Mètriques de productivitat de programari per a la gestió de projectes

Sergio Barquero Duque
ETIG

Jesus Bustinduy Basterrechea

10/01/2005

Resum

En aquest document s'introdueixen els conceptes bàsics necessaris per a l'execució de mètriques de productivitat de programari, després de la introducció, es passa a estudiar amb mes detall les diverses mètriques de productivitat mes utilitzades actualment, que son línies de codi (orientada al tamany del projecte), punts de funció (orientada a la funcionalitat del projecte, específica per a projectes de gestió), punts de característica (similar a punts de funció, però mes genèrica i útil per a altres tipus de projectes) i punts de cassos d'us (també orientada a la funció i específica per a projectes d'orientació a objectes). S'explica com aconseguir a partir d'aquestes mètriques amb ajuda de models d'estimació de productivitat, com el model COCOMO, les estimacions de l'esforç necessari per desenvolupar un projecte de programari i la distribució de l'esforç a totes les etapes del projecte a partir de les estimacions de la fase de desenvolupament. També es tracta, encara que no amb tanta profunditat la mètrica bang i l'estàndard proposat per l'IEEE per a aplicar mètriques de productivitat. Finalment es donen les pautes per a desenvolupar e implantar un pla de mètriques propi.

Resumen

(Métricas de productividad de software para la gestión de proyectos)

En este documento se introducen los conceptos básicos necesarios para la ejecución de métricas de productividad de software, después de la introducción, se pasa a estudiar con mas detalle las diversas métricas de productividad mas utilizadas actualmente, que son líneas de código (orientada al tamaño del proyecto), puntos de función (orientada a la funcionalidad del proyecto, específica para proyectos de gestión), puntos de característica (similar a puntos de función, pero mas genérica i útil pera otros tipos de proyectos) y puntos de casos de uso (también orientada a la función y específica para proyectos de orientación a objetos). Se explica como conseguir a partir de estas métricas con ayuda de modelos de estimación de productividad, como el modelo COCOMO II, las estimaciones del esfuerzo necesario para desarrollar un proyecto de software y la distribución del esfuerzo en todas las etapas del proyecto a partir de las estimaciones de la fase de desarrollo. También se trata, aun que no con tanta profundidad la métrica bang y el estándar propuesto por el IEEE para aplicar métricas de productividad. Finalmente se dan las pautas para desarrollar e implantar un plan de métricas propio.

Summary

(Productivity software metrics for the project management)

This document introduces the basic concepts necessary get for the execution of software productivity metrics, after the introduction we start to study in more detail the most used productivity metrics nowadays, this are code lines (oriented to the project size), function points (oriented to the functional project size, management projects especific), caracteristical points (similar to function points, but more standard and useful for other kind of projects) and use case points (especifically oriented to the oriented object projects). This project explains how to achieve basing in that metrics and helped by stimation models (like COCOMO II) the necessary effort estimations to develop a software project and the effort distribution in all the project phases based in the development phase stimations. This projecs does a light splanation about bang metric and the IEEE standard to apply productivity metrics. Finally exposes the basic guides to develop and implant an own metrics plan.

Índex de continguts

Índex de figures

Índex de taules

1 Introducció

1.1 Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC.

La societat moderna s'ha tornat cada vegada mes depenent del programari. Podem trobar els programes a molts aspectes de la vida quotidiana i a la pràctica totalitat de processos empresarials. El programari és un producte de consum en el qual cada vegada s'inverteix més per part d'empreses e institucions per poder oferir més i millors serveis o per reduir els costos operatius i de gestió.

Degut a la gran quantitat de recursos que s'inverteixen actualment en el programari, és molt important poder tenir el màxim control del producte que es vol obtenir. Segons la revista "Digital Planet" la inversió en tecnologia ha canviat el seu objectiu del maquinari al programari, provocant que la relació entre els dos aspectes, hagi passat d'un 32,5% al 1996 a un 40% en 2000 i amb una clara tendència a créixer, de manera que actualment la relació pot estar per sobre del 50%.

Una de les característiques del programari és la seva intangibilitat, propietat que el fa molt difícil de mesurar. De fet existeixen molts exemples de projectes que han fracassat degut a una estimació dolenta de les necessitats. Per aquest motiu podem dir que Mesurar la productivitat dels projectes de programari és molt important per a poder gestionar-los correctament, ja que s'obtenen dades quantitatives molt importants per a l'anàlisi i la millora de la productivitat o com explica el següent corol·lari "no es pot predir el que no es pot mesurar".

Moltes empreses (sobretot les dedicades a produir software) conscients d'aquesta necessitat han buscat els seus propis mètodes per poder obtenir dades concretes de la productivitat dels recursos dedicats a l'execució dels projectes de programari i d'aquesta manera poder millorar la gestió de projectes futurs.

Donada la importància de les mètriques, organismes com "IEEE" al 1992 i "ISO" al 2002 han definit els seus estàndards de mètriques de productivitat.

Aquest TFC pretén aportar una base per a qualsevol que estigui interessat en conèixer com aplicar les mètriques de producció mes utilitzades actualment, per aconseguir millores en la productivitat i una major precisió a l'hora de fer estimacions sobre l'esforç necessari per al desenvolupament dels projectes de programari.

1.2 Objectius

Els objectius d'aquest treball son els següents:

Definir de manera genèrica les mètriques de programari i específicament les mètriques de productivitat de programari.

- Enumerar i explicar els factors mes importants que s'han de tenir en compte a l'hora de dissenyar o aplicar una mètrica.

- Explicar els diferents àmbits d'aplicació de les mètriques dins del desenvolupament del projecte.
- Investigar les diferents mètriques de productivitat de programari utilitzades actualment.
- Dins de les mètriques de productivitat estudiar les que son considerades estàndard per ISO i IEEE
- Donar les pautes a seguir per a dissenyar una mètrica pròpia.

En definitiva realitzar un estudi exhaustiu de les mètriques de productivitat de programari incloent-hi els estàndards utilitzats actualment.

1.3 Enfocament i mètode seguit.

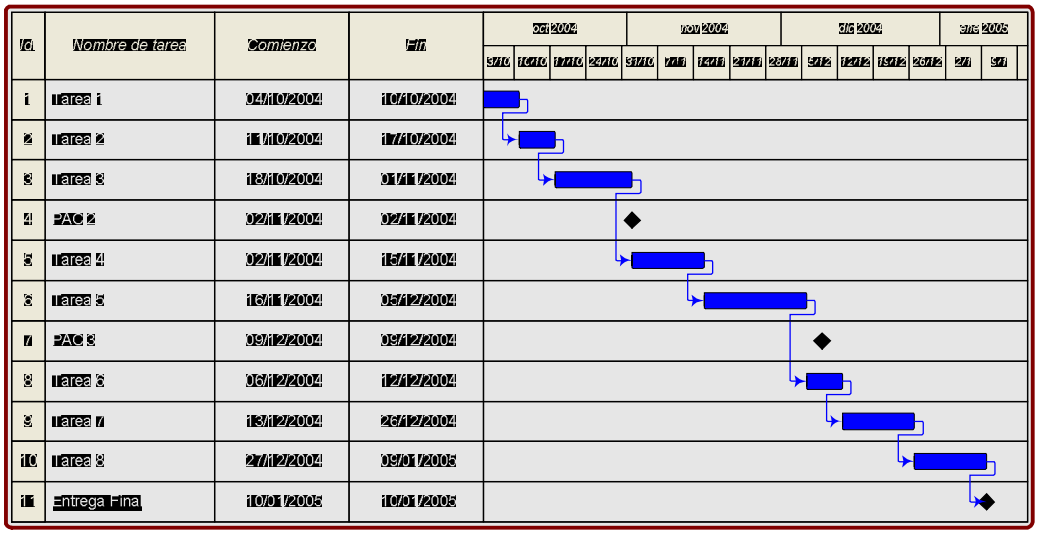
Per al desenvolupament del projecte, s'ha optat per un model en cascada que marca en cada moment quines accions o tasques s'han de realitzar, aquestes tasques estan definides a la planificació del projecte.
En optar per un model en cascada, totes les activitats han de ser seguides en l'ordre en que estan exposades, encara que algunes d'elles poguessin ser paral·lelitzades.

1.4 Planificació del projecte.

Tenint en compte la importància del TFC i el temps que se li ha de dedicar, faré l'esforç de dedicar una estona cada dia entre el 4 d'octubre (dia que començaré el desenvolupament del projecte) i el 10 de gener (Entrega del TFC), per tant el calendari seran tots els dies hàbils des del 4 d'octubre fins al 10 de gener, la descomposició de tasques es la següent:

Id.	Tasca	Duració	Precedents
1	Redacció primer capítol objectius, antecedents, abast.	1set.	
2	Cerca d'informació sobre definició de mètriques, conceptes bàsics i les seves característiques.	1 set.	1
3	Desenvolupar el tema i exposar-lo a un capítol.	2 set.	2
4	Cerca d'informació sobre les mètriques de productivitat utilitzades actualment	2 set.	3
5	Analitzar, processar i desenvolupar la informació sobre les mètriques de productivitat.	3 set.	4
6	Relacionar els paràmetres mes utilitzats per als mesuraments (el tamany i la funcionalitat)	1 set.	5
7	A partir de tota la documentació i el desenvolupament realitzat, donar les pautes necessàries per dissenyar una mètrica pròpia.	2 set.	6
8	Revisió de tota la documentació, realització de la presentació i entrega	2 set.	7

Per a acomplir el plaç del projecte, es proposa el següent diagrama de Gantt:



Les fites de control del projecte son les del pla docent del TFC les dades proposades son:

Fites de control		
Títol	Contingut	Data
Lliurament Pla de Treball	Pla de treball	27.09.2004
PAC2	Tenir finalitzat fins al capítol 5 del projecte	02.11.2004
PAC3	Lliurament Capítol 6 al 8.	09.12.2004
Lliurament Final	Memòria i presentació	10.01.2005

1.5 Productes obtinguts.

El producte obtingut d'aquest projecte serà la memòria en si i una presentació virtual que sintetitzi el contingut de la memòria.

1.6 Descripció de la memòria.

Després d'una breu definició de conceptes i de l'explicació dels principals factors que influeixen a les mètriques (Capítol 2), es passa al Capítol 3 on es parla dels diferents àmbits d'aplicació de les mètriques i la seva importància dins d'un pla global de mètriques per passar al capítol 4 a explicar en profunditat les tècniques de Línies de codi (LDC) per a mesurar el tamany del programari i les diferents mètriques per a mesurar el tamany funcional del programari, com son Punts de funció, punts de característica o punts de cassos d'us. Dins d'aquest apartat també es parla dels estàndards existents actualment (ISO e IEEE). Al capítol 5 es donen les pautes per a fer estimacions de l'esforç a partir del tamany funcional estimat i per a estimar la distribució de l'esforç a tot el projecte a partir de l'esforç estimat de l'etapa de desenvolupament.
El capítol 6 pretén ser una guia per a establir un pla de mètriques orientades a la productivitat.
Finalment s'ha inclòs un Glossari de Termes i Acrònims, les Referències Bibliogràfiques i un annex que introdueix al lector en un dels models d'estimació mes utilitzats actualment i indicat a la memòria per a fer les estimacions de l'esforç a partir del tamany, el model COCOMO II

2 Conceptes bàsics

2.1 Mesures

Una mesura proporciona una indicació quantitativa d'extensió, quantitat, dimensions, capacitat y tamany d'alguns atributs d'un procés o producte. Aplicat al programari una mesura és la quantitat de línies de codi que te un programa.

L'acte de determinar una mesura és la medició, i per a realitzar-les s'han d'establir els mètodes i les pautes que indiquin de manera unívoca com s'obtindrà la mesura que estem determinant. En el cas anterior s'haurà d'establir que es considera una línia de codi i com s'han de contar.

Existeixen mesures directes i mesures indirectes. La mesura directa d'un atribut és aquella que no depèn de cap altre atribut, en canvi la mesura indirecta és aquella en la qual està involucrada la medició de un o mes atributs.

2.2 Mètriques

Segons L'IEE, una mètrica és una mesura quantitativa del grau en que un sistema, component o procés posseeix un atribut donat.

Les mesures no serveixen per a comparar, per això necessitem les mètriques, que permeten relacionar y comparar mesures.

L'objectiu principal de les mètriques és ajudar a l'enginyer a establir una manera sistemàtica i objectiva d'aconseguir una visió interna de la seva feina i com a resultat millorar en algun aspecte els resultats.

2.3 Indicadors

Mètrica o combinació de mètriques que proporcionen una visió profunda, del procés de programari, del projecte de programari o del producte en sí (Ragland, 1995). Per tant les mètriques son els fonaments dels indicadors.

- Els **indicadors de procés** permeten tenir una visió mes profunda de l'eficàcia d'un procés ja existent. Es recopilen de tots els projectes de l'organització durant un període de temps llarg amb l'objectiu d'obtenir millores dels processos de software a llarg termini. Podem dir que son una recopilació dels indicadors de projecte.

- Els **indicadors de projecte** permeten:
  - Avaluar l'estat del projecte en curs.
  - Seguir la pista dels riscos potencials.
  - Detectar arrees de problemes abans que siguin crítiques.
  - Ajustar el flux y les tasques del treball.
  - Avaluar l'habilitat de l'equip del projecte en controlar la qualitat dels productes.

Els **indicadors de producte** permeten avaluar la seva qualitat.

2.4 Atributs interns i atributs externs

Els atributs interns d'un producte, procés o recurs son aquells que podem mesurar purament en termes del producte, procés o recurs en si mateix. Poden ser mesurats directament, per exemple: la longitud d'un programa o el temps transcorregut durant la realització de qualsevol document relacionat amb el programari.

Els atributs externs d'un producte, procés o recurs son aquells que només es poden mesurar respecte a com el producte, procés o recurs es relaciona amb el seu ambient. Aquests atributs solen ser els mes interessants de mesurar i predir per als gestors. Per exemple als gestors de projectes de programari els pot interessar conèixer el cost d'eficàcia en alguns processos o de la productivitat del seu personal. Desgraciadament els atributs externs son els mes complicats de mesurar, ja que la seva medició no es pot fer directament. La següent taula defineix i dona exemples d'alguns tipus d'atributs.

Entitats	**Atributs**	
Recursos	**Interns**	**Externs**
Personal	Edat, sou,..	Productivitat, experiència, Intel·ligència ...
Equip	Tamany, estructures ..	Productivitat, qualitat ..
Programari	Preu, tamany ..	Usabilitat, seguritat ..
Maquinari	Preu, velocitat, memòria	Seguretat, ..
Instal·lacions	Tamany, temperatura, llum,..	Confort, qualitat, ..
...	...	...
Processos		
Construcció d'especificacions	Temps, esforç, número de canvis ..	Qualitat, cost, estabilitat ..
Disseny detallat	Temps, esforç ..	Cost, cost efectiu
proves	Número d'errors trobats, temps	Estabilitat, cost, cost efectiu ..
...	...	...
Productes		
Especificacions	Tamany, modularitat, Funcionalitat ..	Comprensibilitat, manteniment ..
Disseny	Acoblament, ..	Qualitat, complexitat, ..
Codi	Funcionalitat, complexitat ..	Seguretat, manteniment ..
Dades de prova	Tamany, nivell de protecció	Qualitat ..
...	...	...

Tabla 1: Atributs Inerns i externs

2.5 Factors a tenir en compte

Existeixen factors inherents al desenvolupament del programari que poden tenir un impacte significatiu en la productivitat i que per tant s'han de tenir en compte a l'hora de dissenyar les mètriques a aplicar, aquests factors es divideixen en tres categories (projecte, administració i producte).

2.5.1 Del projecte

Dins de l'àmbit del projecte els dos factors que tenen un major impacte en la productivitat són el personal i l'ambient de desenvolupament.

En quant al **personal**, hem de tenir en compte el nivell educatiu i la seva formació, l'experiència laboral dins de l'empresa i en les diferents tecnologies amb les que hagi de tractar, la rotació del personal a l'empresa, la quantitat de personal de l'empresa i la disponibilitat d'experts externs si fos necessària.

En quant a **l'ambient de desenvolupament**, els factors que mes poden incidir sobre la productivitat son la disponibilitat d'eines adequades no només per a la programació en si, si no també per a la documentació, la generació de versions i instal·lables, la gestió documental del projecte, l'automatització de les proves i la depuració del codi. També es molt important que existeixin estàndards o manuals d'estil sobre la manera de codificar i documentar, per a aconseguir que tots els participants en el projecte treballin igual i puguin entendre perfectament els uns la feina dels altres.

2.5.2 De l'administració

Pel que fa a l'administració hem de tenir en compte el nivell de participació i implicació de l'usuari en el projecte, l'estabilitat dels requeriments i els factors que delimiten el projecte que poden ser els costos fixes, els recursos fixes, limitacions de funcionalitat o limitacions de temps.

2.5.3 Del producte

Del producte hem de tenir molt en compte els aspectes mes crítics pel client, com poden ser el temps de desenvolupament o posada en marxa, el nivell de qualitat o les possibilitats d'integració i escalabilitat, també hem de tenir en compte el grau d'innovació (no s'ha fet mai, ho han fet altres abans o ja ho hem fet nosaltres abans) i la complexitat, tant pel que respecta a la programació, el model de dades, la complexitat de la organització del client i de les tecnologies a utilitzar.

3 Mètriques del procés y del projecte

3.1 Mètriques del procés i millores en el procés de software.

Les mètriques del procés de software proporcionen indicadors del procés, que son els que permeten finalment aconseguir millores en el procés gràcies al seu anàlisi.

Les mètriques dedicades a estudiar la productivitat del procés del software son molt importants, ja que com a resultat de la seva aplicació és pot aconseguir una millora general de la productivitat i el rendiment de l'organització.

Com ja s'ha vist, existeixen molts factors que influeixen directa o indirectament en la productivitat del desenvolupament del software con poden ser el personal, l'entorn de desenvolupament o la complexitat de la tecnologia a utilitzar. La raó per la qual la millora de la productivitat del procés és tan important ,és per que aquest és un factor clau i controlable, mentre que el personal o l'entorn de desenvolupament son mes variables que no pas el procés i per tant els beneficis de la millora d'aquest seran mes rendibles per a l'organització i durant mes temps.

La única forma de millorar qualsevol procés és mesurar els atributs del procés, desenvolupant mètriques per a proporcionar indicadors que conduiran cap a estratègies de millora, en la figura 5.1 , es mostra el procés, situat en el centre d'un triangle connectat per tres factors amb una gran influencia no només en quant a la productivitat, si no també en quant a la qualitat, finalment aquest triangle es troba envoltat per aspectes que de l'entorn del procés que també influeixen sobre la productivitat, com son les característiques del client, les condicions del negoci i l'entorn de desenvolupament.

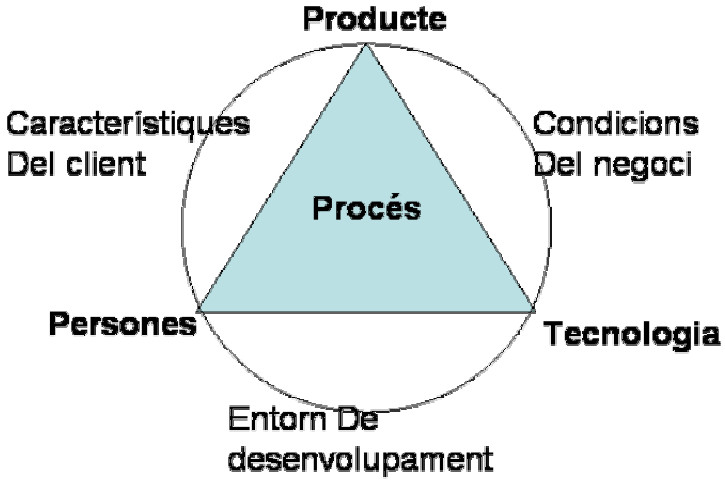


Figura 3-a: El Procés [Presuman 98]

Els atributs del procés, es poden dividir en públics i privats, les mètriques privades serien els indexes de defectes i els errors de desenvolupament, aquestes dades haurien de ser

privades per a l'individu i servir només com a indicador d'aquest individu i per ajudar a millorar la seva feina individual. Les mètriques públiques per a l'equip serien els indexes de defectes en funcions desenvolupades per l'equip, els errors generals de desenvolupament i les línies de codi o els punts de funció. Aquestes mètriques han d'ajudar a millorar el rendiment i la productivitat general de l'equip.

Les mètriques del procés poden ser molt útils, però s'han de saber interpretar, Robert U. Charette suggereix les següents normes bàsiques:

- Utilitzar el sentit comú i una sensibilitat organitzativa a l'hora d'interpretar les dades de les mètriques.
- Proporcionar una realimentació regular a les persones i equips que han treballat en la recol·lecció de dades.
- No utilitzar mètriques per avaluar a particulars.
- Establir objectius clars i les mètriques necessàries per aconseguir-los.
- No utilitzar mètriques que representin una amenaça per a usuaris o equips.
- Si una mètrica identifica una àrea problemàtica no s'hauria de considerar com a negativa. Es merament un indicador per a la millorar del procés.
- Totes les mètriques s'han d'interpretar com a conjunt, no ha de prevaldre cap en particular.

La utilització de mètriques e indicadors fiables dona lloc a una millora estadística del procés del programari basant-se en l'anàlisi de varis projectes de programari per millorar la productivitat i qualitat globals.

3.2 Mètriques del projecte

Les mètriques del projecte de programari son tàctiques, això vol dir que aquestes mètriques i els indicadors derivats son utilitzats pels gestors de projecte i l'equip de desenvolupament per adaptar el flux de treball i les activitats tècniques.

La primera aplicació de les mètriques de projecte a la majoria dels projectes de programari succeeix durant la planificació. La base de coneixement sobre mètriques recopilades de projectes anteriors (mètriques del procés) s'utilitza com a base des de la qual es realitzen les estimacions dels esforços, els recursos i el tems per al projecte de programari actual. A mesura que el projecte avança, les mesures d'esforç i de temps consumit es comparen amb les estimacions originals. El gestor de projecte a d'utilitzar aquestes dades per controlar el desenvolupament del projecte i supervisar el nivell de productivitat de l'equip.

A mesura que comença el treball de l'equip de desenvolupament, altres mesures de projectes comencen a tenir significat. Es mesura els índexs de producció representats mitjançant planes de documentació, punts per funció i línies de codi, a mes es controlen els errors detectats. Conforme el projecte de programari avança, es van recopilant mètriques que serviran per a proporcionar indicadors que influiran en l'enfocament de la part pendent del projecte.

La utilització de mètriques del projecte te dos aspectes fonamentals: En primer lloc, aquestes mètriques s'utilitzen per a minimitzar la planificació del desenvolupament, permetent els afinaments necessaris que eviten retards i disminueixen els problemes i riscos potencials. En segon lloc, s'utilitzen per a avaluar la qualitat.

Tot i que les mètriques d'avaluació de la qualitat no son l'objecte d'aquest projecte, cal esmentar que la millora de la qualitat minimitza el nombre d'errors i defectes, reduint d'aquesta manera la quantitat de treball a refer degut als errors i com a resultat s'obté una millora substancial de la productivitat i evidentment una reducció del cost global del projecte.

3.3 Mètriques del programari

Quan es parla mètriques del programari o mètriques del software es fa referència al conjunt de mètriques del producte i del procés.

La raó per la qual l'aplicació de mètriques orientades a la productivitat és tan important, son obvies. Si no s'apliquen mètriques, no hi ha forma real de determinar si s'està millorant, i si no s'està millorant, s'està perdent encara que no s'estigui empitjorant.

Els gestors de projecte i inclús els directius, dins de les companyes dedicades al desenvolupament de programari, poden establir objectius significatius de millora del procés d'enginyeria del programari sol·licitant i avaluant les mesures de productivitat del programari. El programari és un aspecte de administració estratègic per a moltes companyes, per tant si el procés de desenvolupament de programari pot ser millorat es pot produir un impacte directe en la millora de la companya.

El procés de recopilació de mètriques de programari es pot esquematitzar de la següent manera:

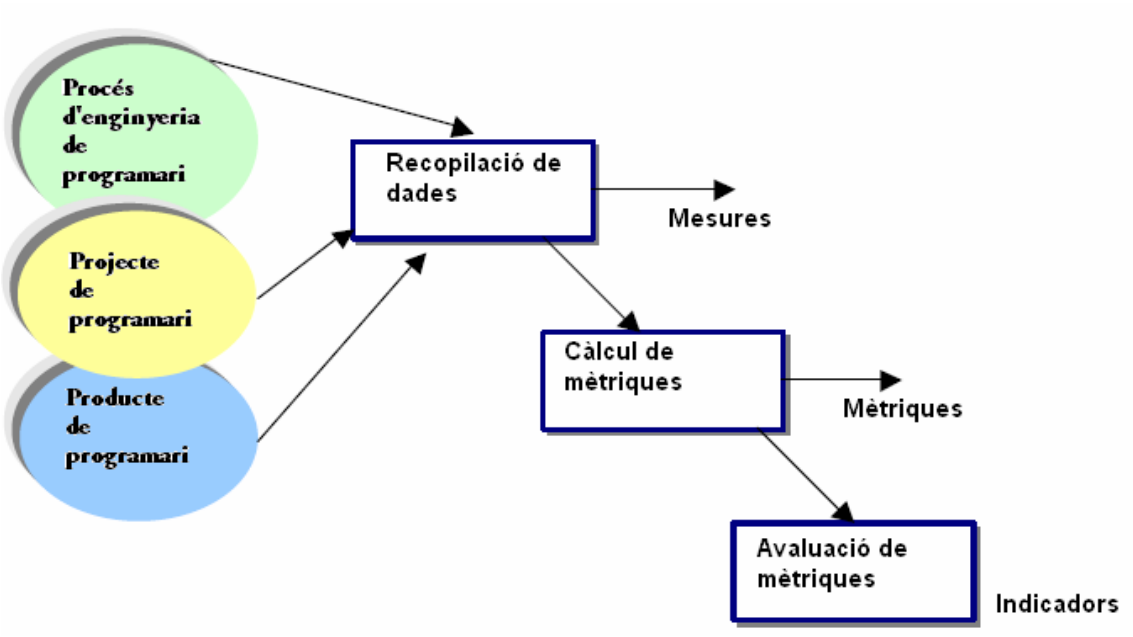


Figura 3-b: Recopilació de mètriques

4 Mètriques de productivitat

4.1 Característiques de les mètriques de Productivitat

4.1.1 Definició

Les mètriques de productivitat de programari son mesures que s'utilitzen per a quantificar aspectes del software, recursos y/o processos de desenvolupament i el seu objectiu principal és la millora del rendiment del procés d'enginyeria del programari. Inclouen aspectes que son mesurables directament, com poden ser les línies de codi, o aspectes que estan calculats a partir d'altres mesuraments.

Les mètriques de productivitat de software estan orientades principalment al tamany i la complexitat (línies de codi o punts per funció), però també poden incloure càlculs basats en mesuraments objectius de qualsevol dels factors que influeixen en la productivitat.

Per que una mètrica sigui realment útil, ha de tenir una traducció clara cap a fets estratègics de la companyia (diners, quota de mercat, acceptació, recursos entre d'altres) i ha d'ajudar a prendre mesures correctives si és necessari, ja que l'objectiu final de tota mètrica de productivitat de programari és millorar el procés de gestió i execució de projectes, la qual cosa sempre es tradueix en guanys per a la companyia. És per aquest motiu que a l'hora de dissenyar una mètrica s'ha de parar especial atenció en aquest aspecte.

4.1.2 Característiques principals

Es desitjable que totes les mètriques del programari incloses les que no estan orientades a la producció, tinguin les següents característiques principals:

- Simple i fàcil de calcular: No han de ser necessaris grans esforços per a obtenir una mesura. El principal avantatge d'aquesta característica es que es pot establir el tamany d'un programari, que pot arribar a tenir milers de línies de codi, en poques hores. El principal inconvenient d'aquesta característica es que la mètrica no pot ser massa sensible a consideracions molt detallades, com poden ser la complexitat de les fórmules que apareixen al programari.
- Empírica e intuïtiva.
- Objectiva, no ha de permetre ambigüitats.
- Consistent a l'hora de mesurar (unitats, tamanys): Els resultats obtinguts per varies persones o en projectes diferents han de ser consistents per permetre comparacions.
- Independent del llenguatge de programació i la tecnologia: Aquesta característica, encara que es molt desitjable, no sempre es compleix. Per exemple les mètriques basades en línies de codi obtindran resultats diferents depenent del llenguatge de programació amb el qual es treballi. El mes rellevant d'aquesta característica és que un cop establerta la funcionalitat requerida, és el moment d'escollir la tecnologia que permeti una major productivitat per obtenir aquesta funcionalitat.

- Eficaç per a complir els seus objectius (en el nostre cas proporcionar dades necessàries per a l'augment de la productivitat).

- Han de tenir la precisió adequada (la justa per a acomplir l'objectiu).

A mes, per a les mètriques de productivitat orientades a la funcionalitat, també son importants les següents característiques:

- Enfocament a la funcionalitat proporcionada: El principal criteri d'avaluació ha de ser quines noves capacitats s'han d'obtenir amb el nou programari, abans de cap avaluació tècnica.

Basada en els requeriments de l'usuari: Aquesta característica ajuda a que l'usuari pugui entendre el significat e implicacions de la dimensió del programari, sense haver de ser un expert en sistemes informàtics. Altre punt molt important d'aquesta característica es que es pot establir un tamany des de la etapa de presa de requeriments i no es necessari esperar a la finalització de tot l'anàlisi o del projecte per conèixer quines dimensions tindrà el projecte.

4.2 Utilitats de les mètriques de productivitat

Un cop que tenim mesurat un programari, podem utilitzar aquestes dades per a molts propòsits, a continuació es mostren algunes de les moltes utilitats de les mètriques de productivitat:

-Administrar la productivitat.- Coneixent el tamany i l'esforç requerits per a un determinat projecte, es poden establir indicadors de productivitat, com per exemple Hores per punts de funció. Aquests indicadors poden servir per a implantar i controlar un pla de millora i per a fer comparacions amb altres empreses.

-Comparabilitat.- Utilitzant la mateixa mètrica en tots els projectes dona als gestors la possibilitat de comparar-los entre ells i encara més, si la mètrica utilitzada és un estàndard generalitzat permetrà comparar els resultats obtinguts amb els d'altres empreses i inclús obtenir dades estadístiques per detectar nous factors que influeixin en la productivitat dels projectes.

-Administrar els projectes.- En comptes de mesurar únicament les hores planificades respecte a les consumides, es pot administrar els projectes basant-se en les noves dades mètriques calculades respecte de les planificades a partir de l'experiència de projectes anteriors.

-Administrar els canvis a l'abast del projecte.- Amb la utilització de mètriques estàndard com els punts de funció, es pot estimar la dimensió dels canvis i partint d'aquesta dada estimar l'esforç necessari.

-Millorar l'avaluació del cost del programari.- Coneixent el cost del programari en funció de les mètriques aplicades es pot avaluar millor el cost econòmic que ha de tenir una aplicació.

-Millorar l'estimació de recursos.- Si basem la estimació dels recursos necessaris per a un projecte en les mètriques de productivitat a mes dels paràmetres habituals la estimació serà mes senzilla i mes precisa.

-Pressupostar el manteniment.- És evident que si les mètriques poden ajudar a estimar els recursos necessaris per al desenvolupament d'un projecte, més encara per a fer estimacions i posteriors pressupostos del cost que ha de tenir el manteniment de l'aplicació. Si a mes els contractes de manteniment son periòdics, les dades de períodes anteriors encara poden ajudar a afinar mes els nous pressupostos.

4.3 Mètriques de productivitat orientades al tamany (Línies De Codi)

Les mètriques de productivitat de programari orientades al tamany provenen de la normalització de les mesures de qualitat i/o productivitat considerant el "tamany" del programari que s'hagi produït. Si una organització dedicada al desenvolupament de projectes de programari manté un registre senzill de diferents paràmetres dels projectes com son l'esforç, el cost, els errors trobats abans i després de l'entrega, el número de planes de documentació generada i el número de persones que han treballat en el projecte en relació amb dades sobre el tamany del projecte representades per línies de codi (LDC), s'obté una taula de dades orientades al tamany, com la mostrada a la taula mostrada a continuació (Pressman '98) que recull les dades de cada projecte de desenvolupament de programari i les seves mesures corresponents.

Projecte	LDC	Esforç	Cost $	Planes doc.	Errors	Defectes	Persones
X	12.100	24	168	365	134	29	3
Y	27.200	62	440	1224	321	86	5
Z	20.200	43	314	1050	256	64	6
-							
-							

Tabla 2: Dades projecte exemple

És important aclarir com es mesura l'esforç, ja que és una de les dades mes interessants a l'hora de conèixer les dades sobre la productivitat. L'esforç en aquest cas és el número de persones que participen al projecte multiplicat pel números de mesos treballats. Segons aquesta manera de mesurar el temps dona igual tenir dues persones treballant durant 3 mesos que 3 persones treballant durant 2 mesos, Aquesta manera de calcular l'esforç associada a la mètrica basada en LDC suscita moltes controvèrsies, ja que és evident que als projectes de desenvolupament de programari no produeixen el mateix 2 persones durant 6 mesos que 6 persones durant 2 mesos, sobre tot tenint en compte els problemes d'organització del treball que poden sorgir.

Com es pot observar a l'exemple de la taula anterior, al projecte X es van desenvolupar 12.100 línies de codi amb un esforç de 24 persones - mes i un cost de 168$. Sempre s'ha de tenir en compte que l'esforç i el cost a registrar per a calcular les mètriques ha d'incloure totes les activitats de la enginyeria del programari incloent-hi l'anàlisi, el disseny, la codificació i les proves, ja que és habitual caure en l'error de computar únicament els esforços i costos de la codificació. Continuant amb el projecte X es pot observar també que es van desenvolupar 365 planes de documentació, es van trobar 134 errors (abans de l'entrega al client) i es van trobar 29 defectes (després de l'entrega al client). També s'ha registrat el nº de persones que van intervenir al projecte, 3 en aquest cas.

Per a desenvolupar mètriques que es puguin comparar entre els diferents projectes, se selecciones les línies de codi com a valor de normalització. Amb la resta de dades

recollides a la taula es pot desenvolupar per a cada projecte un conjunt de mètriques simples orientades al tamany, com les següents:

- Productivitat = KLDC (milers de línies de codi)/persona - mes
- Productivitat = KLDC/Jornades
- $ per LDC
- errors per LDC
- defectes per LDC
- planes documentades per LDC
- Qualitat = errors/KLDC
- Documentació = Planes documentades/KLDC

Per poder desenvolupar aquestes mètriques esmentades, primer ha d'existir un mecanisme o procediment per a calcular les LDC del projecte, cal esmentar però que aquest mètode serveix per a fer els càlculs a posteriori i no per a fer estimacions. El procediment per fer aquesta mesura seria el següent:

- Comptabilitzar cada línea nova o modificada.
- Les línies per a la instrumentalització del codi (per exemple les proves) no han de ser incloses en el recompte total a no ser que tinguin un caràcter definitiu com a part del projecte.
- Les línies de codi de programes de prova només es comptabilitzaran si es desenvolupen amb els mateixos nivells d'exigència que la resta del projecte Lliurable.
- Es comptabilitzen les línies corresponents a les crides al sistema operatiu.
- No es consideraran els comentaris.
- No es comptabilitza el pseudocodi.
- Cada crida a subrutina o llibreria externa es considera com una línea.
- El codi generat per subrutines o llibreries externes només es comptabilitza una vegada.

Les mètriques orientades al tamany no estan acceptades universalment com el millor mètode per a mesurar factors com la productivitat en el procés de desenvolupament de programari. El gran punt de discrepància es la validesa de les (LDC) com a mesura clau. Els defensors de la utilització de LDC afirmen que les LDC son un artifici que es pot calcular fàcilment per a tots els projectes de desenvolupament de programari, que molts models d'estimació dels indicadors del programari existent utilitzen LDC o KLDC com a mesura clau per als seus càlculs. A l'altra banda estan els detractors del mètode que argumenten que les mesures en LDC son dependents del llenguatge de programació, que perjudiquen els programes mes curts però amb un bon disseny i un rendiment i productivitat provats, que costa d'aplicar fàcilment a llenguatges procedimentals, i que la seva utilització per a la estimació requereix un nivell de detall que pot resultar molt difícil d'aconseguir (és necessari conèixer una estimació de les LDC a produir molt abans de completar les etapes d'anàlisi i disseny).

4.4 Mètrica de punts de funció

4.4.1 Introducció

Aquesta mètrica es defineix com una mètrica funcional, donat que s'enfoca a la funcionalitat que el programari proporciona als usuaris. Es una mètrica per a establir el

tamany i la complexitat dels sistemes informàtics basada en la quantitat de funcionalitat requerida i entregada als usuaris.

Els punts per funció mesuren el tamany lògic o funcional dels projectes o aplicacions de programari basats en els requeriments funcionals dels usuaris.

Les característiques d'aquesta mètrica son les següents:

- Tamany: És una mètrica orientada al tamany, no a la qualitat amb la que es va fer el programari, el seu valor o l'esforç necessari per al desenvolupament.
- Aplicacions: Mesura les aplicacions de programari, no considera el maquinari sobre el que funcionaran ni l'administració del projecte, ni la documentació, etc.
- Funcionalitat: Es refereix a la capacitat del programari per permetre a un usuari gestionar dades, realitzar transaccions amb elles i emmagatzemar-les. Si s'analitza amb detall, és possible descriure qualsevol sistema amb aquests elements.
- Usuari: Qui l'ha d'utilitzar i no els desenvolupadors o dissenyadors.

De la mateixa manera que existeixen regles o metros per a mesurar longituds, punt per funció es "la regla" per mesurar el tamany d'una aplicació de programari.

4.4.2 L'estàndard internacional ISO de mètriques orientades a la funció.

La medició de la funcionalitat amb la que compta un sistema informàtic ha estat des de fa anys una important preocupació de la industria. No és suficient comptar amb una mètrica per a mesurar aquest factor, si no que aquesta mètrica ha d'esser estàndard per a poder ser utilitzada entre diferents empreses o per a tenir indicadors a nivell industrial que tothom pugui entendre i operar amb ells. La possibilitat de comparar la productivitat (punts per funció per persona i mes) d'una empresa amb les dades de la resta de la industria, és fonamental en qualsevol pla de millora.

És per aquest motiu que a través de la "Inernational Organization for Standarization" (ISO) s'ha desenvolupat un estàndard internacional, les normes del qual son les següents:

ISO/IEC 14143 - *Information Technology – Software Measurement – Functional Size Measurement*. Aquest estàndard defineix els conceptes d'una mètrica basada en la funcionalitat i les característiques que ha d'acomplir un mètode per a poder estar homologat a l'estàndard i esser considerat com una mesura de la dimensió de la funcionalitat.

Per a poder establir la quantitat de punts per funció que te una aplicació, existeix mes d'un mètode per comptar. En general tots aquests mètodes estableixen una manera de comptar basada en la identificació del tipus de funcions que te l'aplicació, i a cadascuna li associa un número de punts per funció depenent de la seva complexitat. Les variants sorgeixen en buscar maneres de comptar mes precises als punts per funció respecte al tipus d'aplicació. Per exemple, un sistema en temps real te una complexitat molt distinta a un sistema tradicional de negoci o a un sistema operatiu o a una aplicació científica per a automatitzar càlculs complexos, però el resultat de les seves mesures pot ser una dada única i comparable. Els mètodes homologats amb la ISO 14143 (encara que no tots son públics) son:

- ISO/IEC 20926:2003 *Software engineering* – ***IFPUG*** *4.1 Unadjusted functional size measurement method*. Aquest mètode ha estat definit per *l'International*

Function Point Users Group (IFPUG) i ha evolucionat, a partir de la proposta original de Allan Albrecht en IBM, per la qual cosa és el més conegut i el més utilitzat, sobre tot als Estats Units que es el mercat més gran de Programari. Això és molt important per que s'està posicionat com l'estàndard de la industria.

- ISO/IEC 24570. *Software engineerin**g*** -- ***NESMA*** -- *Function Point Analysis*. Estàndard definit per la NEtherlands Software Metrics Users Association. Aquesta és una petita variant del mètode del IFPUG.

- ISO/IEC 20968:2002 *Software engineering* -- ***Mk II*** *Function Point Analysis*. Aquest mètode ha estat desenvolupat per la United Kingdom Software Metrics Association, simplificant el mètode y fent-lo compatible amb noves idees d'anàlisis i disseny estructurat.

- ISO/IEC 19761:2003 *Software engineering* -- ***COSMIC-FFP*** -- *A functional size measurement method*. Aquest mètode ha estat definit pel Common Software Measurement International Consortium, integrat per experts de Austràlia, Canadà, Finlàndia, Alemanya, Irlanda, Itàlia Japó, Holanda y el Regne Unit. La idea principal és adequar-se millor a la medició de sistemes en temps real.

4.4.3 El mètode estàndard d'anàlisi de punts per funció.

Com ja s'ha mencionat, el mètode que s'està convertint en l'estàndard de la industria és el definit per l'IFPUG, que s'anomena *Function Point Analysis* (FPA) i els seus autors el defineixen així:

"Mètode estàndard per a mesurar el desenvolupament de programari des del punt de vista de l'usuari. "

Aquest mètode utilitza com unitat de mesura els PF i la seva versió actual és la 4.2, i el seu manual de referència es pot trobar tant en anglès com en castellà (després de subscriure's per 50$ i pagar 10$ pels manuals). A continuació es descriu breument en que consisteix aquest mètode.

El mètode es basa principalment en la identificació dels components del sistema informàtic en termes de transaccions y grups de dades lògiques que son rellevants per al negoci de l'usuari. A cadascun d'aquests components els assigna un número de punts per funció basant-se en el tipus de component i la seva complexitat; i la sumatòria de tot això dona com a resultat els punts de funció sense ajustar. L'ajustament és un pas final basant-se en les característiques generals de tot el sistema informàtic que s'està mesurant.

Per a entendre millor els conceptes mencionat i veure la seva simplicitat final, s'ha de veure amb mes detall el procediment que a continuació es detalla.

I Procediment

I. Determinar el tipus de conteig.
Aquest pas consisteix a definir el tipus de conteig entre desenvolupament, manteniment o una aplicació ja instal·lada. La següent figura mostra un exemple de com determinar l'objectiu del conteig.

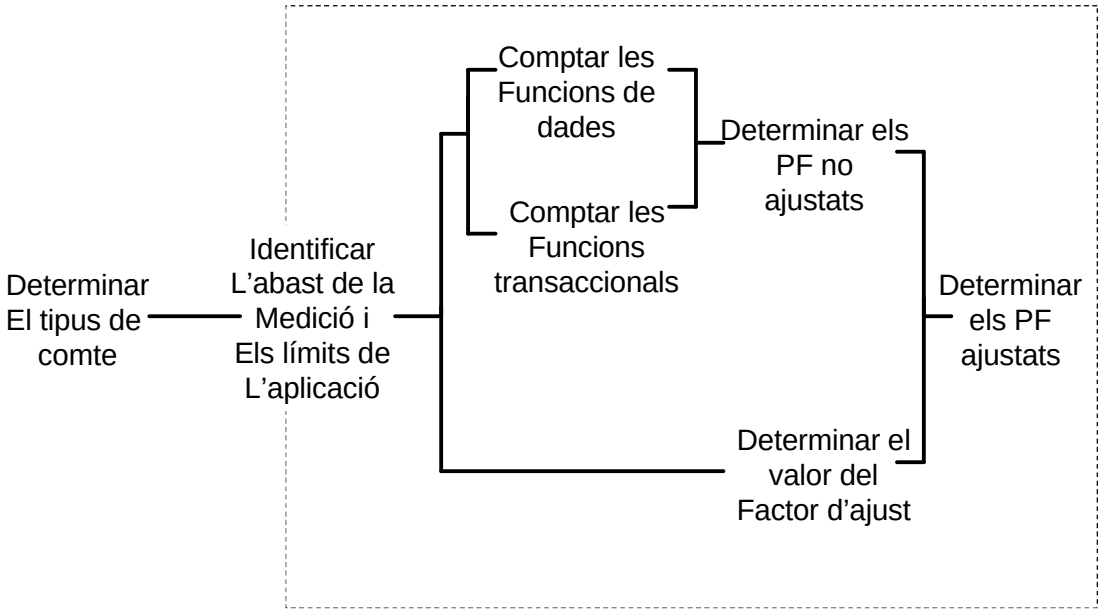


Figura 4-a:Comptar PF

II. Identificar l'abast de la medició i els límits de l'aplicació.

El propòsit d'una medició consisteix a donar una resposta a un problema de negoci. L'abast de la medició defineix la funcionalitat que s'ha d'incloure en una medició específica i pot abastar més d'una aplicació.

III. Comptar les funcions de dades.

Aquest pas consisteix a identificar i comptar la capacitat d'emmagatzemament de les dades. Es distingeixen dos tipus de funcions de dades:

Arxiu Lògic Intern – es un grup de dades relacionades que l'usuari identifica, el propòsit principal és l'emmagatzemament de dates mantingudes a través d'alguna transacció que s'està considerant en el conteig.

Arxiu d'interfase Extern - és un grup de dades relacionades y referides però no mantingut per cap transacció dins del conteig.

A cada component identificat se li assigna una complexitat (baixa, mitja o alta) considerant principalment el número de dades.

IV. Comptar les funcions transaccionals.

Aquest pas consisteix a identificar i comptar la capacitat de realitzar operacions.
Es distingeixen tres tipus de funcions transaccionals:

Entrada externa – El propòsit principal d'aquest procés és mantenir un o mes arxius lògics interns.

Sortida Externa – El propòsit principal d'aquest procés és presentar informació a l'usuari mitjançant un procés lògic diferent al de la simple recuperació de dades.

Consulta Externa – El propòsit principal d'aquest procés és presentar informació a l'usuari llegida d'un o més grups de dades.

A cada component identificat se li assigna una complexitat (baixa, mitja o alta) considerant el número de dades utilitzades en el procés i els arxius referits.

Aquests cinc components lògics bàsics son els que ajuden a descriure la funcionalitat d'una aplicació i els podem representar gràficament de la següent manera:

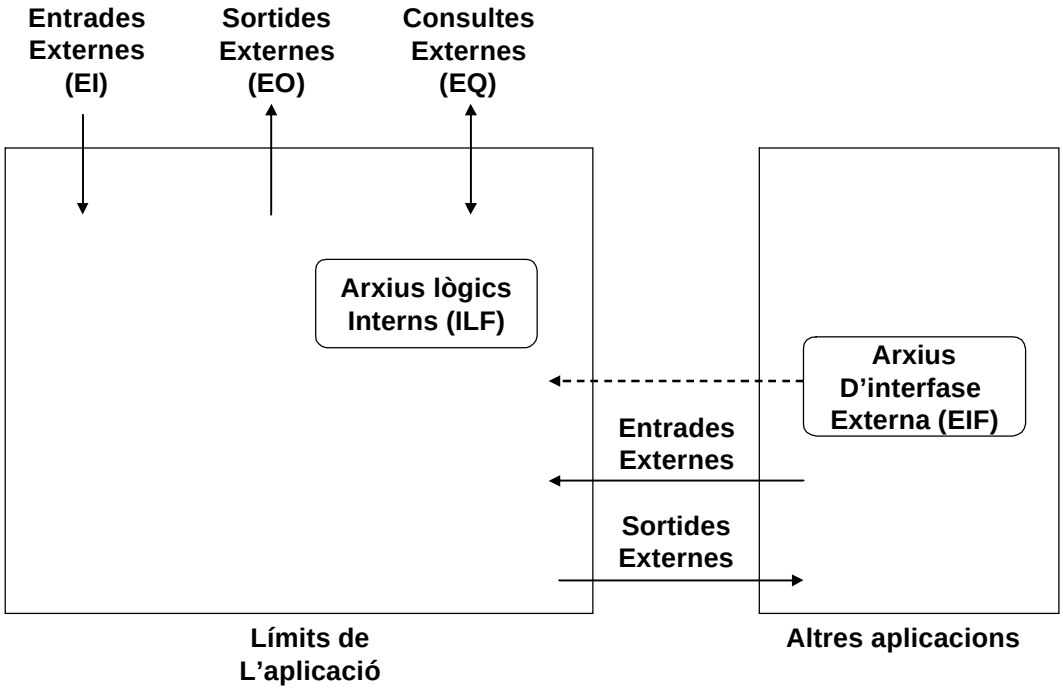


Figura 4-b:Funcions transaccionals

V. Determinar els punts de funció no ajustats.

Aquest pas consisteix a sumar el número de components de cada tipus conforme a la complexitat assignada y utilitzar la següent taula per a obtenir el total.

	Baixa	Mitjana	Alta	Total
EI	___x 3 = __	___x 4 = __	___x 6 = __	_____
EO	___x 4 = __	___x 5 = __	___x 7 = __	_____
EQ	___x 3 = __	___x 4 = __	___x 6 = __	_____
ILF	___x 7 = __	__x 10 = __	__x 15 = __	_____
EIF	___x 5 = __	___x 7 = __	__x 10 = __	_____
				_____

Tabla 3: PF no ajustats

VI. Determinar el valor del factor d'ajustament.

El factor d'ajustament s'obté sumant 0.65 a la sumatòria dels graus d'influència de les catorze característiques generals del sistema, multiplicat per 0.01. Dins de les característiques existeixen criteris com: complexitat del procés, facilitat d'instal·lació, entrada de dades en línea, etc, les podem veure representades a la taula següent:

Característica
Comunicació de dades
Processament distribuït De dades
Rendiment
Configuracions utilitzades De manera forta
Freqüència de transaccions

Entrada de dades on-line
Eficiència de l'usuari final
Actualitzacions on-line
Processament complex
Reutilització
Facilitat d'instal·lació
Facilitat d'operació
Instal·lació en llocs diferents
Facilitat de canvi

Tabla 4: Factors d'ajustament PF

VII. Determinar els PF ajustats.

Per a determinar els PF ajustats es consideren els PF no ajustats pel factor d'ajustament.

4.4.4 Classificació de transaccions i arxius a l'anàlisi de punts de funció

La complexitat de les Transaccions i els arxius a l'anàlisi de Punts de Funció, es pot classificar i quantificar d'acord amb els criteris que es mostren a continuació:

Transaccions
a) Classificació de les Entrades Externes
Per a les Entrades Externes, la classificació ve donada per la següent taula:

Arxius Referenciats	Elements de dades		
	1-4	5-15	>15
0-1	Baixa	Baixa	Mitja
2	Baixa	Mitja	Alta
3 o mes	Mitja	Alta	Alta

Tabla 5: Relació Entrada Externa

A la taula, "Arxius referenciats" representa el número d'arxius Lògics Interns mantinguts per la Entrada Externa, i "Elements de dades" representa la quantitat d'elements que composen la Entrada Externa.

b) Classificació de les Sortides Externes i Consultes Externes
Per a les Sortides Externes i les Consultes Externes, la classificació ve donada per la següent taula:

Arxius Referenciats	Elements de dades		
	1-5	6-19	>19
0-1	Baixa	Baixa	Mitja
2	Baixa	Mitja	Alta
3 o mes	Mitja	Alta	Alta

Tabla 6: Relació sortides externes

A la taula, "Arxius referenciats" representa el número d'arxius Lògics Interns o Arxius de Interfase Externs vinculats amb la Sortida Externa o la
Consulta Externa, i "Elements de dades" representa la quantitat combinada d'elements de dades d'entrada i de sortida que composen la Sortida Externa o Consulta Externa.

c) Assignació de valors numèrics

Els valors numèrics que s'assignen a cada complexitat (Baixa, Mitja o Alta), es mostren a la següent taula, per a cadascun dels tipus de transacció (Entrada Externa, Sortida Externa, Consulta Externa):

Classificació	Valors		
	Sortides Externes	Consultes Externes	Entrades Externes
Baixa	4	3	3
Mitja	5	4	4
Alta	7	6	6

Tabla 7:Assignació valors numèrics

Arxius

a) Classificació dels Arxius Lògics Interns i Arxius de Interfase Externs

Per als Arxius Lògics Externs i els Arxius de Interfase Externs, la classificació ve donada per la següent taula:

Tipus de registre	Elements de dades		
	1-19	20-50	>50
1	Baixa	Baixa	Mitja
2-5	Baixa	Mitja	Alta
>5	Mitja	Alta	Alta

Tabla 8: Classificació Arxius lògics interns i de interfase externs

on "Tipus de registre" representa un subgrup d'elements de dades identificables per l'usuari, i "Elements de dades" representa la quantitat d'elements de dades bàsiques (camps únics) que composen l'arxiu.

El concepte de "Tipus de registre" es pot veure millor mitjançant un exemple:

Suposem que s'emmagatzema la informació d'un CD de música en un Arxiu Lògic Intern. La informació associada al CD és el Cantant, el Productor, el
Títol, la data i les cançons. Cada cançó a la seva vegada te com a informació associada un Nom, un Autor i una Duració.

En aquest cas es tenen 2 "Tipus de registre", la informació del CD i la informació de la cançons. A la seva vegada, existeixen 4 "Elements de dades" per a la informació del CD (Cantant, Productor, Títol i Data), i 3 "Elements de dades" per a la informació de la cançó (Nom, Autor i Duració).

b) Assignació de valors numèrics.

Els valors numèrics que s'assignen a cada complexitat (Baixa, Mitja o Alta), es mostren a la següent taula, per a cadascun dels tipus d'arxiu (Arxiu Lògic Extern, Arxiu de Interfase Extern):

Classificació	Valors	
	Arxiu Lògic Intern	Arxiu de Interfase Extern
Baixa	7	5
Mitja	10	7
Alta	15	10

Tabla 9: Valors numerics

Com a síntesi, només dir que El compte de Punts de Funció comença amb la identificació de les Transaccions (Entrades Externes, Sortides Externes, Consultes Externes) i els Arxius (Lògics Interns o de Interfase Externs).
Un cop identificats aquests elements, s'utilitzen els criteris mostrats de manera de quantificar l'aportació de cadascun d'ells a la suma total de Punts de Funció.

4.4.5 Abast de Punts per funció.

Qualsevol mètrica te un àmbit d'acció i abast definit que s'ha d'entendre per a utilitzar-la correctament, de la mateixa manera que entenem que el metre lineal no és el millor mètode per a mesurar grans distancies al mar.

En aquest cas, Punts Funció, està enfocat a mesurar sistemes informàtics complerts, no programes. En aquest sentit no te un nivell de precisió suficient per a mesurar el tamany de programes individuals. El nivell de atomicitat que pot mesurar la mètrica no es massa petit. Addicionalment el terme programa depèn de la tecnologia i això va en contra del criteri que diu que aquesta mètrica és independent de la tecnologia utilitzada.

Altre aspecte que s'ha criticat a les mètriques funcionals és que requereixen que algú "identifiqui" la funcionalitat i "avaluï" la complexitat basant-se en els criteris i regles establertes; no es pot fer un programa que compti automàticament. Degut a això, distintes persones podrien arribar a un resultat diferent. Per a resoldre això, s'han anat depurant les regles de conteig per a eliminar possibles ambigüitats i cada cop hi ha més material de suport con poden ser exemples. També, existeixen esquemes de certificació com el del IFPUG, on hi ha una avaluació formal de la teoria i casos pràctics. Aquests elements busquen reduir les possibles variacions en un conteig fet per diferents persones. Si tenim en consideració que la mètrica està pensada per a contar projectes o aplicacions complertes, a les hores les petites variacions en un conteig, no han de ser significatives per als indicadors o dades relacionades obtingudes.

4.5 Relació entre línies de codi i punts de funció

4.5.1 Relació entre tamany i funció.

Les LDC i els PF en principi son mesures independents, però és raonable suposar que la funcionalitat d'un sistema i el seu tamany estan relacionats, per exemple els sistemes operatius, a mesura que surten noves versions amb més funcionalitat per a l'usuari el tamany d'aquestes versions també es cada vegada major.

Existeixen estimacions informals del número de LDC necessaris per a construir un PF, com es mostra a la següent taula.

Llenguatge de programació	LDC/PF (mitja)
Assemblador	320
C	128
COBOL	106
FORTRAN	106
PASCAL	90
C++	64
ADA95	53

VISUAL BASIC	32
SMALLTALK	22
POWERBUILDER	16
SQL	12

Tabla 10:LDC per PF segons Llenguatge

Observant la taula es pot dir que com mes avançat és un llenguatge de programació, més expressives son les seves sentencies. De tota manera també hem de tenir en compte que no es pot desenvolupar qualsevol programari amb qualsevol llenguatge de programació (seria inviable programar un sistema operatiu o de control de maquinari amb SQL).

Encara que es podria aplicar directament la relació anterior per a calcular els PF en base a les LDC (tècnica *backfiring*), es pot refinar una mica més la tècnica per obtenir uns resultats mes acurats amb la següent fórmula:

$$PF = LDC_{aplicació}/((LDC/PF_{media})*FAB)$$

On FAB és el factor d'ajustament del *Backfiring*, la base per obtenir-lo és la següent taula:

Complexitat	FAB
Molt simple	0,7
Simple	0,85
Mitja	1
Moderadament complexa	1,2
Complexa	1,3

Tabla 11:Factor d'ajust Backfiring

Aquest factor compensa la funcionalitat amb la falta o excés de tamany depenent de la complexitat.

Per il·lustrar aquesta tècnica amb un exemple, es pot considerar un sistema amb 45.000 LDC escrites en el llenguatge C i una millora posterior de la interfície gràfica d'usuari de 10.000 LDC escrites en el llenguatge C++. Suposem que la part C és complexa i la part C++ simple, la relació seria la següent:

$PF_C = 45.000/(128*1,3) = 270$ PF
$PF_{C++} = 10.000/(30*0,85) = 392$ PF
$PF_{aplicació} = 270(PF) + 392(PF) = 662(PF)$

4.5.2 Conclusions

Una aplicació de programari és un conjunt de línies de codi que s'executen en una computadora, però la major part del cost de produir el programari no està directament relacionat amb la codificació, que s'estima entre un 20 i un 25% del cost total. Elements com l'administració del projecte, el nivell de detall de la documentació tècnica o la documentació de les proves i les proves en si mateixes, també s'han de considerar.

Com s'ha pogut observar prèviament, existeix una relació entre LDC i PF segons el llenguatge de programació, ja que és evident que si s'utilitza la mateixa tecnologia, un programari amb més funcionalitat ha de ser per força més gran i complex. Mitjançant la tècnica del *Backfiring* es pot obtenir un valor aproximat per al PF partint de les LDC i de la complexitat del programari, aquest valor obtingut es pot prendre com a aproximació, però

mai s'ha de prendre com a valor real ja que en el càlcul dels PF es tenen en compte diversos factors que no afecten el càlcul de LDC i el resultat del *Backfiring* es pot allunyar considerablement del valor real de PF.
El càlcul de LDC a partir de punts de funció, mitjançant la taula de conversió segons el llenguatge de programació, pot ser molt útil per aplicar el mètode COCOMO II i estimar l'esforç, ja que mentre que l'estimació de PF es pot aconseguir amb dades bastant encertades, és molt difícil fer una estimació de les LDC que tindrà una aplicació, per moltes dades estadístiques de les que es disposi.

4.6 Punts de característica

4.6.1 Introducció

Com ja s'ha vist, els punt de funció van ser desenvolupats originalment per a mesurar el tamany funcional de sistemes de programari orientats a la gestió. En aquest camp d'aplicació, els Punts de Funció han demostrat esser, probablement, la mètrica mes eficaç, la qual cosa ha fet que sigui una de les mes utilitzades en gran part del mon.

Però si alterem el camp d'aplicació, es a dir, si apliquem aquesta mètrica a sistemes que no siguin de gestió (programari per al control de processos, sistemes en temps real, sistemes embeguts, etc...), els Punts de Funció deixen de ser la mètrica perfecta: ja no es comporta de forma tan eficaç i tan òptima com amb els sistemes de gestió. Sorgeix, a les hores, la necessitat de buscar un sistema de mesurament del tamany funcional alternatiu per als altres tipus d'aplicacions.

Amb aquesta intenció, Caper Jones va desenvolupar al 1986 un mètode experimental par a adaptar la mètrica de Punts de Funció a sistemes software científics i d'enginyeria. Per a evitar confusions amb els Punts de Funció d'Albrecht, es va anomenar Punts de característica (Feature Points) a aquest nou mètode de mesura del tamany funcional. Fins el dia d'avui, son moltes les proves que s'han realitzat amb aquest nou mètode de mesura. Els Punts de Característica s'han anat utilitzant amb gran èxit en el mesurament de sistemes software molt variats com els que ja hem anomenat. A diferencia dels sistemes de gestió, aquest altre tipus de sistemes te dues característiques bàsiques:

- La complexitat algorítmica que implementen.
- El petit número de entrades y sortides que solen tenir.

Donat que el conteig de Punts de Funció no contempla en absolut la complexitat algorítmica dels sistemes, això fa que els resultats obtinguts no siguin realment significatius. En general, i per a aquests sistemes, el resultat obtingut després d'un Anàlisis de Punts de Funció acostuma a ser bastant inferior (entre un 20 y un 35%) al que s'obté amb altres tècniques que sí contemplen la complexitat algorítmica.

Per exemple: per a aplicacions on el número d'algoritmes es molt major que el nombre de fitxers lògics (sistemes de control, embeguts, de temps real, etc...), els Punts de Característica retornen un valor molt mes alt que els Punts de Funció. I al contrario; per a aplicacions amb un gran número de fitxers lògics y poca complexitat algorítmica (sistemes de gestió), los Punts de Característica retornaran un valor inferior al dels Punts de Funció, encara que en algunes ocasions els resultats poden ser bastant semblants.

4.6.2 Procés per contar punts de característica

La mètrica de Punts de Característica és un súper conjunt de la mètrica de Punts de Funció. Introdueix un nou component, els algoritmes, que s'afegeixen als cinc ja existents: Entrades Externes, Sortides Externes, Consultes Externes, Fitxers Lògics Interns y Fitxers Externs d'interfase. Un algoritme é defineix com "un problema de complexitat computacional limitada, que s'inclou dins d'un determinat programa d'ordinador". Caper Jones proposa una definició alternativa: "Es un conjunto de regles perfectament definides, que ajuden a la resolució d'un determinat problema computacional". Però el tema que ens ocupa es el del conteig de
Punts de Característica; en aquesta mètrica, el terme algoritme es definirà com "un problema computacional d'abast y límits ben definits, que s'implementa dins d'una determinada aplicació informàtica".

Per defecte, a aquest nou component (els algoritmes) se li assigna un factor de pes de 3 Punts de Característica. A mes, en el conteig de Punts de Característica es redueixen els pesos assignats als Fitxers Lògics Interns de complexitat Mitja, passant aquests de tenir un pes de 10 Punts de Funció a tenir un pes de 7 Punts de Característica. Pel que fa respecte a la resta dels components del sistema, s'utilitza un únic factor de pes per a cadascun d'ells, com es pot veure en la taula següent.

Component	Complexitat del component (factor de pes)	Total
Entrades Externes	___x 4 = __	_____
Sortides Externes	___x 5 = __	_____
Consultes Externes	___x 4 = __	_____
Fitxers Lògics Interns	___x 7 = __	_____
Fitxers Externs d'interfase	___x 7 = __	_____
Algorismes	___x 3 = __	_____
	Nº Total de punts Característica sense ajustar	_____
	Factor d'ajustament	x____
	Nº Total de punts de característica ajustats	_____

Tabla 12: Punts de característica

Com es pot veure a la taula anterior, el mètode de conteig de Punts de Característica es similar al de Punts de Funció. En aquest cas, per a obtenir el Número Total de Punts de Característica Ajustats (PCA) (equivalent al tamany funcional de l'aplicació), multiplicarem el Factor d'ajustament (VAF) per el Número Total de Punts de Característica sense Ajustar (PCsA), es a dir:

$$PCA = VAF * PCsA$$

4.6.3 Els algorismes en el procés de contar punts de característica

A l'apartat anterior s'han vist definicions del concepte d'algorisme. Els algorismes varien tant en dificultat d'implementació com en complexitat computacional, per tant se'ls han d'aplicar factors de pes diferents. El rang de valors que s'utilitza és de 1 a 10 encara que s'acostuma a aplicar per defecte el factor de complexitat 3. Quan els algorismes es basen únicament en operacions aritmètiques bàsiques, de poca complexitat, s'assigna el valor mínim: 1. No obstant, quan es basen en complicades equacions matemàtiques, operacions amb matrius, etc... s'aplica el valor màxim: 10. De tot això es fàcil deduir que segons augmenti la complexitat algorítmica d'una aplicació, així augmentarà el valor final en Punts de Característica, encara que la correlació no es totalment lineal.

Existeix una certa subjectivitat a l'hora d'estimar cóm de complex es un algoritme. Per a tractar d'evitar-la en la mesura del possible, s'atien als dos aspectes següents:

- El número de passos de càlcul o regles de que consta l'algoritme.
- El número de factors o dades que necessita l'algoritme per a treballar.

Aquests criteris no son ni molt menys determinants ni definitius, ja que la tècnica de contar Punts de Característica està encara en fase experimental. D'altra banda, per a determinar quins algoritmes son realment significatius en el procés de contar i, per tant, quals s'han d'incloure en el compte, s'han definit una sèrie de regles o condicions que aquest algoritmes hauran de complir. Es tracta de las següents:

- Els algoritmes hauran d'estar pensats per a problemes resolubles i amb abast i límits ben definits.
- Els algoritmes hauran de ser finits en temps d'execució, és a dir, la seva execució ha de concloure en temps finit.
- hauran de ser precisos i no ambigus.
- hauran d'admetre una entrada o valors de començament, i tornar una sortida o valors resultat.
- Un algoritme podrà incloure un o mes sub algoritmes, o cridar d'altres algoritmes.
- Tots els passos de que consti s'hauran de poder implementar en la màquina, utilitzant les sentencies típiques de la programació estructurada (if-then-else, do-while, case, etc...).

Altre cop no està de mes recordar que contínuament es tracten d'ampliar, millorar i definir mes explícitament aquests criteris, ja que el conteig de Punts de Característica no és avui dia una tècnica totalment madura.

4.6.4 Punts de funció vs Punts de característica.

És habitual que sorgeixi la següent pregunta: Quan utilitzar Punts de Funció i quan els Punts de Característica?

La resposta és clara, encara que no sempre fàcil de detectar: si l'aplicació que pretenem mesurar te un alt component algorítmic (un gran número d'algoritmes) o aquets, tot i sent pocs, impliquen una elevada complexitat computacional, s'optarà per la utilització de Punts de Característica com a mètode mes apropiat per a mesurar el seu tamany funcional. Pel contrari, si a l'aplicació a mesurar apareixen pocs algorismes (o d'escassa importància) i un alt número d'entrades, sortides, consultes y fitxers lògics, s'optarà per contar Punts de Funció com la opció mes apropiada.

A dia d'avui, son pocs els sistemes el tamany funcional dels quals s'ha mesurat amb el compte de Punts de Característica. Per aquest motiu, els resultats disponibles son realment escassos, però suficients com per a treure certes conclusions. Es pot observar a la taula següent una sèrie de ràtios comparatius entre aplicacions mesurades en Punts de Funció i en Punts de Característica:

Tipus de sistemes	Ràtios en...	
	Punts Funció	Punts Característica
Orientats a la gestió, amb processament per lots	1	0,80
Orientats a la gestió, interactius	1	1
Bases de dades on-line	1	1

Per al control de processos	1	1,28
En temps real	1	1,35
Per a automatització industrial	1	1,50

Tabla 13: Relació PF-PC

Como es pot veure, per a aplicacions de gestió amb processament per lots, ofereix millors resultats la estimació amb Punts Funció que la estimació amb Punts Característica. Per a aplicacions de gestió interactives i aplicacions de bases de dades on-line, ambdós indicadors produeixen els mateixos resultats, es a dir, és indiferent la utilització de l'un o l'altre. Per a la resta d'aplicacions (control de processos, sistemes en temps real, etc...), la estimació amb Punts Característica resulta ser la opció mes apropiada.

4.7 Punts de casos d'us

4.7.1 Introducció

La estimació mitjançant l'anàlisi de Punts de Casos d'us és un mètode proposat originalment per Gustav Karner de Objectory AB, i posteriorment refinat per molts altres autors. Es tracta d'un mètode d'estimació del temps de desenvolupament d'un projecte de programari orientat a objectes mitjançant la assignació de "pesos" a un cert número de factors que l'afecten, per a finalment, contabilitzar el temps total estimat per al projecte a partir d'aquests factors.

A continuació, es detallen els passos a seguir per a l'aplicació d'aquest mètode.

4.7.2 Càlcul de Punts de Casos d'us sense ajustar

El primer pas per a la estimació consisteix en el càlcul dels Punts de Casos d'us sense ajustar. Aquest valor, es calcula a partir de la següent equació:

$$UUCP = UAW + UUCW$$

On ,

- UUCP: Punts de cassos d'us sense ajustar
- UAW: Factor de pes dels actors sense ajustar
- UUCW: Factor de pes dels casos d'us sense ajustar

Factor de pes dels actors sense ajustar (UAW): Aquest valor es calcula mitjançant un anàlisi de la quantitat d'actors presents en el sistema i la complexitat de cadascun d'ells. La complexitat dels actors s'estableix tenint en compte en primer lloc si es tracta d'una persona o d'altre sistema, i en segon lloc , la forma en que l'actor interactua amb el sistema. Els criteris es mostren a la següent taula:

Tipus D'actor	Descripció	Factor De Pes
Simple	Altre sistema que interactua amb el sistema a desenvolupar mitjançant una interfase de programació (API, Application Programming Interface).	1
Mitjà	Altre sistema que interactua amb el sistema a desenvolupar mitjançant un protocol o una interfase basada en text.	2

Complex	Una persona que interactua amb el sistema mitjançant una interfase gràfica.	3

Tabla 14:Factors de pes dels actors

Factor de pes dels casos d'us sense ajustar (UUCW): Aquest valor es calcula mitjançant una anàlisi de la quantitat de Casos d'us presents en el sistema y la complexitat de cadascun d'ells. La complexitat dels Casos d'us s'estableix tenint en compte la quantitat de transaccions efectuades en el mateix, on una transacció s'entén com una seqüència d'activitats atòmica, és a dir, s'efectua la seqüència d'activitats completa, o no s'efectua cap de les activitats de la seqüència. Els criteris es mostren en la següent taula:

Tipus de cas D'us	Descripció	Factor De Pes
Simple	El cas d'us conté d'una a tres transaccions.	5
Mitjà	El cas d'us conté de quatre a set transaccions.	10
Complex	El cas d'us conté mes de 8 transaccions.	15

Tabla 15:Pes dels casos d'us sense ajustar

Exemple

Per aclarir els conceptes mostrats fins el moment, podem tomar un sistema d'administració d'ordres de compra.

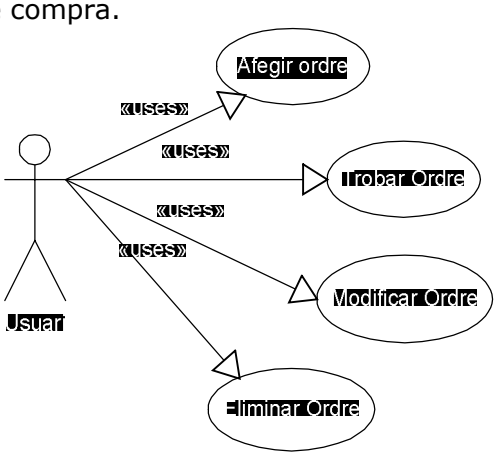


Figura 4-c: Exemple Casos d'us

Aplicant l'anàlisi de Punts de Casos d'us sense ajustar, s'obté:

Factor de Pes dels Actors sense ajustar (UAW)
L'usuari constitueix un actor de tipus complex, ja que es tracta d'una persona utilitzant el sistema mitjançant una interfase gràfica, al qual se li assigna un pes 3. Aleshores, el factor de pes dels actors sense ajustar resulta:

UAW = 1x3 = 3

Factor de Pes dels Casos d'us sense ajustar (UUCW)
Cadascun dels casos d'us "Afegir ordre", "Modificar ordre" y "Eliminar ordre" consten d'una única transacció, i el cas d'us "Trobar ordre" consta de dos transaccions (l'usuari indica la mostra i el sistema la ubica i la presenta). Es tenen aleshores 4 casos d'us de tipus simple (pes 5), amb la qual cosa el factor de pes dels casos d'us sense ajustar resulta:

UUCW = 4x5 = 20

Finalment, els Punts de Casos d'us sense ajustar resulten:

UUCP = UAW + UUCW = 3+20 = 23

4.7.3 Càlcul de punts de casos d'us ajustats

Un cop que es tenen els Punts de Casos d'us sense ajustar, s'ha d'ajustar aquest valor mitjançant la següent equació:

UCP = UUCP x TCF x EF

On,
UCP: Punts de Casos d'us ajustats
UUCP: Punts de Casos d'us sense ajustar
TCF: Factor de complexitat tècnica
EF: Factor d'ambient

Factor de complexitat tècnica (TCF)

Aquest coeficient es calcula mitjançant la quantificació d'un conjunt de factors que determinen la complexitat tècnica del sistema. Cadascun dels factors es quantifica amb un valor de 0 a 5, on 0 significa una aportació irrellevant i 5 una aportació molt important. En la següent taula es mostra el significat i el pes de cadascun d'aquests factors:

Factor	Descripció	Pes
T1	Sistema distribuït	2
T2	Objectius de performance o temps de resposta	1
T3	Eficiència de l'usuari final	1
T4	Processament intern complex	1
T5	El codi ha de ser reutilitzable	1
T6	Facilitat d'instal·lació	0,5
T7	Facilitat d'us	0,5
T8	Portabilitat	2
T9	Facilitat de canvi	1
T10	Concurrència	1
T11	Inclou objectius especials de seguretat	1
T12	Proveeix accés directe a components de tercers	1
T13	Es requereixen facilitats especials d'entrenament A usuaris	1

Tabla 16:Factors de complexitat tècnica

El Factor de complexitat tècnica es calcula mitjançant la següent equació:

$$TCF = 0,6 + 0,01 \times \sum(Pes_i \times Valor\ assignat_i)$$

Factor d'ambient (EF)

Les habilitats i l'entrenament de l'equip involucrat en el desenvolupament tenen un gran impacte en les estimacions de temps. Aquests factors son els que es contemplen en el càlcul del Factor d'ambient. El càlcul d'aquest és similar al càlcul del Factor de complexitat tècnica, és a dir, es tracta d'un conjunt de factors que es quantifiquen amb valores de 0 a 5.

A la següent taula es mostra el significat i el pes de cadascun d'aquests factors.

Factor	Descripció	Pes
E1	Familiaritat amb el model de projecte utilitzat.	1,5
E2	Experiència en la aplicació.	0,5
E3	Experiència en orientació a objectes.	1
E4	Capacitat de l'analista líder.	0,5
E5	Motivació.	1
E6	Estabilitat dels requeriments.	2
E7	Personal part-time.	-1
E8	Dificultat del llenguatge de programació.	-1

Tabla 17:Factors d'ambient

- Per als factors E1 a E4, un valor assignat de 0 significa sense experiència, 3 experiència mitja i 5 amplia experiència (expert).
- Per al factor E5, 0 significa sense motivació per al projecte, 3 motivació mitja i 5 alta motivació.
- Per al factor E6, 0 significa requeriments extremadament inestables, 3 estabilitat mitja i 5 requeriments estables sense possibilitat de canvis.
- Para el factor E7, 0 significa que no hi ha personal part-time (es a dir tots son full-time), 3 significa meitat i meitat, i 5 significa que tot el personal es part-time (ningú és full-time).
- Per al factor E8, 0 significa que el llenguatge de programació es fàcil d'utilitzar, 3 mitja i 5 que el llenguatge és extremadament difícil.

El Factor d'ambient es calcula mitjançant la següent equació:

$$\mathbf{EF = 1,4 - 0,03 \times \Sigma(Pes_i \times Valor\ assignat_i)}$$

Exemple

Continuant amb l'exemple del sistema d'assignació d'ordres de compra, es calculen els punt de casos d'us ajustats.

Factor de complexitat tècnica (TCF)

Factor	Descripció	Pes	Valor assignat	Comentari
T1	Sistema distribuït	2	0	El sistema és centralitzat
T2	Objectius de performance o temps de resposta	1	1	La velocitat és Limitada per les Entrades de l'usuari
T3	Eficiència de l'usuari final	1	1	Escasses restriccions D'eficiència
T4	Processament intern complex	1	1	No hi ha càlculs Complexes
T5	El codi ha de ser reutilitzable	1	0	No es requereix que El codi ho sigui
T6	Facilitat d'instal·lació	0,5	1	Escassos requeriments de facilitat d'instal·lació
T7	Facilitat d'us	0,5	3	Normal
T8	Portabilitat	2	0	No es requereix
T9	Facilitat de canvi	1	3	Es requereix un cost moderat de manteniment
T10	Concurrència	1	0	No n'hi ha
T11	Inclou objectius especials	1	3	Seguretat normal

	de seguretat			
T12	Proveeix accés directe a tercers	1	5	Els usuaris web tenen accés directe
T13	Es requereixen facilitats especials d'entrenament A usuaris	1	1	Pocs usuaris interns, sistema fàcil d'utilitzar

Tabla 18:Factors de complexitat tècnica

El factor de complexitat tècnica resulta:

TCF = 0,6 + 0,01 x 17 = 0,77

Factor d'ambient (EF)

Factor	Descripció	Pes	Valor Assignat	Comentari
E1	Familiaritat amb el model de projecte utilitzat.	1,5	4	El grup està bastant familiaritzat.
E2	Experiència en la aplicació.	0,5	4	La majoria del grup a treballat molt de temps amb l'aplicació.
E3	Experiència en orientació a objectes.	1	4	La majoria del grup en te.
E4	Capacitat de l'analista líder.	0,5	5	És un especialista
E5	Motivació.	1	5	El grup està altament motivat
E6	Estabilitat dels requeriments.	2	2	S'esperen canvis
E7	Personal part-time.	-1	0	Tot el grup és full-time
E8	Dificultat del llenguatge de programació.	-1	3	S'utilitzarà C++

Tabla 19:Factors d'ambient

El factor ambient resulta:

EF = 1,4 – 0,03 x 19,5 = 0,82

Finalment, els punts de casos d'us ajustats resulten:

UCP = 23 x 0,77 x 0,82 = 14,52

4.8 Mètrica bang

És una mesura de la funcionalitat proposada per DeMarco, 1978, 1982. Per a calcular aquesta mètrica s'han d'avaluar un conjunt de primitives:

- Primitives funcionals (PFu)
- Elements de dades (ED)
- Objectes (OB)
- Relacions (RE)
- Estats (ES)
- Transicions (TR)

S'han de determinar comptes addicionals per:

- Primitives modificades de la funció manual (PMFu)

- Elements de dades d'entrada (EDE)
- Elements de dades de sortida (EDS)
- Elements de dades retingudes (EDR)
- Mostres (*tokens*) de dades (TC_i)
- Connexions de relació (RE_i)

Les mètriques *bang* o mètriques de pes de la especificació es calculen per a dos dominis:

- **Domini de la funció**: Mesura del número de primitives funcionals dels diagrames de flux.
- **Domini de les dades**: Mesura basada en el número d'entitats i relacions del diagrama entitat - relació.

RE/PFu < 0,7	Aplicació de domini de funció
0,8 < RE/PFu < 1,4	Aplicació híbrida
RE/PFu > 1,5	Aplicació de domini de dades

DeMarco suggereix que s'utilitzi una mitja de mostra (token) per primitiva:

$$TC_{media} = \Sigma TC_i/PFu$$

Les mètriques es calculen a partir d'increments PFu i OB corregits (IPFuC e IOBC):

TCi	IPFuC
2	1,0
5	5,8
10	16,6
15	29,3
20	43,2

Rei	IOBC
1	1,0
3	4,0
6	9,8

Tabla 20:Calcul métrica Bang

Finalment les mètriques *bang* poden definir-se formalment i ser calculades automàticament mitjançant eines CASE.

4.9 l'estàndard IEEE per a mètriques de productivitat

4.9.1 Introducció.

L'estàndard IEEE per a mètriques de productivitat té les següents característiques:

- No estableix normes de productivitat.
- No recomana mesures de productivitat com a mètode per avaluar els projectes de programari.
- No mesura la qualitat del programari.
- No pretén augmentar la productivitat, només mesurar-la.
- Defineix una forma consistent de mesurar els elements de la productivitat del programari.
- Recomana mesures per a caracteritzar el procés del programari.

Les mètriques de productivitat, depenen de la validesa de les dades utilitzades per al seu càlcul, també és molt important establir el nivell de detall en el moment de recol·lectar les dades per a determinar la precisió i la consistència.

4.9.2 Dades per mesurar les sortides.

L'estàndard IEEE proposa tres categories: línees de codi, punts de funció (opcional) i documents. Per a cadascuna d'aquestes categories proposa els mètodes de mesurament.

I Línees de codi

Quan es compten línees de codi s'han de tenir presents les següents recomanacions:

- S'ha de comptar cada llenguatge per separat, fins i tot dins del mateix projecte.
- Només s'ha de comptar el codi generat per persones.
- S'ha de comptar quan el producte estigui acabat.
- Les macros només es compten un cop.
- Les crides a codi s'han de comptar cada cop que s'utilitzin.

En aquest cas es proposen dos mètodes per a comptar:

- LSS (Logical source statement method). Per comptar instruccions.
  - Es compta el número d'instruccions del programari.
  - S'ha d'acompanyar amb el procediment per a de comptar LDC
  - S'han de reportat comptes separades per llenguatges, macros i crides a codi.
- PSS (Phisical source statement method). Per comptar línees de codi.
  - Es compta el número de línees del programari.
  - 1 línea = 80 caracters.
  - No s'han de comptar les línees en blanc.
  - S'ha de comptabilitzar per separat les línees de comentari.

Els càlculs a utilitzar per a les modificacions son els següents:

- #LDC noves = #LDC afegides + #LDC modificades.
- #LDC esborrades = #LDC modificades + #LDC tretes del programa.
- #LDC reutilitzades = #LDC originals - #LDC modificades - #LDC tretes del programa.

II Punts de funció.

La utilització d'aquesta mètrica és opcional, però si s'utilitza ha d'anar acompanyada de línees de codi per a fer relacions i comparar. És necessari documentar l'algoritme per a calcular els PF o indicar font i versió (per exemple IFPUG 4.2), el tipus de funció que s'utilitza (EI, EO, ..) i la llista de valors utilitzats per a cada tipus.

Com es pot observar l'estàndard IEEE no proposa cap mètode per a calcular PF, deixant oberta la elecció d'algun mètode existent.

III Documents.

S'han de mesurar tots els documents que consumeixin una quantitat no trivial de recursos, tenint en compte papers, imatges, texts o gràfics que s'utilitzin per a proveir d'informació a una o varies persones, incloent-hi els documents que donen suport al desenvolupament o la utilització del programari i els documents que es generen per la pròpia dinàmica del procés del programari, com poden ser calendaris, pressupostos, etc..

La unitat per a mesurar els documents és la plana o pantalla, sense comptar les planes en blanc. El resultat ha de ser un número sencer. Com a complement s'ha de reportar també per a cada plana o pantalla, el seu tamany i el número de paraules, ideogrames i gràfics que conté.

A mes de fer les mesures corresponents, és important guardar els atributs del document, que son: el seu tipus, nom i propòsit del document, l'origen, el percentatge de reutilització ((#tokens sense modificar / #tokens total)*100), la utilització i la disponibilitat, és a dir si estarà disponible nomes per als desenvolupadors o pel usuaris, etc..

4.9.3 Dades per mesurar les entrades

Com ha entrades s'ha de mesurar l'esforç, representat per hores - persona, l'estàndard requereix que es reculli la informació de l'esforç en el moment de la seva inversió, incloent-hi hores extra de treball siguin o no remunerades, només es comptaran les hores dedicades al projecte i s'ha d'especificar la conversió a unitats mes grans d'hores persona.

Els atributs d'hores persones seran les HP directes, tant les entregades a l'usuari com les no entregades, les HP de suport (implantació, formació, suport telefònic). Les activitats relacionades amb el cicle de vida del programari no estan definides per l'estàndard IEEE

4.9.4 Resultats de productivitat

L'estàndard proposa les següents mètriques per a obtenir els resultats de productivitat:

- **Relació de productivitat**.
Mostra la relació del producte de sortida respecte a l'esforç necessari per a produir-lo.

$$Productivitat = \frac{O}{E}$$

O: LDC, PF, Docs
E: HP

La utilitat dels valors de productivitat depèn de les dades utilitzades.

- **Relacions de sortida**.
A partir de la següent taula de relació sortida - sortida es poden calcular les proporcions de sortida.

		Utilització	
		Entregable	No Entregable
Origen	Desenvolupat	LOC Noves (1)	LOC Noves (2)
	No Desenvolupat	LOC Reutilitzades (3)	LOC Reutilitzades (4)
		LOC Esborrades (5)	LOC Esborrades (6)

Tabla 21:Relacions de sortida

$$EntregatNou = \frac{(1)}{(1)+(3)}$$

$$Reutilització = \frac{LCReutilitzat}{LCTotal} = \frac{(3)+(4)}{(1)+(2)+(3)+(4)} \qquad EntregatReutilitzat = \frac{(3)}{(1)+(3)}$$

$$Entregat = \frac{LCEntregades}{LCTotal} = \frac{(1)+(3)}{(1)+(2)+(3)+(4)} \qquad NoEntregatReutilitzat = \frac{(4)}{(2)+(4)+(6)}$$

- **Altres mètriques de sortida.**
  - LDC Executables.
  - LDC Declaració de dades.
  - LDC Directives de compilador.
  - LDC Comentaris.
  - Proporció de LDC executables per mòdul.
  - Proporció de comentaris per mòdul.
  - El mateix per a la documentació

- **Relacions d'entrada.**
  - HP Directes entregades (A).
  - HP Directes no entregades (B).
  - HP de suport (C).
  - Esforç total = A+ B + C.
  - Esforç directe total = A + B**.**
  - Relació d'esforç directe al total.
  - Relació d'esforç de suport al total.
  - Relació d'esforç directe entregat al no entregat.

5 De la estimació del tamany a la estimació de l'esforç

5.1 Punts de Funció Ajustats i Coeficients de Conversió

El primer d'aquests mètodes consisteix en el càlcul dels Punts de Funció Ajustats, seguint el procediment indicat per l'anàlisi de Punts de Funció, que consisteix en el càlcul d'un Factor d'ajustament (vist a l'apartat 4.5.3 El mètode estàndard d'anàlisi de punts per funció) en base a la quantificació de certs coeficients vinculats amb les característiques desitjades del sistema (comunicació de dades, rendiment, facilitats d'instal·lació, d'operació, freqüència de transaccions, etc.).
Per il·lustrar el procés ens basarem en un sistema d'administració d'ordres de compra web que permet les següents característiques:

- Afegir ordre: permet que l'usuari efectuï l'alta d'una ordre de compra al sistema. (Entrada externa).
- Trobar ordre: permet que l'usuari trobi una ordre de compra al sistema. (Consulta externa).
- Modificar ordre: permet que l'usuari modifiqui una ordre de compra en el sistema. (Entrada externa).
- Eliminar ordre: permet que l'usuari elimini una ordre de compra del sistema. (Entrada externa).

En totes les opcions del sistema està implícita la presencia d'un Arxiu Lògic Intern, on s'emmagatzema la informació de les ordres de compra.

D'acord amb l'anàlisi de Punts de Funció, tant les Transaccions (Entrades Externes, Sortides Externes, Consultes Externes) com els Arxius (Arxius Lògics Interns, Arxius de Interfase Externs) han de ser classificats amb una complexitat Baixa, Mitja o Alta, en aquest cas s'agafarà una complexitat mitja.

Aleshores:

- 3 entrades externes de complexitat mitja
- 1 consulta externa de complexitat mitja
- 1 arxiu lògic intern de complexitat mitja

Complexitat				
	Baixa	Mitja	Alta	Aportació
Entrades externes		3		12
Sortides externes				
Consultes externes		1		4
Arxius lògics interns		1		10
Arxius d'interfase externs				
Total	26			

Tabla 22: Transaccions i arxius

Sumant les aportacions de tots els elements s'obtenen els Punts de Funció sense ajustar:

UFP (Punts de Funció sense ajustar) = 12 + 4 + 10 = 26

Càlcul del Factor d'ajust:

Característica	Descripció	Pes
Comunicació de dades	Aplicació web	3
Processament distribuït De dades	No n'hi ha, però Hi ha dades distribuïdes	2
Rendiment	No hi ha requeriments Especials de rendiment	0
Configuracions utilitzades De manera forta	No hi ha restriccions De maquinari	0
Freqüència de transaccions	Pic diari de transaccions	3
Entrada de dades on-line	Totes les dades S'actualitzen on-line	5
Eficiència de l'usuari final	Mitja	3
Actualitzacions on-line	La majoria dels arxius S'actualitzen on-line	3
Processament complex	No n'hi ha	0
Reutilització	Es pretén algun grau de reutilització	2
Facilitat d'instal·lació	No hi ha restriccions	0
Facilitat d'operació	Operació desatesa	5
Instal·lació en llocs diferents	No es requereix mes D'una instal·lació	0
Facilitat de canvi	Mitja	3

Tabla 23: Factors d'ajust

Amb els valors assignats a les característiques, s'obté el Grau Total de Influencia com:

TDI = 3 + 2 + 0 + 0 + 3 + 5 + 3 + 3 + 0 + 2 + 0 + 5 + 0 + 3 = 29

i el Factor d'ajust com:

AF = TDI x 0,01 + 0,65 = 29 x 0,01 + 0,65 = 0,94

Finalment, els Punts de Funció Ajustats donen:

FP = UFP x AF = 26 x 0,94 = 24,44

Després d'obtenir els Punts de Funció Ajustats, es poden aplicar coeficients que transformin aquest valor a d'altres com l'esforç, el cost o el temps. Aquests coeficients s'obtenen fonamentalment de la informació històrica de projectes de la organització, encara que existeixen alguns valors mitjans disponibles, recopilats estadísticament de la industria del software.

Exemple

Aplicant com a valor de conversió que 1PF = 20 hores – home, s'obté l'esforç necessari per al desenvolupament del programari:

E = 24,44 x 20 = 488,8 hores - home

Si a mes es considera que aquest esforç representa un percentatge de l'esforç total del projecte, suposant els valors (anàlisis 10%, disseny 20%, programació 40%, proves 15% i altres activitats 15%), s'obté:

Activitat	Percentatge	Hores – home
Anàlisis	10%	122,2
Disseny	20%	244,4
Programació	40%	488,8
Proves	15%	183,3
Sobrecàrrega (altres activitats)	15%	183,3
Total	100%	1222

Tabla 24: Distribució de l'esforç

5.2 Aplicació de COCOMO II

El segon dels mètodes possibles es l'aplicació del mètode COCOMO II directament sobre els Punts de Funció sense ajustar o sobre les línies de codi en cas que es coneguin a priori. Aquest mètode es el preferit a l'actualitat per a la estimació de l'esforç quan no es te informació històrica a la qual recórrer.

COCOMO II consisteix bàsicament a l'aplicació d'equacions matemàtiques sobre els Punts de Funció sense ajustar o sobre la quantitat de línies de codi (SLOC, Source Lines Of Code) estimats per a un projecte. Aquestes equacions es troben ponderades per certs factors de cost (cost drivers) que influeixen en l'esforç requerit per al desenvolupament del programari. L'annex 1 presenta una breu introducció al mètode i el seus conceptes principals.

A manera d'exemple, s'aplica el mètode COCOMO II a la estimació inicial obtinguda en l'apartat anteriors. Com a resultat de la estimació teníem:

UFP (Punts de Funció sense ajustar) = 26

Per a aplicar la equació de càlcul de l'esforç nominal (equació 1 de l'annex 1), necessitem per una banda convertir els punts de funció sense ajustar a
KLOC (línies de codi, en milers), i per altra calcular el Factor escalar B d'acord a les característiques del projecte. A les hores:

$$PM_{nominal} = A \times (size)^{B}$$

A) prenem el valor per defecte del modelo, ajustat en 2.94

Size: es calcula com el producte dels punts de funció sense ajustar per un factor de conversió que depèn del llenguatge a utilitzar en el desenvolupament del sistema. Suposem que utilitzem C++ (factor de conversió = 53 LOC/UFP). Aleshores tindrem:

Size = 53 x 26 = 1378 LOC

B) es calcula ponderant les variables escalars d'acord al model COCOMO II, mitjançant la equació:

$$B = 0{,}91 + 0{,}01 \times \Sigma(W_i)$$

On les W_i es mostren a la següent taula:

Variable	Descripció	Ponderació	Valor
PREC	El sistema es molt familiar	Molt alt	1,24
FLEX	Una mica de relaxació Quant a la flexibilitat del Desenvolupament	Normal	3,04
RESL	Arquitectura sòlida i amb	Alt	2,83

	Pocs riscos		
TEAM	La interacció de l'equip és Altament cooperativa	Molt alt	1,10
PMAT	La maduresa del procés De programari és baixa	Baix	6,24
	Total	1,05	

Tabla 25: Variables escalars

Finalment, l'esforç nominal resulta:

$$\mathbf{PM_{nominal} = A \times (size)^{B}} = 2{,}94 \times (1{,}378)^{1{,}05} = 4{,}11 \text{ Mesos - home}$$

Per a completar la estimació, s'ha d'ajustar l'esforç nominal d'acord a les característiques del projecte segons el mètode COCOMOII. L'ajust s'efectua aplicant la equació:

$$\mathbf{PM_{ajustat} = PM_{nominal} \times \Pi(ME_i)}$$

On els $\mathbf{ME_i}$ (multiplicadors de l'esforç) varien en funció del model d'estimació seleccionat (Disseny Preliminar o Post arquitectura).
En aquest cas s'aplicarà el model de Disseny preliminar. Aleshores, quantifiquem els multiplicadors d'esforç per aquest model:

Multiplicador	Descripció	Ponderació	Valor
PERS	Se disposa d'analistes i programadors amb alta eficiència i capacitat de treball en equipo. Dedicació full-time.	alt	0,83
RCPX	Les exigències de confiabilitat, documentació i volum de dades son moderades, i la complexitat del producte es baixa.	Normal	1
RUSE	No es pretén reutilitzar res	Baix	0,95
PDIF	No existeixen restriccions en quant al temps de CPU o al consum de memòria, la plataforma es molt estable.	Baix	0,87
PREX	Tant els analistes como els programadors tenen aproximadament 6 mesos d'experiència en la aplicació, la plataforma, el llenguatge i les eines utilitzades.	Molt baix	1,33
SCED	Es requereix terminar el projecte en el temps estimat.	Normal	1
FCIL	Es disposa d'eines CASE simples e infraestructura de comunicacions bàsica.	Baix	1,10
	Total	1,004	

Tabla 26: Multiplicadors de l'esforç

Amb aquests valors, l'ajust de l'esforç resulta:

$$\mathbf{PM_{ajustat} = 4{,}11 \times 1{,}004 = 4{,}13 \text{ Mesos - home}}$$

Expressant el mateix valor en Hores - home, i tenint en comte que un mes es aproximadament 160 hores, l'esforç resulta:

4,13 x 160 = 660,8 Hores - home

Si s'aplica la mateixa distribució que abans per a la distribució del projecte, s'obté la següent taula:

Activitat	Percentatge	Hores – home
Anàlisis	10%	165,2
Disseny	20%	330,4
Programació	40%	660,8
Proves	15%	247,8
Sobrecàrrega (altres activitats)	15%	247,8
Total	100%	1652

Tabla 27: Distribució de l'esforç

5.3 Dels punts de casos d'us a la estimació de l'esforç

Karner originalment va suggerir que cada Punt de Casos d'us requereix 20 hores - home. Posteriorment, van sorgir altres afinaments que proposen una granularia una mica mes fina, segons el següent criteri:

- Es compta quants factors dels que afecten al factor d'ambient estan per sota del valor mitjà (3), per als factores E1 a E6.
- Es compta quants factors dels que afecten al Factor d'ambient estan per sobre del valor mitjà (3), per als factores E7 y E8.
- Si el total es 2 o menys, s'utilitza el factor de conversió 20 hores - home/Punt de Casos d'us, es a dir, un Punt de Cas d'us pren 20 hores - home.
- Si el total es 3 o 4, s'utilitza el factor de conversió 28 hores - home/Punt de Casos d'us, es a dir, un Punt de Cas d'us pren 28 hores - home.
- Si el total es major o igual que 5, es recomana efectuar canvis en el projecte, ja que es considera que el risc de fracàs del mateix és massa alt.

També és important considerar l'experiència dels projectes anteriors per a les estimacions futures.

L'esforç en hores - home ve donat per:

E = UCP x CF

On,
E: esforç estimat en hores - home
UCP: Punts de Casos d'us ajustats
CF: factor de conversió

S'ha de tenir en consideració que aquest mètode proporciona una estimació de l'esforç en hores - home contemplant nomes el desenvolupament de la funcionalitat especificada en els casos d'us.

Finalment, per a una estimació mes completa de la duració total del projecte, es necessari agregar a la estimació de l'esforç obtinguda pels Punts de Casos d'us, les estimacions d'esforç de les demés activitats relacionades amb el desenvolupament de programari. Amb aquest objectiu es pot tenir en compte el següent criteri, que estadísticament es considera acceptable. El criteri planteja la distribució de l'esforç entre les diferents activitats d'un projecte, segons la següent aproximació:

Activitat	Percentatge
Anàlisi	10%
Disseny	20%
Programació	40%
Proves	15%
Sobrecàrrega (altres activitats)	15%

Tabla 28:Distribució de l'esforç

Òbviament, aquests valors no son absoluts si no que poden variar d'acord a les característiques de l'organització i del projecte.

Amb aquest criteri, i prenent com entrada la estimació de temps calculada a partir dels Punts de Casos d'us, es poden calcular la resta d'estimacions per a obtenir la duració total del projecte.

Exemple

Aplicant aquests criteris a l'exemple del sistema d'assignació d'ordres de compra, s'obté l'esforç necessari per al desenvolupament dels casos d'us com:

E = 14,52 x 20 = 290,4 hores - home

Si a mes es considera que aquest esforç representa un percentatge de l'esforç total del projecte, d'acord als valors percentuals de la taula anterior, s'obté:

Activitat	Percentatge	Hores – home
Anàlisis	10%	72,6
Disseny	20%	145,2
Programació	40%	290,4
Proves	15%	108,9
Sobrecàrrega (altres activitats)	15%	108,9
Total	100%	726

Tabla 29:Resultats d'esforç distribuit

6 Disseny d'una mètrica pròpia

6.1 Definicions i conceptes

El primer pas per dissenyar una mètrica és donar una definició estàndard per a cada entitats i els seus atributs a mesurar així com per a tots els elements de comunicació entre els diferents actors que participen al projecte.

Quan s'utilitzen termes com "errors", "defectes", "problema reportat", "tamany" o "projecte", cada persona pot interpretar aquestes paraules dins el seu propi context i donar-les significats que és poden allunyar del significat que els volia atorgar la persona que els va definir inicialment. Aquestes interpretacions augmenten les seves diferencies quan els termes empleats son encara mes ambigus com reutilització, implantació, producció i d'altres.

Els enginyers, programadors i d'altres persones involucrades al projecte poden emprar diferents termes per a referir-se a la mateixa entitat. Per exemple, el terme defecte reportat, problema reportat, incident reportat o report de trucada del client poden ser utilitzats per diverses persones o organitzacions per definir la mateixa cosa. Però també es poden referir a entitats distintes. Per evitar problemes i malentesos deguts a aquesta situació es suggereix el següent:

- Adoptar definicions estàndards i documentar-les.
- Aplicar-les amb consistència.
- Utilitzar els suggeriments de la industria o els estàndards preexistents.
- Triar definicions que s'adaptin als objectius i a l'entorn de treball.

Dissortadament no existeixen molts estàndards a la industria per a la majoria de les definicions dels atributs del programari. Actualment existeixen moltes opinions i una mica de literatura sobre el tema però no sembla que estigui a prop la definició d'un estàndard.

L'actitud mes coherent és la de crear un estàndard propi per a la organització i aplicar aquest estàndard de manera consistent, per a definir aquest estàndard es poen aprofitar suggeriments de la industria per començar, seleccionar les definicions que mes s'adaptin a les necessitats de l'organització, adaptar aquelles que es puguin utilitzar amb variacions i crear-ne de noves per als conceptes propis o específics de l'organització.

6.2 Creació d'un equip de mètriques

L'equip de mètriques consta d'una o mes persones físiques que estan compromeses en el desenvolupament del programari. Normalment les seves tasques implicaran nomes una petita porció del seu temps i podran ser compatibilitzades amb la seva feina habitual. Depenent de la extensió del programari, una o dues persones podrien ser assignades al programa de mètriques "full time". Aquestes persones actuaran com a coordinadors i tindran la tasca específica d'establir el programa de mètriques i portar-lo a terme.

Les següents tasques seran responsabilitat de l'equip de mètriques:

- Implementar el programa.

- Mantenir al personal de la companyia informat respecte al programa i els seus avenços.
- Coordinació de la recol·lecció de dades.
- Anàlisi de les dades recol·lectades.
- Reportar periòdicament les dades als gestors.
- Retroalimentar amb informació els equips de projecte.
- Ajudar en la implantació de les millores i objectius resultants del pla de mètriques

6.3 Definició d'un model

El següent pas es definir un model per a la mètrica. En termes senzills, el model defineix com s'ha de calcular la mètrica. Per exemple, per a la inspecció del codi de les mètriques primitives, es poden utilitzar els següents models:

- Línies de codi inspeccionades = ldc
- Hores emprades en la preparació = hrs-prep
- Mesura de preparació = ldc/hrs-prep

La major part dels models inclouen un element de simplificació. Quan es crea un model de medició de programari s'ha de ser pragmàtic. Si es tracta d'incloure tots els elements que afecte l'atribut o que caracteritzen la entitat, el model utilitzat arribarà a ser massa complicat i la seva pròpia execució es pot convertir en un nou projecte. Ser pragmàtic significa no tractar de crear un model perfecte, sinó saber reconèixer els aspectes mes importants i tenir en compte que el model sempre pot ser revisat en el futur per afegir mes nivells de detall.

Existeixen dos mètodes per a la selecció d'un model:

- En molts cassos no és necessari ser massa innovador. Existeixen molts models de mètriques de programari que son empleats exitosament per altres organitzacions, dissortadament la majoria dels models creats per les empreses no son públics, però en moltes ocasions és possible trobar un que sigui escaient per a cobrir les necessitats.
- El segon mètode es crear un model propi.

6.4 Establir un criteri per contar

El següent pas en el disseny d'una mètrica es dividir el model en les seves mètriques primitives (quantificables en forma directa) i la definició del criteri de conteig per a mesurar cada primitiva.

Les mètriques primitives i el seu criteri de conteig defineixen el primer nivell de les dades que s'han de recol·lectar per a la implantació de la mètrica. Com a exemple utilitzarem el model de línies de codi desenvolupades per hores del personal invertides durant un projecte. Una de les mètriques primitives per a aquest model és el número d'hores dedicades. D'entrada el conteig d'aquesta primitiva sembla senzill, però quan considerem la dinàmica de afegir o treure personal del projecte o es modifica la duració en hores d'una jornada, el criteri per a contar es torna mes complex. Així doncs s'ha de decidir si s'utilitza el número de persones involucrades al final del projecte o una mitjana de participació durant tot el projecte, o si es compta una jornada com a 8 hores dedicades o es fa un promig de les hores diàries dedicades per tots els desenvolupadors,.. De qualsevol forma és necessari conèixer les dates d'incorporació o d'exclusió dels diferents

desenvolupadors al projecte si es vol comparar la productivitat a diferents períodes. Un exemple dels possibles criteris a utilitzar seria el següent:

- Número d'hores - persones al finalitzar el projecte.
- Números d'hores – persones al començament + Número d'hores persones al final / 2
- Ó (suma d'hores – persona cada dia)/número de dies.

6.5 Definir l'escala de valors

Aquest pas en el disseny d'una mètrica consisteix en termes senzills en definir "que es bo". Un cop decidit que s'ha de mesurar i com s'ha de mesurar, s'ha de decidir que fer amb els resultats. És 10 massa poc? És 100 massa? Quins ha de ser els límits superiors e inferiors?

Una font d'informació important per a definir l'escala de valors és la literatura de mètriques preexistents. Existeixen investigacions i estudis que han ajudat a l'establiment de normes acceptades per la industria per a mesuraments estàndard. Si no existeixen dades històriques, és molt recomanable esperar fins a tenir recol·lectada la suficient informació com per a poder definir una escala de valors raonable i útil.

Un cop establerta l'escala de valors, es poden determinar en tot cas les accions necessàries si s'estan superant els límits establerts o revisar l'escala en cas que no sigui necessari prendre mesures correctives per a uns valors fora d'especificació.

Per a determinar el control, gestió i monitoratge d'activitats de millora, es determinarà en el pla de mètriques un ambient on s'hauran de tenir presents quatre Conceptes:

- La meta ha de ser raonable; És correcte establir metes extensibles i ampliables, però si en algun moment s'arriba a superar el límit de les possibilitats i els recursos, aquesta meta es torna irreal i s'ha d'ignorar.
- La meta ha d'estar associada a un marc de temps.
- La meta s'ha de fonamentar en accions sustentades. Per exemple podria ser raonable establir una meta d'un increment del 50% en la productivitat de codi en llenguatge Java, si es crea un equip especial de desenvolupadors especialistes en el desenvolupament de programari en Java i que siguin grans coneixedors d'aquest llenguatge.
- La meta s'ha de dividir en fases. Per exemple si es dona un període d'un any per a assolir una millora, seria mes fàcil si la mateixa meta és divideix en 4, 6 o 12 petites metes establertes en períodes de temps mes petits, augmentant les probabilitats d'èxit tant de les accions intermèdies com de la meta en el seu conjunt.

6.6 Definició de la presentació

Aquest pas consisteix a decidir com reportar i presentar la mètrica. Això inclou la definició del format de report, la obtenció de les dades, els mecanismes de report, distribució i disponibilitat.

El format de report es dissenya segons el que es vulgui donar a conèixer, és per aquest motiu que s'ha de preguntar el següent, està la mètrica inclosa en una taula amb d'altres mètriques per a un període o projecte? S'afegeix com a últim valor a una gràfica de tendències que mostra els valors de la mètrica per a varis períodes o projectes? Aquesta

gràfica de tendència, serà de barres, línies, o àrees? És millor la comparació mitjançant barres apilades o gràfiques de pastís? És millor fer només les taules i gràfiques o s'ha d'afegir als informes texts explicatius sobre l'anàlisi i la obtenció de valors? Son metes o valors de control els que s'inclouen als informes?

En el report de les mètriques s'hauran de seguir els següents quatre passos: cicle d'obtenció de dades, cicle de report de dades, mecanismes de report i finalment distribució i disponibilitat. A continuació es descriuen amb mes detall aquests quatre passos:

- El cicle d'obtenció de dades defineix amb quina freqüència requereix la mètrica "snap-shots" de les dades i en quin moment estan disponibles les dades per a fer els càlculs necessaris.
- El cicle de report defineix amb quina freqüència es generen els informes i quan es preparen per a la seva distribució. Per exemple la raó principal d'un estudi de mètriques pot ser motivada per algun succés, com la terminació d'una fase en el procés de desenvolupament de programari. Altres mètriques poden ser obtingudes i reportades diàriament, setmanalment o mensualment i és en aquest cicle on s'han de definir aquestes freqüències.
- Els mecanismes de report son la projecció dels mecanismes d'entrega de les mètriques.
- La definició de la distribució a de determinar qui rebrà copia del informes o qui te accés a la mètrica. També és en aquest pas on es defineixen les restriccions d'accés a determinades dades o parts de la mètrica i els diferents mecanismes d'aprovació per afegir o denegar accessos i canvis en el procés normal de distribució.

6.7 Altres qualificadors

El pas final en el disseny d'una mètrica és el de determinar qualificadors addicionals. Una bona mètrica és una mètrica genèrica. Això significa que la mètrica es vàlida per a tots els nivells de qualificadors que sigui necessari afegir.

Els qualificadors afegits proveeixen la informació estadística necessària per a varis punts de vista de la mètrica.

La raó principal per la qual els qualificadors addicionals han de ser definits com a part del disseny de la mètrica és que aquests determinen el segon nivell dels requeriments de la recol·lecció de dades. Cap discussió de selecció i disseny de mètriques seria complerta sense tenir en compte com els mesuraments afecten les persones i com les persones afecten els mesuraments.

El simple fet de mesurar afectarà l'ambient dels individus que seran mesurats. Quan algú comença a ser mesurat assumeix automàticament que te importància dins de l'organització. Totes les persones que col·laboren en una organització volen ser ben vistos i valorats positivament, per tant intentaran que els mesuraments que els afecten siguin bons. Quan es crea una mètrica és molt important de decidir quin ambient es vol crear i després s'ha de dedicar un temps a estudiar quin es l'ambient resultant de la utilització de mètriques i tots els aspectes secundaris no previstos, sobre tot els negatius per intentar eradicar-los.

La millor manera de prevenir problemes amb el factor humà a l'hora de treballar amb mètriques és seguir algunes de les següents regles bàsiques:

- No fer mesuraments de l'individuo: Els mesuraments de productivitat individuals son els exemples clàssics d'aquests errors. Si és mesures a nivell individual la productivitat en línies de codi per hora produïdes, els desenvolupadors es concentrarien en el seu propi treball i exclourien l'equip de treball, o es concentrarien en realitzar programació amb línees extra de codi innecessàries. És per aquest motiu que es recomana enfocar les mètriques en el producte o en el projecte i no en les persones.
- No ignorar les dades: Una via que segurament acabarà amb qualsevol programa de mètriques és la de oblidar les dades quan es prenen decisions, ja que son aquestes dades les que indican la situació de la companyia i suggereixen la idoneïtat o no de les decisions. Si les metes que s'estableixen i es comuniquen no son recolzades per les dades i amb accions de suport, es crea una sensació de desorientació en els empleats de l'organització, que realitzaran la seva feina basant-se en l'ambient i situacions actual i no en les metes i objectius.
- Mai utilitzar una única mètrica: El programari és complex i polifacètic i és per això que l'enfocament en una sola mètrica pot causar que l'atribut mesurat millori a costa d'altres atributs no mesurats. La recomanació és la següent:

    - Seleccionar les mètriques basant-se en objectius: Per a tenir mètriques que acompleixin amb les necessitats existents d'informació, s'han de seleccionar mètriques que donin resposta a les preguntes.
    - Proveir retroalimentació: La provisió de retroalimentació amb regularitat proporciona alguns beneficis com els següents:
        - Mantenir enfocada la necessitat de la recol·lecció de dades, farà que l'equip vegi que les dades actuals son emprades i per tant es tornaran conscients de la importància de la seva recol·lecció.
        - Si els membres dels equips de treball son informats regularment de com i per a que s'estan utilitzant les dades, ells seran cada cop menys escèptics sobre la utilitat de la recol·lecció de les dades.
        - Si s'involucra els membres de l'equip en l'anàlisi de les dades i en els esforços de millora del procés, el procés es beneficiarà dels seus coneixements i experiència.
        - La retroalimentació en la recol·lecció de dades i en el resultat final ajudarà en la responsabilitat de la recol·lecció, donant com a resultat dades mes exactes, consistents i a temps.
    - Obtenir implicació de l'equip: Per a obtenir un compromís tant en les metes com en les mètriques, els membres de l'equip necessiten tenir un sentiment de propietat, es per aquest motiu que la participació en la definició de les mètriques per part dels membres de l'equip farà encara mes fort aquest sentiment de propietat. La gent qui treballa amb un procés diàriament, te un coneixement íntim del procés, això dona una perspectiva valuosa de com el procés es pot mesurar millor i com es poden utilitzar els resultats dels mesuraments per a maximitzar els seus beneficis.

7 Conclusions

La majoria dels desenvolupadors de programari encara no mesuren o no utilitzen les mètriques correctament, i desgraciadament aquesta situació no sembla que estigui molt a prop de canviar, la majoria de les companyies dedicades al desenvolupament de programari no s'adapten a cap estàndard, com pot ser la ISO, i no només en quant a mètriques de productivitat es refereix, sinó també a altres temes molt importants com la qualitat (Font PCWorld Oct. 2004). En un intent de recopilar mesures en un entorn on no s'hagi recopilat res anteriorment, és freqüent que s'oposi resistència i sorgeixin preguntes com: "Per que es necessària aquesta tasca?", es poden preguntar els gestors de projectes. Per que és tan important mesurar el procés de la enginyeria del software i el producte (programari) que produeix? La resposta és realment obvia. Si no es mesura, no hi ha una manera real de determinar si s'està millorant la productivitat, i si no s'està millorant, s'està perdent encara que no s'estigui empitjorant.

Els gestors de projecte i inclús els directius, dins de les companyes dedicades al desenvolupament de programari, poden establir objectius significatius de millora del procés d'enginyeria del programari sol·licitant i avaluant les mesures de productivitat del programari. El programari és un aspecte de administració estratègic per a moltes companyes, per tant si el procés de desenvolupament de programari pot ser millorat es pot produir un impacte directe en la millora de la companya.

El procés habitual dels projectes de programari no deixa massa temps per a rumiar estratègies. Els administradors dels projectes estan mes centrats en temes com la producció de sistemes de gran qualitat, la distribució adequada dels recursos i sobre tot terminar el projecte a temps. Mitjançant la utilització dels programes de mètriques per a establir una línea base del projecte, és mes fàcil d'acomplir els objectius de productivitat i per tant si existeix un bon coneixement de la productivitat temes com l'estimació de la durada del projecte son mes fàcils de manegar.

Quan s'apliquen mètriques de productivitat al producte del programari, s'obtenen beneficis immediats, ja que quan es termina la fase de disseny del programari, la planificació és molt mes senzilla gracies a la capacitat de predicció i el major control que ens proporciona la informació obtinguda dels plans de mètriques.

La utilització de mètriques en els projectes de programari ha de començar a canviar per a aplicar-se com a estàndard a mesura que el mercat ho exigeixi. Quan els clients comencin a demanar a les empreses desenvolupadores de programari mètriques i les dades històriques que fonamentin les estimacions en esforç i temps, per a tenir la certesa de que el projecte en que s'està invertint tingui les majors possibilitats de terminar a temps i dins del pressupost, les companyies no tindran mes remei que definir els seus propis plans de mètriques per a ser competitius en el mercat.

8 Glossari

Snap-Shot: Instantània d'una base de dades en un moment concreto del temps.

Token: Unitat sintàctica mes elemental distingible per un compilador.

Backfiring: Tècnica que permet calcular punts de funció en funció de Línees de codi.

full time: Anglicisme que significa temps complert.

part-time: Anglicisme que significa temps parcial.

ISO: "*International Organization for Standardization"* Organització internacional per a l'estandardització.

IFPUG: "International Function Point Users Grup" Grup internacional d'usuaris de punts de funció.

IEEE: "Institute of electrical and electronical engineers" Institut d'enginyers elèctrics i electrònics.

Graf: Un graf està format per un conjunt de nodes o vèrtex i per un conjunt de línies o arcs e manera que cada arc uneix dos nodes.

Node: Cadascun dels vèrtex que composen un graf.

Arc: Cadascuna de les línies que uneix dos nodes d'un graf.

Eines CASE: "Computer-Aided Software Engineering" Eines informatitzades per a la enginyeria del programari

Front-end:Part del programari de la qual es conscient l'usuari i amb la qual es relaciona.

Back-end:Part funcional del programari que s'encarrega de les operacions internes i és transparent per a l'usuari.

MKII: "United Kingdom Software Metrics Association" Associació de mètriques de programari del Regne Unit.

COSMIC: "Common Software Measurement International Consortium" Consorci internacional de mesurament de programari.

Interfase: eina de comunicació e intercanvi de dades entre dos sistemes informàtics o entre un sistema informàtic i l'usuari.

sistemes en temps real:Sistemes que permeten la concurrència i que responen inmediatament i interactuen amb entrades externes, com poden ser les ordres donades per l'usuari.

components de 3GL: Third Generation Language. Els llenguatges de tercera generació son aquells llenguatges de programació utilitzats pels especialistes per a construir aplicacions

que inclouen el procediment. Es a dir, el programador especifica al seu programa que ha de fer l'ordinador i com ho ha de fer. Es tracta d'un pas mes enllà del llenguatge màquina. Son llenguatges de tercera generació Cobol, C, Pascal o Fortran.

SQL: Structured Query Language (Llenguatge de consultes estructurat). Com el seu propi nom indica, SQL es un llenguatge informàtic que es pot utilitzar per a interaccionar amb una base de dades i mes concretament amb un tipus especifico anomenat base de dades relacional.

9 Bibliografia

Dolado, J.J. y Fernández, L. (coordinadores). "Medición para la Gestión en la Ingeniería del Software". Ra-ma, 2000.

Fenton, N.E. y Pfleeger, S.L., Software metrics. A rigorous & practical approach ,1997.

Basili, V.R. y Weiss, D., A Methodology for collecting valid software engineering data, IEEE Transaction on Software Engineering,10(6), 728-38 1984.

DeMarco, T., Structured Analysis and Sistem Specification, Yourdon Press, 1978.

DeMarco, T., Controlling Software Projects, Yourdon Press, 1982.

Halstead, M.H., Elements of Software Science, Elsevier North-Holland, 1977.

IEEE *Software Engineering Standards*,. Standard 610.12-1990, 1993.

ISO/IEC 14143-1 Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts, 2003.

ISO/IEC 19761:2003 Software engineering -- COSMIC-FFP -- A functional size measurement method, 2003.

ISO/IEC 20926:2003 Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting practices manual, 2003.

ISO/IEC IS 24570 Software Engineering - NESMA functional size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis, 2003.

Kitchenham, B., Pfleeger, S.L., and Fenton, N.E., Towards a Framework for Software Measurement Validation, IEEE Transactions on Software Engineering, 21 (12), 929-944, 1995.

Lorenz, M. and Kidd, J., Object_oriented Software Metrics, Prentice Hall 1994.

McConnell, S., *Desarrollo y gestión de proyectos informáticos,* Mc Graw Hill 1997.

Pressman, R.S., *Ingeniería del Software, un enfoque práctico*, Mc Graw Hill, 1998.

Symons, C.R., Software sizing and estimating: Mk II FPA (function point analysis), John Wiley & Sons, 1991

Ragland, B., *Measure, Metric or Indicator: What´s the Difference?*, Crosstalk, 8 (3), 29-30, 1995.

http://www.ieee.org

http://www.iso.org

International Software Benchmarking Standards Group. www.isbsg.org

http://www.ifpug.org

IFPUG. "Manual para la Medición de Puntos Función". Versión 4.2

http://sunset.usc.edu/research/COCOMOII/cocomo_main.html

10 Annexos

10.1 Introducció al model d'estimació COCOMO II.

10.1.1 Descripció general

El mètode d'estimació COCOMO II està basat en dos models: un aplicable al començament dels projectes (Disseny preliminar, en anglès Early Design) i altre aplicable després de l'establiment de l'arquitectura del sistema (Post arquitectura, en anglès Post Architecture).

El model de **Disseny preliminar (Early Design)** contempla la exploració de les arquitectures alternatives del sistema i els conceptes d'operació. En aquesta etapa no se sap lo suficient del projecte com per a fer una estimació fina. Davant d'aquesta situació, el model proposa la utilització de Punts de Funció com a mesura de tamany i un conjunt de 7 factors (cost drivers) que afecten a l'esforç del projecte. Aquests 7 factors son agrupacions dels factors que s'utilitzen a l'altra variant del model (Post Arquitectura).

El model Post arquitectura (Post Architecture) contempla el desenvolupament i el manteniment d'un producte de programari. Aquesta estratègia és mes precisa si s'ha desenvolupat una arquitectura del sistema, validada i establerta com a base per a la evolució del producte. Davant d'aquesta situació, el model proposa la utilització de Línies de codi font i/o Punts de Funció com a mesuradors del tamany, modificadors per a indicar el grau de reutilització i descart del programari, un conjunt de 17 estimadors de cost, i un conjunt de 5 factors que afecten de manera exponencial en l'esforç del projecte.

En ambdós models, la estimació de l'esforç es realitza prenent com a base la següent equació:

$$\mathbf{PM_{nominal} = A \times (Size)^{B}} \quad \text{(eq. 1)}$$

On $\mathbf{PM_{nominal}}$: és l'esforç nominal necessari en mesos home.

Size: és el tamany estimat del programari, en milers de línies de codi (KLOC) o en Punts de Funció sense ajustar (convertibles a KLOC mitjançant un factor de conversió que depèn del llenguatge i la tecnologia).
A: és una constant que s'utilitza per a capturar els efectes multiplicatius en l'esforç requerit d'acord al creixement del tamany del software. El model la calibra inicialment amb un valor de 2,94
B: és una constant denominada Factor escalar, la qual te un impacte exponencial en l'esforç i el seu valor ve donat per la resultant dels aspectes positius sobre els negatius que presenta el projecte.

10.1.2 Valoració del Factor escalar B

El factor escalar B es calcula a partir de la sumatòria de les aportacions de diferents Variables escalars, les quals son variables que indiquen les característiques que el projecte presenta en quant a la seva complexitat i entorn de desenvolupament es refereix. Les Variables escalars de COCOMO II son las següents:

- PREC, variable de precedència u ordre seqüencial del desenvolupament.
- FLEX, variable de flexibilitat del desenvolupament.
- RSEL, indica la fortalesa de l'arquitectura i mètodes d'estimació i reducció de riscos.
- TEAM, aquesta variable reflexa la cohesió i maduresa de l'equip de treball.
- PMAT, relaciona el procés de maduresa del programari.

Cadascuna d'aquestes variables es quantifica amb un valor des de Molt Baix fins a Extra Alt.
La següent taula mostra els criteris i nivells de quantificació per a cadascuna d'aquestes variables:

Factor Escalar (W_i)	**Molt Baix**	**Baix**	**Normal**	**Alt**	**Molt Alt**	**Extra Alt**
PREC	Complerta	Complerta	Una mica	Familiar	Molt Familiar	Absolutament Familiar
FLEX	Rigorós	Ocasional	Una mica De relaxació	Generalment conforme	Una mica De conformitat	Objectius Generals
RESL	Poc (20%)	Una mica (40%)	Freqüent (60%)	Generalment (75%)	Assíduament (90%)	Totalment
TEAM	Interacció molt difícil	Algunes Dificultats D'integració	Integració Cooperativa bàsica	Cooperativa	Altament cooperativa	Interacció Total
PMAT	Molt baix	baix	normal	Alt	Molt alt	Extra alt

Tabla 30:Criteris factor escalar COCOMO II

Els valors que assumeix cadascun d'aquests factors a cada nivell es poden veure a la següent figura:

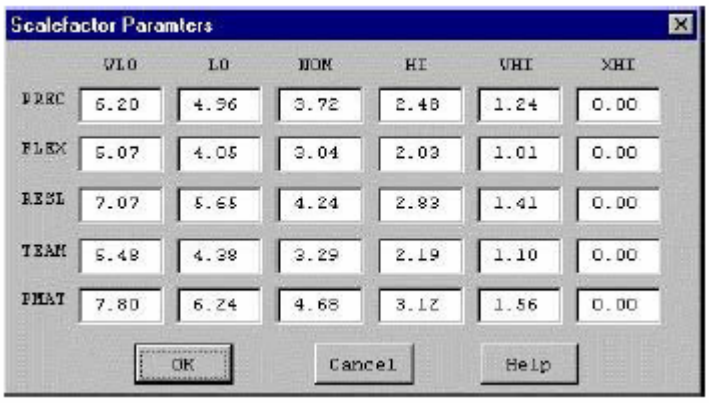


Figura 10-a:Factor escalar B COCOMO II

Després de la ponderació d'aquestes variables, el Factor escalar es calcula mitjançant la següent equació:

$$\mathbf{B = 0.91 + 0.01 \times \Sigma(W_i)} \text{ (Eq. 2)}$$

10.1.3 Ajustament de l'esforç nominal

L'esforç calculat a la equació (1) és un valor nominal i ha de ser ajustat en base a les característiques del projecte. COCOMO II obté les dades necessàries per a l'ajustament de l'esforç nominal considerant un conjunt de Multiplicadors d'esforç (ME), els quals representen les característiques del projecte i expressen el seu impacte en el desenvolupament total del producte de programari.

Els Multiplicadors de l'esforç es quantifiquen amb una escala que va des de Extra Baix fins a Extra Alt, i cada multiplicador te un valor associat a cada nivell de l'escala.

Cadascun dels models d'estimació (Disseny preliminar i Post arquitectura) te un conjunto de Multiplicadors de l'esforç, els quals son acordes amb la informació que es manega a cadascun d'aquests models.

A ambdós models, l'esforç ajustat es calcula mitjançant la següent equació:

$$\mathbf{PM_{ajustat} = PM_{nominal} \times \Pi(ME_i)} \text{ (3)}$$

10.1.4 Multiplicadors de l'esforç en el model Post arquitectura

Per a aquest model, els multiplicadors son 17, agrupats en las següents categories: producte, plataforma, personal y projecte. A continuació es mostren els multiplicadors, amb una breu descripció del seu significat.
Multiplicadors que afecten el producte:

RELY: Fiabilitat requerida del programari. Mesura l'impacte que te una falla en el programari.

	Molt Baix	Baix	Normal	Alt	Molt Alt
RELY	Inconvenients imperceptibles.	Baix, i amb pèrdues fàcilment recuperables.	Moderat, amb pèrdues de fàcil recuperació	Altes pèrdues financeres	Risc per a la vida humana

Tabla 31:Rely

DATA: Tamany de la base de dades. Es mesura com el tamany de la base en bytes sobre el tamany del programa en LOC. S'utilitza per a dimensionar l'esforç requerit per al control i la generació de dades de proba.

	Molt Baix	Baix	Normal	Alt	Molt Alt
DATA		D/P < 10	10 <= D/P < 100	100 <= D/P < 1000	D/P >= 1000

Tabla 32:Data

CPLX: Complexitat del producte. La complexitat es divideix en cinc àrees: Operacions de Control, Operacions de Càlcul, Dependència de Dispositius, Manegament de Dades e Interfícies d'usuari.

	Operacions de control	Operacions de càlcul	Dependència de dispositius	Manegament de dades	Interfícies d'usuari
Molt Baix	Programació lineal, amb molt poques estructures no anidades: DO, IF, CASE. Composició de mòduls simples a través de procediments i funcions	Avaluacions d'expressions simples.	Escriptures i gravacions amb formats simples.	Manegament de vectors simples a memòria. Manegament de consultes i accessos a la base de dades en forma senzilla.	Formularis d'ingrés senzills. Generació de reports simples.
Baix	Estructures ben nidades	Avaluacions moderades de càlculs. Ex. arrel quadrada, exponents	Sense particularitats o dependencies del processador de E/S. Es manegen a través de GET y PUT de conjunts de dades	Manegament de dades sense arxius entremitjos, sense edició. COTS-DB, consultes y actualitzacions moderades	Utilització de GUI (grafic user interface, interfícies gràfiques d'usuari) simples
Normal	La majoria de les estructures son nidades. Amb controls de intercanvi simple de dades entre mòduls a través de paràmetres, taules de decisions, suport per a processament distribuït de baixa complexitat	Utilització de rutines bàsiques i estàndard. Manegament bàsic de vectors i matrius	Operacions de E/S que permeten seleccionar el dispositiu, fent un control de l'estat dels mateixos i processant errors.	Ingrés amb formats múltiples de dades, però conservant un únic format de sortida. Canvis estructurals simples amb edicions. Utilització de consultes i COTS-DB complexos	Utilització simple d'un conjunt de paràmetres de definició de interfase de l'usuari.
Alt	Total utilització de estructures niuades. Manegament de piles i cues. Desenvolupaments per a un únic processador	Anàlisis numèrica senzilla: interpolació, equacions diferencials. Utilització de arrodoniment i truncació.	Operacions de E/S a nivells físics utilitzant direccionament per a la lectura i cerca. Overlap optimitzat per aquestes operacions.	Activació de disparadors a través de dades de la DB. Reestructuracions complexes de dades.	Us de paràmetres de definició d'interfícies. Us simple de característiques multimèdies (entrada a través de la veu)
Molt Alt	Codi recursiu. Manegament d'interrupcions, i sincronització de tasques complexes. Processament distribuït heterogeni. Un únic processador controlat en temps real	Càlculs numèrics difícils però estructurats: equacions matricials, diferencials	Rutines per al diagnòstic d'interrupcions. Manegament de la línea de comunicació entre dispositius. Utilització intensiva del manegament de performance.	Us complex de disparadors. Optimització de consultes. Coordinació de bases de dades distribuïdes.	Us moderat de 2 i 3 dimensions. Habilitades gràfiques complexes i multimèdies.
Extra Alt	Operacions de control múltiples amb canvis dinàmics de prioritats. Control a nivell microcodi. Control en temps real del maquinari distribuït.	Anàlisis numèric complex i no estructurat. Càlculs complexos en paral·lel	Codificació de dispositius asincrònica, amb control crític de performance i processos.	Utilització de llenguatges de manegament de dades, i tècniques d'objectes així com relacionals	Us complex de multimèdia i realitat virtual

Tabla 33:Complexitat

RUSE: Reutilització del codi. Mesura el cost addicional requerit para dissenyar components mes genèrics, millor documentats i mes fiables, per reutilitzar-los en altres projectes.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
RUSE		Res	Per projecte	Per programa	Per línea de Producte	Per múltiples línies de productes

Tabla 34:Ruse

DOCU: Documentació. Avalua els requeriments de documentació al llarg del cicle de vida del projecte.

	Molt Baix	Baix	Normal	Alt	Molt Alt
DOCU	Moltes etapes	Algunes	D'acord a les	Excessiva	Molt excesiva

	del cicle de vida estan sense documentació		necessitats exactes de les etapes del cicle de vida		

Tabla 35: DOCU

Els valors que assumeix cadascun d'aquests multiplicadors a cada nivell es poden veure a la següent figura:

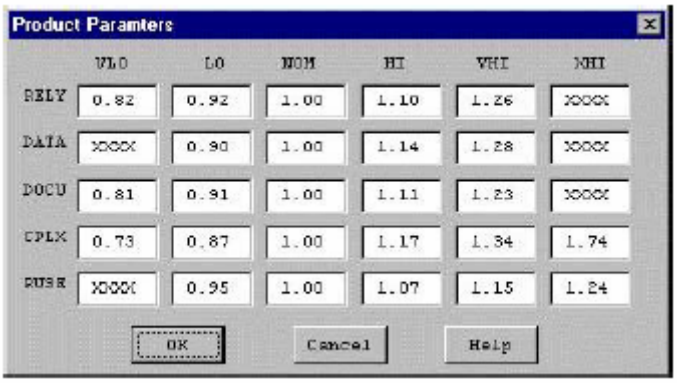


Figura 10-b: Multiplicadors de l'esforç COCOMO II

Multiplicadors que afecten la plataforma:

TIME: Restriccions de temps d'execució. S'expressa en termes de percentatge de disponibilitat de temps d'execució que serà utilitzat pel sistema, versus els recursos disponibles.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
TIME			<= 50% d'us dels recursos disponibles	70%	85%	95%

Tabla 36:TIME

STOR: Restriccions d'emmagatzemament principal. Similar al multiplicador anterior, però relacionades amb l'espai principal d'emmagatzemament.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
STOR			<= 50% d'us de l'espai disponible	70%	85%	95%

Tabla 37:STOR

PVOL: Volatilitat de la plataforma. Expressa la velocitat de canvi del maquinari i el programari usats com a plataforma.

	Molt Baix	Baix	Normal	Alt	Molt Alt
PVOL		Canvis majors cada 12 mesos, i canvis menors cada mes	Majors : 6 mesos, Menors : 2 setmanes	Majors : 2 mesos, Menors : 1 setmana	Majors : 2 setmanes, Menors : 2 dies

Tabla 38:PVOL

Els valors que assumeix cadascun d'aquests multiplicadors en cada nivell es poden veure a la següent figura:

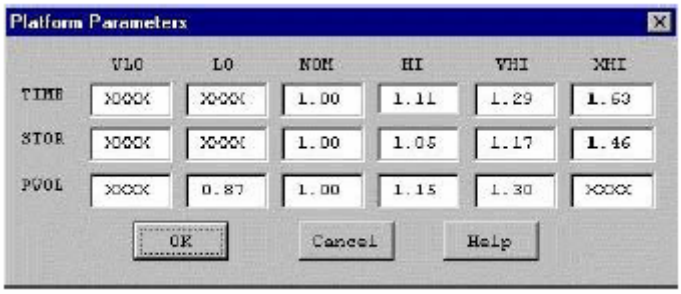


Figura 10-c: Multiplicadors plataforma COCOMO II

Multiplicadors que afecten al personal:

ACAP: Capacitat dels analistes. Es considera la capacitat d'anàlisis i disseny, eficiència, habilitat per a comunicar-se i treballar en equip. No es considera el nivell d'experiència.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
ACAP	15%	35%	55%	75%	90%	100%

Tabla 39:ACAP

PCAP: Capacitat dels programadors. Es considera la capacitat de treball en equip, eficiència i habilitat per a comunicar-se. No es considera el nivell d'experiència.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
PCAP	15%	35%	55%	75%	90%	100%

Tabla 40:PCAP

AEXP: Experiència en aplicacions. Contempla el nivell d'experiència del grup de desenvolupament (principalment analistes) en aplicacions equivalents.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
AEXP	2 mesos	6 mesos	1 any	3 anys	6 anys	>6 anys

Tabla 41:AEXP

PEXP: Experiència en la plataforma. Reflexa la experiència del grup de desenvolupament (principalment programadors) en l'ús d'eines de programari i maquinari utilitzant com a plataforma.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
PEXP	2 mesos	6 mesos	1 any	3 anys	6 anys	>6 anys

Tabla 42:PEXP

LTEX: Experiència en el llenguatge i eines de desenvolupament. Reflexa l'experiència del grup de desenvolupament en el llenguatge de programació i les eines de desenvolupament utilitzades.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
LTEX	2 mesos	6 mesos	1 any	3 anys	6 anys	>6 anys

Tabla 43:LTEX

PCON: Continuïtat del personal. Expressa el percentatge de rotació anual del personal que afecta al projecte.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
PCON	48% a l'any	24% a l'any	12% a	6% a	3% a l'any	<3%

			l'any	l'any		

Tabla 44:PCON

Els valors que assumeix cadascun d'aquests multiplicadors a cada nivell es poden veure a la següent figura:

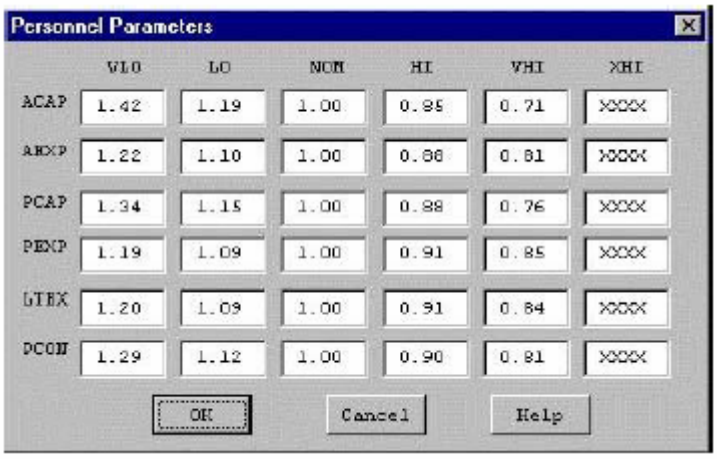


Figura 10-d: Multiplicadors personal COCOMO II

Multiplicadors que afecten al projecte:

TOOL: Ús d'eines de programari. Contempla l'ús d'eines, des de la edició fins a la gestió de tot el cicle de vida.

	Molt Baix	Baix	Normal	Alt	Molt Alt
TOOL	Edició i codificació amb debug	CASE simple i de poca integració	Eines bàsiques per a tot el cicle de vida amb moderada integració	Potents eines a ser usades a tot el cicle de vida amb integració moderada	Eines potents i proactives, molt ben integrades amb el procés, els mètodes i la reutilització

Tabla 45:TOOL

SITE: Desenvolupament en múltiples ubicacions. Involucra la ubicació física i el suport de comunicacions.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
SITE	Telèfon i mail	Fax y telèfons individuals	Xarxa de correu electrònic intern	Comunicacions Electròniques que cobreixen totes les ubicacions	Comunicacions electròniques que cobreixen totes les ubicacions amb la possibilitat de videoconferència	Comunicació total multimèdia

Tabla 46:SITE

SCED: Requeriments de calendari de desenvolupament. Reflexa les restriccions imposades al grup de desenvolupament sobre l'agenda nominal estimada del projecte.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
SCED	75% del nominal	85%	100%	130%	160%	

Tabla 47:SCED

Els valors que assumeix cadascun d'aquests multiplicadors a cada nivell es poden veure a la següent figura:

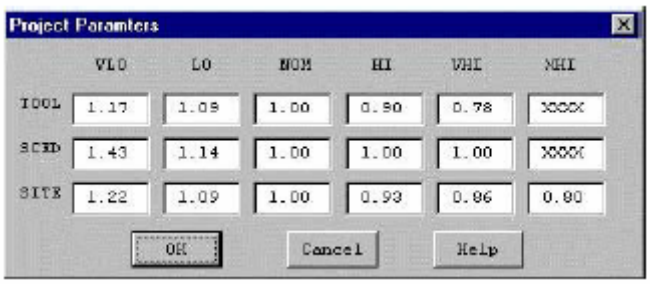


Figura 10-e: Multiplicadors projecte COCOMO II

10.1.5 Multiplicadors de l'esforç al model de Disseny preliminar

Per a aquest modelo, els multiplicadores son 7, i s'obtenen com a combinacions dels multiplicadors del model Post arquitectura. Aquests multiplicadors son:

PERS: Capacitat del personal. Està donat per la suma o la combinació percentual dels multiplicadores ACAP, PCAP y PCON.

	Extra Baix	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
Suma de ACAP, PCAP, PCON	3,4	5,6	7,8	9	10,11	12,13	14,15
Combinació de ACAP i PCAP	20%	39%	45%	55%	65%	75%	85%
Rotació anual del personal	45%	30%	20%	12%	9%	5%	4%

Tabla 48:PERS

RCPX: Complexitat del producte. Ve donat per la combinació dels multiplicadors RELY, DATA, CPLX y DOCU.

	Extra Baix	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
Suma de RELY, DATA, CPLX i DOCU	5,6	7,8	9-11	12	13-15	16-18	19-21
Esforç en la documentació	Molt poc	Poc	Una mica	Bàsic	Fort	Molt fort	Extrem
Complexitat del producte	Molt simple	Simple	Una mica	Moderada	Complexa	Molt complex	Extremadament complex
Tamany de la BD	Petita	Petita	Petita	Moderada	Gran	Molt gran	Molt gran

Tabla 49:RCPX

RUSE: Reutilització. Ve donat pel mateix multiplicador RUSE del model Post arquitectura.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
RUSE		Res	Per projecte	Per programa	Per línea de producte	Per múltiples línies de producte

Tabla 50:RUSE

PDIF: Dificultat de la plataforma. Ve donat per la combinació dels multiplicadors TIME, STOR i PVOL.

	Baix	Normal	Alt	Molt Alt	Extra Alt
Suma de TIME, STOR i PVOL	8	9	10-12	13-15	16,17
Restriccions de TIME i STOR	50%	50%	65%	80%	90%
Volatilitat de la plataforma	Molt estable	Estable	Una mica volàtil	Volàtil	Molt volàtil

Tabla 51:PDIF

PREX: Experiència del personal. Ve donat per la combinació dels multiplicadors AEXP, PEXP i LTEX

	Extra Baix	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
Suma de AEXP, PEXP i LTEX	3,4	5,6	7,8	9	10,11	12,13	14
Experiència a l'aplicació, plataforma, llenguatge i eines utilitzades	<= 3 mesos	5 mesos	9 mesos	1 any	2 anys	4 anys	6 anys

Tabla 52:PREX

SCED: Calendari. Ve donat pel mateix multiplicador SCED del model Post arquitectura.

	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
SCED	75%	85%	100%	130%	160%	

Tabla 53:SCED

FCIL: Facilitats. Ve donat per la combinació dels multiplicadors TOOL i SITE.

	Extra Baix	Molt Baix	Baix	Normal	Alt	Molt Alt	Extra Alt
Suma de TOOL i SITE	2	3	4,5	6	7,8	9,10	11
Suport d'eines de programari	Mínim	Una mica	CASE simples	Eines bàsiques segons el cicle de vida	Bones, moderadament integrades	Molt bones, moderadament integrades	Molt bones totalment integrades
Múltiples llocs de desenvolupament	Suport dèbil per al desenvolupament complex	Una mica de suport per a desenvolupaments complexos	Una mica de suport per a desenvolupaments moderats	Suport bàsic per a desenvolupaments complexos	Fort suport per al desenvolupament de processos moderats	Fort suport per al desenvolupament de processos simples	Molt Fort suport per al desenvolupament de processos complexos

Tabla 54:FCIL

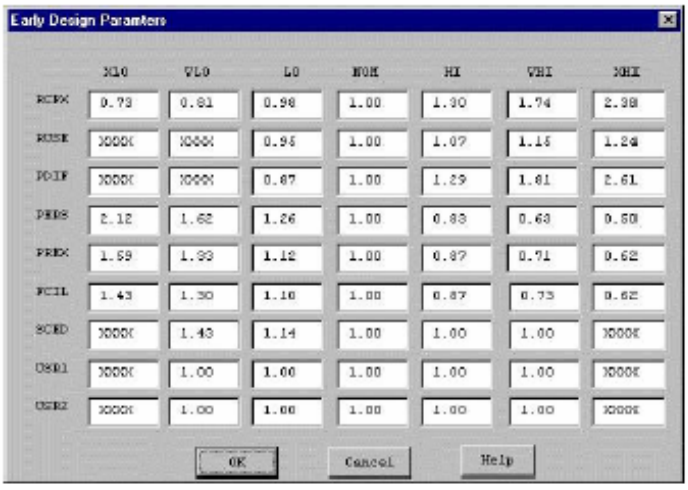


Figura 10-f: Multiplicadors disseny preliminar