

Cele estymacji złożoności i kosztów:

- Identyfikacja i uzasadnienie zapotrzebowania na zasoby,
- Ustalenie budżetu,
- Negocjacja zakresu
- Ocena wpływu ryzyka

Źródła problemów z szacowaniem kosztów projektów inf.:

- Unikalność przedsięwzięć,
- Trudności w określeniu miar wielkości i złożoności przedsięwzięcia,
- Trudności w oszacowaniu wydajności zasobów i organizacji,

Rodzaje szacowania:

- optymistyczne (minimalne)
- pesymistyczne (maksymalne),
- wartość oczekiwana

•z uwzględnieniem najprawdopodobniejszego przypadku

$$K = \frac{k_{\min} + 4 * k_{npr} + k_{\max}}{6}$$

$$K = \sum_{i=1}^n p_i k_i$$

$i=1, \dots, n$ – liczba wariantów oszacowania
 k_i – koszt w i -tym wariantcie
 p_i – prawdopodobieństwo i -tego wariantu

Metoda punktów funkcyjnych (1976 -...)**Miarą złożoności jest ilość punktów funkcyjnych:**

- Wejścia zewnętrzne,
- Wyjścia zewnętrzne,
- Interakcje użytkownika (zapytania),
- Pliki zewnętrzne (interfejsy do innych systemów),
- Pliki wewnętrzne

Współczynniki złożoności:

Rodzaj PF	Prosty	Przeciętny	Złożony
Wejście zewn.	3	4	6
Wyjście zewn.	4	4	7
Interakcja,	3	4	6
Plik zewn.	7	10	15
Plik wewn.	5	7	10

$$UFC = \sum_{i=1}^n k_i w_i$$

- UFC – unadjusted function count (nieskorygowana ilość punktów)
- k_i – ilość punktów o współczynniku w_i
- w_i - współczynnik złożoności

$$FP = UFC * TCF$$

$$TCF = 0,65 + 0,01 \sum_{i=1}^{14} f_i$$

- f_i – współczynniki złożoności technicznej
- Niezawodność archiwizowania / odtwarzania,
- Funkcje rozproszone,
- Głębokie konfigurowanie,
- Złożony interfejs,
- Współużywalność,
- Wymiana danych
- Osiągi,
- Interaktywne wprowadzanie danych
- Złożoność przetwarzania
- Łatwość instalacji

Cechy metody FP:

Zalety:

- Niezależność od implementacji,
- Możliwość oszacowania na podstawie specyfikacji zewnętrznej

Słabości:

- Wymagana pełna specyfikacja zewnętrzna,
- Tendencja do niedoszacowywania (na wstępnych etapach elementy generujące PF mogą być łatwo pominięte),
- Nieco subiektywna zasada zliczania PF,
- Nieuwzględniania ponownego użycia, elementów COTS, nowoczesnych technologii CASE,
- Konieczność oszacowania nakładów pracy na punkt funkcyjny na podstawie danych statystycznych z poprzednich projektów

Metoda COCOMO (1981)

Podstawą oceny złożoności projektu jest przewidywana liczba linii kodu źródłowego (Delivered Source Lines Of Code) w języku typu 3GL.

Zliczane linie kodu nie zawierają:

- Komentarzy,
- Kodu zapożyczonego (biblioteki, moduły włączane itp.)

Poziomy skomplikowania systemu:

Organiczny – obszar problemu znany, stabilne środowisko, swobodny interfejs, niewielki zespół o wysokim poziomie kwalifikacji, dobrze znane metody i narzędzia,

Półoderwany – pewne aspekty przedsięwzięcia, metody i narzędzia nie w pełni znane, członkowie zespołu o różnym stopniu zaawansowania

Osadzony – systemy o bardzo dużej złożoności, obszar problemu nieznan, zmieniające się środowisko, duża część członków zespołu nie ma doświadczeń w dane dziedzinie

Kolejność obliczeń:

- Nakład pracy (osobomiesiące),
- Czas trwania projektu (miesiące) – przy zatrudnieniu optymalnej liczby wykonawców

$$\text{Nakład} = a * \text{KDLOC}^b$$

$$\text{Czas} = 2,5 * \text{Nakład}^c$$

Rodzaj projektu	a	b	c
organiczny	2,4	1,05	0,38
półoderwany	3,0	1,12	0,35
osadzony	3,6	1,20	0,32

W tzw. wersji **pośredniej i złożonej** uwzględnia się przy wyliczaniu nakładu dodatkowe współczynniki trudności (15)

$$P = \prod_{i=1}^{15} p_i \quad \text{Nakład} = P * 2,8 * \text{KDLOC}^b$$

Współczynniki trudności - 6 poziomów

VL – very low, LO – low, NM – nominal, HI – high XH – extra high

Poziomom empirycznie przypisano dla różnych współczynników różne wartości z przedziału (0,70 – 1,66)

RELY	Wymagana niezawodność
DATA	Stosunek rozmiaru danych do kodu
CPLX	Złożoność kodu
TIME	Współczynnik wykorzystania mocy obliczeniowej
STOR	Współczynnik wykorzystania pamięci
VIRT	Zmienność środowiska sprzętowo-programowego
TURN	Czas przetwarzania
ACAP	Umiejętności analityczne
AEXP/LEXP/VEXP	Doświadczenie w zakresie aplikacji/języka i środowiska
MODP	Stosowanie nowoczesnych technik
TOOLS	Używania narzędzi automatyzujących proces
SCED	Spodziewana zmiana harmonogramu

Metoda DELPHI - wykorzystanie opinii grupy ekspertów do oszacowania kosztów i czasu projektu

- Koordynator przedstawia ekspertom specyfikację projektu
- Na spotkaniu ekspertów odbywa się dyskusja oszacowań,
- Eksperti przygotowują własne oszacowania min/avg/max
- Koordynator zbiera oszacowania i oblicza ich średnie
- Na kolejnym spotkaniu ekspertów odbywa się dyskusja aktualnych oszacowań,
- Proces jest powtarzany, aż do uzyskania wystarczająco zgodnych oszacowań

$$K = \frac{k_{\min} + 4 * k_{npr} + k_{\max}}{6}$$

Metoda COCOMO II (1996-1999)**Ulepszenia w stosunku do COCOMO:**

Zastosowanie innych technik szacowania we wczesnych fazach projektu (metoda punktów obiektowych i punktów funkcyjnych),

Dostosowanie do nowszych technik wytwarzania, uwzględnienie ponownego wykorzystania kodu i refaktoringu

Zwiększenie liczby uwzględnianych czynników wpływających na złożoność i koszt projektu

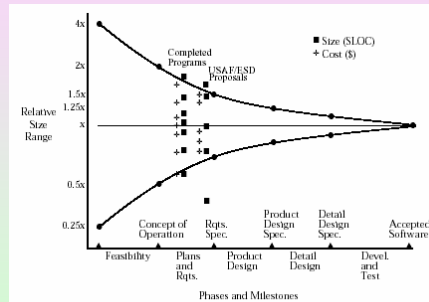


Figure 2. Software Costing and Sizing Accuracy vs. Phase

Źródło: www.sei.cmu.edu

Metoda punktów obiektowych - oszacowanie w fazie kompozycji aplikacji i prototypowania

Rozważane rodzaje punktów obiektowych:

- Formularz,
 - Okno dialogowe,
 - Raport
 - Obiekty i moduły w językach 3GL
-
- Wagi punktów szacowane są na podstawie ich dodatkowych atrybutów związanych ze złożonością
 - Uwzględnianie spodziewanego odsetka ponownego użycia obiektów

Procedura zliczania punktów obiektowych:

Step 1: Assess Object-Counts: estimate the number of screens, reports, and 3GL components that will comprise this application. Assume the standard definitions of these objects in your ICASE environment.

Step 2: Classify each object instance into simple, medium and difficult complexity levels depending on values of characteristic dimensions. Use the following scheme:

Number of Views contained	For Screens			Number of Sections contained	For Reports		
	# and source of data tables				# and source of data tables		
	Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)		Total < 4 (< 2 srvr < 3 clnt)	Total < 8 (2/3 srvr 3-5 clnt)	Total 8+ (> 3 srvr > 5 clnt)
< 3	simple	simple	medium	0 or 1	simple	simple	medium
3 - 7	simple	medium	difficult	2 or 3	simple	medium	difficult
> 8	medium	difficult	difficult	4 +	medium	difficult	difficult

Step 3: Weigh the number in each cell using the following scheme. The weights reflect the relative effort required to implement an instance of that complexity level.:

Object Type	Complexity-Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Component			10

Step 4: Determine Object-Points: add all the weighted object instances to get one number, the Object-Point count.

Step 5: Estimate percentage of reuse you expect to be achieved in this project. Compute the New Object Points to be developed,

$$NOP = (\text{Object Points}) \times \frac{(100 - \%reuse)}{100}$$

Step 6: Determine a productivity rate, PROD = NOP / person-month, from the following scheme

Developers' experience and capability	Very Low	Low	Nominal	High	Very High
ICASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Step 7: Compute the estimated person-months: PM = NOP / PROD.

Wykład 3 (12)

ZARZĄDZANIE PROJEKTEM INFORMATYCZNYM Szacowanie kosztów i złożoności projektu – COCOMO II

Procedura zliczania punktów funkcyjnych

Stosowana we wczesnych fazach projektu

W stosunku do standardowej metody PF nie uwzględnia współczynników złożoności technicznej, ale definiuje sposób oszacowania złożoności punktu

Step 1: Determine function counts by type. The unadjusted function counts should be counted by a lead technical person based on information in the software requirements and design documents. The number of each of the five user function types should be counted (Internal Logical File* (ILF), External Interface File (EIF), External Input (EI), External Output (EO), and External Inquiry (EQ)).

Step 2: Determine complexity-level function counts. Classify each function count into Low, Average and High complexity levels depending on the number of data element types contained and the number of file types referenced. Use the following scheme:

Record Elements	For ILF and EIF			File Types	For EO and EQ			File Types	For EI		
	Data Elements				Data Elements				Data Elements		
	1 - 19	20 - 50	51+		1 - 5	6 - 19	20+		1 - 4	5 - 15	16+
1	Low	Low	Avg	0 or 1	Low	Low	Avg	0 or 1	Low	Low	Avg
2 - 5	Low	Avg	High	2 - 3	Low	Avg	High	2 - 3	Low	Avg	High
6+	Avg	High	High	4+	Avg	High	High	3+	Avg	High	High

Step 3: Apply complexity weights. Weight the number in each cell using the following scheme. The weights reflect the relative value of the function to the user.

Function Type	Complexity-Weight		
	Low	Average	High
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Step 4: Compute Unadjusted Function Points. Add all the weighted functions counts to get one number, the Unadjusted Function Points.

* Note: The word *file* refers to a logically related group of data and not the physical implementation of those groups of data

Wykład 3 (13)	ZARZĄDZANIE Szacowanie			
Zasady zliczania linii kodu:				
Measurement unit:	Physical source lines			
	Logical source statements	4		
Statement type	Definition <input checked="" type="checkbox"/> Data Array <input type="checkbox"/>		Includes	Excludes
<i>When a line or statement contains more than one type, classify it as the type with the highest precedence.</i>				
1 Executable	Order of precedence →	1	<input checked="" type="checkbox"/>	
2 Nonexecutable				
3 Declarations		2	<input checked="" type="checkbox"/>	
4 Compiler directives		3	<input checked="" type="checkbox"/>	
5 Comments				
6 On their own lines		4		<input checked="" type="checkbox"/>
7 On lines with source code		5		<input checked="" type="checkbox"/>
8 Banners and non-blank spacers		6		<input checked="" type="checkbox"/>
9 Blank (empty) comments		7		<input checked="" type="checkbox"/>
10 Blank lines		8		<input checked="" type="checkbox"/>
11				
12				
How produced	Definition <input checked="" type="checkbox"/> Data array <input type="checkbox"/>		Includes	Excludes
1 Programmed			<input checked="" type="checkbox"/>	
2 Generated with source code generators				<input checked="" type="checkbox"/>
3 Converted with automated translators			<input checked="" type="checkbox"/>	
4 Copied or reused without change			<input checked="" type="checkbox"/>	
5 Modified			<input checked="" type="checkbox"/>	
6 Removed				<input checked="" type="checkbox"/>
7				
8				
Origin	Definition <input checked="" type="checkbox"/> Data array <input type="checkbox"/>		Includes	Excludes
1 New work: no prior existence			<input checked="" type="checkbox"/>	
2 Prior work: taken or adapted from				
3 A previous version, build, or release			<input checked="" type="checkbox"/>	
4 Commercial, off-the-shelf software (COTS), other than libraries				<input checked="" type="checkbox"/>
5 Government furnished software (GFS), other than reuse libraries				<input checked="" type="checkbox"/>
6 Another product				<input checked="" type="checkbox"/>
7 A vendor-supplied language support library (unmodified)				<input checked="" type="checkbox"/>
8 A vendor-supplied operating system or utility (unmodified)				<input checked="" type="checkbox"/>
9 A local or modified language support library or operating system				<input checked="" type="checkbox"/>
10 Other commercial library				<input checked="" type="checkbox"/>
11 A reuse library (software designed for reuse)			<input checked="" type="checkbox"/>	
12 Other software component or library			<input checked="" type="checkbox"/>	
13				
14				

Figure 2. Definition Checklist

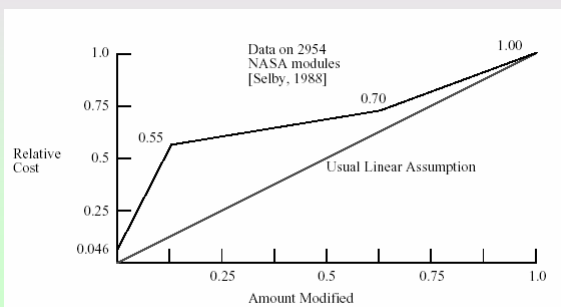
Wykład 3 (14)	ZARZĄDZANIE PROJEKTEM INFORMATYCZNYM Szacowanie kosztów i złożoności projektu – COCOMO II
Ponowne użycie i re-inżynieria	
<ul style="list-style-type: none"> • Koszt ponownego użycia nie jest liniową funkcją ilości zmienionych linii kodu • Koszt zrozumienia kodu, • Koszt selekcji, oceny i assimilacji, • Koszt analizy (ewentualnie naruszonych) interfejsów 	
$ESLOC = ASLOC \times \left(\frac{AA + SU}{100} + 0.4 \times DM + 0.3 \times CM + 0.3 \times IM \right)$	
DM – procent modyfikacji projektu	
CM – procent modyfikacji kodu	
IM – procent działań integracyjnych wynikających z modyfikacji w stosunku do integracji całego modułu	
SU – współczynnik zrozumiałości	
AA – współczynnik ocenialności i dostosowalności	

Table 1: Rating Scale for Software Understanding Increment SU

	Very Low	Low	Nom	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data / control structures.
Application Clarity	No match between program and application world views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world-views.
Self-Descriptiveness	Obscure code; documentation mission, obscure or obsolete	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentations.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment to AAF	50	40	30	20	10

Table 2: Rating Scale for Assessment and Assimilation Increment (AA)

AA Increment	Level of AA Effort
0	None
2	Basic module search and documentation
4	Some module Test and Evaluation (T&E), documentation
6	Considerable module T&E, documentation
8	Extensive module T&E, documentation



Wylizanie nakładu:

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

$$B = 1.01 + 0.01 \sum W_i$$

Scale Factors (W_i)	Very Low (5)	Low (4)	Nominal (3)	High (2)	Very High (1)	Extra High (0)
Precedentedness	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely famil- iar	thoroughly familiar
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Architecture / risk resolution*	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Team cohesion	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
Process maturity [†]	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

Uwzględnienie charakterystyki projektu i procesu – poprzez współczynniki podobne jak w COCOMO. W zależności od fazy różna liczba współczynników może być uwzględniona (7 – 17)

$$PM_{adjusted} = PM_{nominal} \times \left(\prod_i EM_i \right)$$

jakość - ogół cech i właściwości wyrobu lub usługi decydujący o zdolności wyrobu lub usługi do zaspokojenia stwierdzonych lub przewidywanych potrzeb użytkownika produktu

polityka jakości - ogół zamierzeń i kierunków działań organizacji dotyczących jakości, w sposób formalny wyrażony przez najwyższe kierownictwo organizacji, będącej systemem jakości

system jakości - odpowiednio zbudowana struktura organizacyjna z jednoznacznym podziałem odpowiedzialności, określeniem procedur, procesów i zasobów, umożliwiającą wdrożenie tzw. *zarządzania jakością*

zarządzanie jakością - jest związane z aspektem całości funkcji zarządzania organizacji, który jest decydujący w określaniu i wdrażaniu *polityki jakości*

audyt jakości - systematyczne i niezależne badanie, mające określić, czy działania dotyczące jakości i ich wyniki odpowiadają zaplanowanym ustaleniom, czy te ustalenia są skutecznie realizowane i czy pozwalają na osiągnięcie odpowiedniego *poziomu jakości*

Zapewnienie jakości jest rozumiane jako zespół działań zmierzających do wytworzenia u wszystkich zainteresowanych **przekonania**, że dostarczony produkt właściwie realizuje swoje funkcje i odpowiada aktualnym wymaganiom i standardom. Problem jakości, oprócz mierzalnych czynników technicznych, włącza dużą liczbę niemierzalnych obiektywnie czynników psychologicznych.

Podstawą obiektywnych wniosków co do jakości oprogramowania są pomiary pewnych parametrów użytkowych (niezawodności, szybkości, itd.) w realnym środowisku, np. przy użyciu metod statystycznych.

Atrybut jakościowy - cecha lub charakterystyka mająca wpływ na jakość danego obiektu

Atrybuty jakości (wg ISO 9126):

Atrybuty zewnętrzne:

Funkcjonalność:

- odpowiedniość
- dokładność
- współdziałanie
- zgodność
- bezpieczeństwo

Niezawodność:

- dojrzałość
- tolerancja błędów
- odtwarzalność
- stabilność

Efektywność:

- charakterystyka czasowa
- wykorzystanie zasobów

Użyteczność:

- rozumiałość
- łatwość uczenia
- łatwość posługiwania się

Przenośność:

- dostosowywalność
- instalacyjność
- zgodność
- zamiennność

Atrybuty wewnętrzne:

- **elastyczność** – łatwość przystosowania do środowiska lub zastosowań do których nie był pierwotnie projektowany
- **rozszerzalność** – łatwość rozbudowy
- **współużywalność** – łatwość wykorzystania części oprogramowania w innych systemach
- **podatność na zmiany** – łatwość wykonywania zmian w istniejących fragmentach oprogramowania
- **łatwość walidacji**
- **czytelność**
- **zrozumiałość**
- **testowalność**

Zapewnienie jakości dotyczy wszystkich artefaktów powstających w procesie, a w szczególności:

- Dokumentów,
- Kodu,
- Postaci instalacyjnej produktu końcowego,
- Usług (np. wdrożenia, szkoleń itp.)

Działania mające na celu zapewnienie jakości:**Strategiczne:**

- Podjęcie decyzji o wprowadzeniu polityki jakości
- Akceptacja kierownictwa
- Przyznanie środków i zasobów
- Zdefiniowanie i udokumentowanie
- Uświadomienie

Wykonawcze:

- Nadzorowanie,
- Stała kontrola (audyty, inspekcje, przeglądy, testowanie)
- Reagowanie

TQM – Total Quality Management

Koncepcja wymyślona przez Japończyka **Eiji Toyodę** dla potrzeb naprawy japońskiego przemysłu motoryzacyjnego - 1950 r. Główna jej myśl mówiła o tym, że w związku z tym, że to klient stanowi o rentowności przedsiębiorstwa, to należy tak sterować wszystkimi fazami procesu produkcyjnego wyrobu, aby klient był zadowolony z jakości tego wyrobu,

TQM została sformalizowana przez Amerykanów (W.E.Deming, P.Crosby, J.M.Juran, A.V.Feigenbaum), Japończyków (E.Toyoda, M.Imai, K.Ishikawa) i Brytyjczyka J.Oaklanda,

Podstawowe zasady:

- **Orientacja na klienta** - Jakość jest najważniejszym kryterium oceny przydatności produktów dla klienta, a to właśnie klient umożliwia funkcjonowanie wytwórcy tych produktów.
- **Zaangażowanie** - Każdy uczestnik procesu musi świadomie dbać o maksymalizację jakości wyników swojego działania

Czynniki warunkujące wprowadzenie procesu zarządzania jakością w/g TQM i ISO 9000:2001

- Ukierunkowanie na klienta (również klient wewnętrzny)
- Przywództwo (budowa wizji, identyfikacja wartości)
- Zaangażowanie ludzi (satysfakcja, motywacja, szkolenia)
- Podejście procesowe (koncentracja na poszczególnych krokach procesu i relacjach pomiędzy tymi krokami, pomiary)
- Podejście systemowe (całe otoczenie procesu wytwórczego)
- Ciągłe doskonalenie (doskonalenie stanu obecnego, ewolucja a nie rewolucja)
- Rzetelna informacja (zbieranie i zabezpieczanie danych do podejmowania obiektywnych decyzji)
- Partnerstwo dla jakości (bliskie związki producentów z klientami)

Źródła zagrożeń dla procesu zapewnienia jakości (wg ważności):

- niska dojrzałość organizacyjna wytwórcy (brak woli i konsekwencji ze strony zarządu),
- brak nadzoru,
- zbyt małe doświadczenie i uświadomienie personelu niższego szczebla,
- niesformalizowane (tworzone i zarządzane ad hoc) procedury,
- niedostateczne wykształcenie personelu,
- złożoność projektu,
- nowość projektu.