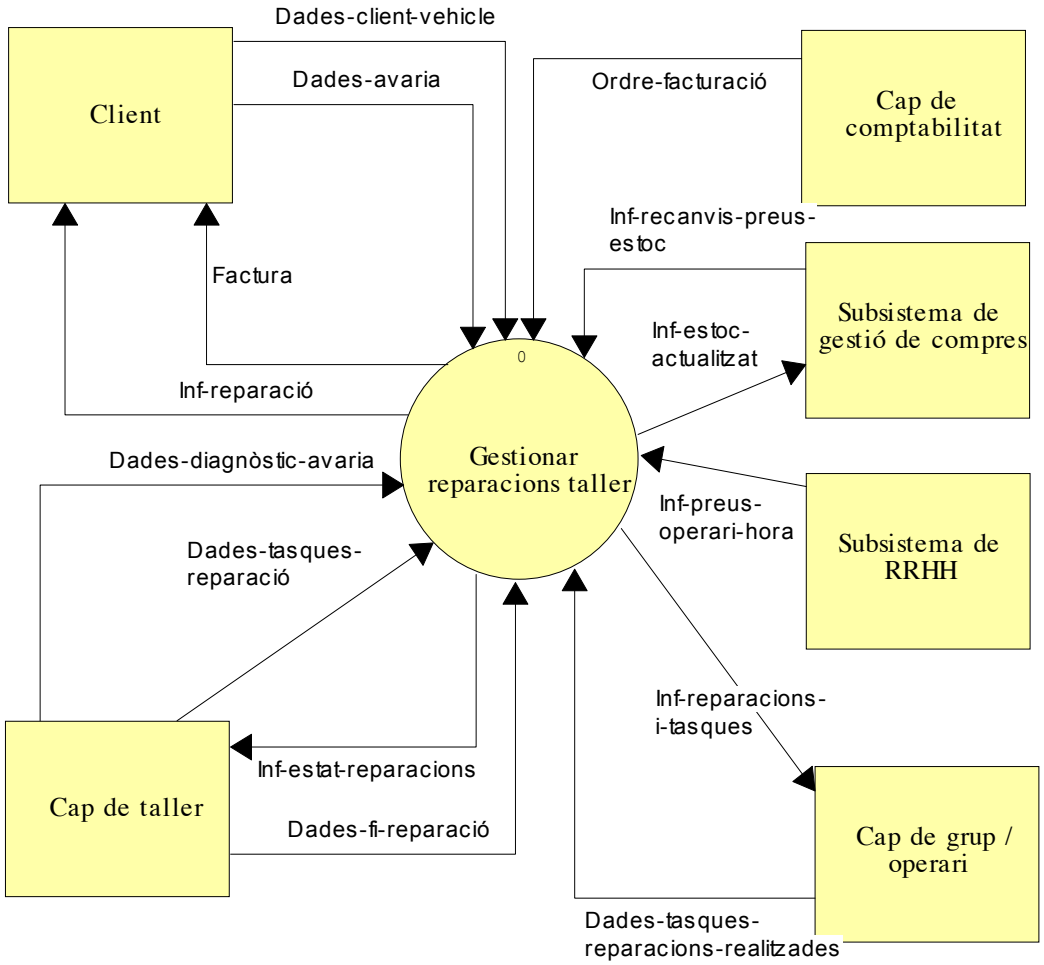
Procés de negoci	Funció informatitzada	Dades/inf. entrada	Qui/què (EE) D'on (magatzem)	Inf. eixida	A qui/què (EE) A on (magatzem)
Un client arriba al taller El cap de taller revisa el vehicle per a detectar l'avaria Es decideix reparar el vehicle	Buscar inf. client i vehicle	NIF/Nom client i/o matrícula Inf. clients i vehicles existents	Client Magatzems de clients i vehicles	Inf. client existeix	Client Cap de taller
	Donar d'alta nou client i vehicle	Dades nou client i vehicle	Client	Inf. nou client i nou vehicle	Magatzems de clients i vehicles
	Modificar dades client, modificar dades vehicle o donar d'alta vehicle	Inf. clients i vehicles existents. Dades client canvis Dades vehicle nou o canvis	Magatzems de clients i vehicles Client Client	Inf. nou vehicle o canvis	Magatzems de clients i vehicles
S'inicia la reparació (podria ser a continuació del procés anterior)	Introduir tasques a realitzar i dades inici reparació	Dades avaria Dades reparació Inf. vehicle	Client (*) Cap de taller Mag. clients i vehicles	Inf. nova reparació Full reparacions imprès	Mag. reparacions Treballadors (Client)
Els treballadors reparen el vehicle	Registrar tasques de fulls de reparació	Dades full reparació Inf. reparació Inf. RRHH Inf. recanvis	Cap de grup Mag. reparacions Mag. clients i vehicles. SS RRHH (1) SS compres (1)	Inf. reparació actualitzada	Mag. reparacions

Procés de negoci	Funció informatitzada	Dades/inf. entrada	Qui/què (EE) D'on (magatzem)	Inf. eixida	A qui/què (EE) A on (magatzem)
El cap de taller consulta l'estat de les reparacions	Consultar estat de reparacions	Dades de recerca Inf. client i vehicles Inf. reparacions	Cap de taller Mag. clients i vehicles Mag. reparacions	Inf. reparacions	Cap de taller
S'acaba la reparació	Registrar fi de reparació	Dades de la reparació (matricula/data fi) Inf. reparació i vehicles	Cap de taller Mag. reparacions Mag. clients i vehicles	Inf. reparació finalitzada	Mag. reparacions
El client arreplega el vehicle i abona la factura	Registrar data d'eixida i facturar	Inf. reparació client i vehicle Preus hora Preus recanvis	Mag. reparacions Mag. clients i vehicles SS RRHH SS compres	Inf. reparació facturada Inf. factura (i pagada?) Factura impresa	Mag. reparacions Mag. factures Client
El client arreplega el vehicle i no abona la factura	Registrar data d'eixida	Inf. reparació client i vehicles	Mag. reparacions Mag. clients i vehicles	Inf. reparació finalitzada	Mag. reparacions
A final de mes es facturen les reparacions no facturades	Inf. reparacions finalitzades no factures Inf. clients i vehicles	Inf. reparació Inf. client i vehicles Preus hora) Preus recanvis	Cap comptabilitat Mag. reparacions Mag. clients i vehicles SS RRHH SS compres	Inf. reparacions facturades Inf. noves factures Factures impreses	Mag. reparacions Mag. factures Clients

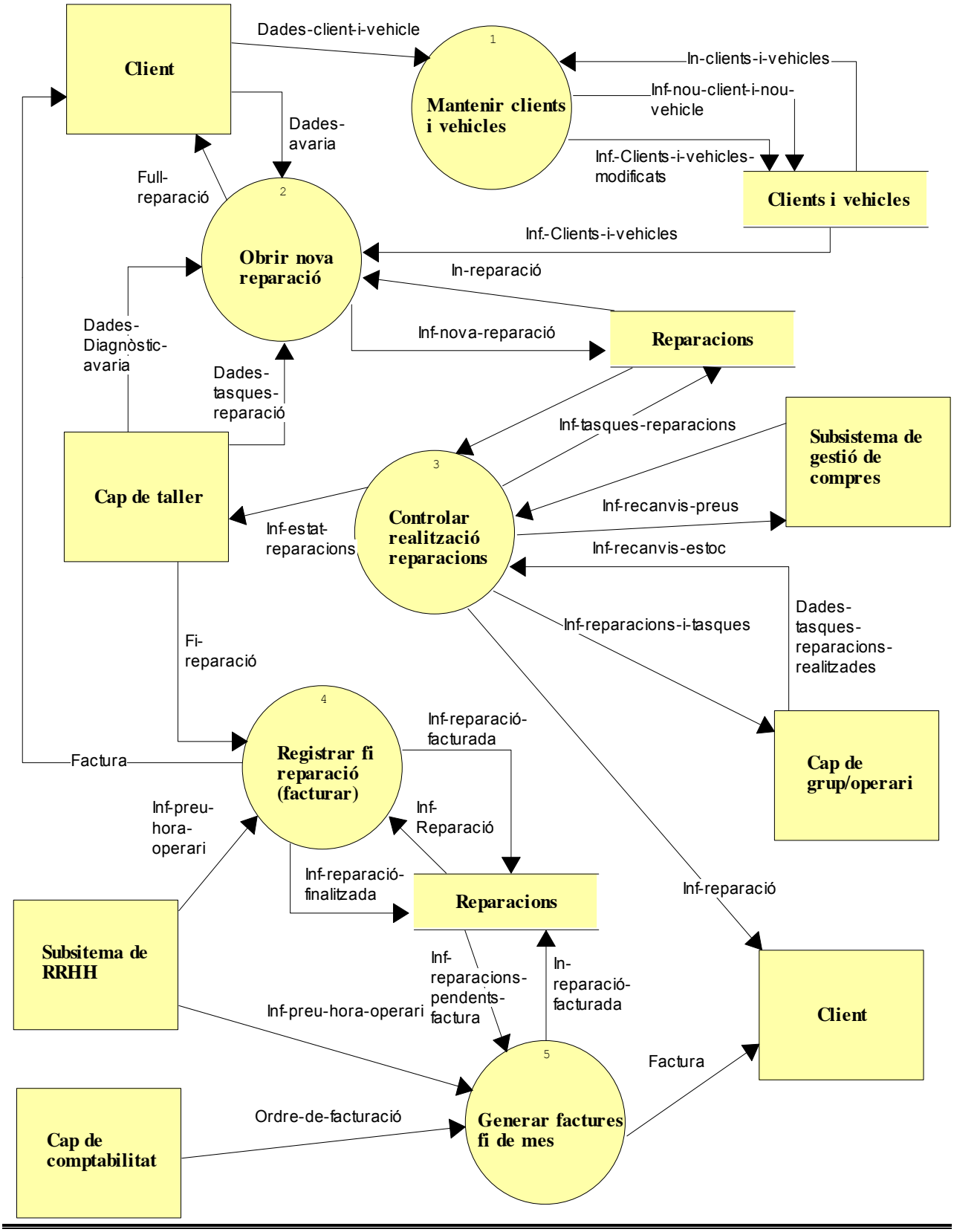
Diagrama de Flux de Dades

A continuació es proporcionen dos nivells del diagrama de flux de dades: el diagrama de context i la primera explosió desenvolupats amb l'eina CASE *Visible Analyst*.

Diagrama de context



Primera explosió



Capítol 4

Anàlisi. Modelització de dades

Objectius

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

- Comprendre la necessitat de representar l'estructura de la informació i les seues relacions per a completar l'anàlisi d'un sistema informàtic.
- Conèixer quins són els components i la notació del model conceptual de dades.
- Saber desenvolupar un model conceptual de dades i aplicar unes regles bàsiques per a obtenir un model sense redundàncies i amb detalls, que permeten reflectir millor els requisits d'usuari.
- Aprendre a documentar un model adequadament.
- Construir models de dades i de processos consistents per al mateix sistema i els mateixos requisits.

4.1. Introducció

El model conceptual de dades (MCD) és una tècnica que permet representar la informació lògica d'un sistema, la seua estructura i les regles necessàries, que serveixen d'informació d'entrada per al disseny de les bases de dades. L'objectiu principal és identificar i representar la informació independentment de com s'hi accedirà i de com estarà informatitzada físicament. El MCD facilita el posterior disseny de les bases de dades.

Per tant, es pot definir el model conceptual de dades com un esquema o descripció d'alt nivell de les dades d'un sistema, el seu contingut i estructura independentment de la implementació posterior de la base de dades.

4.1.1. Consideracions per al desenvolupament del model conceptual

El MCD es desenvolupa a partir dels requisits de dades i de les característiques funcionals 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 del sistema?
- Quins **detalls** caracteritzen aquests objectes i aquesta informació?
- Com estan **relacionats** aquests objectes i aquesta informació?

Per a desenvolupar el MCD del sistema, s'ha de seguir una determinada metodologia, adaptar cada pas a les necessitats de l'usuari i a les característiques particulars del sistema que s'ha de desenvolupar. És necessari considerar diversos factors crítics que, tractats convenientment, proporcionen resultats millors:

- Treballar **interactivament** amb els usuaris. Els usuaris poden comprendre fàcilment els conceptes i representacions utilitzats en el MCD, i poden col·laborar i valorar el model per a revisar requisits i validar la correcció de l'anàlisi del sistema.
- Seguir una **metodologia** senzilla, on s'indiquen els passos que s'han de seguir i les tècniques i ferramentes que es volen utilitzar.
- Considerar tant **l'estructura** de la informació com la seua **integritat**. Aquests dos punts de vista han d'estar presents per a obtenir finalment una correcta base de dades.
- Verificar que el model sobre el paper representa completament la informació que utilitza el sistema, és consistent, lògic i només té les redundàncies indispensables.

- Utilitzar **diagrames gràfics** per a representar el MCD sempre que siga possible.
- Construir un **diccionari de dades** per a completar la informació del model.

4.1.2. Beneficis del Model Conceptual de Dades

El MCD és el primer pas per al disseny de la base de dades. Per tant, del correcte desenvolupament d'aquest depèn l'obtenció d'una base de dades correcta que cobrisca les necessitats dels usuaris, sense inconsistències, amb les mínimes redundàncies possibles i efectiva. A més a més, es poden identificar altres beneficis del desenvolupament correcte del MCD que es resumeixen en els punts següents:

- Permet valorar quina és la tecnologia òptima per a desenvolupar la base de dades del sistema. En el MCD es representa la informació, les estructures i les relacions entre aquestes sense tenir en compte cap impediment tècnic. Açò permet identificar quina és la tecnologia que suporta tota aquesta estructura i s'adapta millor a les necessitats del sistema.
- De la mateixa manera, permet valorar paquets de mercat i sistemes de gestió de bases de dades comercials que s'adapten als requisits de l'usuari i del sistema.
- Permet reconèixer i identificar possibles canvis que en el futur poden afectar el sistema i tenir-les en compte durant aquest procés d'anàlisi.
- Permet que els usuaris comprenguen quines seran les seues dades en el sistema final, sense estar implementat.

4.2. Components del Model Conceptual de Dades (MCD)

Per a desenvolupar el MCD, és necessari desenvolupar un diagrama gràfic, on es representen els conjunts de dades i les relacions entre aquestes. A més, ha de desenvolupar-se un document on es descriguen de forma explícita els objectes representats per mitjà dels símbols gràfics, els seus continguts i altres aspectes que no poden representar-se gràficament de forma clara, o que poden ser necessaris per a comprendre millor el model. Els components principals del diagrama del MCD són les entitats de dades, els atributs i les relacions.

4.2.1. Entitats de dades

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. Generalment, s'assignen noms en singular i es dibuixen amb un rectangle. Per exemple, en un sistema

on es processa informació relacionada amb l'activitat comercial d'una empresa, l'article és, normalment, una entitat i pot representar-se com es mostra en la figura 4.1.

Figura 4.1. Representació gràfica de les entitats

Per a cada entitat de dades, han de poder-se identificar ocurrències que fan referència a cadascun dels objectes individuals o elements que poden considerar-se en una entitat. Per exemple, l'entitat article pot tenir 1.000 ocurrències, depenent del nombre d'articles que es processen en el sistema.

4.2.2. Atributs

Els atributs són les característiques que és necessari definir en cada entitat de dades per a representar la informació del sistema. Els atributs són unitats d'informació relacionades amb una entitat que, des d'un punt de vista lògic, no poden descompondre's més. Per exemple, el preu de venda és un atribut de l'entitat de dades *article*.

En una primera aproximació del desenvolupament del model, no és necessari detallar tots els atributs d'una entitat, però és necessari verificar abans de finalitzar el disseny que estan tots detallats i definits. Els atributs han de complir determinades regles que proporcionen al model consistència i eviten redundàncies:

- Un atribut ha de pertànyer a una entitat i només a una.
- Els atributs han de poder prendre valors per a cada ocurrència d'una entitat, ja que són les característiques que es desitja valorar de cada ocurrència, han de tenir valors adequats que permeten diferenciar unes ocurrències d'altres.
- Cada atribut ha de tenir un significat únic i consistent, és a dir, s'ha d'assignar un nom diferent a cada atribut dins de l'entitat. Açò no és necessari per a diferents entitats, ja que cada entitat qualifica els seus atributs. Per exemple, nom de l'entitat *assignatura* i nom de l'entitat *alumne*.

4.2.3. Identificadors

L'identificador d'una entitat és un atribut o un conjunt mínim d'atributs que prenen un valor únic per a cada ocurrència de l'entitat. Quan es considera un conjunt d'atributs, el valor únic està determinat per la combinació dels valors d'aquests. Per tant, els identificadors permeten diferenciar unes ocurrències d'altres. Per a definir identificadors d'una entitat, han de tenir-se en compte:

- Per a cada ocurrència de l'entitat, l'identificador ha de tenir un **valor únic**.
- Els atributs que formen els identificadors **no poden prendre valors nuls** i no han de ser dades que varien de valor per a una ocurrència donada.
- És aconsellable que siga de curta longitud, d'ús comú i de fàcil memorització.
- Ha de definir-se, almenys, un identificador per a cada entitat.

Poden existir diversos atributs o conjunts d'atributs que són candidats a ser considerats identificadors. En la notació utilitzada es representa l'atribut o atributs que formen la clau primària en la part superior de l'entitat.

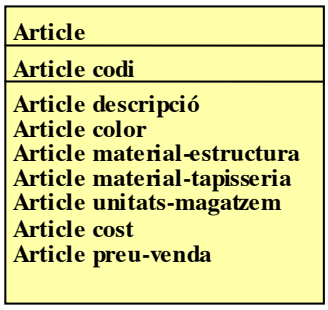


Figura 4.2. Exemple d'identificador per l'entitat article: article codi

4.2.4. Relacions

El MCD ha de representar les connexions existents entre els diferents conjunts de dades que utiltiza el sistema. Les relacions representen les connexions que són d'interès per al sistema entre les ocurrències de dues o més entitats. Aquestes associacions s'identifiquen a partir dels requisits funcionals i dels requisits de dades del sistema. Les relacions es representen, generalment, per mitjà de línies que connecten les entitats relacionades.

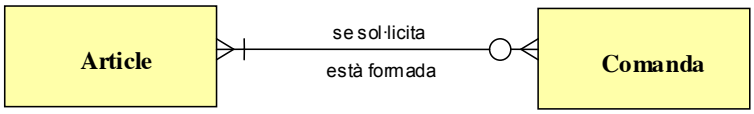


Figura 4.3. Exemple de representació de relació

A la figura 4.3 es mostra una possible representació gràfica de la connexió que existeix entre l'entitat *article* i l'entitat *comanda de client***.** Aquesta relació representa la connexió de cada article amb cada comanda en què se sol·licita l'esmentat article, i cadascuna de les comandes amb cadascun dels articles que són sol·licitats en aquella.

Cada relació s'ha de representar i considerar de forma adequada. S'ha d'avaluar quantes ocurrències d'una entitat poden estar relacionades amb una ocurrència d'una altra entitat, i si aquesta connexió és obligatòria o opcional. Aquest concepte rep el nom de cardinalitat o connectivitat.

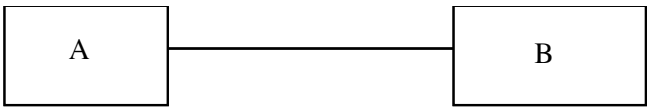


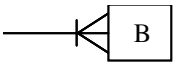
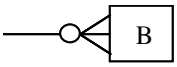
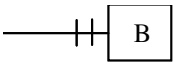
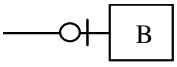
Figura 4.4. Entitats A i B i relació que les connecta

Donades dues entitats A i B, i una relació entre elles, com es mostra en la figura 4.4, per a definir la cardinalitat cal plantejar i contestar les qüestions que es mostren en la taula 4.1.

Qüestió	Resposta
1. Donada una ocurrència de l'entitat A, amb quantes ocurrències com a màxim de l'entitat B pot estar relacionada?	Només una
	Diverses, més d'una, moltes
2. Donada una ocurrència de l'entitat A, és obligatori que estiga connectada amb una ocurrència de l'entitat B?	Sí
	No
3. Donada una ocurrència de l'entitat B, amb quantes ocurrències com a màxim de l'entitat A pot estar relacionada?	Només una
	Diverses, més d'una, moltes
4. Donada una ocurrència de l'entitat B, és obligatori que estiga connectada amb una ocurrència de l'entitat A?	Sí
	No

Taula 4.1. Qüestions per a definir la cardinalitat i obligatorietat de les relacions en el MCD

Tenint en compte les possibles combinacions de respostes, hi ha quatre possibles connexions en cadascun dels extrems. En la taula 4.2 es mostra la representació per a l'extrem de l'entitat B, les mateixes combinacions i representacions s'haurien de considerar per l'extrem de l'entitat A.

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, però també pot estar connectada amb moltes.	
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 màxim només pot estar connectada amb una.	

Taula 4.2. Representació de les combinacions de cardinalitat i obligatorietat per a l'entitat B

Tradicionalment, quan s'anomena una relació es té en compte només la cardinalitat màxima, que és el que es considera a les qüestions 1 i 3 de la taula 4.1, les respostes a aquestes dues preguntes donen relacions que es denominen del tipus *un a un*, si les dues respostes són una, un a molts, si una resposta és una i l'altra moltes, i *molts a molts* si ambdues respostes són moltes, com es mostra en la taula de decisió 4.3.

En la taula 4.3. es mostra una taula de decisió on es resumeixen aquestes combinacions.

Qüestió				
Una ocurrència d'A amb quantes com a màxim de B	Una	Una	Moltes	Moltes
Una ocurrència de B amb quantes com a màxim d'A	Una	Moltes	Una	Moltes
Tipus de relació	Un a un	Un a molts	Un a molts	Molts a molts

Taula 4.3. Taula de decisió de les cardinalitats d'una relació

Exemples

A continuació es mostren exemples concrets dels tres tipus de relacions que es defineixen en la taula 4.3:

- **Un a molts:** un client pot haver fet moltes comandes (màxim n), encara que no necessita tenir una comanda per a ser considerat una ocurrència de l'entitat client (no obligatorietat), una comanda només pot pertànyer a un client (màxim un) i necessita que existisca un client al qual pertany (obligatorietat) per a tenir sentit com a ocurrència (figura 4.5).

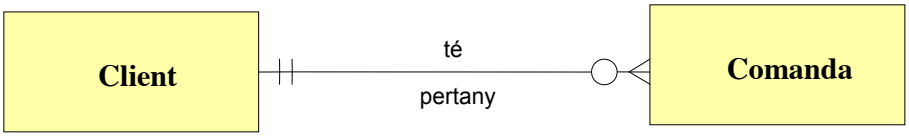


Figura 4.5. Exemple de relació *un a molts*

- **Un a un:** un grup d'estudiants pot tenir un delegat o cap (màxim un, no obligatori), i un delegat sempre és d'una classe i només d'una (màxim un i obligatori) (figura 4.6).

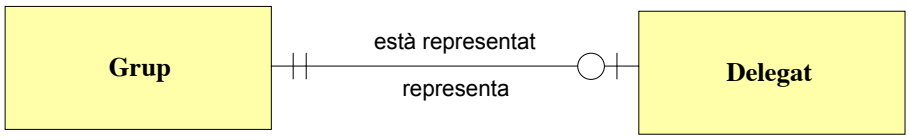


Figura 4.6. Exemple de relació *un a un*

- **Molts a molts**: Un alumne pot estar o haver estat matriculat en moltes assignatures (màxim moltes) i almenys ha estat matriculat en una per ser alumne (obligatorietat) i una assignatura pot tenir molts alumnes matriculats (màxim n), però no necessita tenir cap per a ser assignatura (assignatures noves per exemple, no obligatorietat) (figura 4.7).

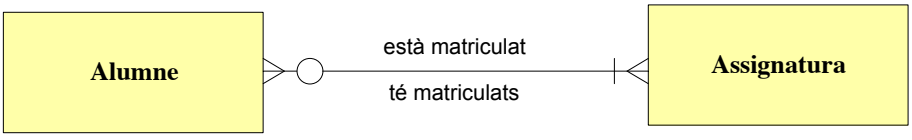


Figura 4.7. Exemple de relació *molts a molts*.

4.3. Documentació del model

El model gràfic per ell mateix només aporta una visió aproximada de les dades i relacions que s'han de tenir en compte per al desenvolupament posterior de la base de dades. No obstant, aquesta informació no és suficient. És necessari proporcionar informació addicional que permeta comprendre en detall què representa cadascuna de les entitats, les relacions i els seus atributs, les seues característiques i peculiaritats que de forma gràfica no poden representar-se.

Hi ha nombroses ferramentes CASE que donen suport al desenvolupament del model i a la generació d'aquesta documentació associada. Aquestes ferramentes utilitzen notacions pròpies i, en general, posseeixen generadors d'informes que faciliten de manera àmplia la documentació del model. En qualsevol cas, tant si el model es desenvolupa amb el suport d'una d'aquestes ferramentes o d'altres, com poden ser processadors de textos o ferramentes de desenvolupament de gràfics, la documentació del model ha de tenir uns continguts mínims que es descriuen a continuació.

4.3.1. Documentació d'entitats i atributs

Cada entitat ha de tenir associada una descripció. Aquesta descripció és fonamental per a comprendre el model. Un mateix nom d'una entitat pot donar lloc a diferents conceptes i pot fer que les relacions siguen correctes o no. Descriure aquests conceptes és una tasca a la qual ha de prestar-se una especial atenció i en la qual la participació d'usuaris i analistes és fonamental.

Exemple: dues definicions diferents per l'entitat *article*, segons el negoci al qual es dedica l'empresa objecte d'anàlisi:

1. *Cadascun dels objectes que l'empresa fabrica per a la seua comercialització. Els articles es fabriquen en cadena i es produeixen nombroses unitats d'un mateix article. Per tant el mateix article pot comercialitzar-se moltes vegades i a molts clients.*
2. *Cadascun dels objectes que l'empresa fabrica per a la seua comercialització. Cada producció d'un article és única, per tant d'un determinat article només es fabrica una unitat. Un article només pot vendre's una vegada i a un client.*

Com a exercici pot estudiar-se com, depenent d'aquestes definicions, varien les relacions entre *article* i altres entitats del model.

A més a més, per a cada entitat han de proporcionar-se possibles àlies, és a dir, els diferents noms amb què es pot fer referència al mateix concepte dins de l'organització. Els identificadors poden indicar-se en el model de forma gràfica, però, també és convenient assenyalar en la documentació si un atribut

forma part o no d'un identificador. En l'exemple que es mostra a continuació, es proposa una columna tipus per als atributs on s'indica aquesta característica.

Per a documentar els atributs s'indica el nom, l'àlies, el domini (tipus de dada i rang), i la descripció. En molts casos, la descripció d'un atribut, com per exemple *Client número-telèfon*, és òbvia i no és necessari especificar-la i en altres, és de gran importància per a completar el model.

Com a exemple del document que es vol obtenir, es mostra a la taula 4.4 una fitxa que s'ha d'emplenar per a cada entitat. Si es disposa de l'ajuda d'una ferramenta CASE la generació de la documentació depèn de la ferramenta.

Entitat: Article			
Descripció: Cadascun dels objectes que l'empresa fabrica per a la seua comercialització. Els articles es fabriquen en cadena i es produeixen d'un mateix article nombroses unitats.			
Àlies: Producte		**Ocurrències**: *Màx i Min*	
Atributs			
Nom	Tipus	Descripció	Domini
Codi	CP	Nombre seqüencial.	Sencer
Descripció	Ca		Cadena (30)
Unitat		Unitat de mesura en què es comercialitza o emmagatzema el producte. Pot ser un atribut codificat.	Caràcter
Preu-venda			Real
Estoc		Quantitat actual emmagatzemada de l'article	Real
...		...	

Taula 4.4. Fitxa de l'entitat article

4.3.2. Documentació de relacions

Igual que en el cas de les entitats, cada relació ha de tenir associada una descripció adequada que proporcione una major comprensió sobre el model. A banda de les entitats que connecta s'ha d'indicar la cardinalitat o connectivitat de la relació i l'obligatorietat. Les relacions poden tenir atributs, encara que, generalment, en aquest cas sol ser convenient reconvertir aquestes relacions en entitats. A la taula 4.5 es mostra un exemple de documentació.

Relació: Realitza					
Descripció: Indica cada comanda a quin client correspon i connecta cada client amb totes les comandes que ha realitzat.					
Entitat 1	Obligatòria	Connec. màxima	Entitat 2	Obligatòria	Connec. màxima
Client	No	N	Comanda	Sí	1

Taula 4.5. Fitxa de la relació realitza

4.4. Passos per al desenvolupament del model conceptual de dades

La metodologia que es proposa en aquest text per a desenvolupar el model conceptual de dades consta dels següents passos:

1. Identificar les principals entitats.
2. Determinar les relacions entre entitats.
3. Definir identificadors.
4. Afegir atributs.
5. Determinar regles de funcionament o de negoci.

A mesura que es desenvolupa cadascun d'aquests passos, s'ha d'anar completant la documentació tant gràfica com descriptiva del model. És convenient realitzar revisions parcials i una revisió final del model, tant amb l'equip d'analistes com amb els d'usuaris.

4.4.1. Identificar les principals entitats

Les principals entitats han de ser objectes reals, objectes tangibles o conceptes abstractes, com s'ha indicat en la definició, ja que són d'interès per al sistema objecte d'estudi.

Encara que no hi ha regles que determinen de forma clara què ha de ser una entitat del model, sí que es poden proporcionar uns principis que permeten identificar els components d'un primer esborrany del model. A partir de la definició de requisits o descripció dels processos de negoci del sistema, els substantius, subjectes d'oracions o objectes directes, poden donar lloc a entitats o atributs. I amb l'anàlisi dels verbs es poden identificar relacions entre entitats. Altres complements, com els circumstancials, permeten també identificar atributs o relacions segons el cas, i les preposicions poden informar de relacions.

Exemple: *Els alumnes es matriculen de les assignatures al setembre*.

- *Alumne* és una possible entitat.
- *Assignatura* pot ser un atribut o una entitat. Si únicament es desitja representar el nom de l'assignatura, sense afegir-ne altres característiques inicialment és un atribut (amb múltiples valors) de l'entitat *alumne*.
- *Setembre* és una dada que pot ser un atribut, si es considera necessari representar aquesta informació en el model, en aquest cas podria ser més correcte considerar la data de matrícula; o bé pot ser una dada de caràcter global del sistema i en aquest cas no cal representar-la.

A mesura que s'identifiquen les entitats s'han d'anomenar, definir i documentar al diccionari de dades o al document de disseny corresponent.

4.4.2. Determinar les relacions entre entitats

Les relacions són els fets que representen la connexió entre les ocurrències de dues entitats. Com s'ha indicat en l'apartat anterior poden identificar-se a partir de verbs, accions i preposicions. Poden classificar-se en les categories següents:

- Existents o de possessió (per exemple un treballador té fills, els fills del treballador).
- Funcionals (el professor explica als alumnes).
- Esdeveniments (el client realitza comandes).

S'ha d'assignar a les relacions noms curts i amb significat adequat en cada cas. Per això, és convenient fer una llista inicial, representar-les en el diagrama i analitzar el significat de cadascuna.

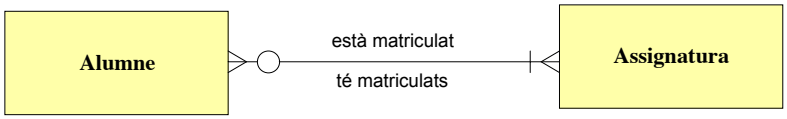


Figura 4.8. Exemple de nom de relació

En el cas de les relacions, el nom pot proporcionar ambigüitat si s'intenta llegir la relació en dos sentits. Per exemple, quan s'utilitza «es matricula» o «estan matriculats», com a nom d'una relació que connecta l'entitat *alumne* i l'entitat *assignatura* (veure la figura 4.8); la relació representa el fet que "*en una assignatura estan matriculats alumnes*" En alguns textos es recomana anomenar les relacions únicament amb la indicació del nom de les entitats que connecten, com ara alumne-assignatura, o bé donar dos noms i dividir la relació en dos trams o arcs.

4.4.3. Afegir atributs i definir identificadors

En aquest pas es comença a afegir al model detalls que proporcionen una comprensió millor sobre el model. A mida que es van definint atributs es pot seleccionar quins són bons candidats a ser identificadors.

Per a definir els identificadors cal considerar atributs o conjunts d'atributs que permeten distingir cadascuna de les ocurrències d'una entitat. Els identificadors i atributs ajuden a completar el model i a comprendre millor si aquest es desenvolupa de forma adequada.

La presència d'un identificador assegura que es pot fer referència a una ocurrència determinada d'una entitat i permet desenvolupar posteriorment mecanismes d'accés a les dades de la base de dades.

Normalment els identificadors són conceptes d'ús comú entre els usuaris, ells estan acostumats a utilitzar dades com número de factura o codi del proveïdor, per a identificar els objectes o documents que utilitzen diàriament. Per tant, en la majoria dels casos s'han de tenir en compte els mecanismes que els usuaris utilitzen per a identificar cadascuna de les entitats definides en el model conceptual, i analitzar-los per a definir quins són els identificadors adequats per a cadascuna d'aquestes.

Els identificadors s'han d'anomenar i documentar de forma adequada al model i al diccionari de dades associats. És convenient desenvolupar estàndards per anomenar tant els identificadors com la resta dels atributs de les entitats, i que els noms siguen curts i consistents (figura 4.9).

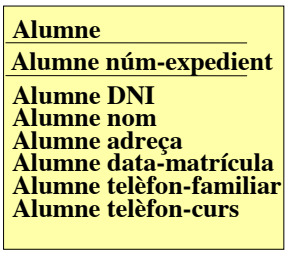


Figura 4.9. L'entitat alumne amb el seu identificador i atributs

És necessari anomenar i documentar els atributs al model de dades i al diccionari de dades. No cal representar els atributs que s'obtinguen a partir d'altres per mitjà de càlculs, però si es fa s'ha de documentar de forma adequada. De vegades una característica de les dades que apareixen en el model pot estar relacionada amb diverses entitats. Els atributs s'han d'assignar a l'entitat en la qual el seu valor es pot determinar pel valor del identificador complet, i sempre, assignar l'atribut a l'entitat en la qual el seu valor és menys repetitiu.

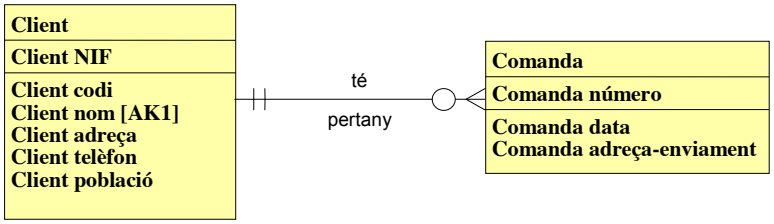


Figura 4.10. Definició d'atributs

Exemple: donat un model en què es defineix una entitat *client* i una entitat *comanda* associada per la relació un client sol·licita comandes. S'identifica la necessitat de definir un atribut *població* del client, i el dubte és ubicar-lo en l'entitat *client* o a l'entitat *comanda*. En aquest cas és un atribut que ha de definir-se en l'entitat *client,* perquè si es defineix en l'entitat *comanda* es repetiria el seu valor tantes

vegades com comandes pertanyen al mateix client (veure la figura 4.10), i perquè l'identificador definit com el NIF/CIF d'un client identifica de forma única el valor de la població del client.

4.4.4. Altres consideracions

Altres consideracions que és necessari tenir en compte i analitzar amb detall per a realitzar un correcte MCD són:

- **Atributs que poden ser entitats**: si un atribut és susceptible de dependre d'altres característiques potser siga necessari definir-lo com una entitat i refer les relacions.
- **Atributs que prenen valors múltiples**: inicialment només és necessari considerar-los i reconèixer-los, posteriorment quan s'apliquen regles de normalització en el cas del model E/R es reconverteixen en entitats. De vegades, és convenient realitzar aquesta conversió a entitats al MCD, per a comprendre millor el seu significat i estudiar detalls sobre el model.
- **Atributs que descriuen relacions**: de vegades s'identifiquen atributs que pel seu significat no es corresponen amb cap entitat sinó que estan associats a una relació entre dues entitats.
- **Atributs codificats**: s'ha evitar la definició d'aquests atributs, perquè impliquen la creació d'una nova entitat i relacions que resten llegibilitat i comprensió al model.
- **Atributs indicadors de processos**: no s'han d'incloure en el model atributs per a guardar indicadors de processos. Quan es dissenya un model de dades lògiques s'ha de representar únicament la informació i les seues propietats inherents. Hi ha algunes dades que encara que estan relacionades amb la realització d'un procés, o poden parèixer diferents estats, són necessàries si aporten informació rellevant de les entitats del model. Per exemple, en l'entitat comanda pot ser necessari conèixer si aquesta s'ha servit o no al client, per tant ha d'haver un procés que actualitze les comandes i permeta actualitzar aquesta informació. En aquest cas el més convenient seria indicar la data en què s'ha servit com a atribut.

4.4.5. Definir regles de funcionament o de negoci

Els passos descrits en els apartats anteriors permeten definir entitats, relacions i alguns dels seus atributs. En aquest pas s'afegeixen al model detalls sobre el funcionament de les dades, que no poden assenyalar-se de forma clara en el gràfic i que són importants, perquè permeten definir regles sobre la integritat de la informació, assegurar la consistència i correcció dels valors de les dades i establir com els requisits dels usuaris afecten certs aspectes del model.

En aquest apartat s'estudien i defineixen regles relacionades amb el comportament del model respecte a la inclusió de noves ocurrències o l'eliminació d'aquestes i les relacions. Les regles que imposen els

models per a mantenir la integritat i la consistència són aspectes que s'estudien de forma àmplia i detallada en els textos de bases de dades. Per tant, l'interès se centra en aquells aspectes que estan relacionats amb els requisits dels usuaris i el sentit del concepte que representa cada entitat i cada relació.

Aquestes regles proporcionen una major comprensió del model i aporten informació de gran ajuda per al disseny posterior de la base de dades i dels processos del sistema.

Per a cada relació han de tenir-se en compte dues qüestions i per tant definir dues regles, una d'inserir i una d'esborrar. La primera permet establir la solució quan el problema és inserir una ocurrència en una entitat que necessita estar connectada amb una ocurrència d'una altra entitat. La segona permet detectar els problemes d'esborrar una certa ocurrència quan aquesta té ocurrències d'una altra entitat relacionades o dependents. Les respostes a aquests plantejaments han de donar-se tenint en compte el model i el funcionament de l'empresa. De vegades l'anàlisi d'aquestes qüestions fa necessari replantejar alguns aspectes del model, com per exemple eliminar l'obligatorietat en la relació.

Inserir noves ocurrències

En aquest apartat s'estudien i determinen les condicions sota les quals es pot inserir una ocurrència d'una entitat B, que ha d'estar relacionada amb ocurrències d'una altra entitat A. El problema es planteja si la relació indica que cada ocurrència de B necessita una ocurrència d'A, per a tenir sentit, és a dir, hi ha obligatorietat per a les ocurrències de l'entitat B respecte d'aquesta relació com es mostra en la figura 4.11.

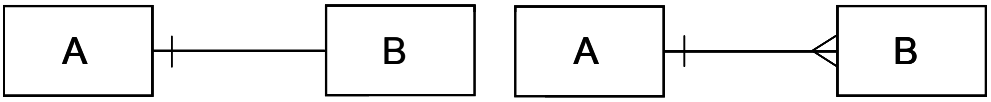


Figura 4.11. Relació *un a un*, i relació *un a molts*

Respostes possibles són:

1. Només es pot inserir una nova ocurrència d'una entitat B si hi ha ocurrència de l'entitat A amb la qual relacionar-la.
2. Sempre es permet la inserció d'una ocurrència de l'entitat B, si no hi ha ocurrència en l'entitat A relacionada, se'n crea una.
3. Encara que no existisca ocurrència de l'entitat A relacionada, es pot inserir una ocurrència de l'entitat B, donant de forma automàtica un valor per defecte a l'ocurrència d'A necessària.
4. Sempre es permet la inserció d'una ocurrència en l'entitat B sense tenir en compte la connexió amb les ocurrències de l'entitat A.

Les respostes 1 i 2 impliquen establir un control i definir processos adequats en el DFD per a la seua posterior implementació. Les respostes 3 i 4 obligarien a replantejar-se l'obligatorietat de la relació, perquè sinó, no són correctes.

De vegades, els requisits o regles dels usuaris no s'adapten a cap de les definides anteriorment. En qualsevol cas, s'ha de proporcionar una descripció correcta i completa d'acord amb els requisits dels usuaris i a les característiques de les entitats relacionades.

Esborrar ocurrències

Les regles d'esborrar determinen en quines condicions és vàlid eliminar una ocurrència d'una entitat A que té ocurrències d'una altra entitat B, que depenen de la primera respecte de la relació que s'estudia.

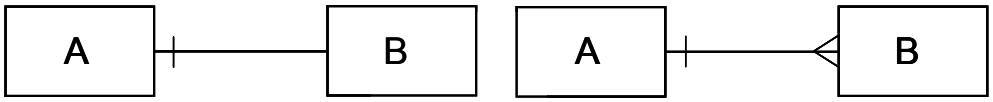


Figura 4.12. Relació *un a un,* i relació *un a molts*

De la mateixa forma que en el cas anterior, les respostes poden ser:

1. Només es poden esborrar ocurrències de l'entitat A si aquestes no tenen cap ocurrència de l'entitat B relacionada.
2. Si s'elimina una ocurrència de l'entitat A, totes les ocurrències de l'entitat B relacionades també s'eliminen.
3. Si s'esborra una ocurrència de l'entitat Atotes les ocurrències de l'entitat B relacionades s'han de connectar a una ocurrència definida per defecte.
4. En qualsevol cas es poden esborrar ocurrències de l'entitat A i no es realitza cap acció respecte a les ocurrències de B relacionades.

Com en el cas d'inserir, les respostes 1 i 2 impliquen establir un control i definir processos adequats en el DFD per a la seua posterior implementació. Les respostes 3 i 4 no són correctes si hi ha obligatorietat, en aquest cas, cal replantejar-se la relació.

4.5. Consistència entre el Diagrama de Flux de Dades i el Model Conceptual de Dades

Com s'ha estudiat en el tema 3, el diagrama de flux de dades representa les dades, la informació, les transformacions que sofreixen les dades, els receptors i generadors d'aquestes dades i aquesta informació i els magatzems de dades del sistema. En aquest tema s'ha estudiat el model conceptual de dades que permet representar la informació, la seua estructura i les relacions que el sistema ha de

mantenir per a complir els requisits dels usuaris. Tot això fa pensar que ha d'existir una determinada concordança entre el MCD i el DFD desenvolupat per al mateix sistema, que faça que aquests dos models siguen consistents i no proporcionen contradiccions. Per a verificar aquesta consistència han de complir-se les següents condicions:

1. Cada magatzem del DFD s'ha de correspondre amb una entitat, una relació o una combinació d'aquests dos del MCD.
2. En el cas d'utilitzar un únic diccionari per a aquests dos models, les entrades han de coincidir, és a dir, les dades elementals del diccionari del DFD s'han de correspondre amb les mateixes dades del MCD**,** els atributs.
3. Hi ha d'haver processos en el DFD per a crear i eliminar ocurrències de cadascuna de les entitats del MCD, menys com a exempció si representen informació externa al sistema.
4. Hi ha d'haver almenys una bombolla o procés del DFD per a assignar valors a cadascun dels atributs de cadascuna de les entitats del MCD, i hi ha d'haver algun procés que els llegesca o utilitze.

Resum

El model conceptual de dades és una tècnica que permet identificar la informació, el seu contingut i les relacions entre els diferents grups de dades, que s'han d'emmagatzemar en el sistema per tal de cobrir els requisits del usuaris. És el punt de partida per tal de desenvolupar una base de dades correcta i sense redundàncies.

En aquest tema s'estudia com es pot desenvolupar un model de dades independentment de la seua implementació física com a base de dades, encara que es tenen en compte certs conceptes propis de les bases de dades relacionals.

El model que s'obté ha de ser consistent amb el model de processos que es desenvolupe per al mateix sistema.

Activitats complementàries

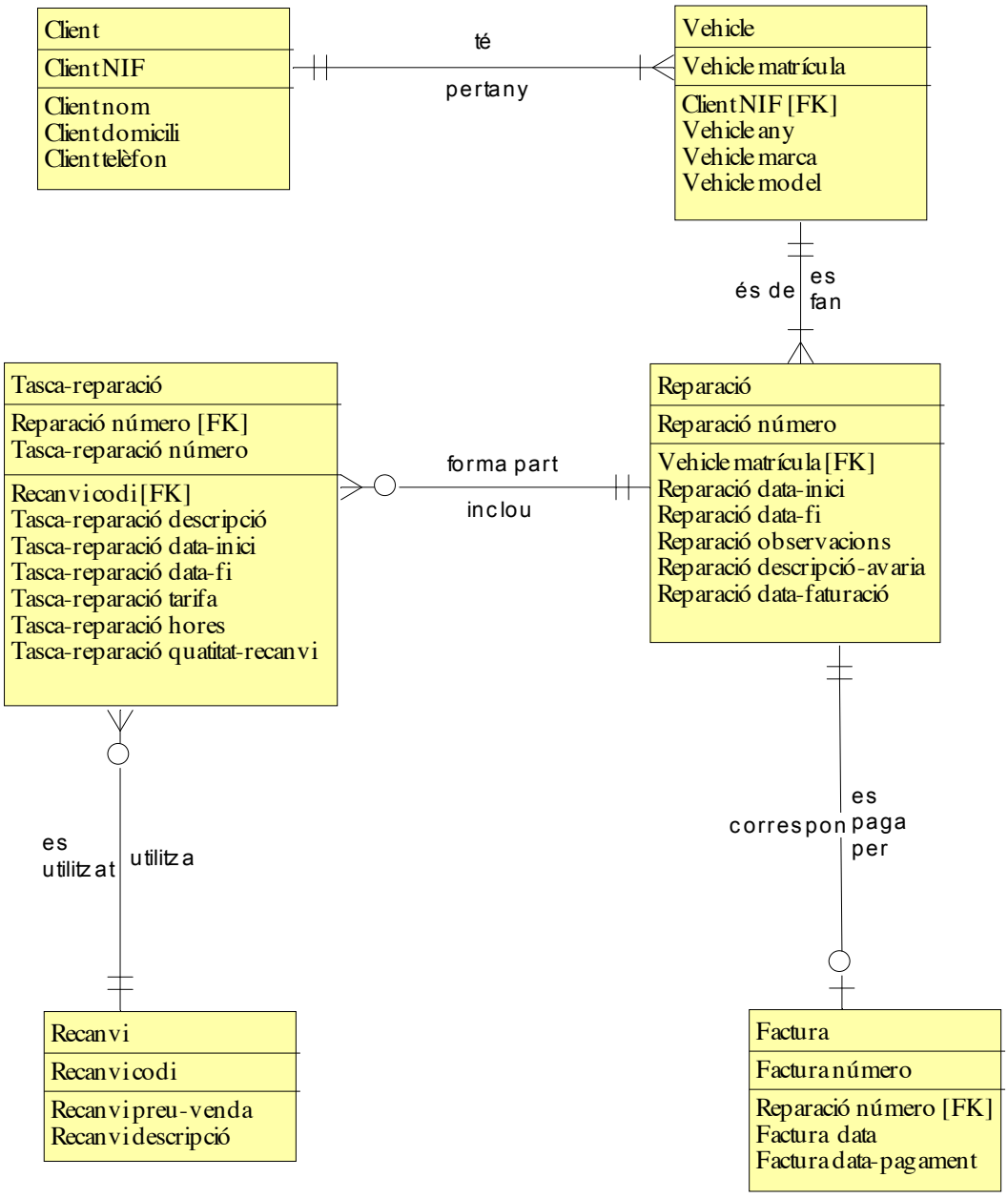
1. Busqueu al diccionari els següents conceptes: instància, ocurrència, relació, entitat, atribut, identificador, àlies.
2. Representeu totes les relacions que apareixen en el tema amb la utilització de la notació de la ferramenta CASE que s'usa en les sessions de pràctiques.
3. Definiu dues regles de negoci (una d'esborrar i un d'inserir) per cadascuna de les relacions del

model conceptual de dades desenvolupat per al cas del taller de reparacions.

4. Feu un model conceptual de dades amb les entitats i relacions que intervenen en el procés automàtic de matrícula d'un alumne en aquesta universitat.
5. Confeccioneu deu preguntes d'examen per a aquest tema.

Cas pràctic: Taller de reparació de vehicles

MCD: A continuació es proporciona una primera versió del model conceptual de dades del cas pràctic del taller de reparacions desenvolupat amb l'eina *CASE Visible Analyst*.



Capítol 5

Disseny

Objectius

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

- Comprendre i assimilar quines són les principals activitats que s'han de dur a terme en la fase del disseny d'un sistema informàtic.
- Saber algunes de les condicions que cal tenir en compte perquè el disseny del sistema proporcione les bases per al posterior desenvolupament d'un programari correcte, eficaç, fiable, mantenible i útil per als usuaris i els propòsits marcats a l'inici com a objectius del sistema.
- Aprendre algunes de les principals consideracions per a dissenyar unes interfícies d'usuari adequades, en particular pantalles i informes.
- Conèixer que hi ha tècniques que permeten modelitzar els mòduls de programari que s'han d'implementar en la següent fase de construcció del programari.

5.1. Introducció

L'anàlisi d'un sistema informàtic defineix les relacions lògiques o conceptuals entre els components del sistema. Es defineixen les dades d'entrada necessàries, la informació d'eixida, els processos que s'han de dur a terme per tal de satisfer els requisits i quines restriccions s'han de tenir en compte,

independentment de com s'han de realitzar físicament. El disseny comporta l'especificació física dels processos, com i quin format ha de tenir físicament la informació d'eixida, de quina forma s'han d'introduir les dades que necessiten els processos dissenyats, sobre quins equips s'han d'implementar aquests processos, l'emmagatzematge de les dades i els formats exactes dels informes.

És a dir, l'anàlisi se centra en què ha de realitzar el sistema, mentre que el disseny es preocupa per la manera com s'han de satisfer aquests processos. Durant la fase de disseny s'identifica, analitza i determina com les dades que s'han definit en la fase d'anàlisi s'han de processar i emmagatzemar, com i en quins formats s'ha d'obtenir la informació d'eixida, i com s'han d'implementar els processos i les restriccions identificades.

L'objectiu és obtenir el disseny d'un sistema que siga efectiu, fiable i mantenible; és a dir, que complisca els requisits analitzats, les especificacions i les restriccions, de manera correcta amb els recursos mínims necessaris, i proporcione mecanismes per a controlar els errors i la seguretat de la informació.

- L'efectivitat fa referència a l'ús del sistema. Un sistema que no s'utilitza no és efectiu. Per tant, una qüestió fonamental és que el sistema siga acceptat pels usuaris i que l'utilitzen i aprofiten al màxim la funcionalitat que el sistema els proporciona.
- La fiabilitat d'un sistema d'informació fa referència a com el sistema reacciona als errors, tant si són errors de dades d'entrada, processos, fallades de maquinari o errors humans. Mai no es pot assegurar que un sistema és completament fiable contra fallades o errors, una aproximació més real hauria de ser intentar construir un sistema amb mecanismes per a resoldre i prevenir els possibles errors i fallades de la forma més ràpida i amb el menor cost.
- Un sistema és mantenible si és senzill de modificar, flexible i simple. No importa com de bé s'ha analitzat i dissenyat, sempre que un sistema es posa en funcionament sorgeixen, prompte o tard, qüestions que cal modificar. Els sistemes han de ser prou flexibles perquè permeten introduir-los modificacions.

Per arribar a dissenyar un sistema amb aquestes característiques es poden tenir en compte les següents consideracions referents als usuaris finals, a les dades i als processos:

- Cal valorar les diferents opcions i considerar l'efecte que causaran en l'usuari final, i pensar que el sistema es dissenya per a ser utilitzat pels usuaris. Com més es facilite el treball dels usuaris, més i millor utilitzaran el sistema.

- Els processos d'introducció de dades han de ser senzills, estar ben documentats i han d'evitar els errors dels usuaris. La informació d'eixida ha d'estar ben presentada, ha de ser fàcil de comprendre i tenir un apropiat nivell de detall.
- Cal considerar quines poden ser les futures necessitats dels usuaris finals. Les decisions en el disseny poden estalviar modificacions posteriors del programari.
- Les dades s'han d'introduir en el sistema on i quan es produeixen. Els retards innecessaris poden produir errors i pèrdues d'informació.
- Les dades s'han de verificar quan s'introdueixen i s'han d'utilitzar mètodes automàtics d'entrada de dades sempre que siga possible.
- En alguns sistemes s'ha de controlar l'accés a les dades introduïdes i la modificació o introducció d'informació amb valors crítics, s'ha de controlar mitjançant informes (informes d'auditoria).
- Les dades s'han d'introduir en el sistema una sola vegada.
- Cal dissenyar processos (mòduls de programari) simples i independents per a desenvolupar un sistema mantenible i fàcil de millorar.

Moltes d'aquestes consideracions, a vegades, entren en conflicte unes amb les altres. És llavors quan l'enginyer informàtic ha de sospesar les alternatives i decidir quina desenvolupar. 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. Cal prendre les decisions que siguen més convenients per als usuaris finals.

5.2. Activitats del disseny

Abans de començar la fase de disseny cal tenir un coneixement complet i comprendre els requisits del sistema i l'anàlisi d'aquest. 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. També pot donar-se el cas que passen setmanes, fins i tot mesos, des que es desenvolupa l'anàlisi del sistema fins que es decideix l'inici del disseny i cal fer-ne una revisió, per tal de comprovar que l'anàlisi està al dia amb les necessitats de l'empresa i que es comprèn perfectament abans de començar aquesta activitat.

Tenint en compte els components identificats en un sistema d'informació i la interconnexió existent entre aquests, plantejar-se un punt de partida pot ser complicat; però, es pot considerar que de tots els

components, els requisits de les dades d'eixida i d'entrada són els que millor defineixen el sistema des del punt de vista de l'usuari, perquè proporcionen la visió física que tenen els usuaris del sistema.

S'ha de tenir en compte que tots aquests components del sistema interactuen entre si. Per tant, el disseny del sistema, i qualsevol altra fase del desenvolupament d'un sistema informàtic, no es pot considerar com un pas independent, sense cap relació, sinó que pot coincidir en diferents punts i qüestions i el disseny d'uns pot afectar el disseny d'altres.

La taula 5.1 mostra un resum de les principals activitats que es poden dur a terme en la fase de disseny.

TASCA	DESCRIPCIÓ-OBJECTIU
1. Revisió de l'anàlisi del sistema i de la definició de requisits	Familiaritzar-se amb l'anàlisi i les especificacions del sistema.
2. Disseny del sistema	
Disseny de les entrades del sistema	Determinar com s'han d'introduir les dades, dissenyar els documents per a la captura de dades, els registres d'entrada i les mesures de seguretat i validacions per a les dades.
Disseny de les eixides del sistema	Dissenyar el format físic per a cada eixida del sistema i com s'han de proporcionar aquestes físicament.
Disseny dels processos del sistema	Dissenyar els processos que ha de dur a terme el programari del sistema, com també qualsevol pas previ que hagen de realitzar les persones involucrades en el sistema.
Disseny dels fitxers del sistema i la BD	Dissenyar el sistema d'emmagatzematge de la informació tenint en compte l'organització, la forma d'accedir a les dades i la utilització d'aquestes.
Disseny i/o selecció del maquinari del sistema	Avaluar les necessitats del maquinari, comunicacions i altres tecnologies necessàries per a posar en marxa el sistema, segons els requisits establerts inicialment.
3. Especificar i presentar el disseny	Generar la documentació de les especificacions del disseny en la qual s'ha de detallar la forma i la implementació d'aquest. S'han d'incloure estimacions en temps i cost econòmic. Obtenir l'aprovació del disseny i de l'inici de la següent fase, el disseny detallat i la implementació.

Taula 5.1. Activitats de la fase de disseny

5.3. 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. La majoria de les interfícies d'usuari es porten a terme amb la utilització d'un mitjà visual. 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.[4] 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 a poder adaptar la interfície a seua capacitat.

S'ha de tenir en compte que el sistema es construeix per a mecanitzar algunes tasques que normalment ja les realitzava l'usuari de forma manual. Aquestes tasques es poden classificar en:

- **Tasques de comunicació:** en les quals els usuaris introdueixen dades al sistema.
- **Tasques de diàleg:** en les quals els usuaris poden introduir i extraure informació del sistema.
- **Tasques de control:** per les quals l'usuari pot controlar les accions que realitza el sistema, ordenar els processos i gestionar la resta de tasques.

El disseny de les interfícies d'usuari 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, són alguns dels punts en els quals les interfícies d'usuari poden causar problemes. S'han de dissenyar les interfícies d'usuari tenint en compte que es dissenya un sistema per als usuaris finals.

El disseny d'interfícies d'usuari està directament relacionat amb els avanços de la tecnologia informàtica. La tecnologia proporciona cada dia interfícies d'ús més senzill, i més agradables per als usuaris. Els entorns multimèdia, que suporten sofisticats sistemes operatius, proporcionen múltiples maneres d'interacció entre l'usuari i el sistema. S'han de conèixer tots aquests aspectes tecnològics, per tal d'utilitzar els que millor s'adapten al perfil dels usuaris i a les tasques que s'han de desenvolupar.

Encara que no funcionen de manera totalment independent, es poden agrupar les interfícies d'usuari en dos grans grups com es mostra en al taula 5.2, en interfícies d'entrada i interfícies d'eixida**.**

La qualitat de la informació que s'obté d'un sistema està en relació directa amb la qualitat i fiabilitat de les dades que s'introdueixen. Per tant, el disseny d'entrades és fonamental per a obtenir uns bons resultats de la informació que proporciona el sistema.

Les interfícies d'usuari més habituals, que poden ser tant d'entrada com d'eixida, són les pantalles amb teclats i diferents 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] També s'han de considerar les capacitats físiques dels dispositius a l'abast dels usuaris per a comunicar-se amb el sistema.

Mecanisme	Interfícies d'entrada	Interfícies d'eixida
De programari	Pantalles i menús	Pantalles (consultes) Informes impresos
De maquinari	Dispositius físics d'entrada de dades: teclat i ratolí, lector òptic de bandes magnètiques i codis de barres, escàner, reconeixedor de veu, etc.	Dispositius externs d'enregistrament de dades: cinta, disc, disquet, CD, etc.
	Dispositius de selecció i senyalització: pantalla tàctils, reconeixedor de veu, etc.	Altres dispositius: eixida audible, plotter, senyalitzadors especials (llums, ordres a mecanismes externs, etc.)

Taula 5.2. Interfícies d'entrada i interfícies d'eixida

5.3.1. Disseny de pantalles

Com s'ha mencionat en l'apartat anterior, les pantalles constitueixen un dels dispositius fonamentals d'entrada de dades i d'eixida d'informació. Són tant part de la interfície d'entrada, com de la d'eixida. 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. A més, és necessari considerar la forma d'accedir d'unes pantalles a altres, menús, llistes desplegables, etc.

Les pantalles que configuren la interfície amb l'usuari, d'entrada i d'eixida, han de complir determinades condicions que proporcionen al sistema una imatge d'homogeneïtat, consistència i professionalitat, al mateix temps que permeten una manipulació àgil, senzilla i eficaç als usuaris. Una pantalla ben dissenyada reflecteix les necessitats i característiques dels seus futurs usuaris, està desenvolupada tenint en compte les restriccions físiques del terminal, utilitza de forma eficient les capacitats del programari i aconsegueix els objectius de negoci per als quals ha sigut dissenyada.

A més de totes aquestes condicions, s'ha de tenir en compte aspectes particulars de les pantalles i de l'ús d'aquestes per a dissenyar-les correctament. A continuació, es descriuen aquests aspectes, tenint en compte les condicions i necessitats per a les quals es dissenyen.

Factors humans del disseny de les pantalles

La utilització que les persones fan d'una pantalla influeix en diferents aspectes del disseny d'aquestes. Cal considerar les característiques humanes generals, i les particulars de cadascun dels usuaris del sistema per tal de dissenyar pantalles adaptades a aquestes característiques, 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.

Quant a les característiques generals, cal tenir en compte la capacitat de retenció de les persones, no saturar pantalles o dissenyar-les amb la utilització de colors que faciliten la visió i la localització de la

informació, tipus de caràcters i mides adequats, etc., són algunes de les consideracions que cal tenir en compte.

D'altra banda, cal considerar els diferents grups d'usuaris del futur sistema. Els usuaris eventuals necessiten pantalles molt senzilles d'ús, amb ajudes i indicacions dels passos que cal seguir. No obstant això, uns altres usuaris amb més coneixements, que utilitzen habitualment el sistema, necessiten pantalles clares, que els permeten realitzar aquestes activitats de forma senzilla però eficaç. Per exemple, en el cas d'un sistema bancari, el mateix procés de reintegrament en efectiu ha de tenir una interfície diferent per als caixers, l'ús de la qual està destinat als clients, que per als terminals que utilitzen els treballadors del banc. En el primer cas, l'ús de grans terminals, de pantalles tàctils que proporcionen instruccions clares i simples, és fonamental perquè els clients realitzen aquestes operacions sense problemes. En el segon cas, és més important l'agilitat i rapidesa amb les quals es porte a terme el procés.

Elements de les pantalles

Per tal de dissenyar pantalles i finestres de forma adequada cal considerar tots els possibles elements que han de mostrar-se en aquestes. Hi ha elements com, per exemple, la data, que es manté en totes les pantalles d'un sistema informàtic, i que s'ha de definir i ubicar d'igual manera en totes aquestes per a donar una imatge homogènia i professional. Altres elements, com per exemple les dades que s'han de llegir, depenen de cadascuna de les pantalles i de l'ús d'aquestes. En general, es pot dir que els elements que han d'aparèixer en una pantalla o finestra són, com es mostra en la figura 5.1:

- **Títol:** per ubicar l'usuari en el procés. Normalment es mostra a la part superior, pot justificar-se a l'esquerra o centrar-se.
- **Dades per identificar la pantalla:** identificador de la pantalla, número de pàgina, usuari, data i hora. Es tendeix a ubicar aquestes dades a la part baixa de la pantalla o en un marc inferior de la finestra, que es ressalta amb un color diferent. En textos formals es recomana ubicar-les a la part superior.
- **Missatges:** inclouen estat, notificacions, ajuda emergent de possibles errors o accions crítiques, i informació d'errors greus. Poden utilitzar-se finestres superposades, en les quals es representen icones senzilles i formes de confirmar l'acceptació de l'error.
- **Barra de menús:** normalment a la part superior de la pantalla sota del títol.
- **Tecles de funció, botons 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, que normalment es mostren al principi, 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 text d'aquests.

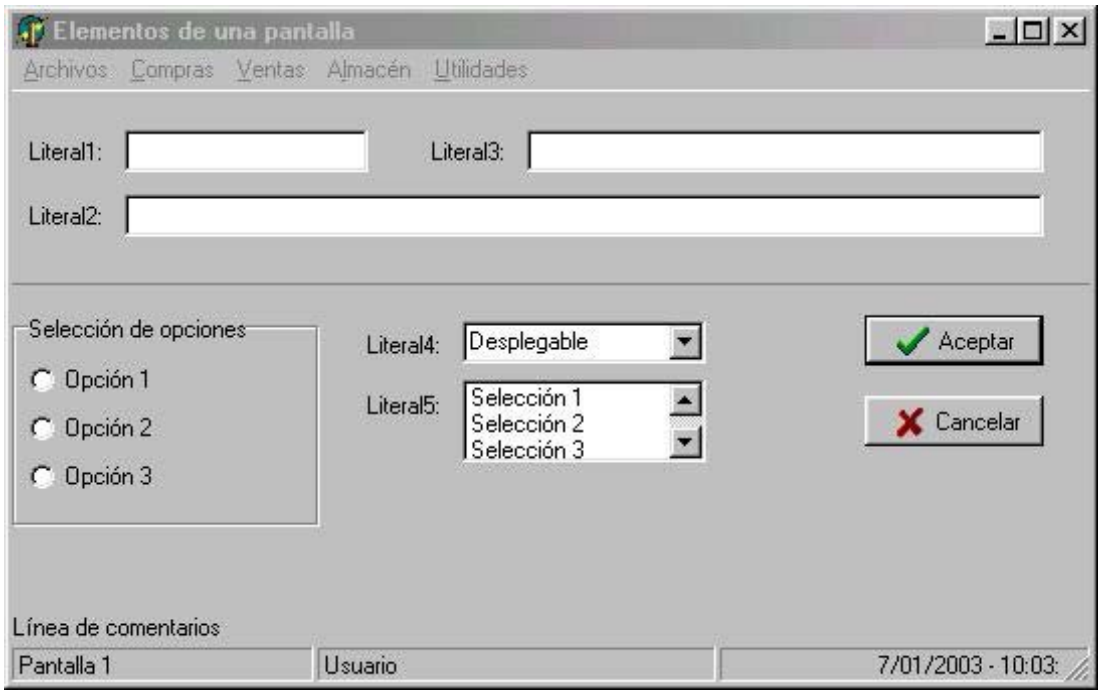


Figura 5.1. Exemple de finestra i elements habituals

Moltes d'aquestes recomanacions tendeixen a desenvolupar-se 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, distribució, colors, fonts, missatges, etc.

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 a 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.
- Les dades de les pantalles s'han d'agrupar per la funcionalitat o semàntica d'aquestes. A més, s'ha de tenir en compte que l'aspecte visual complisca les característiques bàsiques vistes al punt anterior. 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 a 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.
- Es aconsellable crear plantilles per dissenyar les pantalles tenint en compte la imatge de l'empresa.

Documentació del disseny de pantalles

Per a completar el disseny de pantalles, cal desenvolupar:

- Un document gràfic i un descriptiu dels estàndards que s'utilitzaran, comuns a totes les pantalles.
- Un document gràfic i un altre descriptiu (veure taula 5.3) per a cadascuna de les pantalles.
- Inventari de pantalles on es mostra el codi de cada una, el nom, tipus o utilització a què està destinada, usuaris, etc.

- Un document on es mostre la jerarquia de les pantalles, és a dir, com es pot accedir o navegar entre les diferents pantalles que componen aquest apartat de la interfície d'usuari.

<table>
<tr><td colspan="4">DOCUMENTACIÓ DEL SISTEMA [NOM DEL SISTEMA]</td></tr>
<tr><td colspan="3">Enginyer informàtic/dissenyador:</td><td>Data:</td></tr>
<tr><td colspan="3">Tipus de document: Descripció de pantalles</td><td>Document: [P001]</td></tr>
<tr><td colspan="4">Nom: Pantalla entrada dades de clients
Desplaçament: Vertical, (altres atributs)
Descripció/comentaris
Usuaris amb accés:
S'accedeix a: [pantalles connectades Pxxx]
S'accedeix des de: [pantalles des de les quals es pot arribar]</td></tr>
<tr><td colspan="4">CAMPS</td></tr>
<tr><td>CAMP</td><td>TIPUS DE CAMP</td><td>LONGITUD</td><td>FORMAT ATRIBUTS</td></tr>
<tr><td>Número client</td><td>Numèric</td><td>6</td><td>999-999 protegit/vídeo inv.</td></tr>
<tr><td>Nom client</td><td>Alfanumèric</td><td>30</td><td>X</td></tr>
<tr><td>Adreça client</td><td>Alfanumèric</td><td></td><td></td></tr>
</table>

Taula 5.3. Document descriptiu d'una pantalla

El document gràfic ha de ser un esquema o fitxa on es dibuixa la pantalla, amb la capçalera d'aquesta, dades, literals, missatges, etc. Es pot desenvolupar amb una ferramenta informatitzada apropiada per al disseny de pantalles, ferramentes per al desenvolupament de prototips, ferramentes CASE que donen suport a aquesta activitat, o també sobre el mateix entorn de programació on s'ha de desenvolupar el sistema.

El document descriptiu de la pantalla és una fitxa on es detalla la informació rellevant per a la implementació i el seu ús posterior. A manera d'exemple es mostra a la taula 5.3 una possible configuració d'aquesta fitxa.

5.3.2. Disseny d'informes

Com s'ha mencionat al començament d'aquest capítol, els informes constitueixen un dels principals apartats per a proporcionar als usuaris informació del sistema. 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 a 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. Encara que no s'imprimisquen els documents, es poden visualitzar sobre la pantalla del sistema amb un format que requereix un disseny acurat i estudiat.

Tipus d'informes

Els informes es poden classificar pel contingut i per la distribució d'aquests. En el primer cas, es poden definir informes de detall, informes d'excepció i informes resums (habitualment per columnes); i en el segon, es poden considerar informes externs i interns. A continuació, s'estudien amb més deteniment les característiques que cal considerar en el disseny de cadascun d'aquests tipus d'informes, documents o formularis.

1. **Informes de detall:** es generen una o diverses línies d'informació que es refereixen a un únic concepte, en ocasions una pàgina sencera. No és imprescindible imprimir totes les dades d'un registre. Poden incloure línies de totals, parcials i globals. Poden presentar-se en forma de columnes, o bé com a fitxes completes de la informació requerida. Exemples: llistats de vendes per a clients, etiquetes per a l'enviament de correu, factures, etc.
2. **Informes d'excepció:** mostren únicament aquells registres que compleixen una determinada condició de forma excepcional. Són convenients quan l'usuari final no necessita conèixer tota la informació referent a un determinat registre. Exemple: informes de clients amb saldo deutor, llistat d'errors d'usuaris, llistats d'absències de treballadors, etc.
3. **Informes de resum:** a vegades, els usuaris finals només volen conèixer informació molt resumida o d'alt nivell. Habitualment aquests llistats presenten la informació organitzada per columnes i també es poden incloure estadístiques i gràfics. Per exemple, els resums de compres o de vendes per mesos o articles, els totals d'hores treballades per departament i per any, llistats d'hores perdudes per absentisme i baixes per mesos, etc.
4. **Informes interns:** com el nom indica, s'utilitzen de forma interna per a consultes dels treballadors de l'empresa. L'objectiu és obtenir un informe de baix cost i d'alta utilitat. Normalment s'utilitza paper continu o reciclat, i formats que proporcionen molta informació i utilitzen poc paper. Exemple: llistat de vendes per client, llistat de baixes per malaltia, etc.
5. **Informes externs:** són els que proporcionen i utilitzen persones alienes a l'organització (estaments governamentals, clients, proveïdors, pagaments de la companyia, etc.). Alguns necessiten tenir uns formats especials que, a vegades, estan regulats per les normatives legals (declaracions per a Hisenda, informes per a la Seguretat Social, 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 qualsevol informació que calga, segons el tipus de document (CIF de l'empresa, data, tipus de

document, etc.). Molts d'aquests informes es generen sobre paper preimprès, el format del qual és necessari considerar per a desenvolupar la implementació dels processos que els imprimeixen i controlar la col·locació del paper corresponent, si es llancen a un dispositiu d'impressió. Exemple: factures de clients, albarans, etc.

Com es pot deduir, poden haver-hi informes de detall de distribució interna i externa i informes de resum de distribució externa i interna.

Una altra consideració important en el disseny d'informes és establir el tipus de paper escaient, si els documents que es volen obtenir requereixen certa confidencialitat o seguretat, etc., per a controlar els processos on s'emeten i les impressores i el paper que ha d'estar disponible en el moment de la impressió. Per exemple, la impressió de les nòmines, o el pagament en talons que realitzen algunes empreses.

Elements dels informes

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 a poder distingir els valors i per a

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 significat d'aquestes, les dades més importants o els identificadors a l'esquerra, i les dades referents a quantitats o imports, a la dreta.

Ubicació de la informació en els informes

Al 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 correcte ús d'aquests. Segons el tipus d'informe i l'objectiu al qual està destinat hi haurà consideracions particulars, encara que en general es poden tenir en compte algunes consideracions:

- S'ha de crear espais específics amb homogeneïtat per als títols, logotips, data i altres dades rellevants.
- S'agruparà la informació tenint en compte la funcionalitat i context.
- Al igual que en les pantalles, caldrà proporcionar una distribució agradable sense saturar ni aïllar els continguts.
- **Els informes per columnes** mostraran clarament el títol de cadascun dels continguts, i en la primera columna es mostrarà la dada que identifique el concepte llistat. Per exemple en un llistat de clients la primera columna serà el nom o el DNI. Les columnes amb dades numèriques es mostren habitualment a la part dreta dels informes, justificant les xifres a la dreta i amb igual nombre de decimals.
- **Totals i subtotals:** sobretot als informes que mostren dades de tipus econòmic és important controlar els salts de pàgina i totalitzar o mostrar subtotals sempre que es considere necessari per tal de proporcionar una comprensió millor de l'informe.
- **Peus de pàgina:** per a informes que es prolonguen més d'una pàgina, és convenient mostrar línies de resum i indicar que l'informe continua. Un bon costum és indicar el final d'un informe mitjançant el text convenient.

Documentació del disseny d'informes

Per a dissenyar el format real d'un informe imprès es poden utilitzar plantilles amb espais quadriculats, que proporcionen una forma estàndard per a visualitzar els documents. Encara que actualment les impressores proporcionen configuracions i opcions per obtenir informes de qualsevol grandària i format, en el cas d'informes que incloguen columnes, cal tenir en compte les característiques particulars de les impressores que s'utilitzen, la resolució, la mida del paper que s'ha d'usar, etc. En el format s'ha de dissenyar el text fix amb tots els caràcters que corresponguen, i les dades variables indicant la longitud màxima que es pot assolir.

De la mateixa forma que en el disseny de pantalles, s'han de desenvolupar diferents documents:

1. Un document gràfic i un descriptiu dels estàndards que s'han d'utilitzar per cadascun dels tipus d'informes, externs interns, etc.
2. Un document gràfic i un descriptiu (veure taula 5.4) de cadascun dels informes que es dissenyen. En el document gràfic o plantilla cal mostrar les capçaleres, les dues primeres línies de detall, les línies de totals i subtotals de final de pàgina i altres aspectes particulars de l'informe. No cal representar cadascuna de les pàgines, només les que siguen diferents en alguna dada o format. En el document descriptiu s'ha d'aportar informació sobre el nombre de còpies, la distribució, l'ús, el tipus de paper que necessita l'informe i la freqüència d'impressió. A la taula 5.4 es mostra un exemple de document descriptiu.
3. Un inventari actualitzat dels diferents informes que el sistema pot proporcionar. En aquest inventari cal identificar l'informe mitjançant la capçalera, indicar l'objecte de l'informe, la periodicitat i la distribució d'aquest. Aquesta informació permet controlar la creació de nous informes per a evitar repeticions de processos.

DOCUMENTACIÓ DEL SISTEMA [NOM DEL SISTEMA]			
Enginyer informàtic/dissenyador:		Data:	
Tipus de document: Descripció d'informes		Document: [I001]	
Nom: Informe de vendes per client Tipus: intern de detall Tipus de paper: A4 vertical Tipus de cinta/qualitat d'impressió: esborrany Descripció/comentaris: Còpies: 5 Distribució:			
CAMPS			
CAMP	TIPUS DE CAMP	LONGITUD	FORMAT ATRIBUTS
Núm. Client	Numèric	6	999-999 protegit/vídeo inv.
Nom del client	Alfanumèric	30	X
Vendes últim mes			

Taula 5.4. Document descriptiu d'un informe

S'ha d'assignar a cada informe un codi que l'identifique de forma única, i que pot aparèixer en el document imprès per a detectar més fàcilment possibles errors.

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 a realitzar consultes, estadístiques, gràfics, etc.

5.4. Disseny de processos

Els objectius del disseny de processos es poden resumir en els següents punts:

- Tractar els aspectes físics fonamentals que afecten el disseny del programari i delimitar l'abast dels esforços del desenvolupament del programari.
- Ser capaços de subdividir i agrupar els processos del diagrama de flux de dades en mòduls de programari del sistema i utilitzar un criteri d'agrupació apropiat.
- Descriure els aspectes fonamentals que s'han de tenir en compte per a seleccionar les opcions més adequades per a la posterior implementació del sistema.
- Obtenir les especificacions necessàries perquè els programadors puguen, a partir d'aquestes, desenvolupar la codificació dels mòduls que constituiran el programari del sistema.

El disseny dels processos connecta les especificacions i els models desenvolupats en l'anàlisi (diagrames de flux de dades en aquesta assignatura) amb els milers de línies de codi que permeten portar a terme les funcions definides pels requisits i els models de l'anàlisi.

Cada sistema informàtic en un determinat entorn utilitza una plataforma o un entorn específic. Aquesta plataforma és una combinació de maquinari i programari que és necessari considerar per a dissenyar el programari, les estratègies d'utilització d'aquest i els procediments d'usuari. Els equips existents poden prefixar els requisits per a portar a terme determinades transaccions, però normalment la plataforma es dissenya i instal·la perquè s'adapte a les necessitats del sistema.

Per a desenvolupar el disseny dels processos del sistema cal tenir en compte l'entorn que està disponible o la possibilitat d'instal·lar un entorn més adequat. En qualsevol cas, les característiques i utilitats que aquest entorn proporcione haurien de ser considerades per tal d'optar per diferents mètodes de disseny i implementació dels processos del sistema.

Per a dissenyar els processos cal decidir quina és la forma en la qual aquests s'executen al sistema. Aquest aspecte es pot assenyalar en la fase d'anàlisi, però cal tenir-lo en compte i revisar-lo a l'hora de dissenyar un programari adequat.

A continuació, es descriuen els principals mètodes de processament que un enginyer informàtic ha de considerar per a dissenyar els processos de la manera més apropiada, i quins criteris cal tenir en compte per tal d'obtenir un bon disseny de qualitat dels processos.

5.4.1 Tipus de processos en funció de la interacció amb l'usuari

Actualment, excepte casos particulars, tots els sistemes informàtics es desenvolupen incloent tant processos en línia, com processos per lots. Fins i tot en una mateixa transacció apareixen apartats que es duen a terme mitjançant la interacció dels usuaris finals i altres que es realitzen posteriorment com a processaments per lots.

Un procés en línia és aquell en el qual hi ha una interacció directa entre els usuaris i el sistema informàtic. Les transaccions es realitzen quan i on es produeixen i proporcionen informació directament als usuaris finals. Aquests treballen en el seu propi ordinador o bé en un terminal connectat a un servidor. El sistema ha d'estar sempre disponible per a ser utilitzat pels usuaris, ja siga per a obtenir informació o per a realitzar transaccions.

Un procés per lots és aquell en el qual la informació es recull durant un període de temps determinat o de forma eventual i es processa com a grup més tard. Aquesta forma de processar les dades s'utilitzava freqüentment en els anys 50-60 com a mètode únic. Els documents font de les dades del sistema es recopilaven en diferents llocs de treball i es processaven posteriorment en un únic ordinador central. Actualment els processos per lots s'utilitzen per a realitzar processos periòdics que afecten un gran volum d'informació. Les principals característiques d'aquests processos són:

- Les dades es recullen o introdueixen en diferents intervals de temps i en ubicacions físiques diferents del lloc on es processen.
- Existeixen responsables (operadors) d'executar els processos per lots d'acord amb un calendari o una determinada planificació de les tasques del sistema. No cal la interacció dels usuaris finals.
- Normalment s'accedeix a un gran nombre de dades contingudes en els fitxers del sistema, l'accés sol ser seqüencial, encara que no és imprescindible.
- Els processos per lots s'executen normalment quan els usuaris no utilitzen el sistema, o en moments de poca activitat.

Els principals avantatges i desavantatges d'aquests dos mètodes de processament són:

1. En un procés en línia la informació es verifica en el moment en què es produeix i està disponible de forma adequada a partir d'aquell moment. Per tant, la informació sempre està actualitzada.

2. L'entorn necessari per a donar suport a les transaccions en línia normalment utilitza les últimes tecnologies, requereix una inversió econòmica més elevada que en el cas de transaccions de processaments per lots.
3. Els errors del sistema o fallades tecnològiques en un entorn en línia poden produir repercussions que són visibles pels usuaris finals i poden alentir o paralitzar el seu treball.
4. Els processos per lots s'han de dissenyar amb la consideració de diferents punts de control i verificació de la correcció del procés, llistats, comparacions de fitxers, etc. A més, és convenient proporcionar mecanismes per a recuperar les dades i reiniciar el procés en cas de produir-se qualsevol error.
5. Els processos en línia amb temps de resposta elevats paralitzen l'activitat dels usuaris.

5.4.2. Qualitat del disseny

Un dels principis fonamentals del disseny estructurat és que un sistema complex hauria de ser subdividit en mòduls manejables, de forma que cadascun d'aquests fóra tan independent de la resta com fóra possible i que realitze una única funció.

Per tal que els mòduls dissenyats puguen reutilitzar-se i el disseny sustente els principis bàsics d'un programari de qualitat (manteniment, eficiència i fiabilitat, entre altres), cal tenir en compte dos criteris: l'acoblament i la cohesió.

Acoblament

L'acoblament proporciona una mesura de la independència dels mòduls: dos mòduls estan dèbilment acoblats si són independents i fortament acoblats si són dependents. Un baix acoblament entre mòduls indica un sistema amb una bona partició, per la qual cosa l'objectiu és minimitzar l'acoblament, és a dir, obtenir mòduls tan independents com siga possible. Aquest concepte valora la relació *entre* mòduls.

Els tipus d'acoblament es defineixen tenint en compte quines dades es transmeten entre els mòduls:

- **Normal:** dos mòduls A i B estan normalment acoblats si A crida B, i B torna a A, i tota la informació transmesa entre si es realitza mitjançant els paràmetres presents en la crida. Aquesta informació poden ser dades simples, un bloc de dades, dades compostes (exemple: dades de client), o un element de control.
- **Acoblament comú (variables globals):** els mòduls es comuniquen amb l'ús de fluxos de dades i control definits en una àrea global de dades. Aquest acoblament pot provocar problemes de manteniment.

- **Acoblament de contingut:** dos mòduls estan acoblats en contingut quan un dels dos fa referència a una sentència continguda en l'altre sense emprar un format de crida (en alguns llenguatges de programació, la sentència GOTO). Aquest és el pitjor tipus d'acoblament i s'ha d'evitar.

Cohesió

La cohesió dóna una mesura de la «força» de la relació funcional dels elements (instruccions o dades) dins d'un mòdul. És a dir, quantifica la relació dels elements definits *dins* d'un mateix mòdul. L'objectiu és dissenyar mòduls altament cohesionats, maximitzar la cohesió (elements molt relacionats que constitueixen una única funció).

L'acoblament i la cohesió són interdependents: la cohesió d'un mòdul determina el grau d'acoblament del mòdul amb altres mòduls. Assegurar-se que tots els mòduls tenen bona cohesió és la millor forma de minimitzar l'acoblament entre mòduls.

Els tipus de cohesió es defineixen tenint en compte quines activitats desenvolupa un mateix mòdul. De millor a pitjor són els següents (els tres primers són fàcilment mantenibles):

- **Funcional:** tots els elements del mòdul contribueixen a una funció o activitat simple ben definida. Exemples: calcular el cosinus d'un angle; verificar sintaxi alfabèticament; calcular salari net; calcular impostos de vendes. Els mòduls funcionalment cohesius poden ser utilitzats fàcilment en més d'un programa. Els sistemes construïts a partir de mòduls amb baix acoblament i alta cohesió són més fàcils de mantenir.
- **Seqüencial:** els elements del mòdul participen en una seqüència d'activitats on la sortida d'una activitat és l'entrada de la següent. Exemple: assignar nombre de factura i escriure nova factura. Un mòdul seqüencialment cohesiu normalment té bon acoblament i és fàcil de mantenir; el desavantatge que presenta és que no és tan fàcil d'utilitzar en el mateix sistema (o en altres), ja que conté activitats que no tenen perquè anar sempre juntes. Millora: crear un mòdul distint per a cada funció o activitat dins del mòdul.
- **Comunicacional:** els elements del mòdul contribueixen a activitats que utilitzen les mateixes dades d'entrada o d'eixida; l'ordre de les activitats no és important. Exemple: determinar títol, preu, editorial i autor d'un llibre; buscar el nom d'un client i buscar el balanç d'un client. Són mantenibles, encara que es complica la compartició dels mòduls; encara és pitjor si s'intenta compartir codi entre les funcions dins del mòdul (produir un informe dels salaris i calcular la mitjana del salari). Es pot millorar igual que en el cas anterior.
- **Processal:** les sentències del mòdul realitzen activitats que són part del mateix procediment, però que no estan relacionades seqüencialment o comunicacionalment. Exemple: escriure, llegir i editar

alguna cosa; calcular mitjana de les taules A i B (taules no relacionades). El principal problema: les activitats dins del mòdul normalment estan relacionades amb les desenvolupades en uns altres mòduls, la qual cosa comporta un fort acoblament i, per tant, difícil manteniment (igual millora).

- **Temporal:** els elements del mòdul intervenen en activitats que es relacionen pel fet de succeir al mateix temps. Exemple: el clàssic exemple de cohesió temporal és un mòdul d'inicialització d'un programa, es fan coses molt diferents juntes perquè es fan al mateix temps.
- **Lògica:** els elements del mòdul contribueixen a activitats que són parts d'una categoria general, però que no es realitzen al mateix temps. Exemple: activitats que, a pesar de ser diferents, comparteixen una mateixa interfície. Solen ser mòduls difícils de comprendre i mantenir.
- **Coincident:** els elements del mòdul participen en funcions o activitats que no tenen cap relació aparent entre si. Es reconeixen per indicadors de control, noms sense sentit.

5.5. Diagrama d'estructura

La principal tècnica utilitzada en el disseny per a modelitzar i documentar **els processos** es denomina diagrama d'estructura**.** Ofereix una visió de com es divideix el programari del sistema en mòduls (caixes negres), com s'organitzen aquests i quina informació es transmet entre si. Per tant, és un esquema que mostra gràficament els mòduls físics o parts que componen el programari del sistema i les relacions entre aquests.

5.5.1. Components del diagrama d'estructura

En programació es considera un mòdul la unitat més menuda que pot compilar-se independentment. En disseny estructurat no s'imposa aquesta restricció sobre la possibilitat de compilació, un mòdul es considera la representació d'allò que posteriorment serà un conjunt de sentències que realitzaran una activitat.

Per tant, un mòdul és una col·lecció de sentències de programa amb quatre atributs bàsics:

- Entrada/eixida.
- Funció que realitza.
- Mecànica (codi utilitzat per a realitzar l'esmentada funció).
- Dades internes.

Els dos primers atributs constitueixen la «visió externa» del mòdul (què fa el mòdul), mentre que els dos últims en conformen la «visió interna» (com ho fa). A més d'aquests atributs bàsics, un mòdul es

caracteritza per rebre un nom que permet fer referència a aquest com una unitat total, i poder realitzar crides o ser cridat per altres mòduls; per exemple: subprograma de Pascal o C.

El disseny de processos se centra en la visió externa del mòdul; els detalls de la codificació o lògica dels mòduls es reserven per al disseny detallat i la programació que s'ha de portar a terme en la següent fase.

Representació gràfica

Un mòdul es representa mitjançant un rectangle amb el seu nom a l'interior. El nom ha de ser una definició de la funció del mòdul, és a dir, d'allò que realitza cada vegada que és cridat; cal evitar l'ús de noms vagues, ambigus.

Un mòdul creat específicament es representa com es mostra en la figura 5.2. Un **mòdul predefinit** es representa com es mostra en la figura 5.3. S'utilitzen fletxes sòlides per representar que es passa paràmetres de control, o en blanc per representar com es passa la informació entre mòduls, com es mostra en les figures 5.4 i 5.5.

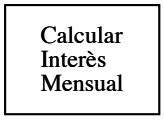


Figura 5.2. Exemple de mòdul

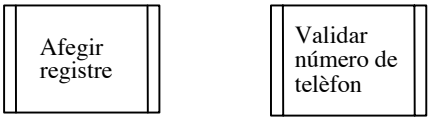


Figura 5.3. Exemple de mòduls predefinits

Figura 5.4. Fletxa sòlida: representació de paràmetres de control

Figura 5.5. Fletxa en blanc: representació de paràmetres de dades

En els diagrames s'indica com els mòduls es criden entre sí, per exemple en la figura 5.6 es mostra que el mòdul A és «capaç» de fer una crida al mòdul B.

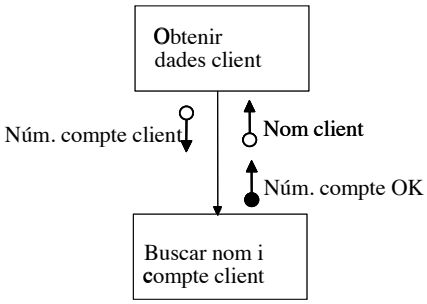


Figura 5.6. Exemple de crida entre mòduls

El diagrama d'estructura no mostra l'ordre en què es realitzen les crides als mòduls. Per exemple, el diagrama més habitual per a indicar que el mòdul X realitza crides a A, B i C (en aquest ordre) és com es mostra en la figura 5.7.

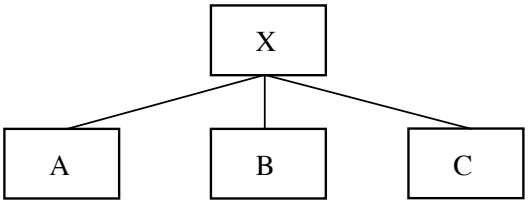


Figura 5.7. Exemple de crides entre mòduls

El nom d'un mòdul ha de resumir no només la funció d'aquest codi, sinó també dels codis de tots els mòduls als quals crida. Per tant, el nom que es dóna al mòdul superior del diagrama resumeix la funció del sistema global.

Especificació dels mòduls

El diagrama d'estructura per si mateix no proporciona informació suficient per a començar la tasca de codificació, per la qual cosa cal adjuntar una descripció de cada mòdul i element de dades que apareixen al diagrama (després de l'anàlisi, ja estan les dades especificades).

Per a realitzar l'especificació dels mòduls és important considerar que s'ha de proporcionar al programador la informació suficient per tal de dur a terme la codificació dels diversos mòduls, però sense indicar-li com ha de fer el treball. Decidir la millor forma de codificar un mòdul és tasca del programador (creativitat), no del dissenyador. Cal buscar un compromís entre especificar massa i (pitjor encara) especificar molt poc.

Especificació d'interfície del mòdul
Mòdul:
Propòsit:
Paràmetres d'entrada:
Paràmetres d'eixida:
Detalls funcionals (a alt nivell):
Propòsit:
Taules de la BBDD implicades:
Interfícies d'usuari involucrades:

Figura 5.8. Exemple d'especificació d'interfície

- **Especificació de la interfície:** és la millor forma de descriure un mòdul, ja que s'indica al programador la funció que ha de realitzar el mòdul i les dades d'entrada i d'eixida d'aquest, sense entrar en excessiu detall. D'aquesta forma, el programador és lliure de triar la implementació (sempre que puga satisfer l'especificació de la interfície).
- Les futures tasques de manteniment del sistema se simplifiquen si juntament amb el codi dels mòduls s'inclou l'especificació (és més fàcil comprendre el mòdul) d'aquest. A més, en el cas que

el codi final fóra un desastre, el programador encarregat del manteniment tindria un punt de partida on començar a reescriure tot el codi de nou.

- **Pseudocodi:** descriu un mòdul de forma molt més detallada que com ho fa l'especificació de la interfície, per la qual cosa existeix menys marge d'error quan el programador el trasllada a codi real. No obstant això, si es detalla en excés deixa menys elecció al programador. Per tant, s'ha de triar el nivell de detall adequat, en funció de l'habilitat dels programadors i del tipus de mòdul que s'estiga especificant.

5.5.2. Desenvolupament del diagrama d'estructura

S'ha de començar el disseny a partir dels models de processos de l'anàlisi (en aquest cas els DFDs), per a assegurar que els programes es construeixen d'acord amb els requisits de l'usuari. Per tant, un primer pas és convertir els DFDs en diagrames d'estructura. Per a això, hi ha dues tècniques: anàlisi de transformació i de transacció.

- **Anàlisi de transacció:** permet construir un diagrama d'estructura a partir d'un DFD que conté processos amb estructura *case* (per exemple, un menú)**.** L'objectiu d'aquest mètode és identificar el procés que determina el camí que segueix el flux de dades d'entrada, procés conegut com a centre de transacció. El diagrama d'estructura es forma amb el centre de transacció com el mòdul superior de la jerarquia.
- **Anàlisi de transformació:** permet construir un diagrama d'estructura a partir d'un DFD que conté processos seqüencials. L'objectiu d'aquest mètode consisteix a obtenir un diagrama d'estructura que estiga disposat com una jerarquia, on els mòduls més a l'esquerra sorgeixen de la conversió dels processos associats amb les entrades al DFD, i els situats a la dreta provenen de processos associats a les eixides del DFD. Els processos que al DFD estan entre les entrades i eixides, denominats àrea de transformació central, arriben a ser els mòduls del centre de la carta d'estructura. Es crea un mòdul a la part superior de la jerarquia, al qual s'assigna el nom del programa.

Normalment, les anàlisis de transformació i transacció s'empren juntes sobre el mateix DFD. És més comú començar per l'anàlisi de transacció, per exemple, per especificar la selecció d'un menú, i després emprar l'anàlisi de transacció per a crear el diagrama d'estructura inicial per a cada selecció del menú.

5.5.3. Definició dels programes

Una vegada creat el diagrama d'estructura on s'indiquen els mòduls que s'han de programar, s'arriba al final de l'etapa de disseny i al començament de la d'implementació. No obstant això, abans de

començar la implementació s'han de combinar les distintes especificacions de disseny creades en un paquet simple, organitzat, que especifique, sense ambigüitat, com s'han d'implementar els diferents programes i mòduls que ha d'utilitzar cadascun.

A partir dels diagrames d'estructura s'ha de configurar la documentació que ha de rebre el programador, per a poder desenvolupar el disseny detallat i la codificació de cadascuna de les unitats de programació que constitueixen el sistema.

A banda dels mòduls definits, cal identificar i definir altres unitats de programació que no s'han tingut en compte fins ara, com són els mòduls d'ús comú, interfícies, còpies de seguretat, extraccions, conversions, proves (comparacions de fitxers), eliminacions periòdiques de dades, impressió d'informació de control, etc. S'han d'identificar tots aquests mòduls i s'han de documentar apropiadament, per al posterior desenvolupament i ús generalitzat d'aquests per part de l'equip de desenvolupament.

Per a definir els programes o unitats de programació s'han de considerar les característiques tècniques de l'entorn de desenvolupament i explotació del sistema. Inicialment es poden considerar com a unitats de programació:

- Les bombolles del DFD.
- Cadascun dels diagrames d'estructura desenvolupats.
- Subconjunts dels diagrames d'estructura.
- Agrupacions dels diagrames d'estructura.

També s'ha de tenir en compte els processos que s'han de desenvolupar per lots i en línia, les característiques particulars de cada tipus i agrupar els mòduls que els componen en programes, segons els següents criteris:

- Temporals.
- Dades comunes d'entrada/eixida.
- Interfícies d'usuari comunes.

Agrupació de mòduls en programes per lots

Com s'ha descrit a l'apartat anterior**,** els processos per lots s'utilitzen per a realitzar processos periòdics que afecten un gran volum d'informació i que habitualment s'executaran sense necessitat de diàlegs amb l'usuari. Per crear els programes que formen aquestes cadenes d'execució cal tenir en compte:

1. Els programes han de ser el més elementals possibles. Cal evitar programes la grandària dels quals en dificulte el manteniment.
2. Els mòduls que afecten independentment diferents fitxers s'han d'implementar en unitats de programació separades.
3. Un mateix programa no ha de contenir un mòdul que cree o genere un fitxer i realitze un procés sobre aquest (descompondre en diferents programes).
4. Definir programes separadament sempre que es produïsca una determinada discontinuïtat com, per exemple, si hi ha una ordenació d'un fitxer i un processament posterior o previ, o quan hi ha un punt de decisió per a la continuïtat o no del procés.
5. Verificar que s'acompleix que els programes o parts només es connecten per fitxers, i que les entrades a cada part o programa identificat es generen amb anterioritat.
6. Evitar l'ús excessiu de fitxers auxiliars.
7. Cal considerar les limitacions pròpies de l'entorn o de la plataforma de desenvolupament: temps d'execució, etc.

Agrupació de mòduls en programes en línia

Els processos en línia es caracteritzen per la relació de l'usuari amb el sistema. Un diàleg finalitza quan dóna la resposta. Per a cadascun d'aquests diàlegs s'ha d'identificar un programa, si es vol continuar amb el procés s'activa un altre programa, però no es deixa mai un sistema amb un programa a meitat processar, perquè utilitza recursos i memòria. Per agrupar els mòduls en programes en línia cal considerar els següents punts:

- L'actualització de fitxers s'ha de realitzar en finalitzar la transacció (alta d'un estudiant).
- S'han de definir programes per als mòduls que cobreixen el procés d'imprimir un informe. Si un mòdul que imprimeix un informe utilitza la mateixa informació que el mòdul d'entrada o de lectura de les dades de l'informe, llavors es pot agrupar aquells mòduls en un únic programa.
- Cal obtenir programes que siguen el més elementals possibles.

A partir del diagrama d'estructura, l'especificació dels mòduls, el disseny de pantalles i d'informes, el disseny de la base de dades i el disseny dels fitxers auxiliars, el programador ha de ser capaç de desenvolupar el codi i les proves necessàries per a implementar les unitats de programació que constitueixen el sistema en funcionament. El resultat del disseny de processos ha de ser una documentació que incloga:

1. **Diagrama d'estructura** i llistat de mòduls classificats per ús. Per a cada mòdul cal obtenir especificació, entrades, eixides, interfícies i fitxers.
2. **Llistat de programes** a desenvolupar, on s'indique la finalitat de cada programa, els mòduls

que el componen, entrades i eixides, interfícies que utilitza i fitxers de la base de dades involucrades.

3. **Model gràfic** on es represente com es comuniquen uns programes amb els altres, menús, accessos per funcions o seleccions, etc. Poden utilitzar-se representacions semblants a les del diagrama d'estructures.

Resum

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

Segons els elements que constitueixen un sistema informàtic, el disseny se centra a estudiar l'anàlisi i 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, i el disseny de processos.

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 i permet estimar els costos temporals, de recursos humans i econòmics d'aquestes activitats.

Activitats complementàries

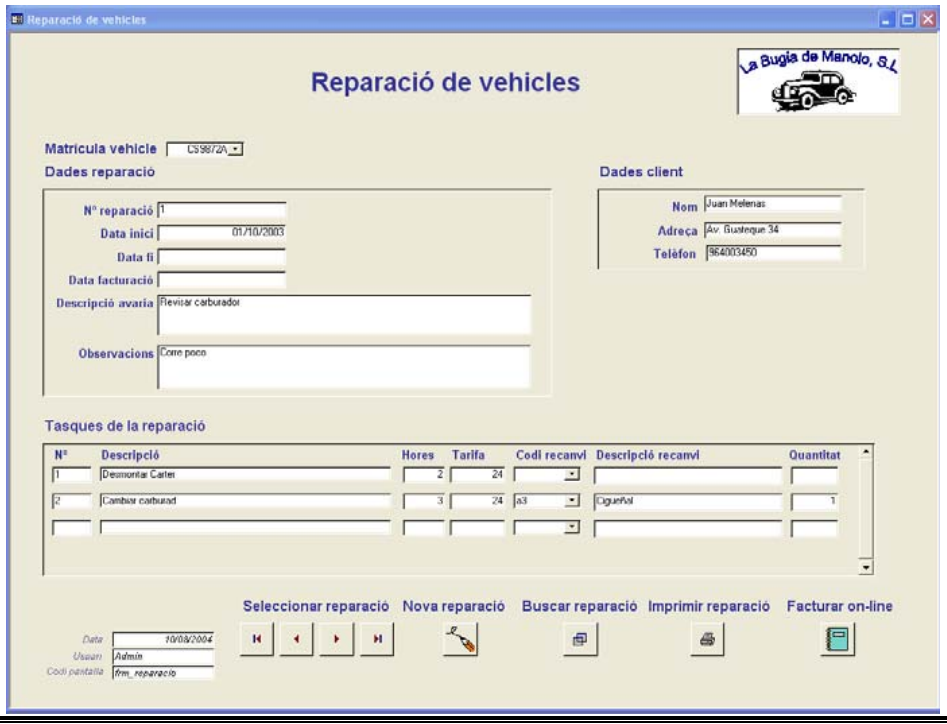
1. Busqueu en el diccionari els següents conceptes: interfície, pantalla, informe, mòdul, acoblament, cohesió, eficàcia, eficiència.
2. Dissenyeu una pantalla per a un procés que done d'alta els clients d'un vídeo club i permeta introduir el préstec d'una pel·lícula.
3. Dissenyeu un informe que proporcione la informació de tots els préstecs fets en un vídeo club, 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 de la pantalla d'exemple del cas del taller 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 al 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.

7. Dissenyeu un diagrama d'estructura per al cas del taller. Especifiqueu-ne els processos.
8. Confeccioneu deu preguntes d'examen per a aquest tema.

Cas pràctic: Taller de reparació de vehicles

Disseny de pantalles, exemple



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 sistema informàtic, la construcció i posada en marxa.
- Comprendre la finalitat que tenen les proves del sistema i ser capaç d'utilitzar les diferents estratègies de prova per tal de poder planificar adequadament les proves.
- Conèixer les diferents tasques que es poden 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 dins de la posada en marxa del sistema informàtic.

- Comprendre les modificacions que s'han de realitzar en les fases de desenvolupament d'un sistema informàtic, en el cas d'optar per la instal·lació d'un paquet estàndard de programari.

6.1. Introducció

En aquest tema es tracta la fase de construcció i posada en marxa del desenvolupament d'un sistema informàtic. El propòsit d'aquesta fase és finalitzar el disseny del sistema, i completar totes les activitats necessàries per a obtenir un programari fiable i adequat que puga ser instal·lat en l'entorn de l'organització i funcionar correctament. Aquesta fase comporta el volum de treball més considerable de totes. A més, és la fase en la qual comencen a treballar els analistes i programadors menys experts.

Per al correcte desenvolupament de les activitats d'aquesta fase cal preparar tres tipus d'entorn:

- 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 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 es crea un de separat per a les proves de rendiment final.

A partir de les especificacions obtingudes en la fase d'anàlisi i de disseny del sistema s'ha de completar el disseny detallat dels mòduls que formen part del sistema, i realitzar la programació i les proves dels diferents mòduls del sistema. Al mateix temps es desenvolupen els procediments d'usuari, i el material i la metodologia necessària per a la formació dels futurs usuaris del sistema, i la documentació que constitueix els manuals d'ús d'aquest. Una vegada provat el sistema completament i convertida la informació necessària existent en l'organització, el sistema està preparat per a entrar en funcionament. És necessari tenir en compte si el sistema pot ser desenvolupat totalment a mida, mitjançant la instal·lació d'un paquet del mercat, o bé de forma compartida a mida i amb la instal·lació d'un paquet de mercat. En cadascun d'aquests tres casos és necessari tenir en compte diferents tasques

que s'adapten a cadascuna de les possibilitats. La visió d'aquest tema se centra en el desenvolupament de sistemes a mida.

6.2. Activitats de la construcció i posada en marxa del sistema

Com es va comentar al tema anterior, abans de començar una nova fase del desenvolupament d'un sistema informàtic cal tenir en compte els resultats obtinguts en les fases anteriors i comprendre'ls en la seua totalitat. Per tant, la primera activitat d'aquesta fase és realitzar una revisió de l'anàlisi i del disseny del sistema realitzat amb anterioritat. També s'han de tenir en compte factors referents al temps transcorregut entre una fase i l'anterior, i altres com els possibles canvis en l'equip de desenvolupament, per tal de fer una revisió més o menys extensa.

ACTIVITATS i TASQUES	DESCRIPCIÓ-OBJECTIU
1. 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 amb anterioritat.
2. Preparació de l'entorn de desenvolupament i prova 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 a poder programar, i provar els diferents mòduls del sistema informàtic a construir i instal·lar-los de forma adequada.
3. Desenvolupament dels components de programari del sistema Disseny detallat Programació Disseny i realització de les proves Preparació de la conversió	Completar el disseny detallat a partir del disseny tècnic obtingut en l'etapa anterior, i a partir d'aquest generar el codi i les proves pertinents.
4. Preparació de l'entorn d'explotació Instal·lació del maquinari, programari i les bases de dades necessàries per a l'ús del sistema per part dels usuaris	Determinar els recursos de maquinari, programari i bases de dades necessaris per al funcionament correcte del sistema informàtic i instal·lar-los de forma adequada.
5. Desenvolupament dels procediments d'usuari i formació Desenvolupament dels procediments d'usuari 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 informàtic, i com s'ha de realitzar la formació del usuaris.
6. Posada en marxa del sistema Conversió Entrega del producte a l'usuari	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.

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

En la taula 6.1 es mostra un resum de les principals activitats que s'han de dur a terme en la fase de construcció i posada en marxa, com també les principals tasques que s'han de realitzar en cadascuna de les activitats i que es descriuen amb més detall als següents apartats.

6.3. Preparació de l'entorn de desenvolupament i prova

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 de treball, 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.

6.3.1. 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 estan 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 a realitzar la codificació dels mòduls del sistema nou.

6.3.2. 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'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.

6.3.3. 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.

6.4. Desenvolupament dels components de programari del sistema

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.

6.4.1. Disseny detallat

El disseny detallat té com a propòsit la definició de la lògica del codi del mòduls definits en la fase de disseny del sistema informàtic. Es poden utilitzar diferents tècniques: algorismes, diagrames de Warnier, diagrames de flux de programes, de Jackson, etc. S'ha de tenir en compte que el disseny detallat dels programes facilite la programació, prova i manteniment posterior.

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 degudes, en part, a les característiques del maquinari i del programari necessari per al seu funcionament. No obstant, les modificacions han de ser avaluades i ha de considerar-se l'esforç afegit que suposen, com també els beneficis que proporcionen i si són imprescindibles.

També és necessari revisar i finalitzar amb detall el disseny de la base de dades i fitxers auxiliars, com també els diferents informes i pantalles que formen part del sistema, per a tenir-los completament definits en la tasca següent que és la programació. A vegades durant el disseny detallat dels programes també s'identifiquen requisits de dades nous que afecten el disseny de la base de dades. Qualsevol possible modificació ha de ser incorporada abans de començar la programació, quan els canvis podrien suposar un cost afegit.

6.4.2. Programació

Habitualment és en aquesta tasca en la qual els enginyers informàtics amb poca experiència comencen a treballar. A partir de les definicions del programes que identifiquen les principals entrades i eixides de cadascun d'aquests, s'encarreguen de generar el codi font i els executables, de documentar cadascun dels programes segons els estàndards establerts per al desenvolupament i també de realitzar-ne la prova.

6.4.3. 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 hi ha ja informació mecanitzada pot realitzar-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 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.

6.4.5. 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 sistema d'informació 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 a 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 de realitzar 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 a 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 són 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.

6.5. Preparació de l'entorn d'explotació

Aquesta activitat de la fase de construcció i 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 SGBD. 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 de l'explotació.
- Instal·lació del programari desenvolupat per al sistema informàtic.
- Creació i entrega de manuals d'usuari. Aquests manuals han de permetre obtenir tot el rendiment possible del sistema als futurs usuaris. 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.

6.6. Desenvolupament dels procediments d'usuari i formació

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

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 de mostrar als usuaris 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 a realitzar la formació.

Les especificacions funcionals són la principal informació d'entrada per al desenvolupament dels procediments d'usuari. A més, els procediments d'usuari han d'incloure informació sobre quines funcions es poden realitzar i quines accions es poden 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:

1. **Desenvolupar procediments d'usuari:** s'han 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 usar tècniques de modelització empresarial, com per exemple l'IDEF0, que es pot veure a la figura 6.1.
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.

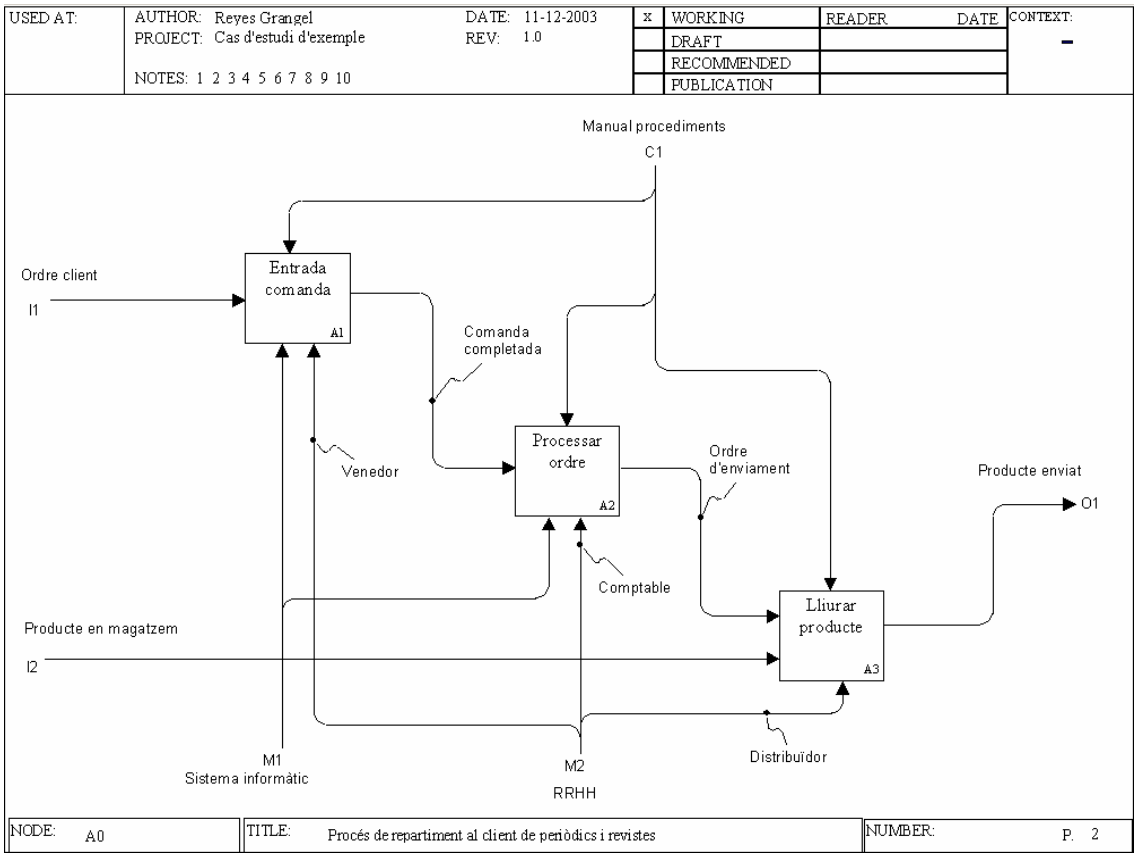


Figura 6.1. IDEF0 d'un procés

3. **Desenvolupar material de formació per als usuaris:** s'han de desenvolupar exemples d'ús de les diferents funcions que poden realitzar-se amb el sistema, perquè els usuaris puguen practicar durant el període de formació.
4. **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.

6.7. 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.

6.7.1. 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 (veure la figura 6.2).

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 amb l'estat del sistema en el moment de posar-lo en funcionament.

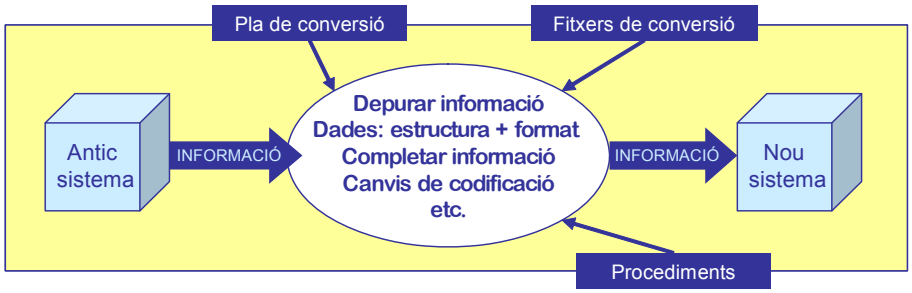


Figura 6.2. Esquema de la conversió

6.7.2. Entrega del producte a l'usuari

Es poden seguir quatre estratègies per a l'entrega del producte a l'usuari. L'entrega del producte ha de realitzar-se el més ràpid possible, hi ha situacions en què és necessari utilitzar una estratègia d'entrega en particular, però sempre han de considerar-se els avantatges i els problemes que poden ocasionar cadascuna d'aquestes. Aquestes estratègies són:

- **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 a poder realitzar els processos en aquests dos sistemes. En algunes ocasions, que els usuaris sàpiguen que poden tornar al sistema antic fa que es produïsca 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 a 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 funcionar bé des de la primera conversió.

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

6.8. 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.

6.8.1. Enfocaments de les proves

A més, per al disseny de casos de prova hi ha tres enfocaments, els quals no són excloents, sinó complementaris:

- **Enfocament estructural o de caixa blanca:** se centra en els requisits estructurals del sistema. Es tracta de comprovar que cada funció és totalment operativa, amb la selecció de casos de prova que només tinguen en compte l'estructura interna del programa i els diferents camins lògics existents. Les proves que segueixen aquest enfocament són responsabilitat del programador i poden aplicar-se tècniques pròpies de programació a més de les consideracions anteriors.

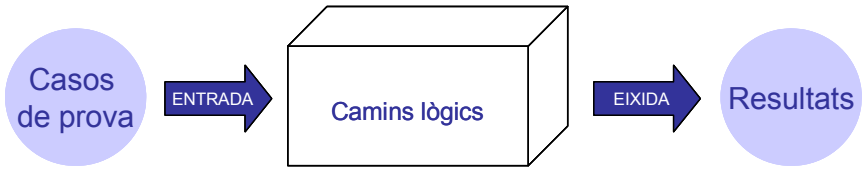


Figura 6.3. Enfocament estructural o de caixa blanca

- **Enfocament funcional o de caixa negra:** se centra en els requisits funcionals. És a dir, s'ha de comprovar que els programes funcionen de forma correcta, sense preocupar-se de la seua forma o estructura lògica interna. Es tracta de seleccionar casos de prova tenint en compte els requisits funcionals que ha de complir el sistema, sense fixar-se en les estructures lògiques que s'han utilitzat en els programes.

- **Enfocament aleatori:** es tracta d'utilitzar, per exemple, models estadístics per tal de definir els casos de prova.

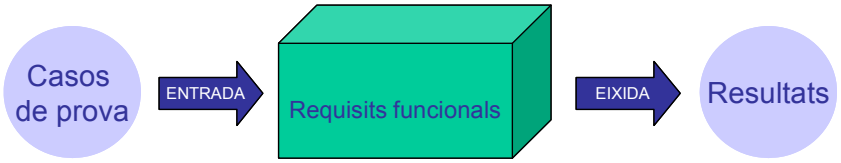


Figura 6.4. Enfocament funcional o de caixa negra

6.9. Estratègies d'aplicació de les proves

Una bona estratègia de prova ha d'estar formada per una planificació d'aquesta, el disseny dels casos de prova i l'obtenció, documentació i avaluació dels resultats obtinguts. Hi ha diferents estratègies d'aplicació de les proves utilitzades i descrites en diferents metodologies, però totes tenen en comú:

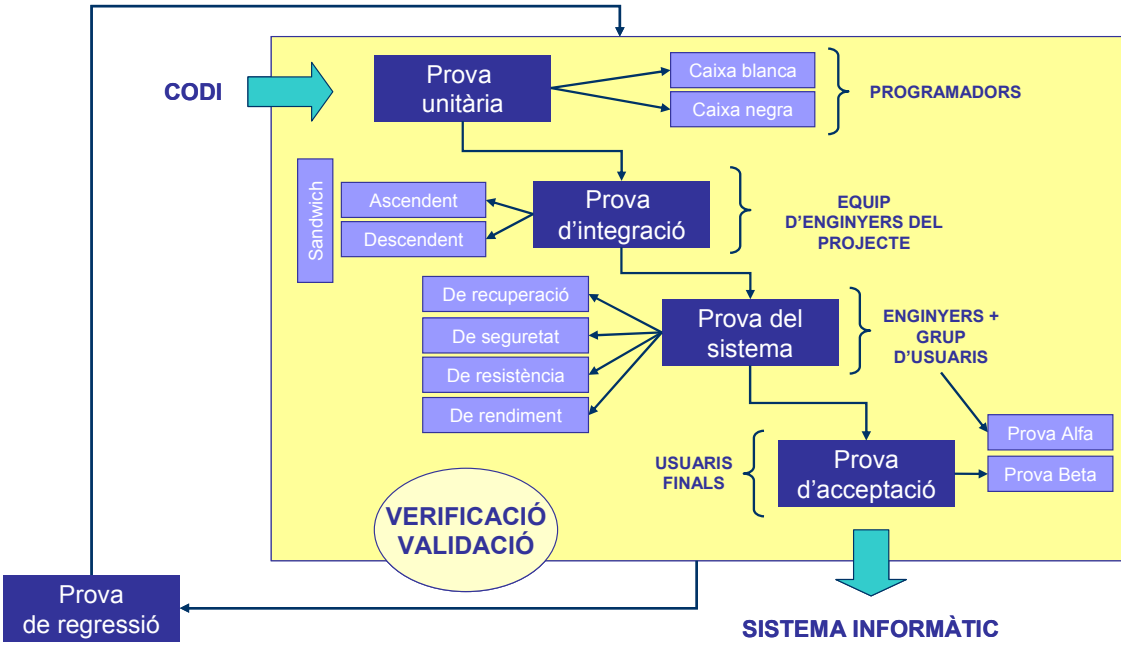


Figura 6.5. Estratègia d'aplicació de les proves

- Tota prova es realitza de dins cap a fora, és a dir, des de mòduls més menuts fins al sistema complet.
- En diferents punts són adequades diferents tècniques de prova.
- La prova la realitza la persona o grup de persones que desenvolupen el sistema o un grup de prova independent per a grans projectes. El programador sempre és responsable de provar les unitats independents o programes, i assegurar-se que cadascuna d'aquestes du a terme la funció per a la qual ha sigut dissenyada de forma correcta.

Tenint en compte que les estratègies d'aplicació de les proves sempre comencen pels mòduls més menuts fins arribar al sistema complet, es poden definir els següents tipus de proves (veure figura 6.5) i que es descriuen amb més detall a continuació.

Pel que fa la importància de les proves, existeixen paradigmes de desenvolupament de programari orientats a les proves, com és el TDD (*Test Driven Development*) (Beck, 2003).

6.9.1. Proves unitàries

Les proves unitàries són les proves que verifiquen i validen cadascun dels programes individualment. Es poden aplicar per al disseny dels casos de prova que s'utilitzaran en aquestes proves, tant l'enfocament de caixa blanca com el de caixa negra. El primer, serveix per a determinar si és correcta la lògica del programa. I el segon, per a comprovar que el programa compleix la funcionalitat per a la qual es va dissenyar. Les tasques que s'han de realitzar en les proves d'unitat són:

- **Supervisar** i assistir la codificació de programes: en el cas de tenir equips de programació de diverses persones és necessari designar una persona amb més experiència per a supervisar el treball dels programadors. Aquests supervisors han de conèixer perfectament el disseny funcional i tècnic realitzat i proporcionar assistència tècnica, per a assegurar que se segueixen les línies estàndards establertes per a la programació i que es codifiquen i proven els programes de forma adequada.
- **Programar i provar els mòduls individualment:** durant la programació el programador ha d'identificar els casos de prova que han de ser provats, tant en la prova d'unitat com en la d'integració.
- **Revisar el programa font:** després d'obtenir una compilació sense errors és aconsellable realitzar una revisió del programa, per a verificar que es compleixen els estàndards i la funcionalitat per a la qual ha sigut definit el programa.

6.9.2. Proves d'integració

Són les proves que verifiquen i validen el funcionament de tots els mòduls que constitueixen el sistema complet de forma conjunta. Per tant, s'ha de considerar l'agrupació dels diferents mòduls i programes realitzats com també l'acoblament de les interfícies. L'objectiu de la prova d'integració té dues parts. La primera és assegurar que les unitats de programació programades funcionen correctament com un tot. La segona és assegurar que els usuaris poden treballar amb el sistema correctament i que aquest cobreix els requeriments inicialment identificats. Les tasques que s'han de realitzar en les proves d'integració són:

- **Realitzar el test d'integració:** aquesta prova es realitza per l'equip d'analistes i programadors del projecte i serveix per a determinar si els diferents programes i mòduls interaccionen de forma correcta. Els errors localitzats han de ser documentats i corregits.
- **Comprovar els resultats esperats:** mentre la prova d'integració es realitza, tots els resultats obtinguts es verifiquen i analitzen per a localitzar qualsevol possible discrepància. Els problemes trobats tant de funcionament com de procés han de ser analitzats i corregits fins a obtenir un sistema correcte i manejable.

6.9.3. Proves del sistema

Són les proves que determinen si el sistema compleix els requisits per als quals havia estat dissenyat. En aquesta prova cal simular totes les funcions que el sistema ha de realitzar, tant d'entrada de dades com de processos *batch* i d'eixides d'informes i consultes. A més, és necessari verificar les eixides i la informació registrada en els fitxers, per a comprovar que els requisits es compleixen adequadament.

Una de les tasques que cal realitzar en les proves del sistema és **r**ealitzar la prova d'usuari: els usuaris involucrats en el sistema han de realitzar una prova exhaustiva d'aquest. Per a això a vegades és necessari formar un reduït grup d'usuaris, prèviament a l'acabament del sistema. Aquesta prova ha d'assegurar que el sistema cobreix els objectius proposats. Per a això és necessari realitzar un pla de proves, que cobrisca completament tota la funcionalitat del sistema. Quan la prova d'usuari està completada i tots els errors o modificacions necessàries han sigut documentats i corregits, el sistema està preparat per a la conversió i la posada en marxa.

6.9.4. Proves d'acceptació

Són les proves que serveixen perquè l'usuari final puga verificar que el sistema s'ajusta als requisits que s'havien fixat.

6.9.5. Prova de regressió

L'objectiu de les proves de regressió és eliminar l'efecte ona, és a dir, comprovar que els canvis sobre un component d'un sistema d'informació no introdueixen un comportament no desitjat, o errors addicionals, en altres components no modificats. Les proves de regressió s'han de dur a terme cada vegada que es fa un canvi en el sistema, tant per a corregir un error com per a realitzar una millora. No és suficient provar només els components modificats o afegits, o les funcions que en aquests es realitzen; a més a més, també és necessari controlar que les modificacions no produïsquen efectes negatius sobre el sistema o altres components. Normalment, aquest tipus de proves impliquen la repetició de les proves que ja s'han realitzat prèviament, amb la finalitat d'assegurar que no

s'introdueixen errors, que puguen comprometre el funcionament d'altres components que no han segut modificats, i confirmar que el sistema funciona correctament una vegada realitzats els canvis. El responsable de realitzar les proves de regressió serà l'equip de desenvolupament junt al tècnic de manteniment, qui, al mateix temps, serà responsable d'especificar el pla de proves de regressió i d'avaluar-ne els resultats. Les proves de regressió poden incloure:

- La repetició dels casos de proves que s'han realitzat anteriorment i estan directament relacionats amb la part del sistema modificada.
- La revisió dels procediments manuals preparats abans del canvi, per a assegurar que es manté correctament.
- L'obtenció impresa del diccionari de dades, de forma que es comprove que els elements de dades que han sofert un canvi són correctes.

6.10. Opcions per a la implantació d'un sistema informàtic

A l'hora d'implantar un sistema informàtic és necessari fer un estudi previ dels processos que es duen a terme en l'empresa en la qual es vol implantar el sistema. L'estudi no només ha de recollir la forma actual de treballar en l'empresa, sinó també diferents propostes de millora que puguen sorgir en l'anàlisi dels processos. Una vegada realitzat aquest estudi ja es pot realitzar la fase de definició de requisits del sistema informàtic, per tal de determinar quins són els objectius, abast i restriccions que ha de seguir el nou sistema informàtic. Després, i depenent de les necessitats analitzades, s'ha d'escollir entre diferents estratègies per a implantar el sistema informàtic; aquestes estratègies es poden resumir en tres:

- Desenvolupar un sistema informàtic totalment a mida.
- Realitzar la instal·lació d'un paquet de programari estàndard del mercat.
- Adoptar una solució mixta, amb la combinació de la instal·lació d'un paquet de mercat amb el desenvolupament de determinats components del sistema.

En cadascun d'aquests casos és necessari tenir en compte que les activitats i tasques que es realitzen en les diferents fases de desenvolupament d'un sistema informàtic s'han adaptar a cadascuna de les possibilitats. La visió dels temes anteriors, en els quals s'han estudiat aquestes fases, s'ha centrat en el desenvolupament de sistemes a mida. En aquest annex s'estudien les diferents actuacions que s'han de dur a terme en aquestes fases en les dues opcions restants.

En el cas d'adquirir un paquet de programari estàndard dels existents en el mercat, bé per a implantar-lo en la seua totalitat, o bé per a implantar algunes de les seues funcionalitats i fer el desenvolupament

a mida d'altres, s'ha de fer una selecció entre els productes existents en el mercat a partir de la definició de requisits obtinguda. Algunes de les activitats i tasques que s'han de dur a terme en la selecció de programari són:

- Seleccionar proveïdors potencials.
- Avaluar productes i proveïdors.
    - Desenvolupar matriu de selecció.
    - Avaluar matriu, costos i recursos.
    - Desenvolupar proposta d'adquisició de programari.

6.10.1. Instal·lació d'un programari de mercat

Aquesta estratègia d'implantació de programari s'adopta quan existeix en el mercat un paquet de programari estàndard que s'adapta de manera adequada als requisits de l'empresa en qüestió. Per tant en aquest cas no és necessari realitzar les fases d'anàlisi, disseny i construcció, ja que en el procés de selecció d'un paquet de mercat s'han avaluat diferents opcions comercials, i se n'ha trobat una que s'adapta als requisits de l'empresa i per tant no cal construir el sistema informàtic en la seua totalitat.

ACTIVITATS i TASQUES	DESCRIPCIÓ-OBJECTIU
1. Desenvolupament del pla d'instal·lació Identificació de tasques i subtasques Estimació de recursos Planificació de tasques i recursos	Realitzar una estimació dels recursos necessaris per a realitzar aquesta fase i planificar i seqüenciar les activitats de la fase d'acord amb els recursos estimats.
2. Preparació de l'entorn d'explotació i prova Instal·lació del maquinari, programari i les bases de dades necessàries per a l'ús del sistema per part dels usuaris Parametritzar i personalitzar el sistema Desenvolupar mòduls o interfícies amb altres sistemes Preparació de l'entorn de prova Disseny i realització de les proves Preparació de la conversió	Determinar els recursos de maquinari, programari i bases de dades necessaris per a la prova i el correcte funcionament del sistema informàtic i instal·lar-los de forma adequada.
3. Desenvolupament dels procediments d'usuari i formació Desenvolupament dels procediments d'usuari Desenvolupament del pla de formació dels usuaris Formació dels usuaris	Definir com s'han de dur a terme els processos de l'empresa per part dels usuaris mitjançant el nou sistema informàtic i com s'ha de realitzar la formació dels usuaris.
4. Posada en marxa del sistema Conversió Lliurament del producte a l'usuari	Traspassar les dades necessàries de l'antic al nou sistema, per a posar-lo en funcionament real i assegurar l'acceptació de l'usuari.

Taula 6.2. Activitats de la posada en marxa del sistema

En alguns casos pot ser útil fer una anàlisi a alt nivell, per determinar si l'aplicació seleccionada s'adapta completament a les necessitats de l'empresa. Pel que fa a la fase de posada en marxa, les activitats que s'han de realitzar es veuran afectades segons es mostra a la taula 6.2.

6.10.2. Solució mixta

La tria d'una solució mixta s'adopta quan hi ha alguns requisits que són coberts per algun paquet estàndard del mercat, però n'hi ha d'altres que no. Pel que fa a la part d'instal·lació del paquet estàndard, els passos que cal seguir són els mateixos que els explicats en l'apartat anterior. En canvi, per als requisits que no són coberts per l'aplicació estàndard i s'han de desenvolupar, és necessari realitzar les diferents fases vistes en tota la seua extensió: anàlisi, disseny, construcció i posada en marxa del sistema. En la fase de posada en marxa, s'han de tenir en compte les activitats necessàries per tal d'instal·lar i provar el paquet de mercat i les activitats d'integració d'aquest paquet amb els components desenvolupats.

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 del desenvolupament d'un sistema informàtic, per tant el seu objectiu és, a partir del resultats obtinguts en la fase anterior de disseny, obtenir un sistema informàtic que siga operatiu. 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 i finalment la posada en marxa del sistema.

En la part de desenvolupament dels components de programari sempre s'ha dedicat més esforç a la codificació que a la prova dels programes. En aquest tema s'ha estudiat la importància de les proves per a obtenir un sistema de qualitat i s'han vist tècniques diferents per al disseny de casos de prova, com també diferents estratègies per a la seua aplicació.

Activitats complementàries

1. Busqueu en el diccionari els següents conceptes: prova, defecte, fallada, error, qualitat, conversió.
2. Dissenyeu els casos de prova segons l'enfocament de la caixa blanca per tal de provar el següent programa en C.

```
/* Mostra si el pH d'una dissolució és àcid, bàsic o neutre */
```

```
/* ENTRADA: Un valor de pH entre 0 i 14 */
/* SORTIDA: Missatge indicant si el pH d'una dissolució és àcid, bàsic o neutre */
#include <stdio.h>
void main ()
{
 float pH;
 printf ("Introduïu el pH de la vostra dissolució i us diré si és àcid o bàsic: ");
 scanf ("%f", &pH);
 while (pH < 0 || pH > 14)
 {
   printf ("Açò no és un valor de pH vàlid, introduïu-ne un entre 0 i 14: ");
   scanf ("%f", &pH);
 }
 if (pH > 7)
   printf ("La vostra dissolució és BÀSICA.\n");
 else
   if (pH == 7)
     printf ("La vostra dissolució és NEUTRA.\n");
   else
     printf ("La vostra dissolució és ÀCIDA.\n");
}
```

3. Realitzeu un pla de conversió per al cas pràctic del "Taller de reparació de vehicles".
4. Confeccioneu deu preguntes d'examen per a aquest tema.

Cas pràctic: Taller de reparació de vehicles

Casos de prova segons l'enfocament de la caixa negra

A continuació es proporciona un exemple de casos de prova desenvolupat per al cas pràctic del "Taller de reparació de vehicles", seguint l'enfocament de la caixa negra per al seu desenvolupament. Proveu si es pot...

Mantenir (donar d'alta, de baixa i modificar) clients/cotxes

Donar d'alta un client sense tenir assignat un cotxe (s'ha de poder).

Donar d'alta un cotxe sense tenir assignat un client (no s'ha de poder).

Assignar un cotxe a dues persones diferents (no s'ha de poder segons el model conceptual, en la realitat pot passar que un mateix cotxe es venga i passe a ser d'un altre client, penseu com canviaríeu el MCD per tal que el sistema ho permeta).

Donar dues vegades d'alta un mateix client (no s'ha de poder).

Registrar dos cotxes amb la mateixa matrícula (no s'ha de poder).

Tenir un client amb més d'un cotxe assignat (s'ha de poder).

Gestionar reparacions (obrir i tancar reparacions, assignar tasques previstes i realitzades incloent MO i recanvis)

Crear reparacions no assignades a cap client/cotxe (no s'ha de poder).

Crear dues reparacions amb el mateix número de reparació (no s'ha de poder).

Fer que la data de finalització de la reparació siga anterior a la data d'inici (no s'ha de poder).

Mantenir (donar d'alta, de baixa i modificar) tasques (s'ha de poder).

Introduir tasques realitzades amb una data d'inici posterior a l'actual (no s'ha de poder).

Introduir tasques realitzades sense MO ni recanvis (no s'ha de poder).

Introduir tasques realitzades sense MO però sí recanvis (no s'ha de poder).

Introduir tasques realitzades amb MO però sense recanvis (s'ha de poder).

Introduir tasques realitzades amb MO i recanvis (s'ha de poder).

Assignar a una tasca realitzada un mecànic o peça inexistent (no s'ha de poder).

Facturar (*on-line* i *batch*)

Crear una factura sense alguna de les dades de la capçalera (data, client, cotxe, reparació) (no s'ha de poder).

Crear una factura sense línies de factura (no s'ha de poder).

Crear una factura d'un client/cotxe/reparació que no existisca (no s'ha de poder).

Crear una factura d'una reparació no tancada (no s'ha de poder).

Facturar més d'una vegada una mateixa reparació, per a la qual cosa es pot realitzar una factura en línia d'una reparació i comprovar que aquesta no es torna a facturar de forma *batch* (no s'ha de poder).

Rectificar una factura comptabilitzada (no s'ha de poder).

Generar informes

Obtenir els informes demanats de forma correcta (s'ha de poder).

En general

Introduir dades en formats incorrectes, dades errònies o deixar dades obligatòries en blanc (no s'ha de poder).

Deixar dades obligatòries en blanc (per exemple, NIF per a donar d'alta un client) (no s'ha de poder).

Duplicar dades que han de ser úniques (no s'ha de poder).

Capítol 7

Manteniment i evolució

Objectius

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

- Assimilar els principals conceptes relacionats amb el manteniment dels sistemes informàtics.
- Conèixer els diferents tipus de manteniment del programari que existeixen.
- Conèixer la problemàtica del manteniment del programari i les possibles accions que es poden aplicar per tal de solucionar-la.

7.1. Introducció

El manteniment de sistemes informàtics és considerat de poc d'interès i rellevància a l'àmbit empresarial, tan pels responsables de la gestió com pels professionals dels sistemes d'informació (Piattini, 2000). Però és una de les activitats més habituals en el treball de molts enginyers en informàtica.

Com s'ha detallat en el capítol d'introducció, el manteniment no és una fase de la construcció del sistema. El manteniment apareix quan es parla del cicle de vida, ja que representa el conjunt d'activitats i tasques que permetran que el sistema informàtic evolucione de manera adequada als canvis de l'entorn, tecnològics i econòmics, a noves necessitats. Si és du a terme un manteniment adequat, de manera semblant al que passa amb altres productes d'enginyeria, el producte podrà tenir una vida útil rendible i adequada a l' inversió feta en la seua construcció.

El manteniment, per tant, comença quan s'han realitzat totes les activitats del desenvolupament d'un sistema informàtic i s'entrega el resultat a l'usuari.

Al començament, durant el període d'inici de l'ús del sistema informàtic, les tasques de manteniment estan relaciones amb la correcció dels errors i carències que poden detectar els usuaris. Posteriorment, en un termini mitja d'ús, les tasques de manteniment estan relacionades amb la incorporació de noves funcionalitats, a partir de la identificació per part dels usuaris de nous requisits. L'ús del sistema durant un període permet als usuaris comprendre la funcionalitat implementada i detectar millores respecte als requisits que es van identificar a l'inici del projecte de desenvolupament del sistema. També poden sorgir necessitats de canvis per adaptar-se a noves tecnologies que aporten beneficis al sistema, o complir amb noves lleis i normatives.

Es pot deduir, per tant, que el manteniment té un casuística i una problemàtica variada. Per a dur a terme manteniment de sistemes és necessari conèixer tecnològicament el producte de programari, com ha sigut desenvolupat, i comprendre i aprendre qualsevol dels aspectes que poden implicar la necessitat d'incorporar un canvi, una millora o una nova funcionalitat. Per tant, a pesar de ser una activitat poc valorada, és una activitat en la qual es requereixen competències i capacitats altes i l'aplicació de mètodes formals que garantitzen la qualitat dels resultats.

7.2. Manteniment del programari

Existeixen en la bibliografia diferents definicions de manteniment:

- Procés de modificació d'un producte de programari després de l'entrega al client per a corregir defectes, millorar el rendiment o altres atributs, o per a adaptar-lo a un entorn canviant (IEEE, 1993a).
- El programari sofreix modificacions de codi i documentació associada, com a conseqüència d'un problema o de la necessitat de millora. L'objectiu és modificar el programari i conservar la seua integritat (ISO/IEC, 12207 1995).

Aquestes definicions tenen en compte que quan es comença a utilitzar un nou producte de programari es poden trobar:

- Errors de programació: el programa no fa correctament allò que es demana.
- Errors de disseny: el programa no fa les coses o els processos com s'havien demanat.
- Nous requisits d'usuari: bé perquè han sorgit noves necessitats en l'empresa o perquè ha canviat l'entorn, i cal que els programes s'adapten als nous requisits.

Encara que als primers capítols es representa el manteniment com una fase que comença al final de la construcció i posada en marxa del sistema (com es veu a la figura 7.1) el manteniment no és independent del desenvolupament.

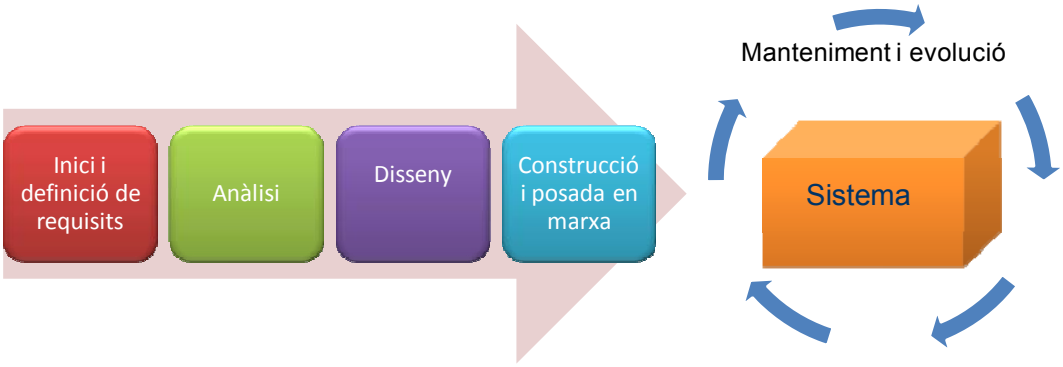


Figura 7.1. Cicle de vida del programari

Segons la idea de Lehman (1997) el manteniment és realment un desenvolupament evolutiu i els mètodes de manteniment només tenen èxit si s'involucren amb el desenvolupament (veure la figura 7.2).

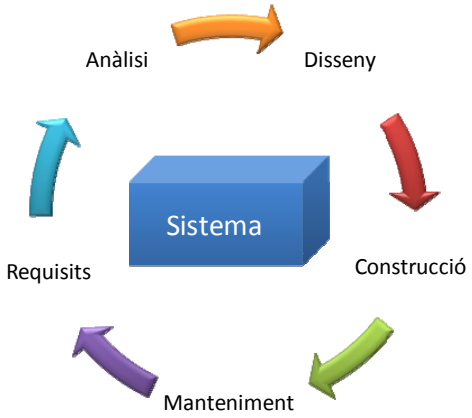


Figura 7.2. Integració del manteniment en el desenvolupament del sistema informàtic

El manteniment ha de ser considerat com un procés d'enginyeria necessari per al funcionament correcte i l'evolució adequada del producte desenvolupat. Per tant és necessari aplicar mètodes i tècniques d'enginyeria del programari, de manera semblant a com es fa quan es construeix un nou sistema.

Segons dades empíriques, les tasques de manteniment suposen la major part del cost del cicle de vida del programari. El cost de manteniment d'un sistema informàtic al llarg de la seua vida útil suposa més del doble dels costos del seu desenvolupament. Aquests costos es poden dividir en:

- Costos tangibles, els quals són difícils de calcular perquè és difícil preveure'ls.
- Costos intangibles, els quals suposen:
    - Pèrdua d'altres oportunitats de desenvolupament.
    - Insatisfacció del client.
    - Reducció global de la qualitat del producte.
    - Retard en altres projectes de desenvolupament.

7.3. Tipus de manteniment

Tenint en compte les definicions de manteniment, els motius de la necessitat de dur-lo a terme, i la finalitat que pot ser correcció, millora o adaptació, es distingeixen els següents tipus de manteniment (veure la figura 7.3) (Piattini, 2000):

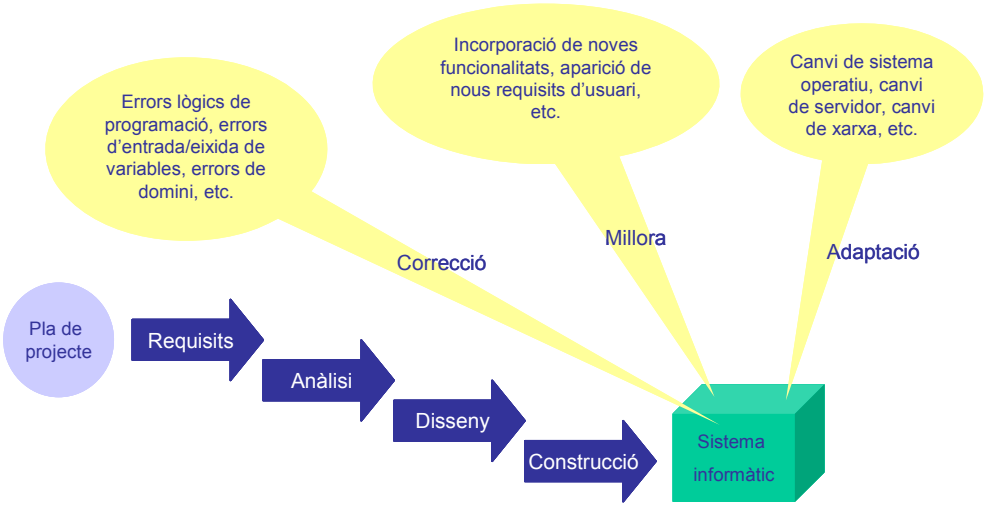


Figura 7.3. Tipus de manteniment

- **Manteniment correctiu:** per a corregir defectes en el programari.
- **Manteniment perfectiu:** per introduir alguna millora en el programari
- **Manteniment preventiu:** per a millorar el rendiment o altres propietats del programari.
- **Manteniment adaptatiu:** per a adaptar el programari a un canvi de l'entorn de maquinari o de programari.

7.3.1. Manteniment correctiu

El manteniment correctiu consisteix a localitzar i eliminar els possibles defectes o errors dels programes, una vegada han sigut provats i lliurats a l'usuari final.

El principal problema d'aquest tipus de manteniment és localitzar les especificacions inicials per tal de conèixer l'objectiu inicial del sistema i poder corregir els defectes o errors del codi. S'ha de distingir entre defecte i fallada, un defecte del sistema és una característica que pot produir una determinada fallada. En canvi, la fallada ocorre quan el comportament del sistema és diferent de l'esperat segons els requisits (Piattini, 2000). Els tipus de fallades del programari poden ser:

- De processament.
- De rendiment.
- De programació.
- De documentació.

7.3.2. Manteniment perfectiu

El manteniment perfectiu es pot definir com la millora de les funcionalitats existents del programari o l'afegiment de noves sol·licitades pels usuaris. Aquest tipus de manteniment té una casuística molt variada, de manera que qualsevol millora en el funcionament, no reflectida en els requisits, que es realitze en el programari estaria inclosa en aquesta categoria.

7.3.3. Manteniment preventiu

El manteniment preventiu consisteix a modificar el programari per a millorar alguna de les seues propietats (mantenibilitat, qualitat, etc.), de manera que no s'alteren les especificacions funcionals i amb la finalitat que el codi siga més fàcil de mantenir.

7.3.4. Manteniment adaptatiu

El manteniment adaptatiu consisteix a modificar el programari per a adaptar-lo a canvis en l'entorn de treball bé de maquinari (*hardware*) o de programari (*software*). L'usuari no veu un canvi directe en l'operativitat del sistema, però es produeix quan és necessari fer aquests canvis en el programari per tal que puga seguir funcionant amb un nou servidor o amb un altre sistema operatiu, per exemple. Per tant els canvis de l'entorn poden ser:

- De maquinari:
    - Canvi en l'arquitectura física del sistema informàtic.
- De programari:
    - Canvi en l'entorn de les dades.
    - Canvi en l'entorn dels processos.

7.4. Activitats del manteniment

El manteniment de programari suposa dur a terme les següent activitats:

- Anàlisi de l'impacte i del cost/benefici.
- Comprensió del canvis que cal realitzar.
    - Estudi de les peticions.
- Comprensió del programari.
    - Estudi de la documentació.
    - Estudi del codi.
- Disseny del canvi.
- Modificació del programari.
- Disseny i realització de proves.
- Documentació.

7.5. Dificultats i solucions inherents al manteniment

A continuació, es presenten un conjunt de lleis proposades per Lehman (1997) sobre les característiques que té el manteniment del programari:

- **Llei del canvi continu:** afirma que un sistema informàtic necessita canviar per tal de ser útil en l'entorn en el qual s'ha instal·lat.
- **Llei de l'increment de la complexitat:** afirma que l'estructura d'un sistema informàtic es deteriora a mesura que aquest evoluciona.
- **Llei de l'evolució del programa:** l'evolució d'un programa és un procés autoregulat.
- **Llei de la conservació de l'estabilitat organitzativa:** al llarg del temps de vida d'un programa, la taxa de desenvolupament del programa és aproximadament constant i independent del recursos dedicats.
- **Llei de la conservació de la familiaritat:** durant el temps de vida d'un producte de programari, l'increment en el nombre de canvis inclosos en cada versió és aproximadament constant.

A més, el manteniment del programari presenta una sèrie de problemes des del punt de vista tècnic que són:

- Absència metodològica: les metodologies s'han centrat en el desenvolupament i no en el manteniment.

- Tendència a la desestructuració: després de molts canvis els programes tendeixen a ser menys estructurats.
- Disminució de la comprensibilitat: els sistemes que tenen un gran manteniment són cada vegada més difícils de canviar.
- Poca participació dels usuaris en el procés de desenvolupament del programari.
- Falta de documentació dels canvis.
- Dificultat del seguiment dels canvis.
- Efectes secundaris no desitjats.

I altres des del punt de vista de gestió:

- La tasca de manteniment no està ben considerada pels programadors.
- Els sous i condicions laborals per a tasques de manteniment solen ser els més baixos.
- Desconeixement dels usuaris.

A banda, els canvis en el codi per raons de manteniment poden provocar l'aparició d'efectes secundaris no desitjats sobre diferents elements del sistema informàtic, com poden ser:

- Sobre el codi, produïts per:
    - Canvis en el disseny que suposen molts canvis en el codi.
    - Eliminació o modificació d'un subprograma.
    - Eliminació o modificació d'una etiqueta.
    - Eliminació o modificació d'un identificador.
    - Canvis per a millorar el rendiment.
    - Modificacions en el tractament de fitxers.
    - Modificació d'operacions lògiques.
- Sobre les dades, produïts per:
    - Redefinició de constants.
    - Modificació dels formats de registres.
    - Canvi en la grandària d'una matriu.
    - Reinicialització de punters.
- Sobre la documentació, produïts per:
    - Nous missatges d'error no documentats.
    - Taules o índex no actualitzats.
    - Text no actualitzat correctament.

Les principals solucions que es poden aplicar a aquests problemes poden ser de dos tipus:

- De gestió o organitzatives:
    - Millora dels recursos dedicats al manteniment.
    - Gestió de la qualitat.
    - Gestió estructurada del procés de manteniment.
    - Organització de l'equip humà.
    - Documentació dels canvis.
- De tipus tècnic:
    - **Reenginyeria**: consisteix en l'examen i la modificació d'un sistema per a reconstruir-lo d'una forma nova.
    - **Enginyeria inversa:** procés d'anàlisi d'un sistema per tal d'identificar els seus components i interrelacions, i crear una representació del sistema en un nivell d'abstracció més elevat.
    - **Reestructuració:** modificació del programari per fer-lo més fàcil d'entendre i canviar, o menys susceptible d'incloure errors en canvis posteriors.

Resum

El tema tracta el manteniment del programari com una fase més dins del cicle de vida del programari. A banda, presenta les diferents tipologies de manteniment de programari existents i les seues característiques, com també els principals problemes i solucions que es presenten a l'hora de dur a terme les activitats que suposen el manteniment del programari.

Activitats complementàries

1. Indiqueu per a cadascun dels exemples de manteniment proposats de quin tipus de manteniment es tracta:

Exemple	Tipus de manteniment
Es modifica un programa per a inicialitzar una variable que no ho estava i provocava un error en l'execució del programa	
Es modifica una aplicació de nòmines per a adaptar-la a una nova llei sobre l'IRPF	
Es modifica un programa que funcionava sobre el sistema operatiu Windows 98 perquè ho faça sobre LINUX	
Es modifica una aplicació de gestió comercial per a afegir-hi una nova utilitat per a l'enviament telemàtic de factures al client	
S'afegeixen comentaris a un programa per a fer-lo més llegible i mantenible	
Es canvia el tipus d'una variable de text a *integer* en un programa de càlcul de costos	
Es modifica l'estructura d'un programa per a fer-lo més llegible i mantenible	
Es modifica una aplicació de gestió de comandes per tal que els clients les puguen realitzar per Internet	
Es realitza una nova versió d'un ERP per tal que funcione sobre ORACLE	
Es modifica una aplicació comptable per tal que reculla un canvi en la llei d'aplicació de l'IVA	

Capítol 8

Gestió de projectes

Objectius

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

- Comprendre la importància d'una correcta gestió dels projectes de desenvolupament de programari.
- Conèixer les diferents etapes que s'han de dur a terme per a la gestió de projectes de desenvolupament de programari.
- Realitzar la planificació temporal d'un projecte de desenvolupament de programari, tenint en compte les activitats que cal realitzar, les seues relacions i els recursos i costos associats.
- Conèixer els diferents mètodes d'estimació de costos en el desenvolupament de programari.
- Aplicar les diferents tècniques que serveixen per a establir les relacions entre les activitats d'un projecte de desenvolupament de programari.

8.1. Introducció

En aquest tema s'analitza el desenvolupament de programari des de la perspectiva de la gestió per projectes, com una manera de gestionar de la forma més eficient possible el conjunt d'activitats que és necessari realitzar per a desenvolupar programari. En primer lloc, es descriu el que s'entén per un projecte de desenvolupament de programari i les diferents etapes que cal seguir per tal de gestionar de forma eficient un projecte d'aquest tipus. Després el tema se centra en l'etapa de planificació del projecte, amb la presentació del concepte i contingut del pla de projecte, les principals activitats que s'han d'executar i les diferents tècniques que ajuden a una bona planificació d'aquestes activitats.

8.2. Gestió d'un projecte de desenvolupament de programari

En els temes anteriors s'han analitzat les diferents fases, activitats i tasques que s'han de bastir per tal de desenvolupar un sistema informàtic; en aquest àmbit es pot diferenciar clarament entre:

- **Producte de programari:** és el resultat que s'obté del procés de programari i que es desenvolupa a petició d'altres.
- **Procés de programari:** és el conjunt d'activitats estructurals que s'han de realitzar per a desenvolupar el programari.

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. Un projecte es pot definir com un conjunt d'etapes, activitats i tasques que es duen a terme per tal d'assolir un objectiu, que implique un treball no immediat a un termini relativament llarg (Piattini, 2004). Qualsevol 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 als anomenats paquets de treball (*Work Packages*, WP).
- Tenir un objectiu final, el qual al mateix temps es pot subdividir en diferents objectius parcials.
- Produir un resultat final i un o diversos resultats parcials (*deliverables*).
- Estar condicionat pel temps, tenint una data d'inici, una data de fi i diverses dates límit o fites que es volen aconseguir.

8.3. Gestió del risc

Durant l'etapa d'extracció d'informació, com en el desenvolupament de qualsevol producte d'enginyeria, en alguns moments l'enginyer ha d'enfrontar-se amb alguns problemes que es poden resoldre millor si es tenen en compte amb certa anticipació. Fins i tot a vegades es poden prendre mesures que fan que aquests problemes no apareguen o si ho fan siga amb conseqüències mínimes.

Els riscos que comporta el desenvolupament d'un sistema d'informació estan relacionats amb les característiques particulars de l'entorn on s'ha d'implantar el sistema, les característiques i actitud dels usuaris, les característiques del mateix sistema, i de l'equip de desenvolupament, com es mostra a la figura 8.1 (Karolak, 1996).

Les característiques de l'entorn fan referència a l'estabilitat de l'organització, de l'àrea funcional on es vol implantar el sistema, la claredat dels objectius i el grau d'estructuració existent en els procediments que es desenvolupen relacionats amb el sistema. En el cas de tenir una organització pobra pel que fa al funcionament o mal organitzada, l'automatització de les àrees és arriscada. És necessari prèviament redissenyar els procediments afectats o considerar una renovació d'aquests prèviament a la informatització. Ràpids creixements de l'empresa, gestió pobre, canvis constatats en la direcció de l'empresa o de les àrees, són problemes que afecten el correcte desenvolupament de qualsevol projecte.

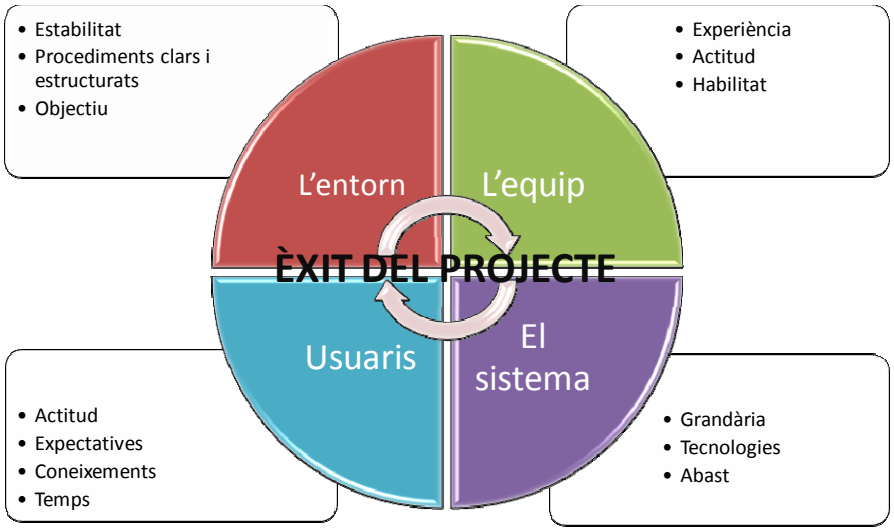


Figura 8.1. Problemes i riscos del desenvolupament de sistemes informàtics

Les característiques dels usuaris fan referència tant als coneixements en l'ús de sistemes informàtics com als coneixements de l'àrea on es vol desenvolupar el sistema. És molt important involucrar els usuaris amb l'equip de desenvolupament del sistema i que aquests tinguen coneixements de l'àrea

funcional sobre la qual vol implementar-se, com també que tinguen una actitud positiva i de col·laboració en tot moment.

Un usuari individualment no coneix el sistema en funcionament de forma global, ni tots els requisits del futur sistema. Per això cal tenir en compte diferents tipus d'usuaris depenent de les fases que es desenvolupen, a més d'usuaris de totes les àrees i nivells involucrats.

Es resumeixen en la taula 8.1 alguns dels aspectes més habituals que cal tenir en compte, per tal de minimitzar la repercussió de les característiques dels usuaris a l'hora de definir requisits.

Possible risc	Mesures per a minimitzar el risc
1. Dificultat per a obtenir detalls dels usuaris, perquè o bé no saben què és el que volen exactament, o pensen que el sistema és molt simple, o perquè temen que el sistema que realment necessiten és molt costós i necessita molt de temps per a desenvolupar-se.	Deduir els requisits a partir del sistema existent, o de solucions existents en organitzacions semblants. Incloure els usuaris en l'equip de desenvolupament, formar-los en els punts on siga necessari i guanyar-se la seua confiança. Preguntar als usuaris què no els agrada del sistema actual i què no volen en el futur. Utilitzar prototips per a formar els usuaris i per a permetre'ls realimentar els requisits.
2. Dificultats per a treballar amb els usuaris, perquè açò suposa una interrupció en el seu treball quotidià o temen el canvi imminent.	Informar (i formar) els usuaris sobre els beneficis del futur sistema. Elegir una estratègia que no necessite una intervenció constant dels usuaris. Utilitzar prototips per a formar els usuaris i permetre'ls realimentar els requisits. Assignar a l'equip de desenvolupament, sempre que siga possible, usuaris realment compromesos.
3. Dificultats de comunicació perquè les perspectives de l'enginyer informàtic i dels usuaris són molt diferents.	Evitar utilitzar termes específics informàtics (argot d'informàtica). Utilitzar gràfics, ferramentes visuals i prototips. Assignar un especialista de l'àrea funcional com a director del projecte, o un enginyer informàtic que conega bé el funcionament de l'àrea.
4. Dificultat per tal de definir els requisits, perquè és molt probable que canvien.	Estudiar organitzacions semblants per a predir els canvis que probablement es produïsquen. Utilitzar tècniques que permeten introduir modificacions de forma senzilla durant l'anàlisi.
5. Dificultats per definir les fronteres del projecte i prioritats, perquè els usuaris tenen expectatives no molt realistes.	Definir el sistema per fases i implementar inicialment només la fase de major prioritat. Realitzar informes de costos bruts en temps i diners dels requisits poc realistes, i fer que els usuaris decidisquen si el benefici a obtenir justifica el cost. Assignar usuaris amb expectatives realistes a l'equip de treball i responsabilitzar-los d'informar i consensuar amb els altres usuaris els temes de costos i beneficis.

Taula 8.1. Riscos i possibles mesures

Les característiques del sistema fan referència a l'estabilitat dels objectius del sistema, la grandària i l'abast. Els sistemes amb procediments estandarditzats i concrets tenen un menor risc en el seu

desenvolupament i és més senzill finalitzar-los amb èxit. Les aplicacions desenvolupades per a ajudar a la gestió i aplicacions estratègiques normalment tenen un risc major. En aquest últim cas hi ha més probabilitats de modificacions en els requisits inicials, ja que els procediments no estan tan estructurats i es veuen afectats per les modificacions de l'entorn. Un projecte es considera gran depenent del nombre de sistemes que s'han de reemplaçar o informatitzar, la quantitat d'interfícies o connexions amb altres sistemes, la complexitat dels requisits, i la seua duració. Més d'un sistema, o connectat amb més d'un sistema, o amb una duració de més de dotze mesos es considera gran, *large*.

Les característiques de l'equip de desenvolupament també és un factor crític per a l'obtenció del sistema amb èxit. Tant els seus coneixements tècnics i funcionals com la seua actitud poden influir notablement en l'èxit final del projecte. Si és necessària la utilització de noves tecnologies i els components de l'equip no tenen experiència el risc de fracàs augmenta. Les habilitats de comunicació de l'enginyer informàtic respecte dels usuaris també influeixen en un grau elevat, a l'hora d'identificar de forma clara i eficient els requisits; saber escoltar els usuaris i obtenir la informació veraç i útil són habilitats que faciliten el treball.

8.4. Etapes en la gestió de projectes

La gestió per projectes és típica en altres disciplines, com per exemple l'enginyeria industrial, en la qual la major part de treballs es realitzen per projectes. Per exemple, es pot realitzar un projecte per a construir una nau industrial dedicada a la fabricació de recanvis auxiliars per a l'automoció. En l'enginyeria informàtica també s'ha adoptat aquest enfocament, per tal d'aplicar tècniques de gestió de projectes que puguen ajudar a una gestió més eficient dels treballs que es realitzen en aquesta àrea. Així, 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 o per a desenvolupar programari a mida. Aquest últim tipus de projecte és l'objectiu d'estudi d'aquest tema.

Cal diferenciar dos tipus d'empreses, les que disposen d'una planificació estratègica amb estratègies i línies d'actuació ben definides pel seus directius pel que fa als clients, al producte objectiu, al màrqueting, als sistemes d'informació, etc., i aquelles en les quals no existeix cap plantejament d'aquest tipus. En el primer tipus d'empresa, dins de la planificació estratègica s'ha definit un pla de sistemes d'informació i per tant estan definits els projectes informàtics que s'ha previst realitzar en un futur i, a més, s'han determinat quines són les prioritats de l'empresa a curt i llarg termini. En el segon tipus d'empresa, no existeix cap tipus de planificació, i per tant el que passa és que els projectes informàtics sorgeixen com a conseqüència de la intenció immediata de solucionar un problema concret

de l'empresa i de la idea que la seua solució passa per la informatització. Alguns dels motius pels que s'inicia un projecte ja s'han descrit al capítol 2.

La gestió d'un projecte passa per tres etapes ben diferenciades, les quals es poden executar en una seqüència cíclica, com es pot observar en la figura 8.2.

1. Etapa de planificació del projecte: en la qual s'identifiquen les activitats que s'han de dur a terme en el projecte i s'estableix un calendari de realització d'aquestes en funció dels recursos i temps disponibles.
2. Etapa de direcció del projecte: en la qual s'organitzen, coordinen i supervisen les activitats segons l'assignació de recursos feta.
3. Etapa de control del projecte: en la qual s'analitza la realització del projecte i la planificació que en el seu dia es va fer, per tal de detectar possibles desviacions i corregir-les.

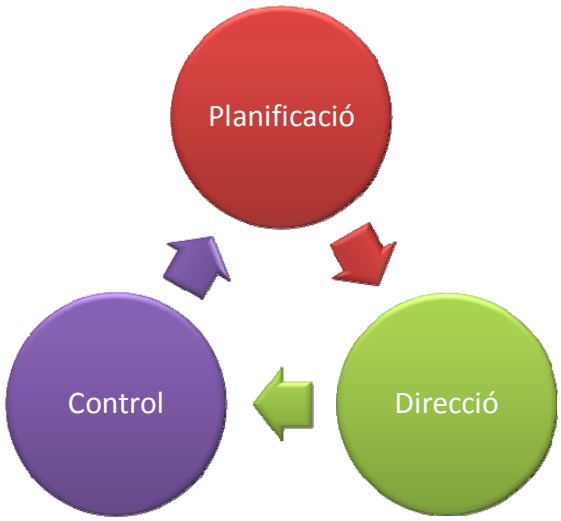


Figura 8.2. Etapes de la gestió d'un projecte

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. A continuació, es mostren les principals característiques de cadascuna de les tres etapes de la gestió de projectes de desenvolupament de programari.

8.4.1. Planificació

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, com també estimar el calendari que cal seguir i els recursos que cal assignar a cadascuna de les activitats. La planificació suposa realitzar les següents accions (Piattini, 2004):

- Fer previsions i estimar el temps que durarà cadascuna de les activitats i els costos associats.
- Definir els objectius del projecte a curt i llarg termini.
- Definir polítiques organitzatives que guien la realització del projecte.
- Determinar els processos estructurals.
- Analitzar les activitats que s'han de realitzar i la possible divisió en tasques més menudes.
- Obtenir i distribuir els recursos per a dur a terme les diferents activitats del projecte.

8.4.2. Direcció

La direcció d'un projecte de desenvolupament de programari és responsabilitat del cap del projecte, el qual es fa càrrec de la bona marxa del projecte i del compliment de les dates límit marcades i dels objectius i resultats parcials estimats. Per tal d'aconseguir-ho el cap del projecte ha d'organitzar el treball de l'equip humà del projecte, i ha de realitzar tasques de coordinació i supervisió de la feina diària de l'equip. A més, és l'encarregat d'intervenir en els possibles conflictes que puguen sorgir durant la realització del projecte. La tasca de direcció implica les següents accions (Piattini, 2004):

- Organitzar el treball.
- Motivar els membres de l'equip de treball.
- Afavorir la comunicació entre els membres de l'equip de treball.
- Exercir de cap.
- Coordinar les activitats a realitzar.
- Valorar l'execució del projecte.
- Resoldre els possibles conflictes que puguen sorgir.
- Mantenir les polítiques d'empresa.

8.4.3. Control

Controlar l'execució d'un projecte de desenvolupament de programari suposa seguir, revisar i comparar els resultats davant les estimacions, els compromisos i els plans de projectes, 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.
- Prendre accions correctives sobre les possibles desviacions detectades.

8.5. Pla de projecte

El resultat final de l'etapa de planificació d'un projecte de desenvolupament de programari ha de ser el pla de projecte. Aquest és un document que descriu els treballs que s'ha previst realitzar al projecte i la forma en la qual el cap en dirigirà el desenvolupament, com també informació bàsica dels costos, recursos i planificació temporal. Entre altres aspectes ha de recollir:

- Què s'ha de fer, és a dir, quines són les activitats que s'han de realitzar per a aconseguir els objectius que es proposen al projecte.
- 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.
- Qui ho farà**,** quins són els recursos de personal dels quals es disposa per a poder dur a terme el treball.
- Quin serà el cost, quin cost suposarà la utilització tant del recursos materials com dels de personal.
- Quan fer-ho, quant de temps i quina serà la temporalització per tal de desenvolupar el projecte.
- Què s'obtindrà com a resultat, el que s'anomena *deliverable*, o entregable. Un entregable és un producte o servei final o intermedi que es produeix durant l'execució d'alguna de les tasques i activitats del projecte de construcció de programari. Durant l'execució del projecte es produeixen diferents *deliverables*.

El pla de projecte ha de servir sobretot al cap del projecte per a organitzar, dirigir i controlar el projecte, i els recursos i temps disponibles en la seua totalitat. Però també ha de ser útil per a l'usuari o client; per tant, el pla de projecte hauria de complir el següents requisits (Piattini, 2004):

- Proporcionar un resum del projecte als alts directius.
- Permetre supervisar el projecte al cap del projecte i als clients.
- Estar orientat al client.
- Ser un document base (*baseline*), és a dir, estar revisat i acceptat pel client i poder ser actualitzat a mesura que avança el projecte.

Alguns dels continguts que hauria d'incloure un bon pla de projectes són els següents:

- Resum del projecte i objectius.
- Productes «entregables» (*deliverables*).
- Procediments i estàndards que cal aplicar.
- Especificació del procés de revisió.
- Pla de comunicació entre l'equip de desenvolupament i el client.

- Diagrama de descomposició del treball (*Working Breakdown Structure*, WBS).
- Xarxa d'activitats que mostre la seqüència d'activitats en el temps i la relació entre si.
- Personal del projecte i assignació en relació amb el WBS.
- Llista de fites (*milestones*).
- Pressupostos i calendaris.

El resultat de la planificació es mostra en el pla de projecte. Però el principal objectiu de l'etapa de planificació d'un projecte de desenvolupament de programari és obtenir un calendari que permeta mesurar i controlar el progrés del projecte, com també assenyalar les activitats crítiques i que es puga modificar a mesura que avança el projecte.

Pàgina de títol	III. Procés de gestió
Full de revisió	A. Objectius i prioritats de gestió
Prefaci	B. Suposicions, dependències i restriccions
Taula de continguts	C. Gestió de riscs
Llista de figures	D. Mecanismes de supervisió i control
Llista de taules	E. Pla de personal
I. Introducció	IV. Procés tècnic
A. Visió general del projecte	A. Metodologia, tècniques i ferramentes
B. Productes finals	B. Documentació programari
C. Evolució del pla de projecte	C. Funcions de suport al projecte
D. Documents de referència	V. Pla de desenvolupament
E. Definicions i acrònims	A. Paquets de treball
II. Organització del projecte	B. Dependències
A. Model de processos	C. Recursos
B. Estructura organitzativa	D. Pressupost i distribució de recursos
C. Fronteres i interfícies organitzatives	E. Calendari
D. Responsabilitats	Components addicionals
	Índex
	Apèndix

Taula 8.2. Pla de projecte segons l'estàndard IEEE (IEEE, 2003)

El calendari ha de ser una representació gràfica de les activitats necessàries per a produir el resultat final del projecte, que permeta al cap del projecte dirigir i controlar de forma efectiva l'equip de desenvolupament durant tot el projecte. A més, ha de permetre al cap del projecte supervisar-lo de forma periòdica, coordinar-lo i saber el seu estat en tot moment.

A la taula 8.2 es presenta l'estructura d'un pla de projecte segons l'estàndard (IEEE, 2000).

8.6. Activitats per a la planificació d'un projecte de desenvolupament de programari

Un projecte de desenvolupament de programari sorgeix, com s'ha comentat abans, per una planificació feta en el seu moment dins de la planificació estratègica o davant d'un problema concret d'una empresa. En qualsevol cas, el projecte tracta de donar resposta a un problema de l'empresa que requereix informatització. Per tant, la planificació del projecte ha de partir de la definició d'aquesta situació problemàtica que es vol resoldre o millorar per mitjà de la informatització, amb la finalitat d'obtenir un pla de projecte. Aquest ha de recollir entre altres les activitats, recursos, temps i costos necessaris per tal d'aconseguir l'objectiu final del projecte. Per a realitzar la planificació és convenient seguir una sèrie de passos o activitats seqüencials, les quals es mostren a continuació i s'expliquen en més detall en els següents apartats:

- Definició dels objectius del projecte.
- Identificació i descomposició de les activitats.
- Estimació del temps i costos de les activitats.
- Relació entre les activitats.
- Identificació i assignació dels recursos.
- Planificació temporal.

8.6.1. Definició dels objectius del projecte

La primera activitat o pas en la planificació d'un projecte de desenvolupament de programari ha de ser la definició dels objectius del projecte. El cap del projecte és el responsable que el producte final acomplisca els requisits marcats pel client, per tant, és l'encarregat d'identificar i especificar els objectius del projecte en termes quantificables. Els objectius han de ser el fonament o punt de partida per a la planificació del projecte. En últim termini, l'objectiu de qualsevol projecte és obtenir un benefici, però s'han de concretar i detallar els objectius tangibles del projecte. Per tant, l'objectiu d'un projecte ha de ser un enunciat que especifique quins són els resultats que s'esperen aconseguir amb la consecució d'aquest.

Per tal que els objectius siguen realitzables, han de complir les característiques següents (Piattini, 2004):

- **Assequible:** l'objectiu no ha de ser massa ambiciós i suficientment realista perquè es puga dur a terme.
- **Definitiu:** l'objectiu ha de concretar què es vol aconseguir amb suficient grau de detall.
- **Quantificable:** l'objectiu ha d'indicar quins són els criteris per al seu fi.

- **Durada específica:** l'objectiu ha d'indicar quina serà la durada de les activitats.

8.6.2. Identificació i descomposició de les activitats

El segon pas és la generació del diagrama de descomposició del treball per tal d'identificar i dividir les activitats del projecte. El diagrama de descomposició del treball (*Working Breakdown Structure*, WBS) és una tècnica que permet representar les activitats que s'han de realitzar en el projecte amb diferent nivell de detall mitjançant un diagrama d'estructures.

El principi de la tècnica es basa a dividir l'activitat més general en activitats cada vegada més simples. Així successivament es reconeixen: els paquets de treball (*Work Packages*, WP), les activitats, les tasques i les subtasques. La figura 8.3 presenta el WBS per al cas del taller mecànic.

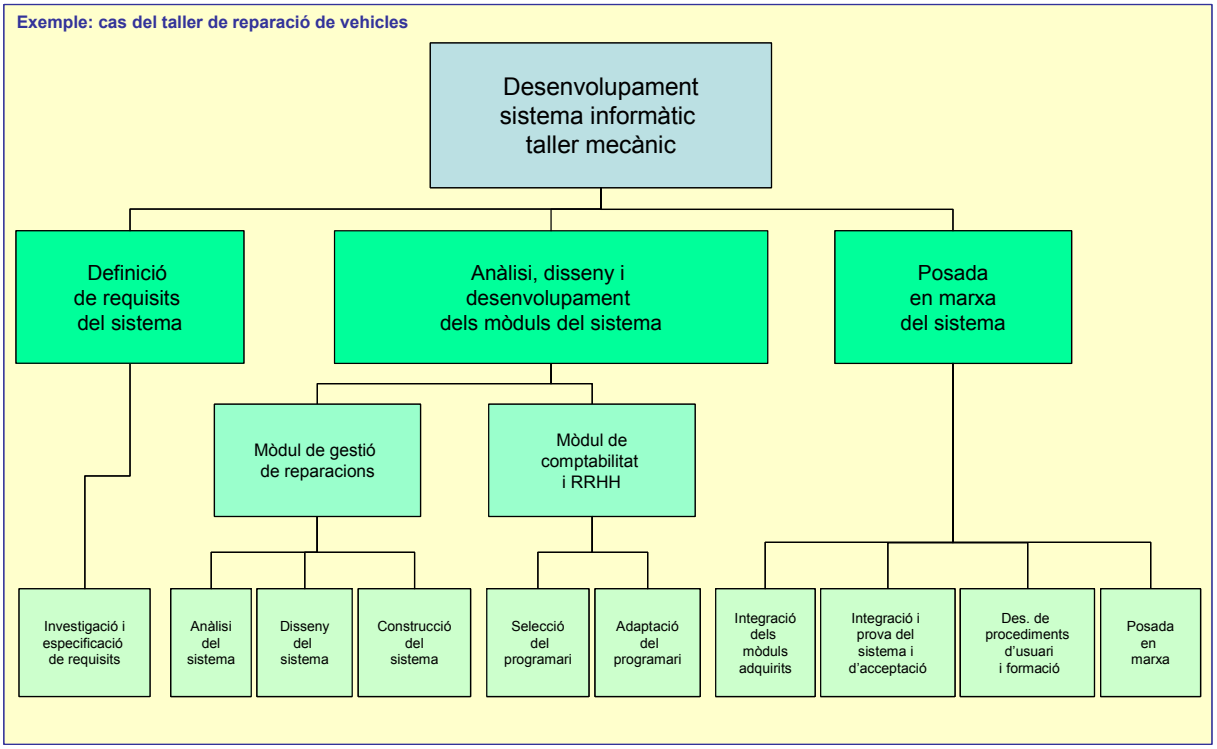


Figura 8.3. WBS del projecte per al cas del taller mecànic

Els passos que cal seguir per tal d'efectuar una descomposició d'activitats són (Piattini, 2004):

- **Identificar els paquets de treball:** aquests són especificacions del treball que s'ha de realitzar per a finalitzar una determinada funció. Poden incloure: personal, durada, recursos i responsable d'activitats.
- **Identificar les tasques i subtasques per a produir els paquets**.
- **Identificar els responsables de la fi:** cap del projecte, enginyer informàtic, programadors, etc. El responsable es pot indicar en la part inferior de cadascun dels quadres de les activitats.

8.6.3. Estimació dels temps i costos de les activitats

L'objectiu d'aquesta tercera activitat, en primer lloc, és estimar el temps que ha de transcórrer entre el començament i la fi d'una activitat i, en segon lloc, el cost que suposa la seua realització. Però abans d'estimar el temps i el cost cal conèixer com es pot mesurar el programari, ja que és necessari conèixer una determinada mesura del resultat de l'activitat que es vol estimar, abans de determinar el temps que s'ha d'invertir per a realitzar-la i el cost associat.

Per exemple, si s'analitza el cas d'un pintor al qual se li demana un pressupost per pintar una casa, aquest segueix els passos següents per tal d'elaborar el pressupost. En primer lloc, el pintor ha de tenir una idea de la feina que ha de fer i per fer-se aquesta idea, mesura la superfície de la casa, per tal d'obtenir la quantitat de metres que ha de pintar. En segon lloc, en funció de la seua experiència, pot realitzar un càlcul del temps que li suposa pintar la casa. I finalment, segons el temps i els recursos de personal i material que estime que ha d'utilitzar, pot establir el cost final que supose pintar la casa.

En el cas del desenvolupament de programari, per tal d'estimar el temps i cost d'un projecte de desenvolupament de programari es pot fer una analogia amb el cas del pintor. En primer lloc, cal realitzar una mesura d'allò que s'ha de produir, és a dir el programari. Després s'ha d'emprar qualsevol mètode d'estimació que permeta obtenir el temps de realització en funció de la mesura utilitzada i, finalment, el cost associat en funció dels recursos que cal destinar.

8.6.4. Establir relació entre les activitats

Una vegada estimats l'esforç, el temps i el cost del projecte, es pot realitzar el següent pas que és establir les relacions de precedència entre les activitats del projecte. És a dir, caldrà establir quines activitats necessiten de la finalització d'altres per a començar. Aquestes precedències marcaran la durada mínima del projecte i establiran restriccions temporals i d'assignacions de recursos.

Com a suport a aquesta activitat es poden utilitzar eines i mètodes com són les xarxes de precedència, grafs (com la tècnica PERT que es descriu més endavant en aquest tema), eines de planificació i assignació de recursos a projectes, etc.

8.6.5. Identificació i assignació dels recursos

El penúltim pas que cal seguir en la planificació d'un projecte de desenvolupament de programari és la identificació dels recursos disponibles i la seua assignació a les activitats del projecte. Els recursos que poden assignar-se a un projecte d'aquest tipus poden ser de tres classes:

- **Recursos de personal:** s'ha d'indicar per al personal quina és la posició que ocupa en l'organigrama de l'empresa, quina és la seua especialització, en quin equip de treball s'inclou, quin cost/hora té associat, quin cost/hora extra, etc.
- **Recursos de maquinari:** sistema de maquinari de desenvolupament i prova, i sistema de maquinari d'explotació.
- **Recursos de programari:** ferramentes de programació, d'anàlisi i de disseny, de depuració, etc.

8.6.6. Planificació temporal

La planificació temporal és l'última activitat de la planificació i té com a objectiu generar un calendari amb les activitats, precedències, temps i recursos estimats fins a aquest moment. Els passos que es poden seguir per a la realització del calendari són els següents:

- Indicar la data de començament del projecte i característiques generals d'aquest.
- Introduir l'esquema lògic (WBS) d'activitats, tasques i fites.
- Determinar la durada de cadascuna de les activitats.
- Identificar les relacions i dependències entre les activitats.
- Identificar i assignar els recursos.

La tècnica que es pot utilitzar per tal de confeccionar el calendari és la tècnica GANTT.

8.7. Mesures i mètriques del programari

Les mesures del programari fan referència a una indicació quantitativa d'un determinat atribut del programari, que pot ser obtingut mitjançant un procés de mesurament (Pressman, 2010). La definició matemàtica del terme (TermCat, 2011) defineix mesura com l'acció i efecte d'avaluar la quantitat d'una cosa per comparació amb una unitat.

Per a determinar les unitats de mesura es defineixen mètriques. Així, quan es parla de mètrica del programari, es fa referència a mesures normalitzades de manera que siguen comparables, és a dir, a mesures en funció o relatives a unitats establertes de mesurament. L'IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) defineix mètrica com "*una mesura quantitativa del grau en el qual un sistema, component o procés pot tenir un determinat atribut*".

Per exemple, una mesura del programari és el nombre total de línies de codi produïdes (LDC), el nombre d'errors detectats, la funcionalitat, la qualitat, la fiabilitat del programari, etc. Una mètrica de la qualitat del programari pot ser el nombre d'errors per línies de codi.

Les mesures que es poden establir sobre el programari poden ser:

- **Mesures directes,** que són fàcils de recollir. Per exemple, en el cas del pintor la quantitat de metres que ha de pintar és una mesura directa. En el cas del programari són mesures directes les línies de codi produïdes (LDC), el nombre d'errors detectats, el nombre de pàgines de documentació produïdes, etc.
- **Mesures indirectes,** que són difícils d'avaluar i només es poden obtenir a partir d'altres mesures directes. Per exemple, en el cas del pintor una mesura indirecta és la qualitat de la pintada. En el cas del programari són mesures indirectes la funcionalitat, la qualitat, la fiabilitat, la facilitat de manteniment del programari, etc.

Pel que fa a les mètriques del programari es pot diferenciar entre (Pressman, 2010):

- **Mètriques orientades a la grandària:** tenen en compte mesures normalitzades en funció de la grandària del programari produït, la qual es mesura normalment en nombre de línies de codi produïdes (LDC). Per tal d'obtenir aquest tipus de mètriques, en primer lloc, es poden mesurar de forma directa paràmetres relacionats amb el programari com poden ser les LDC, el nombre d'errors detectats, les pàgines de documentació produïda, etc. Però aquestes mesures per si mateixes no poden donar una idea de la grandària del projecte, llevat que es normalitzen amb la finalitat d'establir comparacions. Una vegada normalitzades les mesures, és possible crear una mètrica que permeta comparar diferents projectes. D'aquesta manera es poden crear taules amb mètriques creades per a diferents projectes tipus, que permetran realitzar una estimació aproximada d'aquestes per a un nou projecte.
- **Mètriques orientades a la funció:** utilitzen com a valor de normalització una mesura de la funcionalitat que proporcione l'aplicació (IFPUG, 2009). La funcionalitat és una mesura indirecta i per tant és difícil de mesurar. Albretch (Pressman, 2010) va suggerir el 1979 una mètrica per a la funcionalitat anomenada punts de funció. Per a obtenir els punts de funció cal mesurar les cinc característiques següents del programari:
    - Nombre d'entrades d'usuari.
    - Nombre d'eixides d'usuari.
    - Nombre d'arxius.
    - Nombre d'interfícies externes.
    - Nombre de peticions d'usuari.

A partir d'aquestes mesures s'aplica la fórmula següent:

$$PF = \text{suma-total} * [0{,}65 + 0{,}01 * \Sigma F_i]$$

On suma-total és la suma ponderada per als cinc paràmetres mencionats, i F_i és la suma que s'obté després de contestar 14 preguntes sobre la complexitat del projecte, segons una escala que va del 0 al 5. Per a obtenir informació completa de com s'han de calcular els punts de funció podeu consultar el web de la bibliografia. Actualment, s'han definit els punts d'objecte com una alternativa als punts de funció quan s'utilitzen llenguatges de quarta generació (Sommerville, 2005).

Per tant l'objectiu final és obtenir mètriques que permeten comparar alguna mesura del programari que es vol desenvolupar. Aquestes mètriques han de permetre comparar diferents projectes de diferent grandària i característiques, per a poder determinar finalment el temps i cost d'un projecte nou. A l'hora de construir les mètriques es poden seleccionar les LDC o els punts de funció com a valor de normalització. Així per exemple, en funció de les LDC es poden obtenir les següents mètriques:

- Productivitat = LDC /homes-mes.
- Qualitat = errors /LDC.
- Cost = unitats monetàries / LDC.
- Documentació = nombre de pàgines de documentació /LDC.

8.8. Mètodes d'estimació

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

- No ser una ciència exacta, l'estimació és sinònim de predicció, 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 homes-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. En última instància, sabent l'esforç que suposa un projecte, sempre es pot obtenir l'estimació en forma de cost monetari.
- Un dels principals problemes que presenta l'estimació dels projectes de desenvolupament de programari, és que el producte que s'ha de realitzar és difícil de mesurar, no es tracta d'un procés repetitiu i el producte que s'obté no és quasi mai el mateix.

- Es pot distribuir l'esforç entre les diferents fases del desenvolupament d'un sistema informàtic de la següent forma: fase de definició de requisits i d'anàlisi (10-25%), fase de disseny (20-25%), fase de construcció (15-20%), i proves (30-40%). Encara que les característiques del projecte poden fer variar aquesta distribució.

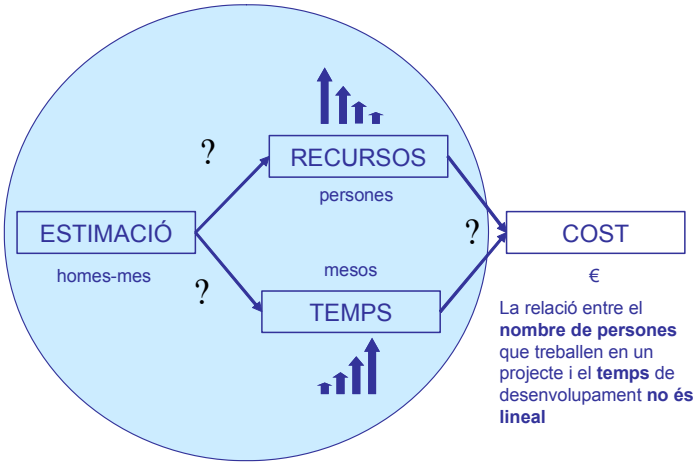


Figura 8.4 Diagrama estimació-cost en projectes de desenvolupament de programari

Per tant l'estimació d'un projecte de desenvolupament de programari es realitza en termes d'esforç, és a dir, homes-mes[5]. Açò vol dir que una estimació de 50 homes-mes, per a un determinat projecte, suposaria el treball de 50 persones durant un mes per tal de desenvolupar-lo. A partir d'aquesta estimació 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 (veure la figura 8.4), 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.

Existeixen diferents mètodes d'estimació de costos en projectes de desenvolupament de programari (Piattini, 2004):

- **Opinió d'experts:** eufemisme de l'endevinació basada en l'experiència personal que s'utilitza per tal de realitzar l'estimació.
- **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 el cost del nou projecte que es vol estimar.

[5] Seria més apropiat utilitzar el terme persona-mes.

- **Estimació per descomposició:** descompondre el projecte fins a un nivell de detall suficient que permeta estimar el cost de les unitats elementals en les quals s'ha descompost el projecte. El cost total del projecte és igual a la suma del cost 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. Es diferencien:
    - Models de costos: proporcionen estimacions directes de l'esforç o durada d'un projecte. Són models que tenen en compte factors empírics, més factors d'ajust obtinguts també a partir d'altres projectes, per exemple el model COCOMO.
    - Models de restriccions**:** relacionen en el temps dos o més paràmetres de cost.

  Dins dels models d'estimació si es té en compte la base utilitzada per a la seua construcció, es pot distingir entre:

    - Models empírics: basats en l'opinió d'experts i en l'estimació per descomposició.
    - Models estadístics**:** són models obtinguts mitjançant l'anàlisi de regressió estadística sobre les dades recollides d'una gran quantitat de projectes.
    - Models lineals**:** obtenen l'esforç en funció d'una equació lineal.
- **Models no lineals:** obtenen l'esforç en funció d'una equació no lineal.
    - Models basats en una determinada teoria**.**
    - Models composts**:** models basats en la combinació de diferents tècniques, com per exemple el model COCOMO que es descriu en detall en (Pressman, 2010) i (Piattini, 2004).

8.9. Xarxes de precedència: PERT

Una xarxa de precedència és una representació gràfica del projecte que relaciona les activitats, de forma que permet visualitzar les que són crítiques i la relació seqüencial que hi ha entre si.

Abans de començar la representació gràfica cal confeccionar el quadre de relacions de precedència. Aquest quadre és una taula de tres columnes: en la primera s'identifiquen les activitats en les quals es descompon el projecte, en la segona s'assigna una etiqueta a cadascuna de les activitats mitjançant una lletra i en la tercera s'indica, mitjançant la lletra corresponent, l'activitat o les activitats precedents per a cadascuna de les activitats del projecte, és a dir, les activitats que cal que finalitzen per a poder començar una determinada activitat. En la taula 8.3. es mostren les relacions de precedència per al cas del taller de reparació de vehicles.

Activitat	Identificador d'activitat	Activitats precedents
Desenvolupament sistema informàtic taller mecànic		
Definició de requisits del sistema		
Investigació i especificació de requisits	A	---
Anàlisi, disseny i desenvolupament dels mòduls del sistema		
Mòdul de gestió de reparacions		
Anàlisi del mòdul	B	A
Disseny del mòdul	C	B
Construcció del mòdul	D	C
Mòdul de comptabilitat		
Selecció del programari	E	A
Adaptació del programari	F	E
Mòdul de RRHH		
Selecció del programari	G	A
Adaptació del programari	H	G
Posada en marxa del sistema		
Integració dels mòduls adquirits	I	F, H
Integració i prova del sistema i d'acceptació	J	D, I
Desenvolupament de procediments d'usuari i formació	K	D, I
Posada en marxa	L	J, K

Taula 8.3. Relacions de precedència per al cas del taller de reparació de vehicles

Dues de les tècniques basades en grafs que tenen diferent utilitat per a la planificació de projectes són:

- **PERT** (*Program Evaluation and Review Technique*), enfocada als successos. Un avantatge d'aquesta tècnica és que els successos es poden considerar fites del projecte i açò pot facilitar el control del projecte. Permet fer una estimació dels temps basada en la probabilitat.
- **CPM** (*Critical Path Method*), enfocada a les activitats. No permet fer una estimació dels temps basada en la probabilitat.

La tècnica PERT es pot aplicar a projectes que compleixen les característiques següents:

- La xarxa ha de tenir un mínim de 20 successos; altrament, una alternativa podria ser el diagrama GANTT.
- La xarxa ha de tenir un màxim de 300 successos; altrament, l'alternativa seria utilitzar una ferramenta automatitzada.
- Projectes molt crítics, de gran incertesa, d'alt risc, que involucren moltes persones, etc.

La tècnica parteix de la descomposició del projecte en activitats i la representació mitjançant un graf on es poden identificar els elements de la figura 8.5.

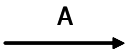
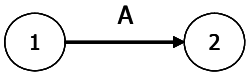
Concepte	Representació gràfica
Vèrtex = Succés	
Arc = Activitat	
Representació mitjançant un graf	

Figura 8.5 Elements de la representació gràfica en la tècnica PERT

Les activitats ocorren entre dos successos, un succés inicial i un final, i existeixen tres tipus de relacions com es pot observar en la figura 8.6, entre les quals es pot fer qualsevol tipus de combinació:

- **Lineal:** indica que existeixen dues activitats, l'A que s'inicia en el succés 1 i té el final en el succés 2; i la B que comença en el 2 i acaba en el 3. A més a més, per iniciar l'activitat B és necessari haver finalitzat l'activitat A. El succés 2 és el succés final d'A i l'inicial de B.
- **Convergent:** indica que per iniciar l'activitat D és necessari haver finalitzat les activitats A, B i C.
- **Divergent:** indica que per a començar qualsevol de les activitats B, C o D, cal haver finalitzat l'activitat A.

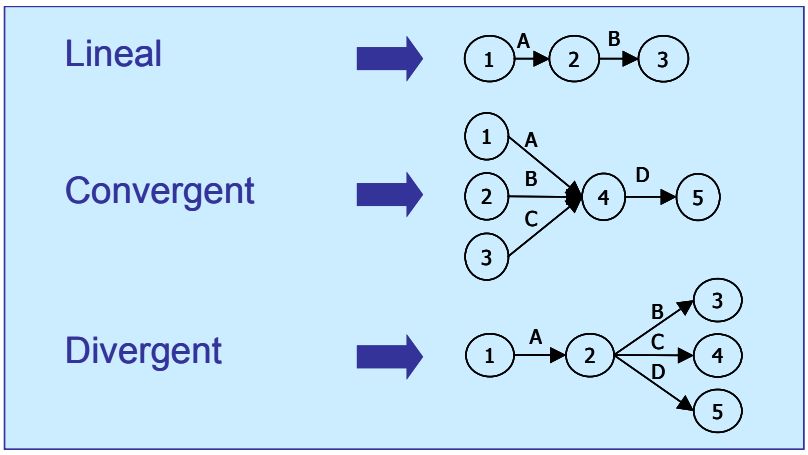


Figura 8.6 Tipus de relacions en la tècnica PERT

A l'hora d'establir precedències poden sorgir alguns problemes. Si s'observa la figura 8.7, s'entén que per a poder començar l'activitat D és necessari finalitzar les activitats A, B i C. De la mateixa manera és necessari acabar aquestes tres activitats per tal de començar l'activitat E. Ara bé, com es representaria el següent enunciat? L'activitat D només ha d'estar precedida per les activitats A i B; i

l'activitat E per les activitats A, B i C. Amb el que s'ha explicat fins al moment s'obtindria el mateix diagrama que es presenta a la figura 8.7, i per tant s'estaria davant d'una ambigüitat.

Aquesta situació ambigua es pot solucionar mitjançant la introducció de les anomenades **activitats fictícies**, com s'indica en la figura 8.8. Una activitat fictícia és una activitat que en realitat no existeix i que només s'afegeix en cas de situacions conflictives de precedència, per tal d'aclarir les relacions de precedència a nivell de la representació gràfica. Aquest tipus d'activitats s'etiqueten amb una lletra prima i tenen una durada 0, per tal de no afectar la durada total del projecte, perquè en realitat no existeixen.

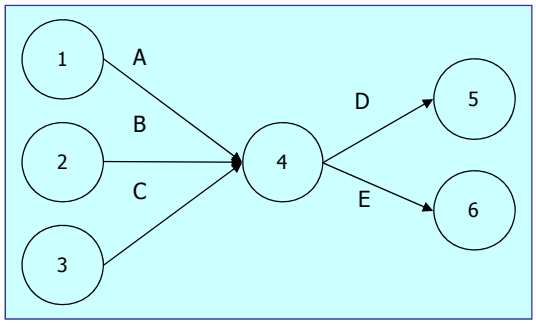


Figura 8.7 Primera situació conflictiva

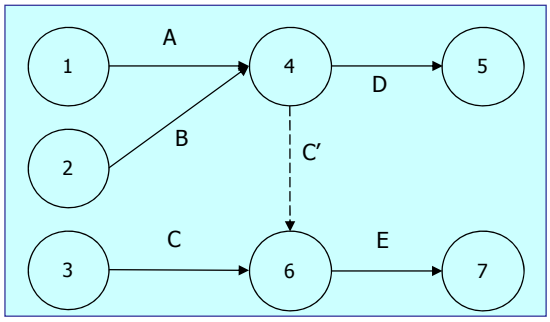


Figura 8.8 Solució a la primera situació conflictiva

Una segona situació conflictiva pot sorgir quan es tenen dues activitats que comencen i acaben en el mateix succés. Açò provoca una indeterminació, ja que cadascuna de les activitats ha de tenir un inici i un final que la identifique de forma única. Aquesta situació és la que mostra la figura 8.9. Per a solucionar aquesta situació es pot utilitzar una activitat fictícia de la forma en què s'indica en la figura 8.10.

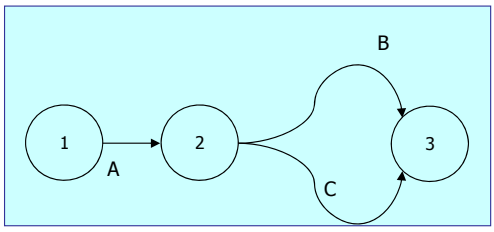


Figura 8.9 Segona situació conflictiva

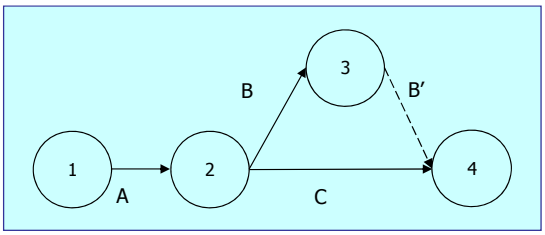


Figura 8.10 Solució a la segona situació conflictiva

A partir de la taula de precedències, i amb la utilització dels diferents tipus de relacions que poden existir, es pot confeccionar el diagrama PERT. El corresponent al cas del taller de reparació de vehicles es mostra a la figura 8.11.

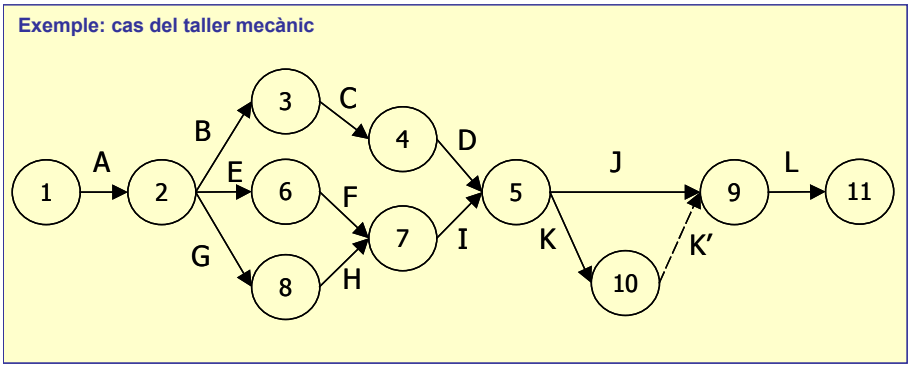


Figura 8.11. Diagrama PERT per al cas del taller de reparació de vehicles

Per poder establir la durada del projecte i les activitats que són crítiques per al seu desenvolupament s'han de calcular els temps més promptes i més tardans, en els quals poden començar i finalitzar les activitats. A efectes de càlcul les activitats fictícies tenen una durada zero. Donats dos successos i, j i una activitat A que transcorre entre aquests dos successos, caldrà calcular cadascun dels valors que es mostren en la figura 8.12.

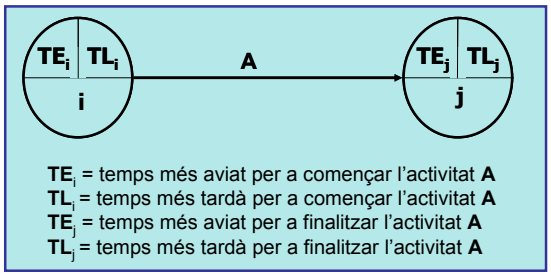


Figura 8.12. Càlcul de temps en un diagrama PERT

En aquest diagrama els subíndexs tenen el següent significat:

i = succés inicial, subíndex relacionat amb el començament o inici de les activitats.

j = succés final, subíndex relacionat amb el final o fi de les activitats.

E = referent al més prompte possible, de l'anglès *early*.

L = referent al més tardà, de l'anglès *late*.

8.9.1. Càlcul del temps *early* o més prompte

En primer lloc cal fer el càlcul dels temps més prompte possible quan pot començar cadascuna de les activitats. Per obtenir aquests temps s'ha de recórrer el diagrama d'esquerra a dreta, és a dir, des del succés d'inici 1 fins al succés final del diagrama. Per a calcular el temps més prompte d'un succés j, i que per tant serà el temps més prompte que es pot iniciar una activitat que comence en aquest vèrtex, s'haurà de calcular el temps més prompte que poden acabar totes les activitats que arriben a aquest succés i elegir el màxim, (fins que no acaben totes les activitats que han de precedir una determinada,

aquesta no pot començar). La fórmula que serveix per a realitzar aquest càlcul es mostra a la figura 8.13 tenint en compte que $TE_1 = 0$.

$TE_j = \max [TE_i + T_{ij}]$, $\forall$ i succés inici d'activitats que acaben al succés j

TE_j = temps més aviat del succés **j**
TE_i = temps més aviat d'un succés **i**
T_{ij} = durada d'una activitat que s'inicia en un succés **i,** i acaba en el **j**

Figura 8.13. Càlcul del temps més prompte

Per exemple, en el cas del taller de reparació de vehicles presentat a la figura 8.14, s'observa que per més prompte que vulga començar l'activitat I no es pot fer fins que no acaben la F i la H, que en el millor dels casos ve determinat pel màxim de sumar el més prompte que poden començar cadascuna de les dues activitats i el seu temps de durada estimada.

Exemple: cas del taller mecànic

Activitat	A	B	C	D	E	F	G	H	I	J	K	L
Durada	10	5	8	15	3	5	2	3	6	8	9	3

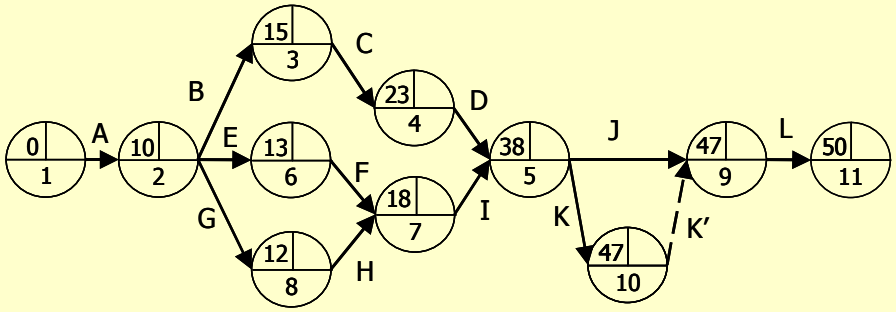


Figura 8.14. Càlcul del temps més prompte per al cas del taller de reparació de vehicles

8.9.2. Càlcul del temps *late* o més tardà.

Una vegada calculats els temps més promptes es pot passar a calcular els més tardans, per a la qual cosa s'ha de recórrer el diagrama de dreta a esquerra, és a dir, des del succés final fins a l'inicial

$TL_i = \min [TL_j - T_{ij}]$, $\forall$ j succés acabament d'activitats que comencen al succés i

TL_i = temps més tardà del succés **i**
TL_j = temps més tardà del succés **j**
T_{ij} = durada de l'activitat que s'inicia en el succés **i**, i acaba en el **j**

Figura 8.15. Càlcul del temps més tardà

A la figura 8.15 es pot observar la fórmula que cal aplicar per tal de realitzar aquest càlcul. El càlcul d'aquest temps per al cas del taller mecànic es pot observar a la figura 8.16.

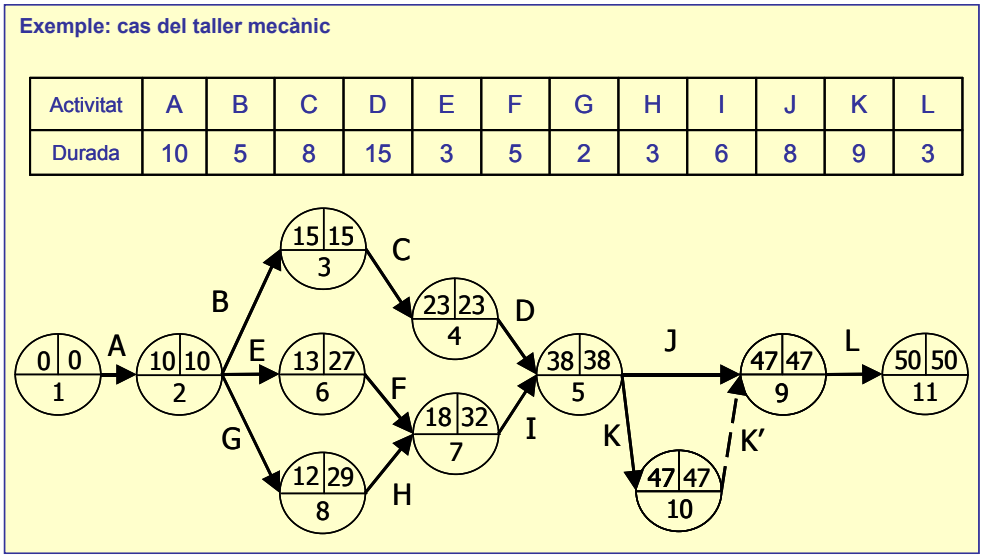


Figura 8.16. Càlcul del temps més tardà per al cas del taller de reparació de vehicles

8.9.3. Folgances

Les folgances donen una idea del marge de temps que pot endarrerir-se un activitat sense que afecte a la resta o a l'acabament del projecte.

La folgança: fa referència a un succés o vèrtex, és el marge de temps entre el temps més tardà i el més prompte que pot començar una activitat que s'inicia en el succés. Aquells successos la folgança dels quals siga zero seran crítics per aconseguir acabar el projecte.

$$F_i = TL_i - TE_i.$$

La folgança total: és el marge de temps que es pot endarrerir una activitat respecte al temps previst, sense que augmente la durada total del projecte. Aquelles activitats la folgança total de les quals siga zero seran crítiques per aconseguir acabar el projecte en el temps estimat inicialment

$$\mathbf{F^T_{ij}} = TL_j - TE_i - T_{ij}$$

La folgança lliure d'una activitat representa la part de la folgança total que pot consumir-se sense que s'afecten les següents activitats**:**

$$F^{LL}_{ij} = TE_j - TE_i - T_{ij}$$

Les folgances ens permeten determinar quines activitats són crítiques, pel que fa a la seua durada i el seu moment d'inici. Poden haver-hi activitats entre dos successos amb folgança zero que no siguen crítiques. A la figura 8.17 es presenten alguns exemples del càlcul de folgances **(F)** per al cas del taller mecànic.

- **Exemple:** cas pràctic del "Taller de reparació de vehicles"

$F_1 = TL_1 - TE_1 = 0 - 0 = 0$

$F_7 = TL_7 - TE_7 = 32 - 18 = 14$

$F^T_{12} = TL_2 - TE_1 - T_{12} = 10 - 0 - 10 = 0$ (Activitat **A**)

$F^{LL}_{12} = TE_2 - TE_1 - T_{12} = 10 - 0 - 10 = 0$ (Activitat **A**)

$F^T_{26} = TL_6 - TE_2 - T_{26} = 27 - 10 - 3 = 14$ (Activitat **E**)

$F^{LL}_{26} = TE_6 - TE_2 - T_{26} = 13 - 10 - 3 = 0$ (Activitat **E**)

$F^T_{87} = TL_7 - TE_8 - T_{87} = 32 - 12 - 3 = 17$ (Activitat **H**)

$F^{LL}_{87} = TE_7 - TE_8 - T_{87} = 18 - 12 - 3 = 3$ (Activitat **H**)

Figura 8.17. Càlcul de folgances per al cas del taller

8.9.4. El camí crític

Activitat crítica és aquella que té folgança total igual a zero. De manera que si es retarda la data de començament o de fi prevista d'aquesta activitat, s'ocasionarà un retard en la durada total del projecte.

El camí o camins crítics estan formats pel conjunt d'activitats crítiques i pels nodes o successos amb folgança zero.

El camí crític determina la durada total del projecte. Per a disminuir el temps total del projecte cal disminuir els temps estimat per a la realització de les activitats que formen part del camí crític. Cal tenir en compte quina variació suposarà aquesta disminució en el cost total del projecte. De vegades aquest camí crític pot no ser el determinant de la durada final del projecte, ja que durant la realització d'aquest poden aparèixer nous camins crítics que en un primer moment no s'havien previst. El camí crític ha de convertir-se en la base per a la planificació, direcció i control del projecte. El cap del projecte ha de ser l'encarregat de fer que no apareguen nous camins crítics que puguen endarrerir l'execució del projecte.

A la figura 8.18 es presenta el camí crític, en color roig, per al cas del taller de reparació vehicles.

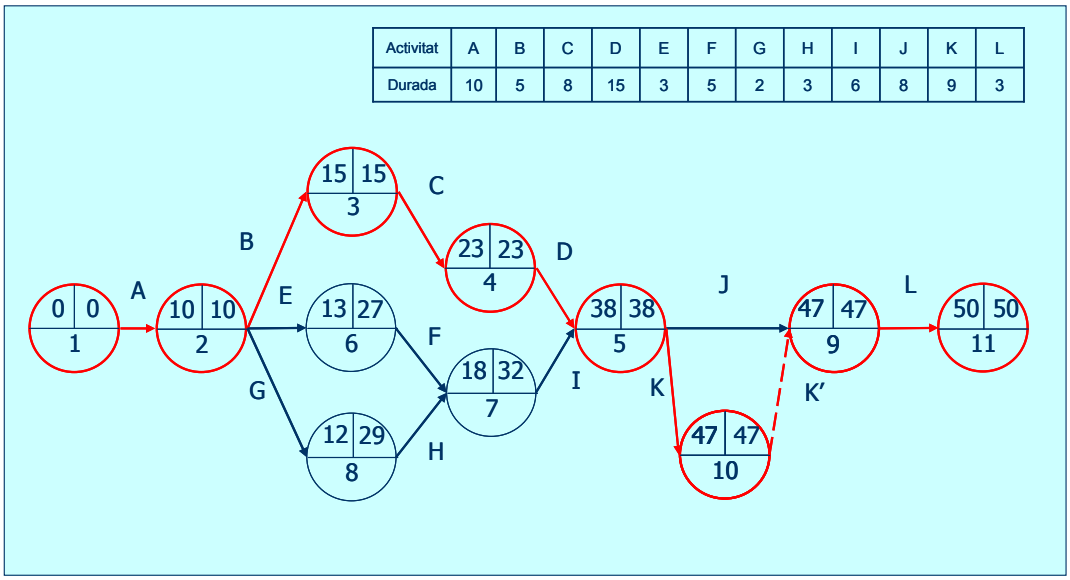


Figura 8.18. Camí crític per al cas del taller de reparació de vehicles

8.10. Diagrama de Gantt

El diagrama de Gantt és una tècnica que permet fer una planificació temporal i permet generar un calendari amb les activitats, precedències, temps i recursos estimats. Els passos a seguir són:

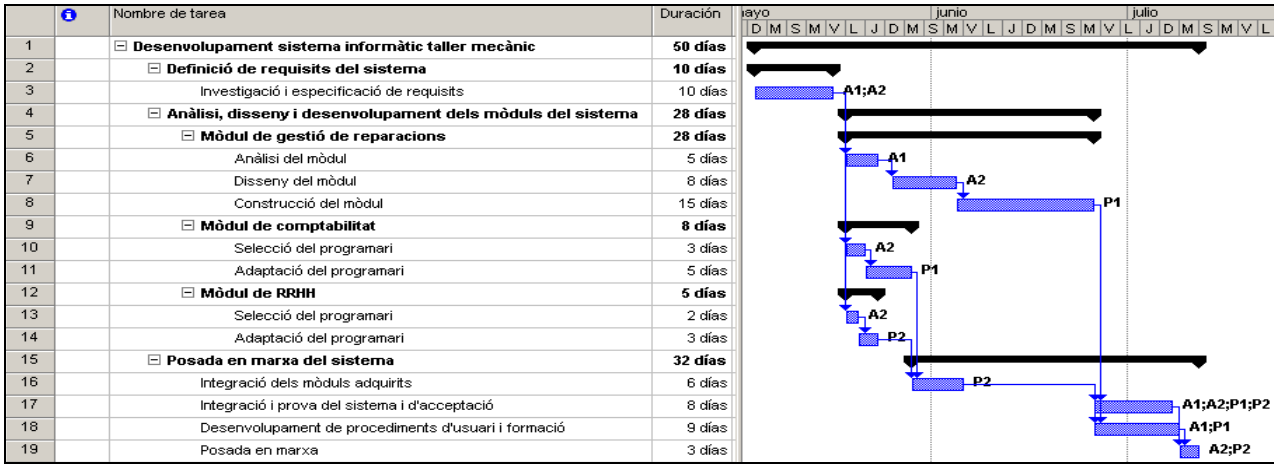


Figura 8.19. Diagrama de Gantt per al cas del taller de reparació de vehicles

- Indicar la data de començament del projecte i característiques generals d'aquest.
- Introduir l'esquema lògic (WBS) d'activitats, tasques i fites.
- Determinar la durada de cadascuna de les activitats.
- Identificar les relacions i dependències entre les activitats.
- Identificar i assignar els recursos.

En la figura 8.19 és mostra l'exemple de diagrama de Gantt fet amb l'eina Microsoft Project i que s'estudiarà en més detall a les sessions de pràctiques.

Resum

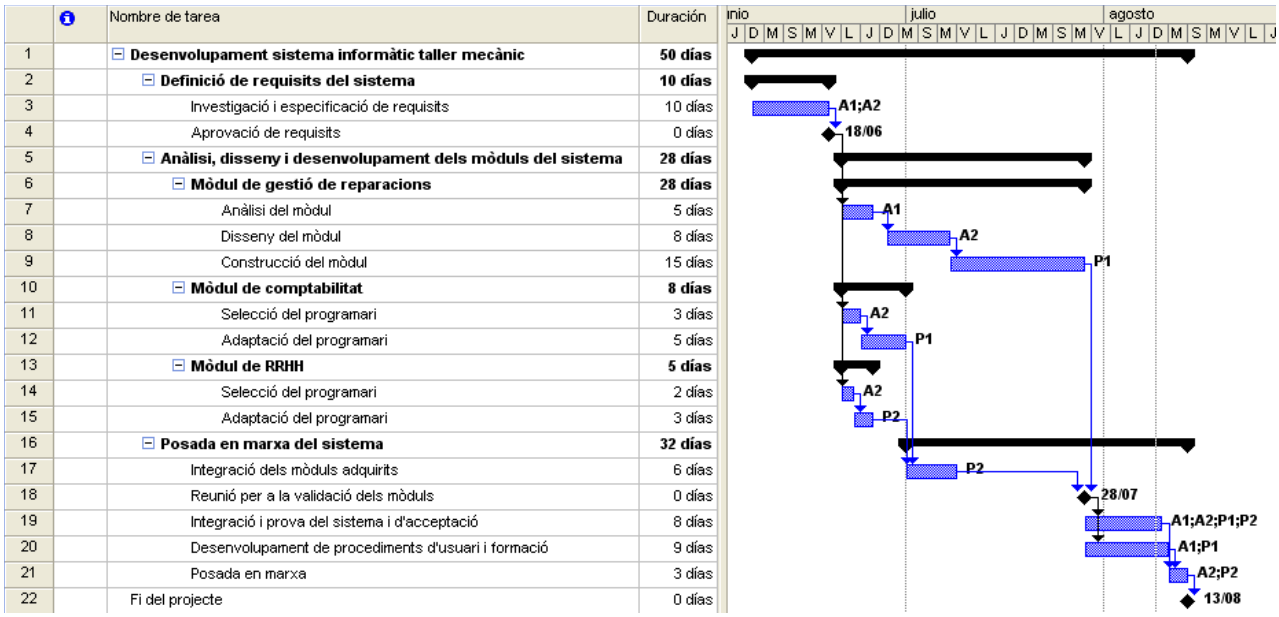
En aquest tema es tracta el desenvolupament de programari des d'una visió de gestió per a projectes. En el tema es presenten les tres etapes que cal seguir per a una bona gestió de qualsevol tipus de projecte, planificació, direcció i control; després s'ha de centrar en la planificació de projectes de desenvolupament de programari. La planificació d'aquest tipus de projecte pot estar guiada per una sèrie de passos que es descriuen al tema. A més, es mostren diverses tècniques d'ajuda en el cas de l'estimació i l'establiment de les relacions de precedència entre activitats.

Activitats complementàries

1. Busqueu en el diccionari els següents conceptes: fita, estimació, mesura, esforç, productivitat, folgança i recurs.
2. Confeccioneu deu preguntes d'examen per a aquest tema.
3. Llegiu el capítol 2 del llibre de la bibliografia *The Mythical Man-Month*.
4. Enumereu almenys sis entregables i sis fites per al projecte que es podria realitzar per al cas del taller de reparacions de vehicles.
5. Transformeu el següent esquema lògic d'un projecte d'exemple realitzat amb la ferramenta Microsoft Project, en el corresponent diagrama de descomposició del treball (WBS).

	Nombre de tarea
1	**Desenvolupament del sistema**
2	Revisió de l'anàlisi
3	**Disseny i desenvolupament dels mòduls**
4	**Mòdul 1**
5	Disseny BD i fitxers
6	Disseny de processos i interfícies
7	Construcció
8	Integració i prova
9	**Mòdul 2**
10	Disseny BD i fitxers
11	Disseny de processos i interfícies
12	Construcció
13	Integració i prova
14	Desenvolupament Mòdul informes
15	Desenvolupament Mòdul comptabilitat
16	Integració mòduls i prova
17	Conversió
18	Formació
19	Entrega del producte a l'usuari
20	Fi del projecte

6. Identifiqueu al següent diagrama del projecte proposat per al cas del taller de reparació de vehicles les fites que n'hi ha assenyalades i les precedències de cadascuna de les activitats. Feu la taula de precedències.



	Nombre de tarea	Duración
1	**Desenvolupament sistema informàtic taller mecànic**	**50 días**
2	**Definició de requisits del sistema**	**10 días**
3	Investigació i especificació de requisits	10 días
4	Aprovació de requisits	0 días
5	**Anàlisi, disseny i desenvolupament dels mòduls del sistema**	**28 días**
6	**Mòdul de gestió de reparacions**	**28 días**
7	Anàlisi del mòdul	5 días
8	Disseny del mòdul	8 días
9	Construcció del mòdul	15 días
10	**Mòdul de comptabilitat**	**8 días**
11	Selecció del programari	3 días
12	Adaptació del programari	5 días
13	**Mòdul de RRHH**	**5 días**
14	Selecció del programari	2 días
15	Adaptació del programari	3 días
16	**Posada en marxa del sistema**	**32 días**
17	Integració dels mòduls adquirits	6 días
18	Reunió per a la validació dels mòduls	0 días
19	Integració i prova del sistema i d'acceptació	8 días
20	Desenvolupament de procediments d'usuari i formació	9 días
21	Posada en marxa	3 días
22	Fi del projecte	0 días

7. Realitzeu el diagrama de descomposició del treball (WBS) i el diagrama PERT, assenyalant el camí o camins crítics en el diagrama i calculant les folgances totals i lliures de les activitats que NO formen part del camí o camins crítics per als projectes d'exemple mostrats en les taules 1, 2, 3, 4 i 5.

Taula 1:

Activitat	Activitats precedents	Durada estimada
A	--	4d
B	A	5d
C	A	3d
D	B	2d
E	C, D	4d
F	C, D	7d
G	D	6d
H	E	5d
I	G, H	2d
J	F, H	3d
K	I	4d
L	J	5d
M	K, L	5d

Taula 2:

Activitat	Activitats precedents	Durada estimada
A	-	6d
B	A	10d
C	A	9d
D	A	8d
E	B	5d
F	B, C, D	8d
G	D	6d
H	D	7d
I	E	10d
J	E	9d
K	E, F, G, H	10d
L	H	7d
M	I, J, K, L	6d

Taula 3:

Activitat	Activitat	Act. precedents	Durada
Estudi inicial	A	-	5d
Presentació del pressupost	B	A	3
Aprovació del pressupost i inici del projecte	C	B	4d
Reunions de control i seguiment del projecte	D	C	24d
Reunions de treball inicials	E	B	5d
Avaluació i Selecció del programari	F	E	7d
Preparar plataforma de maquinari	G	E	5d
Parametrització del producte	H	F;G	10d
Desenvolupament de mòduls a mida	I	F;G	11d
Formació en el producte	J	G	12d
Migració de dades	K	H;I	2d
Posada en marxa	L	J;K	3d
Acceptació dels usuaris i pagament final	M	D;L	1d

Taula 4:

Activitat	Identificador d'activitat	Activitats precedents	Durada estimada	Recursos assignats
Desenvolupament del sistema				
Reunions de definició d'objectius	A	--	2 dies	G
Definició i anàlisi de requisits				
Entrevistes als responsables d'àrees	B	A	3 dies	A (50%)
Revisió del sistema actual				
Anàlisi documental	C	B	4 dies	A (50%)
Anàlisi de processos	D	B	3 dies	A (50%)
Especificació de requisits	E	C, D	5 dies	A (50%), G (50%)
Anàlisi del sistema				
Desenvolupament del model funcional	F	E	5 dies	A (50%), P1 (50%)
Desenvolupament del model conceptual	G	E	3 dies	A (50%), P1 (50%), P2 (50%)
Verificació consistència de models	H	F, G	1 dia	A (50%), P1 (50%), P2 (50%)
Aprovació de l'anàlisi	**I**	**H**	**0 dies**	**FITA**
Disseny del sistema				
Revisió de l'anàlisi	J	I	1 dia	A, P1, P2
Disseny BD i fitxers	K	J	6 dies	A (50%), P1 (50%), P2 (50%)
Disseny de mòduls	L	K	3 dies	A (50%), P1, P2
Disseny d'interfícies d'usuari	M	J	5 dies	P1 (50%), P2 (50%)
Especificació del disseny	N	L, M	2 dies	A, P1 (25%), P2 (25%)
Construcció i posada en marxa del sistema				
Desenvolupament del pla d'instal·lació	Ñ	N	1 dia	A
Preparació de l'entorn de desenvolupament i proves	O	L	4 dies	P2 (75%), T
Desenvolupament dels components de programari				
Disseny detallat	P	Ñ, O	3 dies	A, P1, P2
Programació	Q	P	8 dies	P1, P2, T (50%)
Disseny i realització de proves	R	Q	2 dies	P1, P2, T (50%)
Desenvolupament de procediments i formació				
Desenvolupament de procediments d'usuari	S	R	5 dies	A, P1 (50%), P2 (50%)
Formació	T	S	3 dies	A (50%), P1 (50%), P2 (50%)
Preparació de l'entorn d'explotació	U	R	4 dies	P2 (50%), T
Posada en marxa				
Conversió	V	U	2 dies	P1 (50%), P2 (50%), T
Entrega del producte a l'usuari	W	T,V	1 dia	A (50%), P1, P2, T, G (50%)
Fi del projecte	**X**	**W**	**0 dies**	**FITA**

Taula 5:

Activitat	Identificador d'activitat	Activitats precedents	Durada estimada	Recursos assignats
Revisió de l'anàlisi	A	--	4d	Enginyer en informàtica; Programador1; Programador2; ABC-Soft
Disseny BD i fitxers (Mod. màquines)	B	A	2d	Enginyer en informàtica; Programador1 [50%]; Programador2 [50%]
Disseny de processos i interfícies (Mod. màquines)	C	B	3d	Enginyer en informàtica [50%]; Programador1
Construcció (Mod. màquines)	D	C	4d	Programador1
Integració i prova (Mod. màquines)	E	D	2d	Enginyer en informàtica [50%]; Programador1
Desenvolupament Mòdul informes	F	B, G	12d	ABC-Soft
Disseny BD i fitxers (Mod. reposicions)	G	A	2d	Enginyer en informàtica; Programador1 [50%]; Programador2 [50%]
Disseny de processos i interfícies (Mod. reposicions)	H	B, G	3d	Enginyer en informàtica [50%];Programador2
Construcció (Mod. reposicions)	I	C, F, H	5d	Programador2
Integració i prova (Mod. reposicions)	J	I	2d	Enginyer en informàtica [50%]; Programador2
Desenvolupament Mòdul comptabilitat	K	A	24d	ABC-Soft
Integració mòduls i prova	L	E, J, K	1d	Enginyer en informàtica; Programador1 [50%]; Programador2 [50%]; ABC-Soft [50%]
Conversió	M	L	1d	Enginyer en informàtica; Programador1 [50%]; Programador2 [50%]; ABC-Soft [50%]
Formació	N	L	2d	Enginyer en informàtica; Programador1 [50%]; Programador2 [50%]; ABC-Soft [50%]
Entrega del producte a l'usuari	Ñ	M, N	1d	Enginyer en informàtica; Programador1; Programador2; ABC-Soft
Fi del projecte	O	Ñ	0d	FITA

Capítol 9

Paradigmes i metodologies

Objectius

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

- Estudiar alguns dels paradigmes de l'Enginyeria del programari i saber en quins casos pot ser més apropiat la utilització de cadascun.
- Estudiar el concepte de cicle de vida
- Conèixer el concepte de metodologia i algunes de les metodologies més difoses i utilitzades, tant en l'àmbit nacional com en l'internacional.

9.1. Models del cicle de vida en el paradigma estructurat

La importància del programari s'ha vist incrementada per la baixada dels costos del maquinari. En els últims anys, la relació de costos i el desenvolupament de millors i més senzills ordinadors ha fet que el programari tinga un pes i una influència majors a l'hora de mecanitzar una determinada tasca.

Aquest fet i les crisis que va sofrir el programari en els anys 60 han fet que siga molt important desenvolupar el programari d'acord amb algun model estudiat i prèviament desenvolupat (paradigma). Aquest model o paradigma és el que proporciona els passos que s'han de seguir i serveix com a base per al desenvolupament de les diferents fases del cicle de vida del programari. Un dels models més

utilitzats és el model de cicle de vida descendent o en cascada. A més d'aquest, s'han desenvolupat altres models, que són més o menys utilitzats segons el tipus d'aplicació que es desitja dissenyar.

Es defineixen a continuació els principals conceptes utilitzats en el tema.

- **Procés de programari:** és el conjunt d'activitats i resultats associats que produeixen un producte de programari
- **Model de procés del programari:** descripció d'un procés del programari que es presenta des d'una determinada perspectiva. Els models són simplificacions, per tant el model de procés del programari és una abstracció d'un procés real En aquest àmbit existeixen diferents definicions:

  Aproximació lògica a l'adquisició, subministrament, desenvolupament, explotació i manteniment del programari (Estàndard IEEE 1074, 1990).

  Marc de referència que conté els processos, activitats i tasques involucrades en el desenvolupament l'explotació i el manteniment d'un producte de programari, abasten la vida del sistema des de la definició dels requisits fins la fi del seu ús (ISO/IEC 12207-1, 1994).

- **Model o paradigma de cicle de vida del programari:** model que representa diferents formes d'organitzar els mètodes, ferramentes i procediments per a dur a terme el diferents processos del cicle de vida del programari.

9.1.1. El model en cascada o cicle de vida clàssic

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. Aquest és un dels models més àmpliament utilitzat. Com es pot observar a la figura 9.1, els passos principals corresponents són: definició i anàlisi de requisits d'usuari i del sistema, disseny del sistema, codificació i prova individual, prova global del sistema i manteniment.

La primera tasca consisteix a establir els requisits del sistema**,** és a dir, què ha de complir el sistema i identificar quins d'aquests requisits afecten el programari. Per a comprendre la natura dels programes que finalment s'ha d'obtenir, l'enginyer informàtic ha de comprendre l'àmbit, l'entorn i l'abast del sistema que s'ha de desenvolupar, a més de la informació, les funcions i interfícies que es requereixen.

El disseny del programari està enfocat a l'estructura de les dades que el sistema ha de manejar i emmagatzemar, i a l'arquitectura del programari, és a dir, com han d'estar connectats els diferents programes o mòduls, els procediments i les interfícies amb els usuaris o altres sistemes.

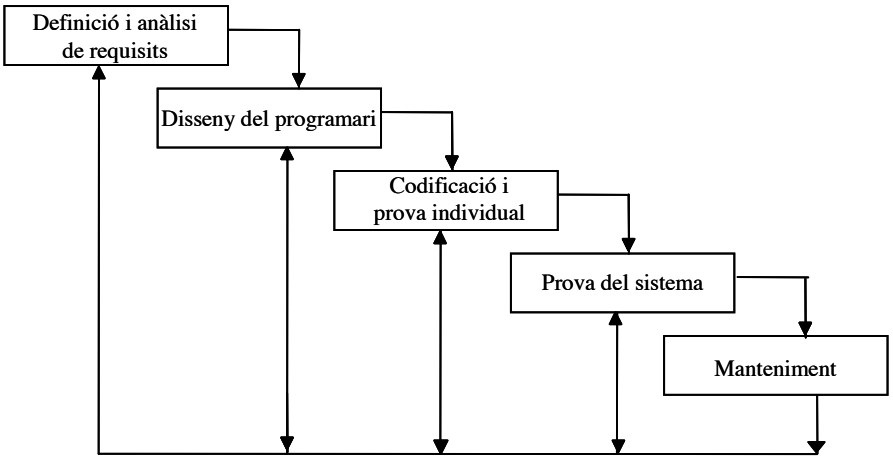


Figura 9.1. Model en cascada o cicle de vida clàssic (Piattini, 2004)

La codificació i prova individual és la traducció del disseny al llenguatge que el maquinari comprèn. Quan el disseny s'especifica de forma detallada, aquesta tasca habitualment és quasi mecànica.

La prova del sistema consisteix a provar tots els mòduls de forma conjunta, la seua interconnexió i la completesa del sistema.

Una vegada el programari es posa en funcionament sofreix canvis, bé pel fet que es troben errors en el funcionament o perquè el programari ha d'adaptar-se als canvis externs que es produeixen en el seu entorn (noves tecnologies, sistemes operatius, legislació, etc.), o perquè els usuaris identifiquen a partir de l'ús ampliacions dels requisits funcionals.

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

- Durant el desenvolupament dels passos rarament se segueix 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.

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.

9.1.2. Desenvolupament de prototips

Normalment la dificultat major per al futur usuari és identificar de forma concreta i completa els requisits del futur sistema, és a dir, conèixer de forma detallada els requisits d'entrada, processos i eixida del 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 a 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.

A la figura 9.2 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.

El disseny ràpid s'enfoca a la representació dels resultats visibles per a l'usuari, com per exemple el disseny de pantalles i d'informes, i condueix a la creació del prototip inicial.

L'avaluació de l'usuari permet determinar requisits no especificats de forma clara inicialment i obtenir una idea més aproximada del que es desitja del sistema final. També facilita a l'enginyer informàtic o programador una millor comprensió del producte que es vol obtenir.

El refinament posterior serveix per a definir de forma concreta i més detallada els requisits del sistema. El prototip obtingut no és el sistema final i ha de rebutjar-se, encara que açò a vegades presenta problemes, tant per part de l'usuari com del programador o enginyer informàtic que l'ha generat.

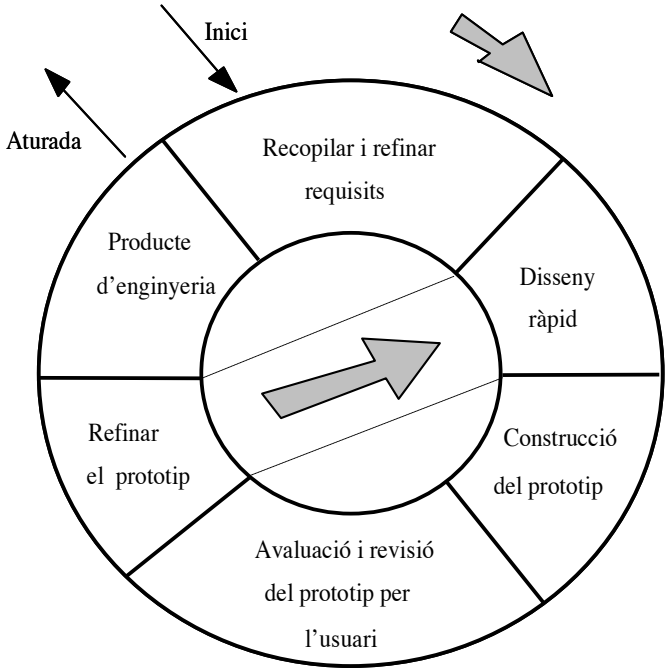


Figura 9.2. Desenvolupament de prototips (Pressman, 1997)

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 tenir en compte 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. Oblidar 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 té 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 tenint en compte tots els aspectes referents a la qualitat, fiabilitat i manteniment del programari.

9.1.3. 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.

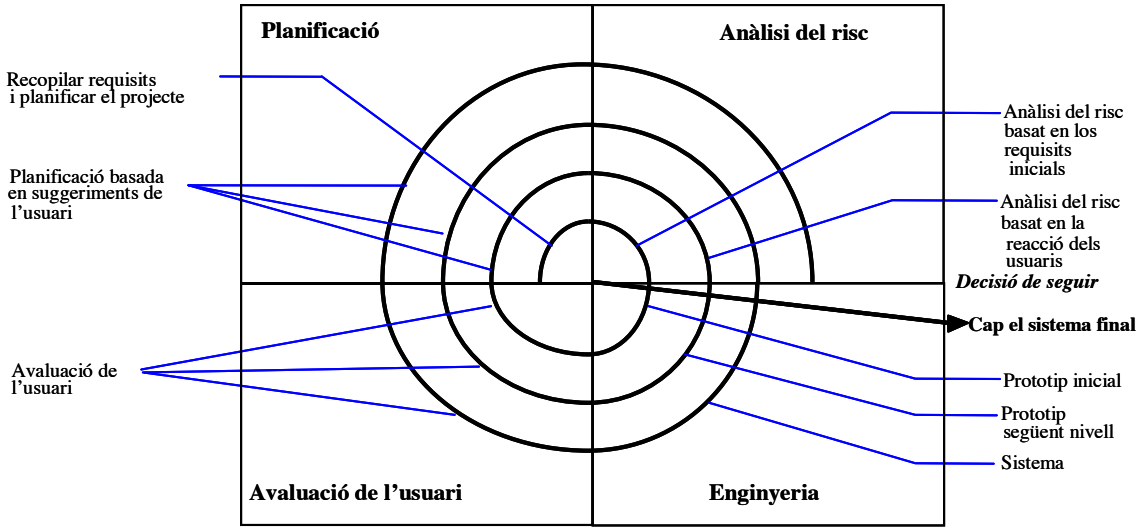


Figura 9.3. Model en espiral de Boehm (Boehm, 1981)

Com es pot observar en la figura 9.3, amb cada iteració representada en el gràfic anterior s'obtenen noves versions de programari o producte cada vegada més completes. En els primers nivells pot utilitzar-se la creació de prototips en l'activitat d'enginyeria que permeten analitzar els objectius i avaluar els riscos. A mesura que el cicle es repeteix les tasques en aquesta activitat augmenten i pot utilitzar-se tant el model de prototips 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 pels 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.

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 a 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.

9.2. Paradigma orientat a objectes

El paradigma orientat a objectes té com a objectiu proporcionar un model o visió de com desenvolupar un sistema informàtic utilitzant la tecnologia orientada a objectes. Aquest llenguatge unificat va ser definit per l'OMG, *Object Management Group*, (OMG, 2010) i proposa un estàndard de facto àmpliament acceptat com a llenguatge de modelització per la indústria del programari. Està suportat i implementat en gran quantitat de ferramentes CASE**.**

UML és un llenguatge visual per especificar, construir i documentar els artefactes d'un sistema i les seues principals característiques són (Booch, 2006):

- És un llenguatge de modelització de propòsit general.
- Pot ser utilitzat amb la majoria de mètodes orientats a objectes i components.
- Pot ser aplicat a qualsevol domini (financer, industrial, telecomunicacions, etc.).
- Pot ser implementat en qualsevol plataforma (J2EE, .NET, etc.).

Els principals objectius d'UML són:

- Modelitzar sistemes des del concepte fins al producte executable utilitzant tècniques orientades a objectes.
- Cobrir qüestions relacionades amb la grandària del sistema a construir.
- Crear un llenguatge de modelització utilitzable tant per les persones com per les màquines.

Encara que s'ha aconseguit un estàndard de facto en quant al llenguatge, no és així en quant al procés de desenvolupament de programari. Basats amb la metodologia orientada a objectes conviuen diferents

processos de desenvolupament estandarditzats i més o menys estesos. *L'Unified Software Development Process* o *Unified Process* (USDP o UP) és un dels processos de desenvolupament orientat a objectes més representatius, originat a partir de les propostes conjuntes de James Rumbaugh, Grady Booch i Ivar Jacobson. Tradicionalment s'anomenava RUP, *Rational Unified Process* (RUP), però la companyia IBM va adquirir els drets d'autor, i per evitar problemes el procés genèric s'anomena USDP o UP de forma simplificada.

L'UP que es mostra en la figura 9.4 es caracteritza per estar dirigit pels casos d'ús, estar centrat en l'arquitectura i ser iteratiu i incremental.

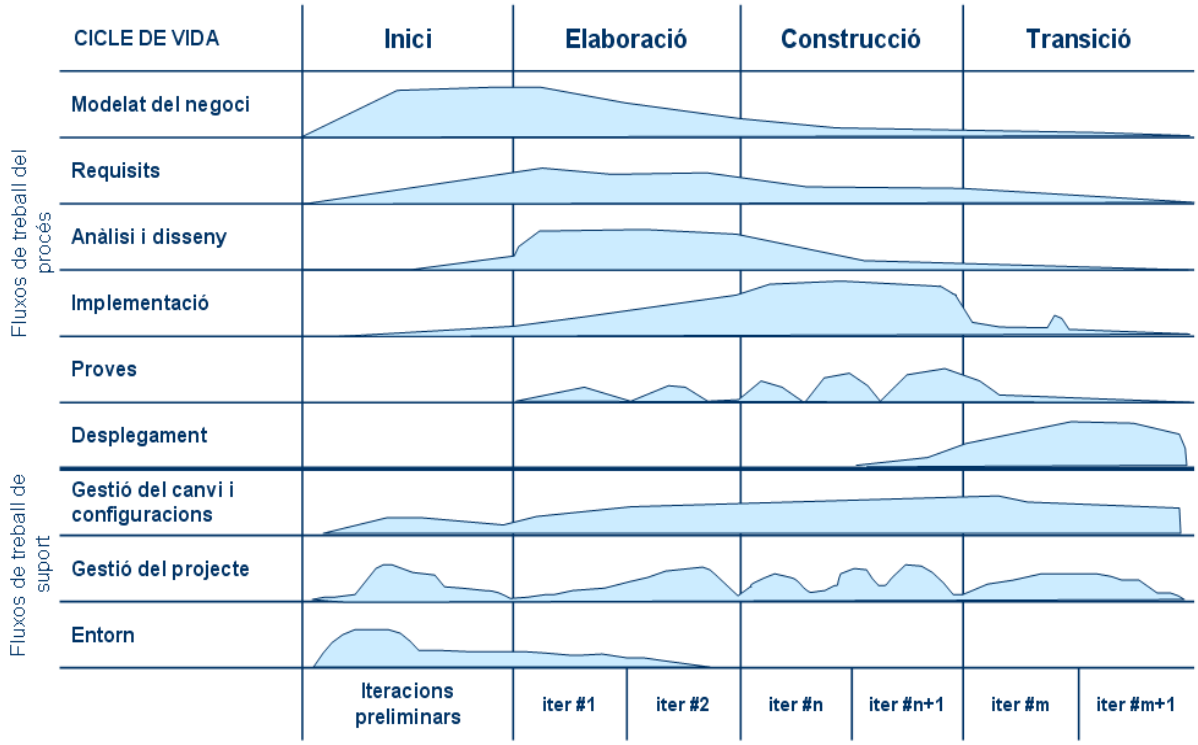


Figura 9.4. Procés de desenvolupament de programari UP (Jacobson, 2000)

UP proposa quatre fases, inici, elaboració, construcció i transició; fluxos de treball del procés i fluxos de suport al procés. Cadascuna d'aquestes fases es va desenvolupant en diferents iteracions.

Les característiques del Procés Unificat de Modelització (UP) són:

- Està dirigit pels casos d'ús, que és un dels models que proporciona UML per modelitzar el sistema des del punt de vista dels usuaris.
- Està centrat en l'arquitectura del sistema.
- És iteratiu e incremental, la qual cosa significa que les fases es desenvolupen en diferents iteracions, i els resultats es refinen successivament en cadascuna de les iteracions.

9.3. Metodologies avalades per organismes oficials

Una metodologia és un conjunt de filosofies, fases, procediments, regles, tècniques, ferramentes, documentació i aspectes de formació per als desenvolupadors de sistemes d'informació. Es poden diferenciar: metodologies estructurades i metodologies orientades a objectes. A banda hi ha diferents metodologies definides per organismes públics de diferents països com són:

- Metodologia SSADM, d'origen anglès.
- Metodologia MERISE, d'origen francès.
- Metodologia MÉTRICA, d'origen espanyol.

9.3.1. Metodologia MÉTRICA

MÉTRICA és una metodologia de desenvolupament de sistemes d'informació en la qual es defineixen un conjunt de fases, procediments, regles, tècniques, ferramentes, documentació i aspectes que faciliten la seua construcció als desenvolupadors de sistemes d'informació. La metodologia va ser desenvolupada pel Ministeri d'Administracions Públiques i l'actual *Consejo Superior de Administración Electrónica*, CSAE (MAP CSAE, 2010) per a donar resposta als problemes que es produïen al voltant del desenvolupament de sistemes informàtics a l'administració pública, com per exemple:

- Es produïa poca o no cap documentació mentre es desenvolupava el sistema, la qual cosa feia difícil el seu desenvolupament, implantació i sobretot el posterior manteniment.
- La falta de comunicació entre els desenvolupadors donava lloc a errors, incongruències i al fet que no estigueren recollits tots els requisits dels usuaris.

Aquesta metodologia parteix de les línies marcades per altres metodologies anteriors, com per exemple la metodologia pública britànica SSADM o la francesa MERISE. Actualment es troba en la versió 3 i els seus objectius es poden resumir en els següents:

- Definir un marc estratègic d'ajuda al desenvolupament de sistemes d'informació de qualitat que s'ajusten als requisits proposats pels usuaris.
- Millorar la productivitat i eficiència dels departaments de sistemes i tecnologies de la informació.
- Facilitar la comunicació dels equips de desenvolupament, de manera que s'eviten errors durant el desenvolupament i la implantació de sistemes i aquests siguen mantenibles i de fàcil ús.
- La metodologia MÉTRICA (MAP MÉTRICA, 2010), en la versió 3 està orientada als processos que es descomponen en activitats i tasques. La versió anterior de MÉTRICA, la 2.1 (Gaitero,

1996), s'estructurava en fases, que es dividien en mòduls, activitats i tasques successivament. Aquestes fases tenien una idea de globalitat i seqüencialitat, i per tant definien els treballs globals que s'havien de dur a terme per al desenvolupament d'un sistema i l'ordre que s'havia de seguir. Mentre que en la versió 3 les diferents activitats i tasques d'un procés poden realitzar-se en ordre diferent al proposat o fins i tot en paral·lel, però el procés no es dóna per acabat fins que no finalitzen totes les activitats i tasques d'aquest.

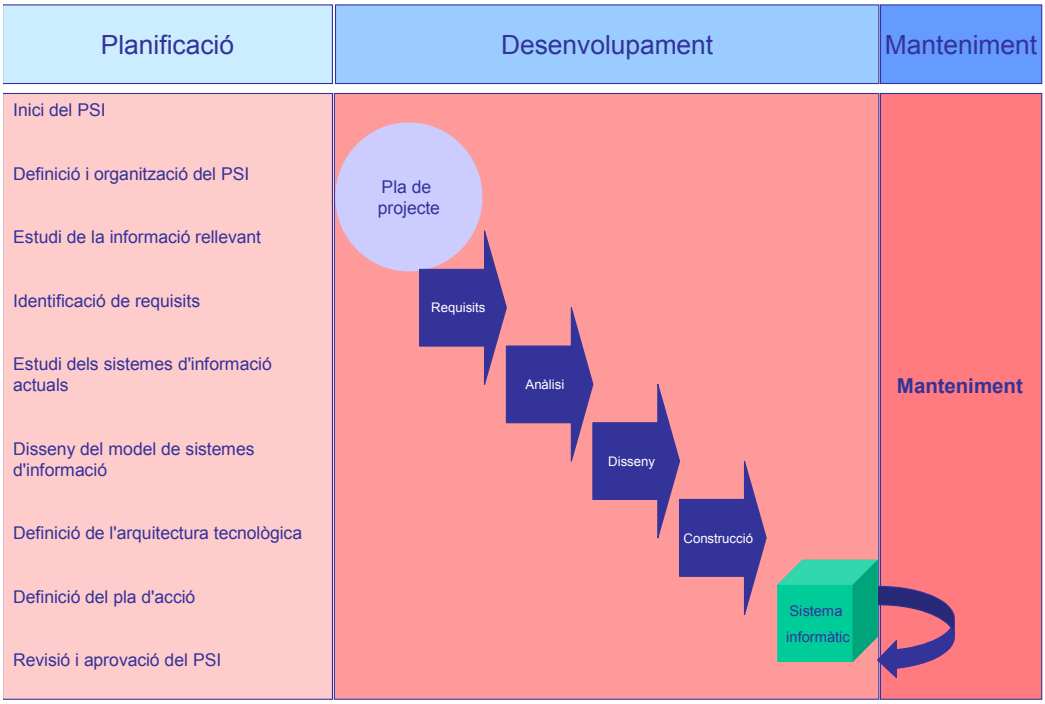


Figura 9.5. Processos de la metodologia MÉTRICA

La figura 9.5 mostra els processos que es defineixen que són : Planificació de sistemes d'informació, Desenvolupament de sistemes d'informació, i Manteniment de sistemes d'informació.

9.4. Metodologies àgils

Les metodologies àgils, de recent aparició, han sorgit com a resposta a la rigidesa i burocràcia que impliquen algunes de les metodologies existents per al desenvolupament de programari, com les descrites en els apartats anteriors. Aquestes últimes estan orientades a projectes de considerable grandària i tracten d'assegurar la qualitat, tant del procés com del producte final, mitjançant la generació de documentació, condicions fortes en els contractes de desenvolupament, planificació detallada del procés, etc. Tots aquests condicionants, que són necessaris en projectes grans i on l'equip humà que hi treballa és nombrós, fan que a l'hora d'aplicar-los en projectes més xicotets la rigidesa i el control siguen punts en contra del seu ús, que es converteix en un procés rígid i poc productiu. Com a

alternativa sorgeix el terme àgil, amb l'ànim d'alleugerar el procés de desenvolupament i dotar-lo només de les activitats que suposen un valor afegit.

L'aparició d'aquestes metodologies cal lligar-la al *Manifesto for Agile Software Development* (Agil Manifesto, 2010) com a millor forma de desenvolupar 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 en tenen, sinó que realment aquestes metodologies valoren de forma més significativa els elements de l'esquerra, per damunt dels de la dreta.

Algunes d'aquestes metodologies són: *eXtreme Programming* (XP), *Scrum Manager* i *Crystal*. Es poden trobar altres exemples en (Pressman, 2010).

9.4.1. eXtreme Programming (XP)

El primer projecte desenvolupat seguint la metodologia *eXtreme Programming* està datat el 1996. *Extreme Programming* (Extreme Programming, 2010) és un procés àgil de desenvolupament de programari, enfocat a aconseguir la satisfacció del client. El procés propugna que els desenvolupadors poden respondre al canvi dels requisits dels clients, fins i tot si aquests es donen en etapes finals del cicle de vida del programari.

A més a més, en aquest procés s'emfatitza el treball en equip. De forma que els gestors, clients i desenvolupadors són tractats per igual, amb la finalitat de crear un entorn col·laboratiu i productiu, i en el qual l'equip s'autoorganitza per tal de solucionar els problemes que van sorgint de la forma més eficient possible.

Els principis en els quals es basa XP per tal de millorar el desenvolupament del programari són quatre:

- Comunicació en tot moment entre els clients i els programadors.
- Simplicitat en el disseny.
- Realimentació a l'hora de validar i verificar el programari.
- Coratge o ànim per lliurar el sistema als clients tan prompte com siga possible i implementar els canvis que aquests suggereixen.

9.4.2. Scrum Manager

Scrum Manager és una metodologia àgil per al desenvolupament de sistemes informàtics basada en principis àgils i flexibles en lloc dels industrials i predictius. Aquest marc té entre les seues premisses el lliurament al client de xicotetes parts del producte final amb valor afegit, les quals s'anomenen increment. En el procés de desenvolupament *Scrum* els participants adopten tres rols:

1. ***Scrum manager***: encarregat de gestionar el procés de desenvolupament, organitzar les reunions, servir d'enllaç entre l'equip i el propietari del producte, etc.
2. **Propietari del producte**: representa el client que coneix els requisits que ha de tenir el producte final.
3. **Equip:** format pels desenvolupadors amb una formació multidisciplinar, organització autònoma, i que són capaços de dur a terme totes les tasques del desenvolupament: anàlisi, disseny, implementació, etc.

El conjunt de requisits que ha de tenir el producte final, és a dir el sistema informàtic a desenvolupar, s'anomena *product backlog* (o pila de producte). Aquest conjunt de requisits es prioritza en la reunió de planificació, i a partir d'ella s'obté el *sprint backlog* (o pila de *sprints*).

Cada *sprint* és un conjunt de tasques que s'ha de dur a terme per obtenir alguns dels requisits de la pila de producte, els quals s'han de poder desenvolupar en un termini entre dues i quatre setmanes. Per determinar els requisits que s'inclouen en un *sprint* és realitza la reunió de planificació del *product backlog,* en la qual el propietari del producte informa a l'equip dels elements de la pila del producte que vol que es desenvolupen.

Per a controlar la consecució del *sprint* es du a terme una reunió diària (molt breu, fins i tot dempeus) en la qual l'equip i el *scrum manager* intercanvien possibles incidències, l'avançament del *sprint*, possibles dificultats, etc. Una vegada el *sprint* és finalitzat, l'equip realitza una revisió, per demostrar al propietari del producte el funcionament de la nova part de l'aplicació desenvolupada, la qual s'anomena increment (Palacio, 2009).

Bibliografia

AGIL MANIFESTO (2010). *Manifesto for Agile Software Development,* http://agilemanifesto.org/.

BECK, K. (2003). *Test-Driven Development by Example,* Addison Wesley.

BOEHM, B. W., (1981). *Software Engineering Economics,* Englewood Vlifs, Prentice Hall.

BOOCH, G., J. Rumbaugh, I. Jacobson (2006). *The Unified Modeling Language User Guide,* Second Edition. Addison Wesley.

DURAN, A., B. Bernárdez (2002) *Metodología para la Elicitación de Requisitos de Sistemas Software (versión2.3)*
http://www.lsi.us.es/~amador/publicaciones/metodologia_elicitacion_2_3.pdf.zip

EXTREME PROGRAMMING (2010). http://www.extremeprogramming.org/

MAP CSAE. Consejo Superior de Administración Electrónica, Ministerio de política territorial y administración pública, http://www.csae.map.es/

MAP MÉTRICA. Versió 3 *Metodologia de Planificación, Desarrollo y Mantenimiento de Sistemas de Información,* http://www.csae.map.es/csi/metrica3/index.html

GAITERO, D. (1997). *Metodologia Métrica, un enfoque práctico*. Editorial Everest.

IDEF *Integrated DEFinition methods*. (2010). http://www.idef.com.

IEEE Institute of Electrical and Electronics Engineers (1990). *Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries,* Institute of Electrical and Electronics Engineers.

IEEE Institute of Electrical and Electronics Engineers (1993). *IEEE 1219: Standard for Software Maintenance*.

IFPUG Punts de funció (2009). *Informació completa sobre els punts de funció,* http://www.ifpug.org

JACOBSON, I., G. Booch, J. Runbaugh, (2000). *El Proceso Unificado de Desarrollo de Software* (UML). Addisson Wesley.

KAROLAK, D. W. (1996). *Software Engineering Risk management,* IEEE Computer Society Press.

LEHMAN, M.M. (1997). *Laws of Software Evolution Revisited, pos*. pap., EWSPT96, Oct. 1996, LNCS 1149, Springer Verlag, 1997, pp. 108-124.

OMG (2010). UML *UML Superstructure*.

OMG (2011). *Object Management Group* http://www.omg.org/.

PALACIO, J., C. Ruata (2009). *Scrum Manager: Proyectos–Formación*. Versió 1.3 http://www.safecreative.org/work/0910244743710

PIATTINI, M.G., J. Villalba, F. Ruiz, T. Bastanchury, M. Polo, M. A. Martínez, C. Nistal (2000). *Mantenimiento del Software*, Ed. Ra-Ma.

PIATTINI, M. G., J. A. Calvo-Manzano, J. Cervera, L. Fernández (2004). *Análisis y diseño de Aplicaciones Informáticas de Gestión. Una perspectiva de Ingeniería del Software*, Ed.Ra-Ma.

PRESSMAN, R. S. (1997). *Ingeniería del Software. Un enfoque práctico* (4ª edición), Mc Graw-Hill.

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

SOMMERVILLE, I. (2005). *Ingeniería de Software* (7ª edició), Addison Wesley.

TERMCAT (2011). Centre de Terminologia Catalana. http://www.termcat.es/

YOURDON, E. (1993). *Análisis Estructurado Moderno*, Prentice Hall.

Lectures recomanades

BROOKS, F. P. (1995). *The mythical man-month: Essays on Software Engineering,* Addison Wesley.

FINKELSTEIN, C. (1992). *An Introduction to Information Engineering. from Strategic Planning to Information Systems,* Addison Wesley.

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