Requisits

Jordi Pradel Miquel
Jose Raya Martos

PID_00171146

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

Com hem vist al mòdul "Introducció a l'enginyeria del programari", els requisits expressen quines necessitats ha de cobrir el sistema que desenvoluparem i quines restriccions ha de satisfer. La identificació i gestió de requisits és, per tant, una de les activitats més importants en tot projecte de desenvolupament. Al cap i a la fi, si els requisits no s'han identificat correctament, el programari no farà el que ha de fer o no ho farà com ho ha de fer.

En aquest mòdul veurem com els requisits són l'eina bàsica de comunicació entre el grup de persones que vol que es desenvolupi el sistema i el grup de persones que l'ha de desenvolupar i com, per tant, la identificació dels requisits és una tasca compartida entre tots dos grups. Veurem també algunes de les tècniques habituals per a identificar aquests requisits, que ens ajudaran a superar les dificultats ja mencionades al mòdul "Introducció a l'enginyeria del programari", i també algunes tècniques per a decidir quins són els requisits més prioritaris.

A continuació estudiarem la documentació de requisits i quines qualitats ha de tenir. Com sabem que és suficientment bona? Hem de tenir en compte que el sistema d'informació que desenvolupem serà tan bo com ho siguin els requisits a partir dels quals l'hem desenvolupat i, per tant, la qualitat dels requisits afectarà en gran mesura la qualitat final de la nostra solució.

Finalment proposarem dues maneres diferents de documentar els requisits, cadascuna associada a un tipus diferent de mètode de desenvolupament i, per tant, adequada en uns contextos determinats: les històries d'usuari, més adequades en processos àgils i menys formals, i els casos d'ús, associats a l'estàndard UML, en els quals ens centrarem.

Objectius

Els objectius que l'estudiant ha d'haver assolit una vegada treballats els continguts d'aquest mòdul són:

1. Saber identificar els requisits candidats d'un sistema d'informació.

2. Saber fer la selecció dels requisits del producte de programari per desenvolupar.

3. Saber identificar similituds i diferències entre tres tècniques de documentació de requisits: l'estàndard IEEE-830, les històries d'usuari i els casos d'ús.

4. Saber documentar la funcionalitat d'un producte de programari fent servir la tècnica dels casos d'ús.

1. Introducció als requisits

1.1. Què són els requisits?

Com hem mencionat al mòdul 1, els requisits "expressen les necessitats i restriccions que afecten un producte de programari que contribueix a la solució d'un problema del món real" i ens serveixen per a delimitar quines de les possibles solucions al problema són adequades (les que compleixen els requisits) i quines no.

***Requisit* o *requeriment*?**

Cal no confondre's. Estem parlant de requisits, no de requeriments.

Segons el GREC, un requisit és una condició exigida o necessària per a una cosa, mentre que un requeriment és l'acció de requerir (demanar, sol·licitar).

Per a poder determinar si una solució compleix o no els requisits és necessari que el requisit faci referència a alguna característica observable del sistema ja que, si no podem fer l'observació d'una característica, no podrem determinar el compliment del requisit.

Exemple de característica observable

La característica "el sistema ha de ser còmode de fer servir" no és una característica observable en el sentit que el seu compliment és subjectiu. Caldria, doncs, trobar la manera de dir el mateix fent referència als fets observables del nostre sistema com, per exemple, "es farà un estudi de satisfacció sobre una mostra de 50 usuaris i caldrà obtenir una nota mínima de 4 sobre 5".

Per tal que una característica observable sigui un requisit, però, és necessari que expressi alguna necessitat o restricció que afecti el programari.

Exemple de característica que no interessa a ningú

Per exemple, "el sistema sempre trigarà més de 30 minuts a carregar la pàgina principal del campus" és un fet observable, però no hi haurà ningú que estigui interessat que el sistema que volem desenvolupar el compleixi; en tot cas, voldrem el contrari, que el sistema trigui menys de *N* segons a carregar la pàgina principal del campus.

Un requisit és una característica observable del nostre sistema que satisfà una necessitat o expressa una restricció que afecta el programari que estem desenvolupant.

1.2. Els *stakeholders*

Si els requisits expressen necessitats i restriccions, qui té aquestes necessitats? Qui decideix les restriccions que ha de complir el programari?

Els *stakeholders* d'un projecte són aquelles persones i entitats que tenen algun impacte o interès en aquest.

És important distingir d'entrada entre els *stakeholders* i els usuaris. Tot i que els usuaris sempre són *stakeholders*, no tots els *stakeholders* són usuaris. Aquesta distinció és important ja que, a l'hora de determinar quins són els requisits d'un sistema, cal tenir en compte tots els *stakeholders* i no solament els usuaris.

Els *stakeholders* d'una aplicació web

Prenem com a exemple una aplicació web. Des del punt de vista dels usuaris, és indiferent en quin llenguatge s'ha programat (Java, C#, PHP, etc.). Per tant, podríem dir que el llenguatge de programació no és un requisit del sistema més enllà del fet que sigui web.

En canvi, des del punt de vista dels administradors de sistemes, aquesta característica és molt important fins al punt que, en moltes organitzacions, els administradors de sistemes limiten les possibles plataformes d'explotació a un conjunt estandarditzat (aquelles que saben administrar) i, per tant, limiten els llenguatges de programació a aquells que es poden fer servir amb aquelles plataformes. En aquest cas, doncs, la característica anterior esdevé un requisit del sistema, tot i que als usuaris els sigui indiferent.

Els requisits ens diuen què és el que els diferents *stakeholders* esperen del nou sistema.

Per tant, la funció principal dels requisits és la de **comunicar** les necessitats i objectius dels *stakeholders* als desenvolupadors: els requisits ens diuen quines condicions o capacitats ha de satisfer el nostre sistema per tal de ser valuós i útil als *stakeholders*.

1.3. Tipus de requisits

Podem classificar els requisits en dos grans grups: els que fan referència a les **necessitats** que ha de satisfer el sistema (**què** ha de fer) i els que expressen **restriccions** sobre el conjunt de solucions possibles (**com** ho ha de fer).

Observació

Compte! Quan diem que els requisits fan referència al *com* no vol dir que els requisits indiquin com ha de ser el sistema internament, sinó que expressen restriccions sobre quines de les possibles maneres són acceptables i quines no.

Per exemple, el requisit "Un alumne ha de poder publicar un missatge al fòrum del grup al qual està matriculat" pertanyeria al primer grup, mentre que el requisit "El sistema estarà programat en un dels llenguatges corporatius" seria del segon tipus. Dels requisits del primer grup en diem **requisits funcionals**, mentre que els del segon grup els anomenem **requisits no funcionals**.

Requisits del producte i del procés

De vegades, també es distingeix entre els requisits del producte (aquells que afecten el sistema per desenvolupar) i els requisits del procés (aquells que afecten el procés de desenvolupament). Segons aquesta classificació, el requisit "Un alumne ha de poder publicar un missatge al fòrum del grup al qual està matriculat" seria un requisit del producte, mentre que "El sistema estarà programat en un dels llenguatges corporatius" seria un requisit del procés.

Un exemple de requisit del producte no funcional podria ser: "El sistema haurà d'estar disponible el 99,9% del temps".

1.3.1. Requisits funcionals

Els requisits funcionals fan referència a la funcionalitat que ha de proporcionar el sistema.

Els requisits funcionals ens indiquen quins càlculs fa el sistema, quines dades manté, com les manipula, etc. En general, podem dir que els requisits funcionals ens indiquen quin és el comportament del sistema davant dels estímuls que li arriben de l'exterior.

1.3.2. Requisits no funcionals

Els requisits no funcionals fan referència a restriccions sobre el conjunt possible de solucions.

Els requisits no funcionals acostumen a tenir forma de restricció i solen afectar gran part del sistema. Per exemple, si diem que el campus virtual que estem desenvolupant ha de ser portable a altres plataformes, això no afecta solament el fòrum o el lliurament d'exercicis sinó que afecta tot el sistema.

Una altra diferència respecte als requisits funcionals és que els requisits no funcionals no inclouen comportament. La seva finalitat és especificar criteris que avaluïn la qualitat general del sistema com seguretat, rendiment, cost, etc., més que no pas especificar quin és el comportament del sistema (això és, com hem dit anteriorment, responsabilitat dels requisits funcionals). Tornant a l'exemple anterior, si volem que el campus virtual sigui portable, ho ha de ser amb independència de la funcionalitat que ofereixi.

1.4. Els requisits al llarg del desenvolupament

Hem vist fins ara que els requisits són clau per a l'èxit o el fracàs d'un projecte de desenvolupament de programari i que, de fet, seran els requisits els que guiaran la resta d'activitats de desenvolupament. Per això, amb independència de quin sigui el mètode de desenvolupament per seguir, hi ha una sèrie d'activitats relacionades amb els requisits que sempre haurem de dur a terme:

- **Obtenció dels requisits**: identificar quins són els requisits que els diferents *stakeholders* volen que compleixi el sistema

- **Gestió dels requisits**: com a part de la gestió del projecte, cal decidir quins requisits són més prioritaris, identificar-ne de contradictoris, decidir quins volem implementar primer, etc.

- **Documentació dels requisits**: crear un document o un model amb els requisits del sistema (amb independència de si aquest artefacte forma part o no de la documentació final del sistema).

- **Verificació del sistema**: comprovar si el sistema desenvolupat compleix o no els requisits.

Nota

En aquest mòdul, no estudiarem les diferents tècniques relacionades amb la verificació del sistema, que es tracten en altres assignatures.

Les tasques i artefactes concrets que farem servir en un projecte de desenvolupament dependran, és clar, del mètode emprat. Així, per exemple, els mètodes amb cicle de vida en cascada faran una documentació de requisits més exhaustiva que els mètodes àgils.

En aquest mòdul veurem algunes de les tècniques que s'utilitzen en els diferents tipus de mètodes per a dur a terme aquestes activitats. Aquest recull de tècniques no és exhaustiu (hi ha tècniques que no veurem) i tampoc no està lligat a cap mètode de desenvolupament concret (per tant, és poc probable que, en un projecte concret, les hàgiu d'aplicar totes).

2. Obtenció dels requisits

La primera activitat relacionada amb els requisits és l'obtenció d'aquests. L'objectiu d'aquesta activitat és obtenir la llista de tots els requisits que, idealment, hauria de complir el sistema per desenvolupar.

Anomenarem **requisits candidats** els requisits obtinguts en aquesta primera etapa, ja que encara no hem decidit si els incorporarem o no al conjunt de requisits del nostre sistema. Per exemple, hem de tenir present que els requisits de partida poden ser contradictoris.

Exemple

En Joan és un estudiant de la nostra universitat i necessita, en el moment de lliurar una activitat, poder-la lliurar en el format de document que li sigui més convenient. La Maria és la professora de l'assignatura i necessita poder limitar els formats de document en què es fan els lliuraments per tal de no haver d'instal·lar infinitat d'aplicacions al seu ordinador.

En aquest cas, tant "L'estudiant ha de poder lliurar l'activitat en qualsevol format" com "L'estudiant ha de lliurar l'activitat en format PDF" són requisits del sistema, ja que tenim un *stakeholder* que ens ha expressat aquesta necessitat. Òbviament, el sistema que desenvolupem no podrà complir tots dos requisits.

Com que els requisits sempre han d'estar determinats per un o més *stakeholders*, el procés d'obtenció de requisits s'acostuma a fer en dues etapes:

- Identificar els *stakeholders* del sistema.
- Identificar els requisits de cadascun dels *stakeholders*.

La correcta identificació dels *stakeholders* del sistema és crítica, ja que, si ens deixem un *stakeholder*, no podem preveure quin serà el seu impacte sobre el projecte de desenvolupament i, en el pitjor dels casos, ens podria portar a haver de cancel·lar el projecte.

Exemple

Per exemple, podria passar que un projecte depengués d'una subvenció pública per al seu finançament però que l'Estat afegís condicions, com per exemple uns mínims d'accessibilitat, per tal d'atorgar la subvenció. Si no es té en compte l'Estat com a *stakeholder* i no es compleixen els seus objectius, el projecte es quedarà sense finançament i s'hauria de cancel·lar.

2.1. Identificació de *stakeholders*

2.1.1. *Brainstorming* o pluja d'idees

Com que sovint no tenim un punt concret de partida, és molt habitual fer servir la tècnica del *brainstorming* per a identificar els *stakeholders* del projecte. Aquesta tècnica també es podrà fer servir per a identificar els requisits d'un *stakeholder* concret (sobretot si no hi tenim accés directe).

Una sessió de *brainstorming* acostuma a ser més o menys així:

1) Creació del grup de treball. Es reuneix un grup que, idealment, hauria d'incloure persones amb rols diferents i variats per tal que tinguin punts de vista diferents i es puguin complementar. La composició del grup dependrà de la finalitat del *brainstorming* però podria incloure, per exemple, desenvolupadors, usuaris, caps de projecte, etc.

2) *Brainstorming* inicial. Cada participant de la reunió proposa tantes idees com sigui capaç d'imaginar. No cal respectar torns i és perfectament lícit fer servir les idees dels altres participants per a proposar-ne de noves (per exemple, si estem identificant usuaris potencials i algú pensa en "estudiants", algú altre pot proposar "estudiant que aquest semestre no està matriculat en cap assignatura"). La finalitat d'aquesta etapa és aconseguir el màxim d'idees possible i, per tant, no es discuteix ni es qüestiona cap proposta.

3) Organització i consolidació del conjunt inicial. S'organitzen les idees i s'identifica el grau d'encavalcament entre aquestes. Aquelles que siguin molt similars es poden consolidar en una de sola. Per exemple, si estem identificant *stakeholders* per al nostre campus virtual, podem veure que "qui contracta el desenvolupament" i "qui pagarà el desenvolupament" són dues propostes molt properes i les podem consolidar en una de sola, mentre que "estudiant" és una proposta totalment diferent de les altres dues i, per tant, és un tipus de *stakeholder* diferent.

4) Refinament. Es descriu amb una mica més detall cadascuna de les propostes per tal d'aconseguir la llista definitiva.

2.1.2. Modelització de rols d'usuari

La modelització de rols d'usuari és una tècnica orientada, concretament, a la identificació dels usuaris del sistema que, habitualment, s'aplica en forma de *brainstorming*.

La idea bàsica d'aquesta tècnica és que, en comptes de buscar els requisits de tots els usuaris individuals del sistema, els podem agrupar segons el rol i assumir que tots els usuaris individuals que tenen el mateix rol davant del sistema tindran requisits similars.

El resultat d'aquesta tècnica és una breu descripció de les característiques principals de cada tipus d'usuari, com ara amb quina freqüència farà servir el sistema, el seu nivell de coneixement del domini del problema, el seu nivell de coneixements informàtics o per a què faran servir el sistema.

Exemple d'un rol d'usuari identificat mitjançant modelització de rols d'usuari

- Rol d'usuari: estudiant.
- No és necessàriament expert en informàtica, tot i que té experiència en l'ús d'Internet i està predisposat a fer servir el campus virtual. Vol poder fer consultes als professors, comunicar-se amb els altres estudiants que estiguin fent les mateixes assignatures i fer el lliurament de les activitats, tot això de manera no presencial.

Aquesta tècnica es pot combinar amb altres com la de Persones, que consisteix a crear una persona imaginària (amb la seva biografia) per a representar cadascun dels rols d'usuari. Així doncs, en comptes de parlar dels estudiants com a cosa abstracta, passaríem a parlar de la Maria. El fet de posar nom (i fins i tot cara) al rol d'usuari ens ajudarà a posar-nos en la seva pell i identificar els seus requisits.

La Maria

La Maria va néixer a Girona fa 23 anys i ha treballat com a auxiliar administrativa des dels 21 anys. Ara es vol posar al dia amb els estudis i s'ha matriculat a Empresarials. Està una mica nerviosa perquè és el seu primer any a la universitat però confia que se'n sortirà. No sap molt bé com funciona el campus virtual però està habituada a fer servir el correu electrònic i les xarxes socials, per la qual cosa creu que no tindrà problemes a entendre's amb el sistema.

2.1.3. Representants dels *stakeholders*

De vegades pot passar que no tinguem accés directe als *stakeholders*, especialment als usuaris. En aquests casos, necessitarem parlar amb algú que faci de representant.

> El representant d'un *stakeholder* és aquella persona que parla en nom seu i que ens ha de transmetre els seus requisits.

Òbviament, un representant d'un *stakeholder* mai no és tan fiable com l'*stakeholder* mateix a qui representa, ja que pot ser que el seu coneixement de la persona a qui representa i les necessitats d'aquesta sigui parcial.

Un dels punts particularment complicats és que el representant sol ser, ell mateix, un *stakeholder*, amb els seus interessos i necessitats propis. Per això, el representant pot tenir tendència a prioritzar els seus interessos propis per davant dels de la persona representada.

Però, a més, segons qui sigui el representant, ens podem trobar amb problemàtiques específiques d'aquell tipus de representant que caldrà tenir en compte.

Un representant habitual dels usuaris és el responsable dins l'organització. En aquest cas cal tenir en compte que és possible que el responsable no sigui usuari del sistema o que l'ús que faci del sistema sigui diferent del que fan els usuaris que ens interessen. Per exemple, és probable que els coordinadors del professorat estiguin molt més interessats a obtenir informes que els facilitin el seguiment de l'activitat dels professors que no pas els professors mateixos.

Un altre cas habitual és aquell en què els desenvolupadors decideixen els requisits sense parlar amb els *stakeholders*. En general, els desenvolupadors són un mal representant dels *stakeholders*, ja que, no solament no coneixen l'ús diari del sistema, sinó que, possiblement, estaran més interessats en els aspectes tecnològics de la solució que en les necessitats reals de les persones a qui representen.

Suport al desenvolupament

Hi ha un cas, però, en què els desenvolupadors són un bon representant dels *stakeholders*: el desenvolupament d'eines de suport al desenvolupament. En aquest cas, els desenvolupadors han de saber separar la seva visió com a desenvolupadors de l'eina de la seva visió com a usuaris.

Els comercials poden actuar com a representants, especialment en casos en què volem desenvolupar un producte per comercialitzar-lo i sobretot si encara no tenim usuaris reals del sistema, ja que són qui té el contacte més directe i més proper amb els clients i els usuaris. En aquest cas, però, cal tenir en compte que és probable que donin prioritat als requisits que els puguin ajudar a vendre el producte, més que no als que realment siguin útils per als usuaris. Cal recordar també que podria ser que els comercials tinguessin accés als clients (aquells qui prendran la decisió de compra) però no als usuaris.

Altres representants interessants poden ser els formadors, el personal de suport tècnic (al cap i a la fi, la seva feina és parlar amb els usuaris i explicar-los com funciona el sistema, de manera que saben quins problemes tenen), els analistes de negoci o els experts en el domini.

2.2. Identificació de requisits

2.2.1. Entrevistes i qüestionaris

Aquesta és una de les tècniques més utilitzades per a l'obtenció de requisits. Consisteix, òbviament, a entrevistar-se amb els *stakeholders* per a obtenir directament d'ells els requisits que tenen sobre el sistema per desenvolupar.

Les claus perquè l'entrevista sigui productiva són:

- **Escollir correctament els entrevistats**. Ens hem d'assegurar la varietat i representativitat de la mostra; en el nostre cas, cal haver identificat correctament els *stakeholders* i assegurar-nos que aquells que entrevistem en són una mostra significativa i representativa.

- **Evitar les respostes condicionades**. Qualsevol de nosaltres, davant de la pregunta "Voldries que el sistema donés resposta en menys d'un segon?" respondria que sí, ja que entenem que si diem que no el sistema serà molt lent. Aquesta resposta, però, no voldria dir que realment el nostre límit estigui en un segon (potser estem disposats a esperar-ne dos o tres abans de desesperar-nos), però la manera en què s'ha formulat la pregunta ha condicionat la nostra resposta.

- **Evitar respostes limitades pel coneixement actual**. Tot i que és molt important tenir en compte el coneixement actual, de vegades pot ser contraproductiu, ja que ens pot limitar la capacitat d'innovar i trobar solucions millors que les actuals.

  Exemple

  En el camp de les aplicacions web, per exemple, és habitual que els usuaris no estiguin al corrent de les possibilitats tecnològiques de la plataforma i que, per tant, no se'ls acudeixi la possibilitat de fer servir determinats controls d'usuari com ara autocompletar una caixa de text o l'ús del *drag'n'drop*.

- **Aportar informació sobre el cost de les alternatives**. Si algú ens demana si voldríem que un sistema fes *X* i no ens diu el cost, el més probable és que la resposta sigui un *sí* amb majúscules (amb independència del valor de *X*!). En canvi, si ens diuen que el cost de desenvolupar *X* fa que el projecte es retardi un any, probablement la resposta sigui diferent.

Com a alternativa (o complement) a les entrevistes, també podem fer servir **qüestionaris**. Tot i que, a causa de la seva naturalesa tancada, no són massa adients per al descobriment de nous requisits, poden ser molt útils per a refinar els requisits que ja hem descobert.

L'ús de qüestionaris pot ser especialment útil amb poblacions relativament grans d'usuaris. Això ens permetrà, per exemple, conèixer el grau d'importància relativa dels diferents requisits.

2.2.2. Observació i prototipatge

Aquesta tècnica consisteix en l'observació directa dels usuaris mentre fan servir un sistema d'informació, sobretot per a detectar requisits relacionats amb la usabilitat.

Exemples d'observació d'usuaris

Per exemple, si veiem que un usuari ha de consultar una dada, apuntar-la i llavors entrar-la en una altra pantalla de l'aplicació, podem pensar que li vindria bé que el sistema recordés la dada per ell.

Un altre exemple: un usuari vol fer una tasca concreta de manera freqüent i necessita passar per 6 opcions de menú abans d'arribar-hi: podríem promocionar aquesta tasca de manera que l'usuari en tingués accés directe.

Encara un altre exemple: estem desenvolupant una aplicació per a un escorxador; si observem els usuaris al seu entorn veurem que, per qüestions de seguretat, han de portar uns guants protectors que fan molt difícil l'ús del teclat, la qual cosa ens hauria de portar a pensar en maneres alternatives d'introduïr la informació.

Aquesta observació es pot fer sobre un sistema ja existent, sobre una implementació parcial del sistema final (en el cas del desenvolupament incremental) o sobre un prototip.

El prototipatge, per tant, és una tècnica que pot facilitar l'observació dels usuaris. A més, però, també pot ser útil en altres escenaris, com ara per a facilitar la comunicació amb els usuaris (ja que entendran millor un prototip que la descripció d'un requisit) o per a explorar alternatives.

El que distingeix un prototip d'una implementació parcial és que el prototip no forma part del producte definitiu.

És molt important recordar que el prototip no s'ha de mantenir, ja que, un cop obtinguts els requisits, aquest codi s'abandona i no forma part del producte final. Ni tan sols cal que fem servir la mateixa tecnologia amb la qual implementarem el producte final, sempre que simuli correctament la funcionalitat que volem explorar.

Com que sabem que no haurem de mantenir el codi del prototip, el podrem desenvolupar més ràpidament i amb menys cost (entre altres coses perquè no cal que la funcionalitat sigui completa) que si estiguéssim treballant amb el producte final.

2.2.3. Llistes predefinides

Una altra tècnica és fer servir llistes predefinides[1] de requisits que ens poden ajudar a no passar per alt alguns requisits. Aquesta tècnica és especialment útil per als requisits no funcionals ja que, com hem vist anteriorment, aquests són independents del comportament del sistema.

(1)En anglès, *checklists*.

Exemple

L'estàndard IEEE 830 ens recorda que sempre hem de documentar, a més dels requisits funcionals, els requisits no funcionals del següent:

- Rendiment (volum d'usuaris, volum de dades, etc.)
- Requisits lògics de la base de dades (tipus d'accés, freqüència dels accessos, entitats, relacions i restriccions d'integritat, etc.)
- Restriccions de disseny (altres estàndards, limitacions de maquinari, etc.)
- Altres atributs
  - fiabilitat
  - disponibilitat
  - seguretat (ús de criptografia, informes sobre les accions dels usuaris, etc.)
  - mantenibilitat
  - portabilitat

A continuació veurem amb més detall els diferents tipus de requisits no funcionals que menciona la plantilla Volere, que inclou una llista predefinida de requisits no funcionals.

Web recomanada

Volere Requirements Specification Template. http://www.volere.co.uk/template.htm (Darrera visita: setembre 2010)

Requisits de presentació

Els requisits de presentació fan referència a l'aspecte visual del sistema per desenvolupar i han de tenir en comptes aspectes com, per exemple, la imatge corporativa de l'organització per a la qual estem desenvolupant el programari.

Exemple

Com a exemple de requisit de presentació, podríem considerar "El producte ha de ser atractiu per a una audiència adolescent. Es durà a terme un estudi en què es posarà el producte a l'abast d'una mostra d'adolescents i s'observarà si comencen o no a fer servir el producte abans de quatre minuts".

Requisits d'usabilitat i humanitat

Els requisits d'usabilitat i humanitat són els requisits relacionats amb el fet que el sistema sigui ergonòmic i usable. La usabilitat del producte dependrà de les capacitats dels seus usuaris i de la complexitat del producte. Alguns dels aspectes que cal tenir en compte en aquest apartat són:

- **Eficiència d'ús**: rapidesa i precisió amb què l'usuari fa servir el producte.

Exemple

Un usuari sense entrenament ha de ser capaç de trobar el pla d'estudis d'una assignatura de la qual s'acaba de matricular en menys de 5 minuts.

- **Facilitat de memorització**: quin volum d'informació ha de memoritzar un usuari no habitual del sistema.

Exemple

Els menús no poden tenir més de 9 opcions i no poden descendir més de 3 nivells.

- **Taxa d'errors**: quantes vegades es pot equivocar l'usuari intentant dur a terme alguna tasca.

Exemple

Un usuari que hagi fet servir el sistema durant un mes només s'equivocarà l'1% de les vegades que intenti fer alguna cosa al sistema.

- **Satisfacció**: quin és el nivell de satisfacció general amb el producte un cop s'ha fet servir.

Exemple

El 75% dels usuaris que provin el nou sistema l'han de continuar fent servir (i no haver tornat al sistema vell) al cap d'un mes d'haver-lo fet servir per primer cop.

- **Retroalimentació**: quanta informació necessita l'usuari per tal de confiar que el producte està fent el que se suposa que ha de fer.

Exemple

Quan el sistema iniciï una tasca que duri més d'un segon, es mostrarà un indicador de progrés.

En aquest àmbit també podem trobar requisits relatius a la personalització del producte o la internacionalització i localització d'aquest.

Internacionalització i localització

Segons la Wikipedia, la **internacionalització** és el procés de dissenyar programari de tal manera que es pugui adaptar a diferents idiomes i regions sense la necessitat de fer canvis en el codi del programa. La **localització** és el procés d'adaptar el programari per a una regió específica.

També hem de considerar requisits relacionats amb la corba d'aprenentatge del producte o, dit d'una altra manera, quina inversió en temps han de fer els usuaris abans de poder fer servir el sistema amb confiança i productivitat.

Una altra categoria de requisits que ens planteja la plantilla en aquesta àrea són els que anomena *de comprensibilitat*, que fan referència a com és d'intuïtiu el sistema i com encaixa en la seva visió del món (per exemple, fer servir un conjunt de símbols conegut, o seguir una guia d'estil concreta per a la interfície gràfica d'usuari).

[2]En anglès, *web accessibility initiative*.

Finalment, no podem deixar de banda els requisits d'accessibilitat. En aquest sentit, hi ha estàndards com la iniciativa d'accessibilitat web[2] que ens poden guiar sobre quins aspectes del nostre sistema hem de tenir en compte a l'hora de garantir l'accés a persones discapacitades.

Requisits de compliment

En aquesta categoria trobem els requisits relacionats amb la manera de complir les responsabilitats del sistema, com per exemple, la velocitat i latència (especialment crítics en els sistemes de temps real), que fan referència a la velocitat amb què el sistema reacciona a un esdeveniment.

Exemple de requisits d'acompliment

Qualsevol acció de l'usuari ha de provocar una resposta visible abans de tres segons.

També hem de considerar els requisits de seguretat en el sentit del dany que podem provocar a altres persones, ja sigui perquè el programari estigui controlant algun element físic que pugui causar un dany o perquè doni instruccions equivocades a la persona. Sovint aquests requisits estaran recollits en alguna mena d'estàndard nacional.

Els requisits de precisió fan referència a la precisió de la informació proporcionada pel sistema com, per exemple, el nombre de decimals que es faran servir per a les conversions d'imports monetaris o quin sistema d'unitats farà servir.

La fiabilitat es pot mesurar com el temps admissible entre dues caigudes del sistema, mentre que la disponibilitat és el percentatge de temps que el sistema està disponible.

Exemple

El sistema no pot caure més de dues vegades el mateix dia.

El sistema ha d'estar disponible el 99% del temps.

La robustesa ens parla de la capacitat del sistema per a continuar funcionant en cas de trobar-se en circumstàncies inesperades.

Llei de Murphy

La famosa llei de Murphy deu el seu nom a l'enginyer aeroespacial Edward A. Murphy i, originalment, feia referència al fet que cal dissenyar els sistemes tenint en compte que si alguna cosa pot anar malament, s'ha de dissenyar el sistema tenint en compte que anirà malament.

Una altra categoria per tenir en compte és la dels requisits relacionats amb la capacitat (volums d'usuaris, de dades, etc.).

Exemple

L'aplicació del campus virtual ha de suportar la connexió simultània d'almenys el 30% dels estudiants el primer dia de curs a les 9 del matí.

Els requisits d'escalabilitat o extensibilitat fan referència a com s'espera que creixin aquests volums al llarg del temps.

Exemple

L'aplicació del campus virtual ha de suportar un volum de 10.000 estudiants el primer any de funcionament, que seran 15.000 al cap de dos anys.

Finalment, considerarem en aquesta categoria els requisits de longevitat: quant de temps s'espera que el sistema estigui en funcionament.

Exemple

La vida útil mínima d'aquest sistema s'espera que sigui de 10 anys.

L'efecte 2000

Al final del segle XX es va produir l'anomenat *efecte 2000*, que consistia, bàsicament, en un error a l'hora de calcular les dates que s'havien emmagatzemant amb dos dígits, suposant que els dos primers eren 19. Aquest efecte va ser degut, en part, a errors en la previsió de la vida útil dels sistemes.

Requisits operacionals i d'entorn

En aquest grup trobem els requisits relatius a la manera o l'entorn en què es farà servir el producte. Això pot incloure requisits relatius a com els usuaris han d'interactuar amb el sistema, a com ha d'interactuar el sistema amb altres sistemes externs (quines interfícies s'han de fer servir), a com s'ha de distribuir o a com s'han de programar els lliuraments.

Exemple

El sistema ha de poder ser usat per una persona mentre condueix un vehicle (i, per tant, l'ha de poder controlar sense fer servir les mans).

El sistema l'ha de poder fer servir una persona que porti uns guants de seguretat de malla d'acer.

El sistema ha de permetre l'intercanvi de moviments bancaris segons l'estàndard Norma 43 del Banc d'Espanya.

El sistema l'ha de poder instal·lar fàcilment una persona sense coneixements informàtics.

L'aplicació es distribuirà en format de codi font i els usuaris l'hauran de compilar per a la seva plataforma.

Es lliurarà una versió nova cada tres mesos.

Requisits de manteniment i suport

Un cop el sistema estigui en explotació, caldrà, per una banda, mantenir-lo (corregir errors i afegir funcionalitats noves) i donar suport als usuaris. S'han de tenir, doncs, en compte, les necessitats de les persones que s'han d'encarregar d'aquestes tasques.

Exemple

En cas d'error, el sistema assignarà un codi a l'usuari, que el personal de suport podrà consultar per a obtenir un registre detallat dels motius de l'error.

El sistema oferirà la possibilitat d'enviar, de manera automàtica, un missatge al personal de suport amb indicacions sobre què estava fent en aquest moment. Aquest missatge inclourà informació sobre l'error que l'usuari no veurà.

L'aplicació ha de ser portable i aprofitar la mateixa base de codi per a les versions de Windows, Linux i MacOS.

Requisits de seguretat

En aquest apartat hem de considerar els diferents aspectes de la seguretat, com ara el control d'accés (qui pot accedir a quines funcionalitats, en quines circumstàncies ho pot fer, i sobre quines dades), la integritat o la privacitat.

Exemple

El professor d'un grup d'una assignatura *X* podrà accedir a totes les notes d'un estudiant del seu grup en aquesta assignatura (no a les de la resta d'assignatures o a les dels estudiants d'altres grups) mentre duri el semestre en què l'estudiant està matriculat del seu grup.

També entrarien en aquesta categoria els requisits relacionats amb la informació d'auditoria: quines accions dels usuaris o esdeveniments interns del sistema han de quedar enregistrats i quina informació s'ha d'enregistrar en cada cas.

Exemple

Cada vegada que un usuari s'intenti identificar al sistema i no ho aconsegueixi, quedarà registrat en un fitxer d'auditoria: el nom de l'usuari, l'adreça des d'on s'ha connectat, la data i l'hora.

Finalment, els requisits d'immunitat són aquells que ens diuen com es defensarà el sistema davant d'un atac.

Exemple

Si un usuari supera el límit de 5 peticions per segon, la seva adreça quedarà bloquejada i no se li permetrà fer més peticions al sistema durant 30 segons.

Requisits culturals i polítics

Els requisits culturals són especialment importants quan l'equip de desenvolupadors no està familiaritzat amb la cultura dels usuaris potencials del producte. Per exemple, podríem recollir la manera de localitzar un programa prèviament internacionalitzat per a una cultura en concret.

Exemple

Les dates es mostraran en format AAAA/MM/DD.

El sistema farà servir el sistema mètric decimal.

Pel que fa als requisits polítics, hem de considerar quina pot ser la reacció dels diferents *stakeholders* a algunes de les decisions que es vulguin prendre, com podria ser externalitzar (o no) part del desenvolupament o fer servir (o no) components de programari lliure.

Exemple

Com que la universitat vol potenciar l'ús del programari lliure, no es comprarà cap llicència de cap producte, biblioteca o bastiment per al qual hi hagi un substitut de programari lliure.

L'aplicació web ha de funcionar amb el navegador *X* versió *Y* perquè és la que té instal·lat el director general.

Requisits legals

L'últim grup de requisits recull les obligacions legals del sistema per desenvolupar, és a dir, quines lleis o quins estàndards s'han de complir.

Exemple

El sistema del campus virtual ha de tenir en compte el compliment de la LOPD pel que fa al tractament de les dades personals d'estudiants i altres membres de la comunitat.

2.3. Dependències entre requisits

Tot sovint, un requisit no és totalment independent dels altres. Pot passar que per a poder satisfer un requisit sigui necessari satisfer-ne, abans, un altre. Així, per exemple, que els estudiants vulguin lliurar les activitats en qualsevol format implica que els estudiants han de poder lliurar activitats, i no podrem implementar el requisit dels formats si no implementem abans el requisit del lliurament d'activitats. De manera similar, un requisit podria ser un subconjunt d'un altre requisit; és a dir, si es compleix el primer automàticament es compleix el segon.

També pot passar que l'esforç necessari per a satisfer un requisit depengui (de manera positiva o negativa) de si se n'ha satisfet un altre o no. Implementar un sistema de pagament electrònic amb targeta de dèbit, per exemple, és més fàcil si ja hem implementat el pagament amb targeta de crèdit que si no ho hem fet. De la mateixa manera, el valor o la importància d'un requisit pot dependre d'un altre requisit.

Siguin quines siguin, serà important tenir en compte aquestes dependències durant l'obtenció de requisits, per tal de poder-les tenir en compte a l'hora d'estimar, prioritzar i seleccionar els requisits.

3. Gestió de requisits

Un cop obtinguda la llista de requisits candidats ens caldrà decidir quins, entre els requisits candidats, són aquells que volem preveure. Per a això, primer ens caldrà estimar el cost de cada requisit en temps i recursos de desenvolupament. També necessitarem saber quin valor o importància té cada requisit per als diferents *stakeholders*. Amb aquestes dues dades ja podrem seleccionar la llista definitiva de requisits.

Una bona gestió de requisits ens permetrà maximitzar el valor obtingut pel projecte de desenvolupament i, per tant, és una part fonamental de la gestió del projecte.

3.1. Estimació de requisits

Hi ha moltes tècniques per a estimar el cost d'implementació d'un requisit. La fiabilitat de totes, però, dependrà de la qualitat de la informació de partida (és més fiable, per exemple, si estimem d'acord amb informació històrica real en comptes de fer-ho amb una altra estimació) i de l'encert que tinguem a l'hora de fer servir aquesta informació.

Un dels problemes habituals durant l'estimació d'un requisit és la diversitat d'opinions: el que una persona pot veure molt complicat una altra ho pot veure d'allò més senzill. Aquests conflictes acostumen a estar determinats per una asimetria en la informació (potser una de les dues persones coneix una certa informació que l'altra desconeix o viceversa).

Per a minimitzar els conflictes en l'estimació és necessari que ens assegurem que tothom treballa amb la mateixa informació.

Pagaments electrònics

Una empresa vol afegir la possibilitat de fer pagaments electrònics des de la seva aplicació. Per als desenvolupadors, aquest requisit és relativament poc costós d'implementar perquè ja tenen experiència amb aquests tipus de sistemes i creuen que poden aprofitar alguns components que tenen desenvolupats per a altres projectes, per la qual cosa creuen que el poden tenir enllestit en una setmana, mentre que per als clients, en canvi, és un procés costós perquè implica que hauran de signar un contracte amb l'empresa proveïdora del sistema de pagaments i aquest contracte haurà de passar per tot un procés burocràtic que s'allargarà tres mesos i no serà fins d'aquí a tres mesos que es podran començar a fer proves d'integració. En aquest cas, l'estimació dels desenvolupadors és incorrecta perquè no tenen la informació que té el client.

3.1.1. Unitats d'estimació

En el moment de fer l'estimació del cost d'un requisit, ho hem de fer en alguna unitat. Aquesta unitat pot tenir una contrapartida real (per exemple, si estimem en hores o dies de feina) o bé ser totalment imaginària (per exemple, si estimem amb "punts").

Totes les unitats tenen els seus avantatges i inconvenients però el que és especialment important és que tothom faci servir la mateixa unitat i que, si volem comparar amb informació prèvia d'altres projectes o altres requisits del mateix projecte, aquesta estigui en la mateixa unitat.

Les unitats "reals" tenen l'avantatge de donar un cost intel·ligible. Per exemple, si diem que un projecte necessitarà 2.000 hores de desenvolupament, ens podem fer una idea dels recursos necessaris per a executar-lo i també del calendari, mentre que si diem que necessitarà 358 punts no podem fer aquest càlcul.

En canvi, l'avantatge de les unitats "fictícies" és que ens obliguen a fer servir informació històrica real per tal de fer el càlcul de recursos necessaris i de calendari previst. Així mateix, ens faciliten ajustar les estimacions al llarg del projecte. Per exemple, si hem estimat un conjunt de requisits en 5, 4 i 7 punts, respectivament, i ens adonem que estem necessitant 6 hores de desenvolupament per punt (en lloc de les 8 que havíem previst), podem calcular ràpidament els nous recursos i el calendari.

3.1.2. Comparació i triangulació

La tècnica més bàsica per a l'estimació d'un requisit consisteix a fer abstracció de la feina requerida per a implementar el requisit, trobar-ne un o més que considerem equiparables (i per als quals ja tinguem una estimació) i prendre aquest valor com a referència de l'estimació.

El principal inconvenient d'aquesta tècnica és que es necessiten estimacions prèvies per tal de poder comparar, de manera que no la podem aplicar a les primeres estimacions.

Per a fer l'estimació més fiable, podem complementar la comparació amb la triangulació, que consisteix a comparar el requisit amb un de més complicat i un de més senzill: si, per exemple, tenim un requisit que hem estimat en quatre punts i un altre que hem estimat en dos, un tercer requisit que estiméssim en tres punts hauria de ser més gran que el de dos i més petit que el de quatre.

Utilitat de la triangulació

La triangulació és especialment útil en el cas de fer servir unitats fictícies per tal d'assegurarnos que el valor de la unitat d'estimació es manté més o menys constant en totes les estimacions que fem.

3.1.3. *Planning poker*

El *planning poker* és una tècnica àgil d'estimació col·laborativa que rep el nom del fet que, igual que en el joc del pòquer, cadascun dels individus fa la seva "aposta" sense saber què pensen els altres i, al final, cadascú "mostra les seves cartes".

El *planning poker* consisteix que cadascun dels participants faci una estimació del cost del requisit sense saber les estimacions que han fet els altres (en un rol o, si es vol, amb una carta d'una baralla especial de *planning poker*).

Les cartes del *planning poker* tenen valors inspirats en la sèrie de Fibonacci per a reflectir el fet que, a mesura que l'estimació és més gran, és menys precisa. Per exemple, té molt més sentit discutir si un requisit té un cost 1 o un cost 2 que no pas si és 21 o 22.

Un cop tothom ha fet la seva estimació, "s'aixequen les cartes" i, si no hi ha consens, s'explica a la resta de parts el motiu pel qual s'ha fet una estimació més gran o més petita que la de la resta.

Un cop s'han discutit el requisit i els motius es torna a fer l'estimació en secret i es torna a posar en comú. Aquest procés es va repetint fins que hi ha consens entre totes les parts.

El més important d'aquesta tècnica és que cadascú pugui fer la seva estimació sense estar condicionat per les estimacions dels altres (per això l'estimació és secreta inicialment) i que, en cas de no coincidir, s'estableixi un diàleg que permeti assegurar que tothom treballa amb la mateixa informació.

És important també arribar a un consens i, en cap cas, caure en la temptació de fer una mitjana entre les diferents estimacions, ja que les estimacions diferents han d'estar basades, per força, en supòsits diferents.

3.2. Prioritització de requisits

Un cop tenim una idea aproximada del cost de cada requisit necessitem calcular el valor aportat. L'avantatge en aquest cas és que, a diferència del cost, no necessitem una mesura absoluta del valor del requisit sinó una mesura relativa a la resta de requisits. Per això parlem de priorització i no pas d'estimació del valor.

Exemple

Per exemple, al nostre campus virtual els professors podrien considerar més important poder saber qui ha fet un lliurament d'activitat que no pas en quin format de fitxer l'ha fet.

D'altra banda, hem comentat anteriorment que de vegades hi ha requisits que entren en conflicte amb altres; en aquest cas, necessitem saber quin és el que aporta més valor per tal que el nostre sistema el compleixi.

Sembla, doncs, que serà relativament senzill prioritzar els requisits: només els hem d'ordenar en funció del seu valor. El problema, però, és que el valor és relatiu als *stakeholders*, ja que, el requisit que per a un *stakeholder* és molt important, per a un altre pot no ser-ho tant o, fins i tot, un requisit que un *stakeholder* considera important incloure al producte, un altre pot considerar important que no s'inclogui.

Un altre problema habitual és que als *stakeholders* els costi prioritzar els requisits (és difícil decidir a quins requisits s'està disposat a renunciar). A continuació expliquem una de les tècniques que es fan servir per a prioritzar els requisits.

3.2.1. Votació amb nombre limitat de vots

La tècnica dels 100 dòlars (Leffingwell i Widrig, 2000) consisteix a donar 100 dòlars imaginaris a cadascun dels *stakeholders* i dir-los que els reparteixin entre els diferents requisits. Idealment, cada *stakeholder* repartirà els seus dòlars entre els requisits segons el valor de cada un. Així, si un requisit és molt important, li assignarà molts dòlars per tal d'assegurar-se que surti escollit, mentre que si no ho és, li n'assignarà menys (o cap).

D'aquesta manera donarem prioritat a aquells requisits que, o bé són molt importants per a alguna de les parts (que haurà posat una gran quantitat de dòlars per a assegurar-se que el sistema compleix aquest requisit) o bé són importants per a moltes de les parts implicades (i, per tant, per acumulació de petits imports al final aconsegueixen sumar un import gran).

Un problema d'aquesta tècnica és que pot passar que un requisit important quedi massa avall a la llista perquè només és important per a un subconjunt petit dels *stakeholders*.

D'altra banda, caldrà assegurar-se que els *stakeholders* votin de manera honesta i, per exemple, evitin prioritzar de menys un requisit perquè saben que altres ja el prioritzaran.

3.3. Selecció de requisits

Un cop equipats amb la nostra llista prioritzada i estimada de requisits podem procedir a la tria. Aquest procés s'haurà de fer per a cadascun dels lliuraments del projecte; en el cas de les metodologies iteratives es farà, també, a l'inici de cada iteració.

La manera més senzilla d'establir quins requisits cal incloure al lliurament següent és començar pels més prioritaris i anar afegint requisits a la llista mentre quedi capacitat (temps i recursos) disponible per a implementar-los.

Un cop ens trobem amb un requisit que no podem incloure perquè seria massa costós, podem decidir entre treure'n algun de la llista per fer-li lloc (tot i que en principi el requisit que estem afegint aporta menys valor que el que estem traient) o passar al següent en prioritat i continuar aplicant el mateix criteri fins que no quedi temps ni recursos disponibles.

A l'hora de triar quins requisits cal implementar hem de tenir en compte la prioritat, el cost i els recursos disponibles.

Un altre factor que cal tenir en compte en la tria són els riscos associats a cada requisit; l'acció per prendre dependrà de la naturalesa del risc i de la seva evolució al llarg del temps.

Importància del risc

El risc és un factor tan important que alguns mètodes de desenvolupament (com, per exemple, UP) aconsellen implementar aquests requisits fins i tot abans de comprometre's a desenvolupar tot el projecte (és a dir, durant la fase d'elaboració del projecte).

Per exemple, si hem d'aplicar una tecnologia amb la qual tenim poca experiència, com que aquest risc no desapareixerà fins que abordem el requisit, segurament ens interessarà tractar-lo el més aviat possible de manera que, si ens trobem amb problemes irressolubles, la inversió feta hagi estat mínima.

En canvi, si el risc que hem detectat és que considerem que és poc probable que un tercer (per exemple, un proveïdor) ens lliuri un component dins del termini establert, potser ens interessa deixar aquest requisit per un pròxim lliurament en què el risc s'hagi mitigat o hagi desaparegut (per exemple, perquè ja ens han lliurat el component).

També hem de tenir en compte el mercat potencial del requisit (entenent com a mercat el grup de persones interessades en aquest). Seleccionant un grup o altre de requisits estem condicionant el nostre mercat potencial; per exemple, si decidim no incloure suport per a internacionalització i localització del nostre sistema, estem reduint el mercat a un únic entorn geogràfic.

D'altra banda, també hem de considerar la finestra de mercat (el període de temps durant el qual el nostre producte és interessant); un mateix producte pot ser absolutament innovador un any i no ser-ho en absolut si surt al mercat dos anys més tard; aquesta finestra de mercat també afectarà el tipus de persona que voldrà comprar el nostre producte.

Entre el mercat potencial i la finestra de mercat hi ha una relació que cal tenir en compte: si afegim funcionalitat estarem ampliant el mercat potencial però, al mateix temps, estarem endarrerint la introducció del nostre producte, mentre que, si traiem funcionalitat, estarem avançant la introducció del producte al mercat a canvi d'escurçar-ne la mida.

En el cas de desenvolupar un sistema per a la comercialització, també hem de considerar el preu que els usuaris estan disposats a pagar pel producte en cada moment i la competència que podríem trobar; moltes vegades un producte inferior ha aconseguit apoderar-se d'un mercat gràcies a haver estat el primer a arribar al mercat o a haver-ho fet amb un cost menor.

Finalment, un altre factor econòmic molt important és el retorn de la inversió[3]. El podríem definir com la mesura de la manera en què l'ús del producte genera ingressos per a compensar el cost de la inversió en el seu desenvolupament.

[3] En anglès, *return of investment* (ROI).

Exemple de retorn de la inversió

Si una empresa gasta un milió d'euros en un nou sistema informàtic, aquest hauria de generar un milió d'euros en beneficis (sigui estalviant costos o generant nous ingressos) per tal de poder considerar la inversió com a recuperada.

Potser un requisit es considera molt important però és difícil justificar, en termes econòmics, el seu retorn i, per tant, podria interessar seleccionar un altre requisit que generi beneficis de manera més immediata.

4. Documentació dels requisits

Anomenem *especificació* l'artefacte, típicament un document textual, que documenta el conjunt de requisits que s'han seleccionat per al projecte.

L'especificació de requisits recull el contracte entre els desenvolupadors i els *stakeholders* i serveix com a eina de comunicació per a tots dos grups. Per això és un element central de qualsevol mètode de desenvolupament, especialment pel que fa a la gestió del projecte.

L'estil i formalitat de l'especificació dependrà del projecte però també del context en què es desenvolupa i de les persones que hi participen.

Per exemple, el mètode *extreme programming* ens recomana tenir un representant dels usuaris integrat dins l'equip de desenvolupament. En aquest cas, el nivell de detall i formalitat necessaris són molt baixos, ja que disposem d'una persona que pot aportar detalls i desambiguar quan sigui necessari. En aquest tipus de projectes la documentació sol prendre la forma d'una llista de requisits, per a cada un dels quals tenim només aquella documentació essencial per tal de poder reprendre la comunicació verbal quan calgui.

A l'altre extrem, hi ha empreses que elaboren l'especificació de requisits en un país i en subcontracten a una altra empresa en un altre país (típicament en una altra zona horària) la implementació. En aquest altre cas ens caldrà un document molt més inambigu i detallat, ja que la comunicació no és tan fluïda. És probable que en un cas així ens plantegem la utilització d'un llenguatge formal d'especificació com ara OCL.

Llenguatge OCL

Object constraint language (OCL) és un llenguatge formal per a expressar restriccions en models UML i, igual que UML, el defineix l'Object Management Group (OMG).

Si, per exemple, volem dir que no hi pot haver dos clients amb el mateix número de telèfon, en OCL diríem *Client.allInstances()->isUnique(numTelefon)*.

Un altre factor que cal tenir en compte és el mètode de desenvolupament que estem fent servir i el cost d'introduir un canvi en els requisits. Així, no requeriran el mateix estil de documentació de requisits el programari que controla una sonda espacial (que un cop llençada a l'espai és molt difícil, si no impossible, d'actualitzar) que una aplicació web en què és relativament senzill desplegar una nova versió.

Finalment, hi ha casos especials, com ara quan disposem d'una implementació del sistema que volem desenvolupar (perquè en tenim una implementació de referència o perquè estem desenvolupant un sistema nou per a substituir-ne un d'existent). En aquest cas la necessitat de documentació serà menor, ja que en cas de dubte podem acudir al sistema existent per a conèixer-ne el funcionament.

Una de les tasques de l'enginyer del programari serà, doncs, avaluar quina és la necessitat de formalitat i detall a l'hora de documentar els requisits per a un projecte. Amb aquesta informació, el cap del projecte podrà decidir quin esforç (i, per tant, quin percentatge del pressupost global del projecte) caldrà dedicar a la documentació de requisits: haurà de tenir en compte que tan perillós és intentar desenvolupar un projecte sense prou documentació de requisits, com dedicar els recursos i temps disponibles per a crear una documentació innecessàriament detallada o formal.

4.1. Qualitats d'una bona especificació de requisits

L'estàndard IEEE 830-1998, titulat "Pràctica recomanada per IEEE per a les especificacions de requisits de programari", documenta, entre altres coses, quines haurien de ser les qualitats d'una bona especificació de requisits. Tot i que pressuposa un procés en cascada, en què el resultat de l'especificació de requisits és un document no ambigu i complet, les guies que dóna poden ser aplicades també en altres cicles de vida.

Segons aquest estàndard una especificació de requisits hauria de ser:

- **Correcta**. Una especificació de requisits és correcta si tots els requisits que documenta ho són realment; és a dir, si tots els requisits enumerats són necessitats que el programari ha de satisfer. Aquesta és, probablement, la propietat més difícil de comprovar, ja que, en realitat, no sabrem si l'especificació és correcta fins que el programari estigui funcionant i els *stakeholders* vegin realment satisfetes les seves necessitats.

- **No ambigua**. L'especificació de requisits és inambigua si cada requisit enumerat té una única interpretació possible. Per a aconseguir aquest objectiu cal fer servir un llenguatge molt clar i, en cas de termes que puguin portar a múltiples interpretacions, definir-los correctament en un glossari. Una altra aproximació és fer especificacions de requisits mitjançant llenguatges formals. En tot cas, cal tenir molt present que l'especificació ha de ser inambigua tant per a qui l'escriu com per a qui la llegeix i que, probablement, aquests dos grups poden tenir un *background* diferent i poden entendre de manera diferent un mateix llenguatge.

- **Completa**. Perquè sigui completa, l'especificació de requisits ha d'enumerar tots els requisits de tots els *stakeholders*, definir el comportament del sistema especificat per a tot tipus d'entrades de dades i situaci-

ons i, finalment, etiquetar i referenciar totes les taules, figures i diagrames del document. Cal tenir en compte que la completesa és molt difícil d'aconseguir, ja que el nombre de combinacions de situacions i dades d'entrada que facin que el sistema es comporti d'una o altra manera pot ser infinit i, per tant, és fàcil que ens deixem algun cas. A més, cal distingir entre els requisits que hem seleccionat per al nostre sistema (que sí que necessitem que estiguin tots) i els requisits que havíem identificat per al nostre sistema abans de fer la selecció (que sovint no cal que quedin documentats o, si més no, no amb tant detall).

- **Consistent**. Diem que una especificació de requisits és consistent si cap subconjunt dels requisits enumerats entra en conflicte; per tant, una especificació no consistent no és realitzable, ja que no podem desenvolupar cap programari que satisfaci requisits contradictoris.

- **Requisits etiquetats**. Cada requisit de l'especificació hauria de ser etiquetat amb informació rellevant per a la gestió de requisits, com pot ser la importància i el cost. Un altre factor que pot ser important és l'estabilitat, ja que alguns requisits seran més estables que altres, en el sentit que s'espera que pateixin menys canvis al llarg del temps.

- **Verificable**. Una especificació és verificable si ho és cada un dels requisits que enumera, i un requisit és verificable si hi ha algun procés finit i de cost efectiu per a determinar si un programari satisfà o no el requisit.

- **Modificable**. Considerem que una especificació de requisits és modificable si l'estructura i la redacció permeten fer canvis de manera fàcil i consistent. Per a això, el document ha d'estar ben estructurat (amb la seva taula de continguts, índex, referències creuades, etc.) i no ser redundant. Tot i que la redundància no és, per si mateixa, un error, pot portar a errors quan es modifica l'especificació de requisits, ja que un canvi en alguna part del document que no actualitzi també allò que és redundant amb el que s'ha canviat, introduiria inconsistències.

- **Traçable**. Finalment, una especificació és traçable si cada requisit enumerat està clarament identificat i facilita la referència en els artefactes desenvolupats al llarg del projecte, ja siguin altres documents (per exemple, de disseny), el programari desenvolupat, les proves de programari, etc. La traçabilitat és especialment important quan el projecte està a mig desenvolupar o ja totalment desenvolupat, ja que, si un requisit canvia, cosa que passa sovint, ens interessarà saber quins documents, programari, proves i altres materials caldrà canviar com a conseqüència.

4.2. Bones pràctiques

Però com aconseguim un document que tingui totes les qualitats d'una bona especificació de requisits de programari? A continuació enumerem algunes bones pràctiques que ens poden ajudar a assolir aquest objectiu:

1) Identificadors de requisits. Sovint necessitarem fer referència a un requisit en algun altre document o en algun altre punt del mateix document. Per exemple, si volem proporcionar traçabilitat, ens caldrà poder referir-nos als requisits des de qualsevol altre artefacte. En aquest sentit, és molt convenient triar un identificador que descrigui de manera breu el valor més important del requisit, més que no pas un identificador numèric. Això evita que acabem parlant de manera excessivament abstracta sobre el requisit 23.4 o el 34A, i pot augmentar, a més, la modificabilitat.

2) Punt de vista global. De vegades, identifiquem com a requisits necessitats des del punt de vista tècnic, que ens parlen d'una part per implementar del sistema més que no pas del conjunt del sistema vist com un tot per part dels usuaris i la resta de *stakeholders*. És el cas, per exemple, de necessitats com ara "la informació de matrícula indicada per l'alumne s'ha de validar i desar a la base de dades". Cal evitar escriure necessitats de parts del sistema des del punt de vista del desenvolupador, ja que satisfer una d'aquestes necessitats no aporta, per si mateix, cap valor als usuaris finals i la resta de *stakeholders* del sistema.

3) Granularitat. Convé definir els requisits amb un nivell de granularitat adequat, de tal manera que no tinguem una llista massa llarga de requisits massa detallats però tampoc una llista molt curta de requisits massa poc detallats. Tot i que l'IEEE recomana que l'especificació de requisits sigui completa, la granularitat i detall exactes depenen molt de les necessitats de cada equip i del coneixement que tingui d'aquests requisits.

En un procés iteratiu i incremental, per exemple, voldrem que els requisits que estem a punt d'implementar a la propera iteració estiguin força detallats i complets, però no ens caldrà que ho estiguin, encara, els requisits als quals els falta molt per a ser implementats. En un procés en cascada, en canvi, voldrem que tots els requisits es documentin amb la granularitat més fina i amb gran nivell de detall.

4) Interfície gràfica d'usuari. De vegades donem per suposada una interfície gràfica d'usuari, a l'hora d'identificar requisits, que encara no hem dissenyat. Així, per exemple, indicar, en un requisit, que l'usuari ha de poder seleccionar l'assignatura d'un desplegable, pot ser indicar massa informació sobre la interfície gràfica d'usuari, sobretot si encara no s'ha dissenyat aquesta interfície.

El problema d'aquest tipus de requisits és que pressuposen una solució (en aquest cas, l'ús d'un desplegable) que, potser, no és la millor i que, sobretot, no ha de ser necessàriament una necessitat per a l'usuari. En el nostre exemple, pot ser que a l'usuari li vagi bé introduir el nom de l'assignatura en un camp de text que autocompleta o seleccionar-la d'una llista (en lloc de fer-ho d'un desplegable).

5) Condicions d'acceptació. Una de les qualitats que perseguim és que els requisits siguin verificables. Una bona manera d'assegurar-se'n és no oblidar-se de documentar clarament, per a cada requisit, les condicions d'acceptació. D'aquesta manera estarem apuntant el procés finit i de cost efectiu que seguirem per a verificar cada requisit.

6) Ús de plantilles. Una altra bona pràctica especialment útil per a aconseguir documents d'especificació de requisits ben estructurats (i, per tant, modificables) i complets, és l'ús d'una plantilla que ens vagi indicant els apartats que cal documentar. En aquest sentit, l'estàndard IEEE 830-1998 ja mencionat o la plantilla d'especificació de requisits Volere són dos bons punts de partida.

Exemple d'ús de plantilles

La plantilla Volere, per exemple, ens dóna un índex dels continguts que ha de contenir l'especificació, una petita plantilla sobre com cal documentar cada requisit i, per a cada apartat de l'índex, una descripció del contingut, una motivació sobre la seva necessitat, alguns exemples de contingut per a aquella secció i, finalment, altres consideracions d'interès. L'índex que ens proposa aquesta plantilla és el següent:

Conductors del projecte
1. Propòsit del projecte
2. Els clients i altres *stakeholders*
3. Usuaris del producte
Restriccions del projecte
4. Restriccions obligatòries
5. Convencions de noms i definicions
6. Fets i assumpcions rellevants
Requisits funcionals
7. Abast del projecte
8. Abast del producte
9. Requisits funcionals i de dades
Requisits no funcionals
10. Requisits de presentació
11. Requisits d'usabilitat i humanitat
12. Requisits de rendiment
13. Requisits operacionals i d'entorn
14. Requisits de manteniment i suport
15. Requisits de seguretat
16. Requisits culturals i polítics
17. Requisits legals
Altres qüestions
18. Temes oberts
19. Solucions prefabricades
20. Problemes nous
21. Tasques
22. Migració al nou producte
23. Riscos
24. Costos
25. Documentació d'usuari i formació
26. Sala d'espera
27. Idees i solucions

4.3. Històries d'usuari

Les històries d'usuari són l'eina de documentació més habitual en els mètodes àgils. Els components bàsics d'una història d'usuari són:

- Una descripció escrita de la història (serveix com a recordatori que existeix la història i és útil per a planificar, etc.).
- Un seguit de converses que serveixen per a definir i aclarir els detalls de la història.
- Un conjunt de proves que documenten els detalls i que permeten determinar quan la història està implementada completament.

De vegades, les històries d'usuari es documenten en targetes de cartró i, per això, es fa servir el terme *CCC* (*card, conversation, confirmation*). És important entendre, però, que la targeta no documenta la història sinó que només la representa (els detalls estan en les converses i en les proves).

Exemple d'història d'usuari

Un exemple d'història d'usuari seria "Com a alumne, vull poder lliurar un exercici per mitjà del campus virtual" (això seria la descripció, el que posaríem a la targeta). Aquesta història hauria d'anar acompanyada d'un seguit de converses entre els desenvolupadors i els *stakeholders* per tal d'anar perfilant detalls com ara quins formats de document són acceptats, si els professors poden o no imposar un format, si cal que el sistema verifiqui les dates límit automàticament, etc. Tots aquests detalls, finalment, s'hauràn de recollir en un conjunt de proves que ens han de permetre determinar si la història s'ha implementat amb èxit o no:

- Verificar que si s'adjunta un arxiu en format PDF, DOC, DOCX o ODT aquest es desa i s'associa a l'usuari.
- Verificar que si el professor limita els formats, només s'accepten lliuraments en els formats indicats pel professor.
- Verificar que si s'adjunta un arxiu en un format incorrecte s'avisa a l'usuari.
- Verificar que si s'intenta fer el lliurament fora de les dates de l'activitat es mostra un error a l'usuari.
- Verificar que si l'usuari surt sense adjuntar l'arxiu el sistema l'avisa i li demana confirmació.
- Verificar que només es pot fer el lliurament en les assignatures de les quals està matriculat l'usuari.

El principal avantatge i, al mateix temps, el principal inconvenient, de les històries d'usuari, és que estan molt enfocades a la comunicació verbal. Això permet que la comunicació sigui més àgil i més fluïda, amb la qual cosa els requisits seran més propers a les necessitats reals dels *stakeholders*, però, en canvi, no queden registrades, amb la qual cosa no tenim tots els detalls per escrit.

Cal tenir en compte, però, que la conclusió final de les converses sí que queda documentada, i a més, en un format que en facilita la validació, ja que estan documentades en forma de proves.

FitNesse i Cucumber

Hi ha eines al mercat com FitNesse i Cucumber que faciliten l'execució automatitzada de proves a partir de la seva documentació.

Un altre avantatge de les històries d'usuari és que estan escrites en el llenguatge dels usuaris o *stakeholders*. De fet, són els usuaris o *stakeholders* mateixos els que, idealment, haurien d'escriure les històries d'usuari.

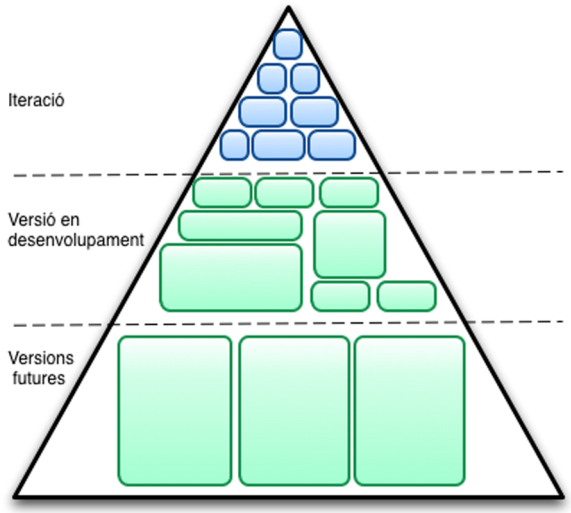
La pila de producte[4] és la llista total d'històries d'usuari d'un projecte de desenvolupament. Dins d'aquest *backlog* podem trobar històries molt grans (anomenades *Epics*), com ara "Hi haurà un mecanisme de registre d'avaluació continuada", o bé històries molt petites, com ara "Un professor pot limitar els formats de lliurament d'una activitat".

(4)En anglès, *product backlog*.

La pila de producte ideal ha de tenir una forma anomenada *d'iceberg*: la part de dalt són les històries que tindrem en compte en la iteració actual (i, per tant, com la punta d'un iceberg, les que són "visibles"). Aquestes històries són més petites i detallades.

La base de l'iceberg (la part "no visible") són històries més grans, menys detallades, que encara no implementarem i que, més endavant, caldrà dividir en històries més petites per a poder-les implementar.

Mida de les històries d'usuari en la pila de producte



Les històries d'usuari són molt útils per als requisits funcionals però també ens poden ser útils per als requisits no funcionals.

Històries d'usuari i requisits no funcionals

Com a director tècnic de l'empresa, vull que altres aplicacions puguin accedir a la base de dades en el futur.

Com a professor, vull saber la data de lliurament amb precisió de minuts.

Com a director tècnic de la universitat, vull que l'aplicació funcioni sobre qualsevol sistema operatiu d'aquesta llista: Linux, Windows, BSD.

5. Casos d'ús

Els casos d'ús són una tècnica de documentació de requisits molt estesa, entre altres motius, perquè UML hi dóna suport. Es tracta d'un enfocament a la manera de documentar requisits que permet fer servir diversos graus de detall i de formalisme, la qual cosa els fa adequats en escenaris molt diversos.

Vegeu també

Podeu trobar més informació sobre l'estàndard UML als mòduls "Introducció a l'enginyeria del programari" i "Anàlisi UML".

5.1. Què és un cas d'ús?

> Un cas d'ús "[...] és una descripció d'un conjunt de seqüències d'accions, incloent-ne variants, que executa un sistema per a produir un resultat observable i de valor per a un actor".
>
> Diversos autors (2010). *Unified Modeling Language™ (UML®)*. Object Management Group.

Cockburn defineix els casos d'ús des del punt de vista de la seva funció.

> "Un cas d'ús captura un contracte entre els *stakeholders* d'un sistema pel que fa al seu comportament [...]. El sistema respon [...] protegint els interessos dels *stakeholders*."
>
> Alistair Cockburn (2001). *Writing Effective Use Cases*. Addison-Wesley.

Per tant, podem dir que un cas d'ús recull el contracte entre el sistema i els *stakeholders* mitjançant la descripció del comportament observable del sistema.

En descriure un cas d'ús, hi ha un *stakeholder* que és especial (el qual anomenarem **actor principal**), que és qui vol fer servir el sistema per a satisfer un objectiu concret. El cas d'ús ens descriu quin és el comportament observable del sistema durant aquesta execució del cas d'ús, de manera que puguem assegurar que es compleix l'objectiu de l'actor principal tenint en compte els interessos de la resta de *stakeholders*.

Sistema informàtic, actor principal

Es podria donar el cas que l'actor principal no fos un *stakeholder* sinó un sistema informàtic. Per exemple, si estem desenvolupant un servidor de correu IMAP sense interfície gràfica, l'actor principal de la majoria de casos d'ús és una aplicació client de correu però l'*stakeholder* seria la persona que fa servir l'aplicació client.

Una característica interessant dels casos d'ús és que, com que no estem parlant del comportament intern del sistema, podem descriure la seva funcionalitat sense necessitat de descriure com s'implementa el comportament.

Quin aspecte té un cas d'ús? A continuació en posem un exemple.

Cas d'ús: llegir un missatge del fòrum (nivell usuari)

Un usuari demana llegir un missatge del fòrum i el sistema li mostra el tema i el contingut. El sistema també enregistra que l'usuari ha llegit el missatge.

Si l'usuari vol escriure una resposta, el sistema li demanarà el tema i el contingut de la resposta i enregistrarà un missatge associat al missatge original amb aquest tema i contingut i la data actual.

La descripció del cas d'ús anterior és molt informal i poc detallada. En alguns casos, necessitarem documentar de manera més detallada el cas d'ús. A continuació podem veure el mateix cas d'ús documentat amb un format més adequat a aquests casos. De moment no ens preocuparem per saber què vol dir cadascun dels apartats, ja que ho veurem més endavant.

CU001: llegir un missatge del fòrum
Actor principal: usuari
Àmbit: campus virtual
Nivell d'objectiu: usuari
Usuari: vol llegir el missatge.
Professor responsable de l'aula: vol llegir el missatge i saber quins estudiants l'han llegit.
***Stakeholders* i interessos**:
Precondició: l'usuari s'ha d'haver identificat al sistema.
Garanties mínimes: el sistema enregistrarà l'intent de lectura del missatge.
Garanties en cas d'èxit: el sistema mostrarà a l'usuari el contingut del missatge i n'enregistrarà la lectura.
Escenari principal d'èxit:
1. L'usuari indica quin missatge vol llegir.
2. El sistema enregistra l'intent de lectura del missatge per part de l'usuari.
3. El sistema valida que l'usuari tingui accés al fòrum.
4. El sistema mostra el tema i el contingut del missatge.
5. El sistema enregistra que l'usuari ha llegit el missatge.

Extensions:
1a. L'usuari tanca l'ordinador.
1a1. El sistema enregistra la lectura del missatge encara que l'usuari potser no l'ha vist.
5a. L'usuari vol escriure una resposta a un missatge.
5a1. L'usuari indica que vol escriure una resposta a un missatge.
5a2. El sistema demana el tema i el contingut de la resposta, suggerint *RE:<Tema del missatge al qual responem>* com a tema del missatge.
5a3. L'usuari modifica el tema si vol i introdueix el contingut de la resposta.
5a4. El sistema enregistra la resposta amb el tema i contingut introduïts, assigna la data actual com a data de publicació del missatge i l'associa al missatge original.
5a3a. L'usuari introdueix un tema o una resposta buits.
5a3a1. El sistema indica que no es pot deixar buit el tema ni la resposta i tornem al punt 2a3.
5b. La base de dades no està disponible (error de servidor).
5b1. El sistema indica a l'usuari que no ha estat possible recuperar el missatge i li demana que ho torni a provar passada una estona.
5c. El missatge té documents adjunts i l'usuari en vol baixar un.
5c1. L'usuari indica quin document adjunt vol baixar.
5c2. El sistema baixa el document adjunt a l'ordinador de l'usuari.

Com podem veure, tot i que el concepte de cas d'ús és molt senzill, la seva descripció es pot complicar força. Per això mateix, és habitual combinar diversos estils de descripció dels casos d'ús en un projecte, deixant aquest nivell tan detallat per als casos d'ús més complicats.

5.2. Actors i *stakeholders*

Segons hem vist fins ara, els casos d'ús estan molt lligats als *stakeholders*. Hem mencionat també els "actors", dient que hi ha un *stakeholder* que actua com a actor principal del cas d'ús. Això ens podria portar a pensar que els actors i els *stakeholders* són la mateixa cosa, tot i que això no és cert.

Tal com indica Cockburn (2001): "Un *stakeholder* és algú que participa en el contracte. Un actor és algú que té comportament". Per tant, mentre que els *stakeholders* participen en el cas d'ús mitjançant els seus interessos (per exemple, el professor vol que quedi registrat qui ha llegit el missatge) els actors interactuen amb el sistema.

Un actor pot ser una persona, una organització o un sistema informàtic: pot ser qualsevol cosa que tingui capacitat d'interactuar amb el nostre sistema i de tenir un comportament propi.

> Més formalment, "un actor representa un conjunt coherent de rols que els usuaris del cas d'ús tenen en interactuar amb aquest".
>
> Diversos autors (2010). *Unified Modeling Language™* (*UML®*). Object Management Group

Per tant, un actor no és una persona o una organització en concret sinó el conjunt de rols que aquesta persona pot tenir en relació amb el cas d'ús. Una mateixa persona, organització o sistema informàtic pot tenir diferents rols i, per tant, aparèixer com a diferents actors.

Exemple

Per exemple, un empleat d'una entitat financera pot aparèixer amb el rol *Empleat* quan es connecta al sistema informàtic des de l'ordinador de l'oficina per gestionar els comptes dels seus clients, però pot aparèixer com a rol *Client* si es connecta via Internet des de casa seva per gestionar els seus comptes propis. Tot i que es tracta de la mateixa persona, des del punt de vista del sistema informàtic es tracta d'actors diferents.

Mentre que tots els actors seran *stakeholders* (tots tenen algun interès en el comportament del sistema, ja que els afecta en interactuar-hi) també hi pot haver *stakeholders* que no siguin actors. És important identificar aquests *stakeholders* per tal d'assegurar que el comportament descrit té en compte els seus interessos.

Exemple

Al nostre exemple, és important identificar les necessitats del professor ja que, encara que aquest no participi en el cas d'ús, s'ha de tenir en compte la seva necessitat que hi hagi registre sobre qui ha llegit un missatge.

5.2.1. Actors principals i de suport

En el context d'un cas d'ús concret, anomenem actor principal aquell *stakeholder* que fa una petició al sistema per a rebre'n un dels serveis i així satisfer un objectiu.

En un mateix cas d'ús, però, a més de l'actor principal hi poden aparèixer un o més actors de suport, també anomenats *secundaris*. Aquests són actors externs al sistema que proporcionen un servei al sistema.

Els dos casos més habituals d'actors de suport són els sistemes informàtics externs (com, per exemple, un sistema per a autoritzar pagaments amb targeta de crèdit) i els usuaris que participen en el cas d'ús sense ser-ne l'actor principal (per exemple, un operador d'una centraleta de trucades que interactua amb el sistema en nom d'una persona que truca).

Operador d'una centraleta

En el cas de l'operador d'una centraleta, diem que és un actor de suport perquè el sistema no satisfà el seu objectiu sinó el de la persona que truca. Diem que l'actor principal (la persona que truca) és qui fa la petició al sistema tot i que ho faci de manera indirecta (per mitjà de l'operador).

5.3. Anatomia d'un cas d'ús

Anteriorment hem vist que hi ha diverses maneres de descriure un cas d'ús. En aquest apartat entrarem en més detall en els diferents elements que formen part del cas d'ús.

5.3.1. Nom

D'entrada, hem vist que, com a mínim, un cas d'ús ha de tenir un nom. Aquest nom és molt important ja que es farà servir per a fer referència al cas d'ús des de qualsevol artefacte al llarg del projecte. Des d'aquest punt de vista, el nom ha de recollir la informació més important relativa al cas d'ús: l'objectiu que satisfà. D'aquesta manera, només llegint el nom del cas d'ús, els *stakeholders* poden valorar quin és l'objectiu que satisfà sense haver de buscar el document on es descriu.

Cada cas d'ús ha de tenir un nom que indiqui què aconsegueix l'actor principal en la seva interacció amb el sistema.

Típicament, es descriu l'objectiu mitjançant una frase en forma activa que comenci amb un verb (millor "Llegir un missatge" que "Un missatge és llegit" o "Lectura de missatge", tot i que el que és realment important és mantenir una mateixa convenció per a tots els noms de cas d'ús d'un sistema.

De vegades s'associa un identificador alternatiu als casos d'ús (com a l'exemple, CU001), de manera que es faci més fàcil organitzar-los. Aquest identificador, però, té el problema de ser poc descriptiu, per la qual cosa és recomanable fer servir el nom per a identificar el cas d'ús sempre que sigui possible.

5.3.2. Actors

Un altre element molt important és documentar clarament els actors del cas d'ús i quin d'ells és l'actor principal.

5.3.3. Objectius i àmbit

Tan important com identificar els actors és identificar els seus objectius. Però, com hem vist abans, no en tenim prou amb els objectius i interessos dels actors, sinó que hem de tenir en compte els de tots els *stakeholders* amb independència de si participen o no de la interacció.

Com veurem més endavant, és molt útil classificar aquests objectius segons com siguin de generals o de concrets, de manera que puguem fer una descripció coherent del sistema en què no barregem objectius generals amb objectius específics.

També caldrà documentar quin és l'àmbit considerat en descriure el sistema. Com també veurem més endavant, l'àmbit ens diu què queda fora i què queda dins del sistema que estem prenent en consideració.

5.3.4. Precondicions i garanties mínimes

Les precondicions del cas d'ús ens indiquen quines condicions s'han de donar per tal que es pugui dur a terme la interacció descrita. A diferència de les condicions d'error (que sí que tenim en compte com a extensions) els casos en què no es compleixin aquestes condicions no els tindrem en compte i no formaran part de la descripció del cas d'ús.

A la descripció de l'exemple també hem afegit les garanties mínimes (allò que el sistema ha de garantir en qualsevol dels escenaris possibles) i les garanties mínimes en cas d'èxit (allò que el sistema ha de garantir per tal de considerar amb èxit la interacció).

5.3.5. Escenaris

També hem definit un escenari principal i unes extensions. Com podem veure, un cas d'ús pot acabar de diverses maneres. Cadascuna d'aquestes possibilitats és un escenari. Així doncs, hi ha un escenari (el principal) que és la seqüència d'esdeveniments que espera l'actor principal quan engega l'execució

del cas d'ús. L'escenari principal ha de ser un escenari d'èxit (on es compleixin les garanties mínimes d'èxit) on l'actor principal satisfà l'objectiu pel qual ha iniciat la interacció amb el sistema.

La resta d'escenaris (siguin d'èxit o d'error) formen el conjunt d'escenaris alternatius o extensions: són escenaris que es poden donar, però que no són l'escenari principal. Una característica important de les extensions és que, en descriure-les, sempre ho fem en referència a la seqüència d'esdeveniments de l'escenari principal: l'extensió comença perquè, en algun pas de l'escenari principal, es dóna una certa condició que dóna pas a l'execució de l'extensió.

Tots els escenaris els descrivim com una seqüència d'accions que poden descriure:

- **Interaccions entre l'usuari i el sistema**: què fa l'usuari en aquest punt de la interacció ("L'usuari indica el tema i el contingut del missatge") o quina resposta dóna el sistema ("El sistema mostra les pràctiques per entregar aquest mes").

- **Validacions per part del sistema**: "El sistema valida que l'usuari tingui accés al fòrum".

- **Canvis a l'estat intern del sistema**: "El sistema enregistra que l'usuari ha llegit el missatge".

Les frases que escrivim per a descriure les accions han de ser senzilles i mostrar clarament quina és l'acció que porta a terme l'usuari o el sistema. Ha d'estar clar qui ha de dur a terme l'acció descrita (l'usuari o el sistema) i, habitualment, ignoren la tecnologia per tal de no limitar les possibilitats d'implementació.

A més, convé mostrar només aquelles accions rellevants que fan avançar el procés i descriure-les en forma d'objectius i intencions i no pas de "moviments".

Exemple

Així, per exemple, en lloc de fer servir 4 passos per a descriure que "El sistema demana el nom", "l'usuari introdueix el nom", "el sistema demana el cognom", "l'usuari introdueix el cognom", preferirem un únic pas "l'usuari introdueix el nom i el cognom", ja que aquesta frase recull el fet essencial que l'usuari proporciona nom i cognom.

5.4. Classificació de casos d'ús

Podem classificar els casos d'ús segons dos grans eixos: el nivell en què es descriuen els seus objectius i l'àmbit que es considera en descriure el cas d'ús.

5.4.1. Nivell dels objectius

Per a poder comparar els casos d'ús entre si ens hem d'assegurar primer que siguin efectivament comparables. Per exemple, no té sentit comparar el cas d'ús "Resoldre un dubte al fòrum" que inclou que l'alumne enviï un missatge amb el dubte, rebre respostes dels companys, potser una resposta del professor, etc.) amb el cas d'ús "Lliurar la solució d'un exercici". Tots dos casos d'ús tenen un nivell de granularitat diferent: mentre que el primer és molt general, el segon és molt més específic.

Cockburn (2001) ens proposa una primera classificació en tres nivells:

- **Usuari** (*user goals*): són els més importants, ja que són els objectius concrets que els actors principal volen aconseguir en fer servir el sistema (per exemple, "Escriure un missatge al fòrum" o "Lliurar una pràctica").

- **General** (*summary goals*): són el nivell més general i ens serveixen per a donar context i agrupar els objectius d'usuari (per exemple, "Resoldre un dubte al fòrum" o "Cursar una assignatura").

- **Tasca** (*subfunctions*): són el nivell més concret i només ofereixen una part del valor que l'actor espera (per exemple, "Adjuntar un arxiu a un missatge" o "Posar una qualificació d'una entrega a un alumne").

Per a classificar un objectiu dins del seu nivell podem observar, per exemple, que els casos d'ús més generals acostumen a respondre la pregunta *per què*, mentre que els més específics estan a prop del *com*.

Quan descrivim un cas d'ús haurem de documentar clarament quin és el nivell de l'objectiu que hem pres en consideració. Si ens trobem que ens cal concretar més, en un moment donat, podem descompondre un cas d'ús en casos d'ús més concrets.

Cas d'ús: resoldre un dubte al fòrum
Actor principal: l'estudiant amb el dubte
Actors de suport: altres estudiants, professor
Nivell: general
Àmbit: campus virtual
Escenari principal d'èxit:
1. L'estudiant amb el dubte escriu un nou missatge al fòrum.
2. Altres estudiants del grup llegeixen el missatge i les respostes que hi pugui haver publicades.
3. Alguns d'aquests estudiants escriuen respostes al missatge original o bé escriuen respostes a les respostes.
4. El professor de l'aula llegeix el missatge original i les respostes.
5. El professor de l'aula decideix que cal intervenir amb una resposta "oficial" i escriu una resposta al missatge original o a alguna de les respostes amb la seva resposta oficial.
6. L'estudiant amb el dubte i la resta d'estudiants del grup llegeixen el missatge amb la resposta "oficial" del professor.
Escenaris alternatius:
5a. El professor decideix que no cal intervenir amb una resposta oficial i s'acaba el cas d'ús.
6b. Algun estudiant decideix que la resposta oficial no és satisfactòria.
    6b1. L'estudiant escriu un missatge de resposta a la resposta del professor demanant més indicacions i tornem al pas 2.

Si necessitem concretar més, podem veure que aquest cas d'ús inclou diversos passos que podem considerar, cada un, un cas d'ús a escala d'usuari:

- Escriure un missatge al fòrum.
- Llegir un missatge o resposta del fòrum.
- Escriure una resposta a un missatge del fòrum.

El cas d'ús "Llegir un missatge del fòrum" ens és familiar, ja que l'hem descrit anteriorment. En aquest cas, veiem que el seu nivell és d'usuari, ja que compleix un objectiu concret d'un usuari: llegir un missatge del fòrum. Ara, però, podem veure un dels possibles contextos en què pot tenir lloc aquest cas d'ús: com a part del cas d'ús general de comentar un dubte al fòrum.

Pel que fa als casos d'ús de tasca, cadascun dels passos dels escenaris descrits al cas d'ús "Llegir un missatge al fòrum" el podríem considerar un cas d'ús de tasca i especificar-lo individualment, tot i que no és habitual documentar-los.

5.4.2. Àmbit

Quan descrivim un cas d'ús és important tenir clar quin és l'àmbit de la descripció que estem fent: què queda dins el sistema que estem descrivint, com a element documentat, i què queda fora, com a element extern.

Podem identificar tres grans tipus d'àmbit:

- **Àmbit d'organització**: estem modelitzant com a cas d'ús el comportament de l'organització com un tot o d'una de les seves unitats, com ara un departament. Tots els sistemes (informàtics o no) i les persones de dins l'organització que defineix l'àmbit formen part del sistema que documen-

tem i, per tant, no apareixen com a actors. Però, en canvi, altres persones, organitzacions o sistemes amb qui interactua l'organització analitzada seran actors.

- **Àmbit de sistema**: estem modelitzant un sistema informàtic (típicament el que volem desenvolupar). Tots els components (de programari o maquinari) del sistema són interns i, per tant, no apareixen com a actors.

- **Àmbit de subsistema**: estem modelitzant una part del sistema informàtic, com ara un component. Els altres components i tots els usuaris directes del component seran actors en el nostre model. Aquest àmbit només és rellevant quan volem deixar explícitament fora d'estudi una part del sistema.

Tots els casos d'ús que hem vist fins ara tenien àmbit de sistema. Vegem un exemple d'un cas d'ús amb àmbit d'organització.

Cas d'ús: matrícula
Actor principal: estudiant
Nivell: general
Àmbit: universitat (organització)
Escenari principal d'èxit:
1. L'estudiant tria les assignatures de les quals es vol matricular i el grup de cada assignatura.
2. La universitat confirma que la selecció és correcta i indica el preu de matrícula.
3. L'estudiant fa el pagament de la matrícula.
4. La universitat enregistra la matrícula de l'estudiant i dóna confirmació a l'estudiant.
Escenaris alternatius:
2a. La selecció d'assignatures no és correcta (perquè l'estudiant no compleix els requisits necessaris).
    2a1. L'estudiant refà la seva selecció i tornem al pas 2.
4b. El pagament no es fa dins el límit establert per la universitat.
    4b1. La matrícula s'enregistra com a fallida i es dóna a l'estudiant una moratòria de 30 dies.
    4b2. Si l'estudiant fa el pagament correctament dins el nou límit, anem al pas 4. En cas contrari, la matrícula queda anul·lada.

Aquest cas d'ús té com a àmbit la universitat, que inclou el personal que hi treballa i tots els sistemes informàtics dels quals disposa. Així, per exemple, el pas 2 pot ser un procés automàtic fet per un sistema informàtic, però també podria incloure intervenció per part del personal de la universitat en el cas, per exemple, que un comitè prengui decisions sobre acceptació de matrícula en casos especials.

Normalment trobarem certa relació entre l'àmbit d'un cas d'ús i el nivell de l'objectiu del seu actor principal: quan descrivim casos d'ús amb àmbit d'organització, tendim a fer servir objectius de nivell més general, mentre que els casos d'ús amb àmbits més concrets tendeixen a descriure objectius de més baix nivell.

5.5. Identificació i descripció de casos d'ús

Una de les tasques més complicades en la creació del model de casos d'ús és la identificació mateixa dels casos d'ús. En aquest apartat veurem un enfocament senzill basat en la identificació dels actors principals, els seus objectius i la classificació d'aquests objectius en funció del seu nivell de generalitat.

5.5.1. Identificació d'actors i objectius

Anomenem *actor iniciador d'un cas d'ús* l'actor que, en un moment donat, porta a terme una acció (envia un missatge, prem un botó, selecciona una opció d'un menú) que provoca l'execució del cas d'ús.

Tot i que habitualment és l'actor principal qui engega el cas d'ús (l'actor principal i l'iniciador són el mateix), hi ha alguns casos en què no és així, com ara quan l'usuari del sistema actua en nom d'algú altre o quan l'escenari s'engega com a resultat d'un esdeveniment temporal.

En el cas que l'usuari actuï en nom d'una altra persona com, per exemple, en el cas de l'operador del centre telefònic, hem de tenir en compte que, mentre que l'usuari del sistema és un *stakeholder* vàlid (òbviament, tindrà els seus interessos sobre el sistema), l'actor principal és la persona que ha trucat per telèfon, ja que és la persona que vol aconseguir un objectiu.

En el cas dels casos d'ús que s'executen de manera automatitzada com a resposta a un esdeveniment temporal, s'acostuma a fer servir un actor virtual anomenat *Rellotge*, tot i que, òbviament, el rellotge no té cap interès a executar el cas d'ús. En aquest cas, cal identificar l'actor principal, que serà aquell *stakeholder* que està interessat que el cas d'ús s'executi.

Un cop identificats els actors principals (siguin o no usuaris directes del sistema) podem enumerar, per a cada un d'ells, quins són els objectius que volen obtenir del sistema. Per cadascun dels objectius que trobem, caldrà tenir un cas d'ús que descrigui la interacció de l'actor principal amb el sistema per tal de satisfer l'objectiu.

5.5.2. Escenaris alternatius i extensions

Tot i que l'estudi de les extensions d'un cas d'ús pot semblar secundari (al cap i a la fi, el que és important és veure com l'actor principal satisfà el seu objectiu), és força habitual descobrir nous casos d'ús i nous requisits mentre s'estan explorant les possibles extensions d'un cas d'ús.

Exemple

Suposem que estem identificant casos d'ús per a un sistema de vendes i estem explorant el cas d'ús "Comprar productes". En arribar al pagament pot passar que el client pagui en efectiu, amb targeta, amb transferència, pot ser que iniciï el pagament amb transferència però la transferència no arribi mai (en aquest cas necessitem un cas d'ús per a detectar

aquestes comandes i tractar-les adequadament) o que arribi la transferència i no quadri l'import (necessitem un cas d'ús per a notificar al client del problema), i així amb infinitat d'escenaris que, potser, en pensar en el cas d'ús "Comprar productes" no ens haurien vingut al cap.

5.5.3. Relacions entre casos d'ús: inclusió

De vegades ens trobem que quan documentem els escenaris d'un cas d'ús ens pot resultar útil fer referència a un altre cas d'ús. Quan això passa es crea una dependència d'inclusió del primer cas d'ús cap al cas d'ús que aquest menciona. Es tracta d'una dependència perquè per a entendre el primer cas d'ús caldrà llegir també la documentació dels casos d'ús que aquest inclou.

Exemple

Fer entrega d'una pràctica (nivell usuari)

Un estudiant vol fer entrega d'una pràctica que ha fet. L'estudiant indica l'assignatura i la pràctica i adjunta l'arxiu amb la solució.

Adjuntar un arxiu (nivell tasca)

L'usuari selecciona l'arxiu del seu ordinador i indica un nom (per defecte el nom local). L'arxiu s'envia al sistema, que desa l'arxiu, el nom indicat, qui l'ha pujat i la data.

Ús del subratllat

És habitual l'ús del subratllat per a indicar, en l'escenari d'un cas d'ús, quines accions es corresponen a casos d'ús inclosos.

En aquest exemple diem que el cas d'ús "Fer entrega d'una pràctica" inclou el cas d'ús "Adjuntar un arxiu", ja que en l'escenari del primer indiquem que es fa servir el segon cas d'ús.

Inclusió per reutilització

L'escenari més típic d'inclusió és aquell en què trobem una sèrie de passos que són comuns a més d'un cas d'ús. Per exemple, el cas d'ús "Escriure un nou missatge al fòrum" i el cas d'ús "Escriure una resposta a un missatge" comparteixen la part de redacció del missatge on l'usuari indica el tema i el contingut del missatge i, opcionalment, adjunta un o més arxius.

En aquests casos, podem extreure la part comuna en un nou cas d'ús i establir una relació d'inclusió entre aquests: tots dos casos d'ús de l'exemple inclourien el cas d'ús "Redactar missatge".

En aquest escenari, quan un cas d'ús sigui inclòs en un altre, ho serà, com a mínim, en un tercer (així, al nostre exemple, "Redactar missatge" és inclòs per "Escriure un nou missatge al fòrum" i "Escriure una resposta a un missatge").

Inclusió per descomposició

Com hem dit, de vegades voldrem descompondre un cas d'ús d'objectiu general en casos d'ús d'objectius més concrets per a afegir detall i concreció a la documentació. En aquest cas, el cas d'ús més general inclourà els casos d'ús més concrets que haurem documentat.

Aquest ha estat l'ús que s'ha donat a la inclusió a l'exemple del subapartat 5.4.1, tot i que, allà, encara no s'havia explicat el significat dels subratllats.

Separació d'una extensió en un cas d'ús inclòs

Un altre cas típic d'ús d'inclusió es produeix quan tenim un cas d'ús amb una extensió molt llarga. Sovint les extensions d'un cas d'ús impliquen variar un parell de passos i tornar a l'escenari principal, però de vegades ens trobem amb extensions que impliquen una sèrie llarga de passos o bé que poden tenir extensions dins de l'extensió. En aquest cas, es fa difícil copsar tota la variabilitat d'escenaris del cas d'ús original i, per tant, és recomanable separar l'extensió en un cas d'ús independent.

A l'exemple detallat del subapartat anterior "Llegir un missatge del fòrum", ens trobem, per exemple, amb l'extensió "L'usuari respon al missatge", que és igual de llarga o més que l'escenari principal: aquest és un bon candidat a separar-se com a cas d'ús independent.

Cas d'ús: llegir un missatge del fòrum
Actor principal: usuari
Àmbit: campus virtual
Nivell d'objectiu: usuari
Usuari: vol llegir el missatge.
Professor responsable de l'aula: vol llegir el missatge i saber quins estudiants l'han llegit.
***Stakeholders* i interessos**:
Precondició: l'usuari s'ha d'haver identificat al sistema.
Garanties mínimes: el sistema enregistrarà l'intent de lectura del missatge.
Garanties en cas d'èxit: el sistema mostrarà a l'usuari el contingut del missatge i n'enregistrarà la lectura.
Escenari principal d'èxit:
1. L'usuari indica quin missatge vol llegir.
2. El sistema enregistra l'intent de lectura del missatge per part de l'usuari.
3. El sistema valida que l'usuari tingui accés al fòrum.
4. El sistema mostra el tema i el contingut del missatge.
5. El sistema enregistra que l'usuari ha llegit el missatge.

Extensions:
1a. L'usuari tanca l'ordinador.
1a1. El sistema enregistra la lectura del missatge encara que l'usuari potser no l'ha vist.
5a. L'usuari vol escriure una resposta a un missatge.
L'usuari respon al missatge al fòrum
5b. La base de dades no està disponible (error de servidor).
5b1. El sistema indica a l'usuari que no ha estat possible recuperar el missatge i li demana que ho torni a provar passada una estona.
5c. El missatge té documents adjunts i l'usuari en vol baixar un.
5c1. L'usuari indica quin document adjunt vol baixar.
5c2. El sistema baixa el document adjunt a l'ordinador de l'usuari.
5c3. Si l'usuari vol descarregar més documents, tornem al pas 5c1.

Cas d'ús: respondre un missatge del fòrum.
Actor principal: usuari.
Àmbit: campus virtual.
Nivell d'objectiu: usuari.
***Stakeholders* i interessos**:
Usuari: vol respondre un missatge que ha llegit.
Professor responsable de l'aula: vol fer el seguiment dels debats que es produeixen a la seva aula.
Precondició: l'usuari s'ha d'haver identificat al sistema.
Garanties mínimes: el sistema enregistrarà la resposta al missatge.
Garanties en cas d'èxit: el sistema enregistrarà la resposta al missatge.
Escenari principal d'èxit:
1. L'usuari indica que vol escriure una resposta a un missatge.
2. El sistema demana el tema i el contingut de la resposta, suggerint *RE:<Tema del missatge al qual responem>* com a tema del missatge.
3. L'usuari modifica el tema si vol i introdueix el contingut de la resposta.
4. El sistema enregistra la resposta amb el tema i contingut introduïts, assigna la data actual com a data de publicació del missatge i l'associa al missatge original.

Extensions:
3a. L'usuari introdueix un tema o una resposta buits.
3a1. El sistema indica que no es pot deixar buit el tema ni la resposta i tornem al punt 3.

5.5.4. Relacions entre casos d'ús: extensió

Originalment hi havia una pràctica força estesa consistent a no modificar els documents d'anàlisi un cop finalitzats i, en lloc d'això, escriure documents addicionals que indiquessin els canvis que s'havien produït en els requisits.

Això va propiciar que s'introduís una nova relació entre casos d'ús diferent de la d'inclusió i especialment pensada per a documentar extensions complexes (com la de "Respondre un missatge del fòrum").

La diferència, quan es fa servir una relació d'extensió, és que el cas d'ús que conté l'escenari principal no es modifica: no s'hi menciona l'extensió en cap moment. En lloc d'això, és el cas d'ús que documenta l'extensió el que indica com i on s'afegeix l'extensió al cas d'ús original.

Suposem, per exemple, que no podem modificar els documents d'anàlisi i que, en descriure el cas d'ús "Llegir un missatge del fòrum", no hem tingut en compte l'opció de respondre. El document de partida seria el següent.

Cas d'ús: llegir un missatge del fòrum
Actor principal: usuari
Àmbit: campus virtual
Nivell d'objectiu: usuari
Usuari: vol llegir el missatge.
Professor responsable de l'aula: vol llegir el missatge i saber quins estudiants l'han llegit.
***Stakeholders* i interessos**:
Precondició: l'usuari s'ha d'haver identificat al sistema.
Garanties mínimes: el sistema enregistrarà l'intent de lectura del missatge.
Garanties en cas d'èxit: el sistema mostrarà a l'usuari el contingut del missatge i n'enregistrarà la lectura.
Escenari principal d'èxit:
1. L'usuari indica quin missatge vol llegir.
2. El sistema enregistra l'intent de lectura del missatge per part de l'usuari.
3. El sistema valida que l'usuari tingui accés al fòrum.
4. El sistema mostra el tema i el contingut del missatge.
5. El sistema enregistra que l'usuari ha llegit el missatge.

Extensions:
1a. L'usuari tanca l'ordinador.
1a1. El sistema enregistra la lectura del missatge encara que l'usuari potser no l'ha vist.
5a1. El sistema indica a l'usuari que no ha estat possible recuperar el missatge i li demana que ho torni a provar passada una estona.
5b. El missatge té documents adjunts.
5b1. L'usuari indica quin document adjunt vol baixar.
5b2. El sistema baixa el document adjunt a l'ordinador de l'usuari.

En aquest exemple, podríem afegir l'opció de respondre mitjançant la relació d'extensió i, per tant, sense modificar el cas d'ús original.

Cas d'ús: respondre un missatge del fòrum.
Estén: el cas d'ús "Llegir un missatge del fòrum" al punt 5
Actor principal: usuari.
Àmbit: campus virtual.
Nivell d'objectiu: usuari.
***Stakeholders* i interessos**:
Usuari: vol llegir el missatge.
Professor responsable de l'aula: vol llegir el missatge i saber quins estudiants l'han llegit.
Precondició: l'usuari s'ha d'haver identificat al sistema.
Garanties mínimes: el sistema enregistrarà l'intent de lectura del missatge.
Garanties en cas d'èxit: el sistema mostrarà a l'usuari el contingut del missatge i n'enregistrarà la lectura.
Escenari principal d'èxit:
1. L'usuari indica que vol escriure una resposta a un missatge.
2. El sistema demana el tema i el contingut de la resposta, suggerint *RE:<Tema del missatge al qual responem>* com a tema del missatge.
3. L'usuari modifica el tema si vol i introdueix el contingut de la resposta.
4. El sistema enregistra la resposta amb el tema i contingut introduïts, assigna la data actual com a data de publicació del missatge i l'associa al missatge original.

Extensions:
3a. L'usuari introdueix un tema o una resposta buits.
3a1. El sistema indica que no es pot deixar buit el tema ni la resposta i tornem al punt 3.

Quan un cas d'ús (com ara "Respondre un missatge del fòrum") n'estén un altre ("Llegir un missatge del fòrum"), hi ha una dependència del primer cap al segon, ja que el segon no menciona l'extensió per enlloc i, per tant, no en depèn.

A diferència de la inclusió, és molt estrany que un cas d'ús sigui extensió de dos casos d'ús diferents: seria molta casualitat que tinguéssim dos casos d'ús de partida amb la mateixa variació.

Quina relació entre casos d'ús és, doncs, més adequada per a una situació donada? Segons Alistair Cockburn: "Com a regla general, si un cas d'ús *A* menciona un cas d'ús *B*, aleshores *A* inclou *B*. Gairebé sempre és més fàcil i més clar fer servir inclusió".

5.6. Casos especials

5.6.1. Autenticació d'usuaris

El de l'autenticació d'usuaris és un cas que genera una certa controvèrsia. Per una banda, es pot considerar un requisit funcional i, per tant, documentar el comportament del sistema durant l'autenticació com un cas d'ús. Per altra banda, es pot considerar un requisit no funcional, com un detall necessari per a aconseguir la seguretat del sistema amb independència de la funcionalitat del sistema.

Una cosa que és clara és que identificar-se com a usuari no és un objectiu de nivell d'usuari en el sentit que quan algú s'identifica al sistema ho fa amb un altre objectiu (comprar un producte, lliurar un exercici, etc.). Per tant, queda clar que, en cas de ser un cas d'ús, l'autenticació d'usuaris seria de nivell tasca i no pas de nivell usuari.

Tenint en compte això, en el moment de documentar la necessitat d'autenticació, hem de decidir si ho fem com a precondició o com a un pas més de l'escenari.

Triarem la primera opció (precondició) quan no volem que sigui possible iniciar el cas d'ús sense estar identificat (per exemple, el cas d'ús "Lliurar un exercici" requereix que l'usuari estigui identificat abans de començar el cas d'ús).

En canvi, triarem la segona opció (pas de l'escenari) quan vulguem indicar, dins de la seqüència d'esdeveniments de l'escenari, quina part es pot dur a terme sense estar identificat i en quin moment és necessari que l'usuari estigui identificat. Per exemple, és molt típic que les aplicacions de comerç electrònic ens deixin fer comandes sense estar identificats al sistema i només ens obliguin a identificar-nos en el moment de fer el pagament.

5.6.2. Alta d'usuari

Un altre cas especial relacionat amb la identificació d'usuaris és el registre o alta d'usuaris. A diferència de la identificació d'usuaris, l'alta pot ser un objectiu en si mateix (té sentit pensar que algú accedeix al nostre sistema només per donar-se d'alta i, un cop s'hagi donat d'alta, ja no faci res més fins un altre dia).

En els sistemes en què els usuaris es donen d'alta ells mateixos és habitual que aquest cas d'ús es pugui executar de manera independent o, per facilitar les coses, com a extensió del cas d'ús d'identificació. Això se'ns fa una mica estrany, ja que tenim un cas d'ús de nivell usuari que és una extensió d'un cas d'ús de nivell tasca. Doncs bé, haurem de conviure amb aquest fet.

Com a conclusió, podem dir que aquest cas d'ús està en un nivell intermedi entre el nivell usuari i el nivell tasca, de manera que el podem documentar com a cas d'ús independent de nivell usuari o com a cas d'ús de nivell tasca (sigui independent o sigui com a extensió del cas d'ús d'identificació) dins d'un altre cas d'ús de nivell usuari.

5.6.3. Manteniments (CRUD)

Quan identifiquem els casos d'ús d'un sistema ens solem trobar amb un munt de casos d'ús del tipus "Crear un *X*", "Consultar un *X*", "Actualitzar un *X*" i "Esborrar un *X*". Aquests són el que anomenem *manteniments* o casos d'ús *CRUD*[5] (crear-recuperar-actualitzar-esborrar).

(5)De l'anglès, *create-read-update-delete*.

Segons el que hem vist fins ara, cada un d'aquests petits casos d'ús és un cas d'ús de nivell d'usuari, ja que el seu actor principal té un objectiu clar que espera assolir mitjançant aquell cas d'ús.

Ara bé, identificar i documentar cada manteniment com un conjunt de 4 o 5 casos d'ús a escala d'usuari pot ser, moltes vegades, una tasca repetitiva, sense valor i propensa a errors, ja que tots els casos d'ús de creació s'assemblaran molt entre si, tots els de consulta també, etc.

Una opció totalment vàlida, per tant, és crear, en aquests casos, un únic cas d'ús de nivell general que aglutini els diversos objectius d'usuari del manteniment en un únic cas d'ús. Això té l'avantatge de simplificar la feina i evitar la repetició innecessària.

Cas d'ús: gestionar campus, edificis i aules
Actor principal: un administrador del sistema
Nivell: general
Escenari principal d'èxit:
1. L'usuari selecciona una consulta de campus, edificis i aules.
2. El sistema mostra una llista dels campus del sistema.
3. L'usuari selecciona un campus.
4. El sistema mostra una llista dels edificis del campus.
5. L'usuari selecciona un edifici.
6. El sistema mostra una llista de les aules de l'edifici.
7. L'usuari torna al pas 1, 3 o 5 fins que queda satisfet (i el cas d'ús s'acaba).

Escenaris alternatius:
a. Crear una nova aula
   1. L'usuari indica que vol crear una nova aula a l'edifici seleccionat.
   2. L'usuari indica el nom de l'aula.
   3. El sistema enregistra la nova aula i torna al punt 6.
b. Esborrar una aula
   1. L'usuari indica que vol esborrar una aula.
   2. El sistema verifica que l'aula no està en ús al calendari del semestre actual.
   3. El sistema marca l'aula com a eliminada (tot i que continuarà apareixent als horaris dels semestres passats) i torna al punt 6.
c. Canviar el nom d'una aula
   1. L'usuari indica que vol canviar el nom d'una aula.
   3. L'usuari indica el nou nom de l'aula.
   4. El sistema enregistra el canvi de nom i torna al punt 6.
d. Crear un nou edifici
   1. L'usuari indica que vol crear un nou edifici al campus seleccionat.
   3. L'usuari indica el nom i adreça de l'edifici.
   4. El sistema enregistra el nou edifici i torna al pas 4.

Com a inconvenient, agrupar els manteniments en casos d'ús de nivell general impedeix detallar informació concreta de cada objectiu de més baix nivell. Així, per exemple, si volem documentar els permisos que cada actor té sobre cada operació, ens caldrà fer-ho en un document a part.

Un enfocament proposat per Cockburn (2001) consisteix a començar fent servir casos d'ús de manteniment de nivell general i, només si descobrim que ens cal donar més detalls o identificar de manera individual alguna part del manteniment, extreure casos d'ús de més baix nivell i documentar-los a part. En tal cas, podem fer servir una relació d'inclusió per a indicar, en el cas d'ús més general, en quin cas d'ús més detallat es descriu la part que hem extret.

5.6.4. Casos d'ús parametritzats

Tot sovint ens trobarem que hem d'escriure diversos casos d'ús molt semblants entre si. És el cas, per exemple, de molts manteniments ja mencionats i d'altres casos d'ús en què identifiquem un patró comú que després s'aplica a diversos casos d'ús.

El cas d'ús d'exemple de manteniment, al subapartat anterior, és prou específic, ja que agrupa els manteniments de campus, edificis i aules. Però, probablement, molts altres manteniments seran senzills, de l'estil "Gestió de persones", "Gestió d'assignatures", "Gestió de departaments", etc.

Tots aquests casos d'ús, probablement, seguiran una mateixa estructura. Posem per cas que són del tipus següent.

Cas d'ús: gestionar assignatures
Actor principal: un administrador del sistema
Nivell: general
Escenari principal d'èxit:
1. L'usuari demana fer una cerca d'assignatures.
2. L'usuari introdueix el nom o fragment de nom de l'assignatura.
3. El sistema mostra una llista de les assignatures coincidents (totes, si no s'ha introduït cap nom d'assignatura).
4. Opcionalment, l'usuari canvia el criteri d'ordenació (nom o bé nombre de crèdits) i tornem a pas 3.
5. Els passos 2 a 4 es repeteixen fins que l'usuari queda satisfet (i el cas d'ús s'acaba).

Escenaris alternatius:
a. Crear una nova assignatura
   1. L'usuari indica que vol crear una nova assignatura.
   2. L'usuari indica nom i nombre de crèdits de l'assignatura.
   3. El sistema enregistra la nova assignatura i torna al punt 3.
b. Esborrar una assignatura
   1. L'usuari indica que vol esborrar una assignatura.
   2. El sistema verifica que l'assignatura no s'imparteix al semestre actual.
   3. El sistema marca l'assignatura com a eliminada (tot i que continuarà apareixent als horaris dels semestres passats) i torna al punt 3.
c. Actualitzar una assignatura
   1. L'usuari indica que vol actualitzar una assignatura.
   2. L'usuari indica el nom o nombre de crèdits.
   3. El sistema enregistra el canvi de nom o de nombre de crèdits i torna al punt 3.

En aquest cas d'ús podem observar que, probablement, tots els manteniments auxiliars seran molt semblants en estructura, tot i que en cada un variarà:

- De què fem el manteniment (en aquest cas, d'assignatures).
- Quines són les dades de què cal informar en crear una nova instància (en aquest cas, nom i nombre de crèdits).
- Quines dades són modificables (en aquest cas, nom i número de crèdits).
- Quines dades es mostren a la llista (en aquest cas, de nou, nom i número de crèdits).
- Quines dades es poden fer servir per a filtrar la cerca (en aquest cas, només el nom de l'assignatura).
- Quines dades es poden fer servir per a reordenar la cerca (en aquest cas, nom i nombre de crèdits).

- Quines verificacions cal fer en esborrar una entitat (en aquest cas, que l'assignatura no s'imparteixi el semestre actual).

Escriure cada un d'aquests casos d'ús per separat no és només tediós sinó propens a error, ja que el comportament comú d'aquests casos d'ús voldrem, probablement, que sigui totalment consistent al llarg del sistema, i una errada en un dels casos d'ús podria trencar aquesta consistència. D'altra banda, tenir informació duplicada també és un inconvenient a l'hora de fer manteniment dels casos d'ús, ja que si volem canviar l'estructura general dels casos d'ús ens caldrà canviar molts casos d'ús diferents.

Per a resoldre aquests problemes podem escriure una mena de plantilla, un cas d'ús parametritzat. Per a cada element de la plantilla que pot variar, introduïm un paràmetre.

Cas d'ús: gestionar entitats
Nivell: general
Escenari principal d'èxit:
1. L'usuari demana fer una cerca d'entitats.
2. Opcionalment, l'usuari introdueix un o més camps de cerca.
3. El sistema mostra una llista de les entitats coincidents (totes, si no s'ha introduït cap camp de cerca).
4. Opcionalment, l'usuari canvia el criteri d'ordenació (indica algun camp d'ordenació) i es torna al pas 3.
5. Els passos 2 a 4 es repeteixen fins que l'usuari queda satisfet (i el cas d'ús s'acaba).

Escenaris alternatius:
a. Crear una nova entitat
   1. L'usuari indica que vol crear una nova entitat.
   2. L'usuari indica camps de creació de l'entitat.
   3. El sistema enregistra la nova entitat i torna al punt 3.
b. Esborrar una entitat
   1. L'usuari indica que vol esborrar una entitat.
   2. El sistema verifica que que l'entitat sigui esborrable.
   3. El sistema marca l'entitat com a eliminada i torna al punt 3.

Un cop tenim un cas d'ús plantilla, podem definir casos d'ús fent servir la plantilla. Per a això cal indicar quina plantilla es fa servir (utilitzant una relació d'inclusió) i quins valors prenen els paràmetres de la plantilla.

Cas d'ús: gestionar assignatures
Actor principal: un administrador del sistema
Nivell: general
Escenari principal d'èxit:
1. L'usuari gestiona assignatures fent servir el cas d'ús Gestionar entitats, en què:
   - L'entitat és *assignatura*.
   - Els camps de cerca són el nom de l'assignatura.
   - Els camps de creació, actualitzables i d'ordenació són el nom i el nombre de crèdits.
   - L'entitat és esborrable si l'assignatura no s'imparteix el semestre actual.

5.6.5. Modelització basada en casos d'ús de processos de negoci

L'ús més típic de modelització basada en casos d'ús és el de documentar els requisits d'un sistema informàtic desenvolupat per a una organització. En aquest escenari, la majoria de casos d'ús es descriuran amb àmbit de sistema i gairebé tots amb l'objectiu de nivell usuari.

Un ús dels casos d'ús ben diferent d'aquest és la modelització de processos de negoci. Es tracta de modelitzar mitjançant casos d'ús un sistema que, en lloc de ser un sistema informàtic, és una organització que inclou sistemes informàtics, maquinari, persones, etc. Per tant, per definició, els casos d'ús tindran àmbit d'organització.

A partir d'aquí, podem veure que els actors primaris seran aquelles persones o organitzacions que demanen serveis a l'organització per tal d'aconseguir certs objectius, i cada cas d'ús serà un servei que l'organització ofereix a un d'aquests actors primaris per a satisfer un dels seus objectius.

Un cop fet un model de processos de negoci basat en casos d'ús, si, a més, volem analitzar un dels sistemes informàtics que formen part d'aquesta organització, també podem fer servir casos d'ús. La qüestió, aleshores, és si hi ha alguna relació entre els casos d'ús amb àmbit d'organització i els casos d'ús amb àmbit del sistema informàtic.

La connexió no és, en tot cas, trivial. Tot i que podem partir dels casos d'ús de l'organització per a identificar casos d'ús del sistema, normalment no tindrem una connexió clara entre uns i altres per al 100% dels casos.

Cal tenir en compte, doncs, que si fem modelització de processos de negoci i d'un dels sistemes informàtics de l'organització, tots dos basats en casos d'ús, caldrà tenir molt clar a quin dels dos documents pertany cada cas d'ús i no barrejar casos d'ús d'un amb els de l'altre.

Resum

En aquest mòdul ens hem centrat en els requisits del programari, que hem definit com a característiques observables del sistema per desenvolupar que satisfan necessitats o expressen restriccions d'aquelles persones o entitats que tenen algun impacte o interès en el projecte, que anomenem *stakeholders*. Els requisits es poden classificar com a funcionals, que fan referència a les necessitats que ha de satisfer el sistema, o com a no funcionals, que són aquells que expressen restriccions sobre el conjunt possible de solucions.

Per a desenvolupar un projecte caldrà identificar quins són els *stakeholders* del nostre sistema i obtenir-ne els requisits. Tècniques com el *brainstorming*, les entrevistes i qüestionaris, l'observació, el prototipatge i les llistes predefinides ens ajuden a fer un recull de requisits de més qualitat.

Al llarg del projecte també caldrà gestionar aquests requisits per a decidir quins requisits es tindran en compte, per a la qual cosa caldrà estimar-ne el cost, prioritzar-los i fer-ne una tria per tal de maximitzar el valor obtingut per al desenvolupament tenint en compte els recursos disponibles.

Els requisits recollits al llarg del projecte es documenten en el que anomenem *especificació*, que actua com a contracte entre els desenvolupadors i els *stakeholders* i com a eina de comunicació entre ells. Aquest document és important que tingui una gran qualitat, ja que d'això depèn la qualitat del programari que obtindrem: hem d'aconseguir mantenir uns bons criteris de qualitat, com els mencionats per l'IEEE, per a la qual cosa convé seguir unes bones pràctiques.

En aquest mòdul hem vist dos estils de documentació de requisits. Les històries d'usuari, més adequades a desenvolupaments àgils, confien molt en la comunicació oral entre desenvolupadors i *stakeholders* i només documenten un títol i unes proves d'acceptació. Els casos d'ús, en canvi, documenten de manera textual la interacció entre un actor i el sistema per a obtenir un cert objectiu.

Activitats

1. Suposeu que tenim un cas d'ús "Corregir una activitat" amb l'especificació textual breu següent:

Corregir una activitat (nivell global)

El professor baixa del campus virtual tots els documents d'entrega de l'activitat entregats fins a la data i el sistema enregistra com a consultades totes les entregues disponibles fins a la data. El professor corregeix les entregues (fora del sistema). El professor introdueix les qualificacions al sistema, que comprova que només introdueixi qualificacions a entregues que prèviament hagi consultat.

Feu l'especificació textual detallada, fent servir la plantilla detallada del subapartat 5.1. Creeu casos d'ús de nivell d'usuari i incloeu-los en aquest per a aquelles parts en què ho cregueu oportú. Feu també l'especificació detallada d'aquests nous casos d'ús.

2. Indiqueu si els requisits següents estan expressats correctament. En el cas que no ho estiguin, reescriviu-los correctament (vegeu el subapartat 4.2):

a) El sistema ha de funcionar sobre plataforma Linux i Windows.
b) Farem servir un índex amb els camps *nom* i *descripció del producte Apache Lucene* per tal que els usuaris puguin cercar els productes amb qualsevol paraula clau.
c) La web ha de funcionar sobre els navegadors més usats.
d) Farem servir el bastiment .Net Entity Framework per a accedir a la base de dades.
e) Els usuaris podran omplir la sol·licitud fàcilment.

3. Suposem que estem desenvolupant un sistema de venda d'entrades per a teatres per mitjà d'Internet. Dieu si les entitats i persones següents són o no *stakeholders*. En el cas que també siguin actors del sistema, indiqueu-ho. Justifiqueu la vostra resposta.

a) La persona que vol comprar l'entrada.
b) La persona que atén la centraleta telefònica d'atenció a l'usuari.
c) Els propietaris dels teatres.
d) Els actors de les obres.
e) L'acomodador del teatre.
f) Les persones que van al teatre però que no han comprat directament l'entrada.
g) El personal de la taquilla del teatre.
h) Els administradors de sistemes de l'empresa on estan hostatjats els servidors.
i) Els desenvolupadors de l'aplicació.
j) El comercial que ha de convèncer els teatres que entrin a la xarxa i posin les seves entrades a la venda per mitjà de l'aplicació.
k) El programari que fan servir els teatres per a gestionar les vendes d'entrades a la taquilla.
l) Els desenvolupadors del programari que fan servir els teatres per a gestionar les vendes d'entrades a la taquilla.
m) El cap de projecte.

4. Suposeu que estem desenvolupant un sistema per a la gestió d'ajuts públics a la formació en empreses. Identifiqueu tres rols d'usuaris i dos *stakeholders* potencials que no siguin usuaris i justifiqueu quina és la seva implicació en el projecte.

5. Suposeu que us han demanat que desenvolupeu una aplicació molt senzilla: es tracta de permetre als usuaris (treballadors d'una empresa) publicar anuncis de tipus classificats ("venc el meu cotxe", "busco un ordinador de segona mà", etc.) a la Intranet de l'empresa. No cal que es pugui respondre, només publicar els anuncis (amb títol, text i data de caducitat), i no us poden donar més documentació ni podeu parlar amb ningú més. En aquest cas, podem dir que tenim representant dels *stakeholders*?

6. Suposem ara que, en el desenvolupament del sistema anterior, han decidit que podem enviar un qüestionari als usuaris finals del sistema i ens han donat les preguntes següents. Creieu que són adequades? En cas contrari, indiqueu quin és el problema i com es podrien millorar.

a) Creieu que s'haurien de poder adjuntar fotografies als anuncis?
b) Creieu que s'haurien d'afegir més camps?
c) Trobeu bé que els anuncis caduquin automàticament al cap d'un mes?
d) Un usuari hauria de poder modificar un anunci un cop publicat?

7. Doneu dos exemples per a cadascuna de les categories de requisits de l'IEEE 830 mencionades al subapartat 2.2.3.

8. Suposeu que tres persones estan fent una estimació mitjançant la tècnica del *planning poker* i estimen el requisit en 2, 4 i 8 punts. Quin és el valor que hem de fer servir per a l'estimació?

9. Dels requisits del nostre sistema sabem que:

- El requisit A trigarem 1 dia a implementar-lo.
- El requisit B és ben bé el doble de complicat que A.
- El requisit C és el doble de complicat que B.
- El requisit D és el triple de complicat que B.
- El requisit E és el triple de complicat que C.
- El requisit F és més complicat que C però menys que D.
- El requisit G està entre B i C.
- El requisit H està entre el doble de C i el doble de F.

Quants dies trigaríem a implementar tot el projecte? Què passa si ens hem equivocat, en estimar A, en un 10%?

10. A continuació us presentem un cas d'ús extret de la documentació d'OpenUP. Compareu i comenteu les diferències de la plantilla emprada amb la que hem fet servir al subapartat 5.1. L'exemple complet el podeu trobar a l'EPF Wiki (darrera visita, octubre 2010).

Nom (no apareix com a apartat): treure diners (d'un caixer automàtic)

1. Descripció breu

Aquest cas d'ús descriu com el client del banc fa servir el caixer automàtic per a treure diners del seu compte.

2. Actors

Client del Banc
Banc

3. Precondicions

Hi ha una connexió activa amb el banc.
El caixer té efectiu disponible.

4. Flux bàsic d'esdeveniments

1. El cas d'ús comença quan el Client del Banc insereix la seva targeta.
2. S'executa el cas d'ús "Validar usuari".
3. El Caixer Automàtic mostra les diferents alternatives que el client té disponibles (vegeu el requisit de suport SR-xxx per a la llista d'alternatives). En aquest cas, el Client del Banc selecciona "Treure efectiu".
4. El Caixer Automàtic demana quin compte s'ha de fer servir. Vegeu el requisit de suport SR-yyy per als tipus de compte que s'han de suportar.
5. El Client del Banc selecciona un compte.
6. El Caixer Automàtic demana l'import.
7. El Client del Banc introdueix l'import.
8. S'envia l'ID de targeta, PIN, import i compte al banc com una transacció. El Consorci del Banc respon amb "continuar / no continuar" segons si la transacció ha anat bé o no.
9. Es dispensa l'efectiu.
10. Es retorna la targeta.
11. S'imprimeix el rebut.
12. El cas d'ús finalitza amb èxit.

5. Fluxos alternatius

5.1. Usuari incorrecte

Si al pas 2 del flux bàsic el cas d'ús "Validar Usuari" no es completa amb èxit, llavors,

1. El cas d'ús finalitza amb error.

5.2 Compte equivocat

Si al pas 8 del flux bàsic el compte seleccionat pel Client del Banc no està associat amb aquesta targeta, llavors,

1. El Caixer Automàtic mostrarà el missatge "Compte incorrecte; si us plau, torni-ho a provar".
2. El cas d'ús continua al pas 4.

[...]

5.7 No hi ha resposta del Banc.

Si al pas 8 del flux bàsic no hi ha resposta del Banc en tres segons, llavors,

1. El Caixer Automàtic ho torna a provar, fins a tres vegades.
2. Si no hi ha resposta del Banc, el Caixer Automàtic mostrarà el missatge "Xarxa no disponible; provi de nou més tard".
3. El Caixer Automàtic retornarà la targeta.
4. El Caixer Automàtic indicarà que està "Tancat".
5. El cas d'ús finalitza amb una condició d'error.

[...]

6. Escenaris clau

6.1. No hi ha resposta del Banc.

7. Postcondicions

7.1. Postcondicions d'èxit

L'usuari ha rebut l'efectiu i els registres interns s'han actualitzat.

7.2. Postcondicions d'error

Els registres s'han actualitzat com pertoca.

8. Requisits especials

[SpReq:WC-1] El caixer dispensarà efectius en múltiples de 20$.
[SpReq2:WC-2] L'import màxim en un reintegrament és de 500$.
[SpReq:WC-1] El caixer mantindrà un registre que inclourà data i hora de totes les transaccions amb el banc (siguin completes o incompletes).

11. Identifiqueu un cas d'ús per a un sistema de lloguer públic de bicicletes (com el Bicing de Barcelona) en què l'actor principal no és l'usuari que lloga bicicletes.

12. Imagineu que esteu desenvolupant el programari que gestiona un telèfon mòbil i esteu en un mode de diagnòstic en què surten unes xifres que no sabem què volen dir. Quin cas d'ús pot estar relacionat amb aquesta funcionalitat? Qui en seria l'actor principal?

13. Suposeu que passeu pel carrer i veieu un amic vostre concentrat en el seu telèfon mòbil. El vostre amic us explica que està comprant unes entrades per al teatre i que està buscant el botó de pagar però no el troba. Indiqueu objectius globals, d'usuari i de tasca en aquesta situació.

14. Suposeu que estem desenvolupant una aplicació per a la gestió de preguntes i respostes d'usuaris a l'estil de Yahoo Answers o Stack Exchange. Identifiqueu tres actors, i per a cada un d'ells, identifiqueu com a mínim un objectiu de nivell global, un d'usuari i un de tasca que estiguin relacionats entre si. Descriviu de manera informal el cas d'ús que permeti assolir cadascun dels objectius de nivell usuari o global identificats.

Exercicis d'autoavaluació

1. Quines són les dues condicions més bàsiques perquè una certa característica es pugui considerar un requisit?

2. Quina relació hi ha entre el concepte d'usuari i el de *stakeholder*?

3. Quina és la funció principal dels requisits?

4. Quina diferència hi ha entre els requisits funcionals i els que no ho són?

5. Quines són les principals activitats relacionades amb els requisits que caldrà dur a terme al llarg del projecte?

6. És possible que un sistema hagi de satisfer dos requisits contradictoris?

7. Durant el *brainstorming* inicial s'aporten idees sense entrar en valorar-les, sense evitar repeticions i sense fer cap selecció. Com s'arriba a un resultat útil que no tingui aquests problemes?

8. Indiqueu tres activitats relacionades amb els requisits que es puguin fer mitjançant un *brainstorming*.

9. Quines tècniques ens poden evitar haver de pensar en els requisits de cada un dels usuaris de manera individual?

10. Com en diem de la persona que parla en nom d'un *stakeholder* al qual no tenim accés? Quines problemàtiques ens pot donar?

11. Per què la pregunta "Vols que el sistema retorni tots els resultats paginats de 20 en 20 en menys d'1 segon" pot ser inadequada per a una entrevista d'identificació de requisits?

12. Podem construir el sistema final a partir d'un prototip?

13. Poseu dos exemples d'aspectes que cal tenir en compte pel que fa a la usabilitat i la humanitat, segons Volere.

14. A quin tipus de requisit de la llista Volere pertany cada un d'aquests aspectes?

- Consideracions sobre la imatge corporativa de l'organització
- Satisfacció dels usuaris
- Accessibilitat per part de persones discapacitades
- Temps mitjà entre caigudes del sistema
- Requisits sobre com s'han de programar els lliuraments dels diferents artefactes i versions del sistema
- Sistemes operatius que cal que suporti el sistema
- Unitats que es faran servir per a mostrar longituds, pesos, etc.
- Llenguatges en què es pot mostrar la interfície del sistema

15. Quina informació ens caldrà tenir, de cada requisit candidat, abans de poder seleccionar quins prendrem realment en consideració?

16. Quins dos grans tipus d'unitats podem fer servir per a estimar requisits? Doneu-ne un exemple de cada.

17. Si disposem d'uns quants requisits ja estimats quines tècniques ens poden ajudar a estimar la resta?

18. Per què és important que hi hagi consens quan es fa servir la tècnica d'estimació *planning poker*? Per què no resolem els conflictes per majoria?

19. Quines són les principals dificultats de prioritzar els requisits?

20. Quina tècnica ens pot ajudar a prioritzar els requisits quan tenim múltiples *stakeholders* o *stakeholders* que no tenen clar com cal prioritzar?

21. Quines són les qualitats d'una bona especificació de requisits segons l'IEEE?

22. Quin defecte té una especificació de requisits que es pot entendre de maneres diferents i incompatibles entre si?

23. Quin defecte té una especificació de requisits en què diversos requisits es porten la contrària?

24. Quin defecte té una especificació de requisits si alguns dels requisits són tals que requeriria una gran quantitat de diners i temps per a comprovar si un sistema els satisfà?

25. Quin defecte té una especificació de requisits en la qual es fa difícil introduir canvis?

26. Una història d'usuari és una targeta amb una descripció i un conjunt de proves d'acceptació?

27. Quina mida haurien de tenir les històries d'usuari d'una pila de producte?

28. Quina diferència hi ha entre un actor d'un cas d'ús i un *stakeholder*?

29. Què passa si la precondició d'un cas d'ús no es compleix?

30. Què és l'escenari principal d'un cas d'ús?

31. Quins tres nivells d'objectius de casos d'ús identifica Cockburn?

32. Quin és l'àmbit en què una persona pot formar part del sistema que estem estudiant en lloc d'aparèixer com a actor?

33. Quin és l'àmbit en què un component de programari del sistema que estem estudiant es pot considerar un actor que interactua amb un subsistema del qual escrivim casos d'ús?

34. L'actor iniciador d'un cas d'ús n'ha de ser l'actor principal? En cas contrari, doneu-ne contraexemples.

35. Com podem evitar que dos casos d'ús continguin una bona part de la documentació igual i, per tant, repetida en dos documents?

36. Com podem separar la documentació d'una extensió molt llarga del document del cas d'ús per tal de poder-li definir, al seu torn, extensions?

37. Si disposem de l'especificació textual d'un cas d'ús en un document que no podem modificar (per exemple, perquè la política de l'organització és no modificar documents d'anàlisi un cop publicats) i volem afegir una extensió, com ho podem fer?

38. Indiqueu dues maneres de modelitzar l'autenticació d'usuaris en un model de casos d'ús.

39. Com podem evitar una explosió en el nombre de casos d'ús semblants quan estem documentant manteniments amb casos d'ús de l'estil CRUD?

40. En què consisteix un cas d'ús parametritzat?

41. Quin àmbit més habitual tindran els casos d'ús si fem modelització de processos de negoci?

Solucionari

Activitats

1.

Cas d'ús: corregir una activitat
Actor principal: professor
Àmbit: campus virtual
Nivell d'objectiu: general
***Stakeholders* i interessos**:
Professor: vol corregir els lliuraments de l'activitat.
Estudiants: volen que els corregeixi l'activitat i, per tant, saber la nota.
Precondició:
El professor s'ha d'haver identificat al sistema.
L'estudiant ha d'haver lliurar l'activitat.
Garanties mínimes: el sistema registrarà el fet que el professor ha baixat els documents.
Garanties en cas d'èxit: el sistema registrarà la nota de cada lliurament.
Escenari principal d'èxit:
1. El professor baixa els documents dels lliuraments de l'activitat.
2. El sistema registra el fet que el professor ha baixat els documents.
3. El professor introdueix les qualificacions dels lliuraments.
4. El sistema valida que l'usuari hagi baixat abans els documents.
5. El sistema registra la qualificació de cadascun dels lliuraments.

Extensions:
4a. El professor no ha baixat el document del lliurament.
4a1. El sistema indica al professor que no pot acceptar qualificacions de lliuraments que no han estat baixats.
4a2. Tornen al pas 1.

Cas d'ús: baixar els documents dels lliuraments d'una activitat
Actor principal: professor
Àmbit: campus virtual
Nivell d'objectiu: usuari
***Stakeholders* i interessos**:
Professor: vol baixar els documents per corregir el lliurament.
Precondició:
El professor s'ha d'haver identificat al sistema.
Garanties mínimes: -
Garanties en cas d'èxit:
El professor obté els arxius.
El sistema registrarà el fet que el professor ha baixat els documents.
Escenari principal d'èxit:
1. El professor tria una activitat.
2. El sistema mostra la llista d'estudiants que han fet el lliurament.
3. El professor demana el document d'un estudiant.
4. El sistema envia al professor el document del lliurament de l'estudiant.
5. Si el professor vol baixar més documents, torna al pas 3.

Extensions: -

Cas d'ús: introduir qualificacions de lliuraments
Actor principal: professor
Àmbit: campus virtual
Nivell d'objectiu: usuari
***Stakeholders* i interessos**:
Professor: vol publicar les notes dels lliuraments.
Precondició:
El professor s'ha d'haver identificat al sistema.
Garanties mínimes: -
Garanties en cas d'èxit: el sistema enregistra les notes.
Escenari principal d'èxit:
1. El professor tria una activitat.
2. El sistema mostra la llista d'estudiants que han fet el lliurament.
3. El professor indica un estudiant i una nota.
4. El sistema registra la nota del lliurament.
5. Mentre quedin notes per introduir, tornem al pas 3.

Extensions:
4a. El professor no ha baixat el document del lliurament.
4a1. El sistema indica al professor que no pot acceptar qualificacions de lliuraments que no han estat baixats.
4a2. Tornen al pas 2.

2.

a) "El sistema ha de funcionar sobre plataforma Linux i Windows" és difícil de verificar, ja que no indica quines versions concretes de Linux i Windows s'han de suportar. Una versió correcta seria: "El sistema ha de funcionar sobre plataforma Linux (nucli 2.6 i entorn gràfic Gnome 2.32) i Windows 7".

b) "Farem servir un índex amb els camps *nom i descripció del producte Apache Lucene* per tal que els usuaris puguin cercar els productes amb qualsevol paraula clau" no té una visió global del sistema (Lucene és un detall d'implementació que no interessa als usuaris). La versió correcta seria "Els usuaris podran fer cerques de productes per paraules clau".
c) "La web ha de funcionar sobre els navegadors més usats" és difícil de verificar. Una versió correcta: "La web es visualitzarà correctament amb els navegadors Internet Explorer, Firefox, Chrome i Safari en la seva versió més recent en el moment de posar el projecte en producció i la versió immediatament anterior". Tot i que no indica versions completes, és fàcil de verificar en el moment de posar el projecte en producció.
d) "Farem servir el bastiment .Net Entity Framework per a accedir a la base de dades" no és un requisit des del punt de vista de l'usuari (si no és que hi ha algun *stakeholder* que ens ho ha demanat). En aquest cas, no té sentit reescriure'l.
e) "Els usuaris podran omplir la sol·licitud fàcilment". També és difícil de verificar. Una versió verificable: "Un usuari sense entrenament ha de poder omplir correctament la sol·licitud sense ajuda exterior en menys de cinc minuts".

3.

a) La persona que vol comprar l'entrada és un *stakeholder*, ja que és un usuari (i, per tant, un actor) del sistema.
b) La persona que atén la centraleta telefònica d'atenció a l'usuari és un *stakeholder* potencial, ja que, per a resoldre les incidències dels usuaris, és probable que necessiti accedir al sistema com a usuària.
c) Els propietaris dels teatres són *stakeholders* ja que, en última instància, seran ells qui posaran els diners per a finançar el desenvolupament.
d) Els actors de les obres no són *stakeholders*, ja que no els afecta el sistema.
e) L'acomodador del teatre és un *stakeholder*, ja que té, com a mínim, una necessitat per cobrir: que el sistema li digui quin és el seient que ha d'ocupar cada persona (encara que sigui imprimint una entrada amb el número de seient).
f) Les persones que van al teatre però que no han comprat directament l'entrada no són *stakeholders*, ja que no tenen cap interès ni interacció amb el sistema.
g) El personal de la taquilla del teatre són *stakeholders*, ja que el sistema afecta la seva manera de treballar i, en tot cas, la seva feina és complementària a la del nou sistema. Podria ser que fossin actors si fan servir el nou sistema per a vendre les entrades que es compren a la taquilla.
h) Els administradors de sistemes de l'empresa on estan hostatjats els servidors són *stakeholders*, ja que han de poder administrar el nou sistema.
i) Els desenvolupadors de l'aplicació no són *stakeholders*.
j) El comercial que ha de convèncer els teatres que entrin a la xarxa i posin les seves entrades a la venda per mitjà de l'aplicació és un *stakeholder*, ja que necessita que el sistema sigui fàcil de vendre.
k) El programari que fan servir els teatres per a gestionar les vendes d'entrades a la taquilla no és un *stakeholder*, tot i que podria ser un actor.
l) Els desenvolupadors del programari que fan servir els teatres per a gestionar les vendes d'entrades a la taquilla són *stakeholders*, ja que necessiten que el nou sistema es pugui integrar amb el programari que han desenvolupat.
m) El cap de projecte no és un *stakeholder*.

4. Usuaris:

- El funcionari que gestiona les sol·licituds
- El funcionari que gestiona el pressupost
- La persona de l'empresa que demana l'ajut

Stakeholders:

- Els diferents governs (local, autonòmic, etc.)
- Les empreses que imparteixen la formació

5. En aquest cas, el representant dels *stakeholders* és el desenvolupador mateix de l'aplicació.

6.

a) En "Creus que s'haurien de poder adjuntar fotografies als anuncis?" no s'indica el cost d'afegir aquesta funcionalitat al sistema, per la qual cosa és difícil que ens diguin que no. Una manera més correcta seria: "Creus que s'haurien de poder adjuntar fotografies? Tingues en compte que afegir aquesta funcionalitat implicarà que no es pugui posar data de caducitat als anuncis."
b) "Creus que s'haurien d'afegir més camps?" és una pregunta massa oberta. Seria millor "Quin camp afegiries?" o, millor encara, suggerir quins camps es poden afegir.
c) "Trobes bé que els anuncis caduquin automàticament al cap d'un mes?" és una pregunta correcta per a un qüestionari.

d) "Un usuari hauria de poder modificar un anunci un cop publicat?" té el mateix problema que la primera pregunta: com que no dóna cap indicació del cost la resposta natural és un sí. Una possible solució: "Un usuari hauria de poder modificar un anunci un cop publicat? En cas afirmatiu, quina funcionalitat hauríem de sacrificar per tal d'incorporar aquesta?".

7.

- Rendiment
  - El sistema ha de suportar 2.000 usuaris connectats al mateix temps.
  - El sistema ha de desar les dades relatives a 2 milions de sol·licituds.

- Requisits lògics de la base de dades
  - El sistema ha de desar, per a una comanda, quins productes es van comprar, quina quantitat de cada producte i quin era el preu.
  - La suma dels imports dels pagaments d'una comanda ha de ser igual a la suma del valor dels productes comprats.

- Restriccions de disseny
  - El sistema ha de funcionar sobre un telèfon mòbil amb 256 MB de RAM, processador ARM v8 i sistema operatiu Android.
  - El disseny del sistema ha de seguir el recull de bones pràctiques i patrons de l'organització.

- Fiabilitat
  - En cas de problemes amb el servidor de bases de dades, en cap cas pot quedar el sistema en un estat inconsistent.
  - El sistema ha de desar automàticament la feina feta per tal de recuperar-la automàticament en cas de caiguda no prevista.

- Disponibilitat
  - El sistema ha de poder operar de manera que, si cau un servidor, un altre s'encarregui de continuar donant servei als usuaris.
  - El sistema ha de poder estar disponible el 90% del temps.

- Seguretat
  - El sistema ha de registrar els accessos a informació sobre clients fets pels usuaris.
  - El sistema ha d'impedir que els usuaris d'una oficina accedeixin a les dades personals dels clients d'una altra oficina.

- Mantenibilitat
  - El sistema ha de tenir en compte que, en el futur, s'incorporaran nous proveïdors per al servei de missatgeria i es voldrà poder canviar entre aquests fàcilment.
  - El sistema ha de registrar, en cas d'error inesperat, informació sobre la transacció en curs i també el punt exacte en què s'ha produït l'error per tal de facilitar-ne el diagnòstic.

- Portabilitat
  - El sistema ha de poder funcionar sobre un sistema gestor de bases de dades diferent de l'actual.
  - El sistema ha de poder funcionar sobre els sistemes operatius Linux 2.6, Windows 7 i MacOS X 10.6.

8. Cap dels tres valors. Cal que parlin i exposin el motiu de l'estimació de cadascú i, un cop posats d'acord, tornar a donar una estimació.

9. Trigarem 42 vegades el que trigui A, és a dir, 42 dies. Si ens hem equivocat d'un 10% en estimar A però la resta d'estimacions estan bé, trigarem un 10% més (46,2 dies) a acabar.

10. La plantilla d'OpenUP no té en compte els objectius dels actors sinó només el seu comportament. D'altra banda, incorpora informació que no hi ha a la plantilla que hem emprat al subapartat 5.1 com ara els escenaris clau o els requisits especials.

11. Un exemple podria ser "Retirar una bicicleta espatllada". En aquest cas, l'actor principal seria l'encarregat del manteniment de les bicicletes.

12. Un cas d'ús relacionat amb aquesta funcionalitat podria ser "Diagnosticar problema intern" i l'actor principal seria l'encarregat de reparar el telèfon.

13. Objectiu global: anar al teatre

Objectiu d'usuari: comprar les entrades

Tasca: iniciar el pagament

14. Actor: visitant. És un tipus d'usuari que acostuma a arribar a l'aplicació de manera casual (per exemple, per mitjà d'un cercador) amb un interès concret i que, un cop hagi trobat la resposta a la pregunta que té, no continuarà fent servir el sistema. El seu nivell tècnic pot ser molt baix i, fins i tot, pot ser un usuari molt ocasional d'Internet. La seva motivació principal per a fer servir el sistema és trobar una resposta a una pregunta que té. Hi ha la possibilitat que es registri com a membre per plantejar la pregunta si no troba la resposta.

Actor: membre de la comunitat. És un usuari que s'ha registrat al sistema per tal de poder fer preguntes o respostes. El seu nivell tècnic no ha de ser necessàriament gaire alt però està habituat a fer servir aplicacions a Internet. La seva motivació principal per a fer servir el sistema és participar a la comunitat per tal de guanyar-se un cert prestigi, ja que no paguem per les intervencions.

Actor: moderador. Pot ser un membre de la comunitat amb un cert prestigi o bé algú posat per l'empresa que gestiona l'aplicació. En qualsevol cas és algú amb autoritat acceptada per la resta de membres de la comunitat que té com a finalitat identificar les intervencions poc apropiades i els membres que no es comportin segons els criteris establerts.

Actor	Nivell	Objectiu
Visitant, Membre	Global	Resoldre un dubte propi
Visitant, Membre	Usuari	Trobar una resposta
Visitant	Usuari/Tasca	Registrar-se com a membre
Visitant, Membre	Tasca	Llegir una pregunta amb les seves respostes
Membre	Usuari	Crear una pregunta
Membre	Tasca	Redactar la pregunta
Membre	Tasca	Veure la llista de les seves preguntes
Membre	Global	Resoldre un dubte a algú altre
Membre	Usuari	Trobar una pregunta per a intervenir-hi
Membre	Tasca	Veure la llista de preguntes amb poques o cap resposta
Membre	Tasca	Redactar una resposta
Moderador	Global	Moderar la comunitat
Moderador	Usuari	Trobar intervencions poc adequades
Moderador	Tasca	Veure la llista d'intervencions sospitoses

Casos d'ús:

Resoldre un dubte propi (global)

L'usuari (sigui visitant o membre) té un dubte i accedeix al sistema per trobar una resposta. Si no troba la resposta i és un membre de la comunitat, pot crear una pregunta. Si no és membre, pot registrar-se en aquest moment i convertir-se en membre.

Registrar-se com a usuari (usuari)

Un usuari visitant decideix que vol formar part de la comunitat i es vol registrar. El sistema li demanarà les dades personals que formen el seu perfil i li assignarà unes credencials per a identificar-se en el futur.

Crear una pregunta (usuari)

Un membre de la comunitat pot crear una nova pregunta en qualsevol moment. Un cop creada la pregunta, aquesta apareixerà a la llista de les seves preguntes per tal que pugui anar fàcilment a trobar les respostes.

Trobar una resposta (usuari)

L'usuari pot indicar una sèrie de paraules clau i el sistema li mostrarà una llista de preguntes que, o bé a la pregunta o bé a la resposta, contenen les paraules clau esmentades. L'usuari n'escull una i el sistema mostra la pregunta juntament amb totes les seves respostes. Alternativament, l'usuari pot arribar directament a la pàgina que mostra la pregunta ja sigui per mitjà d'un cercador o dels favorits del seu navegador. Si l'usuari era un membre, també pot veure la llista de les seves preguntes i arribar al detall des d'allà.

Resoldre un dubte a algú altre (global)

L'usuari (un membre de la comunitat) vol participar per resoldre un dubte d'algú altre. El primer que fa és trobar una pregunta per a intervenir-hi.

Trobar una pregunta per a intervenir-hi (usuari)

L'usuari pot cercar una pregunta per paraula clau (per exemple, per a buscar preguntes sobre els temes que domina) o bé pot veure la llista de preguntes amb poques o cap resposta. Un cop triada la pregunta en què vol intervenir, redactarà la resposta i l'enviarà.

Moderar la comunitat (global)

Els moderadors són els responsables de trobar intervencions poc adequades, respondre a les queixes dels usuaris i tancar una pregunta quan aquesta ja té un nombre suficient de respostes.

Trobar intervencions poc adequades (usuari)

El moderador demana una llista d'intervencions sospitoses i el sistema li mostra aquelles intervencions (siguin preguntes o respostes)

Exercicis d'autoavaluació

1. Ha de ser observable i expressar una necessitat o restricció que afecti el programari per desenvolupar. (Vegeu el subapartat 1.1.)

2. Els usuaris sempre són *stakeholders*, ja que tenen interessos en el sistema per la raó mateixa que el faran servir. Però no tots els *stakeholders* són usuaris, ja que hi ha persones que no faran servir directament el sistema però hi tenen interès. (Vegeu el subapartat 1.2.)

3. Comunicar les necessitats i objectius dels *stakeholders*. (Vegeu el subapartat 1.2.)

4. Els requisits funcionals expressen necessitats que ha de satisfer el sistema, què ha de fer, mentre que els no funcionals expressen restriccions sobre les possibles solucions, com ho ha de fer. (Vegeu el subapartat 1.3.)

5. Obtenció, gestió, documentació i verificació. (Vegeu el subapartat 1.4.)

6. No, ja que no seria possible satisfer-los. Aquest és un dels motius pels quals, un cop tenim una llista de requisits candidats, cal fer una selecció dels que realment volem preveure. (Vegeu els apartats 2 i 3.)

7. Durant la fase d'organització i consolidació del conjunt inicial i la de refinament es filtren i documenten millor les idees seleccionades. (Vegeu el subapartat 2.1.1.)

8. La identificació de *stakeholders*, la identificació de requisits i la modelització de rols d'usuari. (Vegeu els subapartats 2.1.1 i 2.1.2.)

9. La modelització de rols d'usuari i les Persones. (Vegeu el subapartat 2.1.2.)

10. Representant. (Vegeu el subapartat 2.1.3.)

11. Perquè condiciona les respostes (menys d'1 segon, de 20 en 20) i es limita al coneixement actual en el sentit que pot ser que hi hagi solucions alternatives a la paginació per a retornar

grans quantitats de dades (per exemple, una barra de desplaçament contínua). (Vegeu el subapartat 2.2.1.)

12. No. Podem fer servir el prototip per a resoldre dubtes sobre els requisits del sistema però, per definició, el prototip no es fa servir com a part del producte definitiu. Si el féssim servir ja no es tractaria d'un prototip sinó d'una implementació parcial. (Vegeu el subapartat 2.2.2.)

13. La llista d'aspectes mencionats són: eficiència d'ús, facilitat de memorització, taxa d'errors, satisfacció, retroalimentació, corba d'aprenentatge, comprensibilitat i accessibilitat. (Vegeu el subapartat "Requisits d'usabilitat i humanitat".)

14.

- Consideracions sobre la imatge corporativa de l'organització (de presentació)
- Satisfacció dels usuaris (usabilitat i humanitat)
- Accessibilitat per part de persones discapacitades (usabilitat i humanitat)
- Temps mitjà entre caigudes del sistema (acompliment)
- Requisits sobre com s'han de programar els lliuraments dels diferents artefactes i versions del sistema (operacionals i d'entorn)
- Sistemes operatius que cal que suporti el sistema (manteniment i suport)
- Unitats que es faran servir per a mostrar longituds, pesos, etc. (culturals i polítics)
- Llenguatges en què es pot mostrar la interfície del sistema (culturals i polítics)

(Vegeu el subapartat 2.2.3.)

15. Ens caldrà una estimació del cost d'implementar-lo, la importància que té per als *stakeholders* i el risc que implica. També caldrà tenir en compte els recursos disponibles, que no són informació dels requisits candidats. (Vegeu els apartats 3 i 3.3.)

16. Reals, com, per exemple, les hores, o fictícies, com ara, diguem-ne, punts. (Vegeu el subapartat 3.1.1.)

17. La comparació i la triangulació. (Vegeu el subapartat 3.1.2.)

18. Perquè cal que cadascú pugui fer la seva estimació sense estar condicionat i, per tant, tampoc no ha d'estar condicionat per la majoria i és important que, en cas de dubte, hi hagi diàleg. Així es pot descobrir quins són els supòsits diferents que ens han dut a estimacions diferents. (Vegeu el subapartat 3.1.3.)

19. Que el valor és relatiu als *stakeholders*, que pot ser que no es posin d'acord, i que pot passar que als *stakeholders* els costi prioritzar. (Vegeu el subapartat 3.2.)

20. La votació amb nombre limitat de vots. (Vegeu el subapartat 3.2.1.)

21. Ha de ser correcta, no ambigua, completa, consistent, verificable, modificable i traçable i ha de tenir els requisits etiquetats. (Vegeu el subapartat 4.1.)

22. És ambigua. (Vegeu el subapartat 4.1.)

23. És inconsistent. (Vegeu el subapartat 4.1.)

24. No és verificable. (Vegeu el subapartat 4.1.)

25. No és prou modificable. (Vegeu el subapartat 4.1.)

26. No, una història d'usuari és això i una conversa que no queda registrada. (Vegeu el subapartat 4.3.)

27. Les històries per tenir en compte en la iteració actual, les més prioritàries, haurien de ser més petites i detallades. Les històries que encara no implementarem, les menys prioritàries, haurien de ser més grans. (Vegeu el subapartat 4.3.)

28. L'actor té comportament en la interacció amb el sistema, interactua amb el sistema, mentre que l'*stakeholder* pot no tenir-lo. (Vegeu el subapartat 5.2.)

29. Si la precondició d'un cas d'ús no es compleix no és possible que el cas d'ús s'executi i, per tant, no s'hi podrà dur a terme la interacció descrita. (Vegeu el subapartat 5.3.4.)

30. La seqüència d'esdeveniments que espera l'actor principal quan engega el cas d'ús i que el porten a satisfer el seu objectiu. (Vegeu el subapartat 5.3.5.)

31. De més concrets a més generals: Tasca, Usuari i General. (Vegeu el subapartat 5.4.1.)

32. El d'organització. (Vegeu el subapartat 5.4.2.)

33. Àmbit de subsistema. (Vegeu el subapartat 5.4.2.)

34. L'actor iniciador també podria ser un actor de suport que interactua amb el sistema per ordre de l'actor principal (com ara l'operari d'una central de trucades) o pot ser el rellotge (que fem servir per a representar l'iniciador quan un cas d'ús s'inicia en resposta a un esdeveniment temporal). (Vegeu el subapartat 5.5.1.)

35. Podem crear un cas d'ús amb la part comuna a tots dos i incloure'l en tots dos casos d'ús. (Vegeu el subapartat "Inclusió per reutilització".)

36. Podem documentar l'extensió com un nou cas d'ús i incloure'l en el cas d'ús original. (Vegeu el subapartat "Separació d'una extensió en un cas d'ús inclòs".)

37. Podem crear un cas d'ús nou per a l'extensió i fer servir la relació d'extensió per a indicar en quin punt del cas d'ús original en modifica el comportament. (Vegeu el subapartat 5.5.4.)

38. Podem indicar com a precondició dels casos d'ús que ho requereixin que cal estar autenticat. Una altra solució és documentar l'autenticació com un pas més del cas d'ús; en aquest segon cas, si fos complexa es podria considerar un cas d'ús de nivell de tasca i incloure-la on sigui necessari. (Vegeu el subapartat 5.6.1.)

39. Podem crear un únic cas d'ús per a simplificar la documentació resultant. (Vegeu el subapartat 5.6.3.)

40. Consisteix en un cas d'ús que fa servir una plantilla que permet evitar redundàncies creant diversos casos d'ús que segueixen la mateixa plantilla. (Vegeu el subapartat 5.6.4.)

41. En el cas de modelitzar processos de negoci, el més habitual és que els casos d'ús tinguin àmbit d'organització. (Vegeu el subapartat 5.6.5.)

Glossari

actor *m* Algú que té comportament en un cas d'ús. Pot ser una persona, però també una organització o un sistema (de programari o d'un altre tipus) extern al sistema que analitzem.

actor de suport *m* Actor extern al sistema que proporciona un servei al sistema.

actor iniciador *m* Actor que inicia la interacció amb el sistema. Pot ser l'actor principal, però també pot ser un intermediari entre aquest i el sistema. També pot ser que el cas d'ús s'iniciï de manera automàtica o en reacció a un esdeveniment extern, cas en el qual l'actor iniciador seria el rellotge o aquest esdeveniment, respectivament.

actor principal *m* *Stakeholder* que fa una petició al sistema per a rebre'n un dels serveis i així satisfer un objectiu.

àmbit (d'un cas d'ús) *m* Definició de què és el que considerem el sistema analitzat: què en queda dins i què en queda fora. Hi ha tres grans tipus d'àmbits: d'organització, de sistema i de subsistema.

àmbit d'organització *m* Àmbit en què el sistema analitzat és una organització o negoci. Tots els sistemes (informàtics o no) i les persones de dins l'organització analitzada formen part del sistema que documentem i, per tant, no apareixen com a actors.

àmbit de sistema *m* Àmbit en el qual el sistema analitzat és un sistema informàtic (típicament el que volem desenvolupar). Tots els components del sistema són interns i, per tant, no apareixen com a actors.

àmbit de subsistema *m* Àmbit en què el sistema analitzat en el cas d'ús és una part d'un sistema informàtic, potser un subsistema o un bastiment. Els altres components del sistema informàtic i tots els usuaris directes del component seran actors en el model.

artefacte *m* En el context de l'enginyeria del programari, cada un dels documents, models, programes, etc., que es generen com a resultat del treball de l'enginyer.

autenticació *f* Acte d'establir o confirmar l'autenticitat d'algú (típicament l'usuari del sistema informàtic); és a dir, confirmar que l'usuari és, efectivament, qui diu ser. Una manera típica d'autenticació és demanar a l'usuari una contrasenya que només la persona que diu ser coneix.

backlog *m* Llista d'històries d'usuari, ja sigui de totes les històries d'un projecte en desenvolupament (*product backlog*) o de les que s'inclouen en una iteració.

brainstorming *m* Pluja d'idees. Tècnica de treball en grup dissenyada per a generar un gran nombre d'idees per a solucionar un problema. En el context de l'enginyeria del programari es pot fer servir, per exemple, per a identificació de *stakeholders* i per a identificar requisits, entre d'altres usos.

cas d'ús *m* Document que recull el contracte entre el sistema i els seus *stakeholders*. Descriu el comportament del sistema i les interaccions en diversos escenaris mostrant com el sistema respon a les peticions i objectius dels *stakeholders*.

cas d'ús d'extensió *m* Cas d'ús que descriu una extensió d'un altre cas d'ús (que anomenem *de base*).
Vegeu **extensió**

cas d'ús de base *m* Cas d'ús del qual hi ha una extensió descrita en un altre cas d'ús (que anomenem *d'extensió*).
Vegeu **extensió**

checklist *f* Terme anglès per a *llista predefinida*.

create, read, update and delete Funcionalitats bàsiques de l'emmagatzemament persistent que, tot sovint, s'ofereixen als usuaris dels sistemes informàtics per al manteniment de certes dades. Es poden documentar com a casos d'ús individuals o agrupar en un cas d'ús que agrupi diverses d'aquestes operacions quan es fan sobre unes mateixes dades.
Sigla **CRUD**

disponibilitat *f* Capacitat d'un sistema informàtic d'estar disponible per als seus usuaris. Si un usuari no pot accedir al sistema pel motiu que sigui, diem que el sistema "no està disponible". Habitualment es mesura en percentatge de temps que el sistema està disponible (90%, 99%, etc.).

escenari *m* Seqüència d'accions i interaccions que es produeixen en un cas d'ús sota certes condicions, expressades sense (o amb poques) ramificacions condicionals.

escenari principal *m* Escenari descrit completament per un cas d'ús des de l'inici del cas d'ús fins la seva compleció i que porta l'actor principal a complir el seu objectiu. Tot i que és un escenari d'èxit no ha de ser necessàriament l'únic del cas d'ús.

extends Nom oficial de la relació d'extensió entre casos d'ús a UML.
Vegeu **extensió** (2).

extensió (d'un cas d'ús) *f* 1. Un fragment d'escenari que comença sota certa condició des d'un altre escenari. 2. Relació entre dos casos d'ús en què un cas d'ús que anomenem *d'extensió* descriu una extensió d'un altre cas d'ús que anomenem *de base*, de tal manera que el cas d'ús de base no té cap referència al cas d'ús d'extensió sinó que el cas d'ús d'extensió indica on del cas d'ús de base es produeix l'extensió.

fiabilitat *f* És la capacitat d'un sistema informàtic de proporcionar informació correcta de manera robusta (és a dir, tolerant a possibles errors per part dels usuaris o del maquinari).

garantia mínima (d'un cas d'ús) *f* Allò que podem assegurar que passarà en finalitzar el cas d'ús sigui quin sigui l'escenari que es produeixi.

història d'usuari *f* Tècnica de documentació de requisits que fa èmfasi en la comunicació verbal i minimitza la documentació escrita. Una història d'usuari està formada per tres components: el nom, la conversa entre els desenvolupadors i els clients i la llista de proves que cal fer per a verificar que s'ha implementat correctament.

include Nom oficial de la relació d'inclusió entre casos d'ús en el llenguatge UML.
Vegeu **inclusió**

inclusió (entre casos d'ús) *f* Relació entre dos casos d'ús en què en un pas d'un escenari d'un cas d'ús es crida un altre cas d'ús (que anomenem *subcàs d'ús*).

mantenibilitat *f* Capacitat d'un sistema informàtic de ser fàcil de mantenir. És a dir, que sigui fàcil tant corregir els errors que es trobin (manteniment correctiu) com afegir-hi noves funcionalitats (manteniment evolutiu).

pila de producte *f* Vegeu ***backlog***.

portabilitat *f* Capacitat d'un sistema informàtic de funcionar sobre més d'una plataforma tecnològica o de funcionar sobre una plataforma diferent a aquella per a la qual va ser desenvolupat.

precondició *f* Condició que s'ha de complir prèviament. En el cas dels casos d'ús, condicions que s'han de donar abans de poder-se executar el cas d'ús.

product backlog Vegeu ***backlog***.

prototip *m* Producte que es crea per a demostrar una funcionalitat del producte final. El prototip només serveix per a obtenir informació sobre com ha de ser el producte final però no en forma part (és a dir, un cop obtinguda la informació que es volia obtenir el prototip es descarta) per la qual cosa no és necessari ni tan sols fer servir la mateixa tecnologia que es farà servir per a produir el producte final.

rendiment *m* Mesura de la relació entre el volum de feina feta i el consum de recursos emprat.

representant (d'un *stakeholder*) *m* Persona que actua com a intermediari entre l'*stakeholder* i l'equip de desenvolupament.

requeriment *m* Acció o efecte de requerir. Sovint es fa servir com a sinònim de *requisit*.

requisit *m* Condició exigida o necessària per a una cosa. En el camp de l'enginyeria del programari, necessitat o restricció que afecta un producte de programari definint quines solucions són adequades (el compleixen) i quines no (no el compleixen) des del punt de vista d'aquesta necessitat o restricció. Una solució serà adequada si satisfà tots els requisits del sistema.

requisit candidat *m* Necessitats o restriccions obtingudes en una primera etapa d'obtenció de requisits i que es convertiran en requisits només si són seleccionades com a tals a l'hora de definir l'abast del projecte per desenvolupar.

requisit funcional *m* Requisit que descriu la funcionalitat esperada del sistema; és a dir, que descriu el comportament del sistema davant els estímuls que li arriben de l'exterior.
Vegeu també **requisit no funcional**.

requisit no funcional *m* Requisit que restringeix el conjunt de solucions possible, típicament en forma de restricció, sense parlar de la funcionalitat.

seguretat *f* Capacitat d'un sistema informàtic de protegir la informació davant de robatoris, usos fraudulents, corrupció de les dades, etc.

stakeholder *m* Persona o entitat amb un interès sobre el producte que estem desenvolupant.

subcàs d'ús *m* Cas d'ús que és inclòs en un altre cas d'ús.
Vegeu **inclusió**.

triangulació *f* Tècnica d'estimació que consisteix a obtenir l'estimació d'un requisit a partir de dos requisits prèviament estimats (un de més complex i un de més senzill).

usabilitat *f* Capacitat d'un sistema informàtic de ser fàcil d'utilitzar.

usuari *m* Persona que fa servir un sistema informàtic.

Bibliografia

Bibliografia principal

Cockburn, A. (2001). *Writing Effective Use Cases*. Addison-Wesley.

Aquest llibre ens dóna indicacions sobre com cal aplicar, de manera eficaç, els casos d'ús a la gestió de requisits. Tot i centrar-se en l'ús de casos d'ús per a la documentació de requisits, gran part del discurs és aplicable amb independència de l'estil de documentació.

Bibliografia complementària

Cohn, M. (2004). *User Stories Applied*. Addison Wesley.

Aquest llibre tracta, principalment, de la tècnica de les històries d'usuari però, igual que el de Cockburn, gran part del discurs és aplicable a l'obtenció, gestió i documentació de requisits amb independència de l'estil de documentació.

Davis, Al. M. (2005). *Just Enough Requirements Management*. Dorset House.

En aquesta obra, Alan Davis proposa un enfocament pragmàtic per a l'obtenció i la gestió de requisits.

Referències bibliogràfiques

Cockburn, A. http://alistair.cockburn.us/Use+case+questions (Darrera visita: setembre 2010)

Constantine, L. *Users, Roles and Personas*.
http://www.foruse.com/articles/rolespersonas.pdf (Darrera visita: setembre 2010)

Cucumber. http://cukes.info/ (Darrera visita: setembre 2010)

Diversos autors (2010). *Unified Modeling Language™ (UML®)*. Object Management Group.

Fitnesse. http://fitnesse.org/ (Darrera visita: setembre 2010)

Leffingwell, D.; Widrig, D. (2000). *Managing Software Requirements - A Unified Approach*. Addison Wesley.

Volere Requirements Specification Template. http://www.volere.co.uk/template.htm (Darrera visita: setembre 2010)