Mètriques de productivitat i qualitat del programari per a la gestió de projectes

Carlos Mateu Lopez
Estudiant d'Enginyeria Tècnica d'Informàtica de Sistemes

Consultor: Xavier Martínez Munné
Data Lliurament: 13/1/2016


Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL

Títol del treball:	Mètriques de productivitat i qualitat del programari per a la gestió de projectes
Nom de l'autor:	*Carlos Mateu Lopez*
Nom del consultor:	*Xavier Martínez Munné*
Data de lliurament (mm/aaaa):	*01/2016*
Àrea del Treball Final:	*Gestió de Projectes*
Titulació:	*Enginyeria Tècnica d'Informàtica de Sistemes*

Resum del Treball (màxim 250 paraules):

Aquest projecte és un estudi de les tècniques disponibles per avaluar el cost de desenvolupament d'un programari i projectar consegüentment els recursos que seran necessaris per l'elaboració del mateix. També es recullen mètriques de mesura de qualitat del producte, amb una atenció especial als estàndards ISO dedicats a la matèria.

Abstract (in English, 250 words or less):

This project is a study of the available techniques to evaluate the expenses of a software development and, consequently, predict de resources that will be needed to perform it. It is also a collection of measurements about the quality of the software taking in consideration ISO's standards relative to this subject.

Paraules clau (entre 4 i 8):

Productivitat; qualitat; mètriques

Índex

Llista de figures

Llistat de taules

1 Introducció

1.1 Context i justificació del Treball

La gestió d'un projecte informàtic, depèn en bona mesura de la seva planificació. Per elaborat aquest pla de treball és necessari mesurar correctament els costos que se'n derivaran de la seva producció i el temps necessari per portar-lo a terme. És per això que resulta cabdal l'estudi de les mètriques que defineixen la productivitat, quines son les tècniques existents i com s'han d'utilitzar.

De la mateixa manera es vol donar un visió de les tècniques per mesurar la qualitat d'un producte mitjançant l'estudi del procediment que s'ha portat a terme i els estàndards de qualitat Europeus que regulen l'elaboració de productes informàtics.

1.2 Objectius del Treball

Es tracta d'un estudi de les tècniques disponibles per avaluar el cost de desenvolupament d'un programari i projectar consegüentment els recursos que seran necessaris per l'elaboració d'un projecte informàtic.

També es recullen les mètriques que mesuraran la qualitat del producte segons els mètodes emprats en la seva elaboració i fase de proves

1.3 Enfocament i mètode seguit

El mètode seguit per a l'elaboració del projecte és la cerca de informació a Internet, biblioteca especialitzada en Informàtica, articles publicats i consultes a experts gestors de projectes de software en la seva experiència professional.

D'aquesta manera es pretén recopilar primerament el coneixement teòric existent per a després estudiar les tècniques pràctiques emprades en projectes reals.

1.4 Planificació del Treball

La planificació temporal del projecte s'inicia el dia 23 de Setembre (dia del plantejament dels possibles temes a escollir), fins al 13 de Gener que és el termini màxim de lliurament.

Entre aquestes dues dates s'estableixen 3 avaluacions parcials (PAC1, PAC2, PAC3) per tal de fer un seguiment del anàlisi que s'ha portat a terme i reconduir la planificació si s'escau.

Durant 3 setmanes d'Octubre (del 4 al 26) no tindré disponibilitat per avançar feina del TFC, pel que entregaré la planificació del projecte de forma anticipada el dia 4 d'Octubre i iniciaré el desenvolupament de les fites marcades per a la PAC2 el dia 26/10.

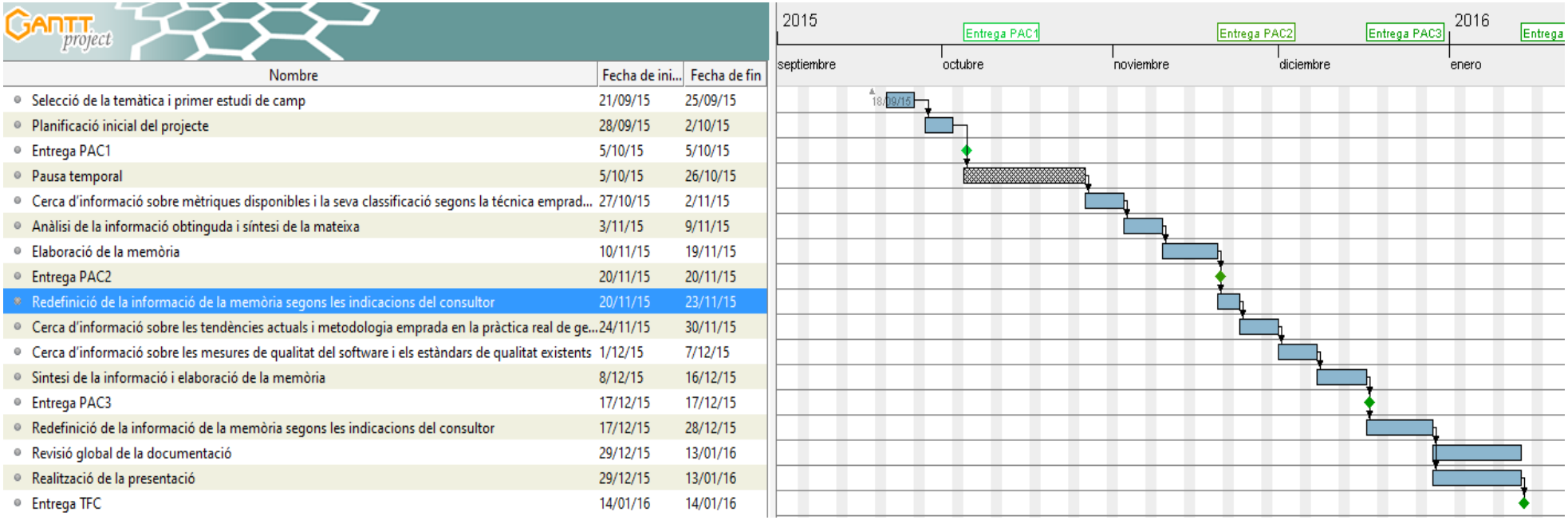
La descripció de les tasques identificades i el temps planificat per assolir les fites és la següent:

Tasca	Temps emprat	Data de lliurament prevista	Data de lliurament efectiva	FITA
Selecció de la temàtica i primer estudi de camp	1 setmana	4/10/2015	**3/10/2015**	**PAC1**
Planificació inicial del projecte	1 setmana			
Cerca d'informació sobre mètriques disponibles i la seva classificació segons la tècnica emprada per al càlcul de costos	2 setmanes	18/11/2015	**20/11/2015 (*)**	**PAC2**
Anàlisi de la informació obtinguda i síntesi de la mateixa	1 setmana			
Elaboració de la memòria	1 setmana			
Entrega PAC2				
Redefinició de la informació de la memòria segons les indicacions del consultor	1 setmana			Post-PAC2
Cerca d'informació sobre les tendències actuals i metodologia emprada en la pràctica real de gestió de projectes	1 setmana	16/12/2015	**16/12/2015**	**PAC3**
Cerca d'informació sobre les mesures de qualitat del software i els estàndards de qualitat existents	1 setmana			
Síntesi de la informació recopilada i elaboració de la memòria	1 setmana			
Entrega PAC3				
Redefinició de la informació de la memòria segons les indicacions del consultor	1 setmana			Post-PAC3
Revisió global de la documentació	2 setmanes	Del 10 al 13 Gener 2016		**Final TFC**
Realització de la presentació				
Entrega TFC				

Taula 1

(*) La data d'entrega de la PAC2 es veu compromesa per una causa aliena a la planificació i desenvolupament del projecte, degudament justificada i acceptada pel consultor.

Aquesta és la planificació temporal del projecte representant les tasques i les seves dependències en un diagrama de Gantt:



Nombre	Fecha de ini...	Fecha de fin
Selecció de la temàtica i primer estudi de camp	21/09/15	25/09/15
Planificació inicial del projecte	28/09/15	2/10/15
Entrega PAC1	5/10/15	5/10/15
Pausa temporal	5/10/15	26/10/15
Cerca d'informació sobre mètriques disponibles i la seva classificació segons la técnica emprad...	27/10/15	2/11/15
Anàlisi de la informació obtinguda i síntesi de la mateixa	3/11/15	9/11/15
Elaboració de la memòria	10/11/15	19/11/15
Entrega PAC2	20/11/15	20/11/15
Redefinició de la informació de la memòria segons les indicacions del consultor	20/11/15	23/11/15
Cerca d'informació sobre les tendències actuals i metodologia emprada en la pràctica real de ge...	24/11/15	30/11/15
Cerca d'informació sobre les mesures de qualitat del software i els estàndars de qualitat existents	1/12/15	7/12/15
Síntesi de la informació i elaboració de la memòria	8/12/15	16/12/15
Entrega PAC3	17/12/15	17/12/15
Redefinició de la informació de la memòria segons les indicacions del consultor	17/12/15	28/12/15
Revisió global de la documentació	29/12/15	13/01/16
Realització de la presentació	29/12/15	13/01/16
Entrega TFC	14/01/16	14/01/16

Il·lustració 1

1.5 Breu sumari de productes obtinguts

El producte obtingut de l'elaboració d'aquest projecte serà la memòria, que contindrà el resultant del objecte d'estudi i una presentació virtual de la mateixa, on es sintetitzarà el contingut en un vídeo explicatiu.

1.6 Breu descripció dels altres capítols de la memòria

1. El primer capítol és una introducció de la definició dels termes més utilitzats en relació amb les mètriques.

2. A continuació es tracten les mètriques de productivitat, la seva definició i característiques, els conceptes de productivitat principals (temps, mida i esforç), així com les tècniques principals més conegudes per al càlcul de la productivitat en el desenvolupament de programari.

3. En el capítol 4 es tracten les eines de productivitat més conegudes per al càlcul de mètriques de productivitat. Per aquesta finalitat s'han escollit COMOMO 81 i COMOMOII, que donen una visió de com han evolucionat aquestes eines amb el temps per adaptar-se a les noves tècniques (de LOC a l'adaptació a punts funció).

4. En el cinquè capítol s'introdueix un mètode més actual que segueix en evolució i que mira d'adaptar-se al pas de temps i les noves necessitats, es tracta dels *full function points* amb COSMIC-FFP.

5. Per últim, el capítol 6 descriu les mètriques destinades a la millora de la qualitat del desenvolupament, la descripció dels models inicials de MCCall i Boehm, que encara avui són referència, seguit de la descripció de les normes ISO que tracten d'elaborar estàndards en el terreny de la qualitat del programari.

2 Definició de conceptes

2.1 Entitat

Un objecte que serà caracteritzat mitjançant la mesura dels seus atributs [ISO -15939].

Podem entendre com a entitats:

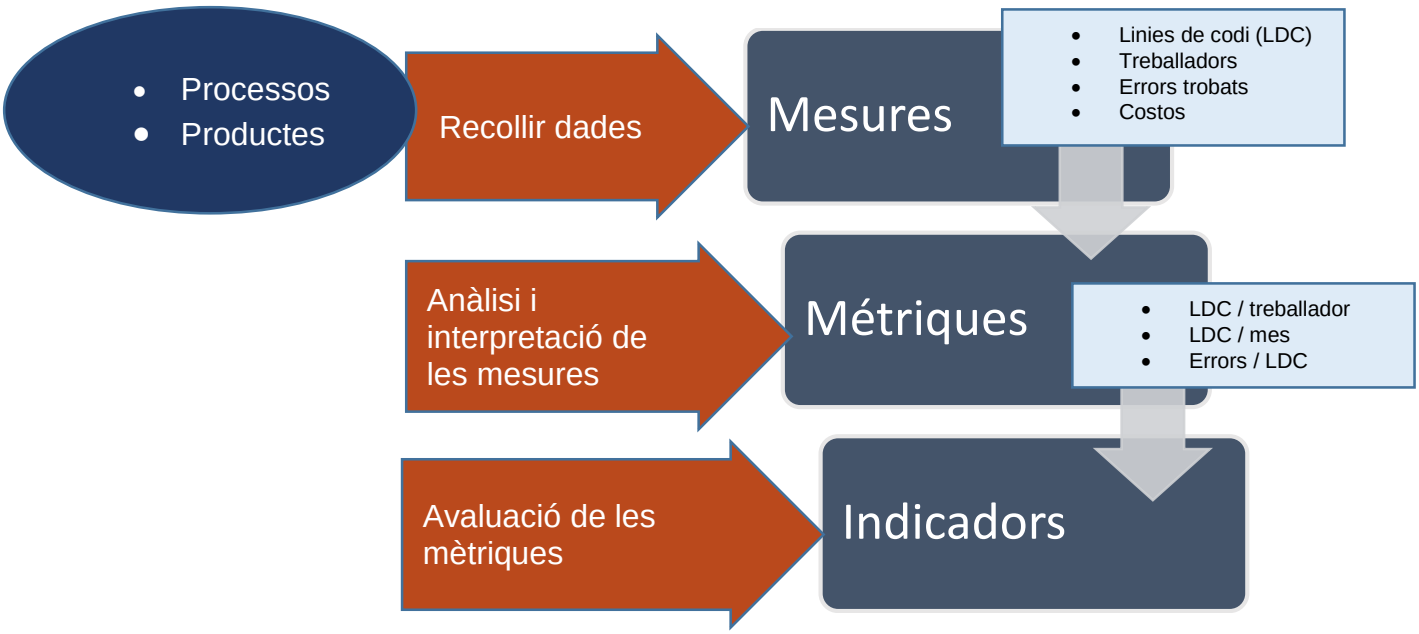
- Productes
- Projectes
- Serveis
- Processos

2.2 Atribut

Una propietat mesurable, física o abstracta, d'una entitat [ISO 14598 - 1:1999].

Podem diferenciar dues classes d'atributs:

- Atributs interns: Es poden mesurar de manera directe i són independents d'altres atributs (preu, memòria, edat...)
- Atributs externs: Els atributs que relacionen l'entitat que s'estudia amb el seu entorn. Per a la seva mesura caldrà d'altres atributs. Es tracta d'atributs com per exemple: Adaptabilitat, productivitat, usabilitat...



Il·lustració 2

2.3 Mesura

Les mesures proporcionen una indicació quantitativa sobre els atributs que avaluen. En el seguiment d'un procés o desenvolupament d'un producte recollim dades per crear-ne mesures.

Algunes mesures podrien ser:

- Línies de codi
- Persones que treballen en el procés o producte
- Errors que s'hi troben
- Costos derivats de l'execució del procés o producte

Les mesures no serveixen per a comparar, amb les mètriques podem relacionar y comparar mesures.

2.4 Mètrica

Les mètriques tenen com a objectiu avaluar un producte o procés mitjançant la comparació de mesures obtingudes en projectes passats i d'aquesta manera incrementar l'eficàcia i reduir els errors en la planificació i gestió d'un projecte .

L'estudi de de les mètriques ens permet:

- Controlar la qualitat del producte
- Millorar la qualitat d'un procés o producte
- Obtenir indicadors per preveure el temps necessari i cost estimat d'un projecte.

2.5 Indicador

Les mètriques no es poden interpretar per si mateixes, per a obtenir un resultat de l'estudi de les mètriques cal comparar-les amb un barem d'acceptació o criteri base. Aquesta línia base és la que estableix si la mètrica és en el llindar previst o està fora del rang establert com desitjable.

Per exemple:

Podem definir un indicador que estableix el rang:

De 0 a 40 : (I) Inacceptable
De 40 a 70 : (A) Acceptable
De 70 a 100 : (D) Desitjable

Si la mesura obtinguda és superior a 40, aquesta mètrica serà acceptada, per que l'indicador es =A

3 Mètriques de productivitat

3.1 Descripció i característiques

En el camp de la informàtica en general i la construcció de programari en particular, trobem una gran diversitat de mètriques destinades a avaluar els productes per fi de millorar tant la seva qualitat, com la planificació dels processos i estimar els costos que se'n deriven.

3.1.1 Utilitat de l'ús de mètriques

L'enginyeria de programari és una disciplina que es troba a mig camí entre l'art i la ciència, això implica que hi ha moltes maneres de programar un mètode que faci una determinada funció, molts llenguatges de programació disponibles per a diferents plataformes i diversitat de professionals que segons la seva experiència poden ser més o menys productius i creatius.

Per aquesta raó podem trobar una gran diversitat de mètriques destinades a:

- Optimitzar la **productivitat** analitzant els processos seguits en la construcció del producte.
- Millorar la **qualitat** minimitzant-ne els errors.

3.1.2 Manca de garanties

L'ús de mètriques no garanteix que el proper projecte actual serà millor que l'anterior, ni tampoc que el programari deixarà de contenir errors o es minimitzaran. Altrament, si està demostrat que si les mesures obtingudes son objectives i adients, l'ús de mètriques ajuda a millorar els procediments i la qualitat.

3.1.3 Objectivitat de les mesures

El problema de l'ús de mètriques per mesurar el programari radica en que les mesures recollides per l'avaluació de la productivitat i qualitat les ha de fer una persona, normalment implicada en el projecte i que coneix molts aspectes del mateix.

Per aquest motiu, és especialment important que les mesures i els indicadors siguin objectius ja que la resta del procés dependrà d'aquesta fase inicial. Quan més alienes a la subjectivitat siguin les mètriques i els indicadors, millors resultats s'obtindran.

3.2 Mètriques de productivitat

3.2.1 Les 3 mètriques principals de la productivitat

Un gestor de projecte ha de conèixer les mètriques i el seu ús i ha de saber quines mètriques escollir per saber el grau de qualitat dels seus projectes així com la posició que ocupen amb els projectes de la competència. Les 4 mètriques principals que permeten traçar una línia de referència son:

- Temps
- Mida
- Esforç

3.2.1.1 Temps

La mètrica que té com a objecte d'estudi el temps, s'encarrega de mesurar i comparar el temps necessari per a assolir un producte. Normalment es combinarà amb altres mètriques com les línies de codi construïdes en un període de temps per un treballador.

Exemple d'una combinació de mètriques per estimar el temps necessari per a la realització d'un projecte, tenint en compte el nombre de classes que contindrà, el temps estimat per a la realització d'una classe i la mètrica *Effort*:

Temps (dies) = (TC * Day-Class) / **EffortApplied**

TC (Total Classes) : Indica les Classes Totals del projecte.
Day-Class: Temps necessari per a programar una classe.
EffortApplied : és la mètrica que indica l'esforç del projecte

Podrem veure aquesta i altres fórmules en l'apartat dedicat a COCOMO (3.2.2)

3.2.1.2 La mida i el concepte "SLOC"

La mida d'un projecte de programari és una mètrica que s'ha utilitzat durant anys per mesurar la productivitat. Es tracta d'una mètrica fàcil de comprovar a posteriori però de poca utilitat per a calcular els costos al iniciar un projecte, ja que, és improbable encertar a priori les línies de codi que tindrà un programa.

Es tracta d'una mètrica senzilla i objectiva adient pels llenguatges de programació en cascada dels anys 70 i 80 però no tant per a la programació moderna orientada a objectes que utilitzen els entorns d'escriptori.

3.2.1.2.1 Mètrica LOC

Qualsevol programari es pot mesurar pel número de línies de codi que conté (LOC: “Lines of Code”, SLOC: “Source Lines of Code) o de forma pràctica els milers de línies de codi (KLOC). Aquesta mètrica és una de les primeres en aparèixer i ha estat àmpliament utilitzada des de la seva aparició. Podem trobar defensors i detractors del seu ús, segons els avantatges i inconvenients que ofereix:

Avantatges:

- Fàcil de mesurar: el seu còmput es realitza comptabilitzant les línies de codi per instrucció del programa. No comptabilitzen els comentaris i una sola instrucció és una línia, independentment de com hagi estat representada (if-then-else, és una sola línia).

- Fàcil de combinar: Existeixen moltes mètriques derivades de la combinació de les LOC amb altres mètriques. Per exemple, una mètrica de qualitat, nombre d’errors per línies de codi.

- Dóna una mesura de la complexitat d’un programa: No es tracta d’una mètrica fiable, ja que depèn de la traça del programador i la optimització realitzada, però a grans trets si que dóna una idea de complexitat.

Inconvenients:

- Més línies de codi no equivalen a un millor producte: un programador amb experiència i un bon coneixement del llenguatge de programació que utilitza pot aconseguir les mateixes funcions en un menor nombre de línies de codi. Aquesta optimització de codi, que es en realitat un gran avantatge, podria aparèixer com un indicador negatiu, pel que no es fiable aquesta mesura, de forma aïllada per comprovar la productivitat del personal..

- Penalitza llenguatges d'alt nivell: els llenguatges d'alt nivell aconsegueixen amb menys línies de codi les mateixes o més funcionalitats que els llenguatges de baix nivell. Passa el mateix amb el ús de *frameworks* i llibreries, aconseguim reutilitzar codi i executar funcions amb menys instruccions. Per aquesta raó per a comparar la productivitat del programari, cal que el llenguatge de programació utilitzat sigui el mateix i resulta difícil extreure conclusions.

- La complexitat d'un programari no depèn directament de la seva mida. Els programes encastats que controlen maquinari industrial solen ser comparativament reduïts en nombre de línies però altament complexes pel que la mètrica LOC seria inadequada, de nou, per aquest escenari.

La mètrica LOC ens permet rebre una visió orientativa, però que per si sola no pot ser concloent. En realitat aquesta afirmació es compleix per a totes les mètriques, son mesures que cal analitzar per tal que serveixin d'ajuda en la planificació i anàlisi d'un projecte.

3.2.1.3 Esforç

Per a obtenir l'esforç necessari per a portar a terme un projecte, es calculen els costos que s'esdevindran en el desenvolupament del mateix.

Això equival a fer la suma del temps emprats per cadascun dels programadors multiplicat pel sou dels mateixos. Per altra banda, si volem obtenir una mesura més acurada caldrà afegir els costos derivats de l'aplicació del projecte, com ara viatges per la presentació del projecte i captació de requeriments, amortització de maquinari...

Les mètriques més utilitzades per aproximar els costos del projecte són: persona-hora, persona-mes, ja que normalment del sou del personal se'n deriven la majoria dels costos. En el apartat 4 es presenten algunes fórmules per calcular l'esforç mitjançant l'eina COCOMO

3.2.2 Mètrica de Punts Funció

Prevenir el cost d'un projecte al inici del mateix és una tasca complexa que no està correctament coberta per les mètriques basades en la mida, ni les línies de codi.

Per a millorar les estimacions de costos basades en les mètriques LOC cal un mètode que tingui en compte factors que es coneguin des de l'inici del projecte per tal de permetre una valoració correcte dels costos que se'n derivaran.

Es la idea dels **punts funció (FPA: Function Point Analysis)** Aquesta mètrica mesura la mida funcional del programari ponderant els requeriments del usuari i permet establir una aproximació de l'esforç que serà necessari per portar-lo a terme.

Aquest anàlisi per FP no considera ningun aspecte sobre la implementació de la solució, cap especificació interna sobre el llenguatge, metodologia

de programació... Es quantifica el cost de desenvolupar una determinada funció, tenint en compte les connotacions derivades de l'aplicació del requeriment.

Amb l'ús d'aquesta mètrica, dos professionals que analitzin un mateix projecte haurien d'arribar a la mateix conclusió.

Es tracta d'una mètrica de productivitat que ha assolit la terminologia d'estàndard avalada per la ISO/IEC 20926.

3.2.2.1 Breu història de la mètrica basada en Punts Funció:

1975 – IBM inventa les mètriques de punts funció per al seu ús intern: Allan Albrecht i els seus companys de la divisió de procés de dades reben l'encàrrec de desenvolupar una metodologia d'estimació que sigui efectiva per a diferents llenguatges de programació. Se'n adonen que els llenguatges de programació d'alt nivell són penalitzats per les mètriques LOC

1978 – IBM fa publica la metodologia de punts funció a la industria de programari en una conferencia a Monterey, Califòrnia.

1980 – Els punts funció s'utilitzen conjuntament amb les mètriques LOC. Aquest mètode rep el nom de *backfiring*, que consisteix en la conversió directa de LOC a punts funció. Per exemple en COBOL s'estableix que una funció te un mida de 105 SLOC i en Assemblador entre 250 i 350.

1985 – El primer programari d'estimació de costos apareix al Març de 1985, es tracta de SPR/20. D'aquesta manera l'ús de punts funció s'expandeix ràpidament.

1986 – La IFPUG crea una certificació per a les mètriques de punts funció, per tal d'establir unes regles consistents.

1987 – Les mètriques de punts funció són les preferides per l'anàlisi de qualitat per les seves qualitats de basar les estimacions en els requeriments d'usuari. S'estima que a qualsevol programari es troben al voltant de 5 errors per funció de mitja, amb una eficiència en l'eliminació dels errors del 85%, pel que 0.75 erros per funció són entregats com a versions finals.

1988 – Una de les qualitats dels punts funció és al de realitzar comparacions de qualitat i productivitat entre companyies, industries o països. Aquesta és una tendència a l'alça anomenada *benchmarking.*

1990 – Els punts de funció serveixen també per estimar el nombre d'eines d'anàlisi multidisciplinari que es fan servir en una companyia. El nombre d'eines que s'utilitzen per una determinada acció es mesuren també en punts funció, per estimar les necessitats de cada feina.

1991 – Mesurem les funcionalitats de qualsevol aplicatiu per punts funció. Així sabem que un programari de fulla de càlcul ens costa uns 0,25$ per punt funció, en canvi un programa especialitza en borsa costa uns 300$ per punt funció, això ens ajuda a estimar les qualitats dels diferents programaris segons les funcions que ens ofereixen i el preu de cada funció

1992 – Una vegada sabem que podem fer sevir els punts funció per quantificar una companyia sencera, el seu és passa a valorar industries senceres i països.

1997 – La preparació per a l'efecte 2000. Aquesta estimació de reenginyeria de codi, estimada en LOC, implicava que el client pagués per línies de codi sense aplicació, per un programari que ja havia adquirit prèviament. Al voltant d'un 30% del codi són comentaris o línies sense funció pel que els punts funció van ser importants per realitzar les estimacions de costos necessàries.

1998 – Els punts funció serveixen també per estimar el nou programari Web. La productivitat de programari web és més alta que la resta, probablement per que els errors en un lloc web salten a la vista de centenars o milers d'usuaris que visiten el lloc, així que poden ser corregits ràpidament.

3.2.2.2 Metodologia de punts funció

Primerament cal identificar si la funció a analitzar es tracta de:

- Una interacció amb l'usuari
    - Entrada externa (**EI** : External input)
        - Pantalles dedicades a l'ingrés de dades per part del usuari
    - Sortida externa (**EO** : External output)
        - Informes
        - Gràfics
        - Llistats de dades
    - Consulta externa (**EQ** : External query)
        - Funció de Cerca de dades
        - Recuperar y mostrar dades al usuari

- Tractament de dades

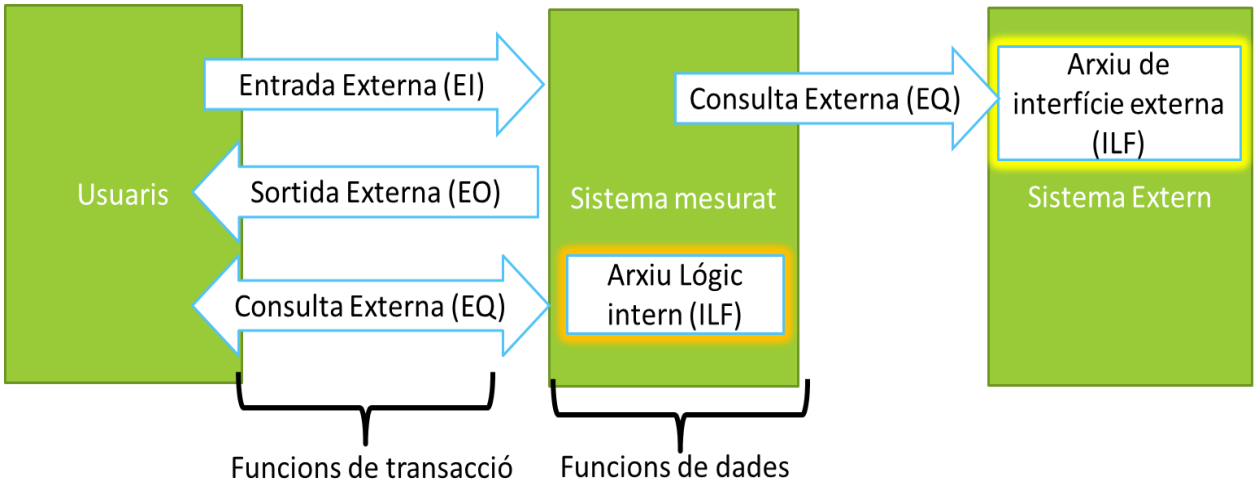
    - Arxiu intern lògic (**ILF** : Internal Logical File)

        - Arxiu des del punt de vista lògic
        - Taules de la base de dades

    - Arxiu d'interfície externa (**EIF** : External Interface File)

        - Dades referenciades en altres sistemes
        - Dades d'altres sistemes externs que es volen utilitzar en el sistema analitzat.

Diagrama de classificació de funcions segons la seva interacció amb el sistema mesurat:



Il·lustració 3

3.2.2.3 Classificació de funcions:

FUNCIÓ	CLASSIFICACIÓ
INSERCIÓ DE DADES	**EI**
ACTUALITZACIÓ DE DADES	**EI**
ELIMINACIÓ DE DADES	**EI**
CERCA DE DADES	**EQ**
GENERAR LLISTAT O INFORME	**EO**
TAULA DE BASE DE DADES	**ILF**

Taula 2

Una vegada que la funció s'identifica i es classifica en un dels esmentats tipus, cal avaluar la seva complexitat i assignar-hi un valor de punts funció.

Per aquesta finalitat el IFPUG (International Function Point Users Group) classifica els tipus de punts funció segons la seva complexitat i assigna els següents valors de FPI (Function Point Index).

Aquesta es la taula de classificació estandard que el IFPUG assigna a cada funció:

Classificació	Complexitat		
	Baixa	**Mitjana**	**Alta**
EI	3 FPI	4 FPI	6 FPI
EO	4 FPI	5 FPI	7 FPI
EQ	3 FPI	4 FPI	6 FPI
ILF	7 FPI	10 FPI	15 FPI
EIF	5 FPI	7 FPI	10 FPI

Taula 3

3.2.2.4 Avantatges del ús de punts funció

L'ús de mètriques de punts de funció millora les mètriques LOC pretén abordar diverses qüestions addicionals:

- Redueix la temptació dels desenvolupadors a generar més línies de codi per a ser, aparentment, més productius.

- És independent del llenguatge de programació i de la plataforma per a l que es desenvolupa.

- Es tracta d'una mesura consistent: Per a mesurar cal exactitud i concreció, calen regles clares per a que la mètrica assoleixi l'estàndard. És per això que els punts funció són àmpliament acceptats, ofereixen una classificació concreta i universal.

- És bona per a l'usuari. Es tracta d'una mètrica que el usuari pot entendre i valorar. Els seus requeriments són directament analitzats i això millora la percepció i *feedback* dels *stakeholders*.

3.2.2.5 Inconvenients del ús de punts funció

El principal inconvenient resideix en que aquest mètode no té en compte les funcions complexes com els algorismes, això és per que és molt difícil interpretar la dificultat que compren desenvolupar un algorisme en comparació a un altre, no existeix un mètode per analitzar-los de forma objectiva.

Per aquesta raó les aplicacions d'ús científic, on l'ús d'algorismes és més intensiva, són difícilment mesurables en termes de requeriments d'usuari i depenent de la seva complexitat, requereixen d'un mètode que tingui en compte aquestes característiques per estimar els costos.

Per altra banda, l'ús de la mateixa requereix d'una dedicació addicional en els projectes de desenvolupament, quant moltes vegades es tracta de pressupostos molt ajustats. En conseqüència l'ús del propi mètode és un cost en hores i dedicació suplementaris al cost del projecte.

3.2.2.6 Estàndards de productivitat (Normes ISO/IEC)

És per això que la mètrica de FP ha anat evolucionant i han aparegut versions més completes i també altres variants que tracten de millorar aquesta mètrica, algunes d'elles han esdevingut estàndard i disposen del seu marcatge ISO, són les següents:

- **IFPUG**: ISO / IEC 20926: 2009 : Es el primer estàndard sobre punts funció. El International Function Points Users Group, es una societat dels Estats Units sense ànim de lucre que mira d'instaurar i revisar les regles de FP per tal de que estiguin actualitzades a les noves metodologies i tendències de programació.

    - Disposa dels manuals d'ús CPM (Counting Practices Manual) versió 4.3 i el SNAPv 2.2, que s'encarrega de les mesures no funcionals.

    - El IFPUG també s'encarrega de regular les certificacions sobre punts funció (CFPS)

- **NESMA**: ISO / IEC 24570: 2005: És el estàndard Holandès (Netherlands Software Metrics Association) i es considera la segona associació dedicada al mesurament del programari més gran del món.

    - La seva proposta incorpora la distinció entre la creació d'una nova funcionalitat i la modificació d'una funcionalitat, que en el cas de IFPUG són comptabilitzades de la mateixa manera.
      Així es generen mètriques per mesurar el manteniment del programari.

    - Per a mesurar les modificacions s'utilitza el concepte *percentatge d'impacte* que considera: partint dels punts funció que es comptabilitzarien per una funcionalitat nova, un tant per cent del total segons la quantitat de modificacions que s'han de realitzar sobre la funció. Si la funcionalitat nova és un 100% i està comptada com a 10 FP, una modificació de la mateixa en un 40% tindrà un cost de 4 FP.

- **COSMIC**[1]: ISO / IEC 19761: 2011 : Common Software Metrics International Consortium. Anomenat també COSMIC-FFP, per les sigles *Full Function Points*. S'introdueix al 1997 per un equip de la Universitat de Quebec i esdevé estàndard ISO al 2003 en la versió 2.2:

    - És considerat el primer mètode funcional de mesurament de programari de segona generació

    - *Full Function Points*: Intenta cobrir les mancances dels FPA sobre els sistemes encastats i de temps real. On les funcions poden tenir una gran quantitat de *subprocessos* amb diverses funcions i aquestes no tenen cobertura en els FPA.

- **FISMA**: ISO / IEC 29881: 2008 : Finnish Software Measurement Association. És una associació sense ànim de lucre Finesa:

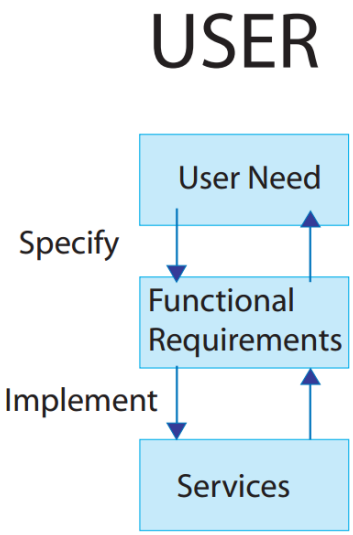
    - El mètode de FISMA es orientat al servei i no al procés, això significa que s'identifiquen cadascun dels diferents serveis d'un programa i no tots els seus processos funcionals.

    - *Base Functional Component* (BFC) és la unitat de FISMA, es tracta de Serveis per Funció.

    - *BFC type*, és una categoria definida per als BFC. Cada programa incorpora varis tipus de requeriments d'usuari.

    - *BFC class*, és un grup de BFC: Cada BFC representa només un tipus de *BFC type.*

1: Aquesta mètrica es tracta amb més profunditat al apartat "Tendències actuals en el ús de mètriques"



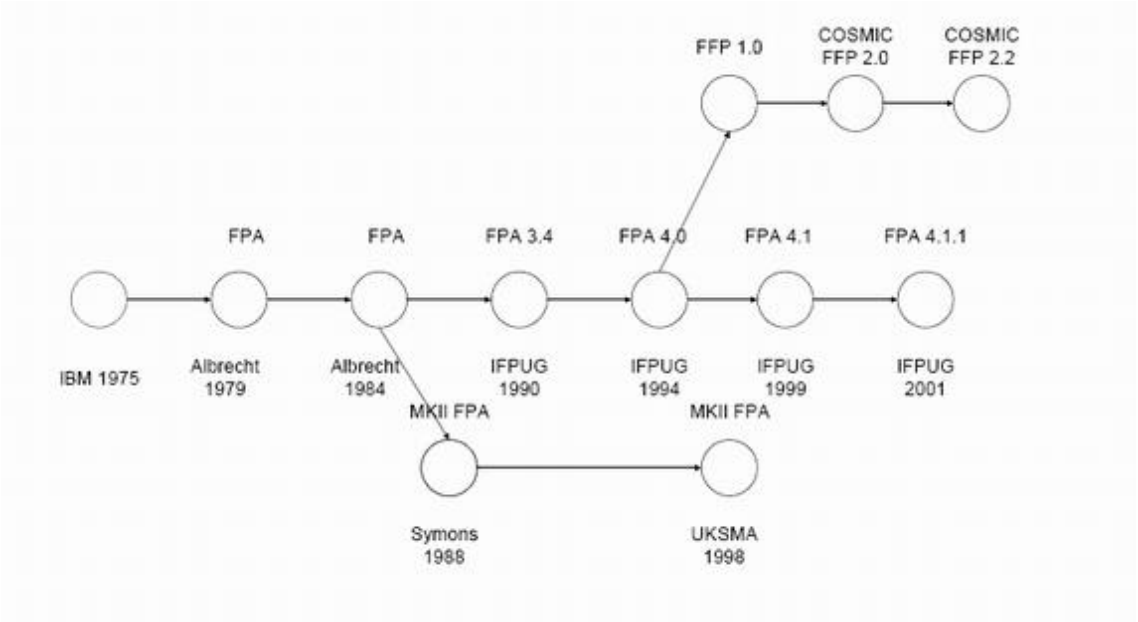
Il·lustració 4

Font: *http://www.fisma.fi/wp-content/uploads/2006/09/fisma_fsm_method_11.pdf*

- **MARK-II**: ISO / IEC 20968: 2002 : És el estàndard del Regne Unit, UKSMA (United Kingdom Software Metrics Association).

    - És una proposta de Charles R. Symmons al 1988 i la darrera versió apareguda és la 1.3.1 publicada al 1998.

    - Aquest mètode intenta fer un mesurament continua del procés de vida del desenvolupament del software. Davant del IFPUG que resulta més estàtic



Il·lustració 5

Evolució dels mètodes d'avaluació basats en els PF (http://www.monografias.com/trabajos55/estimacion-por-puntos-de-funcion/estimacion-por-puntos-de-funcion2.shtml)

3.2.2.7 Altres evolucions dels Punts Funció

3.2.2.7.1 Feature Points (Punts Característica):

Aquesta mètrica va ser introduïda per Caper Jones [Jones, 1987] afegeix canvis al FPA per millorar l'aplicabilitat a sistemes amb processament intern significatiu (per exemple, sistemes operatius, sistemes de comunicacions). Això permet tenir en compte les funcions que no són perceptibles per l'usuari, però essencials per al seu correcte funcionament.

Bàsicament inclou els conceptes estàndard dels punts funció però introdueix un nou paràmetre anomenat *algorismes,* que com el nom indica s'encarrega de valorar la càrrega algorítmica de la funció. Per aquesta finalitat s'intenta estandarditzar al manera de comptabilitzar la complexitat algorísmica segons la carrega matemàtica del mateix.

Segons el nombre d'algorismes i la complexitat dels mateixos es puntua el cost d'aquest paràmetre. Per defecte, un algorisme tindrà una puntuació de 3, si aquest inclou operacions amb matrius o càlcul d'equacions, la complexitat matemàtica s'eleva fins a un màxim de puntuació de 10, que seria la màxima complexitat. Es multiplica aquest valor pel nombre d'algorismes i es suma a la resta de paràmetres per obtenir el total de punts característica sense ajustar.

3.2.2.8 Backfiring

Aquesta recollida de dades és en un primer moment feixuga i costosa, degut a que si no s'ha treballat amb punts funció prèviament, no es disposa d'aquestes dades. Per ajudar en aquest procés existeix un mètode anomenat *backfiring* que permet convertir les línies de codi dels nostres programes en punts funció i al inrevés.

Aquest mètode introduït per Capers Jones al 1995 [Jones, 1995] té en compte el llenguatge i les línies de codi. Per a realitzar el càlcul s'utilitza una taula on estan representats la majoria de llenguatges de programació i la mitjana de LOC per FP segons una estadística sobre el llenguatge en qüestió. En la següent taula es recullen alguns llenguatges:

Llenguatge	Nombre de LOC per Punt Funció
Basic Assembly	575
C	225
FORTRAN	210
Pascal	160
C++	80
Java	80
Oracle	60
Perl	50
JavaScript	50
VB Script	50
Cobol II	175

Taula 4

El resultat és un mètode útil per començar a fer servir FP sense tenir dades sobre FP dels nostres projectes passats. Per altra banda, és palès que l'exactitud d'aquest mètode és molt precària, en concret algunes estimacions parlen d'una desviació d'entre el 100% i el 400% del real, pel que és poc recomanable el seu ús com a norma general.

4 Eines per al càlcul de mètriques de productivitat

4.1 L'eina d'estimació d'esforç COCOMO

Existeixen eines d'ajuda per a l'estimació de costos i productivitat, que han contribuït en bona mesura a l'evolució de les mètriques de programari. Una de força coneguda és COCOMO, que en la seva primera versió (COCOMO 81) utilitza com a unitat de mesura les línies de codi.

4.1.1 Algunes concrecions sobre la forma de contar les LOC:

- Només les LOC de la versió final són comptabilitzades (no es tenen en compte les proves).
- Es comptabilitzen les instruccions com una sola línia de codi: una instrucció *if-then-else* ocupa una sola línia.
- Les declaracions de variables es tenen en compte com a LOC.
- No es comptabilitzen els comentaris.

4.1.2 Categorització del software per a COCOMO 81:

El defineixen tres categories de desenvolupament en funció dels tipus de projectes:

- Orgànic: Projectes de mida petita o petita-mitjana. Menors de 50 KLOC, es considera que es té una experiència de projectes similar i es troba en entorns estables.
- Semi-acoplat: Projectes de mida mitjana en complexitat (menors de 300 KLOC), on la experiència en aquest tipus de projectes es variable i les restriccions ni molt altes ni molt baixes.
- Sistema encastat: Projectes molt complexos, amb poca experiència, on la innovació tècnica té un paper important. Sovint aquests projectes estan associats a la utilització d'un hardware concret i sofisticat.

4.1.3 Equacions bàsiques de COCOMO:

- L'esforç que requerirà el projecte, es calcula en unitats *persones-mes*:

$$\boldsymbol{Effort\ Applied}\ (\boldsymbol{E}) = \boldsymbol{a}(\boldsymbol{KLOC})^{\boldsymbol{b}} * \boldsymbol{m}(\boldsymbol{X})$$

  On:
  a,b són constants, els valors de les quals depèn del model
  KLOC són les línies de codi (en milers)
  m(X) és un multiplicador que depèn de 15 atributs

- El temps de desenvolupament. En funció del Esforç requerit calculat anteriorment. S'expressa en mesos:

$$\boldsymbol{Development\ Time}\ (\boldsymbol{DT}) = \boldsymbol{c}(\boldsymbol{E})^{\boldsymbol{d}}$$

  On:
  c,d són constants, els valors de les quals depèn del model
  E, és l'esforç calculat anteriorment

- Nombre de persones que requerirà el projecte:

$$\boldsymbol{People\ required} = \frac{\boldsymbol{E}}{\boldsymbol{DT}}$$

4.1.4 Models de precisió de COCOMO 81

Com hem vist les fórmules anteriors es basen en unes constants que venen definides pel model que s'està analitzant. Podem escollir la precisió amb la que volem realitzar els càlculs segons els següents models disponibles:

- Model Bàsic

S'utilitza per obtenir una primera aproximació de l'esforç. La següent taula indica el valor de les constants segons el tipus de projecte en el que estem:

TIPUS	a	b	C	d
Orgànic	2.40	1.05	2.50	0.38
Semi – Orgànic	3.00	1.12	2.50	0.35
Encastat	3.60	1.20	2.50	0.32

Taula 5

Podem comprovar com les constants a i b augmenten al augmentar la complexitat del tipus de projecte, això farà que la fórmula de l'esforç requerit, abans presentada, augmenti considerablement segons el tipus de projecte en el que estem.

- Model intermig

Si utilitzem aquesta opció, afegirem als càlculs anteriors 15 modificadors que tenen en compte diferents aspectes de l'entorn de treball, d'aquesta manera s'aconsegueix una major concreció en el resultat. A més els valors de les constants "a,b" abans presentats varien. Els valors vàlids per aquest model són els següents:

TIPUS	a	B
Orgànic	3.20	1.05
Semi - Orgànic	3.00	1.12
Encastat	2.80	1.20

Taula 6

A continuació cal quantificar els atributs segons la seva complexitat, per tal de calcular el modificador m(X) de la fórmula del esforç.

Els valors adients s'han de cercar en la següent taula:

Atributs	Valor					
	Molt baix	Baix	Nominal	Alt	Molt alt	Extra alt
Atributs de programari						
Fiabilitat	0,75	0,88	1,00	1,15	1,40	
Mida de Base de dades		0,94	1,00	1,08	1,16	
Complexitat	0,70	0,85	1,00	1,15	1,30	1,65
Atributs de maquinari						
Restriccions de temps d'execució			1,00	1,11	1,30	1,66
Restriccions de memòria virtual			1,00	1,06	1,21	1,56
Volatilitat de la màquina virtual		0,87	1,00	1,15	1,30	
Temps de resposta		0,87	1,00	1,07	1,15	
Atributs de personal						
Capacitat d'anàlisi	1,46	1,19	1,00	0,86	0,71	
Experiència en l'aplicació	1,29	1,13	1,00	0,91	0,82	
Qualitat dels programadors	1,42	1,17	1,00	0,86	0,70	
Experiència en la màquina virtual	1,21	1,10	1,00	0,90		
Experiència en el llenguatge	1,14	1,07	1,00	0,95		
Atributs del projecte						
Tècniques actualitzades de programació	1,24	1,10	1,00	0,91	0,82	
Utilització d'eines de programari	1,24	1,10	1,00	0,91	0,83	
Restriccions de temps de desenvolupament	1,22	1,08	1,00	1,04	1,10	

Taula 7

- Model detallat

El model detallat és bàsicament igual que l'intermig excepte que els factors corresponents als atributs són sensibles o dependents de la fase sobre la qual es realitzen les estimacions. Aspectes com ara l'experiència en l'aplicació, utilització d'eines de programari, etc., tenen més influència en unes fases que en altres, a més van variant d'una etapa a una altra.

4.2 COCOMO II

COCOMO II es va publicar l'any 1997 i és un model evolucionat de COCOMO que millora alguns aspectes de la primera versió i amplia les seves funcionalitats, per exemple, amb la incorporació de les mètriques de punts funció, encara que la mètrica LOC segueix present.

La majoria de les estimacions s'obtenen de la mateixa manera que COCOMO. Els principals canvis son en el nombre i tipus de paràmetres de costos i el càlcul de les variables de l'equació en lloc de la utilització de constants.

COCOMO II considera els següents models, que són diferents dels de la primera versió (COCOMO 81):

- Model *Application Composition:*

En una primera etapa del desenvolupament és necessari realitzar un prototipatge i plantejar tots els dubtes que es resoldran en les següents fases del cicle de vida d'un programa, aquest model dona suport a aquesta etapa.

- Model *Early Design*:

Aquest model realitza estimacions en les primeres etapes d'un disseny de projectes abans de disposar de totes les dades.
Aquest model és possible gràcies a la incorporació del ús de Punts Funció, com hem vist és un model més eficaç en la predicció de costos. COCOMO II redueix a 7, els factors de cost per aquesta etapa, ja que considera que no és possible conèixer la informació de la resta de factors en una estimació inicial d'un projecte de desenvolupament.

- Model *Post-Arquitectura*:

És el més detallat i s'utilitza una vegada és coneguda l'arquitectura general del projecte, quan estem en disposició de començar el desenvolupament del programari. En aquest punt podem utilitzar els punts de funció o LOC per a realitzar les estimacions. A més, disposem dels 17 factors de cost per utilitzar, que permeten una estimació dels costos més acurada.

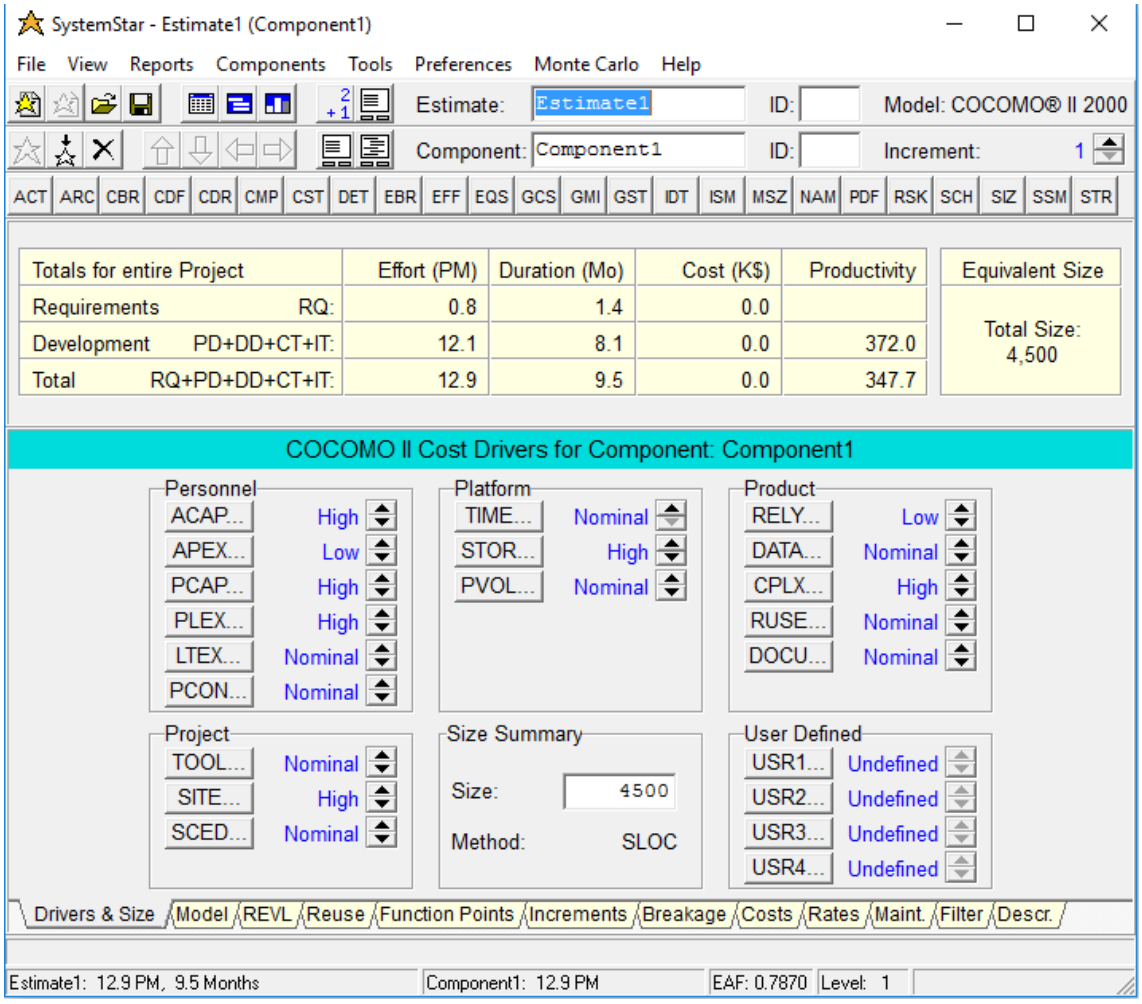
4.2.1 Factors de Cost a COCOMO II

Els factors de cost per COCOMO II es poden valorar en una escala de Molt Baix a Extra Alt de la mateixa manera que es feia a COCOMO 81.

Factors relatius al producte	
RELY	Fiabilitat
DADES	Mida base de dades
CPLX	Complexitat del producte
RUSE	Requeriment de Reutilització
DOCU	Documentació del cicle de vida
Factors relatius a la plataforma utilitzada	
TEMPS	Temps d'Execució
STOR	Emmagatzematge
PVOL	Volatilitat de la Plataforma
Factors relatius al Personal	
ACAP	Capacitat del Analista
PCAP	Capacitat del Programador
AEXP	Experiència en Aplicacions similars
PEXP	Experiència sobre la Plataforma
LTEX	Experiència en el Llenguatge i Eines emprades
PCON	Continuïtat del Personal
Factors relatius al Projecte	
EINA	Ús d'Eines de Software
LLOC	Desenvolupament conjunt en diferents emplaçaments
SCED	Desenvolupament del Cronograma

Taula 8: Llista de paràmetres de costos de COCOMO II (Boehm et al, 1995)

Exemple d'estimació de COCOMO II, fent servir una versió d'avaluació del programa d'estimació de costos SystemStar. Aquest programa incorpora varies mètriques per escollir entre les que està COCOMO II



Il·lustració 6

4.2.2 Calibratge de COCOMO II

Per a que un model COCOMO sigui el més exacte possible ha de ser calibrat. Aquesta pràctica s'aconsegueix alimentant la base de dades històriques.

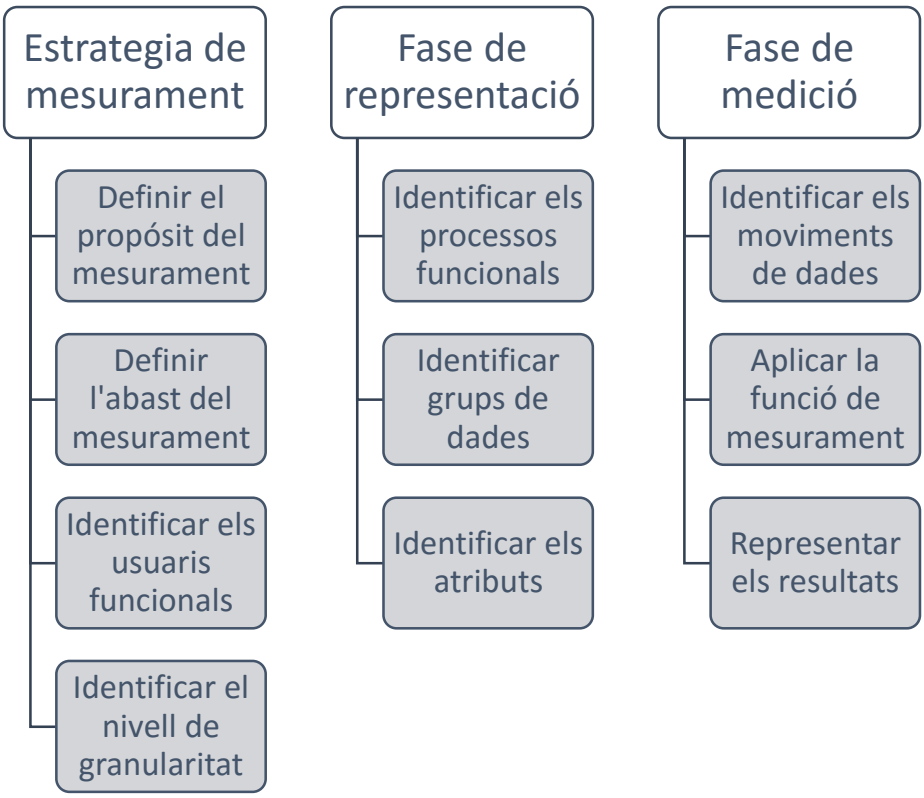
Els usuaris poden introduir les dades dels seus propis projectes per a ser utilitzades com a referència i aquesta és la millor forma de calibrar possible, ja que està basada en la forma de fer del propi usuari i els models que utilitza per als seus projectes, però és important que aquest procés de recollida de dades perduri amb el temps per tal d'augmentar encara més la precisió dels resultats de les estimacions en el futur.

5. Tendències actuals en l'ús de mètriques

Actualment (any 2015), disposem de gran quantitat de mètriques disponibles per utilitzar, tot i que només 5 d'elles han aconseguit l'estatus de norma ISO. Tot i que l'estàndard IFPUG segueix sent el més utilitzat a tot el món per ser el primer i per tenir una bona acceptació, COSMIC-FFP, ha esdevingut molt important darrerament per les seves característiques úniques.

5.1 COSMIC-FFP

COSMIC és una metodologia estructurada en 3 fases. Cal aplicar-les totes per a portar a terme un mesurament:



Il·lustració 7

5.1.1 Estratègia de mesurament

5.1.1.1 Definir el propòsit del mesurament

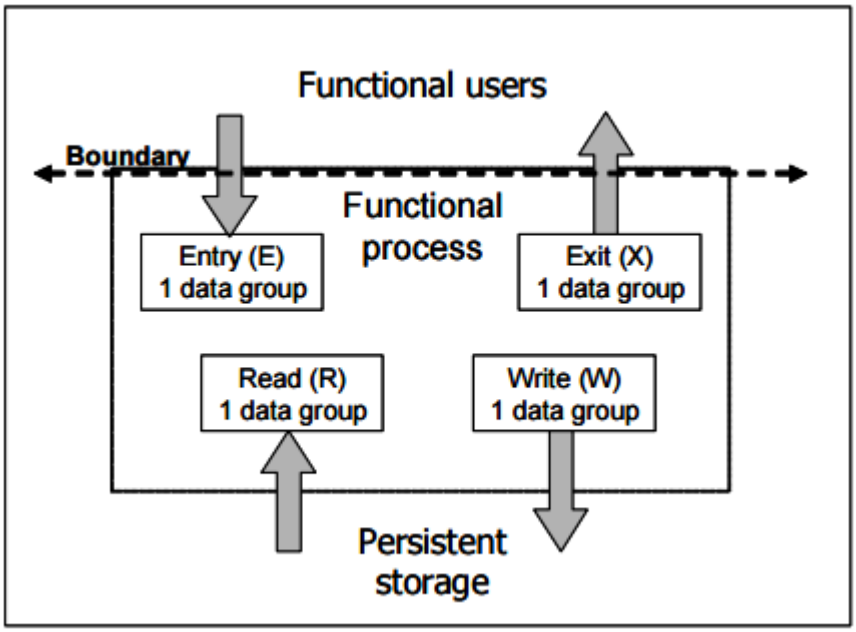
Es tracta de definir el perquè del mesurament i per a que utilitzarem el resultat.

5.1.1.2 Definir l'abast del mesurament

COSMIC permet definir limitacions en l'abast de l'amidament, fent servir una classificació del entorn per **capes** (*layers*). Això és especialment útil en sistemes de temps-real (Real Time Software), on es disparen processos i *subprocessos* de forma síncrona i independent. Aquestes capes es separen per **fronteres** que determinen els límits del entorn a mesurar.

5.1.1.3 Identificar els usuaris funcionals

Pot ser qualsevol emissor o receptor de dades: una persona, un software extern (com a IFPUG) o en el cas de COSMIC, un element hardware que interactua amb la capa analitzada.



Il·lustració 8

Font: http://www.cosmicon.com/portal/public/Spanish_COSMIC_%20v3-0-1.pdf

5.1.1.4 Identificar el nivell de *granularitat*

És el grau de precisió de la informació que disposem dels requeriments del usuari. Normalment a mesura que avança el procés de desenvolupament el nivell de *granularitat* dels requeriments millora, per que s'estableixen més clarament els objectius.

5.1.2 Fase de representació

5.1.2.1 Identificar els processos funcionals

Les entitats externes i internes de IFPUG, passen a denominar-se processos funcionals i poden contenir 1 o més requeriments funcionals del usuari.

La diferència principal amb les entitats (EI, EO, EQ) de IFPUG és que a COSMIC un procés funcional pot tenir associades vàries operacions de dades d'entrada i sortida, fet que a IFPUG no és possible, degut a que cada entitats va lligada per definició a una sola operació de dades.

5.1.2.2 Identificar els grups de dades

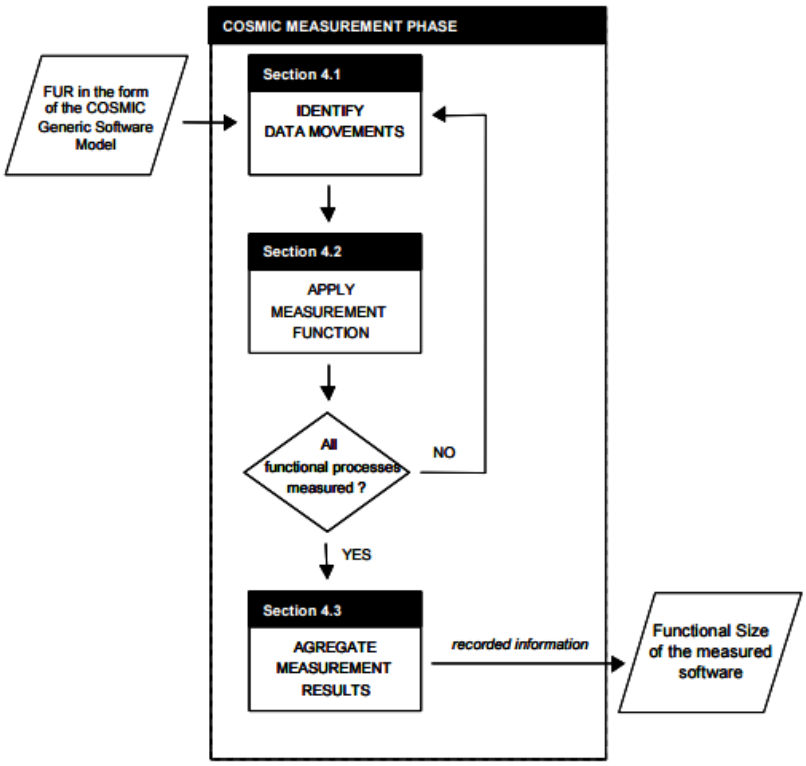
A COSMIC, la complexitat d'una funcionalitat no va lligada al nombre de d'atributs que es veuen afectats si no que depenen de la mida de les dades que es maneguen i el nombre d'operacions que es porten a terme (són els anomenats grups de dades). Aquestes operacions d'entrada i sortida de dades es mesuren com a 1 CFP (Cosmic Function Point).

5.1.2.3 Identificar atributs

Per aquesta raó la representació dels atributs deixa de ser imprescindible, però cal tenir-la en compte quan la representació d'un nou atribut modifica la funcionalitat que s'està descrivint.

5.1.2.4 Fase de mesurament

El mètode general per a mesurar una aplicació per a la mètrica COSMIC es pot resumir en següent quadre:



Il·lustració 9

Font: http://www.cosmicon.com/portal/public/Spanish_COSMIC_%20v3-0-1.pdf

5.1.2.5 Identificar els moviments de dades

Els moviments d'un grup de dades poden constar de diferents operacions classificades com a :

- entrada (E)
- sortida (X)
- lectura (R)
- escriptura (W)

Un procés funcional ha de contenir almenys entrada i sortida, però tantes lectures i escriptures com calgui, pel que podem trobar, per exemple, un procés que inclogui :

- ➔ 1 entrada
- ➔ 3 lectures
- ➔ 2 escriptures
- ➔ 1 sortida

5.1.2.6 Puntuació de les funcions

La quantitat de CFP (Cosmic Function Points) assignats a cada procés funcional no està limitada com en el cas de IFPUG, on cada funcionalitat ha de ser classificada segons una escala de valors, per exemple, per les entrades (EI) podem assignar 3,4 o 6 FP segons la dificultat de l'operació.

5.1.2.7 Aplicar la funció de mesurament

Cal valorar amb 1 CFP cadascuna de les operacions que es fa amb un grup de dades (E, X, R, W).

Una vegada identificats els moviments de dades de cada procés funcional, calcular el seu valor és senzill, ja que consisteix en sumar les operacions de cada tipus per a obtenir el total.

Les següents fórmules defineixen els càlculs de COSMIC:

- **Mesura del procés funcional**

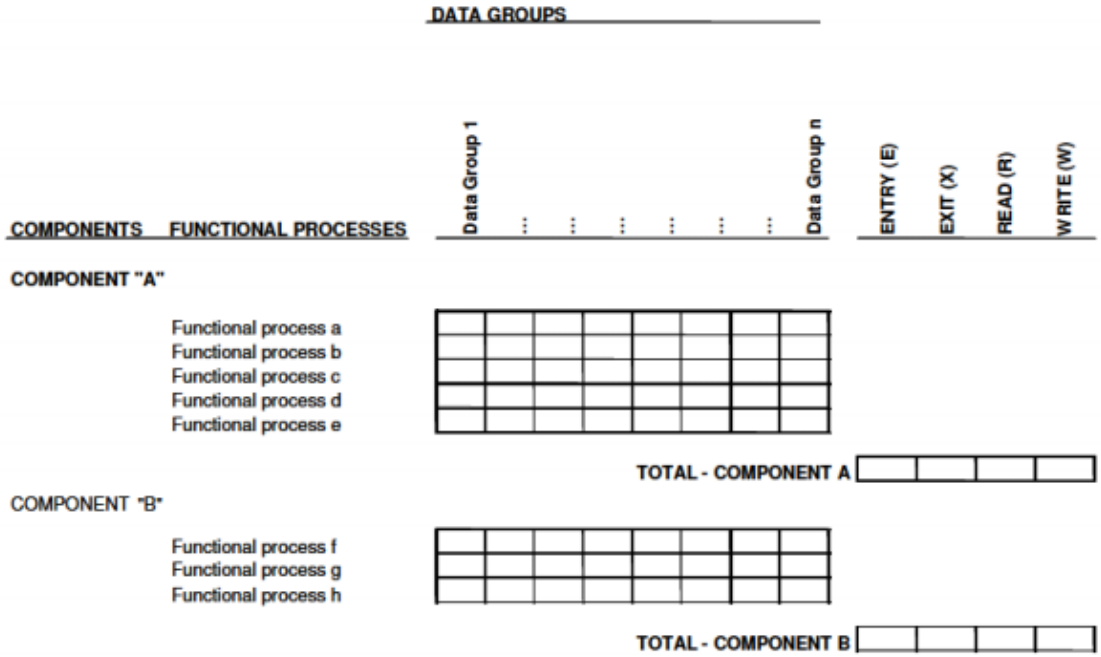
$$Mida\ total = \sum E + \sum S + \sum R + \sum W$$

- **Mesura del canvi d'un requeriment d'usuari**

$$Mida\ del\ canvi = \sum(mov.dades\ afegides) + \sum(mov.dades\ modificades) + \sum(mov.dades\ eliminades)$$

5.1.2.8 Representar els resultats del mesurament

Una vegada calculada la mida de cada procés funcional per a cada grup de dades cal representar les mesures per a trobar un nombre únic, per a cada capa i finalment una mesura global del producte. Una proposta de la representació de les dades seria la següent:



Il·lustració 10

Font: http://www.cosmicon.com/portal/public/Spanish_COSMIC_%20v3-0-1.pdf

6 Mètriques de qualitat

No existeix el programari sense errors, per definició és així i està acceptat que sempre apareixen petits errors difícils de detectar i que majoritàriament provoquen problemes assumibles.

És molt difícil lliurar un programari sense defectes, perquè el temps per aconseguir-ho el faria inviable econòmicament, pel que, al avaluar un programari, el que interessa és que la seva qualitat sigui l'adequada per a la satisfacció del client.

Quan es finalitza un programa i s'entrega al client, es poden produir dues situacions possibles:

A. Si el producte conté un nombre d'errors reduït, aquests suposaran un cost addicional pel projecte que ha d'estar contemplat en la seva planificació i no hauria de suposar un problema.

B. Si el producte conté un gran nombre d'errors, probablement els costos per reparar els problemes superin la planificació inicial i el producte pot esdevenir inviable.

Fent ús de mètriques de qualitat podem mesurar els punts forts del nostre producte i identificar aquells que requereixen una millora. poden suposar la rendibilitat d'un producte, de tota la feina feta. Cal detectar el major nombre d'errors possible abans de donar la feina per acabada.

6.1 Els models de qualitat

Un model de qualitat del programari és un conjunt de bones pràctiques per al cicle de vida del programari. Normalment els models aporten una llista de comprovació que es pot fer servir per revisar la qualitat del programari.

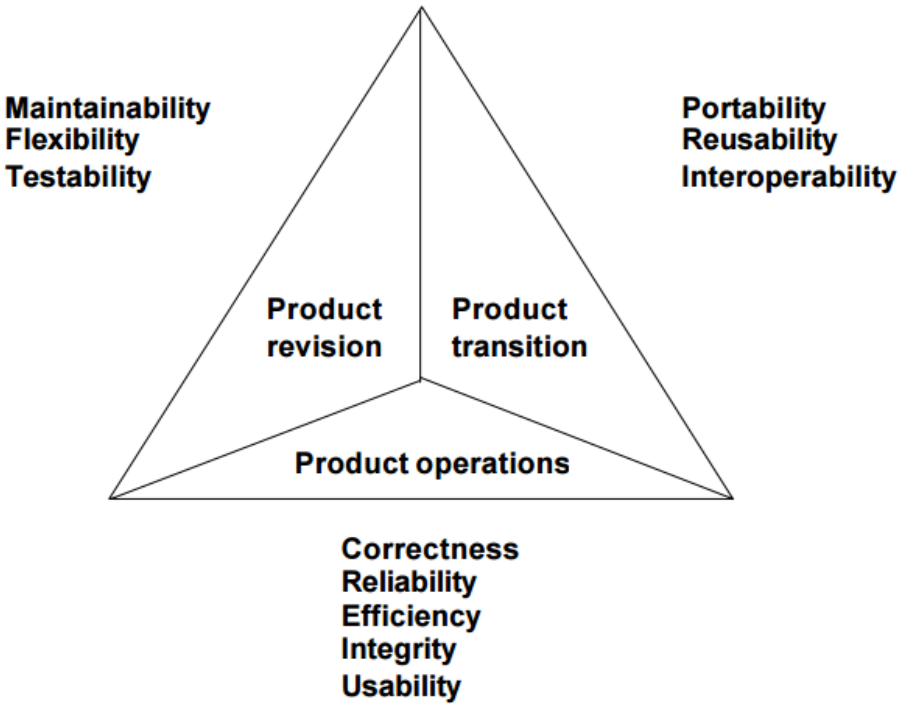
6.1.1 Model MCCALL

Jim McCall va produir aquest model per a la Força Aèria dels Estats Units al 1976 i la intenció era apropar la perspectiva dels usuaris i desenvolupadors. En el seu model va identificar tres perspectives principals per la categorització d'un producte. Aquestes perspectives són:

1 - Revisió del producte
2 - Transició Producte
3 - Operacions de producte

Cada perspectiva inclou factors de qualitat (11 factors), per a cada factor de qualitat McCall defineix varis criteris de qualitat (41 criteris en total), així doncs, l'avaluació de la qualitat d'un producte depèn finalment de la ponderació d'aquests criteris, per aquest motiu, el càlcul de les mètriques resulta molt subjectiu.

A continuació veurem la descripció de les perspectives amb els corresponents criteris d'avaluació:



Il·lustració 11

Font: http://www.csse.monash.edu.au/courseware/cse3308/cse3308_2005/assets/McCall_Checklist.pdf

6.1.1.1 Revisió del producte

La perspectiva de revisió del producte identifica factors de qualitat que influeixen en la capacitat de canviar el producte de programari, aquests factors són:

Factor	Descripció	Criteris
Manteniment	Esforç requerit per localitzar i corregir errors	Consistència Simplicitat Consistència Auto-descripció Modularitat
Flexibilitat	Facilitat per a realitzar canvis	Expansibilitat Generalitat Auto-descripció Modularitat
Incorporació de proves	Facilitat per a realitzar les proves, per assegurar que el producte no té errors i compleix amb l'especificació	Simplicitat Instrumentació

Taula 9

6.1.1.2 Transició del Producte

La perspectiva de transició producte identifica factors de qualitat que influeixen en la capacitat d'adaptar el programari a nous entorns:

Factor	Descripció	Criteris
Portabilitat	Esforç requerit per a transferir entre diferents ambients d'operació	>Auto-descripció >Modularitat >Independència del maquinari >Independència del sistema operatiu
Reusabilitat	Facilitat per a reutilitzar el programa en diferent context	>Generalització >Modularitat >Auto-descripció >Independència del sistema operatiu
Interoperabilitat	Esforç necessari per a connectar el producte amb un altre programari.	>Modularitat >Interoperabilitat en dades i connectivitat

Taula 10

6.1.1.3 Operacions del producte

L'operació del producte inclou els següents factors de qualitat i criteris :

Factor	Descripció	Criteris
Correctesa	Grau en què el producte compleix amb la seva especificació	Traçabilitat Completesa Consistència
Fiabilitat	Habilitat del producte de respondre davant de situacions inesperades	>Tolerància a errors >Temps de recuperació del error >Probabilitat d'error >Simplicitat
Eficiència	Optimització del ús dels recursos del sistema	>Eficiència en temps d'execució >Eficiència en l'ús d'emmagatzemament
Integritat	Protecció del programa i les seves dades d'accessos no autoritzats	Control d'accés Auditoria de dades
Usabilitat	Facilitat d'operació del producte per part dels usuaris	Operativitat Entrenament Comunicació Volum d'E / S Taxa d'E / S

Taula 11

6.1.1.4 Inconvenients dels model MCCALL

El problema principal d'aquest model és que no defineix els requisits funcionals. Per altra banda alguns dels criteris de qualitat no són fàcilment definibles, sobretot en les primeres etapes d'estimació del projecte

6.1.2 Model de Qualitat de Boehm

Barry W. Boehm va introduir un model de qualitat al 1978, que de forma similar al model de McCall intenta definir la qualitat del programari com un conjunt d'atributs i mètriques distribuïts en una estructura jeràrquica.

En el nivell més alt del seu model, Boehm defineix tres usos d'Alt nivell, que representen requisits primaris del programari. Aquests tres usos principals són:

- La utilitat, la mesura en què l'estat en què està el programari es pot utilitzar (és a dir, la facilitat d'ús, fiabilitat i eficiència).

- Mantenibilitat, la facilitat d'identificació del que ha de ser canviat, així com la facilitat de modificació i tornar a provar.

- Portabilitat, la facilitat de canviar el programari per adaptar-se a un nou entorn.

Aquests tres usos principals tenen factors de qualitat associats que representen el **nivell intermedi** en el model jeràrquic de Boehm.

- Portabilitat, el grau en què el programari treballarà sota diferents configuracions d'equip (és a dir, sistemes operatius, bases de dades, etc.).

- La fiabilitat, la mesura que el programari realitza segons sigui necessari, és a dir, l'absència de defectes.

- L'eficiència, l'ús òptim dels recursos del sistema durant l'execució correcta.

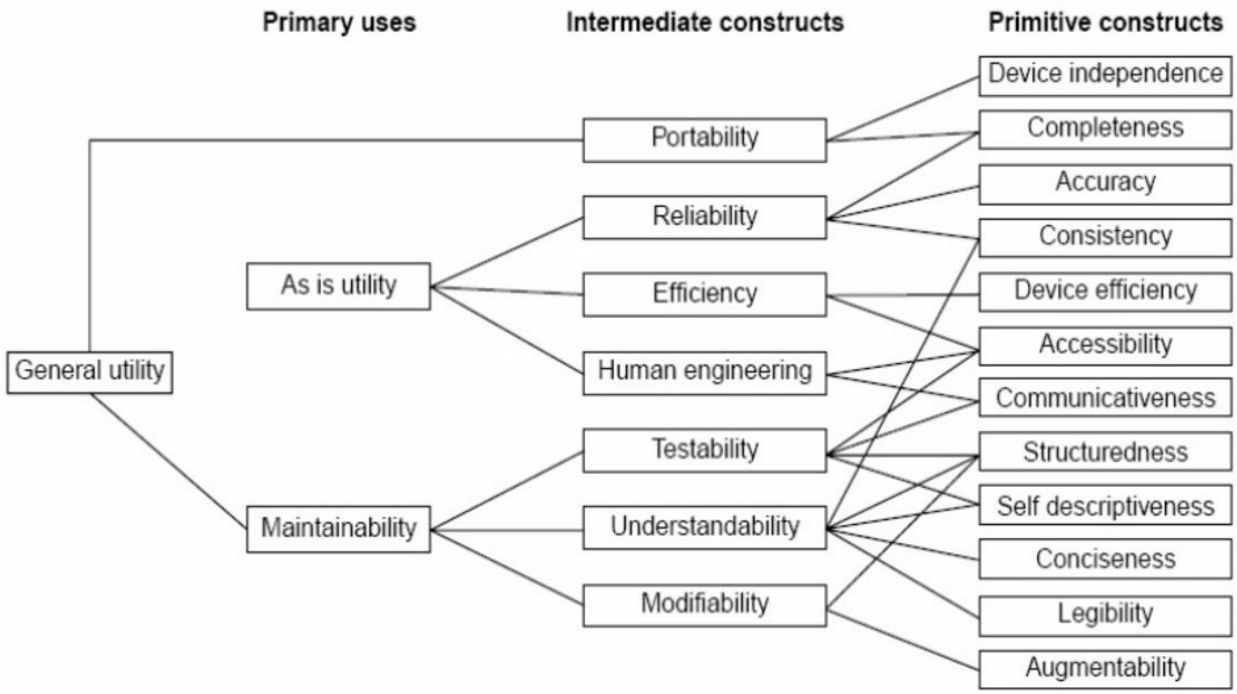
- Usabilitat, la facilitat d'ús.

- Capacitat de proves, facilitat de validació, que el programari compleix amb els requisits.

- Comprensibilitat, la mesura que el programari és fàcilment comprès pel que fa al propòsit i l'estructura.

- La flexibilitat, la facilitat de canviar el programari per complir amb els requisits revisats.

Aquests factors de qualitat es desglossen en construccions primitives que permeten mesurar el nivell de qualitat del programa, són criteris de qualitat semblants als de McCall. A continuació es representa un quadre resum d'aquests criteris, juntament amb les relacions jeràrquiques dels tres nivells:



Il·lustració 12

Font: http://www.slideshare.net/abhinavtheneo/chapter-7-software-reliability

6.1.3 FURPS+

Es tracta d'un model de qualitat desenvolupat per Robert Grady a Hewlett-Packard al 1987. El model FURPS+ estableix cinc característiques com a factors de qualitat que són les sigles del acrònim que li dóna nom.

La següent taula introdueix els factor de qualitat de FURPS i alguns atributs d'exemple:

Factor de qualitat	**Atributs**
Functionality (Funcionalitat)	Característiques i capacitats del programa Generalitat de les funcions Seguretat del sistema
Usability (Usabilitat)	Factors humans Factors d'estètica Consistència Documentació

Reliability (Fiabilitat)	Freqüència i severitat dels errors Precisió dels resultats Temps mitjà d'errors Capacitat de recuperació davant d'un error. Capacitat de predicció
Performance (Rendiment)	Velocitat de processament Temps de resposta Consum de recursos Rendiment efectiu total Eficàcia
Supportability (Suport)	Extensibilitat Adaptabilitat Capacitat de les proves Manteniment Compatibilitat Requisits de instal·lació

Taula 12

- A continuació descriurem amb més detall el significat cada element:

  Funcionalitat:
  Descriuen què és el que un usuari ha de poder fer al utilitzar el sistema. Des del punt de vista dels requeriments d'usuari els requisits funcionals representen generalment les principals característiques del producte.

- La resta de requisits que representa FURPS+ són *no-funcionals*:

  Usabilitat:
  La facilitat d'ús inclou tots aquells atributs que faciliten la interacció d'un usuari amb el sistema.

  Fiabilitat:
  Agrupa els requeriments que tenen a veure amb la solidesa i robustesa d'un sistema durant la seva execució.

  Rendiment:
  Es refereix a la velocitat del sistema i la seva eficiència en utilització de recursos

  Suport:
  Inclouen requisits d'instal·lació i configuració, així com facilitats per mantenir i administrar l'operació del sistema.

- **El signe "+" de FURPS+**

Són requeriments addicionals que acostumen a ser restriccions o limitacions:

Restriccions de disseny	Limiten les opcions de disseny del sistema, com per exemple, que cal fer ús d'una base de dades relacional
Restriccions d'implementació	Es refereix a les regles per a la programació, com la utilització especifica d'un llenguatge, o el protocol d'un estàndard
Restriccions d'interfície	Indica els elements externs amb el qual un sistema ha d'interactuar, com limitacions en els formats o altres factors
Restriccions físiques	Especifica una restricció física del maquinari per a que funcioni el programa

Taula 13

6.2 Estàndards de qualitat

6.2.1 ISO 9000

La ISO és el organisme encarregat de buscar estàndards de normes per a diversitat de productes y trobar els processos que garanteixin la qualitat i seguretat de les fabricacions, com diu la Wikipèdia:

“La Organització Internacional de Normalització o ISO és l'organisme encarregat de promoure el desenvolupament de normes internacionals de fabricació (tant de productes com de serveis), comerç i comunicació per a totes les branques industrials” (Varis autors, 2015).

De la diversitat d’estàndards existents la norma ISO 9000 és probablement la més coneguda, en concret la ISO 9126 tracta la qualitat del producte de programari:

ISO 9000	Sistemes de Gestió de la Qualitat - Fonaments i vocabulari
ISO 9001	Sistemes de Gestió de la Qualitat - Requisits
ISO 9004	Sistemes de Gestió de la Qualitat - Directrius per a la millora del compliment
ISO/IEC 9126	**Qualitat del Producte de Programari** • **ISO / IEC 9126-1 Part 1: Model de Qualitat** • **ISO / IEC 9126-2 Part 2: Mètriques Externes** • **ISO / IEC 9126-3 Part 3: Mètriques Internes** • **ISO / IEC 9126-4 Part 4: Mètriques de Qualitat en Ús**

Taula 14

Font: https://es.wikipedia.org/wiki/Anexo:Normas_ISO#ISO_1000.E2.80.93ISO_9999

6.2.2 ISO 9126

La norma ISO 9126 data de 1991 i va ser revisada al 2001 convertint-la en quatre parts que es presenten a continuació. Recentment ha estat substituïda per la norma ISO 25010:2011.

Tracta de definir un model per mesurar la qualitat d’un projecte de desenvolupament de programari.

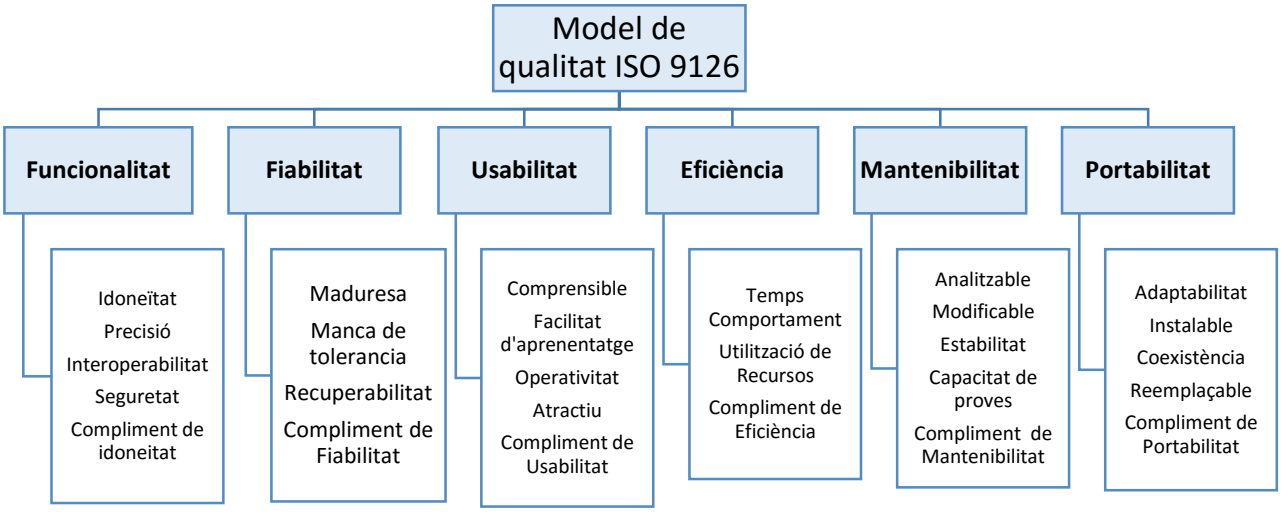
La ISO 9126:2001, es divideix en 4 parts:

- Model de qualitat
- Mètriques externes
- Mètriques internes
- Mètriques de qualitat en l'ús.

La ISO 9126 distingeix entre errors i no conformitat, una fallada comporta el no compliment dels requisits previs, mentre que la no conformitat contempla els requisits especificats. Una distinció similar és feta entre la validació i la verificació.

6.2.2.1 Model de qualitat

El model de qualitat classifica el programari en un conjunt de factors i atributs que s'atribueixen al programari per tal de mesurar-lo. Aquesta classificació és una variant dels models que s'han vist anteriorment, com el de McCall i Bohem.



Il·lustració 13

6.2.2.2 Mètriques de qualitat externes

Són aquelles que mesuren el comportament de tot el programari o part d'ell, a través de proves i observacions del programari executable en el sistema. Les mètriques externes permeten comprovar si el programa desenvolupat o en procés compleix amb la qualitat necessària durant la fase de proves.

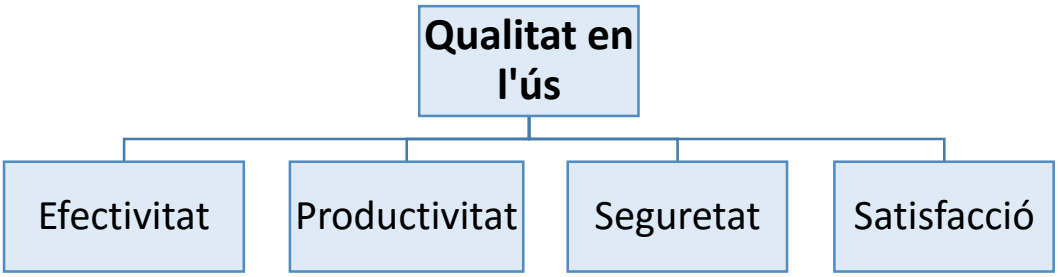
6.2.2.3 Mètriques de qualitat internes

Són les mètriques que poden ser aplicades durant el disseny i la codificació del programari no executable (per exemple codi font). Aquestes mètriques permeten mesurar la qualitat del programa mentre s'està desenvolupant per a prendre les correccions adequades en cas de ser necessari

6.2.2.4 Mètriques de qualitat d'ús

Mesura si el producte compleix amb les necessitats descrites pels usuaris en els seus requeriments. Aquesta avaluació del programari es realitza en escenaris específics per tal de comprovar si els usuaris han assolit els seus objectius.

El model de qualitat en l'ús indica unes característiques que contenen atributs que són mesurables, però en aquest cas no s'especifica quins són aquests atributs ja que es considera que depenen del producte desenvolupat i seran diferents segons el llenguatge utilitzat i la seva complexitat:



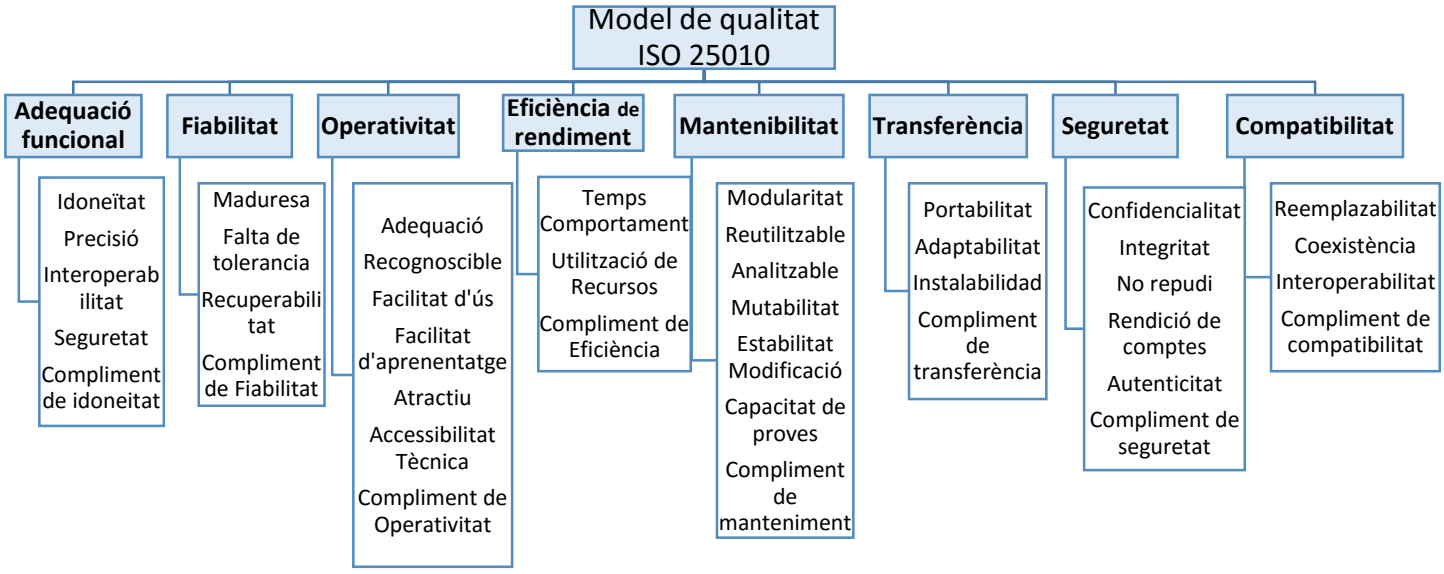
Il·lustració 14

6.2.3 ISO 25010

El 2005 es crea una nova versió d'aquesta norma, la ISO / IEC 25010. Una guia per a l'ús dels nous estàndards internacionals anomenats Requisits i Avaluació de Qualitat de Processos de Software (Square). S'estableixen criteris per a l'especificació de requisits de qualitat del programari, mesures i avaluació. Actualment la darrera versió actualitzada és de 2011.

El nou model té vuit característiques en lloc de sis, que són molt similars al ISO 9126. S'han afegit 2 noves característiques principals que són la seguretat i la compatibilitat. Quant als atributs, alguns s'han afegit i altres han vist el seu nom modificat per trobar un terme més precís per a la seva descripció.

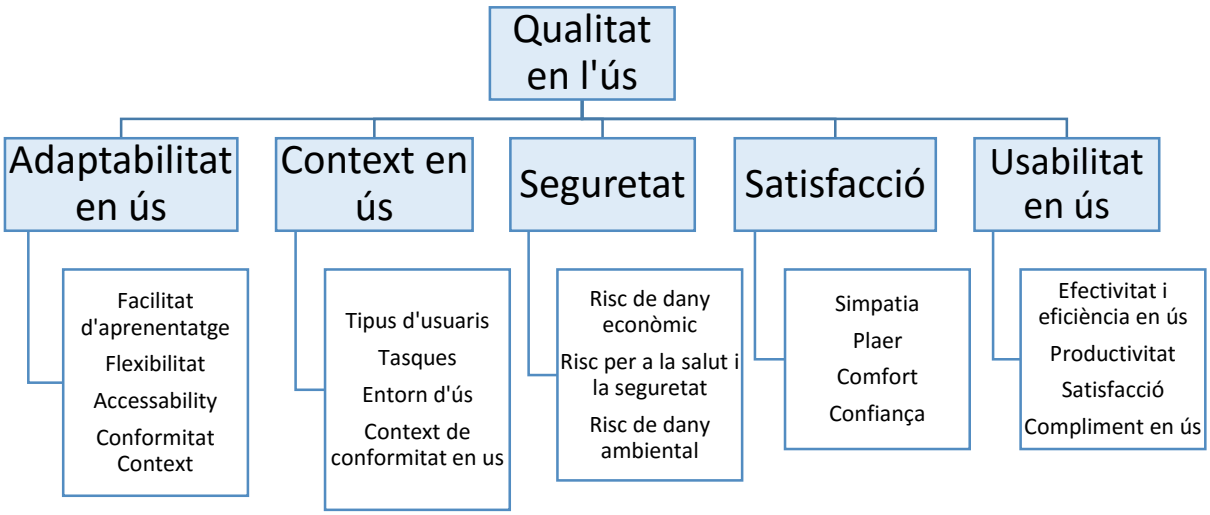
La norma ISO 25010 hereta les mètriques internes i externes de qualitat que ja s'havien introduït en les darreres versions de ISO9126:



Il·lustració 15

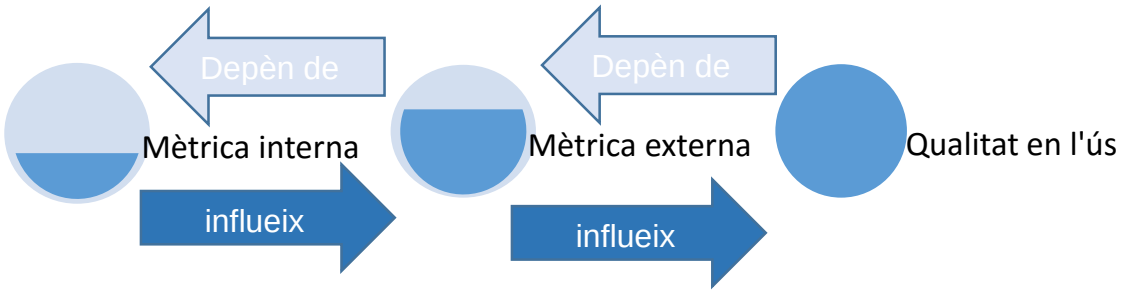
6.2.3.1 Qualitat en l'ús

El model qualitat de programari en ús també es veu modificat. La productivitat i eficiència formen part de la usabilitat i s'introdueixen els nous factors de adaptabilitat en ús i context:



Il·lustració 16

Per aconseguir la qualitat en ús, cal assolir primer la qualitat externa, que depèn a la seva vegada de la qualitat interna. Per garantir una bona mesura cal assolir els 3 nivells de qualitat.



Il·lustració 17

7 Conclusions

Per concloure, cal destacar la quantitat d'informació teòrica existent sobre les mètriques de productivitat, així com de qualitat, encara que en menor mesura. Es tracta d'un camp molt treballat, on s'hi han dedicat grans esforços, tot i així els estàndards resultants d'aquest estudi són pocs i molt exigents.

Es tracta d'una disciplina que requereix d'una bona base teòrica per desenvolupar els seus conceptes en la pràctica. Aquesta necessitat d'estudi i manca d'immediatesa en la seva aplicació, fa que per a projectes petits i petit-mitjans estigui poc estesa la seva utilització, d'altra banda, les millores que s'obtenen en la planificació d'un projecte a mig i llarg terme, incentiven l'ús de mètriques d'una manera més global per aconseguir una millor productivitat i qualitat del programari.

7.1 Comentaris sobre la memòria

Quant a la planificació inicial, he hagut de realitzar algunes modificacions. La data d'entrega de la PAC2, que va ser entregada 3 dies després de la data inicialment planificada per fets aliens al desenvolupament de la mateixa, que van obligar a refer tota la feina feta des de l'inici.

En un principi es volia introduir una aportació pràctica de la feina que es realitza en una empresa real, que desenvolupa programari en un entorn *agile*, pel que semblava atractiu incorporar la forma de planificar i mesurar la productivitat del programari que es realitza, però la realitat ha estat que aquesta empresa no utilitza mètriques de productivitat, o si més no, la forma de mesurar no és estàndard ni part de cap model pel que no semblava adient la seva inclusió.

A mode de comentari, en la cerca de informació sobre les normes ISO, desconeixia el fet de que la documentació s'ha de pagar per a descarregar. Potser sempre ha estat així però és quelcom que no havia tingut en compte pel fer de tractar-se d'estàndards.

Quant al contingut, ha quedat pendent d'analitzar més a fons una base de dades de mètriques com el ISBG (http://www.isbsg.org/), o un altre entitat dedicada al enregistrament de mètriques de projectes. Aquestes mètriques s'utilitzen com a punt de partida per avaluar altres projectes similars que no disposen de dades històriques de productivitat.

8 Glossari

IFPUG: International Function Point Users Group

COSMIC: Common Software Measurement International Consortium

FISMA: Finnish Software Measurement Association

NESMA: Netherlands Software Metrics Association

UKSMA: United Kingdom Software Metrics Association

CFP: Cosmic Function Point

FP: Function Point

FPA: Function Point Analysis

ISO: International Standard Organization

COCOMO: Constructive Cost Model

FURPS: Functionality Usability Reliability Performance Supportability

9 Bibliografia

Llibres:

- Articles de nombrosos autors: Capers Jones, Michael Mah, Howard Rubin, Lori Holmes...*IT Measurement (Practical Advice from de Experts)*, Addison Wesley. International Function Point Users Group.
- Project management institute, *A guide to the Project Management Body of Knowledge (PMBOK Guide),* Global Standard
- Miquel Barceló García i Joan Antoni Pastor i Collado, *Gestió d'organitzacions i projectes informàtics*, UOC

Internet:

A través de la Biblioteca de UOC (catàleg.uoc.edu)
Articles online:
http://www.hindawi.com/journals/ase/2010/307391/

Pfleeger, S. L. (2008). Software metrics: Progress after 25 years? *IEEE Software, 25*(6), 32-34. doi:http://dx.doi.org/10.1109/MS.2008.160
http://0-search.proquest.com.cataleg.uoc.edu/docview/215838329?pq-origsite=summon

Relacio metriques-indicadors
http://www.ciw.cl/recursos/Charla_Metricas_Indicadores.pdf

Mesures per a POO. Metodologies per mesurar la complexitat:
http://0-search.proquest.com.cataleg.uoc.edu/docview/232577877/3C6A6D2718F44CD3PQ/10?accountid=15299

Mètriques per a metodologies Agile:
http://searchsoftwarequality.techtarget.com/guides/Quality-metrics-A-guide-to-measuring-software-quality:

Wikipedia:
https://es.wikipedia.org/wiki/Calidad_de_software
https://es.wikipedia.org/wiki/COCOMO
https://en.wikipedia.org/wiki/Function_point

Consultoria especialitzada y documents:
http://fattocs.com/es/recursos/articulos.html

Punts funció a com eina de valoració de software:
http://searchsoftwarequality.techtarget.com/guides/Quality-metrics-A-guide-to-measuring-software-quality

Punts funció:
http://www.monografias.com/trabajos55/estimacion-por-puntos-de-funcion/estimacion-por-puntos-de-funcion2.shtml

Ampliació Mida y LOC
http://calidadysoftware.blogspot.com.es/2011/09/metricas-loc.html

Ampliació Punts funció
https://www.youtube.com/watch?v=0wbALQ9lz7o
http://www.monografias.com/trabajos55/estimacion-por-puntos-de-funcion/estimacion-por-puntos-de-funcion2.shtml
http://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf

COCOMO2 :
http://www.computing.dcu.ie/~renaat/ca421/report.html
http://www.utdallas.edu/~John.Cole/CoCoMo2.pdf

Base de dades de mètriques ISBSG:
http://www.isbsg.org/
http://www.isbsg.org/ISBSGnew.nsf/WebPages/011EB5EC65FC0562CA2574740018F1D7?open

Punts funcio vs COSMIC:
http://sunset.usc.edu/events/2003/Presentations/Evolving%20Standards%20in%20Function%20Point.pdf

De IFPUG a COSMIC:
http://www.leda-mc.com/descargas/de_ifpug_a_cosmic.pdf
http://www.cosmicon.com/portal/public/Spanish_COSMIC_%20v3-0-1.pdf

Mètriques per a la qualitat del software:
http://www.ecured.cu/index.php/Metricas_para_la_calidad_del_software
http://www.oei.eui.upm.es/Asignaturas/PInformaticos/ficheros/transparencias/TEMA_2.pdf

Presentació sobre la qualitat:
http://es.slideshare.net/isisparada/metricas-de-calidad-de-software?related=1

Eines per a utilitzar mètriques de qualitat de programari:
http://www.javiergarzas.com/herramientas-software-recomendadas

MCCAL i BOEHM
http://www.csse.monash.edu.au/courseware/cse3308/cse3308_2005/assets/McCall_Checklist.pdf
http://www.sqa.net/softwarequalityattributes.html
http://www.academia.edu/4899160/Different_Software_Quality_Model

Punt de vista de Microsoft
https://msdn.microsoft.com/es-es/lilidad brary/bb385914.aspxv

FURPS+:
http://www.ibm.com/developerworks/rational/library/3975.html

ISO 9126:
https://es.wikipedia.org/wiki/ISO/IEC_9126
http://ocw.upm.es/lenguajes-y-sistemas-informaticos/software-quality/contenido/slides/Unit%204.%20SoftwareQualityFeatures_Models.pdf
Part2:
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=22750
Part3:
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=22891
Part4:
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39752

ISO 25010
http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733
http://ryreitsma.blogspot.com.es/2011/07/software-has-new-quality-model-iso.html
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35733