Funkciópont elemzés a gyakorlatban

MTA Információtechnológiai Alapítvány

2003

Tartalomjegyzék

Ábrák

Szabályok

Táblázatok

1 Funkciópont elemzés[1]

Ebben a szakaszban röviden áttekintjük a funkciópont elemzést, mint az egyik projektirányítást segítő technikát, és ismertetjük a legfontosabb alapfogalmakat.

1.1 Miért használjuk a funkciópont elemzést

Mi a célja a funkciópont elemzésnek: az informatikai, számítógép alapú rendszerek nagyságának, méretének megbecslése, mert ez lehetővé teszi, hogy az informatikai részlegek illetve a projektek irányításáért felelős személyek méretezni tudják a feladatot úgy, mint a hagyományosabb, korábban kifejlődött mérnöki tudományoknál.

A hagyományos gyártási folyamatokban a következő mérőszámokat használják:

Egy termékre jutó költség	Összes költség / előállított termékek száma	Termelékenység
Kibocsátott mennyiség hetente	Előállított termékek száma / idő ráfordítás	Kibocsátás
Hiba százalék	Hibás termékek száma / előállított termékek száma	Minőségi jellemző

1-1. táblázat: Mérőszámok

A funkciópont elemzés úgy tekinti az információrendszert mint két nagy alkotórészből álló rendszer, nevezetesen az információ feldolgozó rész és a műszaki megvalósítás. Az információ feldolgozó rész foglalkozik a rendszer bemenő és kimenő adataival, illetve ezek feldolgozásával és átalakításával. A műszaki megvalósítás az információfeldolgozási tevékenységhez szükséges műszaki korlátokkal és peremfeltételekkel foglalkozik.

Például egy rendszer havi jelentést készít a kinnlevőségekről: az információ feldolgozó rész folalkozik a havi jelentés összeállításával, ennek műszaki megvalósítása lehet kötegelt feldolgozás vagy interaktív (on-line), vagy akár nagyon gyors válaszidőre felkészített rendszer.

1.2 Funkciópont metrikák[2]

[1] Function Point Analysis

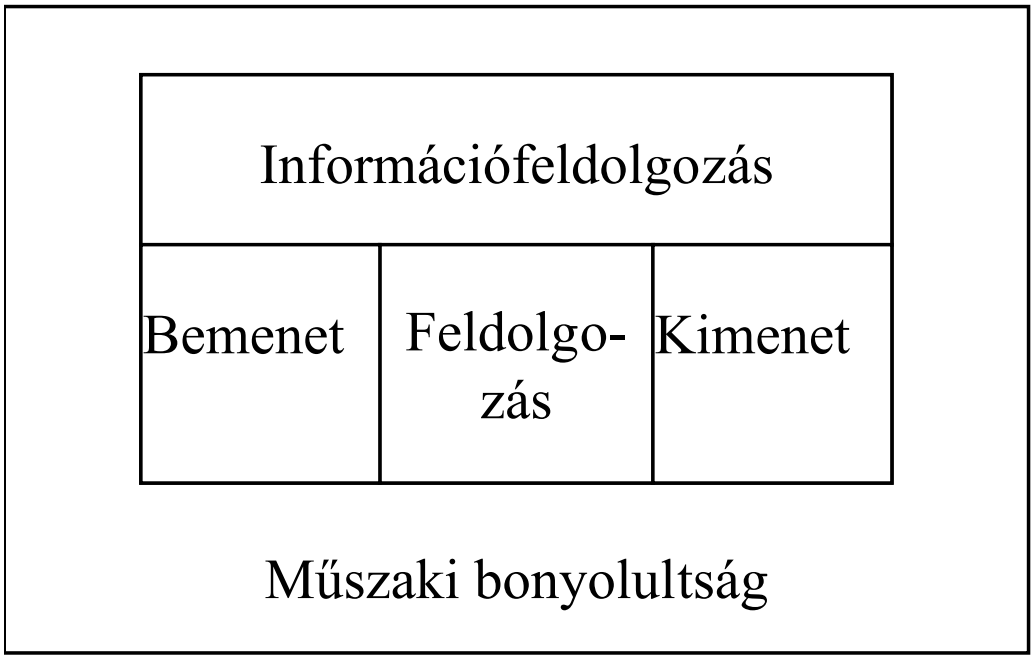
Két nagy iskola van, a gyakorlatban bármelyik megközelítés jól használható, a két módszer által adott értékek jó közelítéssel átszámolhatók egymásba.

'IFPUG' Funkciópont (International Function Point Users Group, Nemzetközi Funkciópont Felhasználói Csoport)

'MkII' Funkciópont

1.3 A rendszer méretének kiszámítása

Egy információrendszer méretét funkciópont index formájában a következő módon lehet megállapítani. A rendszer összes logikai tranzakciójára össze kell számolni a bemenő, a kimenő adatokat és a feldolgozás során érintett entitásokat (adatcsoportokat). Ezután még két lépés van. Az első teljesen matematikai, amelyben a korrigálatlan funkciópontot számítják, a másodikban pedig a műszaki bonyolultságnak megfelelően korrigálják az eredményt, figyelembe véve a technológiai tényezőt.



1. ábra. Egy információrendszer szerkezete[3]

---

[2] lsd., MkII FPA Counting Practices Manual Version 1.3.1, Author: UK Software Metrics Association – Metrics Practices Committee; http://194.143.167.33/library/Resources/MkIIr131.pdf (2000. január 12.)

Sizing and estimating Software in Practice, Making MK II Function Points Work, by Stephen Treble, Neil Douglas, McGraw-Hill, 1995

[3] lsd. még IBM klasszikus HIPO módszerét (Hierarchical Input Porcess Output).

A korrigálatlan funkciópont kiszámítása egyszerű ha minden logikai tranzakcióra vonatkozó alapadat rendelkezésre áll. A bemenetek, kimenetek és az érintett entitások számát egy alkalmas súllyal meg kell szorozni, majd ezeket össze kell adni.

Bemenet súlya	0,58
Információ feldolgozás súlya	1,66
Kimenet súlya	0,26

1-2. táblázat: Súlytényezők[4]

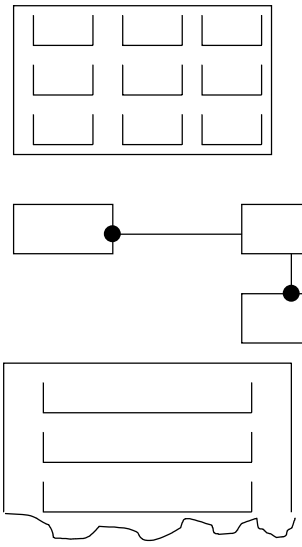
A korrigálatlan funkciópont kiszámításának a képlete:

$UFP= N_i \times W_i + N_e \times W_e + N_o \times W_o$

N_i	Bemeneti típusú mezők száma
N_e	Az entitások száma
N_o	Kimeneti típusú mezők száma
W_i	Bemenet súlya
W_e	Információ feldolgozás súlya (entitások)
W_o	Kimenet súlya

1-3. táblázat: Funkciópont számítási tényezők

A korrigálatlan funkciópontot megszorozzák egy műszaki bonyolultsági tényezővel (TCA, Technical Complexity Adjustment). 19 ilyen tényező van, amelyet egy ötfokú skálán értékelnek, majd ebből alakítanak ki egy összevont, aggregált értéket.



- Bemeneti adatelemek × súly

+

- Érintett entitások × súly

+

- Kimeneti adatelemek × súly

2. ábra. Egy információrendszer alkotórészeinek súlyozása

[4] MKII funkciópont elemzés súlytényezői. http://194.143.167.33/library/Resources/MkIIr131.pdf (2001.01.23).

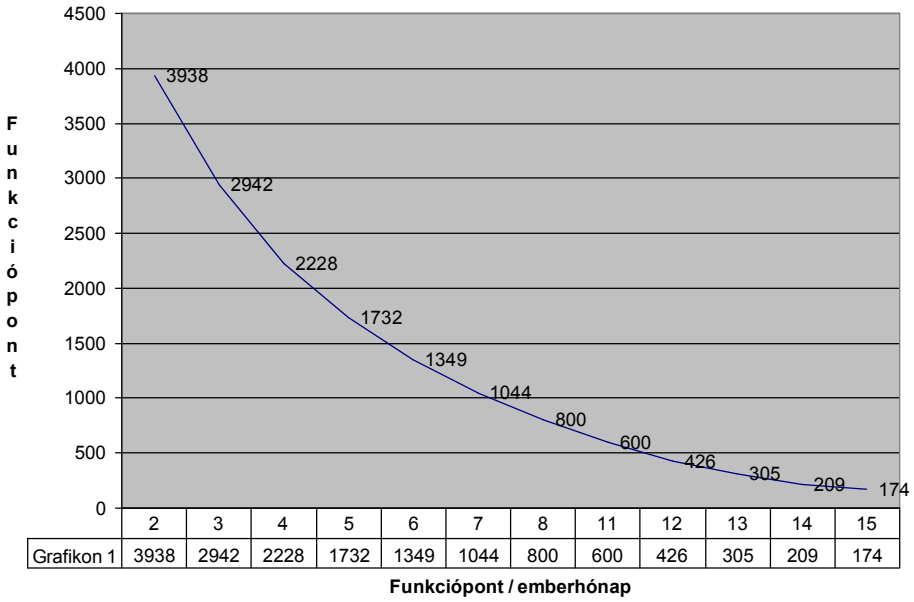
Végül a funkciópont indexet úgy kapjuk, hogy a rendszer információ feldolgozó részének méretére vonatkozó korrigálatlan funkciópont értéket megszorozzuk a technológiai, műszaki bonyolultsági faktorral:

FPI= UFP × TCA

1.4 A funkciópont használata a gyakorlatban

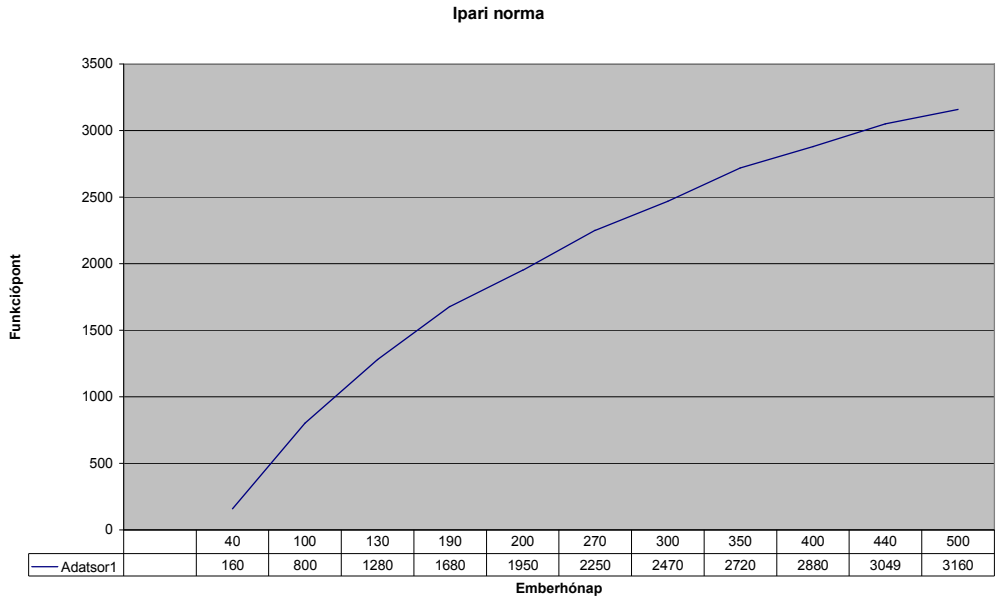
A funkciópont lehetővé teszi az iparágon belüli termelékenységi adatok összevetését, a fajlagos funkciópont költségek összehasonlítását.

1.4.1 Ipari termelékenységi mutató

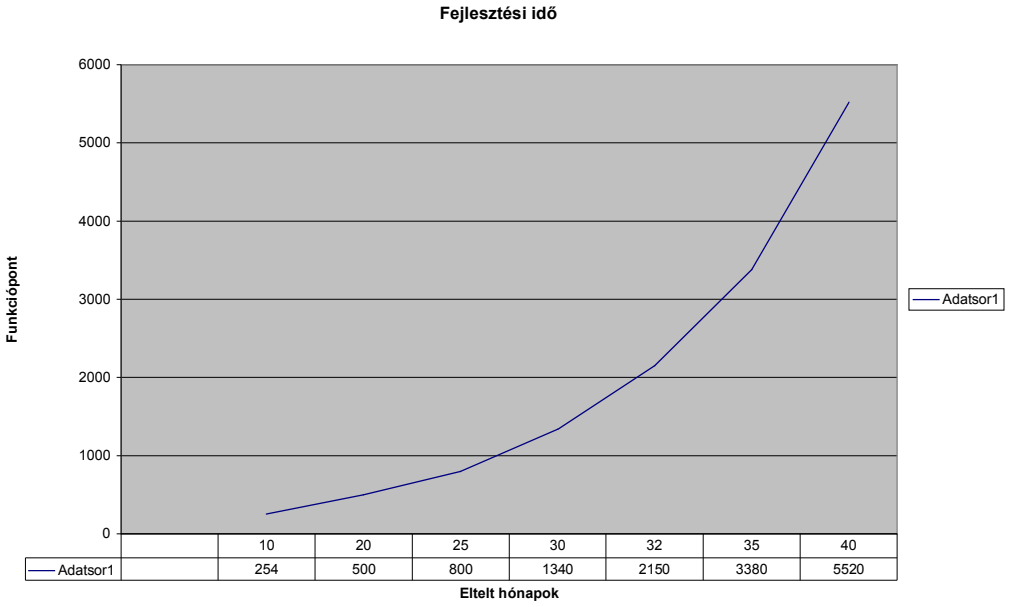


	2	3	4	5	6	7	8	11	12	13	14	15
Grafikon 1	3938	2942	2228	1732	1349	1044	800	600	426	305	209	174

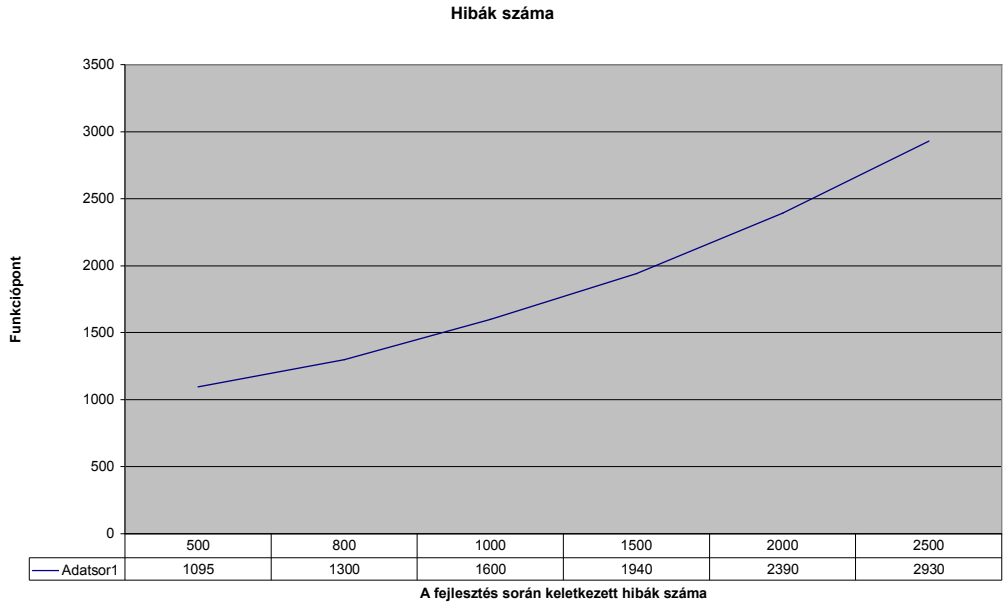
3. ábra. Ipari norma szerinti termelékenységi adatok /funkciópont /emberhónap)



4. ábra. Ipari norma szerinti termelékenységi adatok (IFPUG)



5. ábra. Ipari norma szerinti termelékenységi adatok (fejlesztési ráfordítási idő) [IFPUG]



6. ábra. Ipari norma szerinti termelékenységi adatok(hibaszám) [IFPUG]

1.4.2 Funkciópont méretezés ökölszabályai

A különböző átadandó termékek méretének meghatározása a szoftver / rendszerfejlesztési költségbecslés alapja. A funkciópont metrika használata a rendszer méretezésének bonyolult feladatát egy viszonylag pontos és könnyen végrehajtható eljárássá alakította, bár a rendszerfejlesztés olyan korai fázisában amikor azok az információk még nem állnak rendelkezésre amelyek a funkciópont teljes elemzésének elvégzéséhez szükségesek — a méretezés meglehetősen pontatlan.

1.4.2.1 A teljes követelményelemzést megelőzően végrehajtható funkciópont méretezés

Az egyik lehetőség: több oldalról közelíti a kérdést több különböző szempontrendszer szerint.

Az ilyen szempontlista általában úgy van kialakítva mint a földrengések erejét jelző Richter-skála. A nagyobb számértékek jelentősége sokkal nagyobb, mint a kisebbeké. Azonban ez a durva megközelítés túlságosan pontatlan ahhoz, hogy egy komoly rendszer vagy szoftverfejlesztés megalapozott projekt becslése legyen, bár megvan az a nagyszerű előnye, hogy már akkor is használható amikor semmilyen más ismert mértezési megoldás nem alkalmaható.

A terjedelem, az osztály és a típus értelmében jellemzett szoftver funkciópont méretének kiszámításához csak annyit kell tenni, hogy a terjedelemre, osztályra és típusra vonatkozó lista értékeket összegezzük, majd ezt a számot

2,35-ik kitevőre emeljük. Ez a szám az IFPUG[5] 4. Verziónak megfelelő funkciópont érték első közelítése lehet.

A módszer alkalmazásához szükséges információk már a rendszerfejlesztés első napján rendelkezésre állnak, tehát lehetőség van egy nagyon korai, bár meglehetősen pontatlan rendszerméret becslés elvégzésére. A módszer alkalmazásához három dolgot kell tenni:

A méretezendő projektre vonatkozó lista értékeket gyűjtsük ki a terjedelemre, az osztályra és a típusra.

Adjuk össze a lista értékeket.

Emeljük fel az összeget a 2,35 hatványkitevőre.

Terjedelem	Osztály	Típus
1. Szubrutin	1. Egyedi szoftver	1. Nem-procedurális (pl. SQL, stb.)
2. Modul	2. Shareware	2. Web aplet
3. Újrafelhasználható modul	3. Egyetemi, kutatási fejlesztés	3. Kötegelt (Batch)
4. Eldobható prototípus	4. Egy helyszínen telepítendő — belső alkalmazásra	4. Interaktív
5. Evolúciós prototípus	5. Több helyszínre telepítendő — belső alkalmazásra	5. Interaktív grafikus (GUI[6])
6. Önmagában megálló, működő program (standalone)	6. Szerződés projekt — polgári alkalmazás	6. Kötegelt adatbázis feldolgozás (Batch database)
7. Rendszer alkotórész	7. Time sharing (időosztásos) rendszer	7. Interaktív adatbázis alkalmazás
8. Rendszer kibocsátási verzió	8. Katonai szolgálatok számára alkalmazások	8. Ügyfél-kiszolgáló, kliens/szerver alkalmazás
9. Új rendszer	9. Internet	9. Matematikai
10. Összetett rendszer	10. Bérelhető, lízingelhető szoftver	10. Rendszer szoftverek
	11. Szoftvercsomag	11. Kommunikáció
	12. Kereskedelmi forgalomba kibocsátott szoftver	12. Folyamatirányítás

[5] International Function Point Users' Group

[6] Graphical User Interface

Terjedelem	Osztály	Típus
	13. Szolgáltatás kihelyezésben készített szoftver (Outsourcing)	13. Megbízható rendszer (Trusted system)
	14. Kormányzati szerződésben készített szoftver	14. Beágyazott rendszerek (embedded)
	15. Honvédelmi (katonai) szerződésben készített szoftver	15. Képfeldolgozás (Image processing)
		16. Multimédia
		17. Robotika
		18. Mesterséges intelligencia, ismeretalapú rendszerek
		19. Neurális hálók
		20. Hibrid rendszerek: fentebbiek keveréke.

1-4. táblázat: A terjedelem, osztály és típus jellemzésre példák

1. szabály: A terjedelem, osztály és típus alapján a projekt / rendszer durva méretezése

Néhány számítási példa:

Terjedelem	6	(standalone)
Osztály	4	(belső alkalmazás, egy helyen)
Típus	8	(kliens/szerver)
Összeg	18	

Elvégezve a hatványozást $18^{2.35}$ = 891,026-t kapunk, ami egy meglehetősen jó közelítés, mivel a kliens / szerver alkalmazások általában 1000 funkciópont körüliek, vagyis ez az érték jónak mondható.

Nézzük a számítást egy kis, személyes célokra szánt alkalmazásnál:

Terjedelem	4	(eldobható prototípus)
Osztály	1	(egyedi program)
Típus	1	(nem-procedurális)
Összeg	6	

Elvégezve a hatványozást $6^{2.35}$ = 67-t kapunk az alkalmazás funkciópont méretének közelítésére. Ez egy jó érték, mert az ilyen jellegű, személyes használatra szánt alkalmazások funkciópont mérete álatalában 100 körüli.

Terjedelem	9	(új rendszer)
Osztály	14	(kormányzati alkalmazás)
Típus	13	(megbízható rendszer)
Összeg	36	

Ismételten elvégezve a hatványozást $36^{2.35}$ = 4543-t kapunk, ami jól érzékelteti, hogy egy komoly államigazgatási projekt meglehetősen bonyolult. A hasonló nagy rendszerek, valamint a katonai alkalmazások 5000 funkciópont nagyságrendbe esnek. Ez a durva becslés addig érvényes, amíg további információk nem állnak rendelkezésre a funkciópont méret meghatározásához.

Alkalmazás terjedelme	Méret funkciópontban	Méret, forráskód sorban
1. Szubrutin	1	100
2. Modul	3	300
3. Újrafelhasználható modul	5	500
4. Eldobható prototípus	7	700
5. Evolúciós prototípus	10	1000
6. Önmagában megálló, működő program (standalone)	25	2500
7. Rendszer alkotórész	100	10000
8. Rendszer kibocsátási verzió	1200	120000
9. Új rendszer	5000	500000
10. Összetett rendszer	15000	1500000

1-5. táblázat: Csak a terjedelem sajátosság használata a funkciópont méret becslésére.

Ezek az adatok is közelítő értékek, hisz ha az SAP / R3 rendszerére gondolunk, akkor az az összetett rendszerek kategóriájába tartozik és a mérete messze meghaladja a 15000 funkciópontot (közelebb van a 250 000 funkcióponthoz).

Egy másik példa összetett rendszerre a Microsoft Office rendszere, amely szövegszerkesztőből, táblázatkezelő kalkulátor programból, adatbázis-kezelőből, grafikus eszközből, és személyi időtervező eszközből áll. Az összes elem valamilyen szinten integrált és együtt is tud dolgozni. Mindegyik elem kb. 3000 funkciópont körül van, amihez jön még az OLE szolgáltatás, amin keresztül az egyes elemek megoldják az információk cseréjét, megosztását. Így mindösszesen a teljes rendszer a 20 000 funkciópont nagyságrendjébe esik.

1.4.2.2 Méretezés analógia alapján

A kifejlesztendő rendszer méretezésének egyik lehetséges módja, ha ismert alkalmazások funkciópont méretét vesszük alapul.

Alkalmazás	**Típus**	**Célterület**	**Méret, funkciópontban**
Grafikus tervező eszköz	kereskedelmi	CAD	2700
IEF (Information Engineering)	kereskedelmi	CASE	20000
IMS	kereskedelmi	Adatbáziskezelő	3500
CICS	kereskedelmi	Adatbáziskezelő	2000
Lotus Notes	kereskedelmi	Csoportmunka	3500
MS Office Professional	kereskedelmi	Irodai alkalmazás	16000
Word 7.0	kereskedelmi	Irodai alkalmazás	2500
Excel 6.0	kereskedelmi	Irodai alkalmazás	2500
MS Project	kereskedelmi	Projekt irányítás	3000
Repülőgép radar	katonai	Honvédelem	3000
Löveg irányítás	katonai	Honvédelem	2336
Repülőgép hely- és jegyfoglalás	MIS (vezetői információrendszer)	Üzleti	25000
Biztosítási kárigények	MIS (vezetői információrendszer)	Üzleti	15000
Telefon díj kiszámlázása	MIS (vezetői információrendszer)	Üzleti	11000
Személyi jövedelemadó bevallás	MIS (vezetői információrendszer)	Üzleti	2000
Főkönyv	MIS (vezetői információrendszer)	Üzleti	1500
Rendelés feldolgozás	MIS (vezetői információrendszer)	Üzleti	1250
Személyügy (Humán erőforrás kezelés)	MIS (vezetői információrendszer)	Üzleti	1200
Értékesítés	MIS (vezetői	Üzleti	975

Alkalmazás	Típus	Célterület	Méret, funkciópontban
	információrendszer)		
Költségtervezés	MIS (vezetői információrendszer)	Üzleti	750
Windows 95	Rendszerszoftver	Operációs rendszer	85000
MVS	Rendszerszoftver	Operációs rendszer	55000
UNIX V5	Rendszerszoftver	Operációs rendszer	50000
Dos 5	Rendszerszoftver	Operációs rendszer	4000

1-6. táblázat: Méretezés analógia alapján

1.4.2.3 A leszállítandó papír dokumentáció méretezése

A szoftverkészítés nagyon papírigényes iparág. Több mint 50 különféle projekttervezési, ütemezési, követelményelemzési, követelmény specifikációs és felhasználók számára készítendő dokumentum keletkezhet egy projekt során. Nagy rendszerek esetében — különösen a nagy katonai alkalmazásoknál — a papír dokumentáció készítésének költsége meghaladja a forráskód elkészítésének költségét.

A következő ökölszabály durva útmutatást ad az elkészítendő papír dokumentáció megbecslésére.

2. szabály: Szoftver tervek, specifikációk és kézikönyvek méretezése

A rendszer funkciópont méretét az 1,15-ik hatványra emelve a fejlesztés során elkészítendő papír dokumentumok oldalszámának közelítő értékét kapjuk.

Azonban ha a fejlesztők ISO 9000-9004 szabványokat alkalmaznak, az 1,17 hatványkitevőt célszerű használni.

1.4.2.4 A lopakodó felhasználói követelmények növekedésének méretezése

A szoftver világ egyik legsúlyosabb problémája a kezdeti követelményelemzés után fellépő, folyamatosan változó és növekvő felhasználói igények korrekt kezelése.

A funkciópont metrika különösen hasznos a követelmények növekedési sebességének méretezésére. Egyre gyakrabban előfordul, hogy szolgáltatás kihelyezési, de közönséges fejlesztési szerződéseknél is megjelenik a „funkciópont / költség” használata. Fejlesztési szerződésekben ez az érték általában egy mozgó skálát jelent, melyben egy új szolgáltatási igény elkészíttetésének költsége annál magasabb, minél később kerül be a követelmények közé a fejlesztési életciklus során.

3. szabály:A lopakodó felhasználói követelmények méretezése

A tervezéstől a kódolás szakaszáig a lopakodó felhasználói követelmények átlagos növekedési rátája: 2 % / hónap.

Tegyük fel, hogy egy fejlesztési szerződésben megállapodás születik arról, hogy egy pontosan 1000 funkciópontos rendszert kell elkészíteni. Az ökölszabályból az következik, hogy minden eltelt hónapban a tervezés és a programkódolás szakaszában, az eredeti követelmények havonta 20 funkcióponttal fognak nőni.

Mivel egy 1000 funkciópontos projekt kb. 16 hónapig tart, ennek felét kb. 8 hónapot kell a tervezési és programkódolási szakaszra fordítani. Az ökölszabály alapján ebből az következik, hogy kb. 16 százalékos növekedés áll be funkciópontban mérve a követelmények tekintetében, vagyis az alkalmazás mérete 1160 funkciópont lesz az eredeti 1000 funkcióponthoz képest.

Ténylegesen és projektektől függően a lopakodó követelmények következtében beálló növekmény a közel 0 százaléktól az 5 százalékig terjed. A jobb minőségű követelményelemzési és tervezési módszerek jelentősen csökkentik -akár 1 % alá is - a lopakodó követelmények növekedési rátáját. Ilyen módszerek lehetnek:

- Közös alkalmazás fejlesztés (JAD, Joint Application Development);
- Szigorúan kézben tartott prototípus alapú fejlesztés ;[7]
- Követelményelemzés minőségi átvétele és bevizsgálása (pl. strukturált módszerek vonatkozó eljárásai: SSADM követelményelemezési és követelményspecifikációs szakaszának termékei és a felhasználókkal végrehajtott formális és informális minőség szemlék [lsd. még a 19. lábjegyzetet]).[8]

Ha azonban a tervezést és a fejlesztést olyan stílusban hajtják végre, amit a „gyors piszkozat[9]" készítés jellemez, amint az gyakran a fegyelmezetlen gyors alkalmazás fejlesztéseknél, vagy kiszolgáló-ügyfél (kliens/szerver) technológiájú alkalmazásoknál előfordul, akkor a lopakodó követelmények növekedési üteme meghaladhatja akár az 5 %-ot is, ami a projektet olyan helyzetbe hozza, hogy a sikeres befejezés valószínűsége erősen közelít a nullához.

A belső fejlesztésű információrendszereknél általában nincs látható hátrányos következménye a lopakodó felhasználói követelményeknek, a tendencia az, hogy kezelhetetlenül növekednek, a rendszer határai úgy viselkednek, mint a táguló világegyetem. Szerződésben vagy szolgáltatás kihelyezésben kivitelezendő fejlesztéseknél viszont közvetlen pénzügyi hatása van a tervezési fázis késői szakaszában végrehajtandó követelménybővítési igényeknek.

Ezért indokolt a szoftverfejlesztési szerződésekben egy mozgó költség skálát meghatározni a fentebbi problémakör eredményes kezelésére. Tegyük fel, hogy egy 1000 funkciópontos rendszer kifejlesztésére vonatkozó szerződésben a kezdeti

---

[7] gyors alkalmazás fejlesztésegyes módszerei, RAD, Rapid Application Development, DSDM, Dynamic System Development Method, pl. http://www.dsdm.org/index.asp,

[8] http://www.itb.hu/ajanlasok/a4,

[9] „quick and dirty"

megállapodás az, hogy egy funkciópont értéke 150 000 Ft[10], vagyis a teljes vállalkozási díj kezdetben 150 000 000 Ft (150 millió Ft).

A szerződés ekkor a következő táblázatot tartalmazhatja a követelménybővítési igények miatt fellépő növekvő költségek kezelésére[11]:

Kezdetben 1000 funkciópont	150 000 Ft / FP
A szerződés aláírása után 3 hónappal hozzáadandó új igényekre	180 000 Ft/ FP
A szerződés aláírása után 6 hónappal hozzáadandó új igényekre	210 000 Ft/ FP
A szerződés aláírása után 9 hónappal hozzáadandó új igényekre	270 000 Ft / FP
A szerződés aláírása után 12 hónappal hozzáadandó új igényekre	360 000 Ft / FP
Felhasználói kérésre törölt vagy elhalasztott felhasználói követelmények	45 000 Ft / FP

1-7. táblázat: Funkciópont ára fejlesztési szakaszonként

Hasonló szerződéses megállapodásokat alkalmazhatunk a rendszer továbbfejlesztésére vagy karbantartására vonatkozó szolgáltatás kihelyezési szerződésekben is.

Szokásos karbantartási és hiba javítási tevékenység esetében	37 500 Ft / FP
Számítóközpont (mainframe) architektúráról való áttérés kliens/szerver architektúrára	60 000 Ft / FP

Fejlesztési és karbantartási szerződéseknél a funkciópont metrika használatának előnye, hogy a felhasználói követelményekre alapul és nem lehet a szerződő partner részéről egyoldalúan módosítani.

A szerződés vagy rendszer funkciópont értékét nem lehet egyoldalúan csak a szállító oldaláról megállapítani, hanem közvetlenül a felhasználói követelményekből kell kiindulni, így a megrendelő számára is áttekinthető.

1.4.2.5 A bevizsgáláshoz szükséges teszt esetek méretezése

A funkciópont metrika használata különösen hasznos a teszt esetek számának megállapítására, mert a funkciópont elemzés során azokat a szerkezeteket vizsgáljuk, amelyek párhuzamba állíthatók azokkal az elemekkel, amelyek helyes működését a bevizsgálás alatt ellenőrizni kell.

4. szabály: Teszt esetek számának meghatározása

[10] kb. 500 US $, vagy 500 Euro.

[11] Nemzetközi adatokból becsült értékek, a 2000. év árfolyamai alapján

A rendszer funkciópont méretét az 1,2-ik hatványra emelve kapjuk a létrehozandó teszt esetek számát.

1.4.2.6 A potenciális rendellenességek számának becslése

Az alkalmazás potenciális rendellenességei alatt azoknak a hibáknak az összességét értjük, amelyek az öt jelentős leszállítandó termékben előfordulhatnak:

1. Követelményelemzési hibák;
2. Tervezési hibák;
3. Program kódolási hibák;
4. Felhasználói dokumentáció hibái;
5. Rosszul kijavított hibák, illetve a hibajavítás mellékhatásaként fellépő másodlagos hibák.

A hibakeresés és javítás költsége általában a legjelentősebb kimutatható költségtétel, ezért a rendellenességek kiküszöbölésére tett erőfeszítések figyelmen kívül hagyása alapjaiban teszi tönkre az összes becslési, ütemezési és költség meghatározási tevékenységet.

5. szabály: A potenciális rendellenességek számának méretezése

A rendszer funkciópont méretét az 1,25-ik hatványra emelve a potenciális rendellenességek számának közelítő értékét kapjuk.

Hasonló szabály vonatkozik a továbbfejlesztésre, ahol a továbbfejlesztési igény funkciópont méretét az 1,27-ik hatványra emelve kapjuk a potenciális rendellenességek számának közelítő értékét. A nagyobb hatványkitevőt az alaptermékben látensen megbújó hibák indokolják, amelyek a továbbfejlesztés során előtérbe kerülhetnek.

A rendellenesség eredete	A rendellenességek száma funkciópontonként (hiba / FP)
Követelmények	1,00
Tervezés	1,25
Programkódolás	1,75
Felhasználói dokumentáció	0,60
Helytelen javítás	0,40
Összesen	**5,0**

1-8. táblázat: A fejlesztés során az átlagos hibaszám (Statisztika az USA-ból)

1.4.2.7 A rendellenség kiküszöbölése hatékonyságának megbecslése

A potenciális rendellenességek a teljes fejlesztési életciklus során keletkező hibákat jelentik, amelyeket kezelni kell. A potenciális rendellenességek számát a termék kiszállítása előtt le kell csökkenteni úgy, hogy 85 százalék (ipari norma) és 99 százalék közt (legjobb ipari gyakorlat) legyen az eltüntetett

rendellenességek száma. Így a megrendelőnek átadott termékben a potenciális rendellenességeknek csak a töredéke szerepel.

A következő érdekes ökölszabályok becslést adnak különböző bevizsgálási eljárások során megtalált hibák számára, és ezek eltávolításának hatékonyságára.

6. szabály: A rendellenesség eltávolítás hatékonyságának becslése

A szoftver tesztelése során minden egyes tesztlépés a jelenlévő hibák 30 százalékát találja meg.

A szoftvertesztelés hatékonysága a hibafeltárás vonatkozásában meglepően alacsony. Az előző ökölszabály következménye — csak minden harmadik jelenlevő hibát találja meg a szoftverteszt — az, hogy magas minőségi szint elérése érdekében egymás után 6 és 12 közötti tesztciklust kell elvégezni ahhoz, hogy kielégítő eredményre jussunk.

A komoly szoftvergyártó cégek több szakaszból álló bevizsgálási eljárásokat alkalmaznak:

- a követelményspecifikáció, a terv minőségi bevizsgálása, formális minőségi szemléje,
- program kód átvizsgálása,
- különböző szoftvertesztelési módszerek, program egység teszt, modul teszt, rendszert teszt, regressziós teszt, stb.

Tegyük fel, hogy egy 100 funkciópontos új alkalmazást fejlesztünk, ekkor a szabály alapján $100^{1,25} = 316$ hibával kalkulálhatunk. Az ökölszabály alapján a tesztelés első lépésében (pl. a program egység, modul tesztben) a hibák 30 százalékát fedezzük fel, vagyis a 316 hibából 95 hibát, így marad még 221. A második teszt fázisban amikor az új funkciókat vizsgáljuk, a maradék 221 hiba 30 százalékát találjuk meg, azaz 66 hibát. Igy két szoftver tesztelési lépés után 155 hiba maradt még a rendszerben. Tehát, ha kizárólag a szoftver tesztelésre hagyatkozunk akkor legalább 6, de lehet hogy 10 tesztelési szakaszra van szükség.

A terv és program kód formális átvizsgálása és minőségi szemlézése esetében a rendellenességek eltávolításának hatékonysága lényegesen magasabb mint a szoftver tesztelésnél.

7. szabály: A formális vizsgálatok és minőségi szemlék rendellenesség eltávolítás hatékonyságának becslése

A terv minden egyes formális átvizsgálása a létező hibák 65 százalékát feltárja és eltávolítja.

A program kód minden egyes formális átvizsgálása a létező hibák 60 százalékát feltárja és eltávolítja.

Ez az ökölszabály mutatja, hogy miért használják a szoftverfejlesztő szervezetek a tesztelést megelőzően, így a terv és a programkód formális átvizsgálását és minőségi szemlézését 95 százalékos kumulált rendellenesség eltávolítás hatékonysággal dicsekedhetnek.

Ráadásul a formális átvizsgálás viszonylag nem drága, és olyan drámai hatása van a tesztelési sebesség felgyorsulására és a karbantartási költségek csökkenésére, hogy az összes szoftvertechnológiai eszköz közül a legjobb a megtérülési mutatója.

A minőséggel összefüggő ökölszabályok lezárásaként, a szoftver forgalomba bocsátása, vagy az átadás / átvételi eljárások sikeres lezárása után jelentkező hibák kijavítási sebességére még egy szabály vonatkozik.

8. szabály: A kibocsátás utáni rendellenesség javítás sebességének becslése

A karbantartó programozók egy emberhónapra vetítve 8 hibát tudnak kijavítani.

Ez a szabály a szoftveriparban már több mint 30 éve ismert, és még mindig érvényes. Természetesen vannak olyan szoftver karbantartással foglalkozó szervezetek, amelyek komplexitás elemzőket, program kód átszervező eszközöket és karbantartást segítő munkapadokat használnak, így ezt a javítási rátát akár meg is duplázhatják.

1.4.2.8 Ökölszabályok az ütemezésre, erőforrásokra és a költségekre

Miután a különböző leszállítandó termékeket és a potenciális rendellenességeket sikerült számszerűsíteni, a projekt becslése során a következő szakasz az ütemtervekre, erőforrásokra, költségekre vonatkozó előrejelzések készítése.

1.4.2.8.1 Határidő becslése

A véghatáridő megállapítása a megrendelőnél, a projektvezetőnél és a fejlesztő cég vezetőinél egyaránt legnagyobb érdeklődésre ad okot.

9. szabály: A projekt befejezésének megbecslése

A rendszer funkciópont méretét az 0,4-ik hatványra emelve a fejlesztésre fordítandó naptári hónapok közelítő értékét kapjuk.

Természetesen ez a hatványkitevő változik a speciális feltételeknek megfelelően, pl. katonai alkalmazásoknál a szigorúbb dokumentálási előírások miatt 0,45 körül van. A megfigyelt értékek 0,32 és 0,45 között szórnak. Az olyan szabványok alkalmazása, ami növeli a rendszer készítés bonyolultságát a kitevő értékét 0,4 fölé viszik, ilyen pl. az ISO 9001 (ISO 9000-3) szabvány alkalmazása.

Hatványkitevő	**Naptári hónapok száma**	**A tartományba bele eső projektek**
0,32	9,12	
0,33	9,77	
0,34	10,47	
0,35	11,22	
0,36	12,02	O-O szoftver
0,37	12,88	Kliens / szerver szoftver
0,38	13,80	Szolgáltatás kihelyezésben kivitelezett szoftver
0,39	14,79	Vezetői információrendszer (MIS)

Hatványkitevő	Naptári hónapok száma	A tartományba bele eső projektek
0,4	15,85	Kereskedelmi forgalomba hozott szoftver
0,41	16,98	Rendszer szoftver
0,42	18,20	
0,43	19,50	Katonai alkalmazás
0,44	20,89	
0,45	22,39	

1-9. táblázat: A követelményelemzés megkezdésétől a végtermék leszállításáig eltelt naptári hónapok

(1000 funkciópontos rendszer készítését feltételezve)

Egy szerződésben nem kell feltétlenül ragaszkodni az ökölszabály által adott eredményhez, de a kapott szám alkalmas annak ellenőrzésére, hogy a szállító javaslata az ésszerűség határain belül mozog-e.

1.4.2.8.2 A fejlesztő csoport nagyságának becslése

A következő ökölszabály az alkalmazás kivitelezéséhez szükséges fejlesztő személyzet létszámának durva megbecsülésére szolgál. Ez a szabály az egy személy által elvégezhető munkamennyiség fogalmára alapul.

Az ide értendő munkakörök: szoftverfejlesztők, minőség biztosítók, szoftver tesztelők, műszaki dokumentáció készítők, adatbázis adminisztrátorok és projekt menedzserek.

10. szabály: A szoftverfejlesztők létszámának becslése

Az alkalmazás elkészítéséhez szükséges személyzet létszámát úgy becsülhetjük meg, hogy az alkalmazás funkciópont értékét elosztjuk 150-nel.

A szabály egyik következményeként a karbantartáshoz szükséges létszám megbecslésére is lehetőség van.

11. szabály: A szoftver karbantartók létszámának becslése

Az alkalmazás karbantartásához szükséges személyzet létszámát úgy becsülhetjük meg, hogy az alkalmazás funkciópont értékét elosztjuk 750-nel.

A szabályból az következik, hogy 1 karbantartó kis javításokat végezve 750 funkciópont értékű alkalmazás részt tud működőképesen tartani.

1.4.2.8.3 A szoftverfejlesztés ráfordítási igényének becslése

A következő szabály két előző szabály kombinációja (9. szabály:, 10. szabály:).

12. szabály: A fejlesztés erőforrás igényének becslése

Az alkalmazás kivitelezésére becsült naptári hónapok számát össze kell szorozni a személyzet becsült létszámával, ekkor a fejlesztésre fordítandó emberhónapok közelítő értékét kapjuk.

A szabály illusztrálásához tegyük fel, hogy van egy 1000 funkciópontos fejlesztésünk.

- A 9. szabály: alapján az 1000 funkciópontot a 0,4 hatványra emelve megkapjuk, hogy a projekt kb. 16 naptári hónapig tart.
- A 10. szabály: alapján az 1000 funkciópontot elosztjuk 150-el és kb. 6,6 teljes munkaidőben foglalkoztatott fejlesztő szükségletet kapunk eredményül.
- A 16 naptári hónapot és a 6,6 személyt összeszorozva kb. 106 emberhónapot kapunk a projekt erőforrás igényére.[12]

2 A funkciópont metrika alapelveinek áttekintése

Az életben nagyon sok, meglehetősen bonyolult mérőszámot, fizikai, kémia mennyiséget, mértéket, stb. használunk. Például használjuk a lóerő fogalmát, az oktánszámot, vitatkozunk az egyes ételek kalória tartalmáról, vagy a koleszterin szintről. A napi életben elegendő ha ezeknek a metrikáknak, mértékeknek a használatát, lényeges tulajdonságaikat, jelentőségüket ismerjük. Nagyon kevés embernek kell tudnia azt, hogy hogyan kell meghatározni a benzin oktánszámát, vagy a motor lóerejét, amíg megbízunk ezekben az adatokban.

Ehhez hasonlóan kevés embernek kell értenie azt, hogy hogyan kell a funkciópontokat kiszámolni, de minden projekt vezetőnek és fejlesztőnek értenie kell a használatához, ismernie kell a termelékenységi és minőségi tényezőket.

A következő termelékenységi adatokat minden projekt vezetőnek illik ismerni:

- 5 FP / emberhónap termelékenységet mutató (több mint 26 munkaóra / FP) projektek az USA ipari átlaga alatt vannak.
- 5-10 FP / emberhónap termelékenységet mutató (13-26 munkaóra / FP) projektek az USA ipari átlagának felelnek meg.
- 10-20 FP / emberhónap termelékenységet mutató (7-13 munkaóra / FP) projektek az USA ipari átlaga fölött vannak.
- 20 FP / emberhónap termelékenységet meghaladó (7 munkaóra / FP) projektek magasan az USA ipari átlaga fölött vannak.

Természetesen ezeknél az adatoknál célszerű figyelembe venni a projekt méretét (funkciópontban), hisz egy 100 funkciópontos projektnél a termelékenység 20 FP / emberhónap körül van, egy 1000 funkciópontos projektnél meglehetősen ritka az ilyen termelékenységi adat.

[12] Egy emberhónap alatt 22 munkanapot, 6 termelésre fordítható munkaórával számolva, ami havonta 132 munkaórát jelent.

Az alap termelékenységi adatokat célszerű minden szoftverfejlesztő szakembernek ismerni még akkor is, ha az általánosságnak az alábbiakban leírt szintjén mozognak:

- 1,5 hiba / FP rendellenességet mutató rendszer gyenge minőségű.
- 0,75 hiba / FP rendellenességet mutató rendszer normális, átlagos minőségű.
- 0,10 hiba / FP rendellenességet mutató rendszer kiemelkedő minőségű.

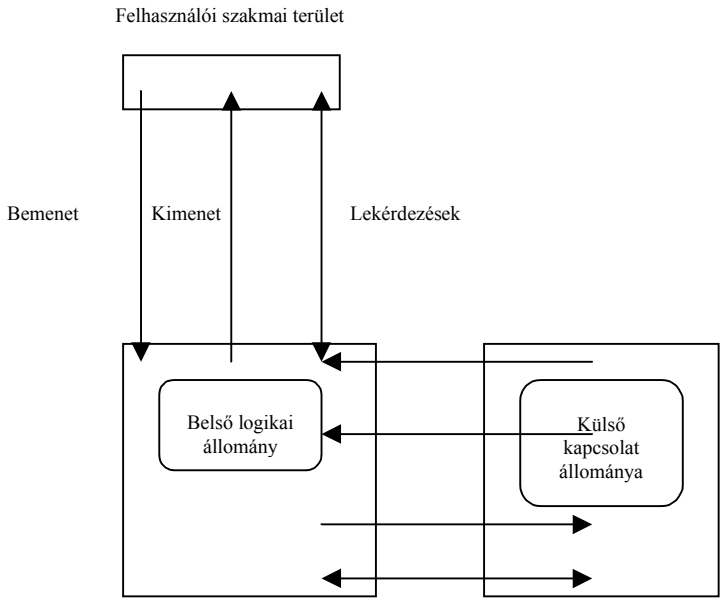
A bevezetőben már találkoztunk a funkciópont számítás alapfogalmaival (1.2).

A funkciópont számítás a következő öt alapvető rendszerelemre alapul[13]:

1) Tranzakció funkciópontok:	Angol megnevezés	Angol rövidítés
a) Bemeneti adatelemek	External inputs	EI
b) Kimeneti adatelemek	External outputs	EO
c) Lekérdezések	External inquiries	EQ
2) Adat funkciópontok		
a) Logikai állományok	Internal logical files	ILF
b) Kapcsoló felületek (interface-k)	External interface files	EIF

2-1. táblázat: Öt alapfogalom

A rendszer három külső attribútuma azokkal a tranzakciókkal foglalkozik, amelyeket a rendszer végrehajt. A másik kettő az alkalmazás által felhasznált adatokkal.



7. ábra. A funkciópont elemzés szempontjából a rendszer határai

[13] IFPUG megközelítés http://www.ifpug.org (2001.01.23.)

Ha a rendszer határait megállapították, akkor a kimeneti, bemeneti adatelemek, lekérdezések, logikai állományok és adatkapcsolatok (kapcsoló felületek, állományok) aktuális száma meghatározható. Ezen a ponton hívható be egy funkciópont meghatározását ismerő szakértő, vagy alkalmazható egy megfelelő eszköz.

A valóságban a folyó projekteknél a fejlesztési munka az új szoftver megírása mellett a meglévő szoftverek módosításából, alkalmanként bizonyos szolgáltatások megszüntetéséből, újak beillesztéséből áll. Az összes ilyen tevékenységet természetesen ki lehet értékelni a funkciópont számítás segítségével, azonban gondosan kell elemezni a táblázatban látható tényezőket.

Projekt típus	Projekt funkciópontok	Alkalmazás funkciópontjai
		Üzembe helyezett funkció-pontok (ÜFP)
Fejlesztési projekt	Projekt FP=új (hozzáadott) FP + áttérés (konverzió) FP	Alkalmazás FP = új (hozzáadott) FP
Továbbfejlesztési, bővítési projekt	Projekt FP= új (hozzáadott) FP + módosított FP + törölt FP + áttérés (konverzió) FP	Alkalmazás FP = eredeti FP + törölt FP + új (hozzáadott) FP + Δ módosított FP

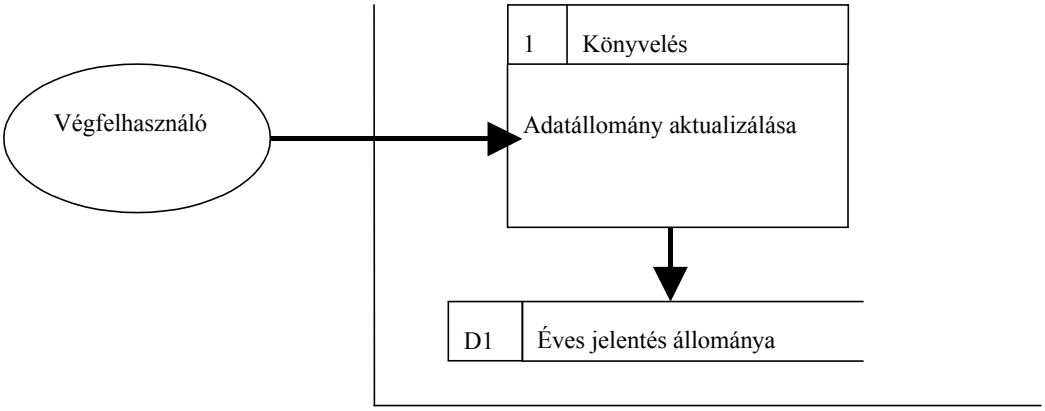
2-2. táblázat: Funkciópontok meghatározása új és továbbfejlesztési projektekre

A funkciópontok meghatározását nagymértékben megkönnyíti a fegyelmezett és szisztematikus rendszerfejlesztési módszerek és az általuk előírt technikák alkalmazása.

A fentebbi tranzakció-funkciópontok kiinduló meghatározásához nagyszerűen alkalmazhatók a különböző adatfolyam ábrák, adatáramlás leíró ábrázolási technikák. Ilyen módszer a már hivatkozott SSADM-ben[8] illetve a hasonló módszerekben[14] megtalálhatók adatfolyam diagrammok formájában.

A funkciópont becslés és a fegyelmezett szoftverfejlesztési költségbecslés szempontjából a gyors fejlesztési módszertanok meglehetősen aggályosak; különösen azok, amelyek elhagyják a követelmények és a követelmény specifikáció írásos rögzítését és fejest ugranak valamilyen gyors ciklusú, evolúciós prototípus fejlesztési megközelítésbe. Ez a megoldás azonban nemcsak a költségbecslést és ezen keresztül a projektvezetés munkáját nehezíti meg, hanem a program kód bevizsgálását és a program tesztelést is, ami a minőségi szempontok gyenge érvényesülésével jár, így gyenge minőségű végeredményt hoz létre. Ilyen esetben nincs megfelelő alapja a funkciópont számításhoz szükséges alapadatok szisztematikus felderítésének.

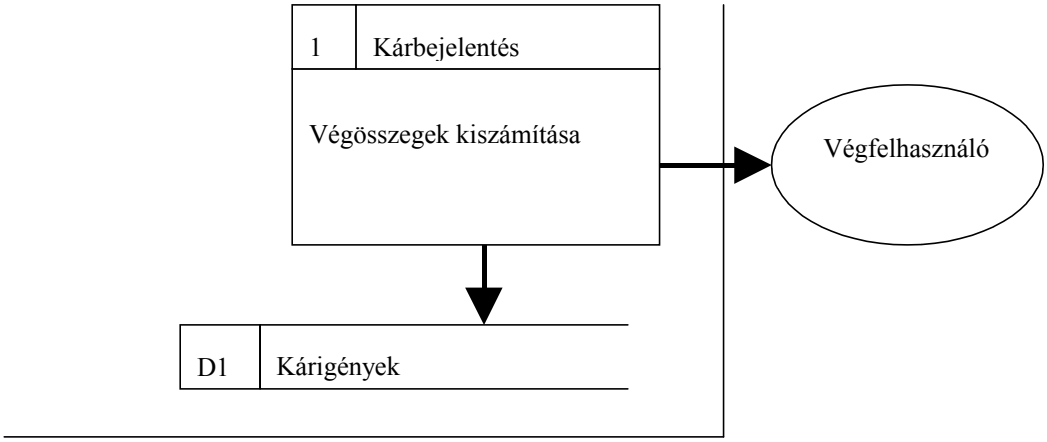
[14] Warnier-Orr,Merise, Merise, Yourdon, Booch-Rumbaugh-Jacobsen féle UML (Universal Modelling Language) illetve az EU jelentősebb országaiban használt, az államigazgatás által támogatott módszerek, mint pl. Vorgehens Modell vagy Life Cycle Modell (NSZK), SDM (Hollandia), stb. Rövid ismertetés magyar nyelven a fentebbi módszerekről található: Molnár Bálint, 'Bevezetés a rendszerelemzésbe', in: Gábor András (szerk.) „Információmenedzsment", Aula Kiadó, 1997, pp 107-239.



8. ábra. A bemeneti adatelemek felismerése egy adatfolyam ábráról (EI)

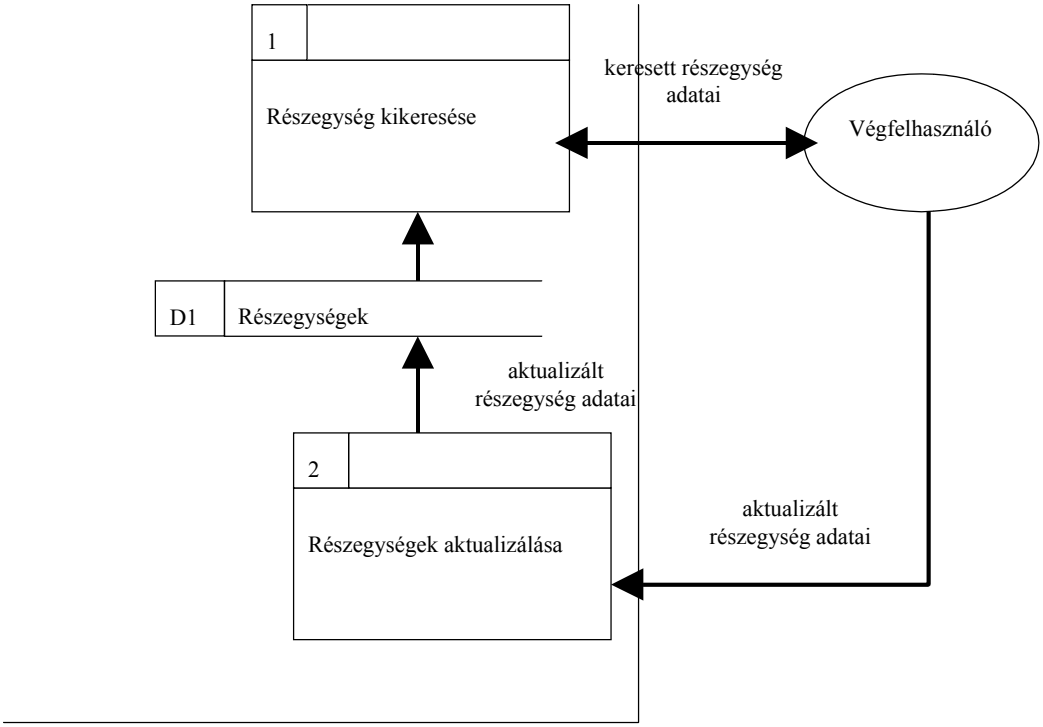
A 8. ábra az adatfolyam ábrák egyik típusának jelölésével érzékelteti, hogy a rendszer határát átlépő adatfolyamok explicite felismerhetők egy ilyen ábráról és a funkciópont számításhoz szükséges leszámlálásuk így elvégezhető.

A 9. ábra kimeneti adatelemek felismerhetőségét illusztrálja az előbbihez hasonló módon. A kimeneti adatok a szoftver alkalmazások egyik fontos jellemzőjét alkotják, és a funkciópont, költségbecslés és projekt méretezés egyik fontos paraméterét jelentik. A rendszereknek ezt az oldalát tárják fel olyan kielégítő mélységben, amely már megfelelő a funkciópont elemzés első közelítésben történő elvégzéséhez. Ez különösen hasznos olyan esetekben, amikor még nem áll rendelkezésre teljes körű információ a funkciópont becslés elvégzéséhez, ekkor csak a kimeneti adatelemek számosságának felhasználásával készíthető egyes módszerekben közelítő becslés a szoftver funkciópont méretére.



9. ábra. Példa kimeneti adatelemek felismerésére (EO)

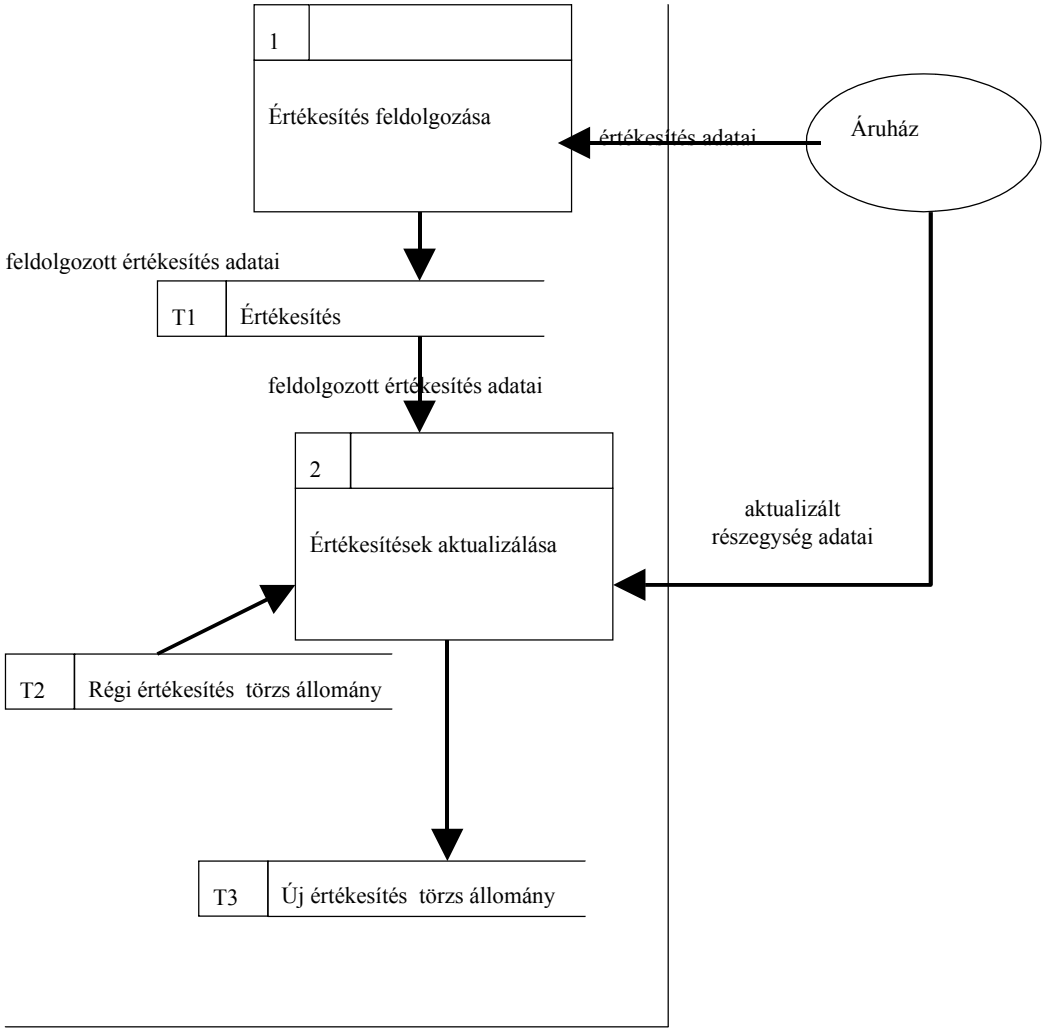
A lekérdezések (10. ábra) a funkciópont elemzés alapparaméterei közül a legtrükkösebbeknek és a legkifinomultabbaknak számítanak. Ugyanis két egymással szoros összeköttetésben lévő elemből állnak össze, nevezetesen egy kérdésről és a ráadott válaszból. Ráadásul az is megfigyelhető, hogy gyakran a lekérdezések lényegében egy bemenet és kimenet pár speciális eseteként foghatók fel.



10. ábra. Lekérdezések felismerése (EQ)

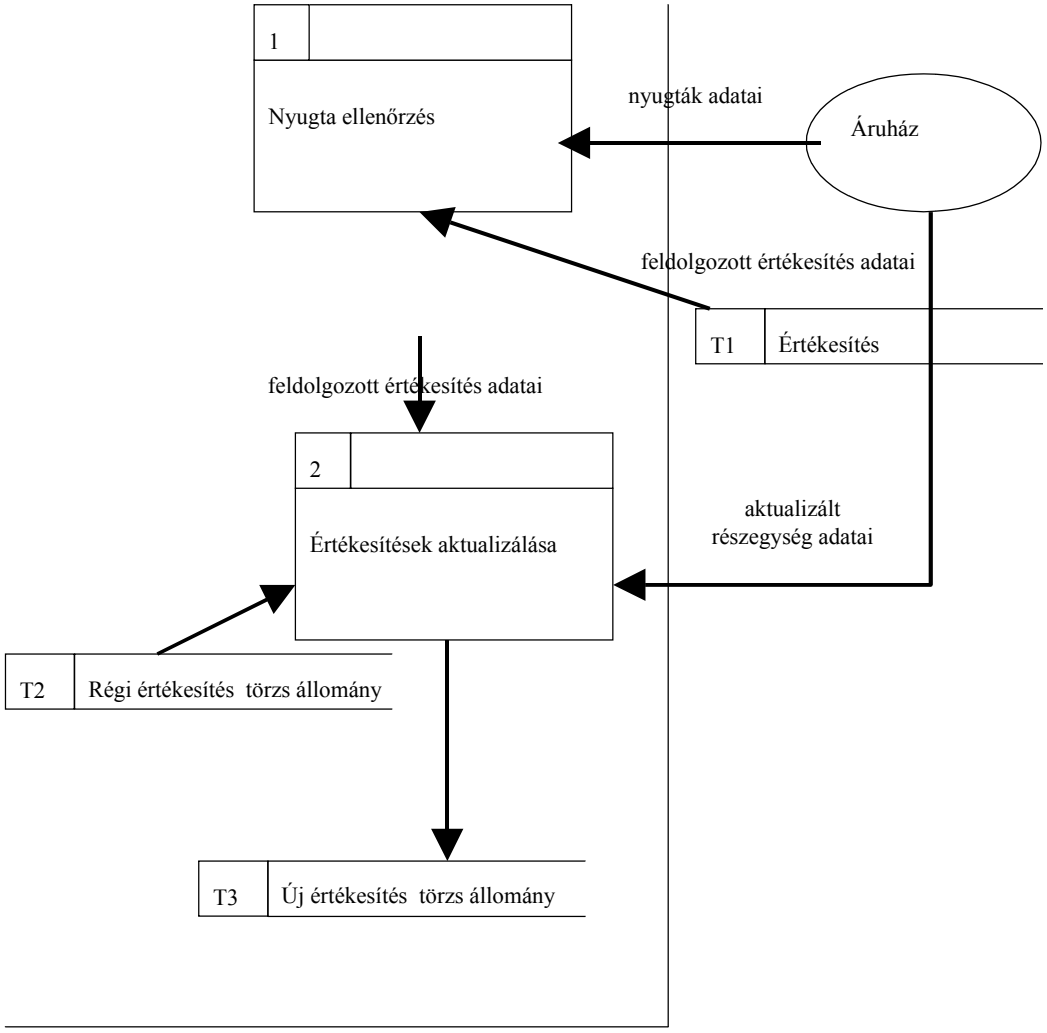
Az információrendszerek egyik alapalkotórészét jelentik a logikai állományok (11. ábra).

Ezért a fejlesztés során a logikai állományok megtervezése, felépítése, kiszolgálása és tartalmuk felhasználása a munka nagyon domináns részét jelentik, az elemzés során a legnagyobb súlyértékkel szerepelnek.



11. ábra. Logikai állományok felismerésére példa (ILF)

Az utolsó funkciópont elemzési paraméter amit az ábra illusztrál (12. ábra), a külső kapcsolatok, felületek megszámlálására szolgál. Ezeket a kapcsolatokat szintén nehéz megvalósítani, ezért a súlytényezője magasabb mint az egyszerűbb, ki- és bemenetek leszámlálására való paramétereknél (kivéve a logikai állományokat).



12. ábra. Kapcsoló felületek (interface-k) felismerésére példa (EIF)

Ezek az adatok, számok a követelményelemzés és a követelményspecifikáció során egyre pontosabbá váló dokumentációból felderíthetők. Természetesen vannak olyan eszközök, amelyek többé-kevésbé integráltak valamilyen követelményelemzést támogató eszközhöz, és ennek alapján az egész rendszerre ki tudnak számolni egy funkciópont értéket.

2.1 Bonyolultsággal korrigált funkciópont

A funkciópont elemzés körében a komplexitás vagy a bonyolultság fogalma meglehetősen korlátozott, és lényegében azzal foglalkozik, hogy az öt alapparaméter milyen kategóriába kerül, vagyis alacsony, átlagos vagy nagyon bonyolult paraméternek tekintendő-e.

A valóságban a bonyolultsági tényezővel való korrigálás gyakorlatot és speciális kiképzést igényel. Azonban első megközelítésként és az alapötlet érzékeltetésére egy, az alábbi táblázatban bemutatott kalkulációs lapot lehet használni.

Funkciópont összegző kalkulációs lap

Alkalmazás:		Alkalmazás azonosítója:	
Készítette:	(dátum)	Ellenőrizte:	(dátum)
Megjegyzés:			

Általános rendszer jellemzők	Értéke	Általános rendszer jellemzők	Értéke
1. Adat továbbítás terminál és gép között		11. Üzembehelyezés egyszerűsége	
2. Elosztott adatfeldolgozás		12. Használat egyszerűsége	
3. Teljesítmény célkitűzések		13. Több telephely, üzemeltetési hely	
4. A cél üzemeltetés környezet munkaterhelése		14. Változtatásra, módosításra alkalmas	
5. Transakciók gyakorisága		15. Más alkalmazások által támasztott igények	
6. Adatbevitel interaktivitásának foka		16. Biztonság, bizalmas adatok védelme,	
7. Végfelhasználót segítő szolgáltatások		17. Felhasználók oktatásának szükséglete	
8. Aktualizálás valós időben		18. Harmadik fél használja-e a rendszert	
9. Bonyolult adatfeldolgozási folyamatok		19. Dokumentáció	
10. Újrafelhasználhatóság más alkalmazásokban			

A befolyásoló tényezők összegzése (TDI)	0
Korrigáló tényező (VAF) = (TDI X .01) + .65	**0,65**

Komponensek:

Állományok (ILF + EIF)	0	Kimenetek (EO)	0
Bemenetek (EI)	0	Lekérdezések (EQ)	0

A korrigálatlatlan funkciópontok összege:	**0**
A teljes funkciópont = Korrigáló tényező x Korrigálatlan funkciópontok:	**0**

2-3. táblázat: Kalkulációs lap a funkciópontok összegzésére és korrigáló tényező figyelembe vételére

Az egyes funkciópont elemzési módszerek hasonló, de nem teljesen azonos szempontokat alkalmaznak a funkciópontok korrigálására, a bonyolultsági, technológiai tényezők figyelembevételére. Ezért az egyes módszereken belül

hasonlíthatók igazán össze az egyes funkciópont értékek, a különböző módszerek által kiszámolt értékek azonban átszámíthatók.[15]

	Software Productivity Research	**IFPUG (International Function Points User Group)**	**Mark II**
1.	Probléma bonyolultsága	Adat továbbítás terminál és gép között	Adat továbbítás terminál és gép között
2.	Program kód bonyolultsága	Elosztott adatfeldolgozás	Elosztott adatfeldolgozás
3.	Adat szerkezet, az adtok bonyolultsága	Teljesítmény célkitűzések	Teljesítmény célkitűzések
4.		A cél üzemeltetés környezet munkaterhelése	A cél üzemeltetés környezet munkaterhelése
5.		Tranzakciók gyakorisága	Tranzakciók gyakorisága
6.		Adatbevitel interaktivitásának foka	Adatbevitel interaktivitásának foka
7.		Végfelhasználót segítő szolgáltatások	Végfelhasználót segítő szolgáltatások
8.		Aktualizálás valós időben	Aktualizálás valós időben
9.		Bonyolult adatfeldolgozási folyamatok	Bonyolult adatfeldolgozási folyamatok
10.		Újrafelhasználhatóság más alkalmazásokban	Újrafelhasználhatóság más alkalmazásokban
11.		Üzembe helyezés egyszerűsége	Üzembe helyezés egyszerűsége
12.		Használat egyszerűsége	Használat egyszerűsége
13.		Több telephely, üzemeltetési hely	Több telephely, üzemeltetési hely

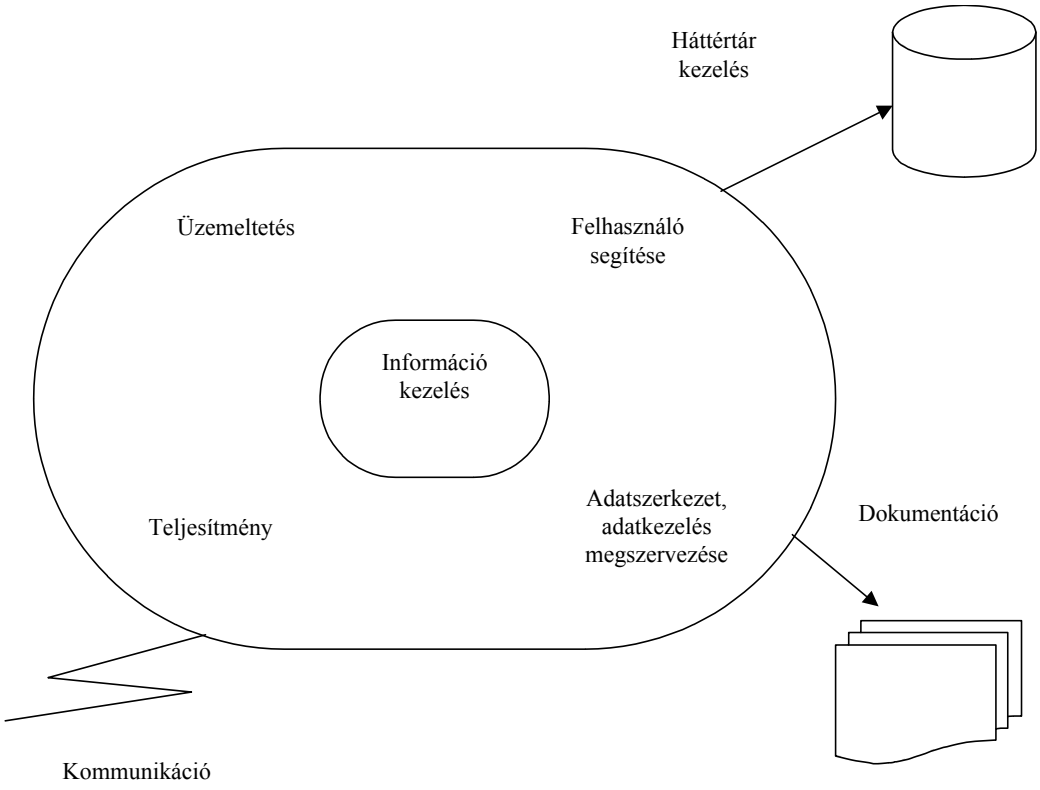
[15] http://194.143.167.33/library/Papers/Symons/AlbvMkII_v3b.pdf (2001.02.05.)

	Software Productivity Research	IFPUG (International Function Points User Group)	Mark II
14.		Változtatásra, módosításra alkalmas	Változtatásra, módosításra alkalmas
15.			Más alkalmazások által támasztott igények
16.			Biztonság, bizalmas adatok védelme, auditálhatóság
17.			Felhasználók oktatásának szükséglete
18.			Harmadik fél használja-e a rendszert
19.			Dokumentáció

2-4. táblázat: Technológiai bonyolultsága értékelése

A technikai bonyolultságot vagy komplexitást úgy tekinthetjük, mint olyan rendszerrel szemben támasztott követelményeket amelyek nem a rendszer információtartalmával, a feladatok valódi nagyságával függenek össze, de nem is a projekt környezetéből származnak.

A rendszer méretét – funkciópontban – az információkezelés, - feldolgozás méretéből kiindulva egy alkalmas technikai bonyolultsági tényezővel való szorzással nyerjük.



13. ábra. Az információkezelés és technikai bonyolultság összefüggése

2.1.1 IFPUG funkciópont méretezés szerinti funkcionális bonyolultság becslése

Az IFPUG funkciópont metrika alapfogalmai segítségével az adott alkalmazás funkcionális bonyolultsága a következő táblázatok segítségével értékelhető ki.

Hivatkozott állományok száma (adatbázis táblák, belső logikai állományok)	**Adatmezők**		
	1-4	**5-15**	**16+**
<2	Alacsony	Alacsony	Átlagos
2	Alacsony	Átlagos	Magas
>2	Átlagos	Magas	Magas

2-5. táblázat: A bemeneti adatelemek bonyolultsági táblázata (EI)

Hivatkozott állományok száma (adatbázis táblák, belső logikai állományok)	**Adatmezők**		
	1-5	**6-19**	**20+**

<2	Alacsony	Alacsony	Átlagos
2-3	Alacsony	Átlagos	Magas
>3	Átlagos	Magas	Magas

2-6. táblázat: A kimeneti adatelemek bonyolultsági táblázata (EO)

Rekord típusok	**Adatmezők**		
	1-19	**20-50**	**51+**
<2	Alacsony	Alacsony	Átlagos
2-5	Alacsony	Átlagos	Magas
>5	Átlagos	Magas	Magas

2-7. táblázat: A külső és belső logikai állományok bonyolultsági táblázata (ILF, EIF)

A lekérdezések (External inquiries, EQ) bonyolultságának megállapításához az általa használt bemeneti és kimeneti adatelemekre kapott bonyolultsági értékből a kettő maximumát képezik és azt fogadják el mint a lekérdezés bonyolultsági értékét. Ekkor a 2-5, 2-6 számú táblázat alkalmazásával EQ=max (EI,EO) számítással határozzuk meg a lekérdezés becsült értékét.

A korrigálatlan IFPUG funkciópont közelítő értékének megállapítására a következő táblázat alkalmazható:

IFPUG alapfogalmak, alkotóelemek	**Funkcionális bonyolultsági szint (tartozó súlytényezők)**		
	Alacsony	**Átlagos**	**Magas**
Logikai állományok, Internal logical files, ILF	×7	×10	×15
Kapcsoló felületek, External interface files, EIF	×5	×7	×10
Bemeneti adatelemek, External inputs, EI	×3	×4	×6
Kimeneti adatelemek, External outputs, EO	×4	×5	×7
Lekérdezések, External inquiries, EQ	×3	×4	×6

2-8. táblázat: IFPUG korrigálatlan funkciópont közelítő értékének becslésére szolgáló súlyok

IFPUG alapfogalmak, alkotóelemek	Az egyes kategóriákba és osztályokba tartozó elemek száma		
	Alacsony	Átlagos	Magas
Logikai állományok, Internal logical files, ILF	3		
Kapcsoló felületek, External interface files, EIF	1		
Bemeneti adatelemek, External inputs, EI	6	2	2
Kimeneti adatelemek, External outputs,EO	1	3	
Lekérdezések, External inquiries, EQ	3	1	

2-9. táblázat: Példa egy leszámolásra IFPUG funkciópont metrikában

13. szabály: A fejlesztés IFPUG funkciópont méretének becslése

IFPUG alapfogalmak, alkotóelemek	Funkciópont becslése		
	Alacsony	Átlagos	Magas
Logikai állományok, Internal logical files, ILF	3×7	0×10	0×15
Kapcsoló felületek, External interface files, EIF	1×5	0×7	0×10
Bemeneti adatelemek, External inputs, EI	6×3	2×4	2×6
Kimeneti adatelemek, External outputs,EO	1×4	3×5	0×7
Lekérdezések, External inquiries, EQ	3×3	1×4	0×6
Összesen	57	27	12
Mindösszesen			96

2-10. táblázat: A rendszer, fejlesztés méretének becslése IFPUG funkciópont metrikában

A két táblázat mátrix alkalmas összeszorzásával kapjuk a rendszer, fejlesztés közelítő funkciópont értékét az IFPUG Funkciópont metrikában.

2.2 Az MKII funkciópont méretezés legfontosabb lépéseinek összefoglalása

Ahogy arra már utaltunk funkciópont elemzés végrehajtásához szükség van olyan dokumentációra, amely kiterjed a rendszer egészére, minden elemére.

A következő dokumentációk jönnek szóba[16]:

1. Adatfolyam diagram, a rendszer határának világos definiálásával.
2. Adatszótár, be- kimeneti adatfolyamok, adatelemek leírása, egyéb kísérő dokumentumok.
3. Adatszerkezet leírása: entitás kapcsolat modell.
4. Adatszerkezeten a navigációs útvonal leírása, táblázatos vagy diagram formában.
5. Entitás élettörténet modellek.
6. Entitás / folyamat mátrix.

Minél teljesebb körű és részletesebb a dokumentáció, annál pontosabban állapítható meg a rendszer mérete. Már kevés információ esetén is lehet egy durva becslést, egy intelligens sejtést kialakítani, de természetesen ez csak egy korai projekt fázis céljának megfelelő közelítés lehet.

1) A primer, altípus és rendszer (adminisztrációs) entitások felismerése.
2) A logikai tranzakciók felismerése.
   a) Adatfolyam diagram.
   b) Folyamat illetve funkció lebontási diagram.
   c) CRUD mátrix (create, read, update, delete; létrehoz, olvas, aktualizál, töröl).
3) Az egyes logikai tranzakciók méretének kiszámítása.
   a) Bemeneti adatok: az összes bemeneti adatot tartalmazó adatelem az adatfolyamokból, vagy az entitások illetve a kapcsolatok bemeneti attribútumai.
   b) Kimeneti adatok: mint előbb a kimenetekre.
   c) Entitások: az adatfolyamokból és a logikai állományokból, a navigációs utakból és CRUD mátrixból az érintett entitások.
4) Összegezzük külön-külön a bemeneteket, kimeneteket és az entitásokat az összes tranzakcióra.
5) Számoljuk a korrigálatlan funkciópont értéket az előző teljes összegből, a megfelelő súlyok alkalmazásával:
   a) pl. NemKorrFP= 0,58 × bemenetek + 1,66 ×entitások +0,26 kimenetek.
6) Határozzuk meg a technikai bonyolultsági tényezőt.
7) Számítsuk a technikai bonyolultsági tényezővel korrigált funkciópont érétket.

[16] Ezeket a dokumentációk, vagy velük ekvivalensek előállítását az összes jelentősebb rendszerelemzési és tervezési módszertan előírja lsd. még 8, 14 lábjegyzetet.

a) pl. KorrFP=NemKorrFP× (technikai bonyolultsági tényező).

2.3 Charles Symons funkcionális szolgáltatáson alapuló becslési eljárása

A megközelítés alkalmazásához előfeltétel, hogy létezzenek olyan dokumentációk, amelyek leírják, hogy a rendszer mit fog nyújtani a felhasználónak. Ilyen dokumentáció lehet: adatfolyam diagram, részletes követelményjegyzék, funkció leírások jegyzéke ki- / bemenetet leíró kísérő dokumentációval.

2.3.1 A logikai tranzakciók felismerése

Egy funkció, vagy alkalmas bemeneti / kimeneti, rendszer határát átlépő adatfolyam pár reprezentálhat egy logikai tranzakciót.

A tranzakciókat felismerés után listába kell foglalni, majd osztályozni kell.

Típus	Osztály	Bemenetek száma	Entitások	Kimenetek
Aktualizáló (létrehoz, módosít)				
	Egyszerű	5	1	2
	Átlagos	15	3	2
	Bonyolult	25	5	2
Lekérdezés / Jelentés				
	Egyszerű	1	1	5
	Átlagos	3	3	15
	Bonyolult	5	5	25
Törlés				
	Átlagos	3	3	3

2-11. táblázat: A tranzakciók osztályozására útmutató

Ez a táblázat segít az egyes tranzakciók besorolásában. Természetesen ez egy tapasztalati táblázat amit számos projekt alapján állítottak össze, de szervezetenként, különleges körülményektől függően változhat. Ezért a befejezett projektek tapasztalati adataival összehasonlítva egy adott környezetre tovább finomíthatók az itt látható értékek.

Tranzakció azonosítója	**CRUD**	**Bonyolultság**
T1	O(olvas)	Á(tlagos)

Tranzakció azonosítója	CRUD	Bonyolultság
T2	L(étrehoz)	B(onyolult)
T3	A(ktualizál)	E(gyszerű)
T4	T(öröl)	Á(tlagos)
	……..	

2-12. táblázat: Az osztályozott tranzakciók

2.3.2 A rendszer méretének becslése korrigálatlan funkciópontban

Az első becslés az egyes tranzakció osztályok átlagos nagyságán (olvasás, aktualizálás, létrehozás, törlés) és osztályba sorolásukon (egyszerű, átlagos, bonyolult) alapul.

A tranzakciók osztályba sorolásából először egy „statisztikai" kimutatást kell készíteni:

Típus	Db szám		
	Egyszerű	Átlagos	Bonyolult
Aktualizáló	3	10	1
Lekérdezés	2	9	5
Törlés		4	

2-13. táblázat: A besorolt tranzakciók számossága kategóriánként

A tranzakció számokat tapasztalaton alapuló súlyokkal kell beszorozni, aminek eredményeként megkapjuk a rendszer becsült méretét korrigálatlan funkciópontban.

Típus	Tranzakció súlyok		
	Egyszerű	Átlagos	Bonyolult
Aktualizáló	4	12	20
Lekérdezés	3	10	17
Törlés	8	8	8

2-14. táblázat: Útmutató a tranzakciók súlyozására

A két táblázat mátrix alkalmas összeszorzásával kapjuk a teljes korrigálatlan funkciópont számot.

14. szabály: A fejlesztés funkciópont méretének becslése a funkcionális szolgáltatások alapján

Típus	Szorzatok		
	Egyszerű	Átlagos	Bonyolult

Típus	Szorzatok		
	Egyszerű	Átlagos	Bonyolult
Aktualizáló	3×4	10×12	1×20
Lekérdezés	2×3	9×10	5×17
Törlés		4×8	
Teljes korrigálatlan funkciópont: 365			

2-15. táblázat: Útmutató a tranzakciók súlyozására

2.3.3 Pontosabb közelítés

Az előbbi eljárásnál pontosabb, ha az egyes tranzakciókra elvégezzük a becslést, számítást, mennyi bemenet, kimenet tartozik a tranzakciókhoz és hány entitást érintenek.

Ez egyszerűen a 1.2 pontban szereplő képletek szisztematikus alkalmazását jelenti minden egyes funkciópont tényezőre, azaz összeadjuk az egyes tranzakciók összes bemeneti, kimeneti elemeit és az érintett entitások számát, majd ezeket a tényezőket megszorozzuk a megfelelő súlyokkal, és az eredményeket ismételten összeadjuk.

Kivitelező részleg / osztály	**Tranzakció azonosító**	**Bemenetek száma**	**Entitások**	**Kimenetek**
Ügyfélszolgálat				
	T1.1	3	3	10
	T1.2	2	2	20
	T1.3	20	3	1
Számvitel				
	T2.1	1	3	20
	T2.2	1	2	2
	T2.3	2	2	1
Értékesítés				
	T3.1	2	2	10
Mindösszesen		31	17	64
FP számítás		31×0,58	17×1,66	64×0,26
		18	28	17

Kivitelező részleg / osztály	Tranzakció azonosító	Bemenetek száma	Entitások	Kimenetek
Korrigálatlan FP mindösszesen				63
Korrigált FP mindösszesen				63× TCA (Technical Complexity Adjustment)

2-16. táblázat: Részletesebb, analitikusabb közelítő érték számítás MKII funkciópont metrikában

A technikai bonyolultsági tényező (TCA) megállapításához a következő képlet alapján juthatunk el: TCA = 0,65 + 0,005 × (a befolyásoló tényezők összegével) . (ld. 2-4, 2-3 táblázatokat). Ökölszabályként alkalmazható a következő: ha a rendszer dominánsan kötegelt feldolgozást végez akkor a TCA = 0,70, ha a rendszer dominánsan interaktív feldolgozást végez, akkor a TCA = 0,90.

2.4 Az adatközpontú megközelítés

Az adatközpontú megközelítés az adatok, az adatszerkezet és elemei között fennálló kapcsolatok oldaláról nézi a rendszert. Ez alapján ad egy becslést a rendszer méretére funkciópontban, azonban ez csak egy közelítő érték, de sokszor alkalmas arra, hogy egy gyors ellenőrzést lehessen végezni akkor, amikor rendelkezésünkre áll egy adatmodell, de ez csak egy durva közelítést ad a funkcionális oldalról végrehajtott becsléshez hasonlóan.

2.4.1 A teljes eljárás

1) Határozzuk meg a modell alkotórészeinek méretét
   a) Az adatszerkezet modellból a következő számokat kell megbecsülni: mennyi entitás van (N_e), hány attribútuma van az entitásoknak(N_a), mennyi az entitások közötti kapcsolatok száma(N_r, „relationship”).
2) A tranzakciók számának megbecslése
   a) Általában feltehető, hogy a rendszer minden egyes entitását egyszer létrehozzák, majd valamikor törlik, illetve az élete során módosítják. Az is valószínű, hogy legalább egyszer lekérdezik az adatait. Ennek alapján feltehetjük, hogy minden entitásra legalább négy tranzakció vonatkozik.
   b) N_t = Tranzakciók száma = $4 \times N_e$
3) Egy tranzakcióban érintett entitások számának megbecslése
   a) Ha az *átlagos konnektivitás* (A_c)alatt a következőt értjük „ egy tetszőleges entitás olvasásakor az érintett entitás átlagosan A_c másikkal van összekötve, áll kapcsolatban”, akkor egy átlagos tranzakcióra azt mondhatjuk, hogy átlagosan a következő számú entitást érinti:
   b) N_{et}=(entitások száma per tranzakció)=$1+A_c$

c) vagyis az adott tranzakcióban főszerepet játszó entitás plusz A_c másik.

d) $A_c = (2 \times N_r)/N_e$ így becsülhető meg (Mivel egy A és B entitás közötti kapcsolat valójában két kapcsolatot ír le).

4) A tranzakciók által használt mezők számának megbecslése

a) Feltételezhetjük, hogy egy tranzakcióban általában van egy alapentitás amelyik a legfontosabb információkat tárolja, és még néhány további, másodlagos jelentőségű entitás. A következő feltételezéssel élünk: a használt mezők számát közelíthetjük az alapentitás mezőinek számával plusz a vele kapcsolatban álló entitások mezői számának a felével. Így tehát:

b) N_{ft} = Mezők száma tranzakciónként (#fields per transactions)

c) $=(N_a/N_e) + \{A_c \times [(N_a/N_e)/2]\} =$

d) $=(N_a/N_e) + \{[(2 \times N_r)/N_e] \times [(N_a/N_e)/2]\} =$

e) $=(N_a/N_e) + (N_r/N_e) \times (N_a/N_e)$

5) Létrehozást és aktualizálást végző tranzakciók méretének megbecslése

a) Az első feltételezés az, hogy egy létrehozó vagy aktualizáló tranzakció esetén a bemenetek jelentősek, gyakorlatilag lényegtelen kimeneti adattal, ezért egy ilyen típusú tranzakció a következőképpen közelíthető

b) $(0{,}58 \times N_{ft}) + (1{,}66 \times N_{et}) + (0{,}26)$

c) A második feltételezés az, hogy tranzakciók fele létrehozó vagy aktualizáló típusú, ezért:

d) SUCT= Az aktualizáló tranzakciók mérete (Size of update and create transactions)

e) $= [(0{,}58 \times N_{ft}) + (1{,}66 \times N_{et}) + (0{,}26)] \times (N_t/2)$

6) A lekérdező tranzakciók méretének megbecslése

a) Az első feltételezés az, hogy egy lekérdező tranzakció fordított tükörképe egy aktualizáló tranzakciónak abban az értelemben, hogy minimális bemeneti eleme van jelentős kimeneti adatelemmel, ezért egy lekérdezés a következőképpen becsülhető meg:

b) $(0{,}58) + (1{,}66 \times N_{et}) + (0{,}26 \times N_{ft})$

c) A második feltételezés az, hogy a tranzakciók egy negyede lekérdezés, és így:

d) SET= Lekérdezések mérete (Size of enquiry transactions)

e) $= [(0{,}58) + (1{,}66 \times N_{et}) + (0{,}26 \times N_{ft})] \times (N_t/4)$

7) A törlő tranzakciók méretének megbecslése

a) Azt tételezzük fel, hogy a törléseknél jelentéktelen be és kimeneti adatok vannak, viszont a feldolgozási rész jelentős. Ezért egy átlagos tranzakció mérete a következő:

b) $(0{,}58) + (1{,}66 \times N_{et}) + (0{,}26)$

c) A második feltételezés az, hogy a tranzakciók egy negyede törlés, és így:

d) SDT= Törlések mérete (Size of delete transactions)

e) $= [(0,58) + (1,66 \times N_{et}) + (0,26] \times (N_t/4)$

8) A rendszer teljes mérete korrigálatlan funkciópontban:

a) NemKorrFP=SUCT+SET+SDT

2.4.2 Egyszerűsített módszer

Az első lépés megegyezik a teljes eljárással, azaz a szükséges adatok begyűjtésével.

A további lépésekben alkalmazott képletek egyszerűsíthetők, illetve a behelyettesítések elvégezhetők N_e, N_a, és N_r –re [lsd. 2.4.1 szakasz, 1)a) pontja] vezetve vissza a számításokat. A következő egyszerűsített kifejezéseket kapjuk:

15. szabály: A fejlesztés funkciópont méretének becslése az adatszerkezet alapján

T_{if}= Az összes bemeneti adatelem (Total input fields)

$= \{2N_a \times [1+N_r / N_e]\}+2 N_e$

T_{ea}=Az összes érintett entitás (Total entities accessed)

$= 4 N_e + 8 N_r$

T_{of}= Az összes kimeneti adatelem (Total output fields)

$=\{N_a \times [1+N_r / N_e]\}+3 N_e$

A korrigálatlan funkciópont = $T_{if} \times 0,58 + T_{ea} \times 1, 66 + T_{of} \times 0,26$

3 SSADM projektek becslése

3.1 SSADM projektek becslése az Mk II funkciópont metrika segítségével[17]

Ahogy azt már az előbbiekben láttuk az Mk II funkciópont méretezési technika egy felül nézetből alkalmazható, felülről lefelé haladó elemzési technika. A rendszer, fejlesztés illetve a projekt funkciópont méretének meghatározása után a többi projekt

[17] "Estimating with Mk II Function Point Analysis", CCTA Information Systems Engineering Library, HMSO 1994,

"Estimating on an SSADM Project", CCTA Information Systems Engineering Library, HMSO 1994,

Az Informatikai Tárcaközi Bizottság ajánlásai, http://www.itb.hu/ajanlasok (2001.01.26.), CCTA kiadványok, http://www.ccta.gov.uk/bestpractice/index.htm (2001.01.31.)

paraméter is becsülhető lesz, például a fejlesztésre fordítandó idő és a fejlesztési költségek.

Az Mk II funkciópont méretezés az információfeldolgozási logika méretének meghatározására irányul, majd ennek korrigálását a műszaki bonyolultság figyelembevételével hajtja végre. (ld. 2.1, 2.3.2, 2.4.2 pontokat). SSADM értelemben egy logikai tranzakció az, ami egy eseményt vagy egy lekérdezési kérelmet dolgoz fel. A műszaki vagy technikai bonyolultsági tényező (ld. 2-4 táblázat) megállapítását 19 vagy még több műszaki sajátosság alapján végzik.

A korrigált rendszer méret funkciópontban segít a normatív becslés előállításában, a teljes ráfordítandó idő és a teljes munka ráfordítás közelítő értékének meghatározásában. A munka és idő ráfordítást szét kell osztani a fejlesztési szakaszok között, majd a projektfüggő különleges feltételeket is számításba kell venni, mint például a kényszerítő határidőket, erőforrás korlátokat, stb.

Az emberi erőforrásoknál a teljes munkaidőben dolgozókkal számolnak, és ezeket kerekítik fel teljes egységekre, majd a rendelkezésre álló alkalmazottakhoz illesztik.

A becslés során nagyon sokszor kell alkalmazni ipari norma vagy súlytényezőket. Egy adott szervezetnél ahol először használják az Mk II funkciópont elemzést, célszerű az ipari normákkal, a *de-facto* szabvány értékekkel élni. Ha azonban egyre több és több tipikus projekt fejeződik be, ezekről érdemes az adatokat gyűjteni, és kalibrálni a projektbecslési eljárást az adott emberi, műszaki és egyéb tényezőkhöz igazodva.

3.1.1 A funkciópont elemzéshez szükséges dokumentáció összegyűjtése

A minimálisan szükséges, a funkciópont elemzésben felhasználandó dokumentációk a következők:

- logikai adatmodell;
- funkció-katalógus, lista, amely tartalmazza a lekérdező és aktualizáló funkciók leírását és besorolásukat a szerint, hogy interaktív (on-line) vagy kötegelt (batch) feldolgozást végeznek-e;
- az igényelt rendszer informatikai, technikai, műszaki jellemzőinek durva felmérése, vagyis például magas tranzakció számra lehet-e számítani, válaszidő igény, stb.;
- a megvalósítás várható informatikai, műszaki eszközeinek besorolása ilyen lehet például, 3GL vagy 4GL fejlesztői környezet, stb.
- szervezeti szintű környezeti útmutató (installation style guide), ami azokat az általános műszaki, technikai, informatikai szabványokat, előírásokat, szabályozásokat tartalmazza, amelyeket az adott szervezetnél minden rendszerfejlesztésnél be kell tartani.

Az első három dokumentáció rendelkezésre áll a „Követelmény specifikációs „ szakasztól (3. szakasz az SSADM-ben) kezdve, az utolsó kettő a maga teljességében csak „A technikai / műszaki / informatikai alternatívák” kidolgozása után áll rendelkezésre. Ebből következik, hogy kielégítő pontosságú becslés csak a követelmény specifikációs szakasz befejezése után, illetve a műszaki alternatívák kidolgozása végén végezhető el.

Ahogy azt a projektirányítási módszertanok is tanítják, a projekttervezés, újra becslés folyamatos tevékenység, amely rugalmasan reagál az egyre pontosabb információkhoz való hozzájutásra. A „Megvalósíthatósági tanulmány” elkészítése után már készíthető valamilyen becslés a fentebbi dokumentumok alapján. Ez egy kiinduló pont, de ahogy a rendszerelemzés és tervezés előrehalad, a becslést természetesen folyamatosan finomítani kell.

Alábbiakban az SSADM módszertan három moduljának termékeit soroljuk fel, amelyek mindegyike a funkciópont elemzés bemeneteként használható fel:

- Megvalósíthatósági tanulmány;
- Követelmény elemzés;
- Követelmény specifikáció (a rendszerszervezési alternatívák kidolgozása után);
- Követelmény specifikáció (a modul, szakasz végén).

3.1.2 Megvalósíthatósági tanulmány

Megvalósíthatósági tanulmány készítése során lehetséges a rendszer méretének megbecslése az ekkor már rendelkezésre álló SSADM dokumentáció alapján, valamint néhány ökölszabály alkalmazásával, amelyek segítségével a logikai tranzakciók osztályozhatóak.

A megvalósíthatósági tanulmányban a nagyvonalúan kidolgozott rendszerszervezési, műszaki alternatívák mindegyikére kell készíteni egy rendszer méret becslést funkciópontban, mert ez a becslés segít az alternatívák közötti választásban.

A következő SSADM termékeket kell ebben a lépésben vizsgálni:

- Az adatfolyam ábrák közül a kontextus diagrammot, amely egy nagyvonalú áttekintést ad az egész rendszerről;
- A hierarchikus adatfolyam ábrákat, ha ilyenek készültek, a kísérő elemi folyamat leírásokkal. Ebből a jelentősebb funkciókat és eseményeket, amelyeket a funkciók feldolgoznak, és ennek révén a logikai tranzakciókat fel lehet ismerni.
- A logikai adatszerkezet és az entitás leírások (ha elkészültek) átvizsgálása annak érdekében, hogy biztosak legyünk abban, hogy az összes olyan logikai tranzakciót azonosították amelyek a fő, a jelentősebb entitások létrehozását, aktualizálását és esetleges törlését végzik.
- „Követelmény jegyzék (katalógus)” alapján le kell ellenőrizni, hogy az összes aktualizáló logikai tranzakciót az adatfolyam ábrák alapján azonosították, valamint a lekérdezéseket is felismerték-e. (A lekérdezések általában nem jelennek meg az adatfolyam ábrákon, általában csak a követelmény jegyzékben).
- „Nagyvonalú műszaki leírása az igényelt technikai környezetnek”, ami a megvalósíthatósági tanulmány egy fejezete, segít a műszaki, technikai / informatikai jellemzők kiértékelésében.

Noha meglehetősen jó becslés készíthető e dokumentumok alapján, azonban hiányzik a funkciók, a logikai tranzakciók részletes leírása, nincsenek meg a be és a kimeneti

adatelemek, az érintett entitások. Ekkor a logikai tranzakciók már megismert kategorizálásával, osztályba sorolásával lehet ezeket a hiányosságokat áthidalni (ld. 2.3.2. fejezetet), vagyis egy logikai tranzakciót a sajátosságai alapján egyszerű, átlagos, vagy bonyolultnak tekintünk és ebből kiindulva adható egy becslés.

3.1.3 Követelményelemzés

A követelményelemzési modul végére a rendszer funkcionális szolgáltatásait már sokkal nagyobb részletességgel ismerik. Az adatfolyam ábrák és a követelményjegyzék sokkal pontosabb becslésre ad lehetőséget. Az adatfolyam modell tartalmazza a rendszer határát átlépő adatfolyamok adatelemeinek leírását a „B/K leírások" formájában, ezzel lehetővé válik a logikai tranzakciók be és kimeneti adatelemeinek, az érintett entitásoknak a pontosabb megszámlálása.

A következő SSADM termékekről beszélünk:

- A „logikai adatfolyam ábra", különösen „Az elemi folyamat leírások" és a „B/K leírások", amik segítségével a jelenlegi rendszer logikai tranzakcióit, be és kimeneti adatelemeit, érintett entitásait meg lehet határozni.
- A „Logikai adattárak / Entitások " kereszthivatkozásait, összerendelését mutató dokumentum alapján az adatfolyam ábrákon megjelenő logikai tranzakciók által használt entitások azonosíthatók.
- A „Logikai adatmodell", különösen az „Entitás leírások" segítenek a logikai tranzakciók felismerése helyességének ellenőrzésében, és a hivatkozott entitások korrekt azonosításában.
- A „Követelmény katalógust" arra kell használni, hogy a felismert logikai tranzakciók körét ki lehessen terjeszteni úgy, hogy bekerüljenek a lekérdezések, a leendő új rendszer további aktualizálási igényei (ezeket csak a követelményjegyzék tartalmazza a fejlesztés jelenlegi szakaszában).
- A „Megvalósíthatósági tanulmány", illetve az informatikai stratégiai terv műszaki koncepció című fejezetéből lehet a leendő rendszer műszaki, technikai, informatikai jellemzőit kiértékelni.
- A kiválasztott „Rendszerszervezési alternatíva".

3.1.4 Követelményspecifikáció (a rendszerszervezési alternatíva kiválasztása után)

A „Rendszerszervezési alternatíva" kiválasztása után az adatfolyam ábrák és a logikai adatmodell már a leendő, az igényelt új rendszerrel szemben támasztott követelményeket tartalmazzák, lényegesen könnyebbé téve a logikai tranzakciók felismerését.

Ezen a ponton a következő SSADM termékekről van szó:

- A „Igényelt rendszeri adatfolyam ábrája", különösen „Az elemi folyamat leírások" és a „B/K leírások", amelyek segítségével az igényelt rendszer logikai tranzakcióit, be és kimeneti adatelemeit, érintett entitásait lehet meghatározni.

- A „Logikai adattárak / Entitások ” kereszthivatkozásait, összerendelését mutató dokumentum alapján az adatfolyam ábrákon megjelenő logikai tranzakciók által használt entitások azonosíthatók.
- A „Logikai adatmodell”, különösen az „Entitás leírások” segítenek a logikai tranzakciók felismerése helyességének ellenőrzésében, és a hivatkozott entitások korrekt azonosításában.
- A „Követelmény katalógust” arra kell használni, hogy a felismert logikai tranzakciók körét ki lehessen terjeszteni úgy, hogy bekerüljenek a lekérdezések is (ezeket csak a követelményjegyzék tartalmazza a fejlesztés jelenlegi szakaszában).
- A „Megvalósíthatósági tanulmány” illetve az informatikai stratégiai terv műszaki koncepció című fejezetéből lehet a leendő rendszer műszaki, technikai, informatikai jellemzőit kiértékelni.

3.1.5 Követelményspecifikáció (a modul végén)

A követelményspecifikáció modul végén a rendszer adatfeldolgozási eljárásait már meglehetősen pontosan meghatározzák a „Funkcióleírások”, az „Esemény kölcsönhatás diagrammok”, és így a „Lekérdezési útvonal” révén egy pontos rendszerméret meghatározás, funkciópont elemzés válik lehetővé. A lekérdezések és események által érintett, illetve hivatkozott entitásokat pontosan megadják az „Esemény kölcsönhatás diagrammok”, és a „Lekérdezési útvonalak”. A bemeneti és a kimeneti adatelemeket a funkcióleírásokhoz tartozó B/K szerkezetek, struktúrák egyértelműen meghatározzák.

Ezen a ponton a következő SSADM termékekről van szó:

- A „Funkcióleírások”, amelyek segítenek az aktualizáló és lekérdező logikai tranzakció felismerésében.
- A „B/K szerkezetek” a logikai tranzakciók be és kimeneti adatelemeinek meghatározásában segítenek.
- Az „Esemény kölcsönhatás diagrammok” az események által érintett, illetve hivatkozott entitások megállapítását segítik.
- A „Lekérdezési útvonalak” a lekérdezések által érintett entitások megállapítását segítik.
- A „Megvalósíthatósági tanulmány”, illetve az informatikai stratégiai terv műszaki koncepció című fejezetéből lehet a leendő rendszer műszaki, technikai, informatikai jellemzőit kiértékelni.

Az „Entitás elérési mátrix” szintén segíthet az érintett entitások gyors feltárásában de gyakran előfordul, hogy a fejlesztés egy adott pontjától kezdve már nem tartják karban, és így csak az „Esemény kölcsönhatás diagrammok”, illetve a „Lekérdezési útvonalak” tartalmazzák a helyes információkat.

3.1.6 Az SSADM projekt funkciópont méretének megállapítása

A 2-16 táblázatot mint kalkulációs lapot kell kitölteni. Ennek megfelelően mindegyik logikai tranzakcióra meg kell állapítani, hogy mennyi bemeneti, kimeneti adateleme van és hány entitást érint. Az interaktív és kötegelt feldolgozású logikai tranzakciókat célszerű külön-külön számolni és az összegeket külön-külön képezni.

A kapott eredményt a technikai bonyolultsági tényezővel korrigáljuk.

Ha az MKII funkciópont elemzést használtuk, akkor az MK II funkciópont értéket átkonvertálhatjuk IFPUG funkciópontra (ld. 15 lábjegyzet), az így kapott funkciópont értékre alkalmazhatjuk 1-11. terjedő szabályainkat.

Funkciópont méret	Átszámítás (M= Mk II. , I = IFPUG)
200-400	$M \sim I$
1500 MK II –ig 2500 IFPUG -ig	$M = 0{,}9 \times I + 0{,}0005 \times I^2$ $I = 1000 \times (\sqrt{(0{,}81 + 0{,}002 \times M)} - 0{,}9)$
az előző fölött (2000+ IFPUG)	M = 0,16 × I × (érintett entitások száma/ összes entitás száma)

3-1. táblázat: IFPUG MK II konverziós tábla

Egyébként pedig az MK II projekt becslési eljárását használhatjuk.

3.1.7 SSADM projekt becslése MK II funkciópont alapján

Lépés	**Projektjellemző**	**Érték**
1.	Becsült termelékenység	0
2.	Munkaóra ráfordítás	0
3.	Funkciópont teljesítés (egységnyi idő alatt)	0
4.	Projekt időtartama	0

3-2. táblázat: SSADM projektjellemzők megbecslése a rendszer funkciópont mérete alapján

Ahhoz, hogy ezeket az értékeket megkapjuk, ismerni kell az ipari átlagokat, amihez ismét Charles Symons statisztikai adatait lehet felhasználni.[18]

Ez a statisztika, illetve grafikon a 3GL környezetben előállított (14. ábra) rendszerekre vonatkozik, a modernebb fejlesztő környezetek esetén — mint például a 4GL környezetek — a termelékenységi adatokat 1,5-el meg kell szorozni.

[18] Stephen Treble, Neil Douglas, „Sizing and Estimating Software in Practice, making MK II Function Points work”, McGraw-Hill Book Company, London,1995

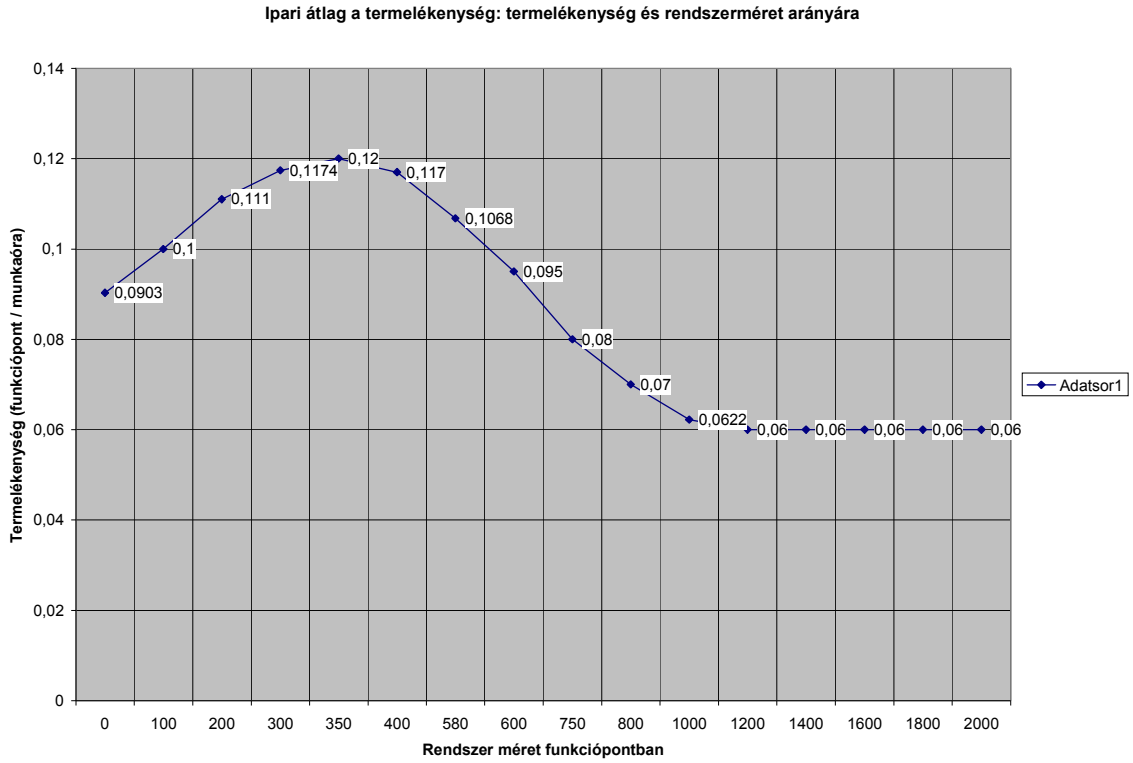
Ha tehát van egy 325 funkciópont méretű rendszerünk, akkor a grafikon alapján a következőt kapjuk: munkaóra ráfordítási igény = 325/ 0,12= 2708 munkaóra.

A funkciópont teljesítés, leszállítási ráta szintén ipari átlag alapján számolható.

Funkciópont teljesítés = 0,45 × √(rendszer méret funkciópontban); [négyzetgyök].

Funkciópont teljesítés = 0,45 × √325 = 8,11 funkciópont per munkahét;

Ebből pedig a projekt időtartama: felhasznált idő = méret / funkciópont teljesítés = 325 / 8.11 = 40 hét.



14. ábra. Az ipari átlag a termelékenység és a rendszer funkciópontban mért mérete között (MK II)

3.1.8 A ráfordítások és a projekt időtartam felosztása a fejlesztési szakaszok között

Miután az SSADM projekt alapjellemzői rendelkezésre állnak, a ráfordításokat és az időket fel kell osztani a projekt szakaszok között.

Szakasz	**Munkaráfordítás (%)**	**Időtartam (%)**
Követelményelemzés	8	13
Követelményspecifikáció	14	22

Szakasz	Munkaráfordítás (%)	Időtartam (%)
Logikai rendszer specifikáció	10	10
Fizikai tervezés	5	5
Kódolás és program modulok, rutinok tesztje	46	25
Rendszer teszt	12	15
Üzembe helyezés	5	10
Összesen	100	100

3-3. táblázat: Ipari átlagok az SSADM fejlesztési szakaszok közti arányokra

Ennek alapján a mintapélda (325 funkciópont) fejlesztésre a következő számokat adja:

Szakasz	Munkaráfordítás (munkaóra)	Időtartam (hét)
Követelményelemzés	217	5
Követelményspecifikáció	379	9
Logikai rendszer specifikáció	271	4
Fizikai tervezés	135	2
Kódolás és program modulok, rutinok tesztje	1246	10
Rendszer teszt	325	6
Üzembe helyezés	135	4
Összesen	2708	40

3-4. táblázat: A mintapélda fejlesztésre az SSADM fejlesztési szakaszok ráfordításai és időtartamai az ipari átlag alapján számolva

Más fejlesztési módszerek alkalmazásánál az általánosan használt és elfogadott rendszertervezési szakaszolás ipari átlag arányai:

Szakasz	Munkaráfordítás (%)	Időtartam (%)
Elemzés	22	35
Tervezés	15	15
Kódolás	46	25
Teszt	12	15
Üzembe helyezés	5	10
Összesen	100	100

3-5. táblázat: Ipari átlagok az (általános) fejlesztési szakaszok közti arányokra

3.1.9 A projekt előnyös és hátrányos tulajdonságainak figyelembe vétele

Szakasz	**Munkaráfordítás (munkaóra)**	**Időtartam (hét)**
Pozitív korrekciós tényező %-ban		
Negatív korrekciós tényező %-ban		
Követelményelemzés		
Követelményspecifikáció		
Logikai rendszer specifikáció		
Fizikai tervezés		
Kódolás és program modulok, rutinok tesztje		
Rendszer teszt		
Üzembe helyezés		
Összesen		

3-6. táblázat: A fejlesztési szakaszokra ható negatív és pozitív tényezők figyelembe vétele

Negatív, illetve pozitív korrekciós tényezőkkel, százalékos értékekkel lehet, kell korrigálni a ráfordítási és az időtartam adatokat. Pozitív tényező lehet például egy alaposan dokumentált rendszer újra írása, ez a munkaráfordításra és az időtartamra is hatással van, ugyanakkor százalékos módosulást nem feltétlenül okoz. Ezért ezeket a százalékokat meg kell különböztetni a táblázatban is.

Negatív tényező lehet például az, hogy a felhasználók nem igazán elkötelezettek a projekt irányában, ezért a ráfordításokat és időtartamokat alkalmasan meg kell növelni. Tipikus probléma ilyenkor, hogy a formális és informális minőségellenőrzési ciklusok a vártnál sokkal lassabbak.

Szakasz	**Munkaráfordítás (munkaóra)**	**Időtartam (hét)**
Technikai, műszaki innovációs korrekciós tényező %-ban		
Követelményelemzés		
Követelményspecifikáció		
Logikai rendszer specifikáció		

Szakasz	Munkaráfordítás (munkaóra)	Időtartam (hét)
Fizikai tervezés		
Kódolás és program modulok, rutinok tesztje		
Rendszer teszt		
Üzembe helyezés		
Összesen		

3-7. táblázat: A fejlesztési szakaszokra ható műszaki, technológiai innovációs tényező figyelembe vétele

A projektbecslésnél be kell számítani az esetleg új módszerek, technikák, technológiák alkalmazását, ami azt jelenti, hogy a ráfordítás és időtartam értékeket egy alkalmas tényezővel meg kell növelni, hasonlóan az előzőekhez szakaszonként, az egyes projektjellemzőkre százalékban megadott korrekciós tényezőkkel módosítani kell. Ezek alapján lehet a projektbecslést dokumentálni.

3.1.10 A határidők és az emberi erőforrás korlátok figyelembe vétele

A fejlesztésre rendelkezésre álló idő hetekben	
Időtartam rövidítési tényező (Schedule Compression Factor, SCF)	

Ezt az értéket a következőképpen kapjuk:

SCF = a rendelkezésre álló idő / becsült projekt időtartammal

Ez a tényező nagyon jól érzékelteti a határidőkkel összefüggő projekt kockázati tényezőt. Ha ez a tényező 0,85 alá esik, akkor a projekt sikertelenségének esélye megnő.

Ha ez az érték 0,5 alá esik, akkor célszerű különböző projekt kockázat csökkentő módszereket alkalmazni, mint például a projekt felbontása kisebb projektekre, a feladat csökkentése, stb.

A rövidítési tényező segítségével újra kell kalkulálni, mindegyik projektszakaszra vonatkozó becslést újra kell számolni:

új munkaráfordítás = régi munkaráfordítás / SCF;

projekt időtartam = a rendelkezésre álló idővel.

új időtartam = régi időtartam × SCF.

Tegyük fel, hogy a mintapéldában a fejlesztésre rendelkezésre álló idő 38 hét, ekkor 38/40= 0,95 az SCF tényező. Ennek alapján újra számoljuk a szakaszokra fordítandó időt és munkát.

Szakasz	Munkaráfordítás (munkaóra)	Időtartam (hét)
Követelményelemzés	228	5
Követelményspecifikáció	399	8
Logikai rendszer specifikáció	285	4
Fizikai tervezés	143	2
Kódolás és program modulok, rutinok tesztje	1311	10
Rendszer teszt	342	6
Üzembe helyezés	143	4
Összesen	2851	38

3-8. táblázat: A mintapélda rövidítési tényezővel korrigált értékei

Miután a munkaráfordítási igényt szakaszonként meghatároztuk, lehetőség van az ipari átlagra támaszkodva a munkaerő igény meghatározására.

Szakaszonként a munkaerőigény a következőképpen számolható:

munkaerő = konstans × munkaóra ráfordítás / időtartam (hetekben)

A konstans a heti munkaórák számából és az alkalmazottak termelékenységéből számolható. Ez a tényező természetesen cégenként, szervezetenként különböző. Nem szabad megfeledkezni a szabadságokról, tanfolyamokról, betegszabadságról és a táppénzes időszakokról sem. Az ipari átlagnál 35 órás munkahéttel és 65%-os termelékenységgel, munkaidő kihasználtsággal számolnak. Ebből következik:

$1 / (0{,}65 \times 35) = 0{,}044$

Szakasz	Munkaerő (főben) Alkalmazottak száma
Követelményelemzés	2,03
Követelményspecifikáció	2,10
Logikai rendszer specifikáció	3,30
Fizikai tervezés	3,30
Kódolás és program modulok, rutinok tesztje	6,06
Rendszer teszt	2,64
Üzembe helyezés	1,65

3-9. táblázat: A mintapélda munkaerő igényének meghatározása szakaszonként

Miután a munkaerő igényt sikerült meghatározni az egyes szakaszokra vonatkozóan, értelmes egységekre kell kerekíteni a kapott értékeket, tekintettel a rendelkezésre álló szakember gárdára.

A kerekítések, az igények, a rendelkezésre álló erőforrások összeillesztése kétféle módon is hat az eddigi eredményekre. Egyrészt az érintett szakasz hossza megváltozik, mivel a szakasz kivitelezésében érintett személyek száma megváltozott; például a 3,3 érték lefelé kerekítésével 3-ra, a szakasz időtartama növekedni fog. Másrészt a személyek számának változása a szervezési, vezetési erőfeszítések iránti igényt módosítja, a fejlesztő csoporton belüli kommunikációs igény növekedhet, illetve csökkenhet, stb.

Ha a határidő merev, nincs sok értelme a lefelé kerekítésnek, mert ennek következménye a határidő tarthatatlansága.

Amikor az alkalmazottak számának módosítása történik, akkor két eljárást kell alkalmazni mindegyik szakaszra. Az első lépés a szakaszra vonatkozó kerekítési tényező meghatározása (Rounding adjustment factor, RAF):

RAF = √(kerekített fő)/ √(kerekítetlen fő); ahol √ a négyzetgyök vonást jelöli

Ennek alapján a következőképpen számíthatjuk a munka- és időigényt:

új munkaráfordítás = régi munkaráfordítás × RAF;

új időtartam = régi időtartam / RAF.

Az alkalmazottak számát felfelé kerekítjük a legközelebbi fél vagy egész értékre, majd az előbbi képletek felhasználásával újra számítjuk a projektjellemző értékeket.

Szakasz	**Munkaerő (főben)** **Alkalmazottak száma**	**Munkaráfordítás (munkaóra)**	**Időtartam (hét)**
Követelményelemzés	2,50	253	4,5
Követelményspecifikáció	2,50	436	7,7
Logikai rendszer specifikáció	3,50	294	3,7
Fizikai tervezés	3,50	147	1,8
Kódolás és program modulok, rutinok tesztje	6,50	1358	9,2
Rendszer teszt	3,00	365	5,4
Üzembe helyezés	2,00	157	3,5
Összesen		3009	36

3-10. táblázat: A mintapélda munkaerő igényének meghatározása a kerekítések után szakaszonként

Ezen a mintapéldán megfigyelhetjük demonstrációs szándékkal a termelékenység és a funkciópont teljesítés alakulását a projekt tervezés során végrehajtott korrekciós lépések végén.

termelékenység = 325/3009 = 0,11;

funkciópont teljesítés = 325/36 = 9,11.

Ez a normatív grafikonunkkal összevetve (ld. 14. ábra) azt mutatja, hogy az ipari norma 0,12, illetve 8,11 megfelelően; ezeket használtuk kiinduló adatként a projekttervezésnél. Ebből látható, hogy a meghozott döntések (kerekítések) a termelékenységet rontották a funkciópont teljesítés érdekében, vagyis ha ez a projekt így teljesül, akkor az átlagnál gyorsabban történik meg a kivitelezése, de a termelékenysége átlagon aluli lesz.

4 Szerződéses technikák a viták minimalizálása érdekében

4.1 A szoftverfejlesztési szerződések konfliktusainak eredete

A szoftverfejlesztés már évtizedek óta nagyon nehéz technológiának bizonyul. Minden más gyártási folyamathoz képest a szoftver előállítás lényegesen több közvetlen — méghozzá elég magasan kvalifikált — emberi munkát igényel. Az alkalmazási rendszereket gyakran olyan területekre tervezik, ahol maguknak a manuális folyamatoknak a megértése sem kielégítő még azok részéről sem, akik esetleg azt nap mint nap végrehajtják. Ezért a szoftverfejlesztés során folyamatosan újabb és újabb követelmények és igények jelennek meg.

A szoftverfejlesztésre vonatkozó szerződések meglehetősen pontatlanok vagy kétértelműek a leszállítandó termékek meghatározásában, a fejlesztés ütemezésében és egyéb mennyiségi paraméterek pontos előírásában. Gyakran a szerződés csak nagy általánosságokat tartalmaz, a helyzetet nem elemzi alaposan, és legfeljebb a fejlesztő személyzet létszámát írja elő. A fejlesztés során bekövetkező követelményváltozásokra általában a szerződés sem a nyelvezetében, sem cikkelyeiben nincs felkészítve.

Annak ellenére, hogy eredményes minőségi ellenőrzésre technikailag lehetőség van[19], az erre vonatkozó előírások ritkán kerülnek be a szerződésbe, és az ezekkel kapcsolatos problémákról nem is vesznek tudomást addig, amíg a megrendelő kézhez nem kapja a terméket és csalódottan tapasztalja az általa nyújtott funkcionalitás,

[19] ISO 9000, ISO 9000-3, illetve az ISO 2000 előírásai a szolgáltatásra (műszaki tervezésre), szoftver termékre vonatkozóan. Továbbá ISO 9126 a szoftver minőségi tulajdonságaira, ISO 6592 a fejlesztés dokumentáltságának minimumára. A Carnegie Melon egyetemen kidolgozott „Capability Maturity Model® (SW-CMM®) for Software", http://www.sei.cmu.edu/cmm/cmm.html (2001.02.07.), „Total Quality Management", http://mijuno.larc.nasa.gov/dfc/tqm.html , http://www.mcb.co.uk/tqm.htm , (2001.02.07.)

illetve szolgáltatások gyenge minőségi színvonalát. Ezért a minőségre vonatkozó előírásokat célszerű magában a szerződésben rögzíteni.

Az alábbi táblázat (4-1.) mutatja a statisztikai gyakoriságát a projektek sikerességének a projektek funkciópont méretével összefüggésben.

	Projekt méretek funkciópontban (IFPUG)			
Projektek kimenetele	**<100**	**100-1000**	**1000-5000**	**>5000**
Törölt	3%	7%	13%	24%
12 hónapnál nagyobb csúszás	1%	10%	12%	18%
6 hónapnál nagyobb csúszás	9%	24%	35%	37%
Nagyjából kész határidőre	72%	53%	37%	20%
Határidőnél korábban befejeződik	15%	6%	3%	1%

4-1. táblázat: A projektek sikeressége a funkciópont méret függvényében (IFPUG)

Ahogy ebből a statisztikából is látható, a nagy méretű projektek csalódást keltően sikertelenek, csúsznak a határidőkkel az eredeti várakozásokhoz, becslésekhez képest.

	Projekt méretek funkciópontban (IFPUG)			
Projektek átlagos időtartama	**<100**	**100-1000**	**1000-5000**	**>5000**
Tervezett idő	6	12	18	24
Tényleges idő	6	16	24	36
Különbség	0	4	6	12

4-2. táblázat: A projektek tervezett és tényleges időtartamainak átlaga funkciópont méretek szerint

A fentebbi táblázat (4-2) mutatja az átlagos fejlesztési időket a követelményelemzéstől kezdve a rendszer telepítéséig, üzembe helyezéséig két dimenzióban: a tervezett projekt időtartam és a ténylegesen teljesített projekt időtartam tekintetében.

A táblázatból látható, hogy meglehetősen nagy a különbség a tervezett, a várt fejlesztési idők és a ténylegesen megvalósuló teljesítések között a nagy alkalmazási

rendszerek esetén. A különbség egy része betudható a nem korrekt becslésnek, másik része pedig a lopakodó felhasználói követelményekből származó feladat bővülésnek, amely a kezdeti, kiinduló becslések és a szerződéses megállapodás után került napvilágra.

A követelmények változásának, növekedésének oka az lehet, hogy a szoftver alkalmazások fejlesztése kiterjeszti a szervezet működési lehetőségeinek körét. Ez egy kicsit arra a helyzetre emlékeztet, amikor valaki olyan ködben mászik hegyet, ami közben fokozatosan egyre magasabbra emelkedik. Először csak a közvetlen környék látszik, majd, ahogy a köd egyre magasabbra emelkedik egyre több és több látszik a környező tájból.

A funkciópont metrika, az elemzés bizonyítottan megfelelő eszköz a lopakodó felhasználói követelmények kezelésére, a belőlük származó következmények kézben tartására a költségek és egyéb projekt paraméterek tekintetében. (ld. még a 1.4.2.4 szakaszt).

A lopakodó felhasználói követelmények kézben tartásához az szükséges, hogy az alkalmazási rendszer, illetve a projekt funkciópont méretét állapítsák meg és rögzítsék amint lehet. Erre először a követelmények tisztázása és első „véglegesítése", befagyasztása után van lehetőség. A fejlesztési szakasz végén lehetséges a rendszer ismételt megmérésére és a különbség, illetve a növekedési ráta ekkor határozható meg. Ha a rendszer eredeti mérete például 100 funkciópont volt, a leszállításkor pedig 125, akkor ez mutatja a követelmények változásából származó növekményt.

Ezt a változást statisztikailag elemezni lehet, közelítőleg megmérhető a követelmények változásának havi átlaga. A táblázat (4-3) mutatja a követelmények változását a rendszer funkciópont méretének százalékában a tervezési, követelményspecifikációs szakasz végétől a fejlesztési szakasz végéig. Természetesen ezeknek az adatoknak nagy a szórása, magas a mérés hibaszázaléka, de a követelmények változási rátájának mérése még így is nagyon hasznos.

Szoftver / alkalmazási rendszer típusa	**Havi követelményváltozás**
Szerződés vagy szolgáltatás kihelyezés keretében készített szoftver	1,0%
Információrendszer szoftver	1,5%
Rendszer szoftver	2,0%
Katonai alkalmazások	2,0%
Kereskedelmi szoftverek	3,5%

4-3. táblázat: A lopakodó felhasználói követelmények átlagos havi növekménye

Érdemes a táblázatban azt is megfigyelni, hogy bár a szerződéses formában készülő szoftvereknél a követelményváltozás rátája viszonylag kicsi, az ezek körül kialakult viták a felek között nézeteltérésekhez vezetnek, amik rendezése esetleg jogi formát is ölthet (per).

4.1.1 Hogyan lehet a szoftverfejlesztésből származó kockázatokat és jogvitákat minimalizálni?

A szoftverfejlesztésből származó kockázatokat és jogvitákat úgy lehet minimalizálni és a bíróságot elkerülni, ha a problémát a gyökerénél kezeljük:

1) A szerződéses tárgyalások során és a szerződésben rögzíteni kell a leszállítandó szoftver termékek méretét;
2) A költség- és időtartam becslésnek formálisnak, hivatalosnak és teljesnek kell lenni;
3) A lopakodó felhasználói követelmények kezelését a szerződésben mindkét szerződő fél számára kielégítően kell rendezni;
4) Független szakértők, tanácsadók bevonásának módját a projekt szakaszok kiértékelésénél szabályozni kell;
5) A minőséggel kapcsolatos kritériumokat, azok elfogadható szintjét a szerződésben rögzíteni kell;
6) A szoftver gyártónak, szállítónak eredményes minőségellenőrzési lépéseket kell alkalmazni a szoftver minőségének biztosítására.

Szerencsére ezeket a kérdéseket megfelelő módon lehet kezelni.

4.1.1.1 A leszállítandó szoftver termékek méretezése

Senki nem vásárol vagy építtet házat úgy, hogy nem tudja hány négyzetméteres lesz. Sem a megrendelő, sem a szállító ne írjon alá szerződést szoftverfejlesztésre anélkül, hogy a munkát hivatalosan és formálisan fel ne mérték volna. Az ebből származó kockázatok minimalizálásnak legjobb módja, ha a szerződést előkészítő tárgyalások során funkciópontban határozzák meg a rendszer méretét.

A funkciópont elemzést, metrikát lehet a legeredményesebben használni a szoftverfejlesztéssel összefüggő összes termék méretezésére: specifikáció, felhasználói kézikönyv, forráskód, teszt esetek. (ld. 1.4.2.3, 1.4.2.5 szakaszokat).

A rendszer méretét funkciópontban kielégítő pontossággal a követelményelemzési, követelményspecifikációs szakasz után lehet meghatározni. Ha a szerződés a követelmény meghatározást is tartalmazza feladatként, a helyzet egyértelmű tisztázása miatt az egységnyi funkciópont árát előre kell rögzíteni.

4.1.1.2 Formális szoftver költség becslés

Ekkor alkalmazhatók a kifejezetten erre a célra kifejlesztett funkciópont elemzésen alapuló szoftverek, melyekből az adatok átvihetők a különböző projekttervező eszközökbe. A korábban ismertetett eljárások természetesen alkalmazhatók, támogatva pl. MS-EXCEL-el. (ld. még 1.4.2, 2.1.1, 3,).

Az utóbbi időben nagy teret kapott a tevékenység alapú kontrolling a gazdasági élet sok területén. Magyarországon ezekre a tevékenységekre vonatkozó nyilvános adatok nem találhatók. Az USA-ból származó példában egy emberhónap költsége 5000 US $ és 100%-os közteherrel számolnak:

				Közterhek nélküli	Közterhekkel együtt (minden rezsi költséget beleértve)
		FP / hónap	**óra /FP**	**Költség/FP (US$)**	**Költség/FP (US$)**
	Tevékenység				
1.	Követelmény	175	0,75	28,57	57,14
2.	Prototípus	150	0,88	33,33	66,66
3.	Architektúra	300	0,44	16,67	33,34
4.	Projekttervezés	500	0,26	10	20
5.	Nagyvonalú terv	175	0,75	28,57	57,14
6.	Részletes terv	150	0,88	33,33	66,66
7.	Terv bevizsgálása	225	0,59	22,39	44,78
8.	Kódolás	50	2,64	100	200
9.	Újrafelhasználható elemek beszerzése	600	0,22	8,33	16,66
10.	Csomag vásárlás	400	0,33	12,5	25
11.	Program kód bevizsgálás	150	0,88	33,33	66,66
12.	Verifikáció és validálás	125	1,06	40	80
13.	Konfiguráció kezelés	1750	0,08	2,86	5,72
14.	Formális integráció	250	0,53	20	40
15.	Felhasználói dokumentáció	70	1,89	71,43	142,86
16.	Modul / rutin teszt	150	0,88	33,33	66,66
17.	Funkció teszt	150	0,88	33,33	66,66
18.	Integráció teszt	175	0,75	28,57	57,14
19.	Rendszer teszt	200	0,66	25	50
20.	Telephelyi teszt	225	0,59	22,22	44,44
21.	Átadás / átvételi teszt (Felhasználók elfogadási tesztje)	350	0,38	14,29	28,58
22.	Független tesztelés	200	0,66	25	50
23.	Minőségbiztosítás	150	0,88	33,33	66,66

24.	Üzembe helyezés / Oktatás	350	0,38	14,29	28,58
25.	Projektvezetés	100	1,32	50	100

4-4. táblázat: Tevékenység alapú költség / funkciópontra példa (IFPUG)

Ahhoz, hogy a 4-4. táblázat használható legyen, a rendszer méretét funkciópontban ismerni kell. Ezután ki kell választani azokat a tevékenységeket, amelyeket az adott alkalmazás, fejlesztés esetén el kell végezni, majd a munkaóra / funkciópont értékeket tevékenységenként összegezni kell, ez megismételhető a költségtényezőkre is. Az 5000$-os emberhónap és 100% közteher értékek értelemszerűen helyettesítendők a megfelelő kiindulási értékekkel. Természetesen ezt az eljárást karbantartási és egyéb projektekre megfelelően át kell alakítani.

4.1.1.3 A lopakodó felhasználói követelmények kezelése

A szoftver projektek, a követelmények a fejlesztés során 90%-ban megváltoznak, ezért a lopakodó felhasználói követelmények a szoftver ipar egyik legnagyobb gondját jelentik. Különböző eljárásokat fejlesztettek ki a probléma kezelésére, néhányat az 1.4.2.4 pontban már felsoroltunk, itt röviden még szót ejtünk róluk.

4.1.1.3.1 Közös alkalmazásfejlesztés

A közös alkalmazásfejlesztés (Joint Application Development, JAD) egy olyan rendszerfejlesztési eljárás, amelyben a felhasználók képviselői és a fejlesztők képviselői együtt dolgoznak egy közös követelményspecifikáció előállításán. Az érdeklődők számára könyvek, tanfolyamok és tanácsadó-szervezetek állnak rendelkezésre.

Az eljárás a régebbi stílusú „ellenséges” hangulatú szoftverfejlesztéshez képest akár 50%-al is csökkentheti a lopakodó felhasználói követelményeket. Ez a megközelítés különösen azoknál a nagy alkalmazási rendszereknél kitűnő választás, amelyek információrendszerek automatizálására készülnek.

4.1.1.3.2 Prototípusfejlesztés

Nagyon sok felhasználói kívánság csak akkor jelenik meg, amikor a felhasználó a képernyőket és a jelentéseket először meglátja, ezért nyilvánvaló, hogy a fejlesztés korai szakaszában készített prototípus bemutatása előre hozhatja a változtatási igényeket, így azok nem maradnak a fejlesztési szakasz végére.

A prototípus alapú megközelítés gyakran eredményes a lopakodó felhasználói követelmények csökkentésében, összekombinálható más módszerekkel is mint például a közös alkalmazásfejlesztési eljárás. Maga a prototípus megközelítés 10% és 25% között képes csökkenteni a lopakodó felhasználói követelményeket.

4.1.1.3.3 Változáskezelési fórum

Változáskezelési fórum (Change Control Board) nem technológia, hanem egy vezetési, szervezési eljárás. A fórum a vezetők egy csoportjából, a szállító képviselőiből és informatikai, műszaki szakemberekből áll, akik eldöntik, hogy egy benyújtott felhasználói változtatási igényt elfogadnak-e vagy elutasítanak.

4.1.1.3.4 Mozgó költség skála az egységnyi funkciópont árára

A 1.4.2.4 pontban már találkoztunk ezzel a táblázattal, itt most USA ipari átlagokra támaszkodva megismételjük:

Kezdetben 1000 funkciópont	500 US $/ FP
A szerződés aláírása után 3 hónappal hozzáadandó új igényekre	600 US $/ FP
A szerződés aláírása után 6 hónappal hozzáadandó új igényekre	700 US $/ FP
A szerződés aláírása után 9 hónappal hozzáadandó új igényekre	900 US $ / FP
A szerződés aláírása után 12 hónappal hozzáadandó új igényekre	1200 US $ / FP
Felhasználói kérésre törölt vagy elhalasztott felhasználói követelmények	150 US $ / FP

4-5. táblázat: Funkciópont ára fejlesztési szakaszonként (példa US$-ban)

Fejlesztési és karbantartási szerződéseknél a funkciópont metrika használatának előnye, hogy a felhasználói követelményekre alapul és nem lehet a szerződő partner részéről egyoldalúan módosítani, ellentétben például a programsorok számára vonatkozó megállapodásokkal.

4.1.1.4 A szoftver szerződés és projekt független értékelése

Nagy rendszerek fejlesztésénél kötelező óvatosságot, kellő körültekintést jelent, ha a fejlesztés bizonyos szakaszaiban független szakértők kapcsolódnak be.

A független szakértők a következő szerepkörökben alkalmazhatók:

1) A szerződés felülvizsgálata abból a szempontból, hogy a tipikus vitákat okozó kérdéseket hogyan rendezték el.
2) Az alkalmazás funkciópont méretének meghatározására, illetve az érték helyességének ellenőrzésére.
3) A költség és idő becslések helyességnek ellenőrzésére.
4) A szoftver minőségbiztosítás módszereinek előírására, illetve a szerződésben előírtak helyességének ellenőrzésére.
5) Az eredetileg tervezettől eltérő projekt, illetve szerződés helyes pályára történő visszaállításának módszertani segítésére.

Ha a független szakértők bevonását a projekt terv, illetve a szerződés a munka elejétől tartalmazza, akkor azt mindkét fél elfogadja.

4.1.1.5 Minőségi követelmények rögzítése a szoftverfejlesztési szerződésekben

A második leggyakoribb vitaforrás az átadott szoftver minősége olyan állítások formájában, hogy gyenge, használhatatlan.

Ezért a rendszer funkciópont mérete alapján célszerű a szerződésben rögzíteni a számszerűsíthető minőségi követelményeket.

Az 1-8. táblázat (1-8. táblázat: A fejlesztés során az átlagos hibaszám (Statisztika az USA-ból)) tartalmazza az USA ipari átlag adatait.

Ezek a számok tartalmazzák azokat a hibákat, amelyek a követelményelemzés korai szakaszától kezdve a szoftver életciklusának végéig felbukkanhatnak. A hibákat a követelményelemzési és specifikációs dokumentumok, a tervek, a programkód szemlézése, bevizsgálása közben fedezik fel, továbbá a különböző tesztelései eljárások során, felhasználók hiba / probléma jelentéseiből derülnek ki.

Az 1.4.2.7 pontban már tárgyaltuk a hiba eltávolítás hatékonyságának fogalmát. Az USA-ban az ipari átlag a hibaeltávolításra 85% körül van, de a legjobb szoftverfejlesztő szervezetek 99%-os hatékonysággal is büszkélkedhetnek.

A szoftverfejlesztési szerződésekben ésszerű egy megfelelő célérték előírása a hiba eltávolítás hatékonyságára. A következő javasolható: az alkalmazás teljes és sikeres telepítése után 6 hónappal a hatékonysági tényező 98% legyen.

Tegyük fel, hogy a fejlesztés során a programozók 90 hibát találtak. A fejlesztő csoportnak, illetve a minőségbiztosító csoportnak a talált hibákról naplót kell vezetni. Amikor a rendszert átadják a megrendelőnek, folytatni kell a hibák nyomon követését a használat első éve alatt. Tegyük fel, hogy a felhasználók az első évben 10 hibát találtak. Megfelelő idő eltelte után (3 hónappal, 6 hónappal, 12 hónappal később) az átadás előtt talált hibákat és az átadás után a megrendelő által észlelt hibákat össze kell adni és ki kell számítani a hiba eltávolítási hatékonyságot.

A fejlesztés során talált hibák száma	90
A felhasználók, illetve a megrendelő által észlelt hibák száma	10
Összes hiba	100
Hibaeltávolítási hatékonyság (átadás előtti hiba / összes hiba)	90%

4.1.1.6 A hibaeltávolítási hatékonyság magas szinjének elérése

Magas minőségi és hibaeltávolítási hatékonyság csak a kor színvonalának megfelelő minőségellenőrzési eljárással érhető el. Nagy alkalmazási rendszerek fejlesztésénél az egyetlen mód a 95%-os hibaeltávolítási hatékonyság elérésére, ha formális tervezési eljárásokat, a programkód formális bevizsgálását, szemlézését és formális tesztelési eljárásokat alkalmaznak. Magas hibaeltávolítási hatékonyság elérése növeli annak valószínűségét, hogy a fejlesztési projekt időben és a tervezett költségeken belül elkészül. A projektek nagy része azért sikertelen, mert az előállított szoftver minősége nem megfelelő. A tesztelés előtti formális bevizsgálási, szemlézési eljárások lerövidítik a fejlesztési időt, csökkentik a költségeket és növelik a felhasználók elégedettségét.

Hiba eltávolítási tevékenység	A hatékonyság értéktartománya	
	-tól (%)	-ig (%)
Informális tervszemle	25	40
Formális terv bevizsgálás	45	65
Informális programkód szemle	20	35
Formális programkód bevizsgálás	45	70
Programegység (modul, rutin) teszt	15	50
Új funkciók tesztje	20	35
Regresszió teszt	15	30
Integráció teszt	25	40
Rendszer teszt	25	55
Kis intenzitású béta teszt (felhasználók száma < 10)	25	40
Nagy intenzitású béta teszt (felhasználók száma >1000)	60	85

4-6. táblázat: A hibaeltávolítási eljárások hatékonyságának értéktartományai

Nyilvánvaló, hogy önmagában egyik hiba eltávolítási eljárás sem megfelelő, a legjobb eredményeket csak a fenti eljárások szinergiájának kihasználásával lehet elérni.

Egy szoftverfejlesztési szerződésben tehát két lehetőség áll rendelkezésre a magas minőségellenőrzési szint eléréséhez:

1) A hiba eltávolítási hatékonyság számszerű értékének meghatározása, a mérés időpontjainak előírása;
2) Az alkalmazandó hiba eltávolítási eljárások egyértelmű és kifejezett előírása.

5 Objektum alapú illetve orientált rendszerfejlesztési környezetek és a funkciópont számítás

A jelentős funkciópont elemzési iskolák továbbfejlesztik a számolási módszereket azért, hogy a korszerűbb technológiák figyelembevételére is alkalmasak legyenek.

Teljes konszenzus még nem alakult ki ezen a téren, de vannak kísérletek amelyek kifejezetten az objektum-orientált környezethez készítnek rendszer, illetve szoftver

méretezési eljárásokat. Mások az IFPUG, illetve az MK II funkciópont metrikát viszik át jó eredménnyel az újabb technológiákat alkalmazó környezetre.

5.1 Felhasználói objektum pontok[20]

A funkciópont számítás körében a felhasználói felülettel kapcsolatos méretezést, a felhasználói objektumokat — használati eseteknek (use case) nevezett, a felhasználói felülethez tartozó rendszerelemeket — is figyelembe veszik.

Egy rendszer aktornak a következőket tekintik:

- egy egyszerű aktor: egy másik rendszert képvisel, meghatározott API-val (Application Programing Interface);
- egy átlagos aktor: vagy egy másik rendszer amellyel a kapcsolatot valamilyen protokollon keresztül tartják (pl. TCP/IP), vagy szöveges alapú, régi, hagyományos ASCII terminál;
- összetett aktor: az, amely grafikus felületet használ (GUI) vagy WEB lapokat.

Szokásos módon az összes szóba jövő elemet le kell számolni — melyik típusból mennyi van — majd a megfelelő súlytényezőkkel megszorozni és a típusonként kapott eredményeket összegezni.

Aktor típusa	**Leírása**	**Súlytényező**
Egyszerű	Program felület, kapcsolat (interface)	1
Átlagos	Interaktívan vezérelt, vagy protokoll alapú felület, kapcsolat	2
Összetett	GUI, WEB	3

5-1. táblázat: Az aktorok súlytényezői

Mindegyik felhasználói felület objektumra, illetve használati eset leírásra meg kell határozni azt is, hogy egyszerű (3-4 tranzakcióból áll), átlagos (4-7 tranzakcióból áll), összetett (8-nál több tranzakcióból áll). A típusok leszámolása és a súlytényezőkkel történő megszorzása után az eredményeket összeadjuk.

Felhasználói objektum típusa (Használati eset)	**Leírása**	**Súlytényező**
Egyszerű	tranzakciók <= 3	5
Átlagos	4-7 tranzakció	10
Összetett	tranzakciók > 7	3

5-2. táblázat: A felhasználói objektum súlytényezői

[20] http://www.sunworld.com/swol-12-1999/swol-12-itarchitect.html (2001.02.07.)

A két fentebbi eljárás eredményének összeadása után kapjuk a korrigálatlan felhasználói objektum (használati eset) pontokat, majd ezeket a technikai, műszaki bonyolultsági tényezőkkel korrigáljuk.

5.2 Funkciópont és objektum orientált rendszerek

A funkciópont elemzés és az objektum orientált rendszerek terminológiája, szakkifejezései nem kerültek még elég közel egymáshoz, a két terület fogalmait még nem sikerült kielégítően egymásra leképezni. Durva ökölszabályként azonban a következő működik: egy olyan objektum osztály, amely megfelel egy üzleti vagy szervezeti fogalomnak, közelítőleg megfelel az IFPUG (belső) logikai állományának. A tervezési és megvalósítási okokból létrejött objektum osztályokat nem kell figyelembe venni. Ha az objektumokat adatbázisban vagy adatállományokban tárolják, akkor ezeket az adatállományokat be kell számítani. Ez az objektumok és a relációs adatbázis megközelítés közti leképezés figyelembe vételének legegyszerűbb módja.

6 Példák, esetek

6.1 A 'SCUD'[21] megoldás bevált Ausztráliában

A Megrendelő kibocsát egy tender felhívást (ITT, Invitation to Tender), amely azonban csak a nagyvonalú követelményeket tartalmazza. A tender felhívás azonban arra kéri a pályázókat, hogy 1 funkciópontra rögzített árat ajánljanak meg. A tender nyertes kiválasztásánál az egyéb szokásos szempontok mellett ez a $/Fp tényező is komoly súllyal esik latba.

Amint az első részletesebb követelményelemzés lezajlik, kialakítják a kiinduló funkciópont bázist, amit összekapcsolnak a funkciópont egységárral. Ezek után dönthet a Megrendelő, hogy ez a munka elfogadható, pénzügyileg megengedhető. A projekt határait és terjedelmét ennek alapján pontosítják, majd a fejlesztés folytatódhat. A végső árat az átadott és átvett rendszer funkciópontja, illetve az egységár alapján állapítják meg. A tisztességes eljárás érdekében a fejlesztés egyes szakaszaiban mindkét fél által független, elfogadható szakértő céget kérnek fel a funkciópont megállapítására.

Ennek az eljárásnak az előnyei:

A tender kiírással kapcsolatos időt, erőfeszítést, ráfordítást csökkenti. A Megrendelő végig biztos lehet abban, hogy a pénzéért jó szolgáltatást kap (összevethető a funkciópont egységár az ipari normákkal). A Megrendelő kézben tudja tartani a teljes fejlesztés költség oldalát a követelmények szigorú meghatározásával, feszes ellenőrzésével, így a projekt fejlesztési kockázatok nagy része átkerül a Megrendelőtől a Szállítóhoz.

[21] Software Charged by Unit Delivered

6.2 Funkciópont szolgáltatás kihelyezésben (outsourcing)

A szolgáltatás kihelyezésnél, hagyományos vagy nem hagyományos fejlesztési és partnerségi szerződéseknél, amikor alkalmazásfejlesztésről, karbantartásról továbbfejlesztésről van szó, jelentős sikereket könyvelhet el ez a megközelítés[22].

6.3 Esettanulmányok

6.3.1 Kiskereskedelmi lánc Nagy Britanniában

Egy kiskereskedelmi lánc informatikai szolgáltatás kihelyezésről folytatott tárgyalásokat a potenciális vállalkozókkal. Fő cél a költségcsökkentés volt, ezért az ajánlattevők számára előírták, hogy az alkalmazásfejlesztési hatékonyságuknak az ipari átlagok felső negyedébe kell esnie. Ezt az értéket funkciópont / emberhónapban mérték az egyik ipari norma figyelő szervezet mérési eljárása szerint. A kiskereskedelmi láncnak nem volt korábban tapasztalata a szoftver tevékenységek teljesítmény mérésében.

A kiindulási állapot megállapítása érdekében termelékenységi méréseket végeztek a saját fejlesztő bázisukra vonatkoztatva. Ez azt mutatta, hogy a kiválasztott kis projekt mintában a termelékenység az ipari átlag harmada volt. Ez először meglepetésként érte a kiskereskedelmi láncot, de pozitív megközelítéssel úgy tekintették az eredményt, hogy a szolgáltatás kihelyezéssel milyen jelentős termelékenység nyereségre és ezen keresztül milyen jelentős költség megtakarításra tehetnek szert.

Ellenőrzésképpen megvizsgálták a „funkciópont teljesítési rátát", „funkciópont / felhasznált hetek" mértékegységben. Ez a paraméter pedig lényegesen magasabb volt az ipari átlagnál. Az derült ki, hogy a mérésre használt projekt minta esetében fő célkitűzés volt, hogy tekintet nélkül a költségekre minél gyorsabban elkészüljenek, mert a rendszerek fontosak voltak a cég versenyképessége szempontjából. Gyorsan szállítottak, de az ára az volt, hogy a projektre számolatlanul fordították az erőforrásokat. A termelékenység és a minőség is alacsony színvonalú volt, azonban az üzleti célokat a terveknek megfelelően elérték, és a projektek üzleti, versenyképességi szempontból nagy sikernek bizonyultak.

Ha a szolgáltatás kihelyezési szerződést csak a termelékenységi oldalról fogalmazzák meg, akkor a kiskereskedelmi lánc olyan helyzetbe kerül, ahol a szállító könnyedén teljesíti az előírt feltételeket, de a kiskereskedelmi lánc a hosszúra nyúlt fejlesztési idők miatt elveszítheti üzleti rugalmasságát. Ez a példa illusztrálja a szoftverfejlesztési, illetve projekt teljesítmény paraméterek közötti finom összjátékot, és az üzleti, szervezeti célokkal összhangban kell egy kiegyensúlyozott mértékrendszert kialakítani.

---

[22] http://www.acs.org.au/president/1997/outsrc/paper.htm (2000. január 12.),

Controlling Software Contracts, Author: Charles Symons, http://194.143.167.33/library/Papers/Symons/esepg97.html (2000. január 12.)

6.3.2 Szoftverkarbantartás szolgáltatás kihelyezésben

Egy angliai közmű vállalat informatikai igazgatója a következő szavakkal fejezte ki nemtetszését az alkalmazási rendszerek karbantartásával megbízott szállítótól kapott szolgáltatások miatt:

„Amikor előre rögzített áras szerződésünk volt, akkor sosem lehetett megtalálni őket. Amióta a felhasznált idő és anyag alapján kalkulálunk, nem tudunk megszabadulni tőlük."

Ez a példa azt mutatja, hogy milyen nehéz a felhasznált erőforrások mennyiségét mérni, és mennyivel könnyebb a teljesített, az elvégzett munka meghatározása. Megoldás volt, hogy kialakítottak néhány egyszerű becslési eljárást, formulát, amelyek illeszkedtek a helyi alkalmazásokhoz és az egyedi környezethez, és amelyek segítségével a megrendelő és a szállító gyorsan és egységesen meg tudott egyezni bármilyen szokásos feladat esetén.

A szokásos karbantartási feladatokat különböző összetettségi, bonyolultsági szintre fogalmazták meg, a karbantartásban érintett, megváltoztatandó állományok, képernyő mezők értelmében. Az egyes feladatok elvégzéséhez norma időket, „norma-órákat" írtak elő. Amikor egy felhasználónak valamilyen változtatásra volt szüksége, akkor az előre megállapodott formula alapján azt meg lehetett határozni. A szállító teljesítmény paramétereinek javítását, így a költség csökkentését szintén ilyen módon lehetett előírni, például a „norma-órák" szűkítését bizonyos feladatok esetén.

6.3.3 Szoftverkarbantartás és támogatás szolgáltatás kihelyezésben

Nagy nemzetközi ipari monopólium világszerte kihelyezte régi, elavult alkalmazási rendszereinek karbantartását és támogatását. A fizetési egység a funkciópont megállapított egységára volt, azaz a támogatás és a karbantartás hány funkciópontra terjed ki.

Ez első fázisban több ezer funkciópontot kellett megszámolni, megmérni. A feladatot úgy oldották meg, hogy nagyon gondosan kiválasztottak egy mintát és egy funkciópont becslési eljárást a szóba jövő rendszerek közül. Így lehetővé vált, hogy kielégítő pontossággal belátható időn és költségen belül a feladatot elvégezzék.

A szerződéses viszony komolyabb viták nélkül, simán működik. Periodikusan független, ipari norma összehasonlításokat végeznek külső referencia pontokhoz való viszonyítással, valamint a tapasztalatok alapján további mértékeket is bevezettek, pl. a minőségre vonatkozóan.

6.3.4 Parancsnoki, bevetési és irányítási rendszerszállító

A parancsnoki és irányítási rendszerek szállítója súlyos becslési és ajánlattételi problémákkal áll szemben. A tender felhívások több kötetesek, de a részletezettségük egyenetlen. Néhány követelményt nagyon részletesen határoznak meg, más esetben néhány egyszerű kijelentés bonyolult funkcionális szolgáltatásokat fed le. Egyre gyakrabban a megoldást különböző forrásokból származó, polcról a kereskedelemben megvásárolható (COTS, Commercial Off-The Shelf) szoftverek és egyéb újra felhasználható programok segítségével készítik el. A szállító feladata, hogy gondoskodjon arról a „ragasztó anyagról", amely ezeket a heterogén elemeket egy egységes, koherens rendszerbe rendezi el. Ezt a rendszer architektúra különböző

szintjein kell végrehajtani: például az ember-gép párbeszéd szintjén, az alkalmazási szinten, a köztes szoftver szintjén (middleware) és az ügyfél, kiszolgáló rendszerek (kliens-szerver) operációs rendszereinek szintjén.

Világos, hogy ilyen esetben az informatikai környezettől és a választott kereskedelmi programcsomagoktól függő (COTS) becslési eljárásokra és teljesítménymérési paraméterekre van szükség. Amikor egy tender felhívás megérkezik, gyorsan ki kell alakítani egy álláspontot a leendő szerződés méretére, hogy az ajánlati stratégia megfogalmazható legyen. Ehhez egyszerű ökölszabályokra van szükség, amelyek a jelentősebb entitások vagy entitás csoportok számára, valamint a szükséges COTS alkotórészek mennyiségére támaszkodnak. A funkciópont elemzés különösen hasznos ott, ahol megrendelésre készített egyedi szoftverekre van szükség.

6.3.5 EDS / XEROX

Az EDS és a XEROX között egy 3,2 milliárd US $-os szerződés esetében a funkciópont metrikát alkalmazva sikerült egy szoftver vitát a bírósági út kikerülésével rendezni.

6.3.6 Jogi precedens

Nagy Britanniában egy bírói döntés precedenst hozott létre egy megrendelő és egy szoftver szállító vitájában.

A megrendelő egy önkormányzati szervezet volt, 1,3 millió fontért indított egy projektet egy új adónem kezelésére.

A szoftver megbízhatatlannak bizonyult, a megrendelő költségei ennek következtében megduplázódtak (+1,3 millió angol font).

A bíróság úgy döntött, hogy a szoftver a beruházási javak, termékek közé tartozik, és a „célnak meg kell felelnie”. Azaz a gyártó felelős azért, hogy terméke úgy viselkedjen, ahogy azt ő állítja, vagy hirdeti.

A bíróság 12 millió font kártérítést ítélt meg a megrendelő számára, a szoftvergyártó cégnek még a negatív sajtóvisszhang következményeit is viselnie kellett.

A sikertelenség okai:

- a követelmények gyenge meghatározása;
- az informatikai, műszaki személyzet nem megfelelő felhasználása;
- rossz projektbecslés és gyenge projekt előrehaladási visszacsatolás.