ESTYMACJA KOSZTÓW OPROGRAMOWANIA

Dr hab. inż. Ilona Bluemke

szacowania

Zarządzający projektem musi znaleźć odpowiedź na pytania:

- ile wysiłku potrzeba na zakończenie aktywności
- ile czasu potrzeba na zakończenie aktywności
- jaki jest koszt czynności.

Pewne szacowania kosztów należy przeprowadzić już we wczesnym stadium (faza strategiczna) np. by określić cenę dla klienta.

Parametry wpływające na koszty:

- sprzętu i oprogramowania
- szkolenia, podróże
- „wysiłku” np. inżynierów i koszty utrzymania firmy (są one zwykle dominujące, np. płace *2 – koszty utrzymania firmy)

Współczynniki brane pod uwagę przy określaniu ceny oprogramowania

- **Rynek** - np. niższa cena by wejść na rynek
- **Niepewność kosztów –** jeśli koszty trudno jest dokładnie oszacować to do zysku trzeba dodać „coś” na niepewność szacowania
- **Kontraktowe** – np. klient może wyrazić zgodę na własność części kodu – do ewentualnego dalszego wykorzystania przez firmę
- **Zmienne wymagania** - Jeśli prawdopodobne, że wymagania się zmienią to można podać niską cenę (by wgrać przetarg) a określić wysoką cenę zmiany wymagań.
- **Kondycji finansowej firmy** – firmy w złej kondycji mogą obniżać zysk by utrzymać się w branży

PRODUKTYWNOŚĆ

Miary:

- związane z wielkością kodu np. liczba linii kodu/miesiąc (różna dla różnych języków programowania)
- związane z funkcjami (function point) (niezależne od języków programowania) Albrecht 1979, Albrecht&Gaffnej 1983

Punkty funkcyjne (function point)

Punkty funkcyjne określa się mierząc:

1. wejścia zewnętrzne
2. wyjścia zewnętrzne
3. interakcje użytkownika
4. interfejsy zewnętrzne
5. pliki używane przez system

UFC (unadjusted function point)

Każdy z punktów funkcyjnych ma przydzielony współczynnik (3- proste wejście zewnętrzne, 15 złożone pliki wewnętrzne)

UFC (unadjusted function point)

UFC = Σ liczby elem.typu * współczynnik

Otrzymana liczba jest modyfikowana współczynnikami określającymi złożoność całego projektu, wydajność, liczbę ponownie użytych elementów.

AFC (average number of lines of code)

Średnia liczba linii kodu na punkt – jest różna dla różnych języków programowania np.

- 2-300 LOC/FC asembler
- 2-40 LOC/FP języki 4GL

UFC i AFC pozwala na oszacowanie wielkości implementacji

TECHNIKI ESTYMACJI

- Algorytmiczne modelowanie kosztów (na podstawie inf. historycznych i przewidywanej wielkości projektu)
- Osąd ekspertów
- Estymacja poprzez analogię (technika możliwa jeśli inne projekty z tej dziedziny zostały zakończone)
- Prawo Parkinsona – praca rozrasta się aż do końca dostępnego czasu. Koszt określają dostępne zasoby i czas
- Cena do wygrania – koszt określa co klient może wydać na projekt

Algorytmiczne modelowanie kosztów

Wysiłek = C * MP^s * M

C – współczynnik złożoności projektu

MP – metryka produktu np. liczba linii kodu

s - bliskie 1 ale odzwierciedla wzrost wysiłku przy bardzo dużych projektach

M – współczynnik określający atrybuty produktu, procesu rozwoju

Powinno się wykonać estymację na najgorszy, oczekiwany i najlepszy przypadek

Model COCOMO (Boehm 1981)

Prosty projekt (aplikacje tworzone przez małe zespoły, aplikacje dobrze rozumiane):

PM = 2.4 (KDSI)$^{1.05}$ * M

Średniej złożoności projekt (bardziej złożone projekty, zespół ma niewielkie doświadczenie w dziedzinie problemu)

PM = 3.0 (KDSI)$^{1.12}$ * M

Złożone projekty:

PM = 3.6 (KDSI)$^{1.20}$ * M

modelu **podstawowym** COCOMO

M=1 korzysta się tylko z szacowanej wielkości kodu

KDSI (kilo delivered instructions) liczba tysięcy dostarczonych instrukcji źródłowych (bez liczenia komentarzy)

Model podstawowy daje punkt startowy estymacji kosztów.

Istnieją modele średni i szczegółowy.

Model **średni COCOMO**

Dodawane są współczynniki określające:

- niezawodność oprogramowania
- wielkość bazy danych
- ograniczenia pamięci, wykonania
- atrybuty personelu
- użycia narzędzi

0.7 (zmniejszony wysiłek) - 1.66 (zwiększony wysiłek)

Atrybuty sugerowane w modelu COCOMO

- produktu – niezawodność, wielkość bazy danych, złożoność produktu
- komputera – ograniczenia czasu, pamięci, stabilność platform sprzętowych na których oprogramowanie pracuje
- personelu – doświadczenie ludzi pracujących nad projektem
- projektu – związane z użyciem narzędzi , nowoczesnych technik, terminarz projektu

Przykład

Np. złożony system 128 000 (DSI)

Model podstawowy 1216 osobo/miesięcy

Niezawodność	1.4(wysoka)
Złożoność	1.3
Org.pamięci	1.2
Narzędzia	1.1(mały)
harmonogram	1.23 (przyśp) 3593(osob/mies)

Przykład 2

Niezawodność	0.75(niska)
Złożoność	0.7
Ograniczenia pamięci	1
Narzędzia	0.9(wysoki)
harmonogram	1(normalny)
	575(osob/mies.)

Kalibrowanie modelu

Porównywanie kosztów

oszacowanych z aktualnymi

daje współczynnik skalowania określający wpływ warunków lokalnych.

Model COCOMO określający czas trwania projektu

Proste projekty **TDEV = 2.5 (PM) $^{0.38}$**

Średnie projekty **TDEV = 2.5 (PM) $^{0.35}$**

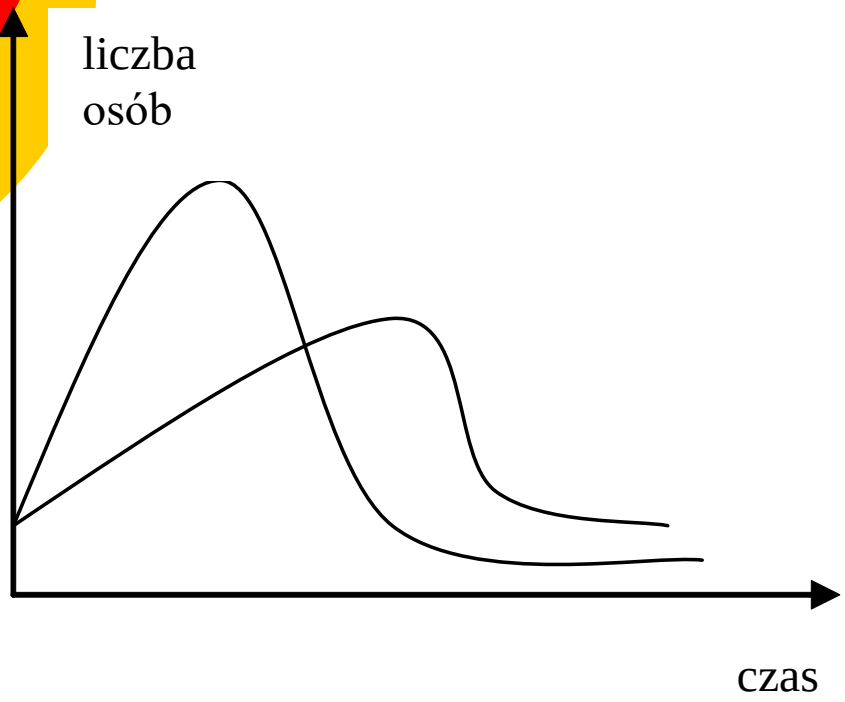
Złożone projekty **TDEV = 2.5 (PM) $^{0.32}$**

Np. 32000 przewidziano DSI

PM= 2.4 $(32)^{1.05}$ = 91p.m

TDEV = 2.5 $(91)^{0.38}$ = 14 miesięcy

Zespół pracujący nad projektem



Krzywa Rayleigh'a prawdopodobieństwa

Mała liczba osób zatrudniona na początku projektu – specyfikacja, planowanie. Po implementacji i testowaniu jednostek spada do 2-3 osób dostarczających produkt.

Model COCOMO 2

W modelu COCOMO 2 wzięto pod uwagę różne podejścia do produkcji oprogramowania. Poziomy modelu są związane z czynnościami procesu produkcji oprogramowania. Model COCOMO 2 proponuje trzy poziomy:

1. wczesnego prototypowania,
2. wczesnego projektowania,
3. post-architektoniczny.

punkty obiektowe (OP)

punkty obiektowe (OP) - Banker i inni w 1992 jako alternatywa dla punktów funkcyjnych.

Punkty obiektowe nie są klasami z projektu

Na liczbę punktów obiektowych składa się:

- **Liczba wyświetlanych ekranów**, proste ekrany - 1, bardziej złożone – 2, 3 .
- **Liczba tworzonych raportów**, raporty proste - 2, dla średnie -5, trudne - do 8 dla raportów,
- **Liczba modułów** 3GL, które należy opracować, by uzupełnić kod 4GL. Każdy moduł -10.

Wysiłek

Punkty obiektowe są podstawą do szacowania wysiłku potrzebnego do realizacji systemu.

Należy uwzględnić procentowe użycie gotowych komponentów (%reuse).

PM= (NOP * (1- %reuse/100)) /PROD

NOP jest liczba punktów obiektowych,

PROD jest produktywnością programisty

Produktywność w punktach obiektowych

Doświadczenia i umiejętności	Bardzo małe	Małe	Przeciętne	Duże	Bardzo duże
Możliwości CASE	Bardzo małe	Małe	Przeciętne	Duże	Bardzo duże
PROD (NOP/miesiąc)	4	7	13	25	50

Poziom wczesnego projektowania

$$Wysiłek = 2{,}5 * W^{s} * M$$

W– wielkość kodu źródłowego podawana jest w tysiącach linii **KDSI** (szacowanie liczby punktów funkcyjnych i przeliczeniu na linie kodu). Takie szacowania dotyczą kodu pisanego „ręcznie”.

Wykładnik ***s*** może mieć wartość z przedziału **1,1 do 1,24** zależnie od charakteru projektu, jego nowatorstwa, zespołu tworzącego oprogramowanie, firmy itp.

*M= RCPX*RUSE*PDIF* PERS*PREX*SCED*FCIL*

Współczynniki atrybutów produktu i przedsięwzięcia:

- RCPX - niezawodność i złożoność produktu
- RUSE – użycie wielokrotne
- PDIF – trudności platformy
- PERS – możliwości personelu
- PREX – doświadczenie personelu
- SCED – harmonogramu
- FCIL – udogodnienia pomocnicze.

Wysiłek związany z produkcją

$$PM = 2{,}5 * W^{s} * M + PMm$$

Gdzie:

$$PMm = (AKDSI * (AT/100))/ATPROD$$

a :

AKDSI – liczba automatycznie wygenerowanych linii kodu źródłowego,

AT – wyrażony w procentach odsetek kodu całkowitego, który został wygenerowany automatycznie,

ATPROD – poziom produktywności dla tworzenia kodu.

poziom postarchitektoniczny

Używa się tych samych formuł, co poziom wczesnego projektowania.

Szacowania powinny być dokładniejsze toteż używa się aż **17 atrybutów**.

W tej fazie szacuje się liczbę wierszy kodu, które muszą być zmodyfikowane, aby dostosować je do zmian wymagań systemowych.

Bierze się pod uwagę także możliwość wielokrotnego użycia kodu. Wielokrotne użycie kodu wiąże się z pracą związaną ze znalezieniem komponentów, zrozumieniem ich interfejsów oraz z modyfikacjami kodu (by dostosować go do komponentów).

Wpływ użycia wielokrotnego na wielkość kodu

ESLOC = ASLOC * (AA + SU + 0.4DM + 0.3CM +0.3 IM)/100

Gdzie:

ESLOC – równoważna liczba wierszy nowego kodu

ASLOC - liczba wierszy kodu użycia wielokrotnego

DM – procenty modyfikowanego projektu

CM - procenty modyfikowanego kodu

IM - procenty pierwotnej pracy integracyjnej wymaganej przy integracji kodu wielokrotnego

SU – określa koszt zrozumienia oprogramowania, przyjmuje wartości z zakresu 10 – dobrze napisany kod obiektowy do 50 – dla złożonego kodu.

AA – odzwierciedla początkowy koszt ustalenia, czy może być użyte oprogramowanie wielokrotne, przyjmuje wartości z zakresu 0 do 8.

Określenie wykładnika

wykładnik określa się na podstawie pięciu czynników skali (wartość z zakresu 0 – „bardzo duży” do 5 –„bardzo mały):

1. doświadczenie firmy z tego typu systemami,
2. elastyczność tworzenia, czyli udział klienta,
3. analiza ryzyka, duża wartość oznacza pełną analizę,
4. zespół zintegrowany, dobrze komunikujący się,
5. dojrzałość procesu w firmie, określa się przez odjęcie poziomu CMM firmy od 5

Określenie wykładnika -2

W celu otrzymania wykładnika wartości powyższych czynników należy

1. zsumować,
2. podzielić przez 100 i
3. dodać do 1.01.

Przykład –obliczanie wykładnika

- po raz pierwszy realizuje tego typu system - wartość 5 (brak doświadczeń),
- brak jest udziału strony klienta – wartość 1 (bardzo duża elastyczność),
- nowy zespół – wartość 3 (przeciętna),
- nie prowadzi się analizy ryzyka (wartość 5 – bardzo mała),
- firma ma poziom 2 CMM, czyli wartość 3.

Suma (5+1+3+5+3) wynosi 17,

a wykładnik będzie miał wartość 1.18.

Atrybuty produktu

1. RELY – wymagana niezawodność systemu,
2. CPLX – złożoność modułów systemowych,
3. DOCU – zakres wymaganej dokumentacji,
4. DATA – rozmiar użytej bazy danych,
5. RUSE – wymagany odsetek komponentów użycia wielokrotnego

Atrybuty komputera

6. TIME – ograniczenia na czas działania
7. PVOL – płynność platformy tworzenia
8. STOR – ograniczenia pamięciowe

Atrybuty personelu

9. ACAP – możliwości analityków systemowych
10. PCON – ciągłość zatrudnienia personelu
11. PEXP – doświadczenia programistów w dziedzinie przedsięwzięcia
12. PCAP – możliwości programistów
13. AEXP – doświadczenia analityków w dziedzinie przedsięwzięcia
14. LTEX- doświadczenie w stosowaniu języków i narzędzi

Atrybuty przedsięwzięcia

15. TOOL – użycie narzędzi programowych
16. SCED – kompresja harmonogramu tworzenia
17. SITE – stopień rozproszenia pracy po różnych ośrodkach i jakość komunikacji między ośrodkami

Szacowanie czasu trwania projektu

$$TDEV = 3 * (PM)^{(0.33 + 0.2 * (B-1.01))}$$

PM to wysiłek potrzebny do realizacji pracy,

B wykładnik, a w modelu wczesnego prototypowania *B=1*.

można uwzględnić skrócenie lub wydłużenie harmonogramu (SCEDPercentage).

$$TDEV = 3 * (PM)^{(0.33 + 0.2 * (B-1.01))} * SCEDPercentage/100$$

Np.

PM= 60 osobomiesięcy, B=1.17

$TDEV = 3\ (60)^{0.36} = 13$ miesięcy