UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

GUIA PER AL DESENVOLUPAMENT EFICIENT D'UN PROJECTE INFORMÀTIC

Marc Calviño i Cornudella

Director: Oliver Díaz Montesdeoca

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 21 de juny de 2020

Agraïments

Ja que aquest projecte marca el final d'una etapa molt important, m'agradaria agrair el suport que he rebut per part de tots aquells que m'han acompanyat durant aquests anys. La família en primer lloc, tots els amics que he guanyat i els professors que han aportat les seves ganes i energia en fer-nos créixer.

També m'agradaria donar les gràcies al meu tutor, l'Oliver, que m'ha escoltat pacientment i m'ha donat consells per a poder treure el màxim rendiment a la idea del treball.

I un altre cop, com a pilar fonamental, agrair als meus pares l'esforç que han fet perquè jo pugui arribar fins aquí, ells saben que no ha sigut fàcil.

Moltes gràcies

Resum

En aquest projecte s'ha treballat per a obtenir unes directrius bàsiques a l'hora de desenvolupar una aplicació informàtica amb l'objectiu d'agrupar tot el procés, des del disseny fins al desenvolupament.

Seguint una estructura lineal, es pot seguir el treball pas a pas i recorrent els dos grans blocs: disseny i desenvolupament del producte.

En primer lloc trobarem tota la informació per a contextualitzar les idees i definir tota l'aplicació pel que fa a funcionalitat i aparença.

En segon lloc trobarem les nocions bàsiques per a definir l'arquitectura del projecte, seleccionar les tecnologies i dissenyar el programa.

Aquesta guia pretén unificar els conceptes de disseny i desenvolupament bàsics en un sol document, que pugui servir per acompanyar als desenvolupadors i ajudar a millorar la qualitat final dels productes.

Resumen

En este proyecto se ha trabajado para obtener unas directrices básicas a la hora de desarrollar una aplicación informática con el objetivo de agrupar todo el proceso, desde el diseño hasta el desarrollo.

Siguiendo una estructura lineal, se puede seguir el trabajo paso a paso y recorriendo los dos grandes bloques: diseño y desarrollo del producto.

En primer lugar encontraremos toda la información para contextualizar las ideas y definir toda la aplicación en cuanto a funcionalidad y apariencia.

En segundo lugar encontraremos las nociones básicas para definir la arquitectura del proyecto, seleccionar las tecnologías y diseñar el programa.

Esta guía pretende unificar los conceptos de diseño y desarrollo básicos en un solo documento, que pueda servir para acompañar a los desarrolladores y ayudar a mejorar la calidad final de los productos.

Abstract

In this project, I have assembled basic guidelines to follow when developing a computer application. This assemblage aims to group the entire process from design to development.

With a linear structure, this project can be followed step by step through two main blocks: product design and development.

First, we find all the information needed to contextualize the ideas and defining the entire application in terms of functionality and appearance.

Secondly, we find elemental notions to define the project architecture, to select the technologies and to design the program.

This guide intends to unify the basic design and development concepts in a single document, thus serving to assist developers and aid them to improve the final quality of products.

Continguts

1 Introducció

Durant les últimes dècades la tecnologia informàtica ha revolucionat la forma en què veiem el món i cada vegada més, tota la nostra vida gira entorn dels nostres dispositius informàtics. Internet of Things (IoT), portàtils i mòbils, són algunes de les eines que fem servir per interaccionar amb l'entorn ja sigui en l'àmbit acadèmic, laboral o social. És per això que la professió "d'inventor" gira cada cop més entorn de la programació en diferents llenguatges i plataformes. Aquest paradigma ens porta a la necessitat d'obtenir recursos i formar a les noves generacions en l'art de transformar idees a aplicacions informàtiques.

Aquest projecte parteix de la voluntat de transportar idees o necessitats etèries, que molts de nosaltres tenim, a un projecte concret preparat per a sortir al mercat o per a començar a desenvolupar el producte final.
Durant tot aquest treball seguirem tot el procés necessari per a centrar la idea tecnològica, ajustar-la a les necessitats dels usuaris i desenvolupar-la usant les millors tecnologies disponibles en funció dels requeriments previstos per l'app.

Durant el transcurs del projecte treballarem sobre un projecte concret per tal d'escenificar la situació referida en el primer paràgraf, amb l'objectiu de fer un recorregut per les tècniques i metodologies que existeixen per a realitzar aquesta tasca.

2 Motivació i objectius

2.1 Motivació

Durant la meva estada en la carrera d'enginyeria informàtica, he participat en molts projectes tant a nivell acadèmic com en l'àmbit laboral i he treballat conjuntament amb els meus companys en ells. En el transcurs d'aquestes activitats, hem dissenyat infinitats d'aplicacions web, mòbil i programes en els quals la programació era el pal de paller i on es discutia sobre l'estructura, recursos i tecnologies a usar. També hem desenvolupat mapes conceptuals de classes i fils d'execució, protocols i moltes altres característiques tècniques dels projectes.

No obstant sempre he trobat a faltar el focus en una part vital del desenvolupament i que sempre va íntimament lligada al procés de programació. El disseny de la interfície i la cura envers la usabilitat del que es desenvolupa és un gran dèficit en la majoria de projectes en els quals he participat.

La intenció d'aquest treball, és proporcionar recursos per a la correcta execució d'aquesta part fonamental en la creació de productes informàtics. Iniciant el procés en la base, la concreció de la idea mare del projecte i recorrent tots els passos fins a desembocar en la creació d'un producte inicial, preparat i enfocat per als usuaris.

Durant tot el projecte es parlarà de forma detallada de cada un dels passos a seguir, els diferents mètodes que existeixen per a dur-los a terme i com testejar-los de forma adequada. Així mateix pretén ser una guia entenedora i a l'abast de qualsevol persona que vulgui desenvolupar un producte informàtic, que ajudi a millorar tot el procés i proporcioni pautes fàcils de seguir per a poder millorar els productes finals.

També es pretén fer patent aquesta mancança que existeix en el camp del desenvolupament informàtic i al final poder fer una reflexió amb coneixement de causa per a poder convèncer als professionals o futurs professionals que tinguin a bé documentar-se amb aquest treball.

2.2 Objectius

Com a objectius d'aquest treball i molt lligats a la motivació del mateix, trobem els següents imprescindibles:

- Detallar i ampliar informació sobre tots els passos del desenvolupament informàtic, iniciant el procés en la conceptualització de la idea bàsica, passant per tots els passos (anàlisis, prototipat, testeig, arquitectura, tecnologies i desenvolupament), fins a arribar a un producte final.
- Mostrar el camí per a obtenir un producte final orientat a fer una primera ronda de prova, és a dir, una demo vàlida i funcional on estiguin reflectides totes les necessitats i finalitats del producte i que pugui ser analitzada per teòrics clients, compradors, inversors, etc.
- Mostrar tot el procés de disseny de forma completa, amb exemples pràctics per tal d'escenificar tots els conceptes dels quals es parli durant el transcurs del treball, amb la intenció que sigui molt més accessible per a tothom que hi tingui interès i no estigui versat en alguna o cap peça d'aquest procés de creació.
- Crear una guia general sobre les tecnologies existents i els punts claus sobre l'elecció d'aquestes. Un espai on poder obtenir un coneixement inicial i una orientació per a poder estructurar correctament el desenvolupament tècnic del nostre producte.
- I finalment aprofitar la claredat i simplicitat de tot el que es transmet per a conscienciar als futurs desenvolupadors en la importància del coneixement integral i transversal amb el qual poder executar de forma molt més encertada la tasca que ens sigui assignada o poder afrontar el projecte en la seva totalitat des d'un punt de vista més ampli.

3 Planificació

3.1 Estructura del treball

Aquest treball té dos grans blocs conceptuals que van molt lligats però que tenen competències diferents. En primer lloc tenim tota la part de disseny i usabilitat, la part més enfocada al client, i en segon lloc tenim la part més tècnica, a on trobarem informació sobre llenguatges, tecnologies i estructuració dels projectes. És per això que el dividirem en:

- **Bloc 1: disseny i conceptualització del projecte.** A on trobarem detallat tot el procés de focalització, anàlisis de requisits d'usabilitat, prototipat i testeig de la usabilitat del nostre producte. Un espai on centrar-se per a millorar el resultat final del nostre producte respecte a la qualitat envers l'ús que en faran els nostres usuaris, millorant així l'èxit que puguem tenir.
- **Bloc 2: estudi i execució dels aspectes tècnics**. En aquest bloc en canvi, trobarem informació sobre les tecnologies que existeixen en el mercat, les característiques en les que ens hem de fixar per tal d'escollir-les i els correctes mètodes a usar per a construir de la forma més robusta i eficient el nostre producte.

És en el coneixement d'aquests dos blocs alhora, on residirà la capacitat de ser un desenvolupador més complet i obtenir millors resultats finals.

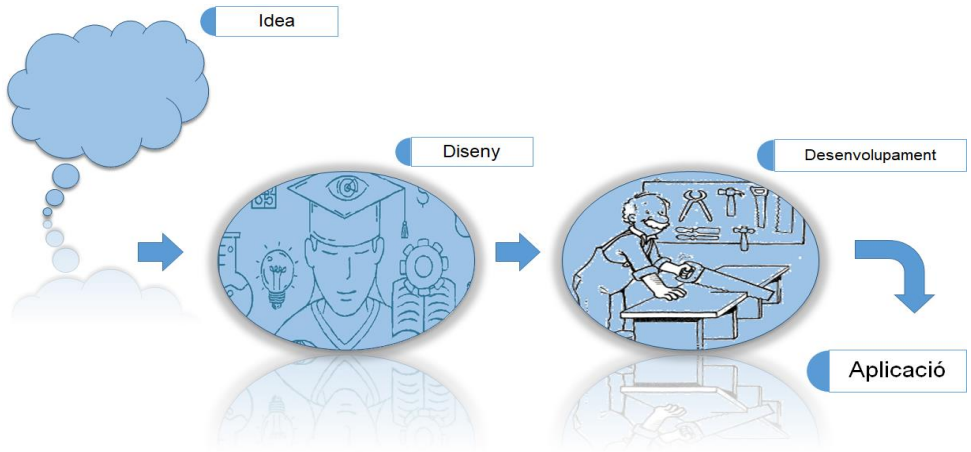


Figura 1 – Estructura de treball

3.2 Etapes del projecte

Per tal de desenvolupar aquesta guia, primer ens centrarem en la recopilació d'informació de totes les fonts possibles per a redactar, i dotar de valor i contingut els passos a seguir durant el transcurs de l'execució d'un projecte. D'aquesta manera construirem una guia per tal de poder posar en ordre tot el que hàgim après. Seguidament redactarem la informació del bloc 1 i contrastarem la part pràctica de forma que tinguem una base per a continuar avançant en el nostre propòsit. Més endavant farem el mateix amb el bloc 2 de manera que puguem utilitzar tota la informació recollida per a executar de manera coherent la construcció del nostre producte.

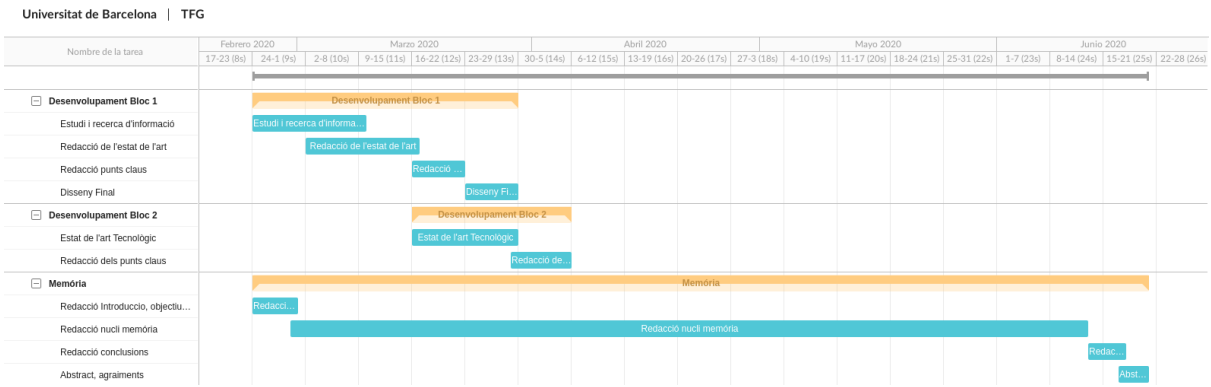


Figura 2 – Diagrama de Gantt

Aquestes quatre fases (cerca en el disseny, execució del model conceptual, cerca de les tecnologies i processos i avaluació d'una l'aplicació) se superposaran en diferents moments, amb l'objectiu de què puguin influenciar-se unes amb les altres.

A l'inici d'aquest treball hem creat un "diagrama de gantt" (Figura 2) per tal de definir les tasques i la seva temporalitat. Les tasques que hem definit per a fer la recerca inicialment són:

- Disseny conceptual
  - Necessitats i característiques.
  - Disseny preliminar.
  - Avaluació i Testeig.
  - Disseny Final.
- Desenvolupament
  - Estat de l'art Tecnològic
  - Creació de l'stak tecnològic
  - Disseny del codi

- Memòria
    - Redacció Introducció, objectius...
    - Redacció nucli de la memòria
    - Redacció conclusions

Durant tot el treball anirem seguint aquestes tasques i si és necessari afegint-ne de noves i readaptant el “diagrama de gantt” a les nostres necessitats temporals, amb l’objectiu de dedicar la millor de les atencions a cada una de les seccions del treball.

4 Bloc 1 - Disseny

El disseny d'una aplicació requereix de diversos passos interactius, on el desenvolupador ha de buscar el punt de vista de l'usuari i transformar-se en ell. En diverses fases com la creació de persones o el prototipat així com l'avaluació de les mateixes ens aproximem a un resultat en que optimitzem l'experiència en la interacció amb l'aplicació i la seva funcionalitat.

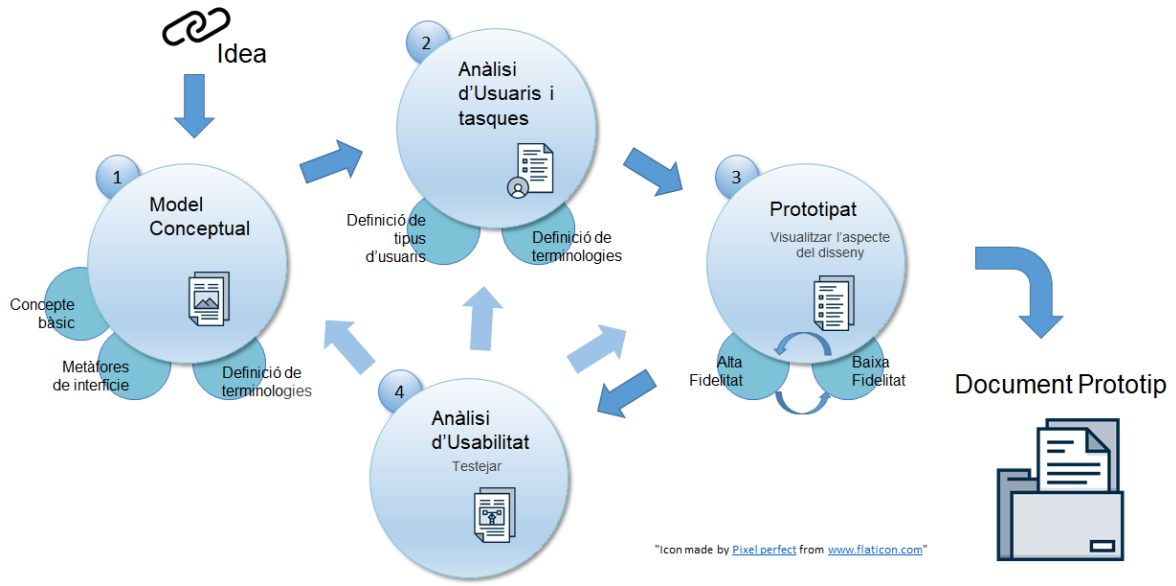


Figura 3 – Esquema del Bloc 1

El producte final emergeix de forma iterativa a partir de diferents cicles de disseny i avaluació que involucren a usuaris i als desenvolupadors en un vincle afectiu i d'empatia.

Els dos aspectes principals per al correcte aproximament a una solució final satisfactoria, son els següents:

- La part conceptual
- L'aspecte concret que es centra en els detalls.

La primera implica desenvolupar un model conceptual a on es manifesti que ha de fer el producte: com es comportarà, quins objectius vol assolir o quina necessitat de l'usuari cobreix. La segona part, s'ha de centrar en els detalls, és a dir, tasques concretes, menús existents o la part gràfica en si. Aquestes idees desembocaran en la creació d'un o varios prototips que podran ser avaluats per usuaris tipo i seguidament millorats.

Aquest procés no té un final clar i definit, i podriem seguir iterant repetides vegades. És el responsable del projecte, o de l'equip, qui ha de decidir quan ens hem aproximat a una solució

vàlida. A partir de les hores, començar a estructurar i planificar el procés de desenvolupament tècnic en el cas de les aplicacions informàtiques.

Un standard en aquest estil de disseny, és el definit en el "*The Design Council of the United Kingdom"* com a "el doble diamant del disseny". Aquest procés estructura 4 fases iteratives amb l'objectiu de fer l'aproximació prèviament esmentada.

- **Descobrir**: indagar i obtenir el màxim d'informació sobre el problema.
- **Definir**: els dissenyadors desenvolupen el model conceptual que emmarca tota la informació útil (idea, entorn, usuaris, finalitat i tasques).
- **Desenvolupar**: es desenvolupen diferents prototips i tècniques per a testejar-lo.
- **Avaluar**: es posen a prova els prototips i es proposen millores per a la següent iteració.

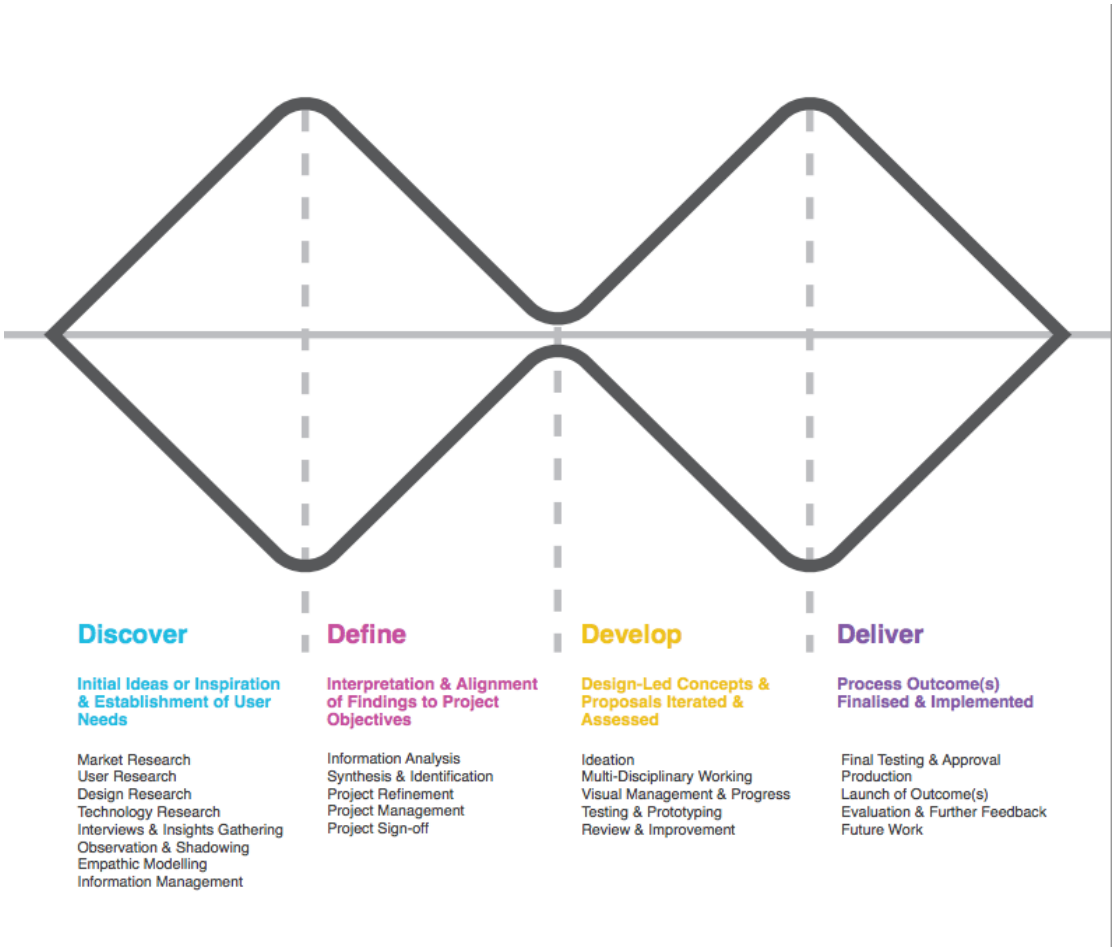


Figura 4 – El doble diamant del disseny

4.1 Contingut del bloc

En aquest bloc s'executarà el procès de disseny, fins a tenir un prototip acceptable per a començar amb l'estudi de l'arquitectura tècnica del producte final, i es detallarà en cada cas les millors maneres d'executar el pas concret de disseny.
Es divideix en quatre fases:

- Desenvolupar el model conceptual,
- Realitzar l'anàlisis dels usuaris i les tasques,
- Analitzar la usabilitat i el contingut necessari, realització del prototipat i com a nexe final
- L'avaluació i testeig del resultat.

En la primera fase, **desenvolupar el model conceptual**, trobarem detallat com sintetitzar la idea i transportar al mon de l'usuari tot el projecte, així com plasmar els conceptes concrets relatius al projecte.
En la segona fase, **anàlisi d'usuari i tasques**, indagarem en els perfils als quals orientem el projecte i intentarem definir-los, també analitzarem les tasques concretes que es podran dur a terme en el producte final per tal de tenir una guia en el moment de desenvolupar.
En la tercera fase, **anàlisi de la usabilitat**, explicarem els tòpics a tenir en compte i la metodologia necessària per a preparar el projecte, amb la finalitat d'obtenir el millor producte final possible.
En la quarta fase, **prototipat**, posarem en pràctica els resultats que hàgim obtingut de les fases anteriors, creant diferents prototips per a poder tenir un resultat parcial.
Al final de tot explicarem les tècniques d'avaluació i testeig que hauran d'acompanyar a cada una de les fases, assegurant així que el resultat final sigui el millor possible.

4.2 Desenvolupament del model conceptual

Solem començar a treballar en la definició del projecte focalitzant l'esforç en la resolució d'aspectes tècnics i deixem de costat l'arrel del nostre projecte. Sovint es diu que tota innovació parteix d'una idea, però rarament el que constitueix en realitat una idea nova es reflecteix en la gestió del projecte conseqüent.

El fet de no tenir una comprensió plena del concepte i les seves dimensions, pot provocar que es rebutgin idees potencialment bones per no ser prou elaborades per rebre una avaluació justa. A més, la falta de definició sòlida d'una idea té implicacions en el desenvolupament del projecte i la seva gestió, ja que l'estudi de la creació i avaluació del projecte són difícils d'executar quan l'objecte estudiat - la idea - no està correctament definit.
En el marc de la informàtica aquest paradigma s'accentua, ja que els aspectes tècnics acostumen a tenir molt més protagonisme incidint en el resultat final.
Segons Andrew Powell-Morse un model conceptual és una representació d'un sistema que utilitza conceptes i idees per formar aquesta representació. És a dir, que s'ha d'utilitzar per a descriure aspectes físics, ideals, socials i d'entorn de forma abstracta, perquè formin un tot amb el qual poder treballar. Aquest model conceptual ha de complir 4 objectius bàsics:

- Millorar la comprensió del sistema al qual fa referència.
- Promoure un coneixement profund dels objectius i necessitats del sistema.
- Proporcionar un punt de referència per al disseny i la implementació del sistema.
- Documentar les fronteres, el contingut i els objectius del sistema per a futures interaccions amb el mateix.

Per tant, el model conceptual ha de permetre fer-se una imatge clara i ben definida de la idea que estiguem representant, buscar una abstracció que descrigui el que les persones poden fer amb un producte i quins conceptes són necessaris per a comprendre com interactuar amb el producte a desenvolupar.
A l'hora de desenvolupar un model conceptual, l'experiència ha portat a la creació d'unes directrius de bones pràctiques amb l'objectiu d'ajudar a fer un bon model conceptual. Un model conceptual hauria de:

- Estar a disposició de tots els membres de l'equip per facilitar la col·laboració i la iteració.
- Poder-se modificar fàcilment, amb l'accés a nova informació actualitzada.
- Tenir contingut tant visual com escrit, per explicar millor els conceptes abstractes que pot representar.
- Establir termes i conceptes rellevants que s'utilitzaran durant tot el projecte.

- Definir els termes i els conceptes.
- Proporcionar una estructura bàsica per a les entitats del projecte.

Els avantatges del model conceptual depenen en gran mesura de la capacitat que es tingui per idear un model fort i molt entenedor. En general, els avantatges principals són:

- **Estableix entitats:** establint i definint les diverses entitats i conceptes que podrien aparèixer al llarg del desenvolupament del projecte, el model pot ajudar a garantir que hi hagi menys sorpreses pel camí, dedicant un temps previ en què analitzar possibles entitats o relacions.
- **Defineix l'àmbit del projecte:** es pot utilitzar el model com una manera de definir l'abast del projecte, i ajudar a la gestió del temps i la programació.
- **Model base per a altres models:** per a la majoria de projectes caldrà generar models addicionals, menys abstractes, més enllà dels conceptes generals definits a la secció del model conceptual. Llavors es pot usar com a excel·lent punt de salt des del qual es poden crear models més concrets, com ara els models lògics de dades o classes.
- **Comprensió d'alt nivell:** serveix com a gran eina proporcionant una comprensió d'alt nivell d'un sistema durant tot el cicle de vida del desenvolupament de projecte.

Per a poder dur a terme aquesta tasca tan important per a qualsevol projecte, és interessant definir un *framework* amb el que treballar, de forma que ens pugui guiar en l'execució. Els components bàsics han de ser:

- Metàfores i analogies que transmeten a les persones per a què serveix un producte i com utilitzar-lo per a una activitat (per exemple, el sistema de fitxers i les "carpetes").
- Els conceptes als quals s'exposen les persones a través del producte, inclosos els objectes de domini de tasques que creen i manipulen, els seus atributs i les operacions que es poden realitzar sobre ells (com ara desar, revisar i organitzar).
- Les relacions entre aquests conceptes (per exemple, si un objecte en conté un altre).
- Les relacions entre els conceptes i l'experiència que l'usuari tindrà amb el producte.

Per tant crearem tres apartats a continuació amb l'objectiu de construir el model conceptual, **el concepte bàsic**, on trobarem una definició de la idea i el context, **els termes**, on aclarirem algunes definicions que creguem necessàries per a la comprensió del projecte i **les metàfores d'interfície**, on intentarem apropar els conceptes de l'aplicació al món físic de l'usuari fent-los més assequibles creant analogies i metàfores.

4.2.1 El concepte bàsic

En aquest apartat haurem de definir mitjançant una redacció, imatges i/o esquemes, l'essència del nostre producte, i tots els aspectes relacionats amb l'usuari i la seva interacció. Podem parlar sobre 4 tipus d'interacció, **ordres directes**, **diàleg amb el sistema**, **exploració** i **notificació**. Cada una d'elles té uns actors principals i uns secundaris, una finalitat i una importància diferent.

Figura 5 – Definició del Concepte bàsic

Durant el procés de redacció haurem de vigilar de no ometre aspectes que ens semblin obis, ser molt específic en les explicacions i cobrir tot l'espai conceptual del projecte. Només així ens quedarà prou complet per a poder continuar amb les següents fases. Hi ha diferents estratègies a seguir per a obtenir un nivell de profunditat òptim i moltes vegades són complementàries entre si.

- **Múltiples aproximacions**: és recomanable començar múltiples documents des d'un enfocament distint per tal d'obtenir resultats diferents i que en el moment d'agrupar-los augmenti la comprensió del projecte.
- **Múltiples iteracions:** és una bona estratègia revisar i escriure el que ja hem redactat amb l'objectiu de detectar errors i millorar el detall després de deixar madurar el resultat.
- **Mur d'idees**: és recomanable tenir un espai on acumular conceptes, paraules claus i idees que anem pensant al llarg del procés, amb l'objectiu d'incloure les que tinguin més sentit en el nostre document final.

Figura 6 – Mur d'idees

Per a tenir un bon model conceptual és important que en el concepte bàsic aportem especificacions sobre els temes més concrets i que en les successives iteracions que es puguin donar, anem ampliant i agrupant aquestes explicacions detallades per a entendre de forma clara a quin àmbit pertanyen i si s'han de tractar com un conjunt o separades.

Figura 7 – Especificacions del model

Totes aquestes estratègies han de ser accessibles i compartides amb tot l'equip, així tots els integrants podran contribuir, millorant entre tots el resultat final. En cas de ser una sola persona pot ser interessant compartir els resultats parcials amb tercers per tal d'aportar noves perspectives.

4.2.2 Les metàfores d'interfície

Es considera que les metàfores són un component central d'un model conceptual. Proporcionen una estructura familiar amb l'usuari que descriu els aspectes d'una o d'unes entitats i proporciona nous conceptes deixant-los més a l'abast. Una metàfora s'integra en la interfície i passa a formar part del vocabulari dels usuaris, com la metàfora de la paperera de reciclatge.

Estan pensades per evocar l'essència del procés a realitzar, permetent a l'usuari enllaçar aspectes menys familiars de la funcionalitat proporcionada amb conceptes més propers del dia a dia. Les metàfores d'interfície tenen com a objectiu proporcionar entitats familiars que permetin a la gent comprendre el model conceptual subjacent i saber què fer a la interfície.
L'objectiu és poder dissenyar noves tecnologies que permetin als usuaris fer coses sense haver de pensar en com utilitzar-les. Exemples d'aquest tipus de metàfores d'interfície es mostren a la taula 1.

***Taula 1**. Exemples de metàfores* d'interfície.

Area d'aplicació	**Metàfora**	**Coneixement familiar**
Entorn operatiu	L'escriptori	Tasques d'oficina, gestió de fitxers
Fulles de càlcul	Llibre contable	Taules encolumnades
Entorns d'aprenentatge	Viatges	Giras, guies, navegació

Malgrat tot, és important fer notar que la metàfora és un recurs perillós del qual no hem d'abusar i és recomanable usar-lo només per als conceptes més importants. En el disseny d'interfícies *Alan Cooper (*1995) en el seu article "*el mite de la metàfora*" ens avisa del perill de les metàfores en la mesura que són complicades de trobar i limiten el pensament, a més a més hem d'estar segurs que l'assimilació que fem nosaltres també la farà l'usuari.

També *Cooper* ens parla dels tres paradigmes en la interfície d'usuari, el tecnològic, l'idiomàtic, i el metafòric. El mètode que millor millora la corba d'aprenentatge és el metafòric, ja que es basa a intuir com funcionen les coses sense necessitat d'una comprensió profunda o d'una explicació detallada com sí que ho requereixen els altres mètodes.

Per a desenvolupar aquest apartat, ens hem de fixar en els processos o accions característiques a realitzar per l'usuari, del nostre projecte i identificar les més importants o complicades. Llavors buscarem metàfores que apropin l'ús a tot el públic objectiu per tal de millorar l'empatia cap al nostre producte i la velocitat d'aprenentatge sobre el mateix.

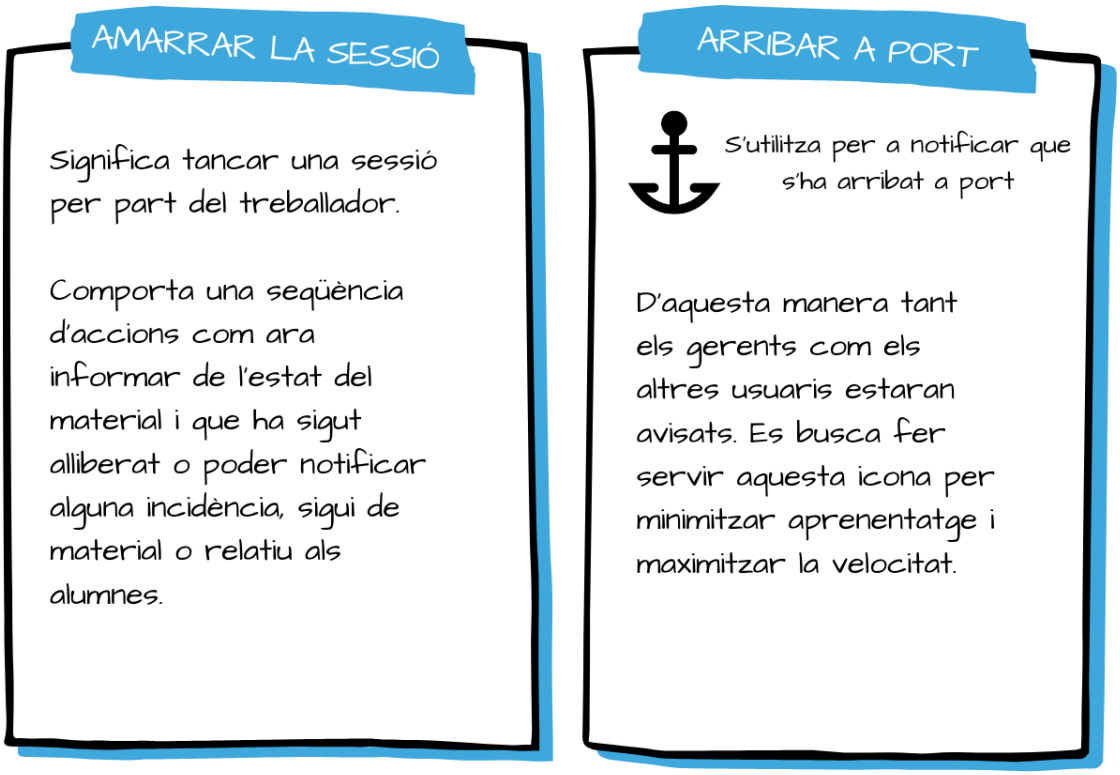


Figura 8 – Definició de metàfores

4.2.3 La terminologia

En tot projecte, on interactuen múltiples entitats i es defineixen tasques complexes, s'acaben creant termes que engloben un seguit d'idees. Moltes vegades al treballar durant un temps llarg amb el projecte acabem absorbint el seu significat i donant per sentada la comprensió de tercers. És recomana, des de l'inici del projecte, realitzar un recull d'aquests termes i del seu significat, detallant l'associació d'idees a la que fa referència i el context.

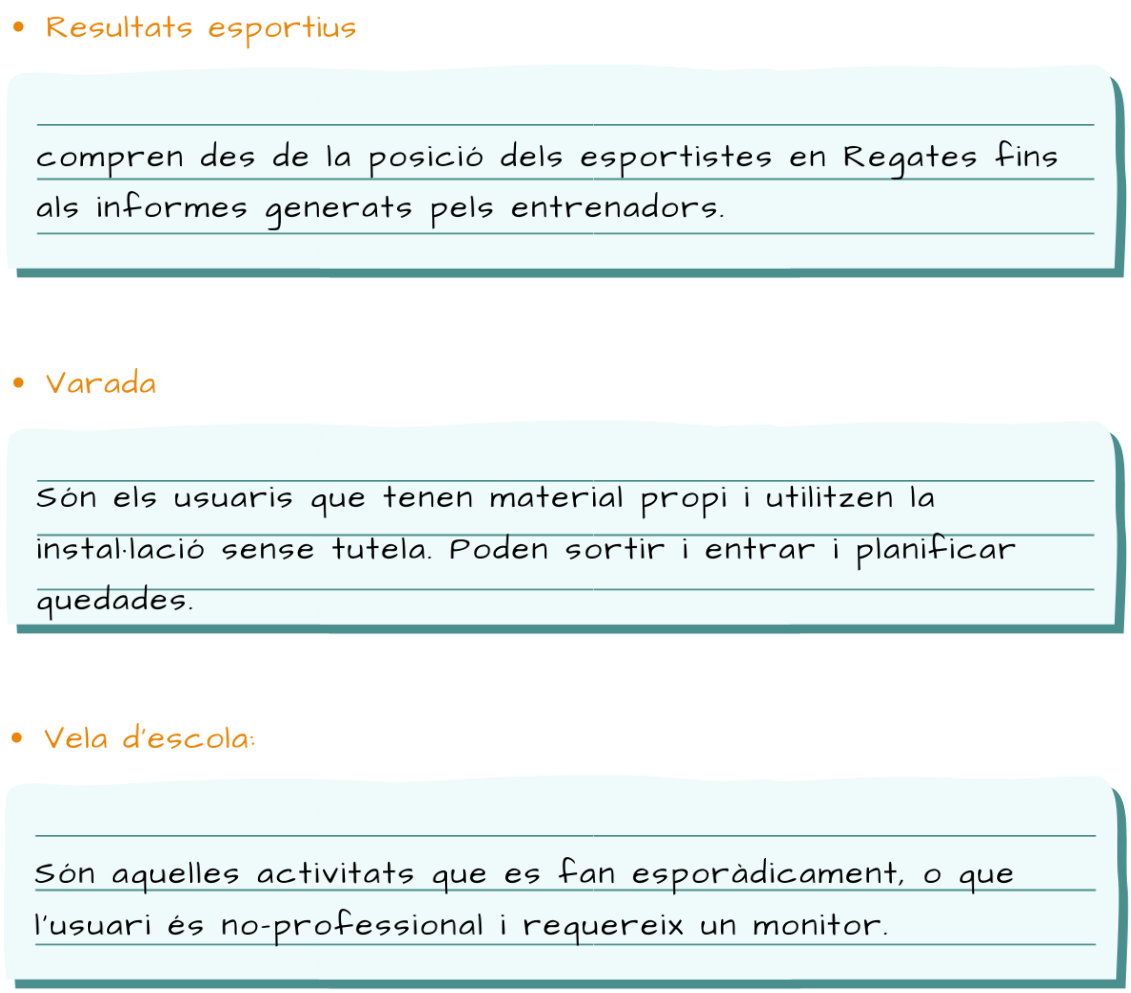


Figura 9 – Definició de la terminologia

4.3 Anàlisis d'usuaris i tasques

Un cop tenim definida la idea i l'essència del projecte, hem de pensar a dissenyar la part física i visual així com els processos que es duran a terme. No parlem de com es programa o quins serveis es fan servir, sinó qui farà servir el nostre programa i com el farà servir. Aquestes dues preguntes són de vital importància i és imprescindible que hi prestem atenció.

Per tant en aquest apartat focalitzarem el nostre interès en els usuaris finals, amb l'objectiu de poder conèixer exhaustivament les seves necessitats i mancances. Aquest tipus de disseny es coneix com a "*User-centered design*" (UCD).

Per a poder definir i avaluar tots els paràmetres d'usabilitat com farem en el pròxim apartat, necessitem una base sòlida que ens permeti conèixer amb profunditat l'usuari i les tasques que ha de desenvolupar. Amb aquesta finalitat desenvoluparem dos subapartats, l'anàlisi d'usuari i l'anàlisi de les tasques, intentant respondre les següents preguntes:

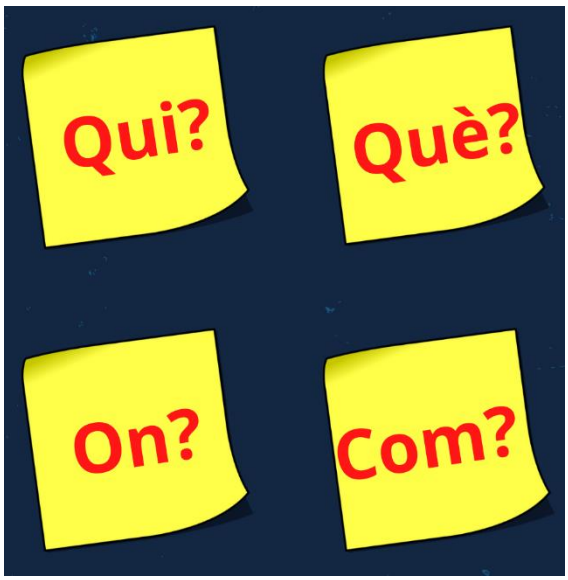


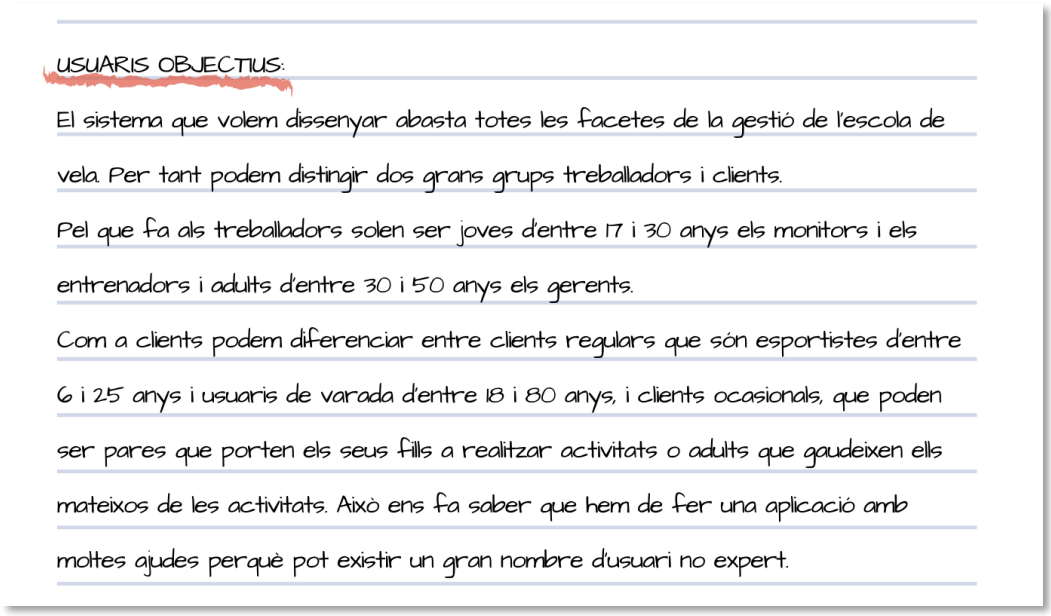
Figura 10 – Preguntes per avaluar la usabilitat

- **Qui?** qui és el nostre usuari, edat, sexe, capacitats, necessitats, etc.
- **Què?** que pretén trobar en el nostre programa, quines de les seves necessitats cobrirem.
- **On?** on farà servir el programa, al carrer, a casa, a l'oficina, etc, i amb quin dispositiu.
- **Com?** com l'usuari interaccionarà amb el nostre projecte, quines tasques farà.

4.3.1 Anàlisi d'usuaris

Una bona interfície d'usuari és la que respon als usuaris finals i els recolza en les tasques que volen realitzar. Es pot fer un disseny sense un bon coneixement dels usuaris i el que volen fer amb el sistema i es podrà fer servir per fer alguna cosa, però pot no fer el que vulguin fer els usuaris. El sistema funcionarà, però no serà necessàriament útil. Això no vol dir que tots els sistemes informàtics s'hagin de dissenyar per donar cabuda a tothom. Els sistemes informàtics s'han de dissenyar per a les necessitats i capacitats dels usuaris a qui estiguin destinats i les eines més potents són sempre les més senzilles.

Com defineix *Alan Cooper* en el seu llibre "*The Inmates Are Running the Asylum, The: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*" (2004), la nostra eina més eficaç és profundament senzilla: Elaborar una descripció precisa del nostre usuari i què vol aconseguir. A vegades pot semblar contraintuïtiu, però centrar el disseny en una persona concreta que reculli les característiques d'un conjunt dels nostres usuaris finals, és molt més efectiu.


USUARIS OBJECTIUS:

El sistema que volem dissenyar abasta totes les facetes de la gestió de l'escola de vela. Per tant podem distingir dos grans grups treballadors i clients.

Pel que fa als treballadors solen ser joves d'entre 17 i 30 anys els monitors i els entrenadors i adults d'entre 30 i 50 anys els gerents.

Com a clients podem diferenciar entre clients regulars que són esportistes d'entre 6 i 25 anys i usuaris de varada d'entre 18 i 80 anys, i clients ocasionals, que poden ser pares que porten els seus fills a realitzar activitats o adults que gaudeixen ells mateixos de les activitats. Això ens fa saber que hem de fer una aplicació amb moltes ajudes perquè pot existir un gran nombre d'usuari no expert.

Figura 11 – Identificació dels usuaris

També és important fer la distinció entre usuari i Persona, on l'usuari és un ésser humà físic, és a dir del món real i la Persona és una imatge que creem d'un grup d'usuaris on es recull de forma efectiva la informació de les característiques importants del grup. Per exemple

podem definir la Persona *Paquita*, que recull la informació de la gent major de setanta anys amb nets (avis) i podem enfocar el disseny en les necessitats d'aquesta Persona.
L'usuari real no deixa de ser important i li hem de dedicar una atenció considerable, però mai deixem que l'usuari afecti directament la solució, ja que és impossible resoldre les necessitats particulars de cada usuari, per contra ens podem centrar a resoldre la de 2, 4 o 6 Persones concretes.

El procés de creació de les persones és el següent:

- En primer lloc cal saber quins són els nostres usuaris objectius (Figura 11), en funció del projecte hem de fer un relat que respongui a les preguntes que s'han plantejat anteriorment (qui?, què?, com?, on?) i amb aquesta informació definir els blocs d'usuaris. Una bona manera de saber a qui pot anar dirigit és fent entrevistes a grups grans de persones o realitzant enquestes per conèixer les necessitats relatives de cada grup.
- En segon lloc crearem un seguit d'usuaris tipus (Persones), entre 20 o 30 per exemple, i els dotarem de característiques típiques del conjunt d'usuaris que vulguem agrupar (Figura 12). D'aquests usuaris també en farem Persones negatives, aquestes ajuden a saber per a qui no volem dissenyar, ajudant a centrar el disseny en les persones que ens interessen.
- Seguidament agruparem aquestes Persones en funció de les necessitats compartides generant cada cop més complexitat fins que ens en quedin entre 5 i 10 per exemple. Aquest mètode iteratiu ajuda a centrar els aspectes més importants del disseny i conèixer quines característiques concretes té cada perfil d'usuari.

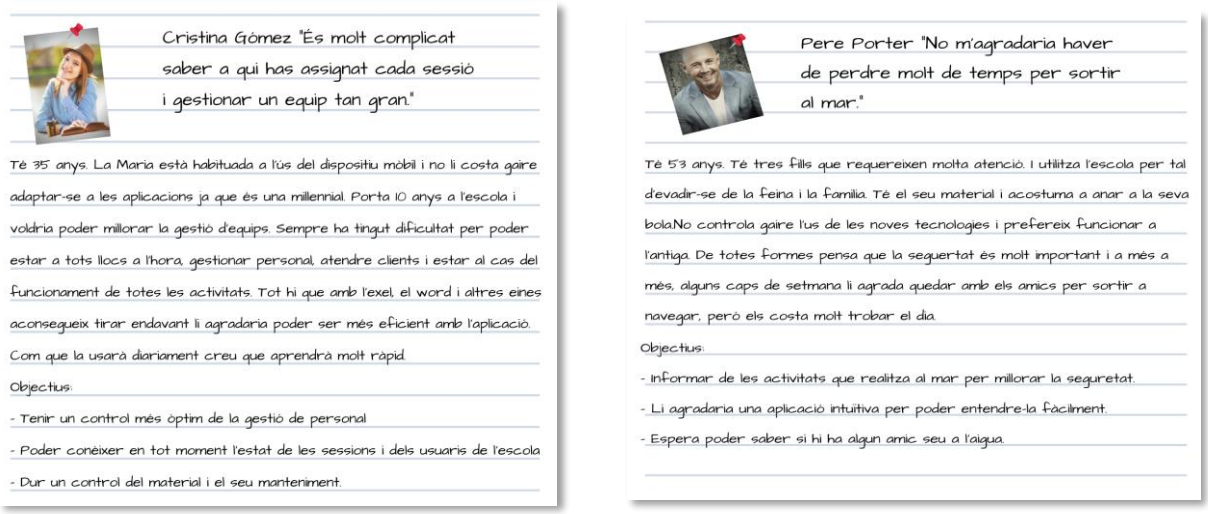


Figura 12 – Definició de persones

Quan hàgim de treballar amb aquestes persones per crear tasques o analitzar la usabilitat, podrem buscar solucions que satisfacin les necessitats individuals sense perjudicar a algun dels altres grups. També ens pot ajudar, en cas que siguin irreconciliables, a saber si estem abastant un espectre massa gran de tasques i potser hem de dividir el projecte en dos subproductes.

4.3.2 Definició de tasques

Després de determinar detingudament el model conceptual i els perfils d'usuari, els desenvolupadors hem d'identificar les tasques a realitzar. Aquest és un dels passos més complexes de tot el disseny i on hi ha més feina. Per a dur a terme aquest pas en el disseny hem de tenir en compte molts aspectes per a poder realitzar una llista final amb tota la informació necessària. Molt sovint l'anàlisi de tasques es fa de manera informal o implícita sense aturar-se a pensar amb un enfocament menys tècnic, que farà la persona que usi la tecnologia que estem desenvolupant. L'anàlisi de les tasques té una història llarga i mixta (Bailey, 1996; Hackos i Redish, 1998), però les estratègies de més èxit solen implicar llargues hores d'observació i entrevista d'usuaris. Això ajuda els dissenyadors a entendre la seqüència de tasques, la seva freqüència i a prendre decisions difícils sobre quines tasques han d'incloure en el projecte. Algunes vegades volem incloure totes les accions possibles amb l'esperança que a algun usuari li resultin útils, però això pot causar saturació i transformar en un caòtic conjunt d'accions el nostre resultat final.

Una tasca fa referència a un objectiu de l'usuari. Els objectius solen ser més conceptuals i les tasques els han d'integrar i definir l'ordre d'accions per a aconseguir-los. A vegades objectius molt importants s'han de dividir en subtasques que combinades apropin, de forma ordenada, a l'objectiu final. Per a poder definir tasques hem de tenir molt en compte el nivell d'habilitat de cada usuari, els objectius que volem resoldre, i els passos per a aconseguir-los.

"Conegueu el vostre usuari" és el primer principi de la “llista clàssica de principis d'enginyeria d'usuaris” de Hansen (1971). El procés de coneixement dels usuaris no s’acaba mai perquè hi ha molt per saber i perquè els usuaris continuen canviant. Cada pas per comprendre els usuaris i per reconèixer-los com a individus amb visions diferents de la del dissenyador és probable que sigui un pas més proper a un disseny amb èxit. Per fer la tasca accessible podem definir tres tipus d’usuaris en funció de les seves habilitats:

- **Usuaris novells**: Tenim els autèntics usuaris novells, per exemple, els avis que envien el seu primer missatge de correu electrònic a un nét i els usuaris que utilitzen per primera vegada aquesta interfície però coneixen els conceptes generals, per exemple, un viatger de negocis que utilitza el sistema de posicionament global (GPS) d'un cotxe de lloguer que mai ha conduït. Els dos grups d'usuaris poden arribar a tenir ansietat que inhibeix l'aprenentatge. Per a aquest grup hem de dissenyar tasques amb un nombre d'accions reduït, de manera que els usuaris novells i inicials puguin realitzar tasques senzilles amb èxit i així reduir l'ansietat, generar confiança i obtenir un reforç positiu.

- **Usuaris intermitents coneixedors**: Moltes persones són usuaris experimentats, però intermitents, de diversos sistemes. Tenen conceptes de tasques estables i un ampli coneixement de conceptes d'interfície, però poden tenir dificultats per seguir l'estructura dels menús o la ubicació de les funcions. L'objectiu de disseny per a aquest grup és mantenir una estructura ordenada dels menús, una terminologia consistent i una aparença identitària de la interfície, que ajuda al reconeixement del lloc. La protecció contra l'error és necessària per afavorir l'exploració relaxada de funcions o l'ús de seqüències d'acció parcialment oblidades.
- **Usuaris freqüents experts**: Els usuaris experts coneixen bé els conceptes de la tasca i la interfície i busquen que el seu treball es faci ràpidament. Exigeixen temps de resposta ràpids, dreceres per dur a terme accions amb poques pulsacions o seleccions i més funcionalitats. Les cadenes d'ordres, les dreceres mitjançant menús, les abreviatures i altres acceleradors són requisits.

Dissenyar per a un tipus és fàcil; és molt més difícil dissenyar per a tots. Quan s'han de tenir diverses classes d'ús en un sol sistema, l'estratègia bàsica és permetre l'aprenentatge en diverses capes. Aquestes capes s'han de veure reflectides en el nostre disseny de tasques, definint múltiples camins per a assolir un mateix objectiu, i definir guies dins del producte per tal d'ajudar als usuaris en el seu aprenentatge.

Per al disseny de les tasques, tindrem en compte els cinc estils d'interacció principals (manipulació directa, selecció de menús, ompliment de formularis, llenguatge de comandaments i llenguatge natural) i les "vuit regles daurades del disseny de la interfície" dels que parlarem més endavant. Per tant el procés de anàlisi de tasques seria:

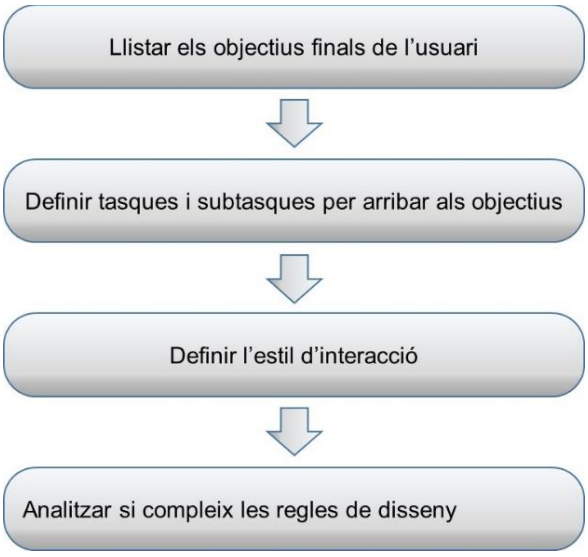


Figura 13 – Anàlisi de tasques

Per obtenir els objectius ens hem de centrar en el que vol l'usuari, pensar quines són les necessitats que té i que espera trobar-se en fer servir la nostra interfície. En aquest punt és molt aconsellable fer enquestes i entrevistes amb els usuaris que puguem, per tal d'aprofundir en la seva manera de pensar i en les necessitats que ells ens puguin revelar.
Un cop tinguem els objectius, hem de descriure tots els passos que s'han de fer per arribar a aconseguir aquest objectiu, com per exemple Objectiu identificar-se en una aplicació, tasques: obrir l'aplicació en el mòbil, clicar al botó d'usuari, introduir email i password, clicar al botó "go".
Haurem de fer aquest fil de tasques per a cada objectiu que identifiquem, per a cada manera diferent de fer-ho, tenint en compte cada perfil d'usuari.

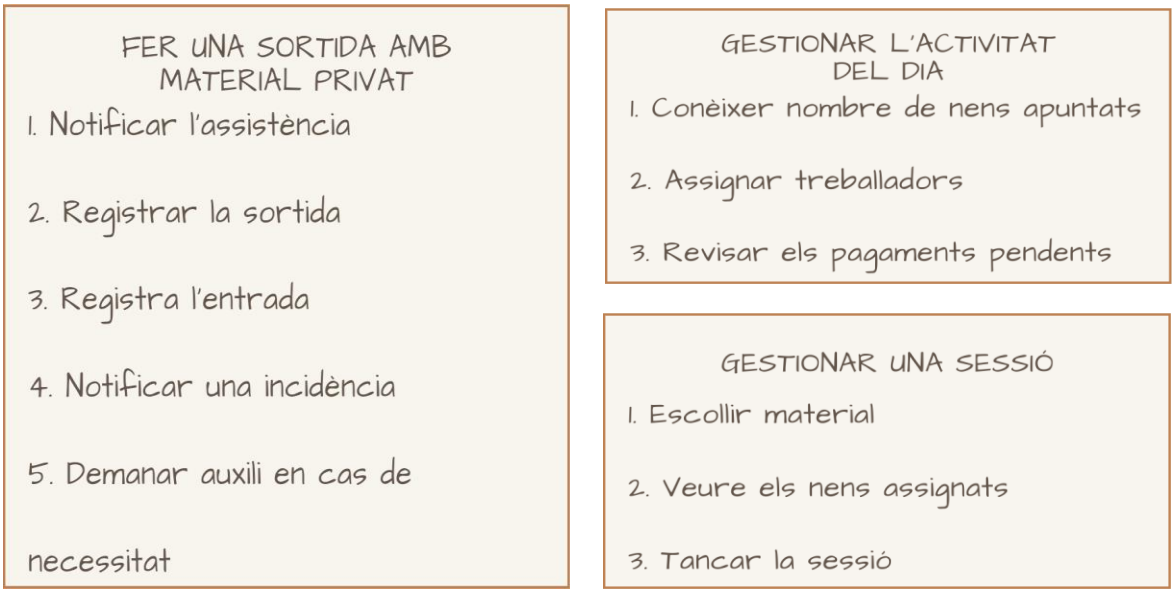


Figura 14 – Definició dels objectius

Un cop tenim els objectius i les subtasques, hem de triar quina és la millor manera d'estructurar el fil de tasques, analitzar si compleix amb totes les normes i com l'usuari interaccionarà amb la plataforma per realitzar-les. Els estils d'interacció elementals ens defineixen com estructurar la interfície i per això, conèixer els detalls és important:

Manipulació directa: Quan podem crear una representació visual efectiva de l'acció, les tasques dels usuaris es poden simplificar molt, perquè facilitem que l'usuari faci una associació directa i hi tingui ràpid accés. Entre altres, com a exemple trobem la metàfora de

l'escriptori de la qual ja hem parlat. Clicant, sigui amb el dit o amb el cursor, representacions visuals d'objectes i accions, els usuaris poden realitzar tasques ràpidament i poden observar els resultats immediatament (per exemple, arrossegar i deixar anar una icona a una paperera).

Selecció de menú: Als sistemes de selecció de menús, els usuaris llegeixen una llista d'elements, seleccionen la més adequada a la seva tasca i observen l'efecte. És important en aquest tipus d'interacció dissenyar controls d'error per evitar mals majors. Si la terminologia i el significat dels elements són comprensibles, els usuaris poden realitzar les seves tasques amb poc aprenentatge i memorització. El benefici més gran pot ser que hi hagi una estructura clara per a la presa de decisions, ja que totes les opcions possibles es presenten alhora.

Emplenament de formularis: Quan es requereix l'entrada de dades, la selecció del menú només, sol ser engorrosa i l'ompliment del formulari (també anomenat emplenar els espais buits) és adequat. Els usuaris veuen una visualització de camps relacionats, mouen un cursor entre els camps i introdueixen dades on vulguin. Amb l'estil d'interacció d'ompliment del formulari, els usuaris han d'entendre les etiquetes de camp, conèixer els valors admissibles i el mètode d'entrada de dades i ser capaços de respondre als missatges d'error.

Llenguatge de comandaments: Els llenguatges de comandament són aquells on es controla la interfície mitjançant comandes escrites que executen accions concretes prèviament definides. Com per exemple la línia de comandes d'un sistema operatiu. Per als usuaris freqüents, els llenguatges de comandaments proporcionen una forta sensació de control. Els usuaris aprenen la sintaxi i sovint poden expressar possibilitats complexes ràpidament, sense haver de llegir indicacions que distreuen.

Llenguatge natural: L'ús del llenguatge natural és aquell on la interfície interpreta el que l'usuari vol realitzar en funció d'una frase o frases que es diuen o s'escriuen. L'esperança que els ordinadors responguin adequadament a frases o frases arbitràries del llenguatge natural involucren molts investigadors i desenvolupadors de sistemes, malgrat l'èxit limitat fins ara.

La combinació de diversos estils d'interacció pot ser adequada quan les tasques siguin diverses, i pensar de forma acurada en quin estil utilitzarem en cada pas és vital per a definir l'arquitectura de la interfície.

Tot i que ho introduirem aquí, aquests principis són transversals en tot el procés de disseny i s'han de tenir presents en tot moment. Anomenats "regles d'or", són aplicables a la majoria de sistemes interactius. Aquests principis, derivats de l'experiència i refinats durant dues dècades per Ben Shneiderman com a principal pensador y difusor, s'han d'ajustar a cada disseny. No hi ha cap llista que sigui completa, però aquesta ha estat ben rebuda com una guia útil per a estudiants i dissenyadors.

- **Esforçar-se per la coherència.**
  Aquesta norma és la més freqüent, però seguir-la pot ser complicada perquè hi ha moltes formes de coherència. S'han de requerir seqüències consistents d'accions en situacions similars; S'hauria d'utilitzar terminologia idèntica en les indicacions, menús i pantalles d'ajuda; i el color, la composició, les majúscules, els tipus de lletra, etcètera, s'han d'utilitzar de forma coherent. Les excepcions, com ara la confirmació obligatòria de l'ordre de supressió o el no fer-se ressò de les contrasenyes, han de ser comprensibles i limitades en nombre.

- **Busqueu la usabilitat universal.**
  Reconèixer les necessitats d'usuaris diversos i dissenyar-ne la plasticitat, facilitant la transformació de contingut. Les diferències, els intervals d'edat, les discapacitats i la diversitat tecnològica d'experts novells enriqueixen tot l'espectre de requisits que guien el disseny. Si s'afegeixen funcions per a principiants, com ara explicacions i funcions per a experts, com ara dreceres i un ritme més ràpid, pot enriquir el disseny de la interfície i millorar la qualitat del sistema percebut.

- **Oferir comentaris informatius:**
  Per a cada acció dels usuaris, hi hauria d'haver una retroalimentació del sistema. Per a accions freqüents i menors, la resposta pot ser modesta, mentre que per a accions freqüents i importants, la resposta hauria de ser més important.

- **Dissenya diàlegs per obtenir un tancament:**
  Les seqüències d'accions s'han d'organitzar en grups amb un principi, mig i final. La informació al finalitzar un grup d'accions proporciona als operadors la satisfacció de la realització i un senyal per preparar-se per al següent grup d'accions. Per exemple, els llocs web de comerç electrònic permeten als usuaris de seleccionar productes a la caixa, finalitzant amb una pàgina de confirmació clara que completa la transacció.

- **Prevenir errors:**
  Sempre que sigui possible, s'ha de dissenyar el sistema de manera que els usuaris no puguin cometre errors greus; per exemple, eliminar els elements del menú que no són adequats i que no permeten caràcters alfabètics en camps d'entrada numèrics. Si un usuari comet un error, la interfície ha de detectar l'error i oferir instruccions senzilles, constructives i específiques de recuperació. Per exemple, els usuaris no haurien d'haver de tornar a escriure un formulari d'adreça de nom complet si introdueixen un codi postal no vàlid, sinó que s'han de guiar per reparar només la part defectuosa. Les accions errònies han de deixar l'estat del sistema sense canvis, o la interfície hauria de donar instruccions sobre com restaurar l'estat.

- **Permet la inversió fàcil de les accions:**
  En la mesura del possible, les accions han de ser reversibles. Aquesta característica alleuja l'ansietat, ja que l'usuari sap que es poden desfer errors, fomentant així l'exploració d'opcions poc conegudes. Les unitats de reversibilitat poden ser una sola acció, una tasca d'entrada de dades o un grup complet d'accions, com ara l'entrada d'un nom i d'un bloc d'adreces.

- **Suport del lloc intern de control:**
  Els operadors experimentats desitgen fortament sentir que s'encarreguen de la interfície i que la interfície respon a les seves accions. Les accions d'interfície sorprenents, seqüències tedioses d'entrades de dades, incapacitat d'obtenir o dificultat per obtenir la informació necessària i la incapacitat de produir l'acció desitjada, generen ansietat i insatisfacció. Gaines (1981) va capturar part d'aquest principi amb la seva regla d'evitar accions no causals i al instar a fer que els usuaris fossin els iniciadors d'accions més que els que responen a les accions.

- **Reduïu la càrrega de memòria a curt termini:**
  La limitació del processament d'informació humana a la memòria a curt termini (la regla general és que els humans poden recordar entre 5 i 9 blocs d'informació) requereix que es mantinguin les pantalles simples, la freqüència de moviment de la finestra s'ha de reduir i assignar un temps d'entrenament suficient per a codis mnemònics i seqüències d'accions.

Aquests principis s'han d'interpretar, perfeccionar i estendre per a cada entorn que dissenyem. Tenen les seves limitacions, però proporcionen un bon punt de partida per a dissenyadors mòbils, d'escriptori o web.

Així un cop haguem aplicat totes aquestes informacions, ens quedaran un seguit d'objectius amb les subtasques que podem realitzar de forma que siguin el més assequible possible per als usuaris i d'aquesta manera el nostre projecte tingui el millor resultat possible.

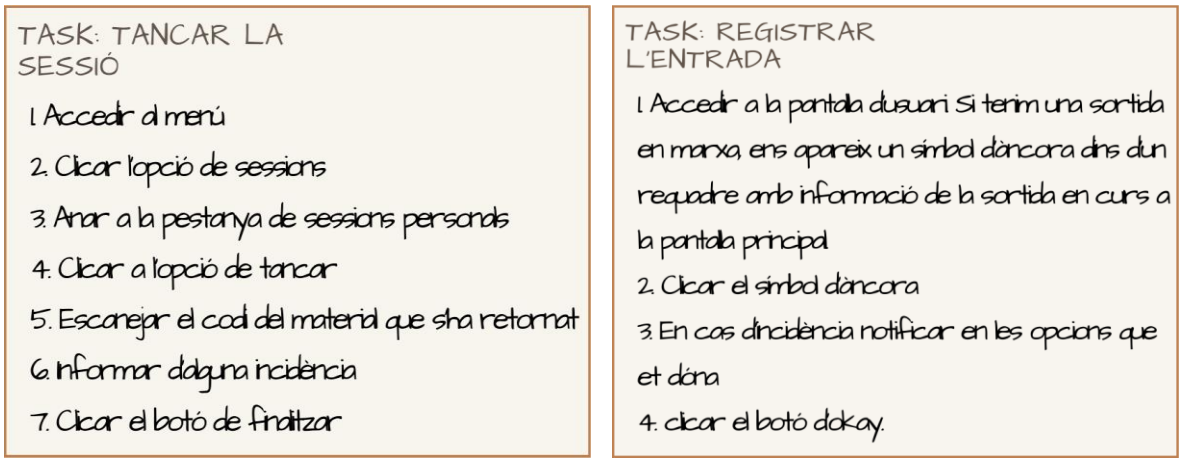


Figura 15 – Especificació de les tasques

4.4 Prototipat

En arribar aquest punt ens estem apropant al final del disseny del nostre programa, més concretament a la seva interfície i objectius.

Ara que hem reunit els perfils d'usuaris, les tasques a desenvolupar i les necessitats generals comencem a pensar que tenim una idea prou formada per a llançar-nos a programar i desenvolupar la nostra aplicació. Malgrat això podem basar la resta del treball en premisses o assumpcions que fem nosaltres que no siguin les mateixes que farien els usuaris i acabar amb un producte final molt distanciat del que necessita l'usuari.

Si aquest error el corregim al principi del desenvolupament, tindrà un cost en temps i diners mínim, però serà impossible modificar-lo del producte finalitzat sense haver de començar de nou.

En lloc d'endevinar de què tracten totes les especificacions de requisits i les necessitats dels usuaris, abans de passar molt de temps dissenyant i desenvolupant un sistema que, basat les especulacions que hàgim fet, pugui ser força equivocat, hem de consultar les nostres idees amb els usuaris.

Ens assegurarem de posar a prova les nostres idees i la millor manera de fer-ho és fent que els usuaris posin a prova prototips.

Es pot utilitzar un prototip, que és un disseny experimental, generalment incomplet, de dues maneres.

- Durant el procés de disseny es pot utilitzar per comunicar i compartir idees entre el dissenyador d'UI i els usuaris i les parts interessades, de manera que es poden aclarir els requisits.
- Més endavant en el procés de disseny es pot utilitzar per explorar i demostrar la interacció i la coherència del disseny.

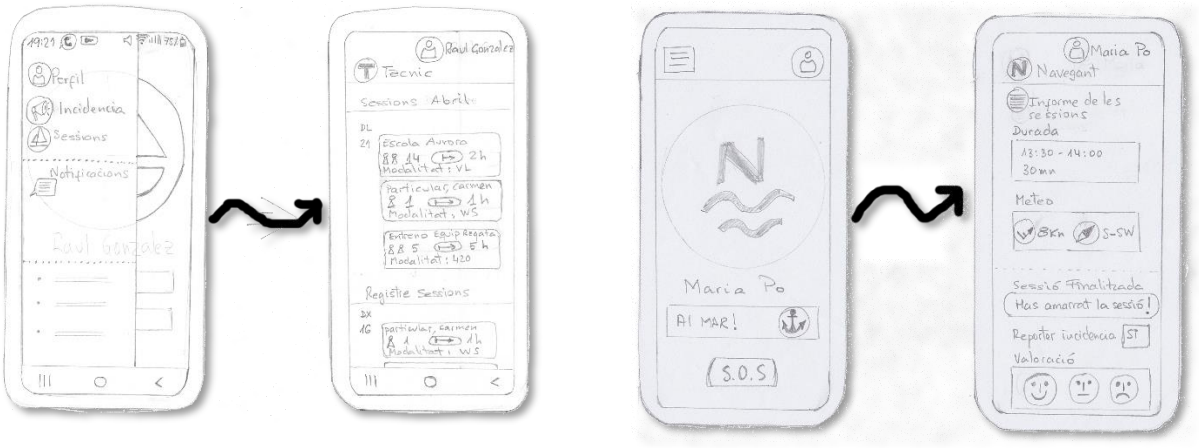


Figura 16 – Prototipat manual (Sketching)

El prototipat no és l'inici del codi real, sinó una façana amb la qual l'usuari pugui interaccionar en diferents etapes del disseny, i poder obtenir informació retroactiva d'aquesta interacció. Aquest prototip es converteix normalment en la base del producte**.** Aquest procés canvia la robustesa i la vida útil del producte. Per tant, podem dir que els propòsits del prototipat són els següents:

- Per comprovar la viabilitat d'idees amb els usuaris.
- Per comprovar la utilitat de l'aplicació.
- Permetre als usuaris contribuir al disseny de l'aplicació.
- Permetre als usuaris provar idees.
- Per validar els requisits (és a dir, per revelar requisits inconsistents o incomplets).
- Negociar requisits.

Durant el desenvolupament del projecte, com ja hem comentat en altres parts d'aquest treball, fem diversos cicles d'aproximació al disseny i per tant també entrarem repetides vegades en aquesta fase del disseny.

Durant tota l'evolució passarem d'un prototip de baixa fiabilitat a un prototip d'alta fiabilitat. Els prototips de baixa fidelitat generalment es basen en paper i inclouen esbossos, maquetes de pantalla i guions gràfics. Els prototips d'alta fidelitat, basats en programari, proporcionen una versió funcional del sistema amb la qual els usuaris poden interactuar i com a tal, mostren el disseny de la interfície d'usuari i la seva navegació. En aquest punt el disseny i la implementació se superposen i els prototips d'alta fidelitat solen acabar sent la base del codi del producte final.

Un prototip ha de tenir en compte tots els següents punts essencials:

- Ha de ser visual i poder-se tocar i interactuar amb tranquilitat pels usuaris.
- Ha d'incloure les funcionalitats bàsiques.
- Hem d'estar oberts a canvis i idees noves i no quedarnos encallats en la nostre idea inicial.
- S'han d'utilitzar guies d'estil per tal d'unificar el projecte i crear un aprenentatge ràpid.
- No tinguem por a l'error, com més ens equivoquem més aprendrem.

4.4.1 Prototip de baixa fidelitat

Els prototips de baixa fidelitat generalment es basen en paper i inclouen esbossos, maquetes de pantalla i guions gràfics. Es poden crear a mà, però també es poden crear amb programes d'edició gràfica. Els prototips de baixa fidelitat són útils en la fase de recollida de requisits del disseny de la interfície. Es poden utilitzar com a mitjà de comunicació entre dissenyador i els

usuaris i parts interessades. Els prototips de baixa fidelitat també ajuden als usuaris a articular el que volen i necessiten d'un sistema, ja que els serà més fàcil parlar d'alguna cosa visual i concreta que no d'idees conceptuals (o abstractes), que poden ser més difícils de compartir. Es poden utilitzar prototips de baixa fidelitat per il·lustrar idees de disseny, dissenys de pantalla i alternatives de disseny. Si bé poden proporcionar als usuaris alguna indicació de l'aspecte de la interfície d'interès, només proporcionen un detall limitat de com funcionarà la interfície o de com es poden dur a terme les tasques.

Els punts bàsics que s'han d'incloure en un prototip de baixa fidelitat són:

- Les pantalles bàsiques de la interfície amb les funcionalitats.
- Les tasques més importants.

Tenim diferents tècniques possibles com l'sketching que té molts usos en el procés de disseny de l'interficie d'usuari. Inicialment es pot utilitzar com una forma d'ajudar a determinar què es vol i es necessita en un sistema. Després de llegir les especificacions, podem dibuixar la interpretació que en fem. A mesura que anem avançant en el disseny i contrastant el croquis amb el client o usuari, podrem saber si teníem la idea ben assimilada o hem de fer alguns ajustaments. Intentar incloure molta informació rellevant i veure com va encaixant visualment ens pot proporcionar pistes de com hem d'estructurar la interfície en el futur.

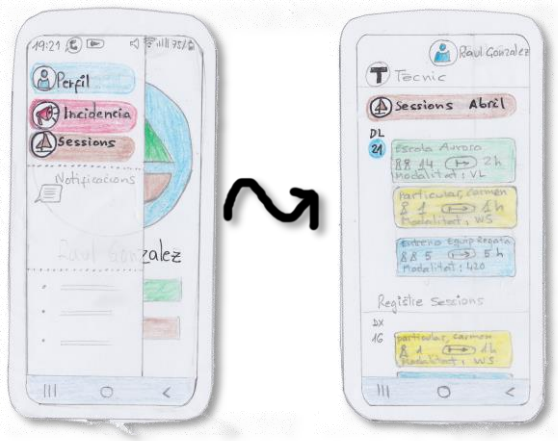


Figura 17 – Iteració del prototip

Una altra tècnica pot ser els "*Screen Mockups*" o maquetes de pantalla. Es poden produir fàcilment mitjançant flipcharts, pissarres o transparències generals i una varietat de bolígrafs de diferents colors. Un altre mètode útil per a les maquetes de pantalla és utilitzar no només bolígrafs de colors diferents en un fulletó o una pissarra, sinó notes enganxoses de diferents colors i formes diferents per representar ginys com botons i menús. L'avantatge d'utilitzar

notes enganxoses com a ginys és que es poden moure i canviar-los fàcilment com a resposta als comentaris dels usuaris que estan provant les idees.
En aquesta tècnica, ens hem de centrar en l'estructura, en la imatge mental que tinguem de la nostra interfície.

L'avantatge més gran dels prototips de baixa fidelitat és que són barats, ràpids de produir i que es poden canviar fàcilment. Tots els problemes descoberts al començament del procés de disseny poden ser corregits abans que el disseny no hagi avançat massa en el desenvolupament per permetre realitzar canvis essencials. Tot i que els prototips de baixa fidelitat no es poden "executar" de la manera que poden fer-ho els prototips d'alta fidelitat, el llibre de Snyder (2003) explica com aconseguir que els usuaris utilitzin prototips de paper amb l'avaluador o moderador actualitzant-lo el prototip en el moment en què l'usuari requereixi.

4.4.2 Prototip d'alta fidelitat

Els prototips d'alta fidelitat han de ser basats en programari i proporcionen una versió funcional del sistema amb la qual els usuaris poden interactuar. Com a tal, mostren el disseny de la interfície d'usuari i la seva navegació. Si l'usuari selecciona una ordre de menú, com ara obrir una finestra o trucar un quadre de diàleg, veurà executada la comanda; els missatges, com ara els missatges d'error, es mostraran segons correspongui. L'usuari pot experimentar l'aspecte del sistema final.

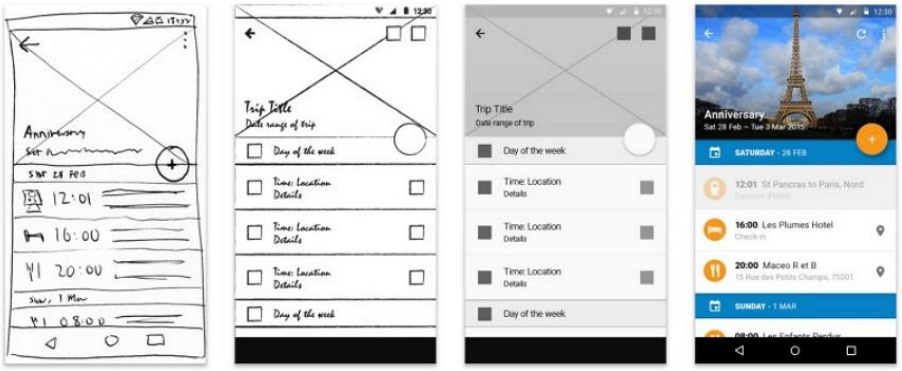


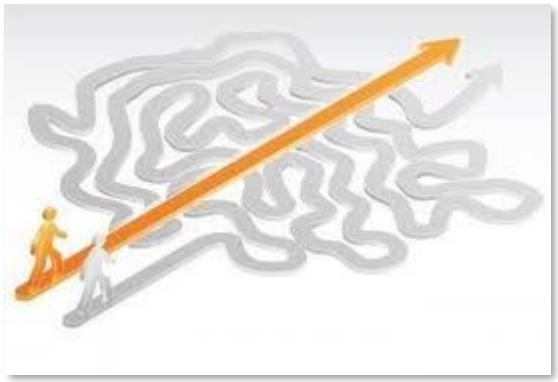
Figura 18 – Evolució del prototipat

És en aquest punt en què l'usuari experimentarà amb el prototip com si estigués treballant amb la versió final, podem realitzar proves d'utilització i recollir informació molt útil per a millorar el sistema. En essència, un prototip d'alta fidelitat sembla i es comporta com si fos el producte final i inclús es pot utilitzar com a eina per comercialitzar el producte final. Abans fer un prototip final requeria moltes hores de feina i un cost molt alt, però actualment hi ha eines que permeten definir interaccions entre "*screen mockups*" per exemple, i faciliten la feina.
Garantir que un sistema proposat tingui la funcionalitat necessària per a les tasques que els usuaris volen fer és una part important de la recollida de requisits i de l'anàlisi de tasques. Si és factible, els prototips d'alta fidelitat poden complir un paper important en la prova de dissenys amb usuaris i en la validació dels requisits. Per exemple, es pot descobrir la funcionalitat que falta, es poden provar seqüències de tasques i es pot valorar la significació de les icones.

4.5 Anàlisis d'usabilitat

En desenvolupar un projecte, hem de pensar en tots els factors que influencien als nostres usuaris, i com ja hem comentat anteriorment, els usuaris poden ser i són molt diversos.
De diversitats en tenim de molts tipus, estan les diversitats greus com podria ser una ceguera o una sordesa i les diversitats més petites que no acostumem a veure com ara una velocitat diferent d'escriptura o un coneixement tècnic desigual. Hem de parar atenció a totes elles.
Pensar en la usabilitat del nostre producte no hauria de ser un punt concret en el procés de disseny sinó formar part de forma integral en cada un dels passos. En aquest apartat exposarem els punts més importants a tenir en compte i intentarem donar unes pautes per a poder millorar el nostre disseny en aquest àmbit.

Com diuen Jeffrey Rubin, Dana Chisnell, Jared Spool en el seu llibre "Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests-Wiley" (2008), el que fa que alguna cosa sigui útil és l'absència de frustració en utilitzar-lo. Així podríem dir que la definició d'usabilitat seria "quan un producte o servei sigui realment útil, l'usuari pot fer el que vulgui fer, de la manera que ell o ella espera poder fer-ho, sense cap impediment, vacil·lació o preguntes". És a dir que el nostre producte ha de ser útil, eficient, eficaç, satisfactori, de fàcil aprenentatge i accessible.



La **utilitat** es refereix al fet que un usuari pugui aconseguir els seus objectius i el grau en què ho assolim determinarà la voluntat de l'usuari a utilitzar el producte en la seva totalitat. Si un sistema no aconsegueix els objectius específics d'un usuari concret, no s'utilitzarà fins i tot si es proporciona de forma gratuïta. Curiosament, la utilitat és l'element que s'oblida més sovint durant el desenvolupament d'un producte.

L'**eficàcia** és la rapidesa amb què es pot assolir l'objectiu de l'usuari de manera precisa i completa i sol ser una mesura del temps.

L'**efectivitat** fa referència a si el producte es comporta de la manera com els usuaris ho esperen i a la facilitat amb què els usuaris poden utilitzar-lo per fer el que pretenen. Generalment es mesura quantitativament amb la taxa d'error.

L'**aprenentatge** és una part de l'eficàcia i té a veure amb la capacitat de l'usuari de fer servir el sistema fins a uns nivells de competència definits després d'alguna quantitat i període de formació predeterminats. També pot referir-se a la capacitat d'usuaris poc freqüents de rellevar el sistema després de períodes d'inactivitat.

La **satisfacció** fa referència a les percepcions, sentiments i opinions de l'usuari del producte, generalment capturades a través de preguntes per escrit o de forma oral.
A més a més de tots aquests atributs, també hem de parar atenció en un factor extra, l'accessibilitat, entesa com que els productes puguin ser utilitzats per persones amb discapacitat. Podríem dir que l'accessibilitat és un món en si sol i podem trobar infinitat de tècniques, protocols, reculls de normes que ens poden facilitar la feina.

Els objectius i les tasques que hem explicat anteriorment en aquest treball, es defineixen normalment en termes quantificables d'un o més d'aquests atributs. De totes maneres la usabilitat no és una ciència exacte i moltes vegades és més aviat un art. Posar-hi molta cura i temps pot marcar la diferència entre un èxit o un fracàs.

És per això que val la pena mencionar que en l'àmbit internacional es defineixen protocols i avaluacions oficials per a mesurar la usabilitat de sistemes. Per exemple la ISO 9241-210:2019 Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems, proporciona orientacions i enumera els principis i activitats essencials per al disseny centrat en humans (usuaris) per aconseguir la usabilitat en sistemes.

La veritable usabilitat és invisible. Si alguna cosa va bé, no ho notem. Afortunadament, però, hi ha mètodes habituals i fiables per avaluar on el disseny contribueix a la usabilitat i on no, i per jutjar quins canvis cal fer als dissenys perquè un producte pugui ser prou útil per sobreviure o fins i tot prosperar al mercat.

4.5.1 Els mètodes d'avaluació

Per a crear un producte usable no hi ha prou en pensar-hi com a programadors, és molt probable que de totes formes tinguem una opinió esbiaixada. Com proposa Jeffrey Rubin en el seu llibre, tenim els següents mètodes per avaluar i millorar la usabilitat del nostre producte:

Recerca Etnogràfica
La investigació etnogràfica utilitza tècniques d'antropologia. Es tracta d'observar els usuaris al lloc on normalment farien servir el producte (per exemple, la feina, la casa, la barra del bar,

etc.) per obtenir dades sobre com els usuaris realitzen el seu objectiu, quines tasques i objectius han relacionat amb el producte i el context en què treballen per assolir els seus objectius. A partir d'aquesta investigació qualitativa, es poden millorar els perfils d'usuaris, persones, escenaris i descripcions de tasques per a millorar-les en futures iteracions del cicle de disseny.

Disseny participatiu

Més que una tècnica és una representació del UCD (User Centered Design), el disseny participatiu col·loca un o més usuaris representatius del propi equip de disseny en el centre de la investigació. Aquest enfocament empeny l'usuari final al cor del procés de disseny des del començament del projecte, aprofitant els coneixements, les habilitats i fins i tot les reaccions emocionals de l'usuari.

Focus Group Research

Totes les investigacions de grups de treball utilitzen la participació simultània de més d'un participant, un factor clau per diferenciar aquest mètode de moltes altres tècniques. Els conceptes que els participants avaluen en aquestes sessions de grup es poden presentar de forma preliminar, com ara dibuixos en paper i llapis, storyboards i / o prototips o models plàstics basats en pantalla. L'objectiu és identificar com d'acceptables són els conceptes, de quines maneres són inacceptables o insatisfactòries i com es poden fer més acceptables i útils. Amb aquest mètode busquem identificar les sensacions dels usuaris en interaccionar amb el producte, i així poder tenir un millor coneixement de la satisfacció que els hi genera.

Enquestes

Administrant enquestes podeu començar a comprendre les preferències d'una àmplia base d'usuaris sobre un producte existent o potencial. L'enquesta pot utilitzar mostres més grans per generalitzar a tota una població. Per exemple, les qualificacions de Nielsen, una de les enquestes més famoses, s'utilitzen per prendre decisions empresarials multimilionàries per a una població nacional en funció de les preferències d'unes 1500 persones. Les enquestes es poden utilitzar en qualsevol moment del cicle de vida, però s'utilitzen més sovint en les primeres etapes per entendre millor l'usuari potencial.

Walk-Throughs

En aquest mètode, el dissenyador responsable del treball guia als usuaris i als membres de l'equip, a través de tasques reals de l'usuari (de vegades fins i tot fent el paper per a l'usuari), mentre que un altre membre de l'equip registra dificultats o preocupacions de l'equip. Es

registra tot el procés i les dificultats del camí, amb l'objectiu d'identificar els punts febles que poden fer una tasca no usable.

Prototips de paper

En aquesta tècnica es mostra als usuaris l'aspecte d'un producte en paper i se'ls fa preguntes al respecte o se'ls demana que responguin d'altres maneres. Per saber si el flux de pantalles o de pàgines que hi ha planificades és compatible amb les expectatives dels usuaris. El valor del prototip de paper és que es pot recopilar informació crítica de forma ràpida i econòmica. Es pot comprovar aquelles funcions que haurien de ser intuïtives i que no ho són, abans que s'hagi escrit una línia de codi. A més, es pot fer servir la tècnica una vegada i una altra amb un mínim de recursos.

Avaluació experta o heurística

Les avaluacions d'experts consisteixen en una revisió d'un producte o sistema, generalment per un especialista en usabilitat o especialista en factors humans que té poca o cap participació en el projecte. L'especialista realitza la seva revisió d'acord amb els principis d'usabilitat acceptats (heurística) del cos de recerca, literatura de factors humans i experiència professional prèvia. El punt de vista és el de la població objectiu específica que utilitzarà el producte. Un especialista " doble ", és a dir, algú expert en principis d'usabilitat o factors humans, així com expert en l'àrea de domini (com ara serveis sanitaris, serveis financers, etc., segons l'aplicació), o en la tecnologia particular utilitzada pel producte, pot ser més eficaç que un sense aquest coneixement.

Prova d'usabilitat

El test d'usabilitat, utilitza tècniques per recollir dades empíriques i observar els usuaris finals representatius que utilitzen el producte per realitzar tasques realistes. Les proves es divideixen en dos enfocaments principals. El primer enfocament consisteix en proves formals realitzades com a veritables experiments per confirmar o refutar hipòtesis específiques. El segon enfocament, menys formal però encara rigorós, utilitza un cicle iteratiu de proves destinades a exposar deficiències d'usabilitat i a poc a poc donar forma o modelar el producte en qüestió.

Estudis de seguiment

Un estudi de seguiment es produeix després del llançament formal del producte. La idea és recopilar dades per al proper llançament, mitjançant enquestes, entrevistes i observacions. Els estudis de seguiment estructurats són probablement les valoracions més eficaces i

precises de la usabilitat, perquè l'usuari, el producte i l'entorn reals estan sempre al seu "hàbitat" i interaccionen entre ells.

4.5.2 El procés de testeig

Les maneres de testejar un producte soń infinites i la forma de dur-ho a terme també. Aquí descrivim un possible sistema per fer-ho de forma estructurada.

Figura 19 – Elaboració del test plan

Desenvolupar el "test plan"

El "test plan" és la base de tota la prova. Explica el com, el quan, l'on, a qui i el perquè del nostre test d'usabilitat. Els formats del "test plan" variaran segons el tipus de prova i el grau de formalitat requerit. Seguidament explicarem els apartats típics a incloure, juntament amb una descripció de cadascun.

- Propòsits i objectius del test: Per a aquesta part del document, s'ha de descriure a un nivell general les raons per realitzar aquesta prova en aquest moment.

- Preguntes de recerca: En aquest punt inclourem totes les preguntes que volem satisfer amb la resolució dels tests. Com per exemple si el nostre producte és usable.
- Característiques del participant: Aquí hem de descriure quins participants han de realitzar cada una de les proves i quines característiques tenen.
- Descriure el mètode o mètodes: En aquest apartat s'ha de descriure quin mètode usarem i quina resposta vol resoldre. També inclourem quines dades hem de recollir en l'informe final.
- Llista de tasques: En aquest punt distribuirem la feina com ara preparar material, buscar als participants, executar el test o recollir la informació.

Planificar els participants

Conèixer tot sobre els teus participants ajudarà a controlar molt més la mostra. També és important identificar el lloc el moment i el context en què vols realitzar les proves i organitzar als participants per a ser efectiu. Si han d'esperar molt o estan avorrits el test es pot veure afectat.

També és molt important compensar als nostres participants d'alguna manera, sigui econòmica o no. I no ens podem oblidar de protegir legalment les informacions que obtinguem. Per això és de vital importància fer firmar als usuaris documents on cedeixin les dades i les informacions que obtinguem del test.

Realització de les sessions de test

Realitzar una sessió de test requereix de molta preparació i de tenir clar l'objectiu, el procediment i les dades a recol·lectar. El moderador de la sessió ha de ser pacient i deixar que els usuaris interactuïn amb el producte intentant interferir el mínim possible.

Analitzar la data

Recollir la informació, netejar-la i mostrar-la de forma entenedora, és un dels passos més importants. Si la nostra informació és extensíssima i difícil de gestionar és molt probable que no ens permeti entendre on està el problema i no ens aporti solucions.

Fer informe final de disseny

En aquesta part final, hem de col·locar tota la informació de forma endreçada, amb les conclusions a les preguntes que ens hem fet en el "test plan". També hauríem de suggerir millores per a les següents iteracions del disseny i incloure tota la informació del punt de partida, és a dir, perfils d'usuaris, tasques, prototips i llavors avaluacions i millores.

4.6 Finalitzar la iteració

Un cop hem realitzat totes les tasques, hem donat voltes al nostre disseny i hem avaluat la seva usabilitat, ens hem de repassar els resultats, posar en context el seu significat i extreure conclusions que tinguin una incidència directa sobre el disseny del nostre producte.

Els resultats que hem obtingut ens proporcionen informació vital per a continuar amb els procés de disseny, però també haurem de saber quan i com finalitzar les iteracions. Amb aquesta finalitat haurem de crear un document detallat que expliqui totes les decisions de disseny, les possibles millores a testejar en futures iteracions i les especificacions de la interfície.

Depèn de l'equip de disseny o del responsable, decidir quan es té una versió prou bona per iniciar la implementació del projecte, traslladant el document que en surti, malgrat que no sigui el definitiu, a l'equip d'implementació.

Amb aquest document, també s'iniciarà enèsima iteració, introduint les millores proposades i millorant el funcionament i la usabilitat de la nostra aplicació. Cada iteració ens aproximarà a un resultat més eficient. Amb les dades que hàgim obtingut realimentarem cada un dels passos, fins a poder tornar a testejar la usabilitat i haver finalitzat la nova iteració.

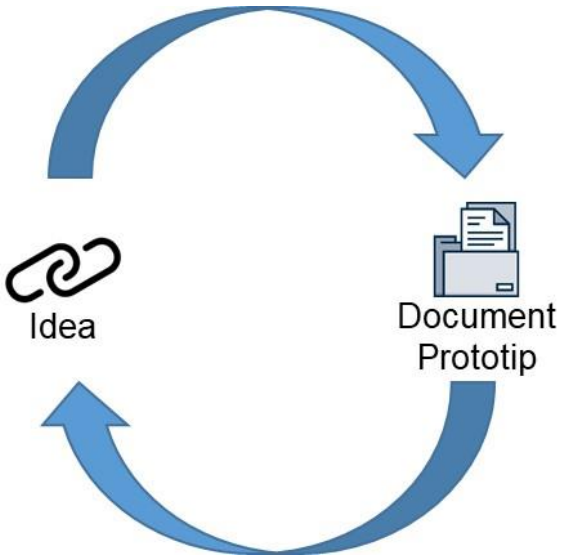


Figura 20 – Cicle de disseny

5 Bloc 2 - Implementació

Implementar un producte requereix molts coneixements tècnics que ens és impossibles recollir en aquesta guia. Les habilitats com a programador, com a tècnic en sistemes, coneixements extensos en tecnologies de base de dades i moltes altres més, són les que ens permeten dur a terme una bona implementació. En aquest bloc intentarem introduir els coneixements bàsics sobre el procés i els punts claus mínims en termes de qualitat de disseny, seguretat, patrons de desenvolupament i metodologies de treball.

Com a programadors, penso que tenim tendència a llençar-nos de cap a la programació de les nostres aplicacions sense meditar de forma adequada i amb prou profunditat tots els passos. És per això que em sembla interessant tenir una estructura a seguir per a dur a terme, i saber-se organitzar, el procés de desenvolupament. D'aquesta manera si creem un patró d'execució, sempre ens assegurarem de no saltar-nos cap pas i poder tenir al final un resultat molt més robust i funcional.

Per a parlar d'una bona implementació, també hem de saber quins són els nostres objectius i quines les nostres limitacions. Per exemple, si requerim dades sensibles als nostres usuaris, serà imprescindible treballar més en la seguretat de la nostra arquitectura, però si, al contrari, no tenim dades o el nostre producte córrer localment sense connexió a la xarxa, podrem relaxar-nos en aquest àmbit i prioritzar en la velocitat i comoditat de l'usuari.

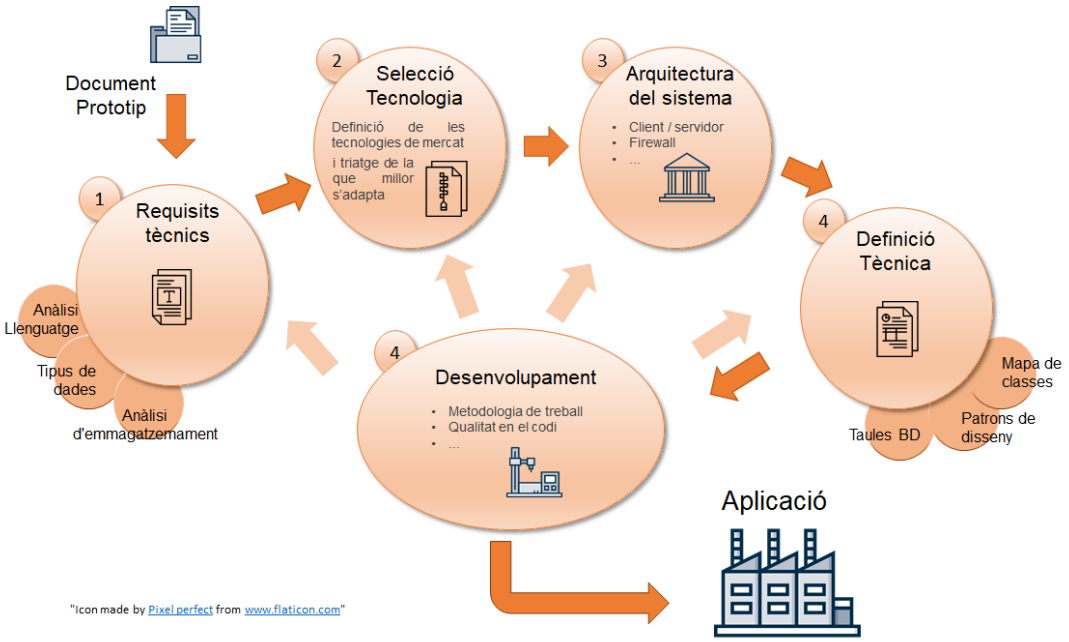


Figura 21 – Esquema del Bloc 2

Totes aquestes decisions han d'estar fonamentades en criteris tècnics que s'han de decidir al principi del procés i que afectaran molt en un futur desenvolupament. Si cometem un error a l'inici, si no desenvolupem seguint els patrons recomanats o si ens oblidem de la seguretat, ens costarà diners, esforç i temps al final del projecte, quan es faci evident l'error. Ens pot portar a descartar tot el que hem fet i començar de nou o arriscar-nos a produir un producte defectuós.

5.1 Contingut del bloc

Aquest bloc comença amb el document de disseny que s'ha generat en el bloc 1 i pretén tenir per sortida el producte final (o una demo). També pretén recollir estàndards i informació necessària per a crear un fil conductor que ens guiï en el procés de desenvolupament.
Per a cada tasca concreta definida en el document de disseny, analitzarem els seus requeriments tècnics, i quan tinguem una visió global haurem de definir l'arquitectura del nostre sistema i fer un anàlisis de les tecnologies que aporten més punts a favor al nostre sistema en funció dels requisits.
Coneixent les tecnologies ens tocarà dissenyar de forma concreta i fer les taules de classes, les taules de les bases de dades, etz. I finalment posar-se a programar el producte.
Per a dur a terme tot això ens serà necessari tenir l'ull posat en diferents aspectes, com la seguretat, els patrons de disseny i les metodologies de treball.
En cada un dels apartats es parlarà sobre la millor manera d'organitzar-se per obtenir els resultats apropiats, així com la manera d'identificar-los. També parlarem de les tècniques més usades i que proporcionen millors resultats.
Per tant en aquest bloc trobarem els punts més importants en la seguretat, el disseny consistent i la metodologia per organitzar el treball, així com els passos a seguir per a estructurar de forma correcta tot el disseny tècnic del nostre projecte.

5.2 Els requisits

Per a poder iniciar la fase de desenvolupament hem d'identificar els requisits de cada una de les nostres tasques o accions dins de la plataforma així com els que tindrem a nivell de dades o sistema. Quan parlem de requisits ens referim als requisits no funcionals.
En el desenvolupament de software podem distingir entre dues categories de requisits, els funcionals i els no funcionals.

Els requisits funcionals (RF) són aquells que fan referència al que ha de fer el programa, és a dir respostes o interaccions amb els usuaris i altres sistemes. En alguns casos, els requisits funcionals dels sistemes també estableixen explícitament el que el sistema no ha de fer. L'especificació dels requisits funcionals d'un sistema ha de ser completa i coherent. Completa vol dir que tots els serveis sol·licitats per l'usuari i / o un altre sistema estan definits. La coherència significa que els requisits no tenen una definició contradictòria.
En aquest treball hem analitzat i definit aquests requisits en el bloc de disseny amb l'objectiu de testejar-los amb els usuaris i identificar les seves mancances.
Els requisits no funcionals (RNF), són requisits que no es refereixen directament a les funcions específiques subministrades pel sistema (característiques d'usuari), sinó a les propietats del sistema: rendiment, seguretat, disponibilitat. També defineixen restriccions de sistema com ara la capacitat dels dispositius d'entrada / sortida i la representació de les dades utilitzades en la interfície de sistema.
Per a expressar-ho de forma senzilla podríem dir que els RF fan referència al que ha de fer el sistema i els RNF fan referència al com ho ha de fer. En aquest apartat hem de generar una llista de requisits no funcionals per cada una de les tasques per a poder identificar totes les necessitats que tenim a nivell tècnic per a dur a terme el projecte.
Hi ha diferents tipus de requeriments que es poden classificar d'acord amb les seves implicacions.

- Requeriments del producte. Especifiquen el comportament del producte; com els requeriments en la rapidesa d'execució de sistema i quanta memòria es requereix; els de fiabilitat que fixen la taxa de falles perquè el sistema sigui acceptable i els de portabilitat.
- Requeriments organitzacionals. Es deriven de les polítiques i procediments existents en l'organització del client i en la del desenvolupador: estàndards en els processos que s'han d'utilitzar; requeriments d'implementació com els llenguatges de programació o el mètode de disseny a utilitzar, i els requeriments de lliurament que especifiquen quan es lliurarà el producte i la seva documentació.

- Requeriments externs. Es deriven dels factors externs al sistema i del seu procés de desenvolupament. Inclouen els requeriments d'interoperabilitat que defineixen la manera en què el sistema interactua amb els altres sistemes de l'organització; els requeriments legals que s'han de seguir per assegurar que el sistema operi dins de la llei, i els requeriments ètics. Aquests últims són impostos a sistema per assegurar que serà acceptat per l'usuari.

5.2.1 Determinar els requisits

Kotonya i Sommerville (1998) defineixen un requisit com a "declaració d'un servei o restricció del sistema". La tasca que hem de realitzar a la fase de determinació dels requisits és definir, analitzar i negociar els requisits amb els clients. Es tracta d'una exploració de conceptes, intentant identificar les necessitats de cada tasca. Una tècnica final de la fase de requisits és fer servir el prototipat ràpid de la solució final de disseny de manera que es puguin aclarir requisits difícils i evitar malentesos. L'anàlisi de requisits inclou les negociacions entre desenvolupadors i dissenyadors. Aquest pas és necessari per eliminar els requisits contradictoris i sobreposats i ajustar-se al pressupost del projecte.
El producte de la fase de requisits és un document de requisits. Es tracta sobretot d'un document de text narratiu amb alguns esquemes i taules informals.

5.2.2 Especificació de requisits

La fase d'especificació dels requisits comença quan els desenvolupadors modelen els requisits mitjançant un mètode determinat (com ara UML). S'utilitza alguna eina per a introduir, analitzar i documentar els models. Com a resultat, el document de requisits s'enriqueix amb models gràfics i informes. En essència, un document d'especificacions substitueix el document de requisits.
Les dues tècniques d'especificació més importants en l'anàlisi orientada a objectes són els diagrames de classes i els diagrames de casos d'ús. Es tracta de tècniques per a dades i especificacions de funcions. En un document d'especificació típic també es descriuen altres requisits, com ara rendiment, "aspecte", usabilitat, manteniment, seguretat i requisits polítics i legals. Els models d'especificacions es poden superposar. La superposició permet visualitzar la solució proposada des de molts angles, de manera que es ressalten i analitzen aspectes específics de la solució. També es comprova acuradament la coherència i l'exhaustivitat dels requisits.

5.3 Selecció de les tecnologies

Amb els requisits de cada tasca ens toca escollir les tecnologies que farem servir per a desenvolupar el projecte. Haurem de fer una cerca de les tecnologies existents per a saber quines compleixen més acuradament amb la nostra aplicació de forma global.
Més endavant parlarem de l'arquitectura a nivell de sistema, per això en aquest apartat hem d'escollir pensant només en els requisits i de la manera més imparcial i lògica possible. En aquesta decisió afecten tant els requisits com les capacitats pressupostàries i de temps. Per exemple, si volem fer un blog, el podem fer des de zero, però ens consumirà molt de temps, com a opció alternativa podem contractar un sistema de gestió de continguts, però ens pot consumir part del pressupost.
És la nostra tasca escollir la millor opció tenint en compte totes les variables. D'aquesta forma ens anirem adaptant a les especificacions que ens vagin arribant durant les diferents iteracions.

5.3.1 Llenguatges

És molt probable que com a programadors tinguem alguna preferència en matèria de llenguatges de programació, no obstant això, per a un projecte hem d'apartar-les per a escollir els més aptes i que compleixin el nombre més gran de requisits.
Cada llenguatge té diferents propietats i característiques que hem de conèixer o recercar per tal de fer la tria més encertada. Per exemple si necessitem un llenguatge que puguem optimitzar podríem escollir C++, però si volem un llenguatge fàcil i portable, podríem escollir Java.
No existeix una recepta màgica que ens porti al llenguatge òptim, l'única opció és tenir un coneixement ampli i fer-se les preguntes adequades. Quines necessitats tinc? Quin tipus de programa estic fent? Web, mòbil? Necessito poder escalar el programa? Quina tecnologia domino ja?
Aquesta última pregunta, tot i que no hauria de determinar els resultats, és un factor molt important. Un altre factor que hem de mirar molt atentament, sobretot si parlem d'aplicacións web, és el framework que farem servir.

5.3.2 Frameworks

Un framework en el desenvolupament de programari és una estructura conceptual i tecnològica que ajuda al programador, normalment, amb artefactes o mòduls concrets de programari, que pot servir de bastida per a l'organització i desenvolupament de programari.

Pot incloure suport de programes, biblioteques, i un llenguatge interpretat, entre d'altres eines, per així ajudar a desenvolupar i unir els diferents components d'un projecte. Representa una arquitectura de programari que modela les relacions generals de les entitats del domini, i proveeix una estructura i una especial metodologia de treball, la qual estén o utilitza les aplicacions del domini.

Escollir un framework adequat per al projecte que volem dur a terme és de vital importància, ja que ens facilitarà la feina en gran mesura. En tenim de molts tipus, des de molt senzills, fins de molt complexos, que s'ocuparan de la gestió de gairebé totes les tasques estandarditzades.

5.3.3 Base de dades

També haurem d'escollir les bases de dades. Per a tal fi hem d'identificar les necessitats que tenim en funció dels requisits i d'aquesta manera buscar la base de dades que més s'ajusti a les nostres necessitats. És molt possible que si el nostre projecte és una mica gran haguem de fer servir més d'una base de dades de tipus diferents.

Figura 22 – Llenguatges i tecnologies

5.4 L’arquitectura del sistema

Els coneixements per al disseny de les arquitectures de sistemes són més àmplis que els necessaris per al disseny de programari, tot i que el disseny de programari sol tenir el protagonisme. El disseny del sistema inclou la descripció d'una estructura del sistema i una descripció dels seus components, en conseqüència, el disseny del sistema de vegades es separa en disseny arquitectònic i en disseny detallat.

El document d’especificació de la fase anterior llista tots els requisits que ha de satisfer el nostre producte. Aquestes són les especificacions es lliuren als arquitectes, dissenyadors i enginyers de sistemes per a desenvolupar models de nivell inferior que defineixin l'arquitectura del sistema (físic) i el seu funcionament intern (lògic). El disseny es realitza en termes de la plataforma de programari i maquinari on es vol implementar el sistema.

Per tant diem que el disseny arquitectònic és la descripció del sistema pel que fa als seus mòduls (components). Inclou decisions sobre com s’implementarà el client i els aspectes del propi sistema. El disseny arquitectònic també es preocupa per la selecció d'una estratègia organitzativa i la modularització del sistema. L'estratègia organitzativa, necessita resoldre els problemes del client (interfície d'usuari) i del servidor (base de dades), així com qualsevol intermediari necessari per "enganxar" els processos client i servidor junts.

És important que la decisió sobre els components bàsics a utilitzar estigui en sintonia amb el disseny detallat dels components, i s’ha d’ajustar a la solució de client-servidor seleccionada. Els models de client-servidor s’estenen amb freqüència per proporcionar una arquitectura de tres nivells, on la lògica d’aplicació constitueix una capa separada. El nivell mitjà és un nivell lògic i, com a tal, pot estar o no suportat per maquinari separat. La lògica de l’aplicació és un procés que es pot executar sobre el client o el servidor, és a dir, es pot compilar al procés del client o del servidor i implementar-lo com a biblioteca d’enllaços dinàmics (DLL), interfície de programació d’aplicacions (API), trucades de procediment remot (RPC), etc.

La qualitat del disseny arquitectònic és tremendament important per a l'èxit del sistema. Un bon disseny arquitectònic produeix sistemes adaptables, és a dir, sistemes comprensibles, mantenibles i escalables. Sense aquestes qualitats, la complexitat que té el sistema no queda controlada. Per tant, és crucial que el disseny arquitectònic ofereixi una estructura del sistema adaptativa, guiï el procés de desenvolupament i es mantingui amb cura després del lliurament del sistema.

5.4.1 El disseny de l'arquitectura

El disseny arquitectònic té aspectes físics i lògics. El disseny arquitectònic físic està relacionat amb la selecció de les solucions per a cada un dels requisits de desplegament i de distribució de la càrrega de treball del sistema. L'arquitectura física he de resoldre els problemes del client i del servidor, així com qualsevol intermediari necessari per "enganxar" el client i el servidor. Des del punt de vista del model UML, el disseny arquitectònic físic utilitza nodes i diagrames de desplegament.

- Tot i que l'arquitectura física resol els problemes del client i del servidor, el client i el servidor són conceptes lògics (Bochenski, 1994).

El client és un procés informàtic que fa peticions al servidor. El servidor és un procés informàtic que dóna servei al client. Normalment, els processos del client i del servidor s'executen en diferents equips, però és perfectament possible implementar un sistema client-servidor en una sola màquina. En un escenari típic, el procés del client és responsable de controlar la visualització d'informació a la pantalla de l'usuari i de gestionar els esdeveniments de l'usuari. El procés del servidor és qualsevol node d'ordinador amb una base de dades de la qual un procés client pot sol·licitar dades o capacitat de processament.

Figura 23 – Arquitectura del sistema

L'arquitectura client-servidor (C/S) es pot ampliar per representar un sistema distribuït. La necessitat de distribució pot derivar de molts factors, com ara:

- un requisit per al processament especialitzat en una màquina dedicada.
- la necessitat d'accedir al sistema des de diverses ubicacions geogràfiques.

- consideracions econòmiques: l'ús de diverses màquines petites pot ser més barat que un ordinador gran i car.
- un requisit d'adaptabilitat per garantir que les futures extensions del sistema puguin augmentar-se bé.

Per tant quan dissenyem l'arquitectura, hem de generar un informe que inclogui les especificacions de l'arquitectura física, així com el disseny lògic del client-servidor. D'aquesta manera facilitarem molt el desplegament del sistema en qualsevol suport.

5.4.2 Els contenidors

L'arquitectura del sistema pot arribar a ser molt complexa, per tant moltes vegades necessitem solucions per a facilitar el desplegament i el manteniment del programari. Els contenidors són una tecnologia que s'usa cada vegada amb més freqüència.

Els contenidors són tecnologies que permeten empaquetar i aïllar les aplicacions juntament amb tot l'entorn que necessiten durant el seu temps d'execució, és a dir, amb tots els fitxers que requereixen per executar-se. D'aquesta manera facilita l'aixecament del sistema i la seva distribució.

Figura 24 – Sistemes de contenidors

Això permet moure l'aplicació que es troben dins del contenidor entre els entorns (desenvolupament, prova, producció, etc.), sense perdre cap de les seves funcions. Els contenidors també són una part important de la seguretat d'IT. També redueixen els conflictes entre els equips de desenvolupament i els d'operacions, gràcies al fet que separen les àrees de responsabilitat. Els desenvolupadors poden concentrar-se en les seves aplicacions, i l'equip d'operacions ocupar-se de la infraestructura.

És per això que és molt recomanable dissenyar l'arquitectura del nostre sistema pensant a fer-lo encaixar amb aquesta tecnologia.

5.5 Definició tècnica

Un cop hem identificat els requisits del nostre projecte, hem definit les tecnologies que més s'adapten a ells i hem dissenyat l'arquitectura, hem de planificar en profunditat l'estructura de cada una de les parts del sistema. Per a realitzar aquesta tasca hem de definir l'estructura de classes, les taules de bases de dades i els seus atributs i planificar com serà, en definitiva, cada una de les parts del projecte global.
En desenvolupar aquesta tasca, és imprescindible seguir unes normes bàsiques de disseny per tal d'aconseguir que tot el que programem sigui comprensible, mantenible i escalable.
El producte final d'aquest pas ha de ser un document amb tota la informació per a posar-se a programar sense entrebancs. És a dir, les estructures de dades representades en taules de dades, el model (o diagrama) de classes amb totes les entitats de cada una de les parts i els diagrames de flux de les accions.

5.5.1 Les taules de dades

Gestionar les bases de dades es pot convertir en una feina en si mateix quan el nostre programa és escalat a grans dimensions. Per tant un bon disseny es fa imprescindible en el moment de crear un projecte.
Existeixen protocols per tal de generar les bases de dades més eficients, el procés de disseny consta dels següents passos:

- Determinar el propòsit de la base de dades: Conèixer l'objectiu de la base de dades en concret i per tant les seves necessitats ens ajudarà amb els següents passos.
- Buscar i organitzar la informació necessària: Seguidament hem de fer un bolcat de tota la informació que volem guardar en la base de dades per tal de tenir totes les dades sobre la taula. En aquest punt no ens hem de preocupar que estiguin optimitzades.
- Dividir la informació en taules: Dividir els elements d'informació en entitats o temes, com productes o comandes principals. Cada tema es converteix en una taula que després optimitzarem.
- Convertir els elements d'informació en columnes: En aquest punt s'ha de decidir quina informació es vol emmagatzemar en cada taula. Cada element es converteix en un camp i es mostra com una columna de la taula. Per exemple, una taula empleats pot incloure camps com cognom i data de contractació.
- Especificar claus principals: Trieu la clau principal de cada taula. La clau principal és una columna que s'utilitza per identificar inequívocament cada fila. Un exemple podria ser l'identificador de producte o Id.

- Configura les relacions de taula: Ara hem d'examinar cada taula i decidir com estan relacionades les dades d'una taula amb les dades en altres taules. Afegir camps a les taules o crear noves taules per clarificar les relacions segons sigui necessari.
- Aplicar les regles de normalització: Aplicar les regles de normalització de dades per veure si les taules estan estructurades correctament. Realitzar ajustos en les taules, segons sigui necessari.

5.5.2 El diagrama de classes o entitats

Tot hi que hi han diferents formes de programar i cada llenguatge és diferent i requereix certs ajustaments, actualment la programació orientada a objectes (OOP) és la més estesa com a estructura bàsica. D'aquesta manera ens permet crear codi molt més ordenat i fàcilment entendible i manipulable. Per tant en aquest apartat haurem de definir les entitats o objectes que seran l'esquelet del nostre programa i les seves relacions. Per tal de fer un bon disseny hem d'usar les bones pràctiques i els patrons de disseny.

En primer lloc tenim els patrons creacionals. Són els que faciliten la tasca de creació de nous objectes, de manera que el procés de creació pugui ser desacoblat de la implementació de la resta de sistema. Els patrons creacionals estan basats en dos conceptes:

1. Encapsular el coneixement sobre els tipus concrets que el nostre sistema utilitza. Aquests patrons normalment treballen amb interfícies, de manera que la implementació concreta que utilitzem queda aïllada.
2. Amaga com aquestes implementacions concretes són creades i com es combinen entre si.

Els més comuns són:

- **Abstract Factory:** Ens proveeix una interfície que delega la creació d'un conjunt d'objectes relacionats sense necessitat d'especificar en cap moment quines són les implementacions concretes.
- **Factory Method**: Exposa un mètode de creació, delegant en les subclasses la implementació d'aquest mètode.
- **Builder**: Separa la creació d'un objecte complex de la seva estructura, de tal manera que el mateix procés de construcció ens pot servir per crear representacions diferents.
- **Singleton:** Limita a un el nombre d'instàncies possibles d'una classe en el nostre programa, i proporciona un accés global a aquest.
- **Prototype**: Permet la creació d'objectes basats en «plantilles». Un nou objecte es crea a partir de la clonació d'un altre objecte.

També tenim els patrons estructurals. Són patrons que ens faciliten la modelització dels nostres programaris especificant la forma en què unes classes es relacionen amb altres. Aquests són els patrons que va definir la Gang of Four:

- **Adapter**: Permet a dues classes amb diferents interfícies treballar entre elles, a través d'un objecte intermedi amb el qual es comuniquen i interactuen.
- **Bridge**: Separa una abstracció de la seva implementació, perquè les dues puguin evolucionar de forma independent.
- **Composite**: Facilita la creació d'estructures d'objectes en arbre, on tots els elements fan servir una mateixa interfície. Cada un d'ells pot al seu torn contenir un llistat d'aquests objectes, o ser l'últim d'aquesta branca.
- **Decorator**: Permet afegir funcionalitat extra a un objecte (de forma dinàmica o estàtica) sense modificar el comportament de la resta d'objectes del mateix tipus.
- **Facade**: Una facade (o façana) és un objecte que crea una interfície simplificada per tractar amb una altra part del codi més complexa, de manera que simplifica i aïlla el seu ús. Un exemple podria ser crear una façana per tractar amb una classe d'una llibreria externa.
- **Flyweight**: En aquest patró una gran quantitat d'objectes comparteix un mateix objecte amb propietats comunes per tal d'estalviar memòria.
- **Proxy**: És una classe que funciona com a interfície cap a qualsevol altra cosa: una connexió a Internet, un arxiu en disc o qualsevol altre recurs que sigui costós o impossible de duplicar.

Per últim ens trobem els patrons de comportament. En aquest últim grup es troben la majoria dels patrons, i es fan servir per gestionar algoritmes, relacions i responsabilitats entre objectes. Els patrons de comportament són:

- **Command**: Són objectes que encapsulen una acció i els paràmetres que necessiten per executar-se.
- **Chain of responsibility**: s'evita acoblar a l'emissor i receptor d'una petició donant la possibilitat a diversos receptors de consumir-lo. Cada receptor té l'opció de consumir aquesta petició o passar-la al següent dins de la cadena.
- **Iterator**: S'utilitza per a poder moure'ns pels elements d'un conjunt de forma seqüencial sense necessitat d'exposar la seva implementació específica.
- **Mediator**: Objecte que encapsula com un altre conjunt d'objectes interactuen i es comuniquen entre si.
- **Memento**: Aquest patró atorga la capacitat de restaurar un objecte a un estat anterior.
- **Observer**: Aquest patró defineix objectes que són capaços de subscriure's a una sèrie d'esdeveniments que un altre objecte va emetent, i seran avisats quan això passi.

- **State**: Permet modificar la forma en què un objecte es comporta en temps d'execució, basant-se en el seu estat intern.
- **Strategy**: Permet la selecció de l'algoritme que executa certa acció en temps d'execució.
- **Template Method**: Especifica l'esquelet d'un algoritme, permetent a les subclasses definir com implementen el comportament real.
- **Visitor**: Permet separar l'algoritme de l'estructura de dades que s'utilitzarà per a executar-lo. D'aquesta manera es poden afegir noves operacions a aquestes estructures sense necessitat de modificar-les.

5.5.3 Els diagrames de flux

En aquest punt hem de passar les tasques definides en el document de disseny a fils d'execució concretats en les entitats definides en el punt anterior. Els passos concrets solen ser: definir els casos d'ús, identificar les entitats que hi intervenen i definir el flux.
El diagrama de seqüència és un gràfic en dues dimensions. Els rols (objectes o entitats) es mostren al llarg de la dimensió horitzontal. La seqüenciació d'accions es mostra de dalt a baix en la dimensió vertical. Cada línia vertical s'anomena línia de vida de l'objecte i si l'objecte està activat, es mostra com un rectangle d'altura vertical.
Una fletxa representa una crida des d'un objecte (remitent) a una operació (mètode) de l'objecte anomenat (objectiu). Com a mínim, s'ha d'especificar el nom del mètode tot i que normalment també s'inclouen arguments reals del missatge i altra informació de control.
El resultat final que haurem d'usar per a programar la nostra aplicació, seran totes les accions possibles del nostre programa definides en aquests diagrames. És molt important usar també els patrons de disseny que hàgim seleccionat en la definició de les entitats per a seguir un bon disseny. També és important afegir que si treballem amb algun framework, pot ser les accions que vulguem definir es vegin afectades pel funcionament intern d'aquest.

5.6 Desenvolupament

Com hem anat remarcant durant tot aquest treball, els desenvolupadors, solem tenir la mirada fixa en aquest pas, sovint transformant-lo en l'únic i exclusiu. De totes maneres, el fet que no sigui l'únic no significa que no sigui important i arribats a aquest punt, ens espera un procés llarg de traslladar el que hem dissenyat i preparat per tal d'obtenir el resultat buscat.
Si hem realitzat tots els passos anteriors de forma adequada, en aquest punt podrem treballar molt millor, assegurant un resultat de més qualitat.
Que el nostre programa resolgui adequadament les interaccions amb l'usuari i que ho faci de forma eficient, marcarà la diferència en la usabilitat i en l'èxit del producte.
També és vital organitzar-se bé a l'hora de desenvolupar, tant si treballem en equip com si treballem sols, ja que una distribució i seqüència de tasques correcta ens permetrà identificar punts febles, dependències i errors a temps, rectificant abans que sigui un problema massa gran.
Així, en aquest apartat, parlarem de metodologies de treball, per tal de poder organitzar-nos eficientment, els punts claus que hem de saber per a fer un bon codi i les regles essencials de seguretat informàtica.

5.6.1 Metodologies de treball

Els models i mètodes de desenvolupament expliquen com executar la producció de programari i constitueixen un concepte, més que un estàndard, de procés definits per organitzacions (com CMM, ISO 9000 o ITIL) o que defineixen un marc de compliment (com COBIT). Aquests models no són excloents i per tant podem desenvolupar el nostre propi procés de cicle de vida barrejant elements de diversos models i mètodes. Al cap i a la fi, el procés de desenvolupament és únic per a cada desenvolupador o organització i ha de reflectir el caràcter únic de cada individu. El procés també variarà de projecte a projecte segons la mida, el domini de l'aplicació, les eines de programari necessàries, etc.

Com a generalització, els processos de desenvolupament moderns són iteratius i incrementals. Quan desenvolupem un projecte, hem de fer moltes iteracions, així cada iteració ofereix una versió incremental (millorada) del producte obtenint un resultat parcial. Això significa que cada increment millora la funcionalitat, la usabilitat, el rendiment i les altres qualitats del sistema existents sense canviar d'abast del sistema creixent, primer, en amplada.
Existeixen diversos models i mètodes representatius per al desenvolupament iteratiu i incremental. Els models i mètodes que més popularitat tenen són (Maciaszek i Liong 2005):

- El model en espiral

El model en espiral (Boehm 1988) és un model de referència per a tots els processos iteratius i de desenvolupament incremental.

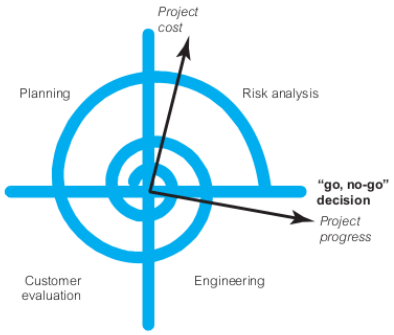


Figura 25 – Model espiral

El model situa les activitats de programació en un context més ampli de planificació del sistema, anàlisi de riscos i avaluació del client. Segons el model espiral, el desenvolupament del sistema comença amb les activitats de planificació. La planificació implica els estudis de viabilitat del projecte i la recollida de requisits inicials. A continuació, el projecte entra al quadrant d'anàlisi de riscos, en el qual s'avaluen els impactes dels riscos en el projecte. L'anàlisi del risc avalua els resultats esperats del projecte i el nivell acceptable de tolerància del risc davant les probabilitats d'assolir aquests resultats. El quadrant d'enginyeria aborda els esforços de desenvolupament. L'avanç del projecte es mesura en aquest quadrant. Abans que el projecte entri a la propera iteració, se sotmet a l'avaluació del client. L'avaluació es realitza segons els requisits coneguts que el sistema hauria de satisfer, però qualsevol altre feedback dels clients també s'envia a l'entrada per a la propera iteració.

- IBM Rational Unified Process (RUP)

L'IBM Rational Unified Process (RUP) es defineix com una plataforma de processos de desenvolupament de programari (RUP, 2003).

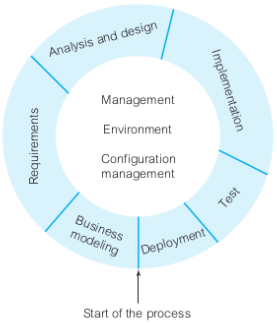


Figura 26 – Model IBM

La plataforma proporciona un entorn de suport al desenvolupament que consisteix en documents de guia i aprenentatge, plantilles de bones pràctiques, tècniques de facilitació basades en web, etc. RUP organitza projectes en termes bidimensionals. La dimensió horitzontal representa les fases successives de cada iteració del projecte. RUP proposa quatre fases: creació, elaboració, construcció i transició. La dimensió vertical representa les disciplines de desenvolupament de programari, és a dir, el model de negoci, els requisits, l'anàlisi i el disseny, la implementació, les proves, el desplegament i les activitats de suport de la configuració i la gestió de canvis, la gestió de projectes i l'entorn.

- Arquitectura model-driven

L'arquitectura basada en models (MDA) (Kleppe et al. 2003; MDA 2006) és una idea antiga. Es remunta al concepte de programació d'especificacions formals i models de transformació. MDA és un marc de modelització executable i es basa en la generació de programes a partir d'especificacions. Els estàndards que permeten definir aquestes especificacions inclouen: Unified Modeling Language (UML) per a les tasques de modelar Meta-Object Facility (MOF) per utilitzar un repositori de meta-models estàndard per tal que les especificacions derivades puguin treballar conjuntament amb XML.

- Desenvolupament Agile

El desenvolupament de programari Agile és una contribució més recent als models iteratius i de desenvolupament incremental.

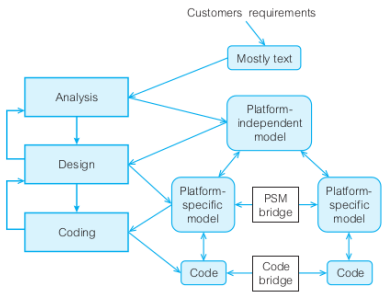


Figura 27 – Model Agile

El desenvolupament Agile inclou el canvi com un aspecte inherent a la producció de programari, proposa mètodes "lleugers" per adaptar-se als requeriments canviants i posa l'etapa com el centre de la programació en el desenvolupament del sistema. Al "Manifest per al desenvolupament de programari Agile", l'Agile Alliance va formular els punts clau de l'agilitat en la producció de programari: individus i interaccions per sobre i per sobre de processos i eines que treballen el programari mitjançant una completa documentació de col·laboració dels clients sobre la negociació de contractes que responen al canvi seguint un pla.

En la metodologia Agile, els programes s'escriuen per passar proves d'acceptació. Aquest procés es coneix com a desenvolupament basat en test i condueix a l'anomenada programació intencionada. Cada iteració en desenvolupament Agile està previst que es completi en un cicle curt d'unes dues setmanes de durada. Els cicles curts impliquen la integració contínua del nou codi amb el codi que ja existeix. El codi integrat al cap de dues setmanes és un lliurament provisional per a l'avaluació del client.

- Aspect-oriented development.

La programació orientada a aspectes (AOP) (Kiczales et al. 1997) no és una idea revolucionaria.

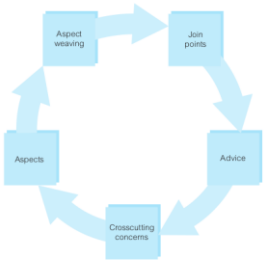


Figura 28 – Model orientat a l'aspecte

L'objectiu principal d'AOP és produir sistemes més modulars identificant les anomenades preocupacions transversals i produint mòduls de programari separats per a aquestes preocupacions. Els mòduls s'anomenen aspectes. Els aspectes s'integren mitjançant el procés anomenat "aspect weaving". Tot i això, cada sistema també ha de complir els requisits no funcionals que determinen qualitats de programari com ara correcció, fiabilitat, seguretat, rendiment, concurrència, etc. Com en el cas del desenvolupament Agile, una mirada més detinguda a aquesta terminologia revela que el desenvolupament orientat a aspectes és només una altra forma d'aplicar processos iteratius i incrementals per produir programari adaptatiu.

5.6.2 El codi

A l'escriure codi, cada programador té el seu estil propi. Aquest fet pot generar dificultats a l'hora de treballar en equip, però a més a més, pot dur a codis que no siguin funcionals. Seguir uns protocols i unes normes a l'hora de programar ens ajudarà a fer codi senzill, i de molta qualitat. Algunes de les normes principals són:

- S'ha de prioritzar la llegibilitat: Mantenir sempre el mateix format i mantenir el codi ben identat amb l'objectiu d'evitar errors i que sigui molt més fàcil llegir el codi.
- Tingues clara l'estructura general del codi per tal de programar amb sentit. Si desconeixem l'abast de cada una de les funcions, podem donar lloc a fils d'execució molt complexes i repeticions de codi.
- Evita repetir codi. Si hi ha dos trossos de codi en una funció que siguin igual, s'ha de fer una nova funció que encapsuli aquest tros.
- Programa funcions curtes que realitzin poques tasques. És important també que siguin el més estàndard possible perquè puguin ser reutilitzables.
- Utilitza noms per a les variables que siguin representatius.
- Comenta només les accions complicades que siguin difícils d'entendre. Intenta evitar omplir tot el programa de comentaris banals.
- Testa tot el codi. D'aquesta manera ens evitarem sorpreses més endavant.
- Tingues control de versions per poder tornar enrere en cas d'un error molt greu
- Simplifica al màxim el codi
- Tingues cura amb caràcters estranys i elements propis del codi amb el qual estiguem programant.

Si en programar seguim aquestes directrius, obtindrem codi molt funcional i que serà altament mantenible per qualsevol altre programador.

6 El procés: una mirada general

Al llarg de tot aquest document hem anat parlant de molts conceptes que, a hores d'ara, poden estar dispersos. L'objectiu d'aquest treball és facilitar la feina als desenvolupadors i agrupar informacions, directrius i tècniques que es veuen implicades en el moment de desenvolupar aplicacions, web o mòbil. Per aconseguir aquest objectiu, hem estructurat la informació en dos blocs diferenciats, el primer versa sobre el disseny creatiu i més enfocat a l'usuari, el segon tracta sobre la part més tècnica.

Tot hi això en un projecte, en funció de les persones que hi treballin, el número i les seves característiques pròpies, els requeriments dels clients o de la pròpia empresa i molts altres factors, el més probable és que aquesta estructura de la qual hem parlat durant tot el treball es vegi modificada de forma substancial.
L'art de desenvolupar és molt personal i per això s'ha de remarcar que cada individu o grup de treballadors se l'ha de fer seu. En molts casos, per exemple, les tasques es poden definir en la part tècnica, fent un exercici més lògic i no tant de la mà dels usuaris. Per tant, en aquest document proposem unes línies generals fonamentades en l'estudi de persones referents com Ben Shneiderman, Alan Cooper, Leszek A. Maciaszek i molts altres. Aquestes línies aporten informació sobre els punts imprescindibles i ofereixen una orientació respecte al procés a seguir, tanmateix, no són inamovibles i han de fluir amb el ritme del projecte en què estiguem embarcats.

Per altra banda, aquest procés no pretén ser estàtic i lineal. Des del començament s'ha anat comentant que les diferents iteracions que puguem realitzar milloraran substancialment el resultat final. És per això que hem de tenir clar que durant el procés passarem repetides vegades per un mateix punt. D'aquesta manera cal ser ordenat i separar clarament cada una de les etapes, tenint clar on comencem i on acabem, extraient un resultat final de cada una per poder alimentar les etapes següents.

Normalment aquest tipus de projectes són desenvolupats per diferents equips en cada etapa o bloc conceptual, com per exemple, l'equip de disseny i els programadors. No obstant això, tenir una imatge mental del procés general i un coneixement bàsic ens aporta molta més comprensió del material que tinguem entre mans i ens permet apreciar d'on venim i a on anem, i així poder realitzar la nostra tasca amb coneixement de causa.

7 Conclusions

A l'inici d'aquest projecte, no era conscient de la magnitud de cada una de les parts que aquest treball vol englobar. M'he trobat literatura i articles per a cada un dels punts dels subpunts, i persones molt més preparades que jo, que han parlat sobre el tema en repetides edicions.
A mesura que he anat avançant en el projecte, totes aquestes informacions, han transformat la idea que hi havia inicialment per a conduir-nos fins al punt actual, aportant molt cada un dels conceptes al resultat final.

De totes maneres, la particular visió de cada un dels autors que he pogut consultar, ha aportat molt valor a les línies d'aquest treball. Els seus coneixements m'han ajudat molt a trobar les paraules i els conceptes a introduir en aquesta recopilació.

Tinc la sensació d'haver retallat massa, vull dir, no haver pogut incloure tot el que m'hauria agradat, però l'objectiu del treball era mostrar els conceptes bàsics i es fa impossible en un treball de 70 pàgines incloure llibres de 300 o 600. De totes maneres, estic molt content amb el resultat, ja que per mi, compleix tots els objectius que ens havíem plantejat des d'un inici.
Com a enginyers, en la meva opinió, és la nostra responsabilitat tenir una visió prou amplia de tot el camp que abasta la nostra professió. Aquest treball fa un recorregut per moltes de les assignatures de la carrera on ens han introduït aquest temari, i intenta agrupar aquesta informació sense deixar cap punt fora de l'equació. Evidentment, quan ens especialitzem, haurem de ser molt bons en el nostre camp per tal de poder treballar eficientment, però no ens podem oblidar de la importància que tenen cada una de les peces d'aquest puzle.

És per això que espero que aquest document pugui servir per presents o futures generacions de desenvolupadors, que torbin en aquestes pàgines un suport i una guia per a millorar els resultats finals i estructurar tot el recorregut, que algunes vegades, si no està ben pensat, pot portar molts maldecaps.

8 Bibliografia

Klaus Pohl.Requirements Enginneering: fundamentals,principles,and techniques.(Springer Berlin Heidelberg, July 2010). p 814. ISBN 3642125778, 9783642125775

Lene Nielsen.Human_Computer Interactions Series.Personas-User Focused Design.(London.Editorial Springer Verlag.2013).p 174.ISBN 2524-4477(Electronic)

Philip Kortum.HCI Beyond the GUI_ Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces (Interactive Technologies). (Editorial Elsevier. Morgan Kaufmann Publismers 2008).p 461.ISBN-13: 978-0-12-374017-5.

Alan Cooper.The Inmates Are Running the Asylum_ Why High Tech Products Drive Us Crazy and How to Restore the Sanity (Sams Publishing.February 24,2004).p 288.ISBN: 0-672-32614-0

Ben Shneiderman, Catherine Plaisant.Designing the User Interface_ Strategies for Effective Human-Computer Interaction-Addison Wesley (Printed in the United States of America.Pearson Education 2005).Executive editor Susan Hartman Sullivan..p 672.ISBN 0-321-19786-0

Debbie Stone, Caroline Jarrett, Mark Woodroffe, Shailey Minocha.User Interface Design and Evaluation (Interactive Technologies).(Elsevier. Morgan Kaufmann series, 2005).p 700.ISBN: 0-12-088436-4

Donald A.Norman.Emotional Design_ Why We Love (Or Hate) Everyday Things.(New York .Basic Books.2004).p 268.ISBN 0-465-05135-9

William J.Brown (Contributor), Raphael C.Malveau, Hays W, Mc Cormick. Antipatterns: Reafactoring Software, Architectures, and projects in Crisis.(Editor Wiley,published april 3rd 1998).p. 336. ISBN: 978-0-471-19713-3.

Donald A.Norman.The Design of Everyday Things (New York.Doubleday/Currency edition,1990).p 261.ISBN 0-385-26774-6

Helen Sharp, Jenny Preece, Yvonne Rogers.Interaction Design_ Beyond Human-Computer Interaction (Indianapolis, Indiana.John Wiley & Sons, 2019).5th edition.p 675.ISBN: 978-1-119-54725-9

Jeff Sauro, James R Lewis.Quantifying the User Experience. Practical Statistics for User Research (Cambridge,United States.Elsevier.Morgan Kaufmann, 2016).Editorial Project Manager: Lindsay Lawrence.2nd edition.p 354.ISBN: 978-0-12-802308-2

Jeffrey Rubin, Dana Chisnell, Jared Spool.Handbook of Usability Testing_ Howto Plan, Design, and Conduct Effective Tests.(Indianapolis, Indiana.Wiley Publishing,2008).2nd edition.p 386.ISBN: 978-0-470-18548-3

Kevin Mullet, Darrell Sano. Designing Visual Interfaces_ Communication Oriented Techniques-Prentice Hall (Universitat de Michigan. SunSoft Press, 1994). Digitalitzat 25 Ags 2010.P 273.ISBN 0133033899, 9780133033892

Klaus Pohl, Günter Böckle, Frank J. van der Linden. Software Product Line Engineering_ Foundations, Principles and Techniques. (Berlin, Heidelberg, New York.Springer, 2005).p 473. ISBN-10 3-540-24372-0

Leszek A. Maciaszek.Requirements Analysis and System Design.(Hampshire-Great Britain.Addison-Wesley, 2007).3rd Edition.p 651.ISBN 978-0-321-44036-5

Mary Beth Rosson, John M. Carroll.Usability Engineering. Scenario-Based Development of Human-Computer Interaction (Elsevier.Morgan Kaufmann, January 2001).p 468.ISBN - 0080520308 - 9780080520308.

Airbrake.io/Application monitoring for your entires stack.Recuperat de: https://airbrake.io/blog/sdlc/conceptual-model

Mads Soegaard and Rikke Friis Dam.Interaction Design Foundation.The Glossary of human Computer Interaction.Cap26.Interaction Styles.Recuperat de:https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-styles

Medium.Requeridos Blog.Requerimientos funcionales y no funcionales, ejemplos y tips.(Apr 20,2018).Recuperat de: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Cercabib.CRAI de la Universitat de Barcelona.Recuperat de: https://cercabib.ub.edu/iii/encore/search/C__SGIGA%20364303?lang=cat

Organización Internacional de Normalización. Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems. ISO 9241-210:2019: https://www.iso.org/standard/77520.html

9 Índex de Figures