AEA1 - Desenvolupament de programari

Part 1 - Programari informàtic

Paula Molina Agut
0487 - Entorns de desenvolupament (2n DAW)

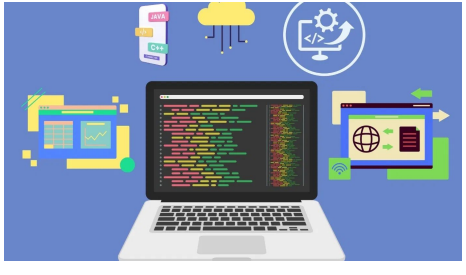
Tota aplicació informàtica, ja sigui utilitzada en un suport convencional (com un ordinador de sobretaula o un ordinador portàtil) o sigui utilitzada en un suport de nova generació (per exemple, dispositius mòbils com ara un telèfon mòbil de darrera generació o una tauleta tàctil PC), ha seguit un procediment planificat i desenvolupat detall per detall per a la seva creació. Aquest anirà des de la concepció de la idea o de la funcionalitat que haurà de satisfer aquesta aplicació fins a la generació d'un o diversos fitxers que permetin la seva execució exitosa.

Per convertir aquesta concepció d'una idea abstracta en un producte acabat que sigui eficaç i eficient hi haurà molts més passos, moltes tasques a fer. Aquestes tasques caldrà que estiguin ben planificades i que segueixin un guió que pot tenir en compte aspectes.

Concepte de programari

Un programa informàtic és un conjunt d'esdeveniments ordenats de manera que se succeeixen de forma seqüencial en el temps, un darrere l'altre.

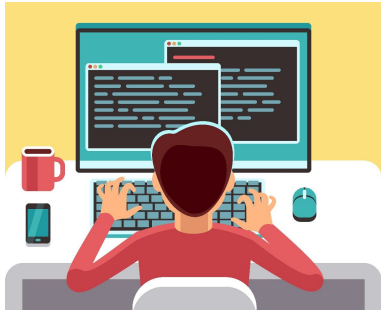
Aquest conjunt d'ordres no és arbitrari, sinó que serveix per dur a terme una tasca de certa complexitat que no es pot fer d'un sol cop. S'ha de fer pas per pas. Totes les ordres estan vinculades entre si per arribar a assolir aquest objectiu i, sobretot, és molt important la disposició en què es duen a terme.



Concepte de programari

Per exemple, el programa d'un robot de cuina per fer una crema de pastanaga seria:

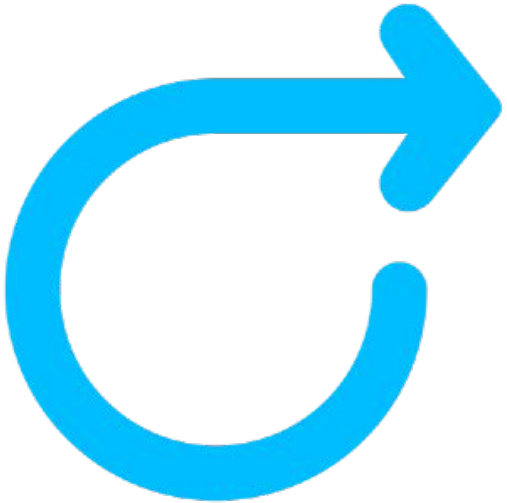
1. Espera que introduïu les pastanagues ben netejades, una patata i espècies al gust.
2. Gira durant 1 minut, avançant progressivament fins a la velocitat 5.
3. Espera que introduïu llet i sal.
4. Gira durant 30 segons a velocitat 7.
5. Gira durant 10 minuts a velocitat 3 mentre cou a una temperatura de 90 graus.
6. S'atura. La crema de pastanaga està llesta!



Concepte de programari

Conté dos elements principals:

- Dades
  - Entrada
  - Sortida
- Estructures de control
  - Condicionals
  - Bucles
  - Etc.



Concepte de programari

Un **algorisme** és:

- un mètode de resolució d'un tipus de problema en un nombre finit de passos.
- independent del llenguatge de programació que finalment s'utilitza per implementar el programa, per tant un programa és l'expressió d'un algorisme en un llenguatge de programació determinat

Concepte de programari

La tasca d'un/a programador/a informàtic/a és escollir **quines ordres** constituiran un programa d'ordinador, **en quin ordre** s'han de dur a terme i sobre **quines dades** cal aplicar-les perquè el programa porti a terme la tasca que ha de resoldre.

La dificultat de tot plegat serà més o menys gran depenent de la complexitat mateixa d'allò que cal que el programa faci.

Concepte de programari

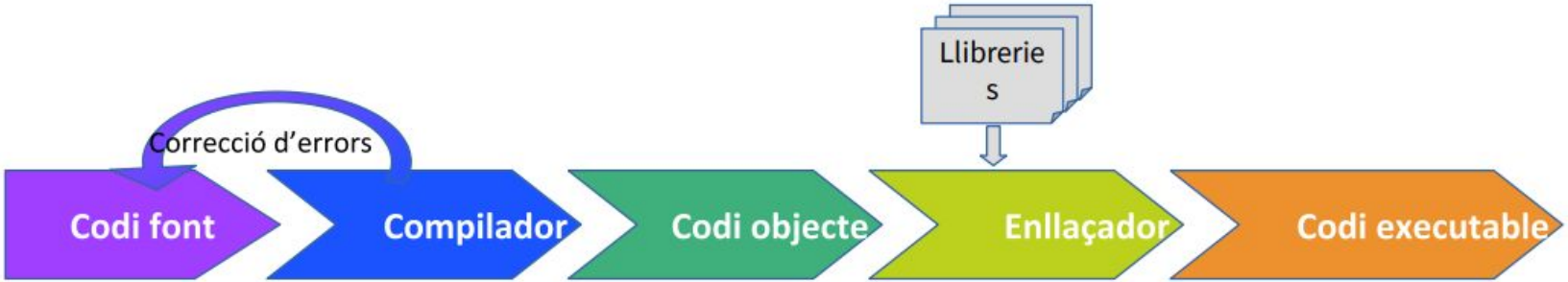
Interacció amb el sistema

No tots els programes tenen accés lliure i directe al maquinari, per això es classifiquen com:

- **Software de sistema**: que s'encarrega de controlar i gestionar el maquinari així com de gestionar el software d'aplicació i per tant exerceix de vincle entre ambdós.

- **Software d'aplicació**: programari que s'executa sobre el software de sistema per tal de relacionar-se amb el maquinari. Incorpora (gràcies al compilador) llibreries per tal d'entendre's amb el SO.

Obtenció del codi executable



Cal dir que els llenguatges interpretats no segueixen aquest cicle.

Codi font

Un cop s'ha acabat d'escriure el programa, el conjunt de fitxers de text resultants, on es troben les instruccions, es diu que contenen el **codi font**.

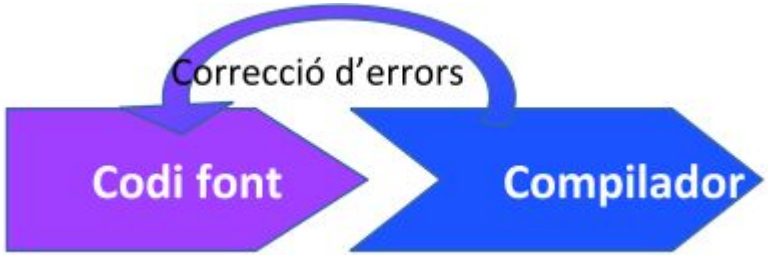
Aquest codi font pot ser des d'un nivell molt alt, molt a prop del llenguatge humà, fins a un de nivell més baix, més proper al codi de les màquines, com ara el codi assemblador.

Compilador

El compilador es un programa que acompanya un llenguatge de programació

- S'encarrega de comprovar que un codi font no conté errors lèxics, sintàctics o semàntics.

- Si el codi font és correcte, el compilador genera el codi objecte mitjançant el procés de compilació.



Codi objecte

El codi objecte és el codi font traduït (pel compilador) a codi màquina, però aquest codi encara no pot ser executat per l'ordinador.

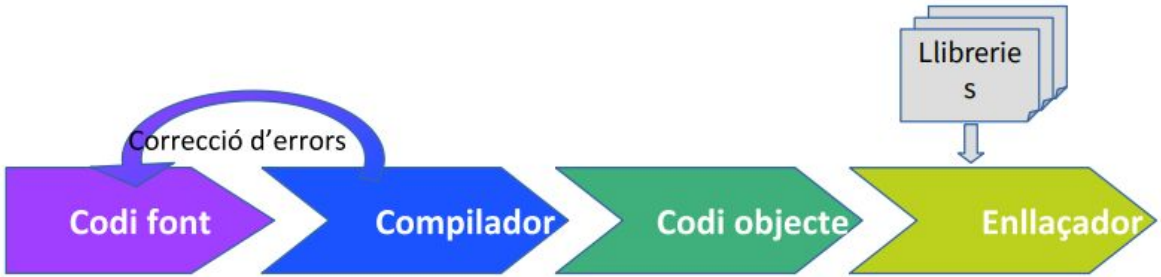
Pot ser en llenguatge màquina o bytecode, i pot distribuir-se en diversos arxius que corresponen a cada codi font compilat.

Enllaçador

L'enllaçador és un programa que acompanya un llenguatge de programació.

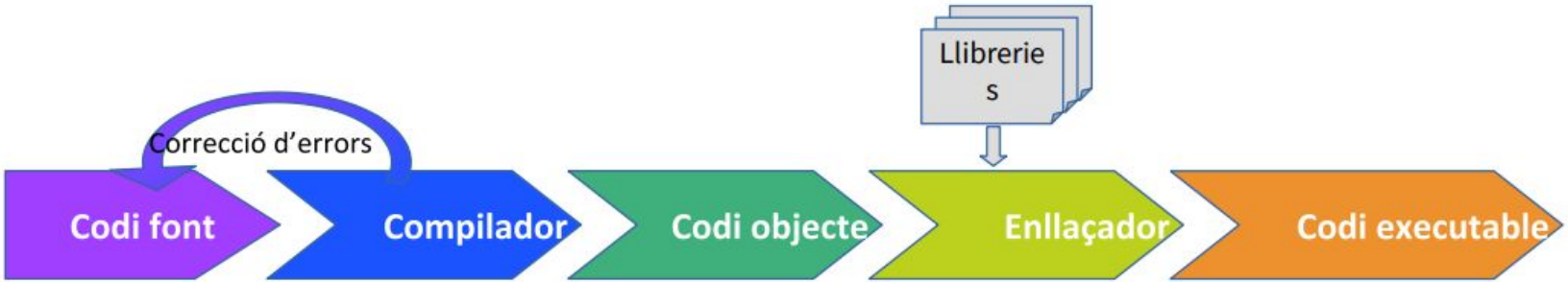
➔ S'encarrega d'enllaçar el codi objecte amb les llibreries externes.
➔ El resultat és el codi executable.

Una **llibreria** és un col·lecció de codi predefinit que facilita la tasca del programador a l'hora de codificar un programa.



Codi executable

El codi executable és la traducció completa a codi màquina, duta a terme per l'enllaçador. El codi executable és interpretat directament per l'ordinador.



Concepte de llenguatge de programació

Un llenguatge de programació és un llenguatge que permet establir una comunicació entre la persona i la màquina. El llenguatge de programació identificarà el codi font, que es desenvoluparà per indicar a la màquina, una vegada aquest codi s'hagi convertit en codi executable, quins passos ha de donar.

Per generació

Tipus de llenguatge: Generacions

Primera generació: Llenguatge Màquina

- ➔ Es donen instruccions directament al microprocessador fent servir codi màquina o binari.
- ➔ Un programador entrenat pot generar codi altament eficient.
- ➔ Difícil d'entendre i de mantenir, altament complex.
- ➔ Estretament lligat el hardware.
- ➔ Executat directament pel processador.
- ➔ **Exemple**: codi binari específic d'una CPU.

Tipus de llenguatge: Generacions

Segona generació: Llenguatges d'Assemblador

- ➔ Es donen instruccions directament al microprocessador fent servir codi assemblador.
- ➔ Cada instrucció en assemblador correspon a una instrucció en llenguatge màquina.
- ➔ Un programador entrenat pot generar codi altament eficient.
- ➔ Difícil d'entendre i de mantenir.
- ➔ Estretament lligat el hardware.
- ➔ **Exemple**: Assembly x86, MIPS.

Tipus de llenguatge: Generacions

Tercera generació: Llenguatges d'Alt Nivell

- ➔ Es fan servir frases senzilles per programar, resultant en un codi més llegible i entenedor.
- ➔ Es presenten estructures de control simples i intuïtives.
- ➔ Prioritat al manteniment per sobre del rendiment.
- ➔ Permet fer aplicacions altament grans i complexes.
- ➔ Fàcils d'aprendre.
- ➔ Portables entre diferents plataformes.
- ➔ Necessiten compilador o intèrpret.
- ➔ **Exemples**: C, Java, Python, Pascal.

Tipus de llenguatge: Generacions

Tercera generació: Llenguatges d'Alt Nivell

Màquines Virtuals

Les màquines virtuals (VM) són clau en l'ecosistema de llenguatges com Java i Python, compilant el codi a bytecode, un llenguatge intermedi que després la JVM (Màquina Virtual Java) o altres M, com la Python Virtual Machine, interpreten i executen en diferents sistemes operatius, assegurant portabilitat i aïllament.

- ➔ Les màquines virtuals permeten hibridar els llenguatges compilats i els interpretats.
- ➔ El compilador transforma el codi font a codi intermedi o bytecode, que la màquina virtual executa.

Tipus de llenguatge: Generacions

Quarta generació: Llenguatges Orientats a Dominis

- ➔ Encara més abstracció i productivitat.
- ➔ Menys codi per fer tasques complexes.
- ➔ Molt usats en bases de dades i aplicacions de gestió.
- ➔ **Exemples**: SQL, MATLAB, R, PowerBuilder.

Tipus de llenguatge: Generacions

Quinta generació: Llenguatges Basats en Coneixement i IA

- ➔ Orientats a la resolució de problemes.
- ➔ Utilitzen restriccions i lògica declarativa.
- ➔ Aplicats en IA i sistemes experts.
- ➔ **Exemples**: Prolog, Mercury.

Problema: Han quedat desbancats per els llenguatges moderns de 3ª i 4ª generació. Gairebé no es fan servir, ni tan sols al camp de la I.A.

Tipus de llenguatge: Generacions

Generació	Característiques principals	Exemples
1GL – Llenguatge màquina	Codi binari (0/1), depèn directament del processador	Codi binari específic d'una CPU
2GL – Llenguatge assemblador	Mnemotècnics (ADD, MOV…), encara depèn del maquinari	Assembly (x86, MIPS)
3GL – Llenguatges d'alt nivell	Sintaxi propera al llenguatge humà, portables, necessiten compilador/intèrpret	Python, C, Java, Pascal
4GL – Llenguatges orientats a dominis	Encara més abstracció, orientats a aplicacions i bases de dades	SQL, MATLAB, R
5GL – Llenguatges basats en coneixement i IA	Resolució de problemes mitjançant lògica i restriccions	Prolog, Mercury

Per paradigma

Tipus de llenguatge: Paradigma

Un paradigma de programació és un estil o metodologia que defineix com els desenvolupadors estructuren i aborden els problemes de codificació per a resoldre'ls.

No és un llenguatge de programació en si, sinó un conjunt de principis i tècniques que ofereixen una "filosofia" per a escriure codi.

Tipus de llenguatge: Paradigma

Imperatiu

- El paradigma imperatiu/estructurat deu el seu nom al paper dominant que exerceixen les sentències imperatives, és a dir aquelles que indiquen dur a terme una determinada operació que modifica les dades guardades en memòria.
- La tècnica seguida en la programació imperativa és la programació estructurada. La idea és que qualsevol programa, per complex i gran que sigui, pot ser representat mitjançant tres tipus d'estructures de control:
  - Seqüència.
  - Selecció.
  - Iteració.
- També es pot desenvolupar amb la tècnica de disseny descendent.

Tipus de llenguatge: Paradigma

Objectes

- ➔ Paradigma de construcció de programes basat en una abstracció del món real. En un programa orientat a objectes, l'abstracció no són procediments ni funcions sinó els objectes. Aquests objectes són una representació directa d'alguna cosa del món real, com ara un llibre, una persona, una comanda, un empleat...
- ➔ Un objecte és una combinació de dades (anomenades atributs) i mètodes (funcions i procediments) que ens permeten interactuar amb ell. En aquest tipus de programació, per tant, els programes són conjunts d'objectes que interactuen entre ells a través de missatges (crides a mètodes).

Tipus de llenguatge: Paradigma

Objectes

Cotxes que...
- Tenen una matrícula
- Arranquen
- Frenen

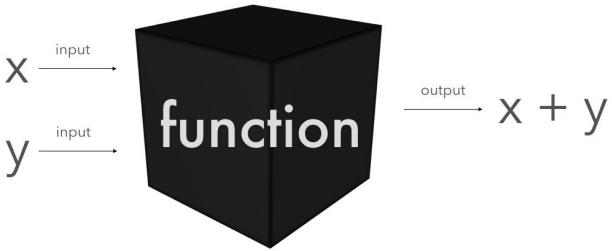
Persones que...
- Tenen nom
- Tenen cumple
- Parlen
- Dormen

Animals que...
- Son de un tipus
- Tenen una raça
- Fan soroll
- Mengen

Tipus de llenguatge: Paradigma

Funcional

- ➔ La idea és que el resultat d'un càlcul és l'entrada del següent, i així successivament fins que una composició produeix el resultat desitjat.
- ➔ S'ha utilitzat principalment en àmbits d'investigació científica i aplicacions matemàtiques.



Tipus de llenguatge: Paradigma

Lògic

- ➔ Té com a característica principal l'aplicació de les regles de la lògica per inferir conclusions a partir de dades.
- ➔ Un programa lògic conté una base de coneixement sobre la que es duen a terme consultes. La base de coneixement està formada per fets, que representen la informació del sistema expressada com a relacions entre les dades i regles lògiques que permeten deduir conseqüències a partir de combinacions entre els fets i, en general, altres regles.

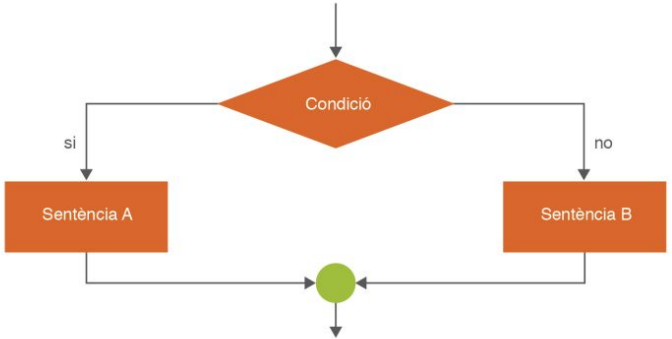
- R1: Si febre, llavors estar a casa en repòs.
- R2: Si malestar, llavors posar-se termòmetre.
- R3: Si termòmetre marca una temperatura > 37º, llavors febre.

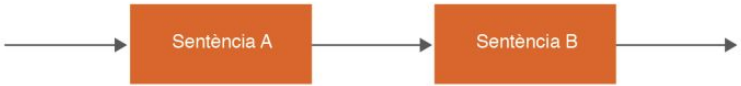
Característiques

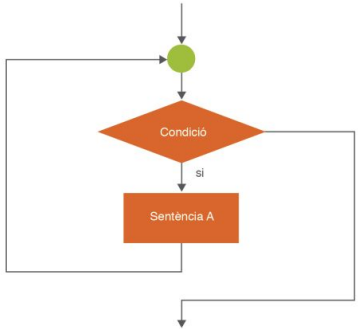
Característiques de tipos de programació

Programació estructurada:

- ➔ Claredat
- ➔ Teorema de l'estructura



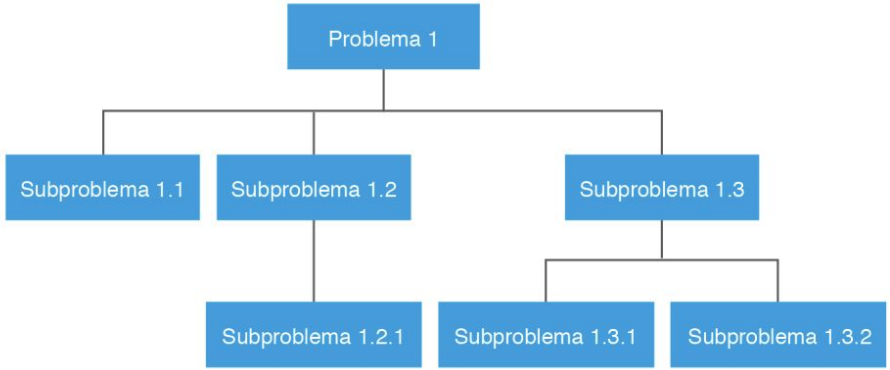




Característiques de la programació estructurada

Programació estructurada:

- ➔ Claredat
- ➔ Teorema de l'estructura
- ➔ Disseny descendent



Característiques de tipos de programació

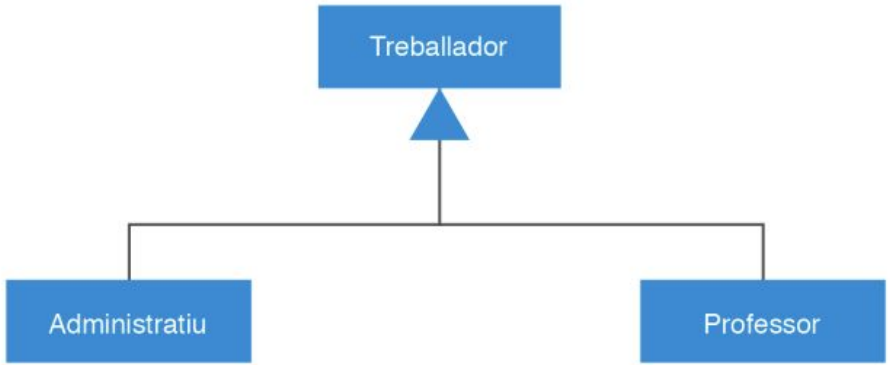
Programació estructurada:

- Claredat
- Teorema de l'estructura
- Disseny descendent
- Tipus abstractes de dades
- Programació modular
  - Procediments
  - Funcions

Característiques de la programació orientada a objectes

POO:

- ➔ Abstracció
- ➔ Encapsulació
- ➔ Modularitat
- ➔ Polimorfisme
- ➔ Jerarquia



Fases del desenvolupament del sistema d'informació

Una aplicació informàtica necessitarà moltes petites accions (i no tan petites) per ser creada. S'han desenvolupat moltes metodologies que ofereixen un acompanyament al llarg d'aquest desenvolupament, proporcionant pautes, indicacions, mètodes i documents per ajudar, sobretot, els caps de projecte més inexperts.

Mètrica v3.0

Mètrica v3.0

Es tracta d'una metodologia per a la planificació, desenvolupament i manteniment dels sistemes d'informació d'una organització. Per al desenvolupament de programari cal fixar-se en la part que fa referència al desenvolupament dels sistemes d'informació (SI), dins la metodologia Mètrica. Divideix el desenvolupament en 5 fases, que se segueixen de forma seqüencial.

És important tenir clarament identificats els rols dels components de l'equip de projecte que participaran en el desenvolupament de l'aplicació informàtica:

- ➔ Parts interessades (stakeholders)
- ➔ Cap de Projecte
- ➔ Consultors
- ➔ Analistes
- ➔ Programadors

Mètrica v3.0


Estudi de viabilitat del sistema
Anàlisi del SI
Disseny del SI
Construcció del SI
Implantació i acceptació del SI

Mètrica v3.0 - Estudi de viabilitat del sistema

El propòsit d'aquest procés és analitzar un conjunt concret de necessitats, amb la idea de proposar una solució a curt termini. Els criteris amb què es fa aquesta proposta no seran estratègics sinó tàctics i relacionats amb aspectes econòmics, tècnics, legals i operatius.

Els resultats de l'estudi de viabilitat del sistema constituiran la base per prendre la decisió de seguir endavant o abandonar el projecte.

Mètrica v3.0 - Anàlisi del sistema d'informació

L'objectiu és aconseguir l'especificació detallada del sistema d'informació, per mitjà d'un catàleg de **requisits** i d'una sèrie de models que cobreixin les necessitats d'informació dels usuaris per als quals es desenvoluparà el sistema d'informació i que seran l'entrada per al procés de Disseny del sistema d'informació.

En primer lloc, **es descriu el sistema d'informació, a partir de la informació obtinguda en l'estudi de viabilitat. Es delimita el seu abast, es genera un catàleg de requisits generals i es descriu el sistema mitjançant uns models inicials d'alt nivell.**

Per tal d'efectuar l'anàlisi se sol elaborar els models de **casos d'ús i de classes**, en desenvolupaments orientats a objectes, i **de dades** i processos en desenvolupaments estructurats. D'altra banda, s'aconsella dur a terme una definició d'interfícies d'usuari, ja que facilitarà la comunicació amb els usuaris clau.

Mètrica v3.0 - Disseny del sistema d'informació

El propòsit del disseny és obtenir la definició de l'arquitectura del sistema i de l'entorn tecnològic que li donarà suport, juntament amb l'especificació detallada dels components del sistema d'informació. A partir d'aquesta informació, es generen totes les especificacions de construcció relatives al propi sistema, així com l'especificació tècnica del pla de proves, la definició dels requisits d'implantació i el disseny dels procediments de migració i càrrega inicial.

Per exemple:

- ➔ Els components del sistema i estructures de dades
- ➔ Procediments i mètodes
- ➔ Excepcions
- ➔ Pla de proves
- ➔ Requisits d'implantació

Mètrica v3.0 - Construcció del sistema d'informació

Té com a objectiu final la construcció i la prova dels diferents components del sistema d'informació, a partir del seu conjunt d'especificacions lògiques i físiques, obtingut en la fase de disseny. Es desenvolupen els procediments d'operació i de seguretat, i s'elaboren els manuals d'usuari final i d'explotació, aquests últims quan sigui procedent.

Es recull la informació elaborada durant la fase de disseny, es prepara l'entorn de construcció, es genera el codi de cada un dels components del sistema d'informació i es van duent a terme, a mesura que es vagi finalitzant la construcció, les proves unitàries de cada un d'ells i les d'integració entre subsistemes.

Mètrica v3.0 - Implantació i acceptació del sistema

Té com a objectiu principal el **lliurament i l'acceptació del sistema en la seva totalitat**, que pot comprendre diversos sistemes d'informació desenvolupats de manera independent, i un segon objectiu, que és dur a terme les activitats oportunes per al pas a producció del sistema.

- ➔ En aquest procés s'elabora el pla de manteniment del sistema, de manera que el responsable del manteniment conegui el sistema abans que aquest passi a producció.
- ➔ També s'estableix l'acord de nivell de servei requerit una vegada que s'iniciï la producció.

Diagrama de Gantt

Definició

El diagrama de Gantt és una eina per a planificar i programar tasques al llarg d'un període determinat. Gràcies a una fàcil i còmoda visualització de les accions previstes, permet realitzar el seguiment i control del progrés de cadascuna de les etapes d'un projecte i, a més, reprodueix gràficament les tasques, la seva durada i seqüència, a més del calendari general del projecte.

Desenvolupat per Henry Laurence Gantt a inicis del segle XX, el diagrama es mostra en un gràfic de barres horitzontals ordenades per activitats a realitzar en seqüències de temps concretes.


Enero
Febrero
Marzo
Abril
Mayo
Junio
Julio
Tarea 1
Tarea 2
Tarea 3
Tarea 4
Tarea 5
Tarea 6
Tarea 7

Funcionament

En l'eix vertical se situen les tasques a realitzar des de l'inici fins a la fi del projecte, mentre en l'horitzontal es posen els temps.

En funció del tipus d'activitats que conformin el projecte, els valors situats en l'eix horitzontal han de definir-se en dies, setmanes, mesos, semestres o, fins i tot, anys.

En una etapa posterior, se li assigna a cada tasca un bloc rectangular que indiqui el seu grau de progrés i el temps restant per a la seva execució plena. Per a les tasques crítiques o estructurals del procés, el més recomanable és usar un color distint.

Passos per a elaborar el diagrama

1) El primer pas per a elaborar un diagrama de Gantt passa per fer una llista de totes les activitats que pot requerir un projecte. Pot ser que, com a resultat, obtinguem una llista massa llarga. Tanmateix, a partir d'això definirem temps per a la realització de cada tasca, prioritats i ordre de consecució. A més, agruparem les activitats per partides específiques per a simplificar al màxim la gràfica.

2) El disseny del diagrama de Gantt ha de ser el més esquemàtic possible. Ha de transmetre el més important, ja que serà consultat amb freqüència. Les persones implicades en el procés han de quedar-se amb una idea clara del que està succeint en un moment concret del procés.

Passos per a elaborar el diagrama

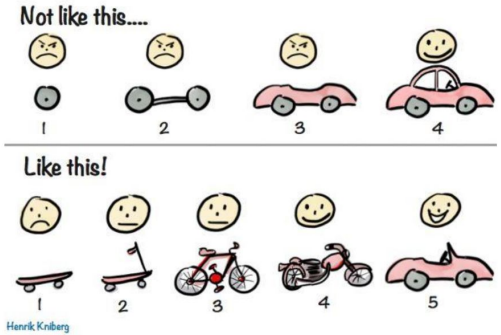
3) Si es desitja, es pot crear i mantenir actualitzada una altra versió més detallada per a la persona que executa el projecte. Gràcies al diagrama de Gantt, és possible un monitoratge clar del progrés per a descobrir amb facilitat els punts crítics, els períodes d'inactivitat i per a calcular els retards en l'execució. D'aquesta manera, ajuda a preveure possibles costos sobrevinguts i permet reprogramar les tasques d'acord amb les noves condicions.

Models de desenvolupament àgil

Concepte

L’agilisme és una resposta als fracassos i les frustracions del model en cascada.

Contempla un enfocament per a la presa de decisions i la forma d'organització en els projectes de programari, basant-se en els models de desenvolupament iteratiu i incremental, amb iteracions curtes (setmanes) i sense que dins de cada iteració hagi d'haver-hi fases lineals.



Característiques

- L'essència de l'*agilisme* és l'habilitat per a adaptar-se als canvis. La superació d'obstacles imprevistos té prioritat sobre les regles generals de treball preestablertes. És a dir, és per a la forma habitual de treball o la forma en la qual es llancen noves versions, per a resoldre un problema.
- També existeixen les fases d'anàlisis, desenvolupament i proves però, en lloc de ser consecutives, estan solapades i se solen repetir en cada iteració. Les iteracions solen durar de dues a sis setmanes.
- En cada iteració es parla amb el client per a analitzar requeriments, s'escriuen proves automatitzades, s'escriuen línies de codi noves i es millora codi existent (refactorització).
- Al client se li ensenyen els resultats després de cada iteració per a comprovar la seva acceptació i incidir sobre els detalls que s'estimin oportuns

Característiques

- ➔ Tot l'equip treballa unit, i el client és part d'ell. No és un oponent. L'estratègia de joc ja no és el control sinó la col·laboració i la confiança.
- ➔ La jerarquia clàssica (director tècnic, analista de negoci, arquitecte, programador sènior, júnior ...) perd sentit i els rols es disposen sobre un eix horitzontal, on cadascú compleix la seva comesa però sense estar per damunt ni per sota dels altres.
- ➔ En lloc de treballar per hores, es treballa per objectius i s'usa el temps com un recurs més. (Però existeixen dates de lliurament per a cada iteració).
- ➔ En qualsevol mètode àgil, els equips han de ser petits, típicament menors de set persones.

Característiques

- ➔ Tot l'equip es reuneix periòdicament, inclosos usuaris. Les reunions tenen hora de començament i de final i són breus.
- ➔ L'aplicació s'acobla i es desplega diàriament, de manera automatitzada. Les bateries de tests s'executen diverses vegades al dia.
- ➔ Els desenvolupadors envien els seus canvis al repositori de codi font almenys una vegada al dia (*commit*).

SCRUM

SCRUM

Scrum és un marc de treball àgil que ajuda equips a desenvolupar productes complexos de manera incremental i iterativa. Es centra en la col·laboració, la transparència i la millora contínua, mitjançant rols definits, reunions i artefactes.

- **Àgil** → adaptació ràpida als canvis.
- **Iteratiu** → es treballa en cicles curts (sprints).
- **Incremental** → cada sprint afegeix valor real al producte.
- **Autogestionat** → l'equip decideix com treballar.
- **Col·laboratiu** → treball proper entre equip, Product Owner i stakeholders.

SCRUM - Beneficis

- ➔ **Resoldre problemes complexos**: Ajuda als equips a abordar desafiaments complexos de manera eficient i creativa.
- ➔ **Entregar valor de manera incremental**: S'entrega funcionalitat al client de manera contínua, en lloc d'esperar al final del projecte.
- ➔ **Adaptar-se al canvi**: Permet als equips adaptar-se ràpidament als canvis en els requisits o l'entorn del projecte.
- ➔ **Fomentar la col·laboració**: Promou un entorn de treball col·laboratiu i autoorganitzat.

SCRUM PROCESS

SCRUM - Rols

1. Product Owner (Propietari del Producte)

- ➔ Representa el client o negoci.
- ➔ Defineix la visió del producte.
- ➔ Gestiona i prioritza el Product Backlog.
- ➔ Decideix què s'ha de fer i l'ordre d'importància.
- ➔ Ha de ser una persona, mai un comité.
- ➔ Tota l'organització ha de respectar les seves decisions i és l'única persona que pot canviar les prioritats.
- ➔ Les decisions que presa es reflecteixen en la descripció de les tasques i en la priorització del Backlog.

Objectiu: assegurar que l'equip treballa en el que té més valor.

SCRUM - Rols

2. Scrum Master

- ➔ És el facilitador del procés Scrum.
- ➔ S'assegura que l'equip entén i segueix les pràctiques Scrum.
- ➔ Ajuda a eliminar impediments que bloquegin l'equip.
- ➔ Fomenta la comunicació i la millora contínua.
- ➔ Assegurar que els objectius, l'abast i el domini del producte siguen entesos per tot l'equip Scrum.
- ➔ Trobar tècniques per a gestionar el Backlog de manera efectiva.
- ➔ Ajudar a l'equip Scrum a entendre la necessitat de comptar amb elements del Backlog clars i concisos.
- ➔ Entendre la planificació del producte en un entorn basat en l'experiència.

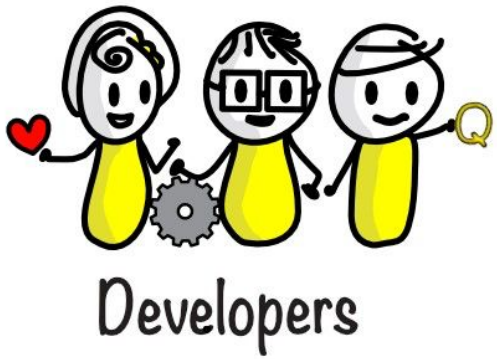
Objectiu: que l'equip pugui treballar de manera eficient i sense barreres.

SCRUM - Rols

3. Equip de Desenvolupament (Development Team)

- ➔ Un grup autoorganitzat i multidisciplinari: Ningú els indica quines tasques del Backlog se seleccionen incrementalment, són les prioritats qui ho defineix.
- ➔ Format per desenvolupadors, testers, dissenyadors, etc.
- ➔ Decideix com fer la feina per aconseguir els objectius del sprint.
- ➔ Sol treballar en equips petits (5–9 persones).

Objectiu: construir l'increment del producte al final de cada sprint.



SCRUM PROCESS

SCRUM - Materials

- Product backlog: és una llista prioritzada amb tots els requisits del client.
- Sprint backlog: és la llista de tasques identificades per a ser completades durant el sprint.
- Increment: és la suma de tots els ítems del Product Backlog completats durante un Sprint.

SCRUM PROCESS

SCRUM - Fases

Planificació del Sprint

- ➔ Es defineix què es farà durant el sprint (1–4 setmanes).
- ➔ El Product Owner presenta el Product Backlog.
- ➔ L'equip selecciona les històries d'usuari i crea el Sprint Backlog.

👉 Pregunta clau: Què farem i com ho farem?

SCRUM - Fases

Execució del Sprint

- ➔ L'equip treballa en les tasques definides.
- ➔ Es fa el Daily Scrum (15 minuts cada dia).
- ➔ El Scrum Master ajuda a eliminar impediments.

👉 Treball diari i coordinació constant.

SCRUM - Fases

Revisió del Sprint

- ➔ Al final del sprint es mostra el producte incrementat.
- ➔ Participen l'equip, Product Owner i stakeholders.
- ➔ Es recull feedback per ajustar el Product Backlog.

👉 Comprovem el que hem fet i escoltem el client.

SCRUM - Fases

Retrospectiva del Sprint

- ➔ L'equip reflexiona sobre com ha treballat.
- ➔ Identifica punts forts i aspectes a millorar.
- ➔ Es decideixen accions per al proper sprint.

👉 Millora contínua en cada iteració.

SCRUM - Temporalització

1. Planning Poker

- Cada tasca es puntua segons la seva **complexitat o esforç** (no hores exactes).
- Es fan servir cartes amb números (normalment Fibonacci: 1, 2, 3, 5, 8…).
- L'equip discuteix i arriba a un consens sobre l'esforç.
- **Avantatge:** ajuda a preveure quantes tasques caben en un sprint sense haver de posar hores exactes.

2. Story Points

- Cada història d'usuari rep un valor que representa el **treball total que implica** (temps, complexitat i risc).
- Ex.: una tasca fàcil = 2 punts, mitjana = 5 punts, complexa = 8 punts.
- Permet **comparar tasques i planificar la capacitat del sprint**.

SCRUM - Temporalització

3. T-Shirt Sizing

- Classificació de tasques en **mida relativa**: XS, S, M, L, XL.
- Ideal per equips que prefereixen estimacions visuals ràpides.
- Després es tradueix a punts o temps aproximat segons la capacitat de l'equip.

4. Timeboxing de tasques

- Tot i que Scrum no assigna hores exactes, **cada tasca pot tenir un límit màxim recomanat**.
- L'equip decideix que una tasca **no hauria d'excedir cert nombre de dies** dins del sprint.
- **Avantatge:** evita que una tasca s'allargui i bloquegi l'sprint.

SCRUM PROCESS