Programari

Jeroni Vivó i Plans
Ramon Costa i Pujol

PID_00153130

Universitat Oberta de Catalunya
www.uoc.edu

Cap part d'aquesta publicació, incloent-hi el disseny general i la coberta, no pot ser copiada, reproduïda, emmagatzemada o transmesa de cap manera ni per cap mitjà, tant si és elèctric com químic, mecànic, òptic, de gravació, de fotocòpia o per altres mètodes, sense l'autorització prèvia per escrit dels titulars del copyright.

Índex

Introducció

El programari (*software*) d'un sistema informàtic constituït pel conjunt de programes executables en aquell sistema.

Els sistemes informàtics, és a dir, la combinació del maquinari (*hardware*), com són els ordinadors, la infraestructura de les xarxes i els sistemes de comunicació (cablejat, etc.), amb el programari, han permès millorar molts aspectes de la societat, ja que l'aplicació d'aquests sistemes engloba tots els àmbits, tant genèrics com, per exemple, de:

- Negocis.
- Medicina.
- Educació.
- Ciència i tecnologia.
- Enginyeria i arquitectura.

O si es tracta d'àmbits propis de les unitats funcionals o departamentals de les organitzacions, com, per exemple:

- Producció.
- Logística i distribució.
- Atenció al client.
- Formació i gestió dels RH.
- Gestió de la planificació.

Tot aquest conjunt de programari es desenvolupa utilitzant uns llenguatges de programació i seguint unes pautes i metodologies conegudes com a *enginyeria del programari*.

La construcció de programes d'ordinador utilitzant directament el llenguatge natiu de la màquina, tal com es feia en els inicis de la informàtica, té molts inconvenients. L'única manera de superar aquests inconvenients consisteix a desenvolupar llenguatges adequats per a programar qualsevol ordinador i que es puguin traduir al llenguatge que entengui una màquina concreta.

Un dels apartats d'aquest mòdul consistirà a analitzar què caracteritza un llenguatge de programació.

Llengüatge de la programació

Un llenguatge de programació és un conjunt de normes sintàctiques englobades sota un nom (com pot ser C, Pascal, Java, etc.) que especifiquen com i quan es pot utilitzar un conjunt d'instruccions determinat.

A partir dels llenguatges de programació es poden dissenyar programes, que després es converteixen a codis màquina per part del compilador corresponent del llenguatge per a

la seva execució i oferir una visió general de com es produeix la traducció de programes escrits en un llenguatge d'alt nivell al llenguatge màquina d'un ordinador.

No pretenem, en aquesta assignatura, conèixer amb detall els llenguatges de programació, sinó oferir una visió abstracta d'aquests llenguatges que ajudi a la comprensió dels aspectes més interessants (com ara els àmbits d'aplicació).

El disseny, desenvolupament i implantació d'un sistema informàtic engloba tant els aspectes de programari (disseny i desenvolupament d'aplicacions, parametrització de programes existents, etc.) com de maquinari (instal·lació i configuració d'ordinadors i xarxes, parametrització de servidors de correu, de fitxers, de xarxa, etc.).

La gestió de totes aquestes tasques es fa seguint els paràmetres de la gestió de projectes.

Una part important de tot sistema informàtic és l'adquisició del programari requerit per a implantar la solució demanada (servidors de xarxes, de fitxers, programes de desenvolupament, aplicacions ja fetes, etc.).

Implantar, posar en marxa una instal·lació informàtica

També s'utilitza aquesta paraula per a referir-se a la fase dins el cicle de vida d'un programa durant la qual es codifica en un llenguatge de programació determinat.

Aquesta adquisició del programa s'ha de fer atenent unes polítiques de compra i ús de llicències, marcades pels fabricants.

En definitiva, l'objectiu d'aquest mòdul és introduir l'estudiant en els conceptes principals:

- del programari i les seves aplicacions,
- el desenvolupament del programari i els llenguatges de programació,
- la gestió de projectes,
- l'ús de les llicències de programari.

Objectius

Els **objectius generals** que l'estudiant pot assolir són els següents:

1. Conèixer el concepte de programari i les seves principals aplicacions.
2. Conèixer les principals eines disponibles per al desenvolupament de programari, en especial pel que fa a llenguatges de programació i a gestió de projectes.
3. Conèixer les tipologies de llicències de programari i les seves implicacions pràctiques.

Aquests objectius generals es desglossen en els **objectius específics** següents:

1. Definir el concepte de programari i enumerar les seves principals característiques.
2. Enumerar les principals aplicacions del programari en funció de la seva finalitat.
3. Descriure el concepte de desenvolupament de programari per a crear programes i aplicacions.
4. Enumerar les principals característiques i diferències entre els llenguatges màquina, assemblador i d'alt nivell.
5. Conèixer un conjunt del principals llenguatges de programació i les diferències.
6. Descriure les principals característiques d'una eina CASE.
7. Conèixer els principals conceptes i elements de l'enginyeria del programari.
8. Identificar els principals conceptes de la gestió de projectes en general i enumerar les principals fases.
9. Descriure els principals conceptes de la planificació d'un projecte i el seu seguiment: tasques, recursos, eines i tècniques.

10. Identificar les principals característiques d'un projecte informàtic i enumerar les seves principals fases i tasques.

11. Identificar i enumerar els principals rols dels membres d'un equip de treball en un projecte informàtic.

12. Descriure el concepte de llicències de programari.

13. Llistar diferents possibilitats i opcions en la compra i ús de llicències de programari.

14. Enumerar les principals característiques de les llicències *freeware* i *sharewa-re*.

1. Programari: una visió general

El programari d'un sistema informàtic, més conegut com *software*, està constituït pel conjunt de programes executables i fitxers necessaris perquè funcioni.

Dins del programari s'inclouen eines tan diverses com:

- El sistema operatiu.
- Les interfícies d'usuari (sistema de finestres, per exemple).
- Llenguatges de programació (Pascal o l'entorn Visual Studio, per exemple).
- Eines o utilitats (compressors de fitxers, antivirus...).
- Aplicacions de qualsevol especialitat i tipus (processadors de text, eines de disseny i gestió d'imatges...).
- Fitxers de dades i bases de dades amb què treballen aquests programes; contenen les dades i les estructures de dades necessàries per al funcionament del programari i les generades pel seu mateix ús.

Una part important del programari d'un sistema la formen les eines o utilitats. Dins d'aquest paquet de programes s'hi pot incloure programari tan variat com programes de full de càlcul, els tractaments de textos com aplicacions d'ús comú, els gestors de bases de dades, els paquets de "tot en un" (eines conegudes com de productivitat personal), els paquets de gestió i altres paquets d'utilitats ja esmentats anteriorment.

El **programari** és tot el conjunt de programes o utilitats que s'executen en un ordinador.

Hi ha moltes empreses que es dediquen a desenvolupar aquest tipus d'aplicacions, és un mercat molt dinàmic i amb una competència ferotge, això provoca que l'aparició de noves versions i revisions sigui constant. Aquestes noves versions volen corregir errors antics i sovint també presenten noves funcionalitats, moltes de les quals les demanen els mateixos usuaris.

Les computadores tenen la capacitat de dur a terme moltes i diverses tasques, sempre que tinguin el programari adequat.

Els ordinadors permeten fer feines que abans requerien un personal molt especialitzat en diversos camps (delineants, analistes financers, programadors...), facilitant l'automatització i simplificació de les tasques de suport d'aquests professionals. El programari no pot substituir el coneixement dels delineants, ana-

listes financers o programadors, però sí que pot automatitzar processos rutinaris i donar elements per facilitar-ne els altres, de manera que la feina es faci d'una manera més eficient.

Avui dia, la gran majoria d'aquestes feines es poden realitzar més fàcilment mitjançant un ordinador personal, el programari adequat i una mínima formació en l'ús i el treball d'aquest programari (sempre que l'usuari tingui un nivell de competència adequat en l'àmbit temàtic d'utilització).

A més a més, els programaris d'intel·ligència artificial permeten anar més enllà de la simple automatització a l'hora d'ajudar a resoldre problemes complexos.

Una primera classificació del programari permetria diferenciar dues grans categories: el programari de sistema i el programari d'aplicació. És a dir, les aplicacions i utilitats que serveixen per a controlar el maquinari (sistemes operatius, servidors de bases de dades, servidors de correu electrònic, servidors de fitxers i impressores...) i els programes que treballen en aquest programari i maquinari (aplicacions empresarials, personals...).

Una possible organització del programari podria ser distingir entre el programari de sistema i el programari d'aplicació.

En aquesta unitat tractarem de les diferents aplicacions del programari, proposant una classificació possible d'aquests programes.

1.1. Aplicacions del programari

El programari es pot aplicar a qualsevol situació en què s'hagi definit prèviament un conjunt específic de passos procedimentals, cosa que es coneix com a *algorismes* (un problema determinat s'intenta descompondre en un conjunt de problemes menors, la qual cosa permet arribar a crear algorismes molt potents).

Per tal de determinar la naturalesa del programari, cal tenir en compte tant el contingut com la informació gestionada.

D'una banda, el contingut té a veure amb el significat i la forma de la informació d'entrada i de sortida.

Aplicacions bancàries

Per exemple, moltes aplicacions bancàries utilitzen unes dades d'entrada molt estructurades (una base de dades) i produeixen "informes" amb formats determinats.

El programari que controla una màquina automàtica (per exemple, un control numèric) actua sobre elements de dades discretes amb una estructura limitada i produeix ordres concretes per a la màquina en una ràpida successió.

D'altra banda, en canvi, la gestió de la informació també ha de tenir en compte la predictibilitat de l'ordre i del temps d'arribada de les dades.

Un programa d'enginyeria accepta dades que estan en un ordre predefinit, executa l'algorisme sense interrupció i produeix dades resultants en un informe o format gràfic.

Programa d'enginyeria

Exemples d'aquesta mena d'aplicacions són:

- Consolidació de moviments mensuals de les comandes d'una empresa.
- Gestió de les nòmines.

En canvi, un sistema operatiu multiusuari rep entrades que tenen un contingut variat o que es produeixen en instants arbitraris, executa algorismes que es poden interrompre per condicions externes i produeix una sortida que depèn d'una funció de l'entorn i del temps. Les aplicacions amb aquestes característiques s'anomenen indeterminades.

Aplicacions indeterminades

Altres exemples d'aplicacions indeterminades serien:

- Sistemes de monitoratge de plantes industrials.
- Aplicacions de control d'alarmes i seguretat en un edifici.

Per tal de determinar la naturalesa del programari, cal tenir en compte tant el contingut com la informació que gestiona.

És molt difícil establir categories genèriques per les aplicacions del programari. A mesura que augmenta la complexitat del programari es fa més complicat establir fronteres nítidament separades.

Per tal de fer un repàs de possibles aplicacions del programari, s'ha decidit categoritzar-les en els apartats següents:

- Programari de sistemes.
- Programari de temps real.
- Eines d'ús personal.
- Programari d'enginyeria i científic.
- Eines de gestió i redisseny de processos organitzatius.
- Aplicacions d'intel·ligència artificial.

S'hauria pogut classificar el programari segons altres taxonomies o detallar més aquesta organització (gestió del coneixement, control de la producció, sistemes de gestió d'informació gràfica...) o per la seva aplicació en l'àmbit d'Internet i les relacions entre empreses i usuaris (B2B[1], B2C[2]...) però s'ha optat per introduir, en aquest mòdul, aquests sis tipus d'aplicacions.

(1)B2B és l'acrònim de les paraules angleses *business to business* ('negoci a negoci'). Abreviatura que s'utilitza molt en el ciberespai per a referir-se a totes les relacions comercials que estableixen les empreses les unes amb les altres per mitjà d'Internet.

(2)B2C és l'acrònim de les paraules angleses *business to consumer* ('negoci a consumidor'). Abreviatura que s'utilitza molt en el ciberespai per a referir-se a les relacions comercials que estableixen les empreses amb els consumidors finals per Internet.

1.1.1. Programari de sistemes

Són un conjunt de programes que s'han escrit per a servir altres programes.

Alguns programes de sistemes com els compiladors o els editors i les utilitats de gestió d'arxius processen estructures d'informació complexes però determinades.

Altres aplicacions de sistemes són, per exemple, determinats components del sistema operatiu, utilitats de gestió dels perifèrics o processadors de telecomunicacions que processen dades molt indeterminades.

En qualsevol cas, l'àrea del programari de sistemes es caracteritza per:

- Una forta interacció amb el maquinari (*hardware*) de la computadora.
- Concurrència d'usuaris.

Aquests dos elements combinats fan que aquest tipus de programari hagi de preveure la compartició de recursos i hagi de fer a la vegada una gestió dels processos molt acurada per deixar satisfets tots els usuaris, entenent com a usuaris els programes que s'executen per sobre del programari de sistema.

1.1.2. Programari de temps real

El programari que mesura, analitza i controla successos del món real a mesura que s'esdevenen s'anomena *de temps real*.

Entre els elements del programari de temps real s'inclouen:

- Un component d'adquisició de dades que recull i dóna format a la informació rebuda de l'entorn extern.
- Un component d'anàlisi que transforma la informació segons ho requereixi l'aplicació.
- Un component de control/sortida que respongui a l'entorn extern.

- Un component de monitoratge que coordina tots els altres components, de manera que es pugui mantenir la resposta en temps real (típicament en el rang d'1 mil·lisegon a 1 minut).

Cal tenir en compte que el concepte "temps real" significa que el sistema ha de respondre dins uns lligams estrictes de temps per a evitar que es produeixi cap desastre.

Exemples de programari de temps real

Alguns exemples són els sistemes de monitoratge d'una planta industrial que controlen totes les màquines que intervenen en la cadena de producció, o el sistema de control de les alarmes d'un edifici o d'una planta nuclear o, més comuns per a l'usuari del carrer, el programari d'un caixer automàtic o el d'un peatge.

1.1.3. Eines d'ús personal

Des de l'aparició dels ordinadors personals (PC) al final de la dècada dels setanta i l'inici dels vuitanta s'han desenvolupat aplicacions i programes pensats i dissenyats per a ajudar el treball personal i les tasques quotidianes, tant a l'oficina com a la llar:

- Processadors de textos.
- Fulls de càlcul.
- Gràfiques per ordinador.
- Reproductors multimèdia de vídeo i so.
- Jocs i aplicacions d'entreteniment.
- Gestió d'agendes personals.
- Aplicacions financeres, de negocis i personals.

Aquestes, i moltes més, són algunes dels centenars d'aplicacions dissenyades per a ajudar les persones en les seves tasques quotidianes.

1.1.4. Programari d'enginyeria i científic

El programari d'enginyeria i científic es caracteritza pels algorismes optimitzats per a la manipulació matemàtica de les dades.

Les aplicacions d'aquesta tipologia van des de:

- suport a l'observació astronòmica,
- predicció meteorològica,
- monitorització de processos industrials,
- aplicacions de suport a la diagnosi mèdica,
- aplicacions de suport a l'R+D,
- etc.

fins a aplicacions més properes a la gent del carrer com:

- control i sincronització de semàfors,
- càlcul d'estructures d'edificis,
- control del flux del corrent elèctric,
- etc.

En els darrers anys, les aplicacions d'enginyeria/ciència s'han allunyat dels algorismes convencionals numèrics i el disseny assistit per ordinador (de l'anglès CAD), la simulació de sistemes o altres aplicacions interactives també tenen característiques del programari de temps real i, fins i tot, del programari de sistemes. En aquests moments moltes aplicacions científiques permeten tractar, en temps real, mesures i dades com, per exemple, el seguiment de la situació dels elements meteorològics.

1.1.5. Eines de gestió i redisseny de processos organitzatius

La gestió d'informació comercial i corporativa constitueix la més gran de les àrees d'aplicació del programari.

Sistemes com la gestió de nòmines, la compatibilitat, la gestió d'inventaris, etc. han evolucionat cap al programari de sistemes d'informació (SI), que accedeix a una o més bases de dades grans que contenen informació de l'organització.

Les aplicacions en aquesta àrea reestructuren les dades existents per facilitar les operacions comercials o gestionar la presa de decisions. A més de les tasques convencionals de processament de dades, les aplicacions de programari de gestió també realitzen càlculs interactius (per exemple, el processament de transaccions en punts de venda).

L'evolució i l'aplicació d'aquest programari ha anat donant suport, cada vegada més, a altres funcions de l'organització i els seus departaments, convertint-se en eines de gestió de la major part dels processos de les companyies:

- El procés comercial.
- El procés de compres.
- Les operacions de producció i fabricació.
- La gestió dels recursos humans.

Per aquestes tasques han aparegut solucions específiques com:

- SCM (*supply chain management*, 'gestió de la cadena de subministrament').
- CRM (*customer relationship management*, 'gestió de les relacions amb el client').
- BPR (*business process reengineering*, 'reenginyeria de processos de negoci').
- e-learning (formació per mitjà d'eines telemàtiques).
- e-rrhh (solucions de suport a la gestió del departament de recursos humans).
- *Content management* ('gestió de continguts i documents').

- *Knowlegde management* ('gestió del coneixement').

1.1.6. Aplicacions d'intel·ligència artificial

El programari d'intel·ligència artificial (IA) fa servir algorismes no numèrics per a resoldre problemes complexos pels quals no és adequat el càlcul o l'anàlisi directa.

Diferents àrees de la intel·ligència artificial (IA) són els sistemes experts, també anomenats *sistemes basats en el coneixement*: reconeixement de patrons, reconeixement d'imatges, de veu i d'escriptura.

Hi ha diferents maneres d'implementar aplicacions d'intel·ligència artificial: la que més s'aproxima al funcionament del cervell humà i dels animals són els algorismes que implementen xarxes neuronals que simulen, fins allà on la tecnologia i els coneixements humans permeten, l'estructura de processos del cervell (les funcions de la neurona biològica), que a la llarga pot dur a una classe de programari que pugui reconèixer patrons complexos i aprendre "d'experiències" passades.

Les aplicacions d'intel·ligència artificial s'usen en àmbits com l'enginyeria, la medicina, l'àmbit militar i l'economia. També s'han usat per a jocs d'estratègia com els escacs i per a fer videojocs.

2. Desenvolupament de programari

Tot programari ha d'haver estat dissenyat i creat, o com es diu en termes informàtics, desenvolupat.

Per a desenvolupar aquests programes i aplicacions, els "programadors" disposen d'un conjunt de llenguatges de programació o desenvolupament.

Hi ha una gran quantitat de llenguatges de desenvolupament. Alguns s'han dissenyat per a crear programes d'un tipus específic o per a un tipus d'aplicacions concretes (aplicacions científiques i tècniques, intel·ligència artificial, gestió, etc.) i d'altres són d'aplicació "general".

Els programes desenvolupats en aquests llenguatges s'han de traduir, posteriorment, al codi font (un llenguatge entenedor per l'ordinador i el seu processador). Hi ha dos tipus de traductors, els compiladors i els intèrprets.

Per tal d'optimitzar aquest procés de desenvolupament i minimitzar, d'altra banda, els possibles errors en la producció del programari, s'ha desenvolupat el que s'anomena *enginyeria del programari*, que engloba un conjunt de mètodes, eines i procediments que faciliten la feina als programadors.

Els objectius d'aquest apartat són:

- Introduir l'alumne en els principals conceptes dels llenguatges de programació.
- Fer un repàs de les "generacions" de llenguatges.
- Enumerar els principals llenguatges de desenvolupament existents i la seva aplicació.
- Indicar les principals diferències entre el llenguatge màquina, el llenguatge assemblador i els llenguatges d'alt nivell i les seves característiques.
- Introduir les eines CASE.
- Enumerar els principals conceptes de l'enginyeria del programari.

2.1. Llenguatges de programació

Els desenvolupadors de programari, o "programadors", utilitzen llenguatges de programació per a crear i desenvolupar els paquets de programari o programes com, per exemple, els processadors de textos, els fulls de càlculs o els sistemes de missatgeria o les aplicacions web que els diferents usuaris utilitzen quan treballen amb els seus ordinadors.

Tota aplicació informàtica ha estat desenvolupada utilitzant un llenguatge de programació.

Es componen dels elements següents:

- un lèxic, que és el conjunt de símbols permesos o vocabulari,
- una sintaxi, que són les regles que indiquen com realitzar les instruccions del llenguatge,
- i una semàntica, que correspon a les regles que permeten determinar el significat de qualsevol construcció del llenguatge.

Els programes que pot executar l'ordinador s'han de "redactar" en el llenguatge nadiu del processador.

És a dir, cada instrucció ha d'estar en codi binari i directament relacionada amb els circuits del processador.

Ara bé, programar instruccions completament en codi binari, és un procés massa lent i subjecte a errors.

Per això, els llenguatges de programació s'han dissenyat per a permetre que el desenvolupador escrigui les instruccions semblants a un llenguatge humà, gairebé sempre l'anglès.

Aquestes instruccions es converteixen, després, en el codi binari corresponent mitjançant uns programes anomenats *compiladors*.

Tal com veurem més endavant, els compiladors tradueixen les instruccions d'un llenguatge de programació d'alt nivell a codi binari de baix nivell, com una persona podria traduir d'un idioma a un altre, per exemple, del català al castellà.

D'aquesta manera, un compilador o traductor, converteix els programes que el desenvolupador ha programat, utilitzant un llenguatge d'alt nivell en un codi font i un codi objecte.

El codi font és el conjunt d'instruccions escrites en un llenguatge de programació, mentre que el codi objecte és el conjunt d'instruccions binàries que pot executar l'ordinador.

Els programes que l'ordinador pot executar han d'estar "redactats" en el llenguatge nadiu del seu processadors. És per això que els programes desenvolupats utilitzant els llenguatges de programació d'alt nivell s'han de traduir a un codi font i un codi objecte, entenedors per al processador de l'ordinador.

2.1.1. Llenguatges màquina, assemblador i alt nivell

En aquest apartat, s'aprofundeix en les principals característiques dels llenguatges màquina, assemblador i d'alt nivell.

Llenguatge màquina

Tal com s'ha esmentat en la introducció, el llenguatge màquina és l'únic llenguatge que entén directament l'ordinador, o més ben dit, el processador, ja que la seva estructura està totalment adaptada als circuits de la màquina.

Per tant, la seva forma d'expressió està molt allunyada de la comprensió i anàlisi de les persones.

La programació amb aquest tipus de llenguatges és molt complicada, ja que requereix un coneixement profund de l'arquitectura física de l'ordinador, per part del programador.

Per contra, el codi màquina permet que el desenvolupador pugui utilitzar la totalitat dels recursos que ofereix l'ordinador, de manera que es poden obtenir programes molt eficients, tant en temps d'execució com en ocupació de memòria.

Aquests llenguatges constitueixen la primera generació de llenguatges de programació.

Les seves principals característiques són les següents:

- Les instruccions s'expressen en codi binari, codificades com cadenes de zeros (0) i uns (1), cosa que fa que sigui molt difícil d'entendre i modificar (mantenir).

- Les dades es referencien per mitjà de les adreces de memòria on es troben i no per noms de variables o constants com passa als llenguatges d'alt nivell.

- Cadascuna de les instruccions realitza operacions molt simples, de manera que el programador ha de saber com combinar el nombre reduït d'instruccions de què disposa per tal de realitzar els algorismes cercats.

- Les instruccions tenen un format molt rígid pel que fa a la posició dels diferents camps (per exemple, CODI OPERACIO – PRIMER OPERAND – SEGON OPERAND), per la qual cosa la redacció de les instruccions és molt poc flexible.

- Tal com ja hem esmentat, el llenguatge màquina o llenguatge nadiu, depèn de la CPU (processador) de cada ordinador i hi està lligat, per la qual cosa els programes en codi màquina no es poden transferir d'un model d'ordinador a un altre, podent-se executar només en el processador amb el qual està vinculat el codi màquina corresponent.

- En un programa escrit en llenguatge màquina no s'hi poden incloure comentaris que ajudin al seu seguiment i comprensió, facilitant-ne la modificació posterior (manteniment).

El llenguatge màquina és l'únic que entén directament l'ordinador (el seu processador). La programació amb aquest tipus de llenguatges és molt complicada, però permet que el desenvolupador pugui utilitzar la totalitat dels recursos que ofereix l'ordinador, obtenint programes molt eficients, tant en temps d'execució com en ocupació de memòria.

Llenguatge assemblador

La majoria de les limitacions comentades dels llenguatges màquina les resol, parcialment, el llenguatge assemblador i gairebé totalment els llenguatges d'alt nivell o també anomenats *simbòlics*.

Per a evitar que els programadors hagin de programar directament en codi binari o màquina, es van desenvolupar uns programes que permetessin traduir les instruccions a codi màquina.

Aquests programes són la segona generació de llenguatges de programació i es van anomenar *assembladors*.

Podien llegir les instruccions que les persones "podien entendre" millor, redactades en llenguatge assemblador, i les convertien en una instrucció de llenguatge màquina.

Tot i així, aquest llenguatge també és de baix nivell, ja que cadascuna de les instruccions correspon a una instrucció del llenguatge màquina.

Cada processador té, per tant, el seu llenguatge assemblador, que tradueix el codi font, línia per línia, a codi de màquina i crea el fitxer executable de l'aplicació.

Codi font

Llistat de text que conté les instruccions que componen una aplicació en un determinat llenguatge de programació que una vegada compilat o assemblat amb la corresponent eina dóna lloc a un programa funcional executable sota un sistema operatiu.

Com a principals característiques es pot enumerar que:

- Aquests llenguatges utilitzen una notació simbòlica o mnemotècnica per a representar el codi d'operació, evitant els codi numèrics tan difícils d'utilitzar. Aquests codis mnemotècnics estan constituïts per abreviatures de les operacions en anglès. Per exemple, la instrucció de sumar es representa en la majoria de llenguatges assemblador amb les lletres ADD.

- En lloc d'utilitzar adreces binàries absolutes a la memòria, les dades es poden identificar per noms, la qual cosa es coneix com a *adreçament simbòlic*.

- El programador pot utilitzar comentaris entre les línies d'instruccions dels programes, convertint-se en més entenedors i simplificant el seu seguiment i posterior modificació.

- El llenguatge assemblador continua essent molt important, ja que ofereix al programador el control total de la màquina i, per tant, li permet generar un codi compacte, ràpid i eficient.

Malgrat tot, el llenguatge assemblador presenta la majoria d'inconvenients que també té el llenguatge màquina com, per exemple:

- Un repertori molt reduït d'instruccions.
- Un format molt rígid per a les instruccions.
- La baixa portabilitat entre processadors i la seva forta dependència del maquinari (*hardware*).

Per a solucionar d'alguna manera la limitació que implica disposar d'un repertori d'instruccions tan reduït, s'han desenvolupat uns assembladors especials anomenats *macroassembladors*.

Els llenguatges que tradueixen els macroensambladors tenen "macroinstruccions" la traducció de les quals dóna lloc a diverses instruccions màquina i no solament a una de sola.

Exemples tipus macroinstruccions

Per exemple, hi ha macroinstruccions per a multiplicar, dividir, transferir blocs de memòria principal a disc, etc.

Es deu a aquestes limitacions que els programador no acostumin a desenvolupar els programes en llenguatge assemblador, sinó que ho fan amb llenguatges de més alt nivell.

Els llenguatges assembladors es van definir per a evitar que els programadors haguessin de programar directament en codi binari o màquina. Cada processador té el seu propi llenguatge assemblador que tradueix el codi font, línia per línia a codi màquina, cosa que fa que també siguin uns llenguatges de baix nivell.

Llenguatges d'alt nivell

La tercera generació de llenguatges de programació tenia com a objectiu facilitar la tasca del desenvolupador en permetre crear programes independents de la màquina, utilitzant una sintaxi molt semblant a l'anglès.

Exemples de llenguatges

Exemples d'aquests llenguatges són el Cobol, el Basic, el Fortran, el C o el Pascal.

Amb aquests llenguatges els programadors poden escriure en una sola instrucció l'equivalent a diverses instruccions complicades de baix nivell.

Per a saber-ne més

Per a més informació sobre llenguatges de programació, vegeu també l'apartat "Principals llenguatges de desenvolupament" d'aquest mateix mòdul.

Tenen unes instruccions més fàcils d'entendre i proporcionen facilitats per a expressar alteracions del flux de control d'una manera força intuïtiva.

Exemple rutina de codi

Per exemple, una rutina de codi desenvolupada en un llenguatge d'alt nivell tindria l'estructura següent:

```
A= 0
FOR i=1 TO 10
A=A+(i*i)
NEXT i
```

En aquest cas, aquesta rutina calcula la suma dels primers 10 quadrats dels nombres naturals (1*1 + 2*2 + 3*3 + ... + 10*10)

Independentment del llenguatge d'alt nivell en què s'escrigui el programa, l'ha de traduir un compilador al llenguatge màquina perquè el pugui executar el processador corresponent.

Les seves característiques principals són les següents:

- Són independents de l'arquitectura física de l'ordinador, per la qual cosa es poden utilitzar els mateixos programes en ordinadors d'arquitectures diferents, sense necessitat de conèixer el maquinari específic de la màquina. Només cal "compilar[3]" el programa amb el compilador corresponent.

Per a saber-ne més

Per a més informació sobre els compiladors, vegeu l'apartat "compiladors" d'aquest mateix mòdul.

(3)Compilar

Acció de traduir codi escrit en format ASCII en llenguatge màquina (codi màquina) mitjançant un compilador.

- Com ja hem esmentat, l'execució d'un programa en un llenguatge d'alt nivell, requereix una traducció al llenguatge màquina de l'ordinador en

què s'executarà. Una sentència en un llenguatge d'alt nivell correspon, un cop traduïda, a diverses instruccions en llenguatge màquina.

- Utilitzen notacions properes a les utilitzades per les persones, amb la qual cosa es pretén una aproximació més gran al llenguatge natural o algebraic, de manera que les instruccions estan expressades en text i és possible introduir-hi comentaris en les línies de codi dels programes i la seva escriptura està, normalment, basada en regles semblants a les humanes.

- Ofereixen instruccions potents com, per exemple, funcions matemàtiques (sinus, cosinus, conversió d'enters a reals...) operadors específics d'entrada o sortida com, per exemple, "PRINT" o operadors de tractament de cadenes de caràcters com, per exemple, "extreure un conjunt de caràcters d'una línia de text" o "cercar un subtext en una cadena". També ofereixen la possibilitat de crear les teves pròpies funcions.

- A diferència dels llenguatges màquina i assemblador, aquests són menys eficients.

Totes aquestes característiques posen de manifest un apropament a les persones i un allunyament de les màquines.

És per això que tots els programes escrits en llenguatge d'alt nivell no es poden tractar directament per l'ordinador, i és necessari traduir al llenguatge màquina per a ser executats.

Aquests programes que realitzen aquestes traduccions s'anomenen traductors i han estat desenvolupats per a cada ordinador específic.

Una evolució d'aquests llenguatges van ser els emmarcats dins la quarta generació (4GL), que permeten generar de manera automàtica la major part dels procediments (rutines o conjunts d'instruccions de codi en llenguatge d'alt nivell) d'un programa. D'aquesta manera, el desenvolupador només ha d'indicar què vol fer, però no com.

Un programador que treballi amb un llenguatge de tercer nivell com, per exemple, el C, escriu instruccions del que s'ha de fer i indica com fer-ho, per mitjà dels algorismes redactats.

En canvi, amb els llenguatges 4GL (que permeten generar codi sense haver de desenvolupar, a partir d'especificacions), els desenvolupadors o usuaris (analistes funcionals, experts no tècnics), poden escriure programes de manera senzilla, per exemple, per a consultar una base de dades o per a crear sistemes d'informació personal o departamentals.

Molts d'aquests llenguatges disposen d'una interfície gràfica.

Aquests llenguatges converteixen les especificacions indicades pel desenvolupador o bé en llenguatges de tercer nivell, que el mateix programador pot refinar, o bé generen les instruccions en codi màquina directament.

Exemples de llenguatges 4GL són la majoria de les eines CASE de disseny de bases de dades, modelització de processos...

Finalment, cal indicar que la cinquena generació dels llenguatges de programació són els anomenats *llenguatges naturals*, que si bé estan en els seus inicis, faciliten una aproximació total al llenguatge humà.

Amb els llenguatges d'alt nivell els programadors poden escriure en una sola instrucció l'equivalent a diverses instruccions complicades de baix nivell. Ara bé, aquests programes han de ser, posteriorment, traduïts a un llenguatge entenedor pel processador.

2.1.2. Traductors

Tal com ja s'ha comentat anteriorment, atès que un ordinador només pot interpretar i executar el codi màquina, hi ha uns programes especials, anomenats *traductors*, que tradueixen programes escrits en un llenguatge de programació al llenguatge màquina de l'ordinador.

Un traductor és un metaprograma que té com a entrada un programa (o part d'un programa) escrit en llenguatge simbòlic, allunyat de la màquina, anomenat *programa font* (*codi font*) i proporciona com a sortida un altre programa semànticament equivalent, escrit en un llenguatge comprensible pel maquinari de l'ordinador anomenat *programa objecte* (*codi objecte*).

En aquest material es tractaran dos tipus de traductors, els compiladors i els intèrprets, que representen dues aproximacions molt diferents a la tasca de permetre el funcionament dels programes escrits en un determinat llenguatge de programació d'alt nivell.

Compiladors

Un compilador tradueix completament un programa font, generant un programa objecte (semànticament equivalent) escrit en llenguatge màquina.

Com a part important d'aquest procés de traducció, el compilador informa al desenvolupador de la presència d'errors en el programa font, passant a la creació del programa objecte (en llenguatge màquina) només en el cas que no s'hagin detectat errors (generalment, se sol cancel·lar la compilació quan es detecten errors).

El programa font acostuma a estar contingut en un fitxer i el programa objecte es pot emmagatzemar com un altre fitxer en memòria massiva (disc dur, CD, etc.) per a ser executat posteriorment sense haver de tornar a traduir-lo.

Per a saber-ne més

Podeu veure els dispositius externs de memòria massiva en l'apartat 1.4 del mòdul "Ordinadors i sistemes operatius".

Els principals avantatges dels compiladors enfront dels intèrprets és que només cal fer la traducció un cop. Una vegada traduït un programa, la seva execució es independent de la seva compilació. Així, els compiladors permeten realitzar optimitzacions de codi en llegir i traduir tot el text.

Intèrprets

Un programa intèrpret permet que un programa font escrit en un llenguatge determinat es vagi traduint i executant directament, sentència a sentència, per l'ordinador.

L'intèrpret revisa una sentència font, l'analitza i la interpreta, donant lloc a la seva execució immediata.

A diferència del procés de "compilació", en aquest cas no es crea cap fitxer o programa objecte emmagatzemable en memòria massiva per a possibles execucions futures.

Si s'utilitza un intèrpret per a traduir un programa, cada vegada que s'executa el programa es torna a analitzar (ja que no es genera un fitxer objecte).

En canvi, amb un compilador, encara que sigui més lenta, només cal fer la traducció una vegada.

A més a més, els traductors no permeten realitzar optimitzacions del codi (que eliminen ordres innecessàries compactant el codi) més enllà del context de cada sentència del programa.

El principal avantatge dels intèrprets vers els compiladors és que resulta més fàcil localitzar i corregir errors dels programes (depuració de programes), ja que l'execució d'un programa sota un intèrpret es pot interrompre en qualsevol moment per a conèixer els valors de les diferents variables i la instrucció font que s'acaba d'executar en aquell moment.

Per aquest motiu, els intèrprets resulten més pedagògics per a aprendre a programar, ja que l'alumne pot detectar i corregir més fàcilment els seus errors. Són famosos els intèrprets del llenguatge de programació Basic, un llenguatge molt utilitzat per a ensenyar a programar.

2.1.3. Principals llenguatges de desenvolupament

En aquest apartat s'enumeren breument, alguns dels principals llenguatges de programació.

L'objectiu és tenir una visió de diferents tipus de llenguatges de desenvolupament i la seva principal aplicació.

Es tracta de tenir els principals conceptes de la seva utilitat i principal funcionalitat, però no de saber-ne la sintaxi i la manera de treballar-hi.

Atès que els llenguatges d'alt nivell són els més utilitzats, tal com s'ha comentat en els apartats anteriors, la classificació i presentació se centrarà en aquests llenguatges.

Es poden classificar els llenguatges d'alt nivell en dues divisions, els llenguatges de propòsit general, que es poden emprar en qualsevol tipus d'aplicacions (com, per exemple, el llenguatge C o el Pascal) i llenguatges de propòsit especial.

Una primera classificació dels llenguatges de desenvolupament pot ser entre els llenguatge de propòsit general i els de propòsit especial.

Tot i així, en ser tan general aquesta classificació també es podrien intentar subclassificar des del punt de vista del camp d'aplicació a què pertany el llenguatge:

- **Aplicacions científiques**: hi predominen les operacions numèriques o matricials pròpies d'algorismes matemàtics. Un llenguatge adequat per a aquest tipus d'aplicacions és el Fortran, per exemple.

  Fortran

  El Fortran es va dissenyar per a utilitzar-lo en aplicacions científiques i tècniques (matemàtiques, ciència i enginyeria). Es caracteritza per la potència en els seus càlculs matemàtics, però està limitat en tot el que fa referència al tractament de dades no numèriques, per la qual cosa no resulta adequat per a aplicacions de gestió, gestió de fitxers i el tractament de caràcters i edició d'informes.

  Tot i tenir els seus orígens en el 1953, de la mà de John Backups, un empleat d'IBM, segueix essent un llenguatge comú en aplicacions d'investigació, enginyeria i educació.

  Fortran significa ***for**mula **tran**slator* ('traductor de fórmules') i es considera el primer llenguatge d'alt nivell; la primera versió va aparèixer el 1957.

- **Aplicacions de processament de dades**: en aquestes aplicacions son freqüents les operacions de creació, manteniment i consulta sobre fitxers i bases de dades. Dins d'aquest camp, hi hauria aplicacions de gestió empresarial, com ara programes de nòmines, comptabilitat, facturació, con-

Per a saber-ne més

Per a més informació sobre llenguatge de base de dades SQL podeu consultar el mòdul 2, "Bases de dades".

trol d'inventari, etc. Alguns dels llenguatges aptes per a aquesta mena d'aplicacions són el Cobol i el llenguatge de Bases de Dades SQL.

Cobol

El Cobol és el llenguatge més utilitzat en aplicacions de gestió. El va crear el 1960 un comitè patrocinat pel departament de defensa dels Estats Units amb la finalitat de disposar d'un llenguatge universal per a aplicacions comercials i avui dia hi ha milions de línies de codi escrites en aquest llenguatge.

El nom Cobol prové de la frase *common business oriented language* ('llenguatge general per als negocis') i al llarg de la seva existència ha sofert diverses actualitzacions.

Les característiques més interessants d'aquest llenguatge és que s'assembla al llenguatge natural (es fa un gran ús de l'anglès senzill), és autodocumentat i ofereix grans facilitats en la gestió de fitxers i també en l'edició d'informes escrits.

En canvi, les seves rígides regles de format d'escriptura, la necessitat d'escriure tots els elements al màxim detall, l'extensió excessiva de les seves sentències (de vegades resulta massa enrevessat i reiteratiu) i la inexistència de funcions matemàtiques, són els seus inconvenients principals.

- **Aplicacions de tractament de textos**: aquestes aplicacions estan associades al maneig de textos en llenguatge natural. El llenguatge C seria adequat per a aquesta mena d'aplicacions.

Llenguatge C

El llenguatge C el va crear el 1972 Dennis Ritchie (qui, juntament amb Ken Thompson, havia dissenyat anteriorment el sistema operatiu Unix), amb la intenció d'aconseguir un llenguatge idoni per a la programació de sistemes que fossin independents de la màquina per a utilitzar-lo en la implementació del sistema operatiu Unix.

Des d'aleshores, tant Unix com C han tingut un enorme desenvolupament i proliferació, fins a convertir-se en un estàndard industrial per al desenvolupament del programari.

El C és un llenguatge modern de propòsit general que combina les característiques d'un llenguatge d'alt nivell (programació estructurada, tipus i estructures de dades, recursivitat, etc.) amb una sèrie de característiques més pròpies de llenguatges de baix nivell.

Aquesta qualitat del C fa possible que el programador utilitzi la programació estructurada per a resoldre tasques de baix nivell, obtenint un codi executable veloç i eficient.

Per això, molta gent el considera com un llenguatge de nivell mitjà.

- **Aplicacions en intel·ligència artificial**: destaquen les aplicacions en sistemes experts, jocs, visió artificial i robòtica i els llenguatges més populars dins d'aquest àmbit són el LISP i el Prolog.

LISP

El LISP va ser dissenyat el 1959 per John McCarthy al MIT (Institut Tecnològic de Massachusetts) per a utilitzar-lo en l'àmbit de la intel·ligència artificial. Aquest llenguatge pren el seu nom de en anglès ***list p****rocessing* ('processament de llistes').

LISP està pensat per a resoldre problemes de manipulació de símbols (que són de gran interès en intel·ligència artificial). Els símbols són elements bàsics d'aquest llenguatge i representen objectes arbitraris del domini d'interès que s'estigui tractant.

És un llenguatge funcional, ja que tot programa (o subprograma) en LISP es pot veure com una funció d'alt nivell que s'aplica sobre altres funcions de més baix nivell per a obtenir determinats resultats. Per a realitzar operacions elementals es poden utilitzar funcions d'una biblioteca.

Aquest llenguatge no s'assembla gens a altres llenguatges de programació. Tot i així, és un llenguatge fàcil d'entendre i és el més comú dins les aplicacions en intel·ligència artificial.

Un dels seus principals problemes era que no es podia executar de manera eficient en molts ordinadors. En l'actualitat, hi ha versions estàndard de LISP, Common LISP, DG Common LISP i LISP portable estàndard.

Prolog

El Prolog (***pro****gramming* ***log****ic*) es va desenvolupar a partir del treball realitzat en la dècada dels anys setanta, principalment en universitats europees, i ha estat el llenguatge més utilitzat en l'àmbit de la intel·ligència artificial a Europa.

Es basa en la lògica, i ha estat utilitzat per a desenvolupar un gran nombre d'aplicacions en bases de dades i intel·ligència artificial.

El Prolog permet que el programador expressi una sèrie de tasques basades en la descripció dels objectes que hi intervenen (fets i regles) i les relacions lògiques que hi ha entre ells (predicats), en lloc de fer-ho mitjançant un algorisme.

Porta incorporada la programació d'operacions, i tot l'esforç de programació consisteix a especificar adequadament els fets i les regles per a després establir preguntes que es podran inferir de manera automàtica.

El Prolog permet desenvolupar sistemes experts a persones sense gaire idea de programació, ja que no requereix programar cap algorisme.

Atesa aquesta facilitat d'ús, una important aplicació del Prolog és l'educació, on es pot utilitzar per a ensenyar lògica, tècniques de resolució de problemes i se sol utilitzar en un gran nombre de bases de dades educatives.

- **Aplicacions de programació de sistemes**: en aquest camp s'inclourien la programació de programari d'interfície entre l'usuari i el maquinari, com són els mòduls d'un sistema operatiu i els traductors. Tradicionalment, tal com s'ha esmentat anteriorment, per a aquestes aplicacions s'utilitzava el llenguatge assemblador. En l'actualitat es mostren molt adequats els llenguatges com l'Ada, el Modula-2 i el C, per exemple.

Ada

El llenguatge Ada va néixer amb l'objectiu d'obtenir un únic llenguatge per a tota mena d'aplicacions (un autèntic llenguatge de propòsit general).

El va encarregar el Departament de Defensa dels Estats Units i la seva estandardització fou publicada el 1983. El nom d'Ada prové d'Augusta Ada Byron, comtessa de Lovelace, considerada la primera programadora de la història.

Entre les característiques del llenguatge s'inclou la compilació separada, la programació concurrent, la programació estructurada, la bona mantenibilitat, característiques de temps real, etc.

El principal inconvenient, però, d'aquest llenguatge és la seva gran extensió, que en pot complicar l'ús.

Modula-2

El mateix Nicklaus Wirth, creador del llenguatge Pascal, va dirigir, al final dels anys setanta, el desenvolupament del llenguatge Modula-2 (anomenat en un principi simplement Modula), amb la intenció d'incloure les necessitats de la programació de sistemes i donar resposta a les crítiques rebudes respecte a les carències del llenguatge Pascal.

A més de les principals característiques del Pascal, aquest llenguatge incorpora funcionalitats com la possibilitat de compilació separada, la creació de biblioteques, la programació concurrent, la millor gestió de cadenes de caràcters, els procediments d'entrada/sortida i de gestió de la memòria, i, a més, aporta grans facilitats per a la programació de sistemes.

Aquest llenguatge també té qualitats didàctiques, per la qual cosa ha esta àmpliament acceptat dins la comunitat universitària com a eina idònia per a ensenyar la programació.

Una altra classificació segons el tipus d'aplicació del llenguatge permetria classificar-los en aplicacions científiques, de processament de dades, de tractament de textos, d'intel·ligència artificial i de programació de sistemes.

Una altra manera de classificar els llenguatges d'alt nivell pot ser segons l'estil de programació que fomenten, és a dir, la filosofia de construcció de programes:

- Llenguatges imperatius o procedimentals: estableixen com s'ha d'executar una tasca, dividint-la en parts que especifiquen com realitzar cadascuna de les subtasques associades. Aquests llenguatges es fonamenten en l'ús de variables per a emmagatzemar valors i l'ús d'instruccions que indiquen les operacions que ha de realitzar sobre les dades emmagatzemades. La majoria dels llenguatges d'alt nivell són d'aquest tipus: Fortran, Basic, Pascal, Ada, Modula-2, Algol, RPG, PL/1, Simula, Smalltalk i Eiffel.

  Simula

  És l'antecessor dels llenguatges orientats a objectes. Construït al voltant d'idees ja presents a l'Algol, el 1962, el Simula inclou, a més a més, els conceptes d'encapsulació i herència, i va ser pioner a l'hora d'utilitzar els conceptes de classe, objectes, missatges i tipus abstractes de dades, tan corrents, temps després, en tots els llenguatges orientats a objectes.

  Un dels aspectes més importants introduïts amb Simula, concebut com un llenguatge que descriu sistemes i elabora simulacions d'aquests sistemes per ordinador, és que va introduir la idea de construir simulacions de sistemes complexos mirant de reflectir el vocabulari del sistema real i del domini del problema.

- Llenguatges declaratius: en aquest cas, el procés pel qual s'executa el programa no apareix de manera explícita en el programa. El programador no necessita indicar el procés detallat de com realitzar la tasca. En aquests llenguatges els programes es construeixen mitjançant descripcions de funcions (llenguatges funcionals, com ara LISP) o expressions lògiques que indiquen les relacions entre determinades estructures de dades (llenguatges de programació lògica, com Prolog).

- Llenguatges orientats a objectes: el disseny dels programes se centra més en les dades i la seva estructura. Els programes consisteixen en descripcions d'unitats anomenades objectes que encapsulen les dades i les operacions que hi actuen sobre. El llenguatge més utilitzat dins d'aquest tipus és el C++. Altres llenguatges poden ser Java i Javascript.

- Llenguatges orientats al problema: aquest tipus de llenguatges estan dissenyats per a problemes específics, principalment de gestió. En aquests llenguatges, els programes estan formats per sentències que ordenen què es vol fer. Generalment, aquests llenguatges solen ser generadors d'aplicacions

que permeten automatitzar tant com poden la tasca de desenvolupament del programari d'aplicació de gestió.

Llenguatge C++

El llenguatge C++ és el successor del llenguatge C i fou desenvolupat per Bjarne Stroustup als laboratoris Bell a principis de la dècada dels vuitanta.

C++ introdueix la programació orientada a objectes. Igual que C, C++ és un llenguatge molt poderós i eficient, però encara és més difícil d'aprendre que el C, tot i així és un dels llenguatges actualment més utilitzats, ja que combina les característiques esmentades del llenguatge C amb la programació orientada a objectes.

També ha evolucionat, essent molt coneguda la versió Visual C++.

Java

És un llenguatge de programació orientat a objectes creat per Sun Microsystems a principi dels anys noranta. És similar al llenguatge C++, però no té elements de baix nivell, per això és conceptualment menys abstracte.

És un llenguatge interpretat. El llenguatge d'alt nivell és compilat a *bytecodes*, aquests *bytecodes* són transversals a qualsevol arquitectura i per tant el programa serà exportable a totes les arquitectures de microprocessadors. Perquè funcionin només cal que la plataforma tingui una màquina virtual Java, és a dir, l'intèrpret dels *bytecodes* adequat a cada plataforma.

Javascript

És un llenguatge orientat a objectes àmpliament utilitzat per a dotar de dinamisme les pàgines web escrites en codi HTML (*hipertext marked language*). De sintaxi molt similar a Java però de molta menys complexitat.

Va ser inventar per Brendan Ecih mentre formava part de Netscape Comunications. Actualment tots els navegadors web suporten inevitablement aquest llenguatge, ja que un percentatge elevadíssim de les pàgines web l'usen.

Finalment, si es té en compte la manera de programar, utilitzant aquests llenguatges es poden categoritzar entre els procedimentals, els declaratius, els orientats a objectes i els orientats al problema.

RPG

L'RPG (*report program generator*) és en realitat un generador de programes que llegeix unes especificacions i adapta els mòduls de programa per a la realització de les entrades i sortides indicades en les instruccions.

Va aparèixer amb els petits ordinadors per a gestió com el Sistema/3 d'IBM al final dels anys seixanta. Modificacions posteriors (les targetes d'instruccions tipus C i les versions RPG III i RPG 400) el configuren amb la possibilitat de desenvolupar algorismes específics, mitjançant l'ús de variables binàries conegudes com a *interruptors*, i també de gestionar bases de dades.

Algol

L'Algol (*algorithmic language* o *>algebraic oriented language*) és un llenguatge algorísmic o orientat a l'àlgebra, important per haver introduït per primera vegada conceptes que s'han revelat bàsics en la programació, principalment la idea d'estructuració. Es va arribar a utilitzar com un llenguatge internacionalment adoptat per a la descripció d'algorismes matemàtics. Es va desenvolupar a la dècada dels seixanta.

PL/1

El PL/1 (*program language* 1) és un llenguatge dissenyat perquè s'utilitzi tant en els problemes de tipus científic com en els de gestió. El seu disseny es basa molt en altres llenguatges com ara el Fortran, el Cobol i l'Algol.

De l'Algol adopta l'estructura de blocs i les instruccions de control de seqüències estructurades; del Cobol, la riquesa en la gestió de fitxers i la declaració de dades de tipus PICTURE, i del Fortran, la forma concisa i simple de les instruccions i el mecanisme de transmissió de paràmetres, entre altres elements.

Basic

El llenguatge Basic es va dissenyar el 1965 amb l'objectiu que l'utilitzessin els principiants en la seva introducció en el món de la programació.

És un llenguatge fàcil d'aprendre, com indica el seu nom: ***b****eginner's* ***a****ll-purpose* ***s****ymbolic* ***i****nstruction* ***c****ode* ('codi simbòlic de propòsit general per a principiants'), i al començament es va enfocar a ensenyar estudiants que es pretenien introduir en el món de la programació, i va esdevenir el llenguatge educatiu més popular del món.

Les principals aportacions de Basic són que és un llenguatge interpretat (traduït per un programa intèrpret) i que és d'ús interactiu.

Amb els anys, aquest llenguatge ha evolucionat cap a versions preparades per a desenvolupar aplicacions per Internet, de gestió i suportar característiques i mètodes orientats a objectes, com ara el Visual Basic.

Pascal

El llenguatge Pascal el va desenvolupar el 1970 el matemàtic suís Nicklaus Wirth, amb l'objectiu de proporcionar un llenguatge adequat per a l'ensenyament dels conceptes i tècniques de programació, i permetre el desenvolupament d'aplicacions fiables i eficients en els ordinadors disponibles en aquell moment.

El llenguatge rep el seu nom en honor del filòsof i matemàtic francès Blaise Pascal, que va inventar la primera màquina de tipus mecànic per a sumar i, amb el temps, ha arribat a ser un llenguatge molt utilitzat en tota mena d'aplicacions, i s'utilitza molt en les universitats per a aplicacions científiques i d'enginyeria.

Pascal s'ha dissenyat per a il·lustrar conceptes clau en programació, com els de tipus de dades, programació estructurada (és un llenguatge molt estructurat) i disseny descendent. El Pascal tracta de proporcionar un mecanisme per a implementar cadascun dels conceptes de programació.

Aquest llenguatge s'ha convertit en el predecessor d'altres llenguatges més moderns, com el Modula-2 i l'Ada.

Smalltalk

Es pot considerar el primer llenguatge orientat a objectes realment "purs", en el sentit que és el primer que es planteja com un llenguatge de nou encunyament per a adoptar íntegrament la nova orientació a objectes. A l'Smalltalk tot es tracta com un objecte, des dels nombres enters fins a les classes.

Concebut com un intèrpret, es va avançar al seu temps en l'ús d'una interfície gràfica i l'ús interactiu. La seva influència ha resultat decisiva en la resta de llenguatges orientats a objectes.

Eiffel

Eiffel es va publicar el 1986 i es va presentar com un llenguatge orientat a objectes que es tradueix primer a un altre llenguatge de programació, la qual cosa, com en el cas de C++, afavoreix la portabilitat.

El llenguatge Eiffel és considerat, en el món acadèmic, com el llenguatge orientat a objectes més ben dissenyat i complet, tot i que no sempre resulti el més eficient.

En realitat, és un entorn de desenvolupament amb eines auxiliars, com sistemes gràfics d'exploració d'estructures, depuradors, suport de persistència, gestió de configuracions, etc.

Llenguatges propietaris

Algunes grans aplicacions es poden personalitzar segons l'empresa que les compri per a adaptar-les al seu negoci i a la seva manera particular d'operar. Per a poder adaptar l'aplicació al negoci i després personalitzar-la segons els seus fluxos operatius particulars cal algun llenguatge que faci aquest pas. Aquests llenguatges no serveixen per a crear aplicacions des de zero, serveixen per a personalitzar i adaptar aplicacions de caràcter genèric com SAP (ERP) o Siebel (CRM).

ERP

Enterprise resource planning (planificació dels recursos de l'empresa).

CRM

Customer relationship management (gestió de la relació amb el client).

2.2. Eines CASE

El terme CASE (*computer aided software engineering*) es va popularitzar en la dècada dels vuitanta, designant una tecnologia que no era nova: les ajudes automatitzades de les metodologies de desenvolupament de programari, i representava l'evolució de les primeres ajudes (editors, compiladors, etc.) que van formar els entorns de desenvolupament de programari dels anys setanta.

2.2.1. Principals conceptes i història

CASE, l'enginyeria de programari assistida per ordinador, inclou el conjunt d'eines i mètodes associats que poden servir d'ajuda en el procés de construcció del programari al llarg del seu cicle de vida.

La novetat del CASE es va fer evident quan es van incorporar a les eines d'ajuda al disseny informàtic noves funcionalitats gràfiques amb les quals es va poder representar, modificar i mantenir actualitzats els diagrames gràfics que han esdevingut clàssics i s'utilitzen en la majoria de les metodologies de desenvolupament de programari.

Una eina CASE permet ajudar el desenvolupador i l'equip de projectes a construir el programa donant-los eines perquè puguin generar documentació, codi, diagrames...

Les veritables eines CASE van sorgir quan els diagrames es van incorporar a una base de dades global de disseny (diccionari de dades o repositori), que també mantenia detalls dels elements de dades i de la lògica dels processos.

Aquest diccionari de dades és, en realitat, una base de dades que incorpora un model en què s'inclouen restriccions d'integritat de les dades.

Exemples d'eines CASE són els sistemes Excelerator, System Architect, EasyCASE, IEF (*information enineering facility*), IEW (*information engineering workbench*), Teamwork, VAW (*visible analyst workbench*), i també tots els apareguts més endavant aprofitant al màxim totes les prestacions de la interfície gràfica.

Dues eines o entorns molt coneguts avui dia d'eines CASE són el programa Microsoft Visio i el paquet Rational Rose, que incorpora aplicacions per a generar documentació, codi, provar els programes...

Si bé hi ha diferents subdivisions, una classificació de les eines CASE es basa en CASE alt (*upper CASE*), mitjà (*middle CASE*) i baix (*lower CASE*) d'acord amb el nivell d'abstracció i generalitat.

Una altra distinció possible s'estableix entre eines (*toolkits*), suposadament aplicables a una única tasca del desenvolupament del programari, i tallers de treball (*workbench*) que es presenten com una col·lecció d'eines integrades que ajuden tot el procés de desenvolupament de programari: anàlisi, disseny i implementació.

Hi ha una gran infinitat d'eines CASE al mercat i es classifiquen per CASE alt, mitjà i baix o també per eines i tallers de treball.

2.2.2. Aplicació del CASE i Eines

La majoria d'eines CASE s'apliquen a les fases d'anàlisi i disseny, però alguns sistemes més complets incorporen generadors de llenguatges i inclouen la possibilitat de generar el codi font de l'aplicació d'una manera automàtica.

D'aquesta manera ajuden tant al procés de construcció del programari com també a l'elaboració i manteniment de la documentació, simplificant bona part de la feina de programar.

De vegades, les eines CASE són simplement un generador que utilitza les especificacions procedents d'una altra eina CASE, com pot passar amb la família de generadors de l'APS Development Center, que pot generar el Cobol a partir, per exemple, de les especificacions que li proporciona Excelerator.

Pel que fa a l'anàlisi i disseny de sistemes, predominen les eines que implementen els diagrames de flux de dades (DFD, *data flow diagrams*) de l'anàlisi i disseny estructurat de Constantine, Yourdon i De Marco i també en la variant de Gane/Searson.

Quant al disseny de dades, s'utilitza sovint el diagrama d'entitat-relació de Chen (ERS, *entity- relationship diagrams*) o la versió més clàssica dels diagrames de Bachman per a convertir-los finalment en el seu equivalent en el model de bases de dades relacionals.

També, per a noves metodologies com MERISE, s'implementen els diagrames de la història de vida d'una entitat (ELH, *entity-life-history-diagrams*).

Unes altres tècniques de disseny molt populars i utilitzades per les eines CASE són els diagrames de descomposició i jerarquia de funcions (HD, *hierarchical diagram*); els diagrames d'estructura de mòduls de la programació modular i el disseny estructurat (SC, *structured chart*); els diagrames d'estructura de dades de la metodologia Jackson (DSD, *data structure diagrams*) o els grafs de disseny de diàlegs (IDS, *invocation dialog structure*) i els dissenys de pantalles i llistats entre altres modelitzacions gràfiques molt utilitzades en l'enginyeria del programari.

Objectiu de l'apartat

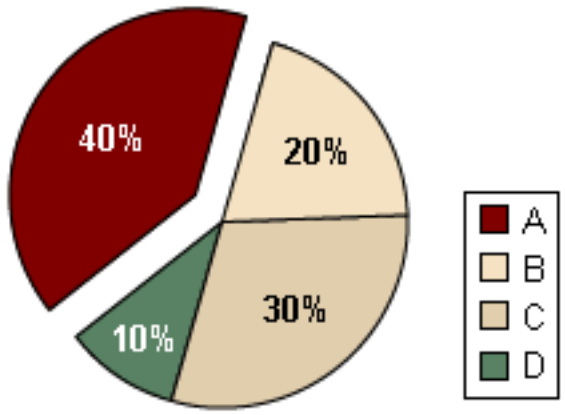
L'objectiu d'aquest apartat no és aprofundir en tècniques i eines de disseny i desenvolupament d'aplicacions. Tot i així, s'enumeren totes aquestes tècniques perquè siguin familiars a l'estudiant.

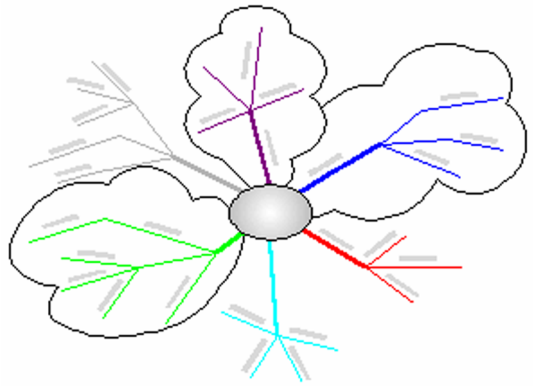
Les eines CASE poden tenir diferents aplicacions. Des de l'anàlisi i disseny fins a la generació del programa i la seva documentació.

Alguns dels models que permeten generar aquestes eines són els següents:

- Diagrames genèrics. Utilitats per a crear dissenys i esquemes per a planificació, comunicació, generació de documentació de suport.

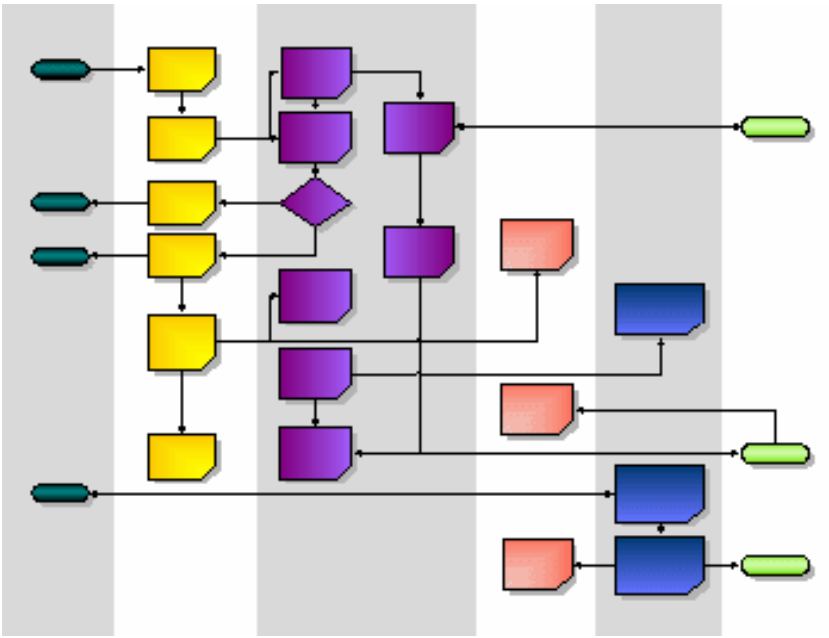
Diagrames genèrics





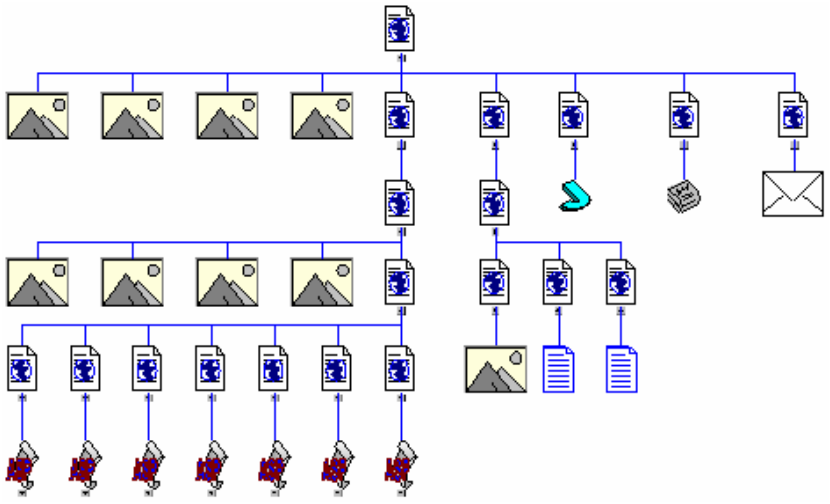
- Diagrames d'auditoria. Emprats per a documentar i analitzar processos.

Diagrames d'auditoria



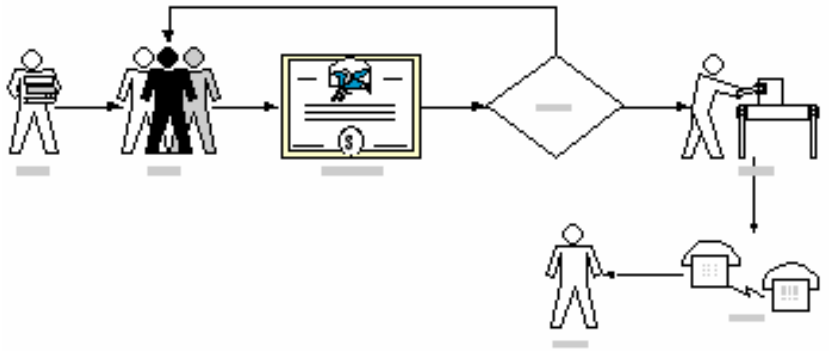
- Diagrames "conceptuals de llocs web". Utilitzats en el disseny i desenvolupament de webs.

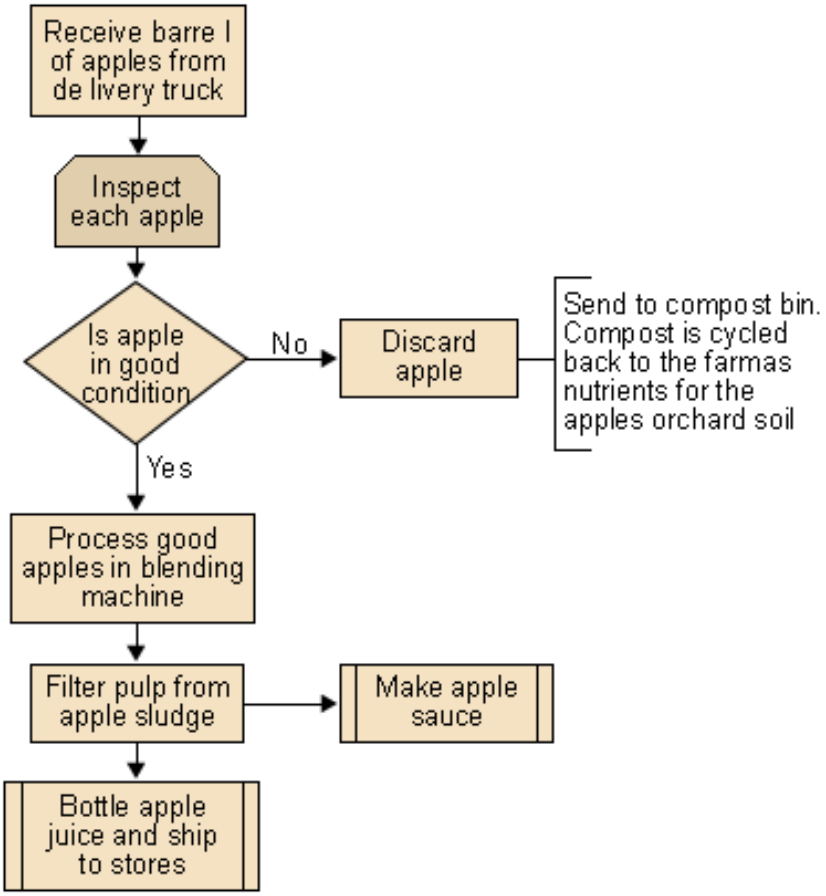
Diagrames "conceptuals de llocs web"



- Diagrames de processos i fluxos de dades. Permeten dissenyar i reflectir els processos de negoci i els fluxos de dades i documents dels diferents departaments d'una companyia i la seva interacció.

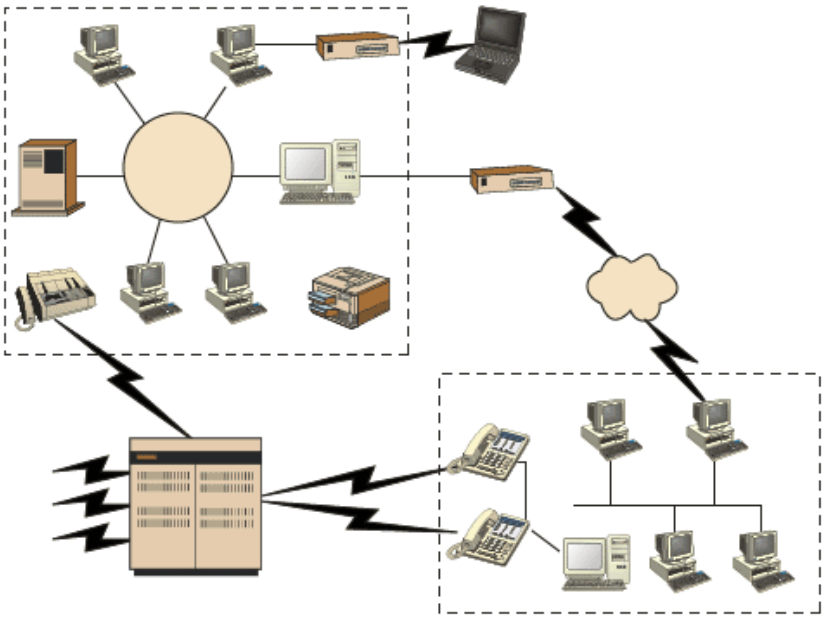
Diagrames de processos i fluxos de dades





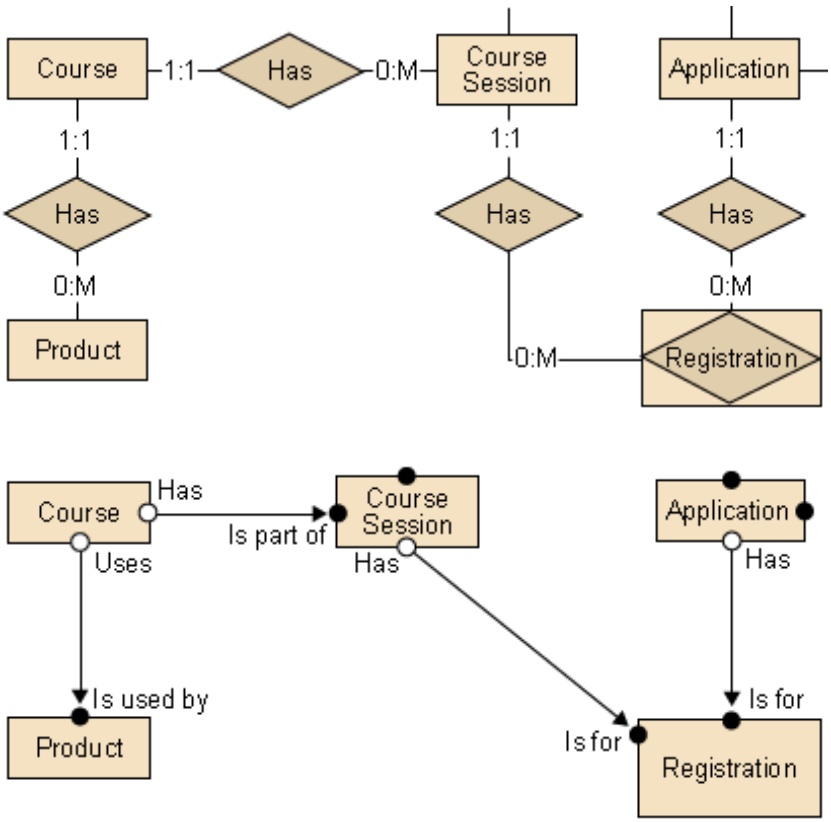
- Diagrames de xarxes. Permeten crear i dissenyar el diagrama tant lògic com físic d'una xarxa d'ordinadors i dispositius.

Diagrames de xarxes



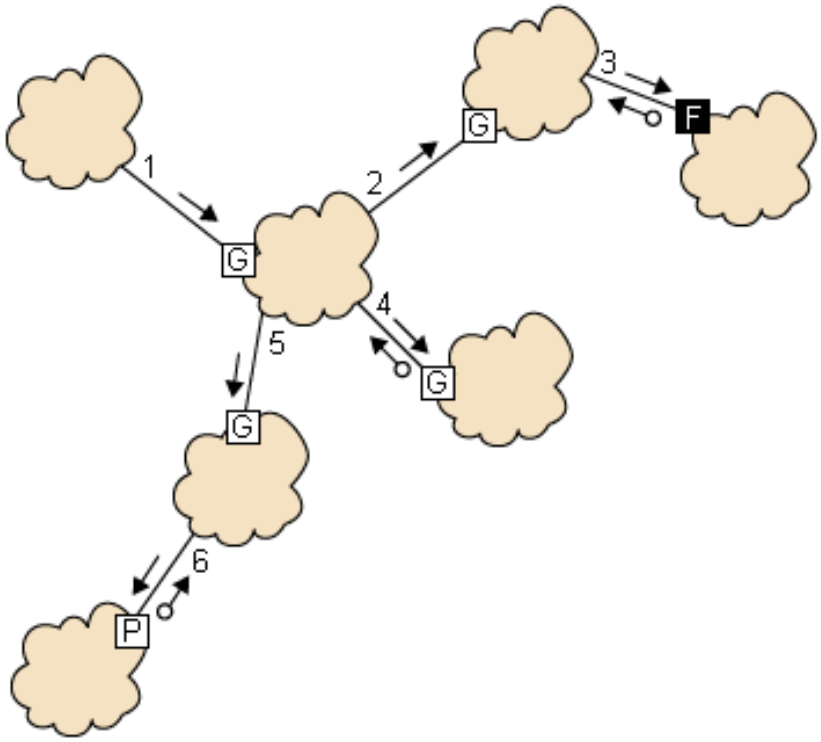
- Diagrames de models conceptuals de BD.

Diagrames de models conceptuals de BD



- Diagrames de models "orientats a objectes". A partir de diferents notacions (Booch OOD, els casos d'ús de Jacobson, la notació ERD Martin, el llenguatge de modelització ROOM o la metodologia OMT de Rumbaugh) aquest tipus de diagrames permeten dissenyar aplicacions orientades a objectes. La majoria de les eines CASE també tenen funcions per a generar codi de programació a partir d'aquests dissenys.

Diagrames de models "orientats a objectes"



2.3. Introducció a l'enginyeria del programari

L'objectiu d'aquest material no és cobrir i tractar l'enginyeria del programari, però sí que és donar una breu introducció i visió de conjunt dels seus principals conceptes.

L'enginyeria del programari va néixer de la necessitat d'estandarditzar el procés de producció del programari i minimitzar el risc d'errors en el seu desenvolupament i manteniment.

A continuació es descriuen les principals fases de la producció del programari, la importància del manteniment dels programes i els principals elements de l'enginyeria del programari.

2.3.1. La producció de programari

Tal com s'ha esmentat en els apartats anteriors, l'objectiu i l'ús dels llenguatges de programació és la construcció de programari, és a dir, aplicacions que s'executaran en un sistema informàtic (maquinari, xarxa...) per tal de dur a terme un conjunt de funcionalitats i operacions.

El procés de producció del programari té una estructura general en què es podrien trobar tres fases genèriques: la definició, el desenvolupament i el manteniment.

Les tasques que es realitzen durant cadascuna d'aquestes fases de producció són les següents:

- **Definició**: aquesta fase es focalitza en el què, és a dir, "quina cosa" cal desenvolupar, i es defineix quina informació s'ha de processar, quina funció i rendiment es desitja del futur programa, quines restriccions s'han d'imposar i quins criteris de validació són necessaris perquè el futur sistema desenvolupat es pugui considerar correcte.

  Presa de demandes

  Dins de la definició hi ha la presa de demandes al sol·licitant del programari. Sovint qui demana el nou programari sap perfectament el problema que té i el vol resoldre amb el nou programari, però no té clara la solució que resol el seu problema. Per a evitar que la fase de definició es converteixi en una carta als reis impossible de complir, cal que l'enginyer de programari conegui molt bé la tecnologia amb la qual vol implementar la solució i que parli amb tots els implicats en el problema, des dels usuaris fins als nivells jeràrquics més elevats. Un cop s'han recollit totes les dades i desitjos de tots els implicats, cal presentar una proposta que faci contentes totes les parts, solucioni el problema i sigui implementable tècnicament en el temps pactat i amb el pressupost que es tingui. Prometre solucions poc realistes farà fracassar el projecte.

- **Desenvolupament**: la segona fase se centra en el com. Així, durant aquesta fase cal decidir com transformar allò definit anteriorment en un producte programari, com s'han de dissenyar les estructures de dades i l'arquitectura del programari, com s'implementen els diferents detalls procedimental, com s'ha de traslladar el disseny a un llenguatge de programació i com s'ha de realitzar la prova de validació un cop generat el programa.

  La tecnologia

  Encertar la tecnologia amb la qual es vol desenvolupar el programari pot ajudar molt a l'èxit. Un bon desenvolupament sempre és més àgil si darrere hi ha una bona anàlisi i les eines de desenvolupament adequades, tant de programari com de maquinari. Tenir els perfils adequats és també molt important per a ser àgil en aquesta fase.

- **Manteniment**: finalment, i un cop que el programa s'ha desenvolupat, s'entra en la fase de realitzar i dur a terme canvis associats amb la correcció d'errors, l'adaptació a nous entorns i l'augment de la funcionalitat.

  Tipus de manteniment

  Hi ha dos tipus bàsics de manteniment, el manteniment correctiu i el manteniment evolutiu. El manteniment correctiu té una gran activitat durant els primers mesos d'existència del programari, però a poc a poc tendeix a anar desapareixent a mesura que es van solucionant tots els errors. El manteniment evolutiu, en canvi, té una gran activitat durant tot el cicle de vida del programari, ja que consisteix a corregir, però també a fer evolucionar i millorar, el programari a mesura que sorgeixen noves necessitats o ideas. Sovint alguns canvis del manteniment evolutiu necessiten ser tractats com un petit projecte de programari, amb la seva presa de demandes, anàlisi, pressupost, etc.

Les tres principals fases del desenvolupament del programari són la definició, el desenvolupament i el manteniment posterior.

2.3.2. El manteniment del programari

El programari no es deteriora, com sí que li passa al maquinari.

El maquinari presenta relativament moltes errades al principi de la seva vida (construcció), sovint atribuïbles a defectes de disseny o de fabricació o d'especificacions.

Quan es corregeixen els defectes, cauen les errades fins al nivell més baix, on es queden estacionaris durant un determinat període de temps, fins que arriba un moment de la seva vida en què comença a exhibir novament errades com a conseqüència del pas del temps i d'altres mals externs que deterioren els components.

Tot i així, la realitat és que el programari queda obsolet abans que es deteriori.

Una de les raons d'aquesta obsolescència es deu a la rapidesa amb què evoluciona el programari, que fa que cada vegada s'exigeixin més recursos de maquinari. Una altra raó és l'evolució de l'entorn empresarial, científic, docent, etc., que fa que els fluxos de treball vagin variant i que el programari ja no respongui a totes les noves necessitats sorgides.

Sovint quan se solucionen les errades inicials s'inclouen modificacions que fan que apareguin o surtin a la llum noves errades. Per a evitar-ho cal emfatitzar la rigorositat de les anàlisis i les fases de test.

En general el manteniment del programari té una complexitat considerablement més elevada que el manteniment del maquinari.

Una errada en el programari és deguda a incoherències en les especificacions, un error en el disseny o un desenvolupament mal resolt. No hi ha peces de recanvi com en el cas del maquinari; això comporta que tota solució s'hagi de resoldre amb més desenvolupament, també sotmès a la possibilitat de contenir errades. En el programari no hi ha peces de recanvi, però un bon disseny pot ajudar a simular-les. Totes les tasques del programari es poden dissenyar i implementar com a mòduls individuals que interactuen, de manera que si una part queda obsoleta per la tecnologia es pot refer només el mòdul implicat. Tenir el codi modulat també en facilita el manteniment i un nou ús en diferents projectes.

El programari no es deteriora com el maquinari. Tot i així, un error en el disseny i desenvolupament d'un programa obliga a la seva reprogramació, ja que no hi ha peces de recanvi.

2.3.3. Principals elements de l'enginyeria del programari

Atesa la importància de garantir el menor nombre d'errors en la producció del programari, i alhora, la necessitat d'estandarditzar el desenvolupament de les aplicacions, va aparèixer l'enginyeria del programari.

L'enginyeria del programari engloba un conjunt de tres elements clau (mètodes, eines i procediments) que faciliten que el gestor controli el procés de desenvolupament del programari, i també subministrar les bases per a construir programari d'alta qualitat d'una manera productiva.

- Els mètodes de l'enginyeria del programari indiquen com construir tècnicament el programari i inclouen un ampli espectre de tasques des de la planificació i estimació de projectes fins a l'anàlisi dels requeriments, el disseny, la codificació, la prova i el manteniment.

- Les eines de l'enginyeria del programari subministren un suport automàtic o semiautomàtic per als mètodes. Avui dia, hi ha eines per a suportar cadascun dels mètodes mencionats anteriorment. L'enginyeria del programari assistida per ordinador, coneguda normalment com a CASE (*computer aided software engineering*) (vegeu "Eines CASE"), és una eina que permet l'automatització del procés de producció de desenvolupament del programari. La idea bàsica d'una eina CASE és proporcionar un conjunt d'eines integrades que estalviïn treball, enllaçant i automatitzant totes les fases del cicle de vida del programari.

- Els procediments de l'enginyeria del programari són el nexe d'unió entre els mètodes i les eines i faciliten un desenvolupament racional i oportú del programari. Els procediments defineixen la seqüència en què s'apliquen els mètodes, els lliuraments (documents, informes, codis, etc.) que es requereixen, els controls que ajuden a assegurar la qualitat i coordinar els canvis, i les guies que faciliten als gestors dels programari establir-ne el desenvolupament.

L'enginyeria del programari inclou i defineix els mètodes, les eines i els procediments per al desenvolupament del programari.

3. Gestió i planificació de projectes informàtics

3.1. La gestió de projectes

Si es volgués definir què és un projecte, es trobarien diversos conceptes i definicions.

Segons l'Associació Francesa de Normalització, un projecte és "un procés específic que permet estructurar metòdicament i progressivament una realitat futura".

Una altra definició seria considerar un projecte com "un conjunt de recursos humans i materials necessaris per a aconseguir un objectiu delimitat per uns costos, uns terminis i uns resultats".

I finalment, atenent-nos al PMI, Project Management Institute (www.pmi.org), un projecte es defineix com "un conjunt d'esforços, temporals, dedicats a crear un servei o producte únic".

En tot cas, independentment de la definició, es pot concloure que un projecte té les característiques següents:

- Una delimitació temporal.
- Uns objectius precisos de terminis, costos i resultats a obtenir.
- No és una acció repetitiva sinó que és una acció creativa.
- Està compost per un conjunt de tasques complexes que requereixen un treball d'anàlisi, estudi i planificació.

3.1.1. Gestió d'un projecte

La gestió d'un projecte és la planificació, organització i direcció de les tasques o activitats i dels recursos necessaris per a aconseguir un objectiu definit, limitat temporalment i amb un pressupost econòmic fixat.

Així doncs, la finalitat de la gestió de projectes és aconseguir un objectiu específic en una data màxima i amb un pressupost determinat.

Exemples de projectes poden ser:

- La construcció d'un local, supermercat, edifici...
- El llançament d'un nou producte: pla de màrqueting, pla de vendes
- El desenvolupament d'un producte nou: pla d'R+D, pla de desenvolupament

- L'organització d'esdeveniments esportius o culturals
- El desenvolupament d'una aplicació informàtica
- El desenvolupament del procés de fabricació d'un vehicle nou
- La gestió d'una campanya electoral
- La implantació de nous processos organitzatius en una empresa
- O la instal·lació d'un sistema informàtic

Un projecte o objectiu pot ser tan "simple" com dissenyar el prospecte de publicitat d'un producte nou o tan "complex" com la construcció d'un centre comercial.

En cada cas, cal:

- dividir el projecte en unes tasques o activitats,
- programar-les en dates,
- assignar-los els recursos necessaris
- i, finalment, fer-ne un seguiment per a avaluar el progrés del treball i detectar possibles desviacions i poder-les corregir.

La gestió de projectes ajuda a respondre preguntes per a poder avaluar si el projecte té desviacions, una bona gestió de projectes ha de poder contestar a:

- Quant de temps exigirà aquest projecte?
- Si una tasca s'endarrereix, quant de temps s'endarrerirà el projecte?
- Es tenen prou recursos, humans i materials, per tal de completar el projecte, tal com hem definit?
- Quins són els costos associats als recursos pel projecte?
- En cas de retard, què puc fer per a recuperar temps?
- Si es volgués avançar l'acabament del projecte, quins recursos caldria afegir i quin impacte econòmic tindria?

La gestió d'un projecte engloba la planificació, l'organització i la direcció de les tasques o activitats i dels recursos necessaris per a aconseguir un objectiu definit, limitat temporalment i amb un pressupost econòmic fixat.

3.1.2. Les fases de la gestió d'un projecte

Generalment, la gestió de projectes agrupa tres fases: la definició i la planificació, la direcció i el seguiment i la generació d'informes.

- **Definició i planificació d'un projecte**. És la part més important, ja que inclou la definició de les tasques i la seva durada, s'analitzen les relacions entre les tasques i l'assignació de recursos a cadascuna, principalment.

Totes les fases posteriors del projecte es faran segons la informació procedent d'aquest disseny.

- **Direcció i seguiment del projecte**. És la fase de seguiment del projecte un cop que ha començat i acaba amb el projecte.
  Inclou el seguiment de les tasques definides, l'assignació prevista dels recursos i els reajustaments necessaris per a reflectir els canvis que es produeixen mentre avança el projecte.

- **Generació d'informes sobre el projecte**. Un cop acabat el projecte, en aquesta fase es produeixen els diferents informes finals.
  Tot i així, durant la fase de direcció també es generen informes de seguiment i durant la fase de creació s'elaboren els informes de planificació.

En funció del tipus de projecte o objectiu a assolir, aquestes fases es subdividiran en més o menys subetapes, podent parlar, fins i tot, de models de projecte segons la seva tipologia: construcció d'un edifici, aixecament d'un pont, elaboració d'una campanya de publicitat o la implantació d'una eina CRM (*customer relationship management*), o el disseny i implantació d'un web, en l'àmbit dels projectes informàtics.

3.2. Planificació i seguiment

La planificació d'un projecte, tal com s'ha vist en l'apartat "Gestió d'un projecte", és la primera fase de tot projecte, i consisteix a definir quines tasques caldran per a aconseguir la fita marcada, quins recursos, materials i humans, caldrà gestionar, quant de temps es necessitarà per a dur-lo a terme i finalment, quins costos associats té.

Un cop s'ha iniciat el projecte, cal fer-ne un seguiment per tal d'avaluar l'avanç de les tasques, comprovar-ne la realització, identificar possibles endarreriments i gestionar les incidències que sorgeixin.

3.2.1. Les tasques d'un projecte

Un projecte està constituït per un conjunt de tasques (o activitats).

Una tasca és necessària per al projecte si la seva execució contribueix a finalitzar-lo i, d'altra banda, si la seva no execució comporta una dificultat en l'assoliment dels objectius.

Una tasca es determina per les característiques següents:

- Nom,
- Durada estimada,
- La seva interrelació amb altres tasques del projecte,
- Els recursos que requereix per a ser realitzada,

- Una descripció sobre en què consisteix.

Un tipus de tasca particular són les fites, que corresponen a tasques de durada nul·la i que serveixen per a indicar punts de control del projecte.

Fites

Exemples de fites són l'inici del projecte o l'assoliment parcial d'objectius.

Les fites també poden correspondre a tasques externes al projecte com, per exemple:

- La concessió d'una llicència d'obres per part de l'ajuntament.
- L'aprovació d'un crèdit bancari per a dur a terme la inversió necessària del projecte.
- Etc.

En aquests casos, atès que la tasca no és controlable per l'equip de treball, no es considera ni un temps ni un esforç concret. Tot i així, en aquests casos, tot i que la responsabilitat de la tasca és externa al projecte, sí que cal considerar els riscos associats, tal com es comentarà més endavant en l'apartat "Avaluació de riscos i eines".

Per exemple, les tasques necessàries per a planificar i dur a terme un programa de formació en l'empresa podrien ser les següents:

1) Identificar les necessitats dels treballadors.
2) Planificar el programa de formació.
3) Dissenyar i elaborar els cursos o contractar uns consultors externs per a dur-los a terme.
4) Convocar els participants.
5) Dur a terme les activitats de formació.
6) Avaluar els participants.
7) Avaluar els formadors i el programa.

En aquest cas, caldria fer una descomposició més exhaustiva d'aquestes tasques o fases.

Un projecte es constitueix d'un conjunt de tasques (o activitats) determinades per diverses característiques com ara el nom, la durada estimada, la seva interrelació amb altres tasques del projecte, els recursos que requereix per a ser realitzada i una descripció sobre en què consisteix.

3.2.2. Les interrelacions entre les tasques

Les diferents tasques d'un projecte no s'executen, habitualment, totes a la vegada ni totes de manera seqüencial, una darrere l'altra. Cal analitzar quines tasques són dependents d'altres, quines són independents, i entre les que són dependents cal prioritzar-ne l'execució en funció de la criticitat i de les de-

pendències que hi ha. Un cop es té l'ordre d'execució, cal ordenar-les en el temps d'acord amb els recursos humans i les seves capacitats, de manera que sigui possible fer-les totes respectant l'ordre i minimitzant el temps d'execució, i per tant també minimitzant el cost.

Fer totes les tasques de manera successiva comportaria allargar innecessàriament el projecte i fer-les totes simultàniament és impossible, ja que algunes poden dependre d'altres (és a dir, per començar una tasca cal haver-ne acabat unes altres).

Per això, un cop s'han identificat les tasques principals del projecte, cal donar un pas més i establir l'encadenament més lògic i convenient.

Unes tindran una prioritat, unes altres s'han de fer en moments diferents, unes terceres seran seqüencials a unes de determinades i unes altres es podran fer en paral·lel.

Per exemple, en la construcció d'un edifici, cal acabar les excavacions abans de poder iniciar els treballs de cimentació i no s'acostuma a pintar fins que els electricistes i els paletes han acabat la feina.

Aquestes relacions es coneixen amb el nom de *precedència i successió de les tasques*.

Les tasques o activitats d'un projecte no es fan totes de manera paral·lela sinó que algunes es faran de manera consecutiva i dependran de la finalització o l'inici d'altres.

3.2.3. La gestió dels recursos

Per a moltes persones, la gestió dels projectes s'atura en la planificació de les tasques, tractades en l'apartat anterior. És a dir, en la descomposició del projecte en tasques i subtasques, en la determinació de la durada de cadascuna i en la definició de les interrelacions i vinculacions existents entre les diverses parts que componen el projecte.

Malauradament, només amb aquesta feina no es pot garantir l'èxit i l'eficiència d'un projecte.

Si no s'introdueix en la planificació del projecte la gestió i assignació de recursos a cadascuna de les tasques, la planificació podrà assolir un dels objectius, que és l'assoliment de les fites, però no es contemplaran aspectes com el control dels costos, la correcta utilització i optimització dels recursos disponibles (persones, maquinària, materials...).

Així doncs, la realització de les tasques que s'han identificat ha d'estar acompanyada de l'assignació dels recursos que s'han d'utilitzar en cadascuna.

La gestió dels recursos és molt important en la gerència d'un projecte per diverses raons:

- Els recursos disponibles, humans, tècnics, financers, sempre es limiten a una empresa o organització.
- Els tipus de recursos utilitzats i la seva quantitat determinen els costos del projecte.
- Els recursos i, en especial, els humans i tècnics, són els responsables de dur a terme les tasques en el temps previst i amb la qualitat necessària.
- Els projectes exigeixen l'ús de recursos molt variats (màquines, especialistes en diferents matèries, mà d'obra, fungibles...).
- Aquests recursos no es concreten de manera estable durant tot el projecte, sinó que per a cada tasca es necessiten recursos diferents en naturalesa i quantitat.

Hi ha dos aspectes bàsics en la gestió de recursos: la previsió i l'equilibri dels recursos.

D'una banda, per *previsió* s'entén que cada activitat, en ser executada, compti amb la garantia de disposar dels recursos necessaris, mentre que, d'una altra, l'equilibri té com a requisit la previsió i procura cobrir les necessitats dels recursos utilitzats durant el projecte, de manera que es minimitzi l'existència de recursos ociosos (o sobrants) durant l'execució.

La realització de les tasques que s'han identificat en un projecte només es poden aconseguir amb l'assignació dels recursos humans responsables de la seva execució i els recursos materials que s'han d'utilitzar en cadascuna.

Per a saber-ne més

En l'apartat "L'equip de treball" d'aquest mateix mòdul es presenta una descripció dels principals papers que poden participar en un projecte de sistemes informàtics.

3.2.4. La redistribució dels recursos

Tal com es comentava en l'apartat anterior, un dels objectius de la gestió dels recursos és minimitzar l'existència de recursos ociosos i, a la vegada, garantir que totes les tasques del projecte disposaran dels recursos necessaris en el moment just.

Així, tal com s'ha vist en l'apartat "Planificació i seguiment", una de les principals activitats en la fase de seguiment és detectar les incidències i els retards de les diferents activitats del projecte i intentar corregir-les.

Els canvis de planificació i retards provoquen també desajustos en la gestió dels recursos, ja que aquests s'han de reassignar en funció d'aquests canvis.

D'altra banda, tal com ja hem esmentat, els recursos solen ser escassos i limitats, per la qual cosa, de vegades la menor disponibilitat d'aquests recursos, respecte a la previsió feta en la fase de definició, obliga a redistribuir les càrregues d'aquests recursos durant la vida del projecte.

Una responsabilitat important del cap de projectes i del gerent de recursos és aplicar tècniques i utilitzar metodologies adequades per a la correcta assignació, reassignació i resolució de sobreassignacions de recursos a les tasques.

3.2.5. Gràfics de temps i diagrames

Hi ha dos diagrames bàsics en la gestió de projectes, el diagrama de Gantt i el diagrama Pert i tenen la forma següent:

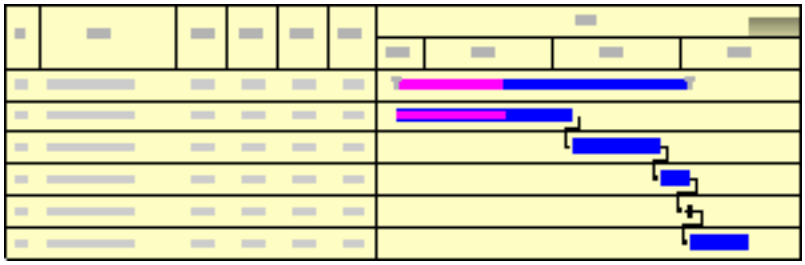


Diagrama de Gantt

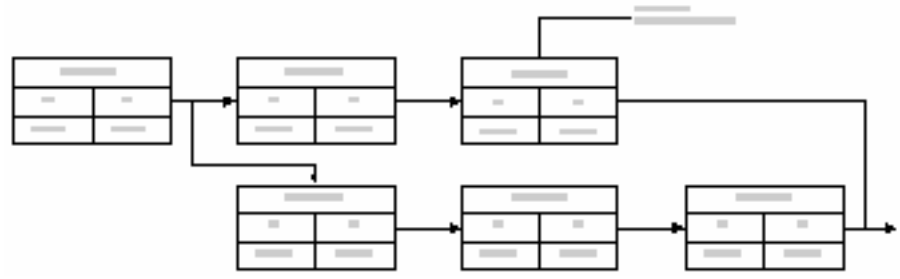


Diagrama PERT

Cadascun té una sèrie d'avantatges i inconvenients, essent l'ús de tots dos combinats la millor opció per a la gestió de projectes.

Una manera no gràfica de representar les tasques en un projecte és per mitjà de taules.

Un exemple podria ser:

Taula de tasques

Núm. tasca	Nom tasca	Durada (en dies)
1	Inici	0
2	A	4
3	B	3
4	C	2
5	D	2
6	E	2
7	F	4
8	G	5
9	H	4
10	I	2
11	J	2
12	Fi	0

El pas següent que s'ha de considerar són les interrelacions entre les tasques, cosa que es coneix com a precedència i successió de les tasques.

Per a saber-ne més

Per a ampliar informació sobre les interrelacions entre les tasques podeu consultar l'apartat "Les interrelacions entre les tasques" d'aquest mateix mòdul.

Suposant les precedències següents:

- ADH (la tasca D no es pot començar fins que es finalitzi la tasca A, i darrere d'aquesta tasca D hi ha la tasca H).
- BEIJ.
- CFIJ.
- CGH.
- IJ (quan acabi la tasca I es podrà començar la tasca J).

Per a exemplificar, podríem trobar que:

- La tasca A és la fita d'inici del projecte.
- La tasca D és la recopilació de les dades necessàries per a dur a terme la definició funcional.
- La tasca H és el disseny funcional de la solució que cal implementar.

O per exemple que:

- La tasca C és la definició de l'arquitectura de la infraestructura del sistema (maquinari).
- La tasca F és l'adquisició del maquinari necessari.
- La tasca I és la instal·lació d'aquest maquinari.
- La tasca J és la configuració de les màquines.

Aquestes dependències es reflectirien de la manera següent en la taula:

Taula amb precedències

Núm. tasca	Nom tasca	Durada (en dies)	Tasca precedent
1	Inici	0	
2	A	4	
3	B	3	
4	C	2	
5	D	2	A
6	E	2	B
7	F	4	C
8	G	5	C
9	H	4	D;G
10	I	2	E;F
11	J	2	I
12	Fi	0	

O si es considerés el número de la tasca, en lloc del nom:

Taula amb precedències

Núm. tasca	Nom tasca	Durada (en dies)	Tasca precedent
1	Inici	0	
2	A	4	
3	B	3	
4	C	2	
5	D	2	2
6	E	2	3
7	F	4	4
8	G	5	4
9	H	4	5;8
10	I	2	6;7
11	J	2	10
12	Fi	0	

Seguint amb l'exemple, atès que les tasques A, B i C (2, 3 i 4) no depenen de cap altra tasca, començaran totes a la vegada i es podrà considerar que el projecte ha finalitzat quan s'hagin dut a terme les tasques H, I i J (9, 10 i 11).

Aquesta taula de tasques es podria representar, també gràficament, amb un diagrama de Gantt.

Id	Nom	Durada	Predecessores
1	Inici	0 dies	
2	A	4 dies	1
3	B	3 dies	1
4	C	2 dies	1
5	D	2 dies	2
6	E	2 dies	3
7	F	4 dies	4
8	G	5 dies	4
9	H	4 dies	5;8
10	I	2 dies	6;7
11	J	2 dies	10
12	Fi	0 dies	9;11

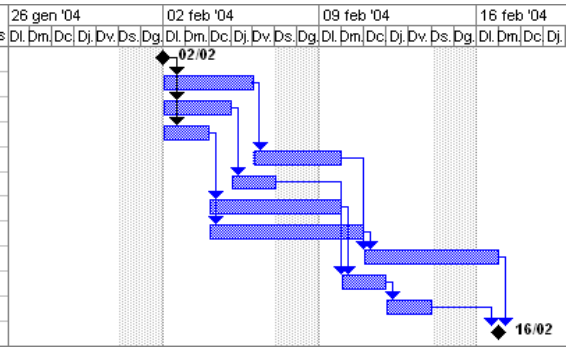


Diagrama de Gantt

Hi ha diversos gràfics i diagrames que representen les tasques d'un projecte i les seves interrelacions. Els principals són el diagrama de Gantt, el diagrama de PERT i les taules de tasques.

3.2.6. Eines, mètodes i tècniques

Camí crític

Una de les principals tècniques o mètodes de gestió de projectes és el càlcul del camí crític.

Un camí és cadascun dels encadenaments necessaris de tasques, partint de les tasques inicials i seguint les successions (precedències) definides.

Exemple anterior

En l'exemple de l'apartat anterior els diferents camins serien ADH, BEIJ, CFIJ, CGH.

Cadascun d'aquests camins té una durada prevista, que és la suma dels temps necessaris per a dur cadascuna.

En estar encadenades, la durada total del camí és la suma de les seves durades.

Seguint l'exemple, les durades dels diferents camins del projecte són les següents:

- ADH = 4 + 2 + 4 = 10 (el temps necessari per a dur a terme les tasques A, D i H és de 10 dies).
- BEIJ = 3 + 2 + 2+ 2 = 9
- CFIJ = 2 + 4 + 2 + 2 = 10
- CGH = 2 + 5 + 4 = 11

És molt important, en tot projecte, detectar el camí crític, que és el més llarg.

La seva importància és donada per dos factors:

- Tot retard en una de les tasques del camí crític provocarà que el projecte, en conjunt, s'endarrereixi. És a dir, que finalitzi més tard.

- Tot avançament en l'execució d'una tasca que pertanyi al camí crític comportarà un avançament en l'execució del projecte.

Exemple anterior (continuació)

Segons l'exemple, un retard d'un dia en una de les tasques B o E no comportaria un retard en el projecte:

- ADH = 10
- BEIJ = 10
- CFIJ = 10
- CGH = 11

En canvi, un avançament (recuperació de temps) de dos dies en la tasca G provocaria que el projecte pogués acabar abans:

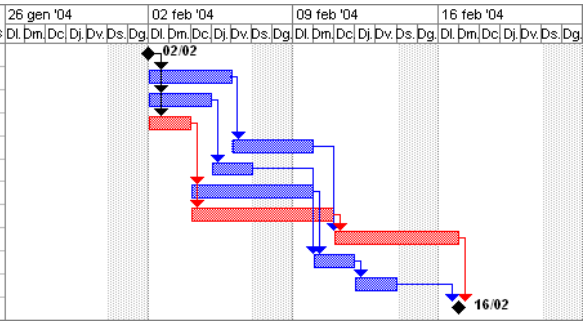
- ADH = 10
- BEIJ = 9
- CFIJ = 9
- CGH= 9

En aquest cas, a més, el canvi de durades de les tasques comporta que el camí crític passi a ser el que formen les tasques ADH.

El camí crític i les tasques crítiques s'acostumen a destacar en els gràfics amb l'objectiu de cridar l'atenció dels responsables del projecte per tal que en maximitzin el seguiment.

Partint de l'exemple inicial.

Id	Nom	Durada	Predecessores
1	Inici	0 dies	
2	A	4 dies	1
3	B	3 dies	1
4	C	2 dies	1
5	D	2 dies	2
6	E	2 dies	3
7	F	4 dies	4
8	G	5 dies	4
9	H	4 dies	5;8
10	I	2 dies	6;7
11	J	2 dies	10
12	Fi	0 dies	9;11



Els marges

Les tasques que no pertanyen al camí crític es diu que tenen marge, és a dir, que el seu moment d'inici i el de fi no estan rígidament marcats sinó que poden variar, un marge, en el temps, sense que això afecti la data de finalització del projecte.

Aquesta informació és molt important per al cap de projectes, ja que compta amb uns marges de temps de retard o avançament de determinades tasques per tal de compensar desviacions del projecte, sobreassignacions de recursos...

A partir de la durada de cada tasca i la relació amb les seves predecessores, aquests mètodes permeten calcular:

- "Tan aviat com" pot començar una tasca.
- "Tan aviat com" pot finalitzar una tasca.
- "Tan tard com" pot començar una tasca.
- "Tan tard com" pot finalitzar una tasca.

De manera simplificada s'anomena *tan aviat* el temps més d'hora que una tasca pot començar o acabar i *tan tard* el temps més tard admissible per a una tasca.

Per a això s'utilitzen dos mètodes de càlcul de les dates "tan aviat" i "tan tard" d'un projecte atenent a l'ordenació "tan aviat" i a l'ordenació "tan tard" respectivament.

- L'ordenació "tan aviat com" determina per a cada tasca la data d'inici i fi "tan aviat com", tenint en compte les relacions amb les tasques predecessores i les seves durades. En l'exemple, suposant que el projecte s'inicia el dia 1, la tasca inici no tenia durada, per la qual cosa, la tasca A comença el dia 1 i acaba el dia 4 i la tasca D començarà el dia 5 i acabarà el dia 6.

- L'ordenació més tard calcula per a cada tasca la data de començament "més tard" i la data d'acabament "tan tard" sense canviar la data del projecte determinada per la planificació "tan aviat". Per tal de determinar la planificació "tan tard" s'ha de començar per l'última tasca del projecte i "remuntar" successivament fins a l'inici.

Un cop calculades les dates d'inici i acabament "tan aviat" i "tan tard" es poden calcular els marges.

L'ús correcte dels marges és important per a la gestió dels projectes, ja que les tasques amb marge es poden desplaçar en el temps, avançar-se o endarrerir-se, dins dels marges establerts per la mida del marge sense que això afecti el termini del projecte.

Càlcul de les dates "tan aviat"

El projecte s'inicia el dia 1:

a) Per a una tasca amb una sola predecessora,

- el començament "tan aviat" d'una tasca és igual a la finalització "tan aviat" de la tasca predecessora + 1. Seguint l'exemple, per a la tasca A = 0+1 = 1 i per a la tasca D = 4 + 1 = 5

- el final "tan aviat" d'una tasca és igual al començament "tan aviat" de la tasca + la durada de la tasca – 1. Per exemple, el final "més aviat" de la tasca D és igual a 5 + 2 – 1 = 6. La tasca acabarà el dia 6 i per això es resta una unitat en la fórmula.

b) Per a una tasca amb diverses predecessores,

- El començament "tan aviat" d'una tasca és el valor màxim dels acabaments "tan aviat" de les tasques predecessores + 1. Per exemple, per a la tasca H = Màx (6, 7) + 1 = 7 + 1 = 8

- El final "tan aviat" d'una tasca és igual al començament "tan aviat" de la tasca + durada – 1. Per a la tasca H de l'exemple, seria 8 + 4 – 1 = 11.

La taula completa de les dates "tan aviat" de l'exemple seria la següent:

Núm. tasca	Nom tasca	Durada (en dies)	Tasca precedent	Inici "tan aviat"	Fi "tan aviat"
1	Inici	0		0	0
2	A	4	Inici	1	4*
3	B	3	Inici	1	3
4	C	2	Inici	1	2
5	D	2	A	5	6
6	E	2	B	4	4
7	F	4	C	3	6
8	G	5	C	3	7
9	H	4	D;G	8	11
10	I	2	E;F	7	8
11	J	2	I	9(**)	10
12	Fi	0	H;I;J	11	11

(*) La data d'acabament "tan aviat" és el començament "tan aviat" (1) més la durada (4) menys 1 = 4
(**) La data de començament "tan aviat" és l'acabament "tan aviat" de la predecessora (8) més 1 = 9

Càlcul de les dates "tan tard"

El projecte acaba el dia 11.

a) Per a una tasca amb una sola successora

- El final "tan tard" d'una tasca és el final "tan tard" de la successora – la durada de la tasca successora. Per exemple, per a la tasca J seria 11 – 0 = 11 i per a la tasca I, 11 – 2 = 9.

- El començament "tan tard" d'una tasca és el fi "tan tard" de la tasca – durada + 1. Com a exemple, per a la tasca I, el començament "tan tard" és 9 – 2 + 1 = 8.

b) Per a una tasca amb diverses successores

- El final "tan tard" d'una tasca és el mínim dels començaments "tan tard" de les tasques successores – 1. Per exemple, per a la tasca I = Mín dels començaments "tan tard" (J, Fi) – 1 = Mín (10, 11) – 1 = 9.

- El començament "tan tard" d'una tasca és la fi "tan tard" de la tasca – durada + 1. Per a la tasca I de l'exemple, seria 9 – 1 + 1 = 8.

La taula completa de les dates "més aviat" i "més tard" serà la següent:

Núm. tasca	Nom tasca	Durada (en dies)	Tasca precedent	Inici "tan aviat"	Fi "tan aviat"	Inici "tan tard"	Fi "tan tard"
1	Inici	0		0	0	1	1
2	A	4	Inici	1	4	2	5
3	B	3	Inici	1	3	3	5
4	C	2	Inici	1	2	1	2
5	D	2	A	5	6	6	7
6	E	2	B	4	4	6	7
7	F	4	C	3	6	4	7
8	G	5	C	3	7	3	7
9	H	4	D;G	8	11	8	11
10	I	2	E;F	7	8	8	9
11	J	2	I	9	10	10	11
12	Fi	0	H;I;J	11	11	11	11

Es pot comprovar, per exemple, que en algunes tasques, els temps "tan aviat" i "tan tard" coincideixen, és a dir, les dates "tan aviat" i "tan tard" de començament i final són les mateixes com, per exemple, les tasques C, G i H.

En canvi, d'altres tasques com, per exemple, la E, tenen marge, cosa que significa que es poden iniciar en diferents moments o es poden allargar sense alterar el termini final previst.

Mètodes de redistribució

Els mètodes de redistribució dels recursos pretenen aconseguir un diagrama de càrrega tan uniforme com es pugui.

L'aplicació d'aquests mètodes de redistribució es pot deure a diverses raons:

- Canvis en la planificació de les tasques.
- Incidències ocorregudes durant el projecte i retards.
- Canvis en disponibilitat dels recursos previstos.
- Encavalcament temporal de tasques que requereixen el mateix tipus de recurs.

Aquesta operació de redistribució és molt més complicada del que pugui semblar.

Per a poder-la dur a terme, és necessari redibuixar el gràfic de Gantt, basant-se en la taula dels temps d'"Abans i Després" i els marges totals que pot suportar cada tasca. D'aquesta manera es podrà avaluar la possibilitat de realitzar les tasques en moments en què es disposi de recursos sobrants (ociosos o sense assignació).

Les tasques que componen el camí crític no tenen marge i, per tant, si no es disposa de recursos per a dur-les a terme, és impossible que el projecte acabi en la data prevista.

Per aquesta mateixa raó, en la redistribució de recursos, cal donar prioritat a les tasques crítiques.

Un cop redistribuïts els recursos, si continuen havent-hi problemes d'assignació, cal:

- avaluar el possible endarreriment o avançament de tasques, la qual cosa pot implicar un retard en el projecte, provocant una desviació en temps,

- incorporar nous recursos per a dur a terme aquestes tasques, la qual cosa implicarà un augment dels costos previstos, provocant una desviació pressupostària.

3.3. Projectes informàtics

El resultat final d'un projecte informàtic és un sistema informàtic compost d'un conjunt d'elements i mòduls que processen i gestionen informacions i dades, per mitjà de diferents mètodes, procediments i controls.

Un sistema informàtic, tal com hem comentat en les aplicacions del programari (vegeu l'apartat "Aplicacions del programari") pot tenir unes funcionalitats i estar dirigit a uns entorns molt diversos.

En l'apartat anterior hem presentat els principals conceptes de la gestió de projectes generals.

Ara bé, un projecte informàtic té unes particularitats i característiques específiques, sobretot pels elements que componen un sistema informàtic:

- Maquinari. Dispositius electrònics i dispositius electromecànics, com la CPU i la memòria (vegeu el mòdul "El maquinari"), els circuits integrats, components electrònics, targetes de circuit imprès...

- Programari. Són els diferents programes i estructures de dades, i també les aplicacions que s'integraran o utilitzaran amb els sistemes, com els gestors de bases de dades, els servidors de missatgeria, servidors d'aplicacions o Internet, per exemple.

- L'equip de treball. Compost per diversos professionals que adopten un paper o més.

Aquestes característiques i particularitats fan que en la gestió de projectes informàtics calgui tenir en compte diferents fases, com ara el disseny, el desenvolupament i integració, la verificació i la implantació, i també la documentació necessària tant per a explicar l'ús i les operacions del sistema com per donar fe de com s'ha implementat el sistema per a facilitar un manteniment posterior.

En els apartats següents introduirem els principals conceptes en la definició inicial del projecte (avaluació de riscos, recursos, equip de treball, avaluació d'esforços), el disseny de la solució, la gestió dels costos, el desenvolupament i la integració, i també la documentació i el manteniment del programari.

Un projecte informàtic té unes característiques particulars, no solament per les tasques que s'hi han de desenvolupar (enginyeria del programari) sinó també pels elements que componen els sistemes informàtics: el maquinari i el programari.

3.3.1. Fases i tasques principals

Tot projecte informàtic segueix les mateixes fases, encara que en funció de la seva tipologia i naturalesa, es duen a terme més o menys subfases.

En els apartats següents es definiran, amb més detall, cadascuna d'aquestes fases:

- Disseny de la solució. Té com a objectiu crear una solució conceptual, identificant l'aplicació de les principals funcionalitats, les característiques necessàries i les adaptacions a dur a terme. També es defineixen diferents models operacionals per a l'ús i treball del futur sistema. Així mateix, es dissenyen els principals elements i mòduls i es prepara l'entorn de desenvolupament, l'arquitectura del sistema i els processos d'assegurament de la qualitat del producte. En aquesta fase es duen a terme els processos següents, principalment:
  - Inici del projecte.
  - Requeriments de negoci.
  - Requeriments funcionals.
  - Disseny general.
  - Disseny detallat.
  - Preparació de l'entorn de desenvolupament.
- Desenvolupament i construcció. L'objectiu de la fase de desenvolupament i construcció és la creació dels codis dels components, la definició i configuració de les bases de dades, la construcció dels programes de les aplicacions, la documentació i les proves d'assegurament de qualitat. Els dos principals processos són el desenvolupament i la preparació de la implantació.
- Implantació. En aquesta fase final es configura l'entorn de producció, es verifica la solució completa, s'ajusten els paràmetres finals, es fan les proves de funcionalitat i es forma els usuaris de la solució i es lliuren els documents d'ús i treball.

Tot i així, prèviament al disseny, cal definir el projecte avaluant els riscos, els recursos necessaris i definint l'equip de treball.

Aquesta planificació definirà els costos necessaris.

I per a obtenir aquesta planificació cal fer una avaluació i estimació seguint un conjunt d'eines i mètodes.

Un cop definit el projecte, cal fer el disseny de la solució, desenvoluparla i implantar-la.

Avaluació de riscos i eines

Es pot definir el risc en la gestió de projectes com la incertesa que implica l'execució d'una estimació. És a dir, la possibilitat que sorgeixin imprevistos i canvis que no permetin complir la planificació tant en temps com en costos, treball o disponibilitat de recursos, entre d'altres.

Tot projecte comporta uns riscos associats, més o menys probables i greus, en funció de la complexitat.

En tot projecte informàtic hi sol haver diferents components o elements que poden comportar un risc:

- La mateixa naturalesa del programari. Errors de desenvolupament, eines poc provades, incompatibilitats d'integració i comunicació entre aplicacions diferents...

- Problemes associats amb el maquinari. Fallades a les màquines, errors d'integració de sistemes...

- La complexitat del projecte. Com més gran sigui un projecte i més elements hi intervinguin (nombre de gent, proveïdors, màquines, aplicacions...), més probabilitats hi ha que hi hagi fallades en l'equip de producció.

- La mida del projecte. Com més gran i complex sigui el projecte, més complicada serà dur a terme la seva valoració i planificació i, per tant, hi haurà més probabilitats d'error.

Per tant, doncs, en tot projecte informàtic, en la seva fase inicial i dins de la valoració, caldrà avaluar els riscos i prendre les mesures pertinents.

Aquest procés es pot dividir en tres passos:

- Identificació dels riscos,
- Avaluació dels riscos,

- Actuacions corresponents.

Per a identificar els riscos cal tenir en compte els aspectes següents:

- La descripció final del producte o productes del projecte,
- La planificació d'altres àrees amb les quals interaccioni el nostre projecte,
- La informació històrica en el desenvolupament d'altres projectes informàtics similars.

En aquesta primera fase d'identificació dels riscos cal aconseguir els elements següents:

- Relació de possibles fonts de risc. Tenen una probabilitat alta o mitjana de succeir.
- Una llista de riscos potencials, que tenen una probabilitat menor que succeeixin com, per exemple, la marxa, de l'empresa, de part de l'equip de treball o canvis en la tecnologia d'implementació del programari.
- Una llista de símptomes de risc.

Un cop recollida i identificada aquesta informació, cal esbrinar la probabilitat que s'esdevingui cadascun d'aquests riscos i quines conseqüències tindrien per al projecte.

Per a això caldrà dur a terme els passos següents:

- Definir una escala de valors que permeti assignar a cada risc la seva probabilitat que succeeixi.
- Enumerar les conseqüències d'aquests riscos. És a dir, estimar l'impacte del risc en el projecte i en el producte o productes finals.
- Identificar en quin moment podria succeir i quant de temps duraria.

Finalment, el tercer pas, un cop es disposi d'aquestes dades, és decidir quins riscos s'han de prevenir i per quins altres és més rendible assumir-los sense establir cap mesura.

En el segon cas, simplement, cal acceptar-los i desenvolupar, si s'escau un possible pla de xoc, per si succeïssin.

Per al primer grup de riscos es pot optar per dues alternatives:

- Evitar-los, suprimint-ne la causa.
- Si no és possible evitar-los, caldrà reduir la possibilitat que s'esdevinguin.

La gestió dels riscos en un projecte consisteix en l'anàlisi de les possibles incidències que puguin sorgir, l'avaluació de la seva magnitud, l'impacte i les conseqüències que tindrien en el projecte i els productes que s'han de desenvolupar i, finalment, prendre les mesures adequades, tant per a evitar-los com per a disminuir-ne la probabilitat o, simplement, acceptar-los.

Recursos necessaris

Durant la primera fase de definició i planificació del projecte caldrà especificar els recursos necessaris per a portar-lo a terme.

Tal com ja hem comentat en l'apartat anterior caldrà definir quins recursos són necessaris, com es poden aconseguir (contractació, membres de l'organització, lloguer...), quan es necessitaran i durant quant de temps.

La diferència, però, tal com ja hem comentat anteriorment, és que caldrà tenir en compte dues tipologies especials de recursos: els recursos de programari i de maquinari.

Per a saber-ne més

Per a tenir més informació sobre el tipus de recursos i la seva gestió podeu consultar l'apartat "La gestió dels recursos" d'aquest mateix mòdul.

- Els recursos de maquinari, que es necessiten tant per a desenvolupar el projecte (servidors de desenvolupament, màquines de verificació...) com els que portarà o els que formaran part del producte final (infraestructura de producció, terminals, dispositius per als usuaris...).

- Recursos de programari, que engloben tant els llenguatges de desenvolupament els compiladors o intèrprets corresponents, les eines de suport al disseny i producció, com el programari resultant del projecte (programa i/o aplicacions que formaran part del nou sistema informàtic).

Per a saber-ne més

Per a tenir més informació sobre els llenguatges de desenvolupament podeu consultar l'apartat "Principals llenguatges de desenvolupament". Per a tenir més informació sobre compiladors o intèrprets podeu consultar l'apartat "Traductors". Per a tenir més informació sobre les eines de suport al disseny i producció, podeu consultar l'apartat "Eines CASE". Tots aquests apartats els trobareu en aquest mateix mòdul.

En un projecte informàtic no solament cal considerar els recursos humans i materials, sinó també els recursos de programari i de maquinari.

Només com a introducció, enumerem a continuació alguns dels recursos de programari que poden participar i ser necessaris per a dur a terme el projecte:

- Eines d'anàlisi i disseny que permeten crear els models de programari per tal de validar la consistència i la validesa del disseny, disminuint, d'aquesta manera, possibles errors de desenvolupament.

- Eines de simulació i creació de prototipus. Faciliten que tant l'equip de treball com sobretot, l'usuari final del sistema, puguin avaluar i entendre el funcionament del futur sistema abans de desenvolupar-los.

- Eines d'estructura que permeten la definició i gestió de les bases de dades i fonts d'informació (eines de gestió documental, de contingut...).

- Eines de desenvolupament: editors, compiladors, depuradors, llenguatges de desenvolupament, llenguatges d'accés i gestió de bases de dades...

- Programari ja existent, bé siguin productes estàndard, del mercat o llibreria de components desenvolupats anteriorment, que es poden reutilitzar en el projecte.

- Eines de prova que faciliten la tasca de verificació tant del producte final com dels passos intermedis i la integració dels diferents components del sistema informàtic futur.

Entre els recursos de programari cal tenir en compte no solament el producte final sinó també els recursos necessaris per a desenvolupar-los (eines d'anàlisi i disseny, simulació, llenguatges de desenvolupament, llibreries existents o eines de verificació).

La reutilització del programari és molt important en el desenvolupament d'un projecte. Utilitzar mòduls o parts ja desenvolupades per altres projectes fa que el temps de projecte sigui més curt i se n'accentuï la fiabilitat, ja que són parts ja provades per l'usuari. Un alt nivell de reutilització indica experiència, fiabilitat i menys cost.

Avaluació i estimació

Per tal de poder establir els recursos necessaris per a dur a terme el projecte, cal fer-ne una valoració, és a dir, cal estimar la "magnitud" del projecte.

Per a aquest procés, cal disposar, d'una banda, d'informació històrica dels anteriors projectes similars realitzats, i de l'altra, comptar amb l'experiència dels professionals i experts de la companyia.

Així mateix hi ha un conjunt d'indicadors i mètriques de productivitat del programari que permeten donar suport a aquesta tasca:

- Línies de codi. És un dels indicadors més comuns a l'hora de planificar l'esforç necessari.
- Punts de funció o funcionalitats que s'han de desenvolupar en el sistema informàtic.
- Etc.

Per a dur aquesta tasca, els analistes també tenen un conjunt d'eines al mercat que implementen tècniques d'estimació que apliquen fórmules matemàtiques amb paràmetres obtinguts a partir de la informació històrica.

Un dels models més utilitzats és el COCOMO (*constructive cost model*). Aquest model calcula el nombre de persones i el temps de gestió necessari per a finalitzar un projecte a partir de:

- la mida del projecte, és a dir, línies de codi,
- el tipus de programari a desenvolupar,
- i l'organització i experiència de l'equip de treball.

Gestió del cost

L'estimació dels costos es realitza, d'una banda, a partir dels requeriments del projecte i la informació històrica, però sobretot, a partir de la planificació dels recursos necessaris definits (equip de treball, recursos materials, recursos de programari i recursos de maquinari).

Per això és important fer el càlcul dels costos un cop avaluada la planificació i les necessitats de recursos dins la fase de definició inicial del projecte.

Aquesta estimació ha de ser una valoració quantitativa del cost necessari per tal de disposar dels recursos necessaris per a dur a terme el projecte en el temps previst.

Durant tot el projecte, caldrà fer un seguiment del pressupost inicial per a verificar que s'està complint i, en cas contrari, caldrà actualitzar-lo i prendre les mesures necessàries.

Aquest control de la despesa implica vigilar la despesa que es va produint per tal de detectar variacions sobre l'estimació, registrar els canvis que es produeixin i informar els implicats dels canvis autoritzats.

> Cal que la planificació del projecte s'avaluï i defineixi el pressupost inicial per tal de poder-ne fer un seguiment durant l'execució del projecte i prendre les mesures necessàries en cas de desviació.

El cost d'un projecte té dues grans partides:

- Cost del maquinari i cost del programari.
- Cost de les hores de desenvolupament.

El primer dels dos costos no acostuma a tenir grans desviacions; el segon, en canvi, fluctua molt i és el que, si no s'ha fet una planificació acurada, pot dur a grans desviacions que ocasionin pèrdues en el projecte.

3.3.2. L'equip de treball

Tal com s'ha detallat en l'apartat de "Planificació i seguiment", a continuació detallem els principals rols que poden participar en un projecte de sistemes informàtics.

La pretensió és enumerar i descriure la majoria de rols dels membres que poden participar en projectes d'aquestes característiques.

Per tal de descriure aquests diferents rols dels participants d'un equip de projectes, s'agrupen en diferents categories o famílies:

- Gestió del projecte.
- Processos i negocis.
- Màrqueting.
- Disseny gràfic i multimèdia.
- Enginyeria.
- Control de la qualitat.

Depenent de la tipologia del projecte l'equip de treball estarà format per uns rols o uns altres.

En un projecte, una persona pot assumir més d'un rol, per exemple, pot ser que un membre de l'equip adopti el paper de cap de projecte amb una dedicació del 50% del seu temps, i l'altra part del seu temps actuï com a analista funcional, en la fase de disseny i com a verificador, en la fase final de verificació, o pot ser, per exemple, que una mateixa persona assumeixi els papers d'analista funcional, en la primera fase, i d'analista programador en la següent.

Analista programador

Persona que s'encarrega d'analitzar el problema que es vol resoldre mitjançant la creació d'una aplicació de programari específica i que ha de determinar la manera exacta com aquest programa l'ha de solucionar.

Exemple web

Per exemple, en un projecte d'implantació d'una solució web, caldrà que a l'equip hi hagi dissenyadors gràfics i desenvolupadors de webs, mentre que en un projecte d'implantació d'una eina de gestió financera caldrà comptar amb la participació d'un expert en la parametrització d'aquella eina i, segurament, d'un consultor de finances i comptabilitat, entre d'altres.

Segons la tipologia i magnitud del projecte, l'equip de treball estarà format per més o menys papers, segons les necessitats. Tot i així, aquests papers es poden agrupar en sis tipus o famílies.

Gestió del projecte

Dins d'aquesta família, s'identifiquen entre d'altres, tres papers:

- **Gerent del projecte**. Assigna els recursos, estableix les prioritats, coordina la interacció i manté l'equip focalitzat en l'objectiu correcte i estableix les pràctiques que asseguren la integritat i qualitat dels artefactes del projecte.

- **Administrador del projecte**. Porta el control d'hores i del pressupost del projecte, elabora els estats de comptes i la facturació, programa i controla el pagament a proveïdors i dóna suport als membres de l'equip amb les despeses de viatge, l'entrada d'hores, localització...

- **Gestor del coneixement**. És el responsable de revisar la documentació que es genera en el projecte i fer complir els estàndards de documentació i revisa que la documentació generada s'adeqüi als acords definits en el projecte.

Sovint aquests tres rols s'unifiquen en una sola persona amb la denominació de cap de projecte o *Project Management*.

Processos i negocis

Sota aquesta família de responsabilitat, s'agrupen els papers següents:

- **Consultor de negocis**. Realitza l'anàlisi de la indústria i de l'empresa i l'estratègia de negoci del client. Assegura la direcció adequada de l'estratègia de posicionament. Lidera l'anàlisi de requeriments, el disseny organitzacional del client, la seva posterior formació i el seu suport operacional. Assegura la realització de tots els requeriments per part de l'equip de desenvolupament i implantació del projecte.

- **Consultor de processos**. És el responsable de fer l'anàlisi dels processos de negoci de cadascuna de les àrees operatives del client que seran afectades pel projecte (per exemple, la gestió de les comandes, l'atenció postvenda als clients, els processos de facturació...), detallant cadascun dels processos del client en fluxos de treball i elaborant el cas de negoci corresponent amb les funcionalitats descrites.

- **Consultor de catàlegs**. Dóna suport a la selecció i classificació dels productes i serveis del client. Defineix els estàndards tècnics i de presentació del contingut de catàlegs. Dóna suport al procés d'integració de proveïdors. Assegura la instal·lació i accés a les eines per a la càrrega de catàlegs en l'ambient de proves (verificació) i el de producció.

- **Líder de transaccions**. Defineix les regles de negoci per a les transaccions, desenvolupant els diagrames de flux de processos, generals i detallats per les transaccions de negoci.

Aquests rols sovint els adapta l'analista funcional amb l'ajut dels experts en la matèria que pugui aportar l'empresa que demana el projecte.

Màrqueting

Dins d'aquest grup s'engloben tots aquells professionals responsables de considerar la gestió de la marca del client, l'anàlisi del mercat, el tractament de la imatge corporativa...:

- **Consultor d'estratègia de marca**. Combina l'estratègia del negoci d'Internet amb la gestió de marca per a assegurar la creació d'experiències de marca rellevant al públic objectiu i desenvolupa recomanacions d'estratègia de posicionament.

- **Consultor-analista de màrqueting**. Prepara els pressupostos, les previsions i els informes dels processos i activitats de màrqueting per al client. Realitza l'anàlisi detallada de les activitats de mercat i contribueix al desenvolupament de l'estratègia de màrqueting del projecte Internet, en cas de la implantació d'un lloc d'Internet.

- **Consultor de màrqueting**. És el responsable de cercar oportunitats estratègiques en el mercat utilitzant la informació proporcionada pel client i qualsevol altre font fiable, presentant, posteriorment, suggeriments per a la planificació de l'estratègia.

Disseny gràfic i multimèdia

La part gràfica de tot projecte és cada vegada més important. Hi va haver un salt qualitatiu de les aplicacions *host* o servidor a les aplicacions i programes Windows, però el gran salt ha estat amb el desenvolupament d'aplicacions Internet i webs. Hi ha tot un conjunt de papers associats a aquestes tasques:

- **Expert funcional creatiu**. Coordina la imatge del projecte, ofereix solucions i coordina a l'equip creatiu, assessorant i guiant el client en les decisions d'imatge.

- **Expert funcional d'art**. Responsable de la direcció artística del projecte, d'establir consistència en el disseny i de proveir d'experiències i idees creatives a la resta d'equip.

- **Dissenyador gràfic**. Responsable dels elements gràfics.

- **Desenvolupador de la interfície gràfica d'usuari**. Programa els elements que compondran el projecte, tant si és una aplicació web com Windows i optimitza aquests elements.

Enginyeria

L'equip d'enginyeria o tècnic és l'equip pròpiament responsable del desenvolupament de tot el programari, la instal·lació i configuració del maquinari i la integració amb la part de comunicacions i xarxes:

- **Director tècnic**. Comprova que tota la documentació de requeriments, anàlisis i disseny estigui completa, correcta i firmada pel client. Dirigeix el desenvolupament des de l'inici fins a l'acabament amb èxit. Coordina l'equip de desenvolupament, proves, QA. Realitza el seguiment i coordina les activitats relacionades amb proveïdors tecnològics.

- Obté requisits tècnics del client per a desenvolupar l'estratègia tecnològica. Assegura l'escalabilitat de la solució i serveix com a punt de contacte del client per a qüestions tècniques.

- **Administrador de contingut**. Desenvolupa les estratègies i processos per a la gestió de contingut. Adapta els servidors de gestió de contingut estàndard del mercat a les necessitats del client i forma el client en l'ús d'aquestes eines.

- **Líder tècnic de contingut**. Defineix l'esquema i dissenya el cicle de treball de continguts. Ajuda a la integració del servidor de gestió de continguts amb la resta del lloc.

- **Arquitecte d'informació**. Analitza els usuaris i les tasques que han de dur a terme, dissenya l'arquitectura del programa i la interfície d'usuari, incloent l'organització de la informació i les regles de navegació. Recull els objectius de negoci, requisits d'usuari, contingut i funcionalitat. Organitza el contingut i desenvolupa categories, esquemes de noms i jerarquies de navegació. Finalment, crea i documenta casos d'ús.

- **Desenvolupador del *back-end***. És el responsable de creació dels accessos a la informació (bases de dades, pàgines HTML...).

- **Desenvolupador de la lògica de negoci**. Construeix els components de la capa lògica de negocis en una aplicació de tres capes.

- **Desenvolupador *front-end***. És el responsable de programar les pàgines i components d'interfície d'usuari.

- **Enginyer de programari**. Defineix i desenvolupa les classes, paquets i subsistemes que formaran part del sistema informàtic.

- **Enginyer de comunicacions**. Avalua els productes existents i coordina la instal·lació, integració, desenvolupament i configuració dels productes seleccionats.

- **Dissenyador de dades**. Participa amb l'arquitecte de programari en la definició de les bases de dades de les aplicacions i la seva interacció. Dissenya l'estructura de dades, verificant i garantint la integritat, la normalització, l'optimització, la migració i la integració de dades.

- **Administrador de bases de dades (DBA)**. Instal·la, genera i manté la base de dades.

- **Programador de bases de dades**. Crea els objectius de bases de dades necessaris per a la solució.

- **Arquitecte de xarxa**. Dissenya l'arquitectura de xarxa dels diferents ambients del projecte (producció, qualitat, verificació, desenvolupament...).

- **Administrador de sistemes**. Administra els sistemes operatius i aplicacions entre els diferents ambients (producció, desenvolupament, QA).

Tots aquests rols sovint es reparteixen en tres perfils:

- Analistes tècnics.
- Programadors.
- DBA.

Assegurament de la qualitat

- **Responsable de qualitat**. Determina la càrrega de treball per a l'equip d'assegurament de la qualitat segons el tipus de projecte. Genera el pla de proves amb els requeriments del projecte en desenvolupament i dirigeix l'execució de les proves.

- **Verificador de codi**. Revisa els estàndards de codificació i el pla detallat de desenvolupament del projecte, per tal de verificar que el codi de l'aplicació compleixi els estàndards abans de lliurar el producte al client.

- **Verificador de seguretat i accés**. Verifica la seguretat de l'aplicació, la qual cosa inclou proves que en mesurin la fiabilitat de l'accés i seguretat.

- **Verificador de funcionalitat**. Genera, d'acord amb els escenaris que el responsable de qualitat ha definit en el pla de proves, les seqüències (*scripts*) i casos de prova per a identificar errors de funcionalitat i requisits. És el res-

ponsable d'assegurar que l'aplicació compleixi amb la funcionalitat d'acord amb els requisits definits pel client.

- **Verificador de "Look & Feel"**. Comprova que el disseny gràfic de la solució compleix els requeriments de Look & Feel abans del seu lliurament final al client.

La feina de test es fa normalment entre:

- Cap de projecte.
- Analistes.
- Usuaris finals.

3.3.3. Disseny

El disseny és la primera fase d'implementació de tot projecte.

Un cop definides les necessitats i requeriments del sistema informàtic i identificats els recursos necessaris i format l'equip de treball, cal dur a terme el disseny del producte que es vol desenvolupar.

Tal com hem comentat, el disseny parteix de les necessitats del sistema i del model d'interrelacions dels elements que hi han d'intervenir.

El disseny és clau en tot projecte, ja que un mal disseny comportaria errors en el desenvolupament.

El primer factor que s'ha de tenir en compte en el disseny de tot sistema informàtic és la fragmentació del problema, és a dir, definir els diferents components que el formaran, dissenyar-los per separat i després dissenyar-ne la integració.

Aquest procés es coneix com el disseny de l'arquitectura.

Disseny de l'arquitectura

Nom que s'utilitza per a definir un conjunt de components maquinari i programari estandarditzats amb què està dissenyat un sistema informàtic: processador, sistema operatiu, connectors, etc.

D'aquesta manera, el disseny se sol dividir en mòduls que es comunicaran i integraran entre ells per a formar el sistema final.

En la fase del disseny també cal definir la infraestructura necessària i l'arquitectura de programari que el suportarà (servidors d'Internet i aplicacions, bases de dades, sistemes de missatgeria, sistemes operatius...).

Una altra part important en la fase de disseny és el disseny de dades. D'aquest disseny en depenen l'organització, els mètodes d'accés o les alternatives de processament de les dades i la informació que ha de gestionar el sistema informàtic.

Com a resultat d'aquest disseny es defineixen les estructures de dades.

Finalment, una part cada vegada més important en el disseny de solucions informàtics és la interfície d'usuari, és a dir, el disseny de l'aspecte "visual" de l'aplicació. Per a això cal tenir en compte aspectes tant de disseny gràfic com d'usabilitat i navegació.

En la fase de disseny d'un projecte informàtic cal definir tant l'arquitectura del sistema i les infraestructures necessàries, com els mòduls i funcionalitats que s'han de desenvolupar i integrar, i també les estructures de dades i el disseny gràfic, usabilitat i navegació de la solució.

També cal definir, en aquesta fase, les proves de verificació que es duran a terme durant el desenvolupament, la integració i la implantació de la solució.

Per a dur a terme aquestes definicions i dissenys, els analistes informàtics tenen diferents mètodes i eines.

L'objectiu d'utilitzar un mètode per al disseny és garantir l'obtenció d'un programari comprensible en què es puguin detectar els errors amb facilitat i que permetin el manteniment i l'extensió posteriors.

Per a dissenyar el programari, convé disposar d'un bon sistema de notació que permeti entendre el disseny d'una manera simple, més fàcilment que llegint el codi d'un llenguatge de programació.

Les notacions ajuden a obtenir un detall de les funcionalitats del sistema a desenvolupar de la manera més clara i comprensible.

Algunes de les principals notacions que s'utilitzen són els nivells d'abstracció, els diagrames de flux, les taules de decisió i els casos d'ús.

L'objectiu d'aquest material no és aprofundir ni detallar aquestes tècniques, mètodes i notacions, sinó tan sols destacar-ne la importància i ús en el disseny dels sistemes informàtics.

Per a poder dissenyar, de manera entenedora, els sistemes informàtics i les funcionalitats a desenvolupar s'utilitzen diferents mètodes, eines i notacions.

3.3.4. Desenvolupament i integració

La fase de desenvolupament comença després de l'aprovació formal dels requisits del sistema per part de l'usuari o client i de les fases de disseny d'alt nivell (anàlisi funcional) i disseny detallat (anàlisi orgànica).

Anàlisi funcional

Nom amb el qual es coneix la primera fase del desenvolupament d'una aplicació que consisteix a reunir totes les dades disponibles del problema que es vol solucionar i crear un diagrama de flux o un pseudocodi amb els elements generals del programa que es crearà per a solucionar-lo.

Anàlisi orgànica

Fase del disseny d'una aplicació en què es converteix l'algorisme escrit en forma de pseudocodi o diagrama de flux a codi que pugui entendre el compilador, intèrpret o assemblador corresponent.

A partir de l'arquitectura definida del programari i del sistema, cal desenvolupar les diferents funcions necessàries i els components de programari i implementar els fluxos de dades i de control entre aquests components.

L'arquitectura del programari es descompondrà en unitats de menor complexitat i sovint s'adopta una metodologia de tipus *top-down*.

Per a cada element o component del programari s'hauran indicat, durant la fase de disseny, les dades d'entrada, les funcions que realitzarà i les dades de sortida.

Com a resultat de la fase de desenvolupament, o programació, s'obté, a part de codi del programa, el document de disseny detallat i el manual d'usuari del programari.

Tot mòdul desenvolupat serà objecte, posteriorment, d'un test d'integració.

Per a saber-ne més

Per a tenir més informació sobre els tests d'integració, podeu consultar l'apartat "Verificació i proves" d'aquest mateix mòdul.

El programari codificat i validat està preparat per a entrar en l'etapa de transferència, essent instal·lat sobre la plataforma (maquinari) final, per a poder dur a terme els tests d'acceptació especificats.

La fase de desenvolupament i integració és de les més complexes, tal com ja s'ha vist per la diversitat de rols de l'equip de treball que hi poden participar.

Per a saber-ne més

Per a tenir més informació sobre la diversitat de rols de l'equip podeu consultar l'apartat "L'equip de treball" d'aquest mateix mòdul.

L'objectiu d'aquest material no és descriure ni aprofundir en el desenvolupament d'aplicacions sinó simplement indicar la seva importància dins del projecte.

Per a saber-ne més

Per a tenir més informació sobre el desenvolupament d'aplicacions podeu consultar la unitat "Desenvolupament de programari" d'aquest mateix mòdul.

3.3.5. Verificació i proves

En el procés de verificació i proves, a més d'assegurar que el producte o sistema compleix correctament les funcions que el client ha sol·licitat, cal comprovar que els requeriments establerts són els adequats.

Per aquest motiu, en aquesta etapa cal definir una estratègia de proves que ha de considerar:

- la planificació de la prova,
- el disseny de casos de prova,
- l'execució d'aquestes proves,
- i l'avaluació dels resultats.

La verificació d'un sistema informàtic té en compte tant les proves de programari com posteriorment les proves del sistema.

Hi ha, en el mercat, a més a més, un conjunt d'eines i aplicacions que faciliten la tasca de l'equip de proves.

Proves del programari

A l'hora de verificar i cercar possibles errors en el programari desenvolupat hi ha dues tècniques de verificació:

- **Proves de caixa blanca**: comproven que els elements del programari encaixen correctament i que els procediments interns compleixen les especificacions. Se centren en l'estructura de control del programa, intenten cercar errors en el codi. Es posen a prova els camins independents de cada mòdul, les iteracions, les estructures internes de dades i les decisions lògiques.

- **Proves de caixa negra**: comproven que les funcions que s'executen en el programari són operatives i serveixen per a aconseguir la funció per a la qual han estat desenvolupades. Confirmen que el programari respon bé a les entrades, tant vàlides com no, que les sortides són correctes i que les funcions són operatives. Intenten trobar tant errades funcionals de parts que no compleixen les especificacions inicials com errades en funcions i estructures de dades incorrectes, errors en l'accés a bases de dades externes, en la interfície, de rendiment...

Per tal de simplificar i garantir el procés de proves, s'inicien en els nivells més bàsics, els mòduls. Un cop verificat el funcionament correcte dels mòduls, es comprova el funcionament conjunt dels diferents mòduls que componen una part del sistema i, finalment, es verifica el correcte funcionament de tot el sistema. D'aquesta manera, les proves es van duent a terme a partir de la integració dels mòduls i parts del sistema.

D'aquesta manera, la localització dels errors és més fàcil.

Els passos que se segueixen en el procés de prova del programari són els següents:

- Prova de les funcions i mòduls. S'utilitzen proves de caixa blanca, examinant, entre d'altres, la interfície del mòdul, les estructures de dades locals i les seves funcionalitats. Les porten a terme els membres de l'equip de desenvolupament.

- Proves d'integració. En aquestes proves s'utilitzen les tècniques de caixa blanca i caixa negra. Es comprova que cada mòdul funciona bé amb els altres, garantint que el conjunt manté les funcionalitats previstes. Les duen a terme els membres de l'equip de desenvolupament, però hi poden intervenir els usuaris i altres persones expertes en les funcionalitats demanades a les especificacions.

Un cop realitzades les proves de funcionalitat i integració del programari, es realitzen dues proves més: la prova alfa i la prova beta.

- Prova alfa: l'usuari final del programari el prova al mateix lloc on s'ha desenvolupat; en un entorn, per tant, controlat.

- Prova beta: l'usuari prova el programari al lloc on serà utilitzat.

Durant tot el procés de verificació i proves es van detectant errors que s'han de corregir i tornar a provar; per tant, és un procés amb un alt contingut de realimentació i molt delicat, ja que les solucions a alguns errors poden ocasionar errors en funcions ja provades amb èxit i provocar el desconcert i desencant dels futurs propietaris del programari.

Beta tester

Terme anglès que es traduiria per "verificador de beta". S'utilitza per a referir-se a aquelles persones que s'encarreguen de provar el funcionament d'un programa que encara no ha sortit al mercat (es troba en versió beta) amb l'objectiu de trobar errors de funcionament. D'aquests errors se n'informa l'empresa autora del programa que els corregirà perquè la versió definitiva sigui tan perfecta com sigui possible.

Per tal de verificar el correcte desenvolupament i la funcionalitat del programari hi ha dues tècniques: la caixa blanca i la caixa negra.

Del procés de verificació també se'n diu, en argot informàtic, depurar.

Depurar

Paraula utilitzada pels programadors per a referir-se a l'acció de revisar un programa en desenvolupament amb l'objectiu de trobar i eliminar possibles errors de funcionament

que hi pugui haver (coneguts per *bugs*). La depuració també persegueix l'optimització del programa perquè la seva funcionalitat i velocitat siguin com més bones millor. La depuració normalment es realitza amb l'ajuda de programes especialitzats en aquesta tasca, cosa que facilita el treball del programador i escurça el temps fet servir en la fase de depuració. Per a evitar els errors en el funcionament dels programes, se solen llançar diverses versions beta de cada programa, que proven els especialistes.

Proves del sistema

Finalment, un cop verificat el desenvolupament correcte del programari, cal fer les proves del sistema.

En aquest moment, el programari s'integra amb la resta del sistema (maquinari, servidors...).

En aquestes proves hi participen els responsables de desenvolupament de cada element i part del sistema.

Les principals proves del sistema que es porten a terme són les següents:

- **Proves de recuperació**. Avaluen si la recuperació de fallides és l'apropiada. És a dir, es comprova si la reinicialització, els mecanismes de recuperació d'estat del sistema, la recuperació de dades, etc. són els adequats.

- **Proves de seguretat**. Tracten de garantir que el sistema es troba protegit contra robatoris de clau d'accés, atacs amb programari, bloqueigs del sistema, errors provocats per intents d'atac al sistema, etc.

- **Proves de resistència**. Intenten garantir que el sistema té un límits correctes per a aguantar en condicions extremes, per exemple, l'accés concurrent de molts usuaris al lloc web, la disponibilitat de memòria suficient en el cas d'execució de molts processos o que l'amplada de banda de la xarxa permet garantir l'accés i el treball correctes dels usuaris.

Les proves del sistema es divideixen en recuperació, seguretat i resistència.

Eines de suport

Per tal que el procés de verificació sigui com més ràpid millor, hi ha un conjunt d'eines en el mercat que permeten disminuir l'esforç i el cost d'aquesta fase.

Aquests programes es poden classificar en funció del tipus d'ajuda que proporcionen:

- **Generadors de dades de prova**. Produeixen dades que fan que els programes i aplicacions que s'han de verificar es comportin d'una manera determinada.

- **Comparador d'arxius**. Treballen comparant diferents conjunts de dades de sortides de diverses proves.

- **Simuladors**. Permeten imitar l'entorn real del programari i sistema informàtic que s'està verificant.

Hi ha diverses eines al mercat per a ajudar l'equip de proves i assegurament de la qualitat del sistema. Aquestes eines es poden classificar en generadors de dades de prova, comparadors d'arxiu i simuladors.

3.3.6. Operació i manteniment

Molts equips de treball tendeixen a creure que la seva missió ha acabat quan el programari, i el sistema informàtic en general, està instal·lat i funcionant en les instal·lacions del client. Però en la pràctica poques vegades és així.

Les peculiaritats d'un sistema informàtic, atesa la complexitat de tots els elements que hi intervenen (vegeu "Projectes informàtics") fan que calgui supervisar-ne el funcionament correcte durant bastant de temps després d'haver estat lliurat.

Un requisit de disponibilitat o de fiabilitat del sistema, per exemple, és molt difícil garantir que el programa es comportarà adequadament en circumstàncies no previstes o passat un temps determinat, per la qual cosa els tests d'acceptació solen incloure, a petició del client, proves a llarg termini del programari.

Aquesta fase de supervisió (i correcció dels vicis o defectes del producte) rep el nom de *fase d'operació*.

Només després que hagi transcorregut aquest interval, el client accepta definitivament el programari, provisionalment acceptat en acabar la fase de transferència.

Al cap del temps, el programari definitivament acceptat es pot haver de modificar, bé com a conseqüència d'errors no detectats bé com a resposta davant noves necessitats de l'usuari.

Aquesta fase següent rep el nom de fase de manteniment.

Durant la fase d'operació i manteniment, es genera i actualitza el document d'història del projecte, que recull tots els errors i totes les modificacions realitzades sobre el programari i permet calcular i analitzar la fiabilitat del programari i el rendiment de l'equip de treball (per a futurs projectes).

Un cop implantat el sistema, cal iniciar una fase de supervisió, anomenada *d'operació*, durant les primeres setmanes o mesos, iniciant-se posteriorment la fase de manteniment si cal fer modificacions o adaptacions a la solució.

3.3.7. Documentació

Durant el cicle de vida de qualsevol projecte es genera molta documentació relativa, de tipus tècnic, de gestió o documentació complementaria.

La gestió correcta de tota aquesta documentació és vital, no solament pel seguiment del projecte, sinó també per a garantir una informació històrica, tant per a dur a terme futurs canvis i manteniment com també per a tenir informació en què basar-se per a nous projectes similars.

En projectes complexos, de llarga durada o on intervinguin equips nombrosos de persones (o diferents empreses). La seva gestió evita duplicar innecessàriament els documents generats i permet que els integrants de l'equip de treball sàpiguen quina documentació hi ha, quina és l'última versió, on es troba localitzada, etc.

La gestió de la documentació d'un projecte comporta la realització de les tasques següents:

- Redacció dels procediments de gestió de documentació i assignació de referències a documents.
- Assignació de referències als documents, a mesura que es generen.
- Inclusió dels nous documents a la base de dades (llistat) de documentació del projecte.
- Difusió a l'equip de treball.
- Incorporació de la base de dades del projecte a la base de dades general de l'empresa en finalitzar el projecte.

Alguns dels tipus més freqüents de documents d'un projecte són:

- Documentació tècnica. Requeriments tècnics i especificacions, documents de disseny, notes tècniques, plànols, diagrames, llistat de programes, manuals tècnics, d'instal·lació, d'usuari...

- Documents de gestió. Pressupostos i ofertes, minutes i accions, informes de situació, balanç d'hores i despeses...

- Documents complementaris. Llistes de documents, llistes de productes, presentacions, plans de qualitat, procediments, faxos, comunicacions internes, correus electrònics...

La gestió de la documentació, que en petits projectes sol ser part de la tasca del gestor del projecte, es pot delegar a altres persones en projectes complexos o on el volum de la documentació generada ho aconselli.

Una vegada tancat el projecte, la documentació es classificarà en reutilitzable i no reutilitzable.

La documentació reutilitzable s'incorporarà immediatament a l'arxiu de documentació de l'empresa, mentre que la no reutilitzable es pot destruir un cop transcorregut un període prudencial de temps (que inclogui garanties, serveis postvenda o períodes de possible contractació d'una continuació del projecte).

La gestió de la documentació d'un projecte comporta la realització d'un conjunt de tasques com són la redacció dels procediments de gestió de documentació i l'assignació de referències, la inclusió de nous documents a la base de dades de documentació del projecte, la seva difusió entre l'equip de treball i la incorporació de la base de dades del projecte a la base de dades general de l'empresa, quan acabi.

En tot projecte tenen un interès especial les ajudes automàtiques de què pugui disposar l'equip, entenent per això els programes informàtics destinats a facilitar la feina d'ajudar a construir programari.

Tal com ja hem comentat, aquestes ajudes poden facilitar una activitat d'anàlisi, disseny o implementació propis del procediment recomanat per la metodologia.

També poden ajudar a gestionar l'organització del projecte, en particular controlar el procés d'avenç i realització, i facilitar la tasca d'elaborar la documentació exigida per la metodologia.

4. Llicències de programari

La compra d'una llicència d'un producte o aplicació representa per al comprador el dret d'instal·lar i utilitzar un programa.

Per cada aplicació de programari que s'utilitza, el fabricant atorga una llicència d'ús que es documenta en el contracte de llicència d'usuari.

El programari està protegit per la llei de drets d'autor, que estableix que un producte no es pot copiar sense l'autorització del propietari dels drets d'autor.

Una llicència de programari aplica a qualsevol tipus de programa (vegeu l'apartat "Aplicacions del programari"): sistemes operatius, processador de textos, antivirus, servidor de correu, gestor de projectes...

Aquestes llicències es poden classificar, de manera simplificada, en tres tipus:

- Personal.
- Servidora.
- De client.

Una llicència de programari personal implica el dret d'instal·lar un programa en un ordinador "personal". Per exemple, la llicència d'ús d'un processador de textos o d'un gestor de fulls de càlcul.

De vegades la compra d'un programari personal inclou en la llicència la possibilitat d'instal·lar-lo en diversos ordinadors personals (2-3). Aquesta política és molt seguida pels antivirus, de manera que en comprar-ne un el pots instal·lar al teu ordinador personal, al portàtil i a l'ordinador de l'habitació del fill, per exemple.

En canvi, una llicència servidora correspon al dret d'instal·lar i poder utilitzar un programa que atendrà "peticions" d'altres programes "clients". Exemples de llicències servidores són les corresponents als servidors de bases de dades, servidors de correu, servidor d'accés a Internet...

Finalment, una llicència "client" correspon al dret perquè un ordinador accedeixi i treballi amb un programa que s'ha instal·lat com a servidor. D'aquesta manera, per a cada ordinador o usuari que accedirà al servidor de bases de dades o al servidor de correu, caldrà adquirir una llicència client corresponent.

En general, els clients potencials de les llicències servidor i client són empreses i grans corporacions. L'obtenció de llicències servidor i client va molt junt, sovint es compra una o dues llicències servidor i tantes llicències client com

ordinadors estigui previst que accedeixin a les dades del servidor. Per exemple, si es tractés d'un programa gestor de correu electrònic s'hauria de comprar una llicència servidor i tantes llicències client com comptes de correu es vulgui tenir.

L'ús il·legal d'un programa comporta un delicte i, com a tal, està penalitzat.

Hi ha diverses organitzacions, nacionals i internacionals, encarregades de perseguir la pirateria informàtica a les companyies i organismes.

Les llicències es poden classificar de manera simplificada en personals, de servidor i de client.

4.1. Polítiques de compra i ús de llicències

Cada fabricant té una política de venda de llicències diferents i particulars.

Aquestes polítiques, a més a més, evolucionen amb el temps, adaptant-se a les noves característiques de les tecnologies de la informació (TI).

Així, l'aparició d'aplicacions Internet va comportar l'aparició no solament d'un canal nou de venda i distribució, sinó també de noves polítiques de compra i ús de llicències com, per exemple, les aplicacions web.

Aplicacions web

Una aplicació web està instal·lada en un proveïdor d'Internet, de manera que el client no és el propietari d'aquesta aplicació, sinó que en lloga l'ús per a dur a terme alguns processos que requereixen aquesta solució, de manera anàloga al lloguer d'un cotxe per a fer un viatge.

Alguns exemples són els següents:

- Microsoft office en línia: es poden crear, editar i guardar documents d'MS Office (Excel, Word, Access, PowerPoint, etc.) sense tenir instal·lat l'MS Office. Per fer-ho cal registrar-se al web que MS té dedicat a aquest servei.
- Programes convertidors d'arxius: es poden trobar a la xarxa llocs web dedicats a la conversió d'arxius lliures de taxes, per exemple per convertir documents de Word, imatges, etc. a pdf.
- Antivirus en línia: la majoria d'empreses del sector tenen una part de la seva web dedicada a la detecció de virus en línia. D'aquesta manera, des d'allà i sense necessitat de comprar una llicència d'antivirus hi ha la possibilitat d'escanejar i detectar virus en viu. La majoria d'aquests serveis, però, només t'informen de l'existència del virus al teu PC, per eliminar-lo cal comprar la llicència.

Un cop decidida la necessitat d'adquirir un programa, cal seguir tres passos bàsics per a comprar les llicències de programari:

a) Determinar la llicència necessària (tipus de programari i quantitat).
b) Decidir la millor opció entre comprar o lísing (*leasing*).

c) Determinar on realitzar la compra.

Per a determinar la llicència necessària cal tenir en compte les dades o categories següents:

- **Tipus de producte**. Alguns fabricants classifiquen els seus programes segons aquests tres tipus:
  - **Aplicacions**. Programes genèrics de productivitat personal, utilitats o eines de desenvolupament.
  - **Sistemes**. Corresponen als sistemes operatius.
  - **Servidors**. Emmarca totes les aplicacions corporatives o servidors corporatius, com per exemple, els servidors de bases de dades, d'Internet o de correu.
- **El producte**. El programa en si, segons les funcionalitats requerides. Per exemple, un processador de textos, un antivirus...
- **Versió**. Un mateix producte pot tenir diferents versions, tant per a l'evolució dels sistemes operatius en què es poden instal·lar i executar com per a disposar de més o menys funcionalitats i prestacions...
- **Edició**. Especifica el conjunt de prestacions i/o aplicacions incloses en el producte.
  Un producte pot tenir, per exemple, les edicions estàndard, professional i expert.
- **Tipus de llicència.** En funció del fabricant n'hi pot haver diferents tipus com, per exemple, la llicència de nou usuari, corresponent a la compra inicial del producte, l'actualització de versió, que permet l'adquisició d'una versió posterior del programa si es disposa de la llicència de la versió anterior del mateix producte, o l'actualització competitiva, que correspon a la compra d'una llicència de "cost menor" d'un programa, disposant d'una llicència d'un producte relacionat d'un altre fabricant.

Tot i així, això són exemples genèrics de tipus de llicències, ja que cada fabricant té la seva política específica de compra i ús dels seus programes.

Alguns fabricants tenen una política de venda relacionada amb l'ús que el client fa de l'eina, i cobren, per exemple, una comissió per cada transacció feta (plataformes d'*e-commerce*) amb l'aplicació. Aquest darrer tipus d'ús de llicències sol correspondre a sistemes de comerç electrònic, en què el comprador abona un import base (cost fix) per l'eina i una comissió (cost variable) per cada transacció que fa.

Hi ha opcions molt econòmiques que permeten a les empreses disposar de programari d'alta qualitat sense necessitat de comprar el programari ni el maquinari. Així, només es paga per la llicència, i tant el programari com el maquinari són físicament a casa del fabricant, el que es fa és reservar una part dels recursos del maquinari i configurar el programari segons les necessitats del client. Acostumen a ser aplicacions web, de manera que hi pot accedir amb un navegador des de la seva empresa i en fa el mateix ús que si fos una aplicació instal·lada al seu propi maquinari. Un exemple d'aquest cas és el Siebel, una eina CRM molt potent i amb un cost econòmic molt important. La seva versió Siebel *on demand* funciona igual com s'ha descrit i qualsevol pime pot accedir-hi, una eina CRM reservada fins ara a corporacions amb un gran volum de negoci.

Hi ha diferents opcions de compra del programari:

- **OEM (*original equipment manufactures*).** Correspon al programari inclòs en la compra d'un ordinador o un equip informàtic nous. El cas més comú és quan, en comprar un ordinador personal, el fabricant hi inclou la llicència del sistema operatiu i un conjunt d'aplicacions de productivitat personal: processador de textos, full de càlcul, eina de dibuix, etc.
  El programari OEM és una versió especial del programari que s'ha de distribuir de manera preinstal·lada al disc dur del PC.
  Mai no es pot distribuir programari per OEM sense el PC o el maquinari corresponent.

- **Compra del paquet.** Correspon a la compra física de la caixa amb el producte en qualsevol botiga o magatzem.
  Inclou el producte en format magnètic, per exemple, CD o DVD juntament amb les llicències corresponents.

- **Llicència per volum per a organitzacions.** Aquest tipus de compra correspon a les realitzades per les companyies que necessiten una gran quantitat de llicències del mateix programari.

Exemples

Per exemple, una organització de 200 treballadors pot necessitar, per exemple, adquirir 110 llicències d'un processador de textos. En aquests casos, en lloc d'adquirir els 110 CD o DVD del producte, es compren dos o tres CD i les 110 llicències (el dret d'instal·lar el producte a 110 màquines). Això redueix el cost del producte i els costos d'enviament.
La majoria de fabricants apliquen diferents polítiques de preu depenent del nombre de llicències adquirides o del tipus d'organització (administració pública, institucions acadèmiques...).

Per exemple, Microsoft distingeix entre la venda a petites i mitjanes empreses (pimes) amb l'Open Multillicència (adquisició de llicència perpètua) o l'Open Subscripció (llicències no perpètues) i la venda a grans empreses, amb el Contracte Select o Enterprise Agreement Suscription.

Un cop decidida la necessitat d'adquirir un programa, cal determinar la llicència necessària (tipus de programari i quantitat), decidir la millor opció entre comprar o lísing i determinar on cal fer la compra.

4.2. Llicències de programari gratuït (*freeware*) i programari de prova (*shareware*)

En els apartats anteriors s'ha introduït el concepte de programari i les seves aplicacions.

Una solució informàtica és un "programa" que s'instal·la en un ordinador o maquinari per a dur a terme certes tasques, sia per a elaborar i editar documents, gestionar bases de dades, oferir connectivitat entre ordinadors o gestionar projectes.

Utilitzar aquests programes implica tenir-ne una o més llicències.

S'han comentat diferents polítiques de compra i ús de llicències de programari.

Ara bé, hi ha dos "tipus de llicències" no considerades en els apartats anteriors i que cada vegada tenen més importància i són més usuals: les llicències de programari gratuït (*freeware*) i les llicències de programari de prova (*shareware*).

Segons la disponibilitat dels arxius font, el programari es pot classificar en obert (lliure, de domini públic i semilliure) i tancat (gratuït, de prova, de demostració i propietari).

Segons el cost que representa per a l'usuari el programari es pot classificar en gratuït (lliure, de domini públic, semilliure i gratuït) i en programari no gratuït (de prova, de demostració i propietari).

- El programari **gratuït** es pot utilitzar, copiar i distribuir lliurement, però no modificar, ja que no inclou el codi font. Per a l'FSF (Free Programari Foundation), entitat que promou l'ús i desenvolupament de programari d'aquest tipus, el programari gratuït no és lliure, tot i que tampoc el qualifica com a semilliure ni com a propietari. És un programari que no s'ha de pagar per a utilitzar-lo.

- El programari **de prova** es caracteritza perquè és de lliure distribució o còpia, de manera que es pot utilitzar, comptant amb el permís de l'autor, durant un període limitat de temps, havent de pagar, passat aquest temps, per a continuar utilitzant-lo, tot i que l'obligació és només de tipus moral, ja que els autors lliuren els programes confiant en l'honestedat dels usuaris. Aquest programari s'acostuma a distribuir sense el codi font i no

es permet modificar-lo ni redistribuir-lo. Moltes vegades, per ignorància, els programes d'aquesta classe se solen utilitzar de manera il·legal. Sovint es confon el programari de prova amb el programari de demostració, que són programes no 100% funcionals o que deixen de treballar després d'un temps determinat.

Segons la disponibilitat dels arxius font, el programari es pot classificar en obert (lliure, de domini públic i semilliure) i tancat (gratuït, de prova, de demostració i propietari i segons el cost que representa per a l'usuari, es pot classificar en gratuït (lliure, de domini públic, semilliure i gratuït) i en programari no gratuït (de prova, de demostració i propietari).

Hi ha casos que estan en un règim a cavall entre els esmentats, i es permet l'ús del programari fins al moment en què s'utilitza per a tasques lucratives. En aquest punt s'ha de pagar llicència. És a dir, és programari gratuït (*freeware*) fins al moment en què s'usa per treure'n un rendiment econòmic, aleshores cal pagar llicència. Per exemple, la base de dades Oracle.

No cal confondre, però, les llicències de programari gratuït amb el codi font obert o *open source*.

Open source

Nom que es dóna al programari que es pot distribuir lliurement amb l'única condició que sempre es doni el codi font juntament amb l'executable. En cas que el codi contingui errors de programació l'usuari els pot arreglar. Segons quina sigui la llicència de codi font obert, l'usuari també té el permís de distribuir la versió del programa que hagi modificat. Els sistemes operatius Linux o el paquet ofimàtic Open Office són el paradigma del programari obert (*open source*).

Exemples d'aplicació gratuïta

Exemples d'aplicacions gratuïtes són els següents:

- Paquets ofimàtics, amb processador de textos, fulls de càlcul, etc.: OpenOffice.org, Star Office i els gestors de bases de dades DBMaker 4.03 i TreeDBNotes Free.
- Sistemes operatius: OPTIX IV, BeOs i Linux MandrakeSoft.
- Gestors de correu electrònic: FoxMail 4, Desktop Sidebar i Cactus Mail.
- Programes de retoc fotogràfic: PhotoPlus i Photo Librarian Image Editor.
- Programes per a fer còpies de seguretat: WinDriversBackup i InFoAL BackUp.

Exemples d'aplicació de prova

Exemples d'aplicacions de prova són els següents:

- Gestors de bases de dades: SQLConnect 1.6.2.
- Programes de disseny gràfic: 3D Flash Animator, Paint Shop Pro i IntelliCAD 2001 Standard.
- Editor de pàgines web: Ace Expert, CoffeeCup HTML Editor i Web Design.
- Antivirus: Anti-Trojan i MC Afee VirusScan 14.4.0.
- Gestor de correu electrònic: Calypso.
- Programes de gestió de còpies de seguretat: mCopias, Magellan i Backup BaK Again II Workstation.

4.3. La pirateria de programari

El terme *pirateria de programari* cobreix diferents activitats, des de copiar programes il·legalment fins a falsificar i distribuir programari sense el consentiment o acord amb el fabricant corresponent.

4.3.1. Conseqüències de la pirateria del programari

La majoria de les persones pensen que la pirateria de programari només afecta la indústria informàtica.

En canvi, s'ha de tenir en compte que actualment (12/05/2009) el 42% del programari que s'utilitza a Espanya està copiat il·legalment. A escala mundial el percentatge és del 41%. Amb aquestes dades es pot establir l'abast real del problema.

Segons la BSA (Bussiness Software Alliance), la pirateria informàtica causa estralls no sols en la indústria del sector, sinó també en la societat. Als milions d'euros en pèrdues del sector hi cal afegir els milers de llocs de treball que es perden o no es generen a causa de la pirateria. Es poden trobar molts informes actuals referits a la pirateria informàtica des del punt de vista de la indústria del programari al web de la BSA.

Un estudi sobre pirateria de programari realitzat per International Planning & Research Corp., també durant l'any 2000, aquestes accions il·legals van representar la pèrdua de 118.026 llocs de treball als Estats Units.

La utilització il·legal del programa també comporta altres despeses i costos a les organitzacions que l'utilitzen.

Exemple costos derivats

Exemple d'aquests costos derivats són els següents:

- la manca de suport tècnic, per part del fabricant,
- l'accés a noves versions, actualitzacions, adaptacions i resolució d'errors (garantia postvenda),
- l'accés a programes complets amb totes les funcionalitats,
- disposar de programes lliures de virus.

Així mateix, la violació del Codi penal exposa les companyies a diverses accions judicials i al desprestigi i pèrdua de la seva imatge.

Resumint, es podrien classificar aquestes conseqüències segons tres tipus de destinataris: els consumidors, els desenvolupadors i els venedors.

Quan un usuari decideix fer una còpia no autoritzada d'un programa, està renunciant al dret a l'assistència, documentació, garanties i actualitzacions periòdiques.

Si la còpia il·legal del programari s'obté per mitjà de programes P2P el risc que el fitxer estigui infectat per algun virus és molt alt.

P2P o *peer 2 peer*

Són programes que posen en contacte dos ordinadors remots i es poden compartir fitxers. Són programes P2P l'eMule o el BitTorrent, per esmentar-ne dos dels més populars.

En piratejar un producte protegit per les lleis de propietat intel·lectual, l'individu s'exposa al risc legal que això implica.

La pèrdua d'ingressos que comporta la pirateria s'hauria pogut invertir en el producte, aconseguint, així, reduir el preu final per al consumidor. Amb aquestes accions il·legals es perjudiquen moltes i variades empreses que fan del seu mitjà de subsistència el disseny i desenvolupament de programari i la seva distribució i venda.

La pirateria de programari no solament afecta la indústria informàtica, sinó també els consumidors, els desenvolupadors i els venedors.

4.3.2. Formes de pirateria

Habitualment, quan es pensa i parla de pirateria de programari se n'identifiquen les còpies. Tot i que aquesta és una de les formes més esteses de pirateria, n'hi ha moltes altres:

- **Còpies realitzades per l'usuari final**. Són simples còpies sense llicència realitzades per usuaris individuals o empreses. També es poden considerar, en el cas de la compra de llicències per volum, una informació incorrecta sobre el nombre d'ordinadors en els quals s'ha instal·lat el programari.

- **Càrrega en el disc dur**. Tal com hem comentat en l'apartat anterior, sovint els fabricants d'ordinadors tenen contractes amb els fabricants de programari que els permeten distribuir programari per OEM (programari que s'ha de distribuir de manera preinstal·lada en el disc dur del PC). Un tipus molt habitual de pirateria el duen a terme fabricants de sistemes de PC que comercialitzen els ordinadors amb programari preinstal·lat, sense que aquest programari sigui legal. És a dir, sense haver signat un contracte amb el fabricant del programari per a distribuir-lo.

- **Falsificacions**. És la pirateria de programari a gran escala, en què es duplica il·legalment el programari i les seves caixes, dut a terme per xarxes organitzades que distribueixen els productes com a suposadament legals.

- **Distribució per canals fraudulents**. Molts fabricants de programari distribueixen llicències a col·lectius determinats com, per exemple, el col·lectiu educatiu, amb uns descomptes especials. D'altres vegades, per la compra de gran volum, els fabricants ofereixen uns descomptes al comprador (grans empreses, administracions públiques). Una casuística de pirateria correspon al programari distribuït com a llicències amb un descompte especial,

a clients amb un gran volum, a fabricants d'ordinadors o a entitats acadèmiques, que es redistribueix, posteriorment, a altres usuaris que no compleixen els requeriments per a disposar d'aquestes llicències.

- **Pirateria a Internet**. En aquest cas, correspon més al mitjà que al tipus. S'utilitza Internet per a la distribució il·legal del programari, de manera que es pot accedir a còpies dels diferents programes sense adquirir-los al fabricant o distribuïdors autoritzats. Segons informes de la BSA, dos dels grans canals de distribució de falsificacions i còpies il·legals de programari són els programes P2P i el portal de subhastes eBay. El tràfic de dades a Internet degut als programes P2P és tan alt que, segons fonts de la BSA, en alguns països arriba a situar-se entre el 49% i el 89% del tràfic total diürn a Internet i la xifra s'eleva fins al 95% de nit.

Altres exemples

Altres formes de pirateria poden ser, per exemple, instal·lar un producte original en diversos ordinadors sense haver adquirit el nombre de llicències necessari o instal·lar a l'ordinador personal un programari amb legal però adquirit per a utilitzar-lo en l'entorn de treball.

La pirateria de programari va més enllà del simple fet de copiar-lo il·legalment. Hi ha altres maneres de piratejar com la càrrega il·legal en el disc dur, les falsificacions o l'ús de més llicències que les adquirides legalment.

Exercicis d'autoavaluació

1. Classifiqueu els programes i les aplicacions següents seguint la taxonomia presentada en l'apartat "Aplicacions del programari".

Aplicació per al control de les comandes dels clients d'una empresa

- Simulador d'una constel·lació i/o del moviment dels planetes del sistema solar
- Sistema de gestió de projectes
- Programa de reconeixement de la veu
- Sistema operatiu
- Programa d'interfície de comunicacions entre un ordinador i un dispositiu mòbil
- Eina de suport del control aeri
- Sistema de visió artificial
- Aplicació d'agenda
- Eina de suport al disseny de peces
- Geogràfic i seguiment d'un paquet
- Analitzador dels components d'una solució química
- Sistema de control dels semàfors d'una ciutat
- Eina de posicionament

2. Ordeneu, cronològicament, els llenguatges presentats en l'apartat "Principals llenguatges de desenvolupament": Pascal , Eiffel , Cobol, Simula, C, Fortran, Basic, C++, LISP, Algol, RPG, Modula2, Ada, Prolog.

3. Classifiqueu els llenguatges presentats en les quatre categories següents:

Llenguatges d'aplicació en intel·ligència artificial	Llenguatges d'aplicacions científiques	Llenguatges orientats a objectes	Llenguatges de propòsit general	Generadors de programes

(Si creieu que un llenguatge pot estar en més d'una categoria, poseu-los-hi).

4. Classifiqueu els elements d'enginyeria del programari següents segons si són un mètode, una eina o un procediment:

- Redacció i generació d'informes
- Etapes del cicle de vida
- Procés de verificació d'una solució
- Generador del model de bases de dades
- Sistema de gestió de projectes
- Programació modular i programació estructurada
- Disseny del model conceptual d'una base de dades i generació del model físic
- Generador de formularis d'entrada de dades

5. Ateses les tasques següents, indiqueu quines creieu que podrien ser fites:

- Presentació informe detallat de l'anàlisi de requeriments
- Inici del projecte
- Redacció i generació d'informes
- Disseny del model conceptual de la base de dades
- Fitxer generat
- Generació del model físic de la base de dades
- Lliurament del model conceptual de la base de dades
- Mòdul validat pel client
- Inici fase manteniment
- Còpies de seguretat validades

6. Indiqueu per a cadascuna de les interrelacions entre tasques si creieu que són del tipus fi-inici, fi-fi, inici-inici o inici-fi:

- Lectura seqüencial d'un fitxer de dades – generació de la mitjana dels valors llegits.
- Validació disseny detallat – desenvolupament dels mòduls de programari.
- Trasllat material del magatzem del proveïdor a casa del client – vigilància del material del magatzem del proveïdor.

- Traducció del text d'un idioma a un altre – revisió ortogràfica i gramatical d'un text.

7. A partir de la descripció següent de la feina pendent de fer, identifiqueu les possibles tasques, les seves interrelacions i dibuixeu-ne el diagrama de Gantt.

"L'objectiu del projecte és la producció i distribució d'una nova línia de perfums de senyor perquè es presenti a la Fira Internacional del Perfum de París del proper any.

Per a poder dur a terme aquest projecte, cal fer un seguit de tasques:

- Fer un estudi de mercat (4 setmanes)
- Desenvolupar el disseny del producte i les especificacions de cost (2 setmanes + 1 setmana)
- Produir un prototipus (3 setmanes)
- Provar-lo (1 setmana)
- Coordinar la producció amb l'equip de fabricació (8 setmanes)
- Dissenyar i produir l'envàs (2 setmanes + 4 setmanes)
- Envasament del producte (1 setmana)
- Coordinar les campanyes de publicitat per al seu llançament al mercat (4 setmanes)

I per tal de controlar l'avançament del projecte, s'ha decidit posar els punts de control següents:

- Anàlisi de mercat complet
- Disseny i preu aprovats
- Prototipus finalitzat per a les proves
- Prototipus aprovat per a la producció
- Producte empaquetat i enviat al magatzem
- Producte a punt per a sortir al mercat

8. A partir de l'exercici anterior, identifiqueu el camí crític del projecte, indicant quines tasques formen part d'aquest camí crític i, de manera intuïtiva, indiqueu quin seria el marge de cadascuna de les tasques.

9. Associeu per a cadascuna de les tasques indicades quin rol de l'equip de treball dels esmentats les assumiria.

Tasques:

- Instal·lació dels sistemes operatius de les màquines i la seva configuració,
- Disseny de les icones del programa,
- Coordinació de l'equip d'enginyeria,
- Definició dels productes i tipologia per a una solució d'ofertes comercials,
- Disseny del pla de comunicació del llançament d'un nou web al mercat,
- Gestió i control de les hores i del pressupost del projecte,
- Anàlisi del procés de compravenda en un mercat virtual,
- Programació de les rutines de codi del programa,
- Instal·lació i configuració del servidor de bases de dades,
- Comprovació del compliment dels requeriments del disseny gràfic de la solució,
- Coordinació de l'equip d'assegurament de la qualitat.

Rols:

- Consultor de processos
- Consultor de catàlegs
- Consultor de màrqueting
- Dissenyador gràfic
- Director tècnic
- Enginyer de programari
- Administrador de bases de dades
- Administrador de sistemes
- Verificador de "Look & Feel"
- Responsable de qualitat
- Administrador del projecte

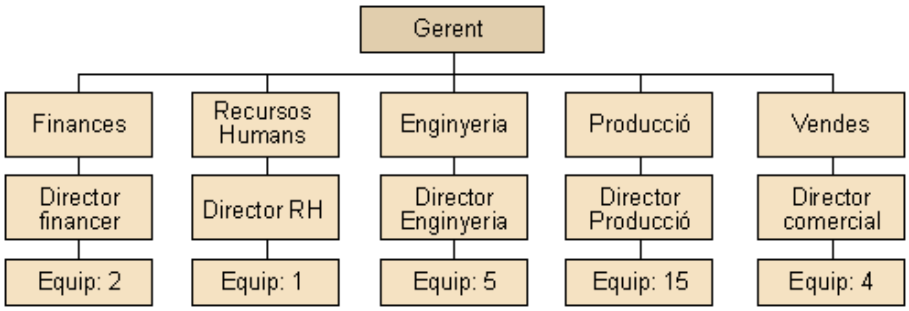
10. A continuació indiquem l'estructura organitzativa d'una empresa i el personal i els càrrecs de cada departament.

S'ha decidit utilitzar el sistema operatiu SOP1, que requereix llicència servidor i una llicència client per cada usuari que hi accedeixi, el processador de textos PTX1, que requereix una

llicència client per cada usuari que l'utilitzi. El mateix passa amb el programa de gestió de full de càlcul FCAL1 i el sistema de gestió de projectes SGP1 i el programa d'antivirus AVS1.

Així, s'adquirirà el sistema de gestió de missatgeria electrònica, SME1, que té els mateixos requeriments de llicències que el sistema operatiu SOP1.

Es demana identificar les llicències de servidor i client de cada aplicació, tenint en compte l'estructura organitzativa indicada i les necessitats funcionals de la companyia següents:



- Es disposarà d'un sol servidor de fitxers i impressores (sistema operatiu) en què també s'instal·larà el servidor de missatgeria electrònica.
- Tots els treballadors tenen accés a aquest servidor, i també correu electrònic.
- Només l'equip d'enginyeria, el director de producció i el gerent han de gestionar projectes.
- Tots els responsables de cada departament, el gerent i els membres dels departaments de Finances i RH han de treballar amb el processador de textos i el full de càlcul.
- Tothom ha de tenir instal·lat a la màquina el programa antivirus.

Solucionari

Exercicis d'autoavaluació

1. Aplicació per al control de les comandes dels clients d'una empresa.

- Programari de sistemes
  Sistema operatiu. Programa d'interfície de comunicacions entre un ordinador i un dispositiu mòbil
- Programari de temps real
  Eina de suport al control aeri. Sistema de control dels semàfors d'una ciutat. Eina de posicionament geogràfic i seguiment d'un paquet
- Eines d'ús personal
  Aplicació d'agenda. Eina de suport al disseny de peces
- Programari d'enginyeria i científic
  Analitzador dels components d'una solució química. Simulador d'una constel·lació i/o del moviment dels planetes del sistema solar
- Eines de gestió i redisseny de processos organitzatius
  Sistema de gestió de projectes. Aplicació per al control de les comandes dels clients d'una empresa
- Aplicacions d'intel·ligència artificial
  Programa de reconeixement de la veu. Sistema de visió artificial

2.

Fortran (1957)
LISP (1959)
Cobol (1960)
Simula (1962)
RPG (Dècada 60')
Algol (Dècada 60')
Basic (1965)
Pascal (1970)
C (1972)
Prolog (Dècada 70')
Modula2 (Finals 70')
C++ (Dècada 80')
Ada (1983)
Eiffel (1986)

3.

Llenguatges d'aplicació en intel·ligència artificial	Llenguatges d'aplicacions científiques	Llenguatges orientats a objectes	Llenguatges de propòsit general	Generadors de programes
Prolog, LISP, "Simula"	Fortran, "Algol", PL1	C++, Simula, Smalltalk, Eiffel	Cobol, Algol, PL1, Basic, C, Pascal, Modula2, Ada	RPG

4.

Mètode	Etapes del cicle de vida, programació modular i programació estructurada, disseny del model conceptual d'una base de dades i generació del model físic.
Eina	Generador del model de bases de dades, sistema de gestió de projectes, generador de formularis d'entrada de dades.
Procediment	Redacció i generació d'informes, procés de verificació d'una solució.

5. Tot i que dependria de cada projecte i del significat de la tasca, i de l'esforç necessari per a dur-la a terme, en aquell projecte, es podrien considerar fites, de manera general. Les tasques següents:

- Lliurament del model conceptual de la base de dades
- Mòdul validat pel client
- Presentació informe detallat de l'anàlisi de requeriments
- Inici del projecte
- Inici fase manteniment
- Fitxer generat
- Còpies de seguretat validades

Totes indiquen accions completades o "punts de control" dins del projecte.

En canvi, les tasques:

- Redacció i generació d'informes
- Disseny del model conceptual de la base de dades
- Generació del model físic de la base de dades

representen una "feina" a fer amb una durada per a completar-la.

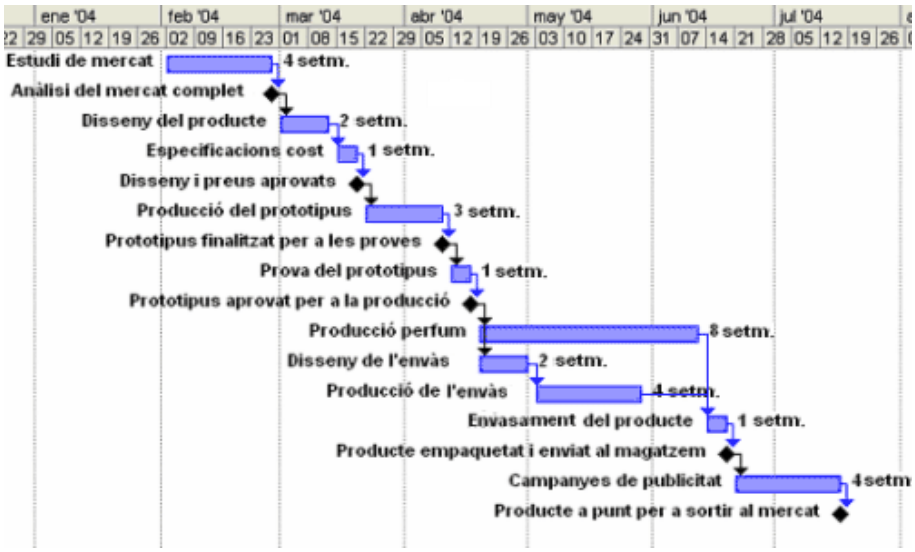
6.

- Validació disseny detallat – desenvolupament dels mòduls de programari (fi-inici)
  Atès que fins que no s'ha validat el disseny detallat de les funcionalitats, no es poden començar a desenvolupar els mòduls segons aquest disseny.
- Traducció del text d'un idioma a un altre - revisió ortogràfica i gramatical d'un text (ini-ci-inici)
  Es poden fer les dues tasques en paral·lel, de manera que quan comenci la traducció, també pot començar la revisió.
  En aquest cas, també es podrien seqüenciar les tasques, de manera que quan acabi la traducció, s'iniciï la correcció.
- Lectura seqüencial d'un fitxer de dades – generació de la mitjana dels valors llegits (fi-fi)
  Considerant que a mesura que es va llegint el fitxer de dades, es pot anar calculant la mitjana dels valors llegits fins al moment, hi hauria una relació fi-fi entre totes dues tasques. També es podria considerar una relació fi-inici, si la mitjana no es calculés fins al final de la lectura de tot el fitxer, en lloc de calcular-se després de llegir cada valor.
- Trasllat de material del magatzem del proveïdor a casa del client – vigilància del material del magatzem del proveïdor (inici-fi)
  En el moment en què s'inicia el trasllat del material del magatzem del proveïdor al client, s'atura la tasca de vigilància del material que hi ha al magatzem (inici-fi).

7. Es descompon el projecte en tasques, s'intenta poder fer tasques en paral·lel, per tal d'avançar la feina (producció del perfum i disseny i producció de l'envàs):

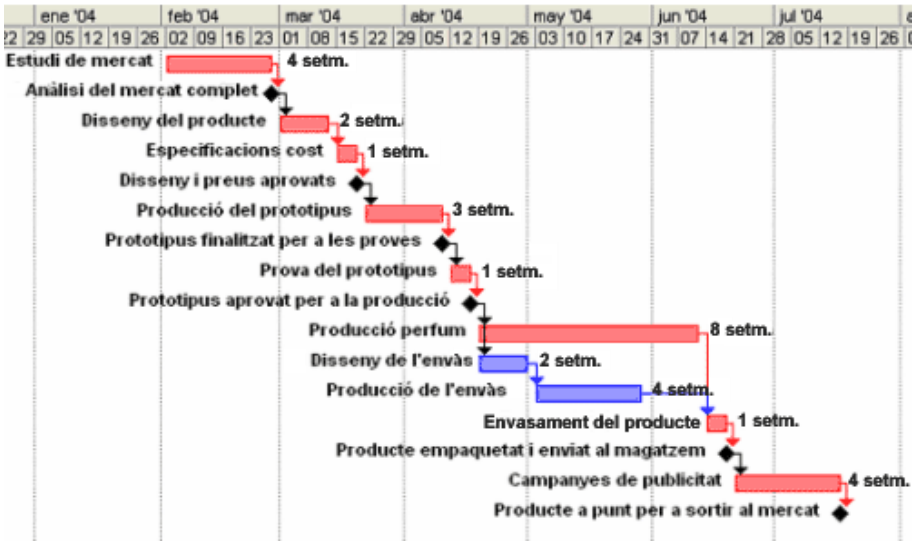
NUM TASCA	TASCA	DURADA	PREDECESSORA
1	Estudi de mercat	4 setm.	
2	Anàlisi del mercat complet	0 dies	1
3	Disseny del producte	2 setm.	2
4	Especificacions cost	1 setm.	3
5	Disseny i preus aprovats	0 dies	4
6	Producció del prototipus	3 setm.	5
7	Prototipus finalitzat per a les proves	0 dies	6
8	Prova del prototipus	1 setm.	7
9	Prototipus aprovat per a la producció	0 dies	8
10	Producció perfum	8 setm.	9
11	Disseny de l'envàs	2 dies	9
12	Producció de l'envàs	4 setm.	11
13	Envasament del producte	1 setm.	12;10
14	Producte empaquetat i enviat al magatzem	0 dies	13
15	Campanyes de publicitat	4 setm.	14
16	Producte a punt per a sortir al mercat	0 dies	15

El diagrama de Gantt tindria l'aspecte següent:



8. El camí crític del projecte està compost per a les tasques que, si s'endarrereixen, provocaran, segur, l'endarreriment de la data d'acabament del projecte.

En l'exemple, són totes les tasques excepte les tasques de disseny de l'envàs i de producció de l'envàs.



Per a avaluar els marges de les diferents tasques, cal centrar-se en les tasques que no formen part del camí crític.

Les tasques crítiques no tenen marge, ja que si alguna s'endarrerís, el projecte també s'endarreriria.

En canvi, les tasques de disseny de l'envàs i de producció de l'envàs tenen un marge d'endarreriment possible.

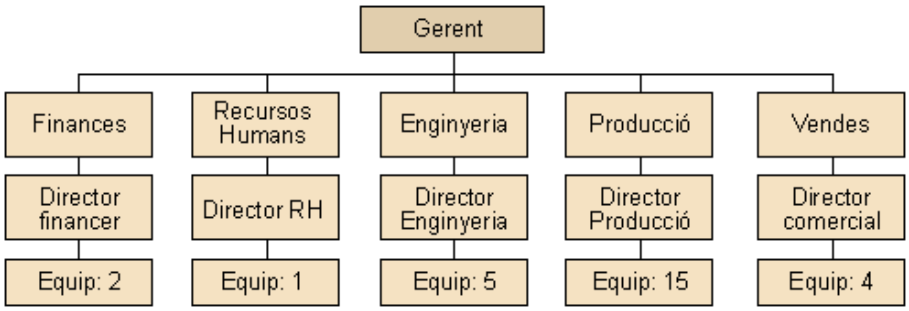
De manera intuïtiva, aquestes dues tasques (durada prevista 2+4 setmanes) es fan en paral·lel amb la producció del perfum (amb una durada prevista de vuit setmanes), per la qual cosa tenen un marge de dues setmanes de possible endarreriment. Si s'endarrereixen més de dues setmanes, poden afectar l'acabament del projecte.

9.

Rols	Tasques
Consultor de processos	Anàlisi del procés de compravenda en un mercat virtual

Rols	Tasques
Consultor de catàlegs	Definició dels productes i tipologia per a una solució d'ofertes comercials
Consultor de màrqueting	Disseny del pla de comunicació del llançament d'un web nou al mercat
Dissenyador gràfic	Disseny de les icones del programa
Director tècnic	Coordinació de l'equip d'enginyeria
Enginyer de programari	Programació de les rutines de codi del programa
Administrador de bases de dades	Instal·lació i configuració del servidor de bases de dades
Administrador de sistemes	Instal·lació dels sistemes operatius de les màquines i la seva configuració
Verificador de "Look & Feel"	Comprovació del compliment dels requeriments del disseny gràfic de la solució
Responsable de qualitat	Coordinació de l'equip d'assegurament de la qualitat
Administrador del projecte	Gestió i control de les hores i del pressupost del projecte

10.



Total treballadors companyia i per departament:

- Gerència: 1
- Finances: 1 + 2
- Recursos Humans: 1 + 1
- Enginyeria: 1 + 5
- Producció: 1 + 15
- Vendes: 1 + 4
- Total empresa: 33

Llicències requerides:

- Sistema operatiu (SOP1): 1 llicència servidor + 33 llicències client
- Sistema missatgeria electrònica (SME1): 1 llicència servidor + 33 llicències client
- Sistema de gestió de projecte (SGP1): 8 llicències
- Processador de textos (PTX1): 9 llicències
- Full de càlcul (FCAL1): 9 llicències
- Antivirus (AVS1): 33 llicències

Bibliografia

Barceló, M.; Costa, M.; Quer, C. (1993). *Anàlisi d'aplicacions informàtiques*. Barcelona: Edicions UPC.

Pressman, Roger S. (1997). *Ingeniería de Software. Un enfoque práctico*. Madrid: McGraw-Hill.

Drudis, Antonio (1999). *Gestión de Proyectos. Cómo planificarlos, organizarlos y dirigirlos*. Barcelona: Ediciones Gestión 2000.

Villarreal, Sonia (1999). *Introducción a la computación*. Mèxic DF: McGraw-Hill.

Domingo Ajenjo, Alberto (2000). *Dirección y gestión de proyectos*. Madrid: Editorial Ra-Ma.