

## Índex de continguts

Introducció.....	1
Quan aplicar aquesta tècnica?.....	2
UCP.....	3
UUCP.....	3
Càlcul del UUCW.....	4
Càlcul de l'UAW.....	4
TCF.....	5
ECF.....	7
E1. Familiaritat amb la metodologia.....	7
E2. Experiència amb el tipus d'aplicació a desenvolupar.....	8
E3. Experiència en desenvolupament orientat a objectes.....	8
E4. Experiència de l'analista principal.....	8
E5. Motivació.....	9
E6. Estabilitat dels requeriments.....	9
E7. Estabilitat de l'equip i grau de subcontractació.....	9
E8. Dificultat del llenguatge de programació.....	9
PF.....	10
I quan pot estar?.....	11
Consideracions finals.....	11
Bibliografia.....	12

## Introducció

L'estimació de projectes de programari sovint està a camí entre la ciència i l'art. Dir quin esforç durà fer un programa en les primeres etapes de l'elaboració és una tasca complexa, molt propensa a errors i que s'acaba deixant moltes vegades a l'experiència de l'estimador, que acaba dient una xifra sense saber massa bé en què es sustenta.

Els casos d'ús representen una manera, potser **la manera**, d'obtenir els processos de negoci i els requeriments d'un projecte de programari quan triam fer en nostre desenvolupament amb orientació a objectes. Fer servir aquest anàlisi per determinar l'esforç que requerirà el projecte ens permet partir d'una base sòlida de coneixements.

A més, com veurem, l'estimació per casos d'ús (UCP)<sup>1</sup> és una tècnica senzilla de fer anar, sols requereix de l'ajuda de una petita fulla de càlcul, molt lluny de les complexitats d'altres mètodes d'estimació com COCOMO, Mk II i punts funció.

---

<sup>1</sup> Use Case Points

La tècnica prové de la tesi doctoral de Gustav Karner al 1993 a la universitat de Linköping, treball que actualment és propietat de Rational Software, com a una metodologia que simplifica la metodologia d'estimació per punts funció.

Aquest mètode d'estimació ens diu que l'esforç necessari per a completar un projecte ve donat per la combinació de quatre factors:

1. El nombre de passos necessaris per a completar cada cas d'ús.
2. La complexitat dels actors que intervenen als casos d'ús.
3. Els requisits tècnics del sistema que volem desenvolupar.
4. Factors externs, tals com l'experiència de l'equip de desenvolupament o la subcontractació, per exemple.

A partir d'aquests factors veurem com arribarem a una estimació de l'esforç en termes de persones-hora.

## Quan aplicar aquesta tècnica?

Aquesta tècnica d'estimació dona molts bons resultats quan l'estimador té un coneixement prou bons del domini de l'aplicació que s'ha de desenvolupar, coneix l'equip de desenvolupament i desenvolupa els casos d'ús des de zero.

La tècnica no es pot aplicar tal com està per projectes de gran abast, on els factors inherents a la coordinació del projecte i els aspectes polítics tenen un pes rellevant.

Tampoc és aplicable tal com l'explicarem a cada iteració d'un projecte. La tècnica ens donarà una estimació prou bona, a nivell d'expert estimador, però sols pel conjunt del projecte i no per cada una de les iteracions.

Hi ha adaptacions que permeten fer servir la tècnica per projectes molt grans o per cada fase del desenvolupament iteratiu, però queden fora de l'abast d'aquest article.

Per una alta banda també hem de tenir en compte que el que ens dona l'estimació no considera ni el desenvolupament dels jocs de proves ni la gestió del projecte. Alguns autors consideren que aquest temps augmenta l'estimació entre un 20 i el 25% del total.

El que no ens dona el mètode és la resposta al temps que durà fer un projecte. Ens diu l'esforç, però el temps final dependrà de cada organització i de la seva gestió de projectes. Una organització que pugui dedicar equips a temps complet a cada projecte hauria de poder comprovar millor la qualitat de l'estimació que una amb equips que fan un poc de tot.

En organitzacions on la gestió de projectes estigui molt ben definida i es dugin registres dels projectes, es poden aprofitar aquests registres per millorar les estimacions, ja que com veurem els paràmetres externs al projecte, els que depenen de l'organització poden significar una variació d'entre el 57.5% de reducció o fins a un 40% d'augment en el temps del projecte.

Com hem dit abans l'estimació que fa té si fa no fa és comparable a la que faria un expert, i una vegada es té prou experiència i dades el rang de desviació s'ha documentat en torn al 9% per als 95% de projectes.<sup>2</sup>

## UCP

La fórmula que aplicarem per calcular l'esforç és:

$$\mathbf{UCP = UUCP \cdot TCF \cdot ECF \cdot PF}$$

on

**UUCP** s'anomena *Unadjusted Case Points*, i representa el resultat que obtenim sols a partir de la consideració dels casos d'ús.

**TCF** s'anomena *Technical Complexity Factor*, que avalua de manera subjectiva la complexitat tècnica del projecte a partir d'un conjunt de tretze paràmetres ponderats.

**ECF** s'anomena *Environment Complexity Factor*, i és un conjunt de 8 factors ponderats que intenten copsar la influència de l'equip de desenvolupament i el propi entorn del projecte.

**PF** és el *Productivity Factor* i és el factor que finalment determina la conversió de punts de cas d'ús a hores-home.

Anem a veure com es calcula cada factor per separat:

### **UUCP**

El factor UUCP a la seva vegada està format per la contribució de dos altres termes, així:

$$\mathbf{UUCP = UUCW + UAW}$$

El factor UUCW és l'*Unadjusted Use Case Weight* i és la contribució de nombre de transaccions que componen cada cas d'ús. En aquest context entenem com a transacció un conjunt d'activitats que o bé se completen del tot o no es completen. Es pot determinar el nombre de transaccions comptant el nombre de passos de cada cas d'ús.

Alguns autors, entre ells Karner proposen no tenir en compte els casos d'ús inclosos o extesos, però la majoria recomanen considerar-los.

### **Càlcul del UUCW**

Per a calcular aquest factor es fa una classificació dels casos d'ús entre simples, migs i complexes, assignant-los un pes de 1, 2 i 3 respectivament. Així:

---

<sup>2</sup> Edward R. Carroll d'Agilis Solutions

Tipus de cas d'ús	Nombre de transaccions	Pes
Simple	$\leq 3$	5
Mig	entre 4 i 7	10
Complexe	$> 7$	15

A més de pel nombre de transaccions hi ha altres criteris per classificar el tipus de cas d'ús que ens ajudaran a verificar que la classificació es correcte:

*Simple*: Cas d'ús implementat per una interfície d'usuari simple i que afecta a sols una entitat de la base de dades, típicament una taula. La seva implementació implica a menys de 5 classes.

*Mig*: Interfície d'usuari més complexa, afecta a dues o més entitats de la base de dades. La seva implementació afecte a entre 5 i 10 classes.

*Complexe*: Interfície d'usuari o procés molt complexe. Afecta a més de tres entitats de la base de dades.

Notem que la classificació que es fa dels casos d'ús està limitada a 3 grups. En projectes molt grans hi pot haver casos d'ús molt complexes i no hi ha manera de donar-los un pes addicional. D'aquí que es tenguin que fer modificacions a la tècnica o a la manera d'escriure casos d'ús per aquests tipus de projectes.

Es considera que un bon cas d'ús tindrà entre 3 i 10 transaccions. Si en té més s'ha de mirar de reavaluar el cas d'ús per fer-ho més manejable i mirar si no estam pecant d'excés de detall. En general a l'hora d'escriure casos d'ús es fan dues recomanacions bàsiques:

- El cas d'ús ha d'estar escrit des del punt de vista de l'usuari de l'aplicació.
- Cada pas del cas d'ús descriu una acció que se duu a terme completament o que no se duu a terme.

Per exemple es considera un error descriure en els casos d'ús les accions que faci l'usuari en la interacció amb la interfície de l'aplicació.

### ***Càlcul de l'UAW***

Aquest terme avalua la contribució de la complexitat dels actors en cada cas d'ús. Els actors es classifiquen com : simple, mig i complex.

Un actor simple representa un altre sistema amb el qual interacciona amb l'aplicació a través d'una API ben definida de comunicacions, com per exemple una aplicació que se connecti a la nostra mitjançant SOAP, XML-RPC, etc.. Un actor mig és aquell representat per una sistema que interacciona amb l'aplicació mitjançant un protocol com el TCP/IP. Per exemple un terminal de

recollida de dades que es connecta a la nostra aplicació per fer deixar-hi les dades emmagatzemades. Finalment un actor complex es reserva pels actors humans, que interaccionen amb l'aplicació mitjançant una interfície gràfica, ja sigui RIA o web.

A cada tipus d'actor se li assigna un pes i l'UAW es calcula fent la suma del nombre d'actors de cada tipus pel pes que els hi correspon.

Tipus d'actor	Pes
Simple	1
Mig	2
Complex	3

## TCF

El TCF intenten computar la contribució de fins a tretze factors tècnics que tenen impacte en la productivitat. Cada factor és ponderat d'acord amb el seu impacte damunt l'esforç que s'ha de realitzar. La fórmula pel calcul del TCF és

$$TCF = C_1 + C_2 \sum_{j=1}^{13} W_j \cdot F_j$$

On  $C_1$  i  $C_2$  són constants de valors 0.6 i 0.01 respectivament.

Per a cada projecte cada un dels factors tècnics que presentarem s'avaluen i se'ls assigna una complexitat subjectiva amb un valor entre zero i cinc. Una complexitat percebuda de zero significa que aquest factor no és rellevant pel projecte, de 3 que té una rellevància estàndard i cinc que és un factor crític. En cas de dubte es recomana assignar un valor 3.

$$TCF = 0.6 + 0.01 \sum_{j=1}^{13} W_j \cdot T_j$$

Factor	Descripció	Pes (W)
T1	Sistema distribuït	2
T2	Rendiment	1
T3	Importància de l'eficiència percebuda per l'usuari	1
T4	Complexitat dels processos interns	1
T5	Orientació cap a la reusabilitat	1
T6	Fàcil d'instal·lar	0.5
T7	Fàcil de fer anar	0.5

T8	Portabilitat	2
T9	Facilitat per canviar-ho i mantenir	1
T10	Concurrència	1
T11	Consideracions especials de seguretat	1
T12	Accessible per tercers	1
T13	Es necessita formació específica als usuaris	1

Fixem-nos en que la contribució més gran a la complexitat del sistema la fan el que el sistema sigui distribuït i la portabilitat. En aquest segon cas, la portabilitat l'entendem com un requisit del sistema, i no que si feim el programa amb Java o Python per exemple tinguem un sistema portable. Portabilitat en aquest cas l'hem d'entendre com que hem de posar especial cura o no en que el nostre programa pugui funcionar en diferents plataformes.

La orientació cap a la reusabilitat fa referència a que si pensam que parts dels programa les farem servir per altres aplicacions ja des del principi i ho programam en conseqüència, això pot afegir un grau més de complexitat en en desenvolupament.

T2 i T3 pareixen conceptes complementaris, i com altres factors subjectes a interpretació per a cada projecte. Però per exemple podríem avaluar a T2 el temps de resposta de l'aplicació davant un procés, i a T3 el temps que es considera acceptable per l'usuari de l'aplicació. És dir, separam la part de procés de càlcul de la part d'utilització de l'aplicació. Per exemple, un requisit de l'aplicació és que ha d'acabar els informes mensuals en 3 dies, això afectaria a T2 però no a T3 ja que no es tractaria d'un procés en temps real. En canvi, un sistema de monitorització ha d'actualitzar les dades i presentar-les en pantalla a l'usuari gairebé en temps real, potser no hi haurà temps de càlcul que afecti a T3 però el tractament i pintat de les dades seria un factor que afectaria a T2.

T12 Quan una aplicació no s'ha d'utilitzar sols "a casa", sinó que s'ha de distribuir a terceres persones, això afegeix un factor extra de complexitat.

$$TCF = 0.6 + 0.01 \sum_{j=1}^{13} W_j \cdot T_j$$

El factor TFC afecta a l'esforç final disminuint-ho fins al 40% en el cas límit que cap factor hi tengui influència o augmentant fins a un 30% la complexitat quan tots els factors són crítics.

## **ECF**

El factor ECF representa la contribució de l'equip de desenvolupament a l'esforç de construcció de

l'aplicació. Un equip motivat i amb experiència té un impacte positiu damunt l'esforç total, en canvi un equip amb poca experiència té un impacte negatiu, augmentant l'esforç total. La fórmula pel ECF és:

$$ECF = 1.4 - 0.03 \sum_{j=1}^8 W_i \cdot E_i$$

Es consideren 8 factors, que podem veure en la següent taula amb els seus pesos corresponents.

Factor	Descripció	Pes(W)
E1	Familiaritat amb la metodologia	1.5
E2	Experiència amb el tipus d'aplicació a desenvolupar	0.5
E3	Experiència en desenvolupament orientat a objectes	1
E4	Experiència de l'analista principal	0.5
E5	Motivació	1
E6	Estabilitat dels requeriments	2
E7	Estabilitat de l'equip/Subcontractació	-1
E8	Experiència/dificultat del llenguatge de programació	-1

Anem a veure com es determinen els valors a assignar a cada paràmetre:

### **E1. Familiaritat amb la metodologia**

- Assignam 0 : Si l'equip de desenvolupament no està familiaritzat amb la metodologia utilitzada en el projecte.
- Assignam 1 : Si l'equip sols té un coneixement teòric de la metodologia
- Assignam 2-3 : Un o més membres de l'equip han utilitzat la metodologia però en tenen poca experiència.
- Assignam 2-4 : Al manco la meitat dels membres de l'equip han utilitzat la metodologia en diferents projectes.
- Assignam 5 : Contam amb un equip experimentat en la metodologia utilitzada.

*En la literatura aquest factor algunes vegades es posa com a coneixement de la metodologia RUP o de l'UML. Crec que és aplicable a qualsevol metodologia de feina, sempre que n'hi hagi una.*

### **E2. Experiència amb el tipus d'aplicació a desenvolupar**

- Assignam 0 : si no hi ha cap membre de l'equip que hagi desenvolupat aplicacions

semblants.

- Assignam 1-2: Si pocs dels integrants del projecte tenen experiència en el desenvolupament d'aplicacions semblants i aquesta experiència és de fins a 18 mesos.
- Assignam 3 : Si tots els membres de l'equip tenen experiència de més de 18 mesos en projectes semblants.
- Assignam 4: Si la major part de l'equip té més de 2 anys d'experiència.
- Assignam 5: Si tot l'equip té més de 2 anys d'experiència desenvolupant projectes semblants.

*Conèixer el negoci i el tipus d'aplicació fa que el desenvolupador i l'analista puguin ser molt més proactius i detectar errades de concepte molt més ràpid.*

### **E3. Experiència en desenvolupament orientat a objectes.**

- Assignam 0: Si cap membre de l'equip té experiència en desenvolupa orientat a objectes.
- Assignam 1: Si tots els membres de l'equip tenen menys d'un any d'experiència.
- Assignam 2-3: Si tots els membres de l'equip tenen més d'un any però menys de 18 mesos d'experiència
- Assignam 4: Si la major part de l'equip té més de dos anys d'experiència
- Assignam 5: Si tots els desenvolupadors tenen més de dos anys d'experiència.

### **E4. Experiència de l'analista principal**

- Assignam 0: Si és un analista novell.
- Assignam 1-2 : Si té experiència d'uns pocs projectes.
- Assignam 3-4 : Si al manco té una experiència de 2 anys en varis projectes.
- Assignam 5 : Si al manco té una experiència de 3 anys en una gran varietat de projectes.

*Aquí jo afegiria que l'experiència ha de comptar sobretot per projectes duits a terme amb èxit, en cas contrari aquest factor sols tendria una dependència directa amb el temps que duu l'analista/cap de projecte en el mateix lloc de feina.*

### **E5. Motivació**

- Assignam 0: si l'equip no està motivat.
- Assignam 1-2: si l'equip està poc motivat.
- Assignam 3-4: si l'equip està motivat cap a fer una bona feina.

- Assignam 5: per un equip motiva i inspirat.

*La motivació i el següent punt són directament proporcionals. El fer feina a un bon ambient, amb tecnologies punteres, per un projecte interessant i engrescador com podem veure tenen una forta influència en el cost total del projecte. El factor humà és un efecte de primer ordre damunt el càlcul de l'esforç total.*

### **E6. Estabilitat dels requeriments**

- Assignam 0: per requeriments gens estables i canvis constants.
- Assignam 1-2: per requeriments inestables. Es demanen canvis a llarg del temps.
- Assignam 3-4: sols hi ha canvis menors.
- Assignam 5: requeriments molt estables. pràcticament no hi ha canvis.

*Es bastant probable que si estam entre 0 i 2 la motivació també estigui per aquest interval.*

### **E7. Estabilitat de l'equip i grau de subcontractació**

- Assignam 0: No hi ha subcontractació. Tot els membres de l'equip ja han fet feina plegats.
- Assignam 1-2: Hi ha fins un 20% de gent nova o subcontractada.
- Assignam 3-4: Hi ha la mitat de l'equip que és nova o subcontractada.
- Assignam 5: Tots els membres de l'equip són nous o subcontractats.

### **E8. Dificultat del llenguatge de programació**

- Assignam 0: Tots els membres de l'equip són experts en el llenguatge de programació triat.
- Assignam 1: La major part de l'equip té al manco 2 anys d'experiència en el llenguatge de programació triat.
- Assignam 2: Si tots els membres tenen més d'un any i mig d'experiència en el llenguatge de programació triat.
- Assignam 3: Si la major part de l'equip té menys d'un any d'experiència en el llenguatge de programació triat.
- Assignam 4: Si sols un percentatge petit de l'equip té experiència.
- Assignam 5: Si no hi ha cap membre de l'equip amb experiència en el llenguatge de programació triat.

Aquest factor, a més ha de reflectir la complexitat del llenguatge de programació del desenvolupament. No és el mateix desenvolupar una aplicació en C++, en Java o en Python per posar-ne un exemple. Tot i això, el llenguatge triat seria un efecte de segon ordre si consideram que

no hi ha experiència en el llenguatge.

*Particularment he trobat efectiu considerar una disminució d'un punt a l'escala quan el llenguatge de programació triat és Python per exemple en lloc de Java, si estam en una escala d'experiència entre 1 i 2. Si estam entre 4 i 5 triar un llenguatge d'script (Python, PHP, Ruby, ..) front un llenguatge clàssic (C++, Java, C#) representaria un bot de 2 cap avall en l'escala, per donar idea de la diferent corba d'aprenentatge que tenen els primers davant els segons. Sols en tenc evidència empírica, no avalada per cap estudi seriós.*

*L'efecte del diferent nombre de línies de codi per un o altre llenguatge es tendria en consideració quan formam definitivament l'equip i passam de punts de cas d'ús a d'hores-home i finalment a línies de codi.*

L'impacte del factor ECF en l'esforç total del projecte és d'un 57,5% de reducció en el cas més favorable i d'un augment del 40% en el cas més desfavorables.

## **PF**

El factor PF representa el factor de conversió entre els punts de cas d'us i les hores-home. En un món ideal, on hi ha equips de projecte estables i se duu acuradament la dedicació a cada projecte aquest *ratio* és un punt a anar ajustant.

El factor dependrà tant de l'equip del projecte com del llenguatge utilitzat en el desenvolupament. Idealment per a cada empresa i per cada equip de projecte s'hauria de dur un històric de l'esforç que s'ha fet per a cada projecte, l'esforç previst, el llenguatge de programació i el tipus de projecte.

Com que això sovint no és així, els experts recomanen un factor d'entre 15 i 30 per començar, segons la fiabilitat del nostre equip en el compliment de plaços. A més fiabilitat menor ha de ser el nombre.

Si l'equip s'ha format per aquest projecte o si és la primera vegada que feim aquests càlculs podem agafar un valor de 20 com a referència. Alguns autors consideren que hi ha dependències entre l'ECF i aquest valor, de manera que proposen un factor 20 si l'ECF menor que 3, 28 si està entre 3 i quatre i 36 si és major que quatre. Sols que en aquest darrer cas es recomana revisar tot el projecte<sup>3</sup>.

## **I quan pot estar?**

Fins aquí hem estimat l'esforç en termes d'hores-home que s'ha de dedicar a un projecte. Una vegada ajustat el factor de conversió convenientment per la nostra empresa i per el nostre equip, tendrem una eina de valoració prou efectiva per poder donar l'import estimat del projecte (amb una variació

---

3 Schneider and Winters

del 9% segons les millors estimacions).

Això, però, no respon a una altra pregunta que sovint hem de contestar al mateix temps: quan pot estar?

Per contestar a aquesta pregunta tenim una petita fórmula que ens diu

$$entrega = 3.0 \cdot mes - home^{1/3}$$

Suposem que la nostra estimació ens ha donat un esforç de 2188 hores-home<sup>4</sup>, considerarem que els dies efectius de treball per any són 210, estimació que pot dependre de cada empresa, i que la jornada és de 8 hores.

Amb aquestes dades i aplicant la fórmula tenim que podríem tenir tot el projecte acabat en 7,5 mesos, la qual cosa implica tenir un equip de dues persones dedicades al desenvolupament.

Aquesta estimació inicial representa el cost mínim en personal per al client. Val a dir que és una estimació molt grollera, ja que el coeficient, 3 en aquest cas, s'ha d'ajustar també per a cada empresa. El que sí ens està dient és que hi ha un temps i un equip òptim per a desenvolupar un projecte a un cost mínim.

Si el client vol un temps menor, llavors ja entrem en el que s'anomena la compressió del plaç d'entrega.

$$\text{factor de compressió} = \frac{\text{plaç desitjat}}{\text{plaç inicial}}$$

$$\text{esforç plaç comprimit} = \frac{\text{esforç inicial}}{\text{factor de compressió}}$$

Es considera que no són assolibles factors de compressió menors de 0,75. Per tant si agafant aquest límit com a referència i pels valors de l'exemple tenim que d'un entregable a 7,5 mesos fet per dues persones passaríem a un entregable en 5,6 mesos fet per un equip de 4 persones, la qual cosa significa passar d'un esforç ajustat de 15,63 hores-home a un esforç de 20,84 hores-home. És a dir, una reducció en el plaç d'entrega del 25% requereix un augment de l'esforç del 33%.

## Consideracions finals

En aquest article hem presentat una metodologia d'estimació de projectes útil però que necessita ser realimentada amb les nostres pròpies dades i experiències.

L'àmbit d'aplicabilitat de la metodologia també està molt ben definit: és aplicable a estimacions de cost quan tenim com a base l'anàlisi per casos d'ús.

Per projectes molt grans l'estimació no és molt acurada, ja que el marge d'error que tenim és molt alt

---

4 Veure l'exemple que acompanya aquest article

per mor de la influència que podem tenir quan hi ha distints analistes, desconeixement de l'equip, canvis en les condicions de treball, etc. Aspectes aquests que normalment no es donen en projectes més reduïts en nombre de gent i temps.

Tot i això hi ha bibliografia de comp aquesta metodologia de càlcul pot ser extesa per a tractar també amb projectes on hi ha implicades més de 200 persones i es vol saber l'esforç que s'ha de dedicar a cada iteració.

En la meua experiència aquest mètode de càlcul dona resultats molt acurats quan coneixem bé el nostre equip i volem a més desenvolupar programari de qualitat. Les xifres que dona sovint pareixen elevades, però hem de tenir en compte que al càlcul se suposa que feim les coses "com toca", és a dir, documentant el que se fa, amb uns estàndards de qualitat, revisions de codi, documentació per l'usuari etc.

Si no hem de mantenir el codi o no s'ha de documentar res, les xifres poden reduir-se molt, de fet la metodologia contempla també això com a factors. El que sí que no es té en compte és una disminució amb la qualitat del programari entregable. És a dir, es pot entregar més aviat fent hores extres a dojo dins d'uns límits, es pot entregar més aviat sacrificant la qualitat, però a la llarga tot això repercuteix en els projectes, en els futurs en el cas de l'excés d'hores i en la satisfacció del client quan sacrificam qualitat per temps.

## Bibliografia

Referència principal:

- Estimating Object Oriented Software Projects with Use Cases. Kirsten Ribu. Master of Science Thesis.

Altres referències:

- Rapid Development. Steve McConnell. Microsoft Press.
- Effort Estimation of Use Cases for Incremental Large-Scale Software Development. Parastoo Mohagheghi & altres.
- QSM Function Point Programming Languages Tables. <http://www.qsm.com/FPGearing.html>
- Software Project Estimation. Kathleen Peters
- Use Case Points. An Estimation Approach. Gautam Banerjee.
- Project Estimation with Use Case Points. Roy K. Clemmons.
- Project Estimation with Use Case Points. RoyClem.  
<http://www.codeproject.com/gen/design/usecasep.asp>