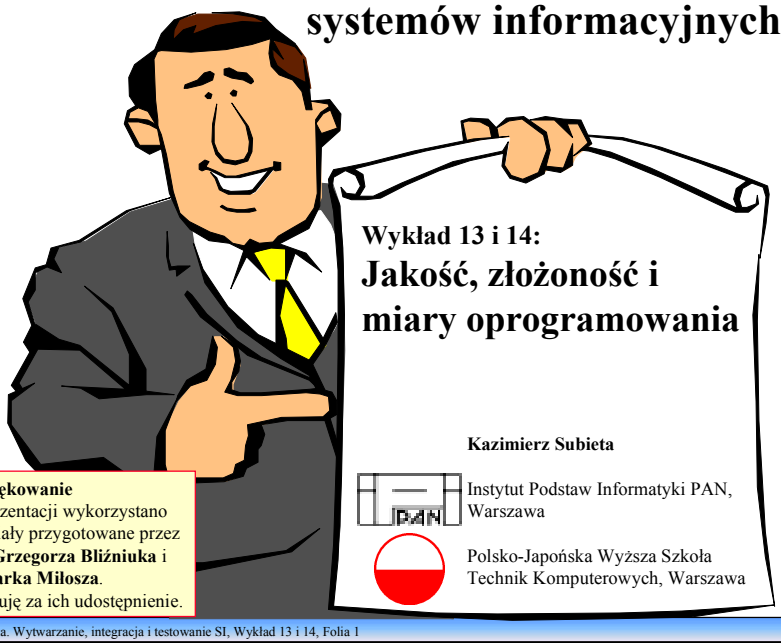
Wytwarzanie, integracja i testowanie systemów informacyjnych



Wykład 13 i 14:
Jakość, złożoność i miary oprogramowania

Kazimierz Subieta

Instytut Podstaw Informatyki PAN, Warszawa

Polsko-Japońska Wyższa Szkoła Technik Komputerowych, Warszawa

Podziękowanie
W prezentacji wykorzystano materiały przygotowane przez **mgr Grzegorza Bliźniuka** i **dr Marka Miłosza**. Dziękuję za ich udostępnienie.

Co to jest "jakość oprogramowania"?

Jest to jeden z najbardziej istotnych problemów inżynierii oprogramowania. W świecie przemysłowo-komercyjnym problem ten zaowocował wieloma artykułami opracowaniami, standardami, technikami, zaleceniami, itd.

- Zapewnienie jakości jest najczęściej rozumiane jako zespół działań zmierzających do wytworzenia u wszystkich zainteresowanych przekonania, że dostarczony produkt właściwie realizuje swoje funkcje i odpowiada aktualnym wymaganiom i standardom. Zatem 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. Niestety, obiektywne pomiary cech produktów programistycznych są utrudnione, niemożliwe lub obarczone ryzykiem błędu.

Trudności z oceną jakości oprogramowania

- Oceny jakości najczęściej muszą być znane zanim powstanie gotowy, działający produkt, co wyklucza zastosowanie obiektywnych metod pomiarowych.
- Wiele czynników składających się na jakość produktu jest niemierzalna.
- Produkty programistyczne są złożone i wieloaspektowe, co powoduje trudności w wyodrębnieniu cech mierzalnych, które odzwierciedlałyby istotne aspekty jakości.
- Produkty programistyczne mogą działać w różnych zastosowaniach, o różnej skali. Pomiary jakości mogą okazać się nieadekwatne przy zmianie skali (np. zwiększonej liczbie danych lub użytkowników), w innym środowisku, itp.
- Pomiary mogą okazać się bardzo kosztowne, czasochłonne lub niewykonalne (z powodu niemożliwości stworzenia środowiska pomiarowego przed wdrożeniem);
- Nie ma zgody co do tego, w jaki sposób pomierzone cechy danego produktu składają się na syntetyczny wskaźnik jego jakości.

W tej sytuacji oceny jakości produktów programistycznych są skazane na metody spekulacyjne, oparte na daleko posuniętych uproszczeniach oraz dowolnych założeniach, algorytmach, wzorach i heurystykach.

Zakres działań dla zapewnienia jakości

- Modele i miary służące ocenie kosztu i nakładu pracy
- Modele i miary wydajności ludzi
- Gromadzenie danych
- Modele i miary jakości
- Modele niezawodności
- Ocena i modelowanie wydajności oprogramowania
- Miary struktury i złożoności
- Ocena dojrzałości technologicznej
- Zarządzanie z wykorzystaniem metryk
- Ocena metod i narzędzi

Klasyfikacja zadań zapewnienia jakości

- Certyfikacja systemów przed skierowaniem do produkcji
- Wymuszanie standardów gromadzenia i przetwarzania danych
- Recenzowanie i certyfikacja wytwarzania i dokumentacji
- Opracowanie standardów dotyczących architektury systemu i praktyk programowania
- Recenzowanie projektu systemu pod względem kompletności
- Testowanie nowego lub zmodyfikowanego oprogramowania
- Opracowanie standardów zarządzania
- Szkolenie

Pomiary odgrywają istotną rolę, jednakże są one postrzegane jako jedno z wielu specjalistycznych działań, a nie podstawa całego procesu zapewnienia jakości.

Jakość w terminologii ISO 9000

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

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

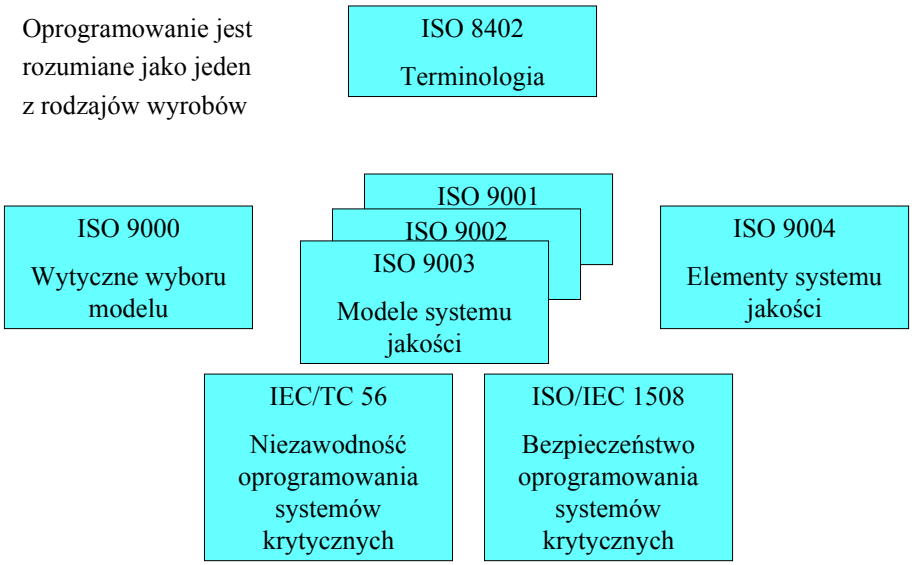
zarządzanie jakością - jest związane z aspektem całości funkcji zarządzania organizacji, który jest decydujący w określaniu i wdrażaniu *polityki jakości*

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

Normy dotyczące jakości

Oprogramowanie jest rozumiane jako jeden z rodzajów wyrobów



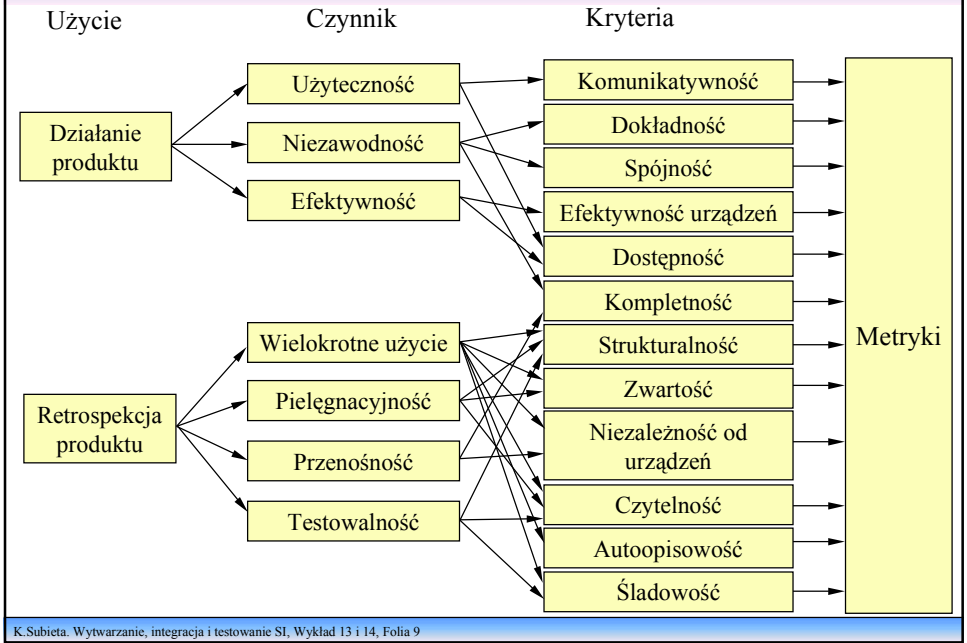
Norma IEEE-730

Norma IEEE-730 podaje ogólne ramy planu zapewniania jakości. Powinien on obejmować następujące zagadnienia:

- analiza punktów widzenia
- referencje wykonawcy
- zarządzanie przedsięwzięciem informatycznym
- dokumentacja
- standaryzacja działań
- przeglądy i audyty
- zarządzanie konfiguracją oprogramowania
- raport napotykanych trudności i podjętych działań prewencyjnych
- wykorzystywane metody i narzędzia
- kontrola kodu, mediów, dostawców
- zarządzanie hurtowniami danych
- pielęgnacja

Norma IEEE-730 została uzupełniona i uszczegółowiona normą IEEE-983.

Model jakości oprogramowania



CMM - model dojrzałości procesu wytwórczego

- wykorzystywany w procedurach klasyfikacji potencjalnych wykonawców oprogramowania dla Departamentu Obrony USA
- wyróżniono 5 poziomów dojrzałości wytwórców (poczynając od poziomu najniższego):
  - poziom początkowy - 1 (proces chaotyczny)
  - poziom powtarzalny - 2 (proces zindywidualizowany)
  - poziom zdefiniowany - 3 (proces zinstytucjonalizowany)
  - poziom zarządzany - 4 (proces + informacje zwrotne dla sterowania procesem)
  - poziom optymalizujący - 5 (proces + informacje zwrotne wpływające na ulepszenie procesu
- niewiele firm uzyskało poziom 3-ci, umożliwiający uzyskanie prawa dostaw dla Departamentu Obrony USA,
- tylko IBM w zakresie oprogramowania promu kosmicznego dla NASA uzyskała poziom 5-ty (najwyższy)

Pomiary oprogramowania

Pomiar (*measurement*) jest to proces, w którym atrybutom swiata rzeczywistego przydzielane są liczby lub symbole w taki sposób, aby charakteryzować te atrybuty według jasno okreslonych zasad. Jednostki przydzielane atrybutom nazywane są **miarą** danego atrybutu.

Metryka (*metric*) jest to proponowana (postulowana) miara. Nie zawsze charakteryzuje ona w sposób obiektywny dany atrybut. Np. ilość linii kodu (LOC) jest metryką charakteryzującą atrybut "długość programu źródłowego", ale nie jest miarą ani złożoności ani rozmiaru programu (choć występuje w tej roli).

Co mierzyć?

Proces: każde określone działanie w ramach projektu, wytwarzania lub eksploatacji oprogramowania.

Produkt: każdy przedmiot powstały w wyniku procesu: kod źródłowy, specyfikację projektową, udokumentowaną modyfikację, plan testów, dokumenatcję, itd.

Zasób: każdy element niezbędny do realizacji procesu: osoby, kompilatory, narzędzia, metody wytwarzania, itd.

Elementy pomiaru oprogramowania - produkty

Obiekty	**Atrybuty bezpośrednio mierzalne**	**Wskaźniki syntetyczne**
Specyfikacje	rozmiar, ponowne użycie, modularność, nadmiarowość, funkcjonalność, poprawność składniową, ...	zrozumiałość, pielęgnacyjność, ...
Projekty	rozmiar, ponowne użycie, modularność, spójność, funkcjonalność,...	jakość, złożoność, pielęgnacyjność, ...
Kod	rozmiar, ponowne użycie, modularność, spójność, złożoność, strukturalność, ...	niezawodność, używalność, pielęgnacyjność, ...
Dane testowe	rozmiar, poziom pokrycia,...	jakość,...
....	....	....

Elementy pomiaru oprogramowania - procesy

Obiekty	Atrybuty bezpośrednio mierzalne	Wskaźniki syntetyczne
Specyfikacja architektury	czas, nakład pracy, liczba zmian wymagań, ...	jakość, koszt, stabilność, ...
Projekt szczegółowy	czas, nakład pracy, liczba znalezionych usterek specyfikacji,...	koszt, opłacalność, ...
Testowanie	czas, nakład pracy, liczba znalezionych błędów kodu, ...	koszt, opłacalność, stabilność, ...
....	....	....

Elementy pomiaru oprogramowania - zasoby

Obiekty	Atrybuty bezpośrednio mierzalne	Wskaźniki syntetyczne
Personel	wiek, cena, ...	wydajność, doświadczenie, inteligencja, ...
Zespoły	wielkość, poziom komunikacji, struktura,...	wydajność, jakość, ...
Oprogramowanie	cena, wielkość, ...	używalność, niezawodność, ...
Sprzęt	cena, szybkość, wielkość pamięci	niezawodność, ...
Biura	wielkość, temperatura, oświetlenie,...	wygoda, jakość,...
....	....	....

Przykładowe pomiary i estymacje (1)

Opracowano ogromną liczba różnorodnych metryk uwzględniających m.in. następujące aspekty

wrażliwość na błędy,
➔możliwości testowania,
➔częstotliwość występowania awarii,
➔dostępność systemu,
➔propagacja błędów,
➔ilość linii kodu, złożoność kodu, złożoność programu,
➔złożoność obliczeniową, funkcjonalną, modułową,
➔łatwość implementacji,
➔rozmiar dokumentacji,
➔ilość zadań wykonanych terminowo i po terminie,

➔współzależność zadań,
➔wielkość i koszt projektu,
➔czas trwania projektu,
➔zagrożenia projektu (ryzyko),
➔czas gotowości produktu,
➔kompletność wymagań, kompletność planowania,
➔stabilność wymagań,
➔odpowiedniość posiadanych zasobów sprzętowych, materiałowych i ludzkich,
➔efektywność zespołu, efektywność poszczególnych osób,
➔.....

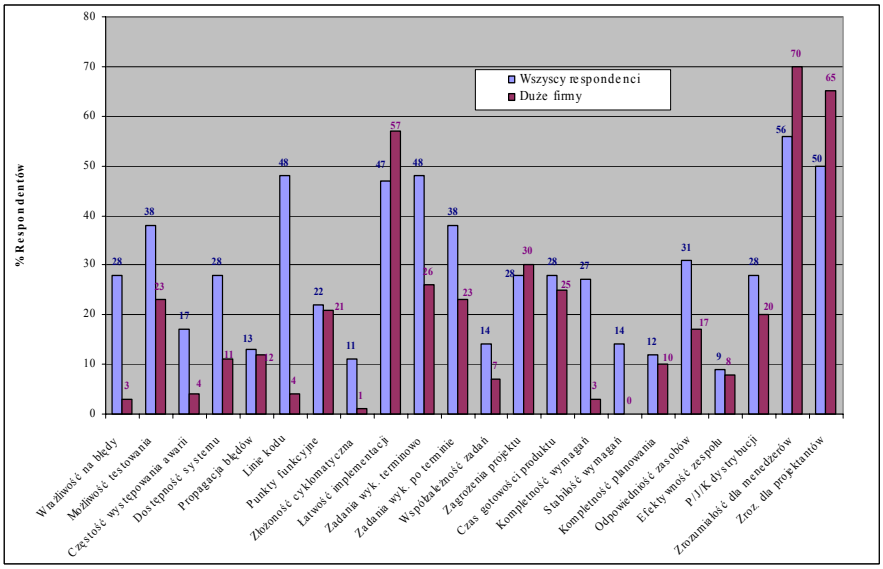
Przykładowe pomiary i estymacje (2)

- Metryki zapisu projektu, kodu programu
  - » rozmiar projektu, kodu programu (liczba modułów/obiektów, liczba linii kodu, komentarza, średni rozmiar komponentu)
  - » liczba, złożoność jednostek syntaktycznych i leksykalnych
  - » złożoność struktury i związków pomiędzy komponentami programu
    (procesy, funkcje, moduły, obiekty itp..)
- Metryki uzyskiwanego produktu
  - » rozmiar
  - » architektura
  - » struktura
  - » jakość użytkowania i pielęgnacji
  - » złożoność

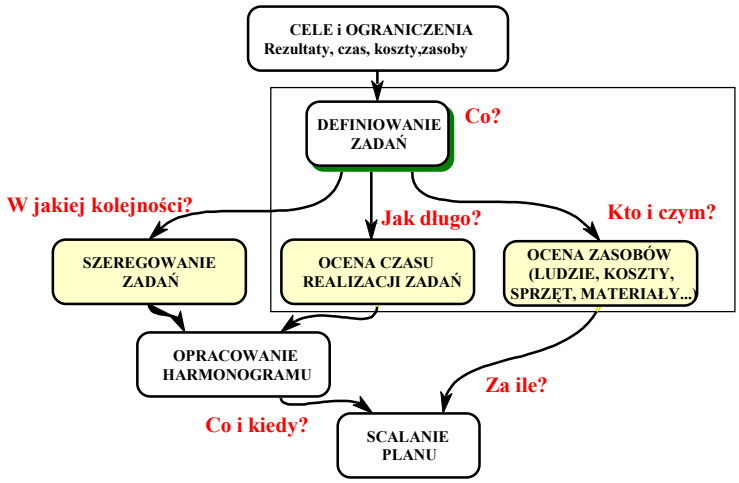
Przykładowe pomiary i estymacje (3)

- Metryki procesu wytwarzania
  - » dojrzałość realizacji systemu
  - » zarządzanie wytwarzaniem oprogramowania
  - » w odniesieniu do cyklu życia oprogramowania
- Metryki zasobów realizacyjnych
  - » w odniesieniu do personelu „zamieszanego” w realizację
  - » narzędzia software’owe, wykorzystywane przy realizacji
  - » sprzęt, jakim dysponuje wykonawca

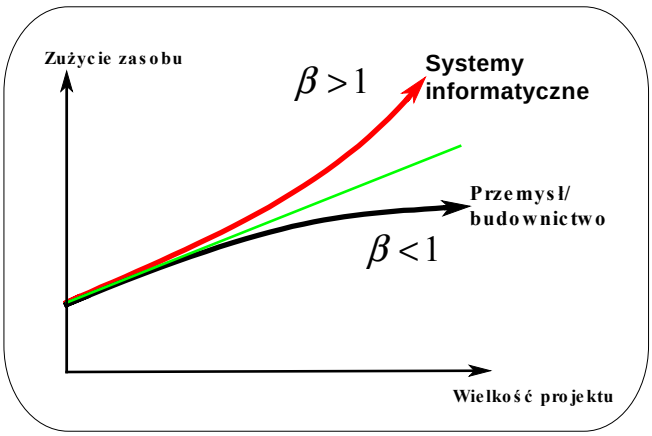
Wykorzystanie metod estymacyjnych



Ocena złożoności w planowaniu projektu



Efekt skali



Efekt skali - za i przeciw

ZA

- Specjalizacja
- „Krzywa uczenia się”
- Narzędzia CASE
- Wspomaganie dokumentowania
- Biblioteki gotowych elementów
- Stałe koszty projektu

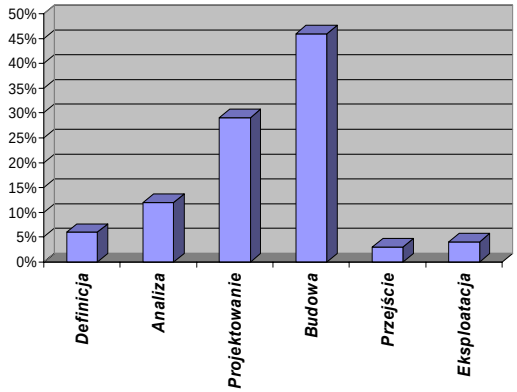
PRZECIW

- Koszty zarządzania (czas produkcyjny/nie)
- Lawinowy wzrost ilości powiązań
- Komunikacja wewnątrz zespołu
- Wzrost złożoności testowania

$$\beta = 1{,}05 \rightarrow 1{,}83$$

Etapy i koszty wytwarzania oprogramowania

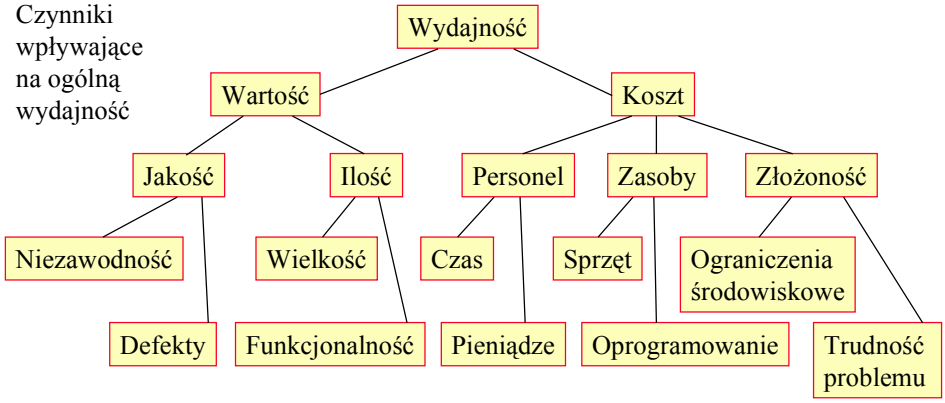
Empiryczne koszty poszczególnych faz wytwarzania oprogramowania systemów informatycznych



Źródło: Oracle Corp. Badaniom podlegały realizacje systemów przetwarzania danych, realizowane metodą CDM, prze użyciu narzędzi CASE firmy Oracle.

Modele i miary wydajności ludzi

Czynniki wpływające na ogólną wydajność



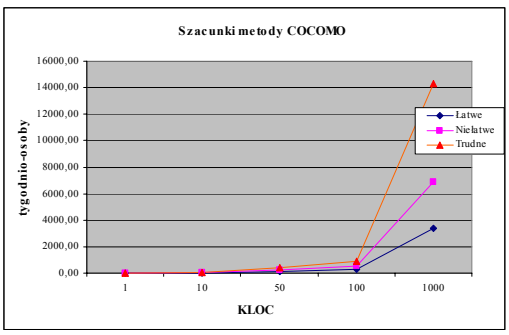
Mylące, wręcz niebezpieczne jest zastępowanie wielu miar jedną miarą, np. długością wyprodukowanego kodu.

Metryka złożoności COCOMO

Metryka *COCOMO* podana przez Boehma w 1981r bada rozmiar programu na podstawie formuły: $E = b * KLOC^{c}$

KLOC - „kilo linie" kodu programu. Wartości stałych *b* i *c* są zależne od złożoności projektu:

- projekty łatwe: b=2.4 c=1.05
- projekty niełatwe b=3.0 c=1.12
- projekty trudne b=3.6 c=1.20



Analiza Punktów Funkcyjnych

Metoda analizy punktów funkcyjnych (FPA), opracowana przez Albrechta w latach 1979-1983 bada pewien zestaw wartości. Łączy ona własności metody, badającej rozmiar projektu programu z możliwościami metody badającej produkt programowy.

$$UFP = 4I + 5O + 4E + 10L + 7F \quad \text{, gdzie}$$

Liczbę nie skorygowanych punktów funkcyjnych wylicza się z formuły:

- *I* - liczba typów (obiektów) wejściowych, wpływających na dane
- *O* - liczba typów (obiektów) wyjściowych, związanych z wartościami danych
- *E* - liczba typów (obiektów) sterujących, nie zmieniających danych
- *L* - liczba wewnętrznych plików roboczych (np. indeksowych)
- *F* - liczba interfejsów z otoczeniem programu

UFP - nieskorygowane punkty

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} * n_{ij}$$

UPF- nieskorygowane punkty funkcyjne

Lp.	Element przetwarzania	Poziom złożoności elementu przetwarzania (j)		
		prosty	średni	złożony
1	Wejścia użytkownika	**3**	**4**	**6**
2	Wyjścia użytkownika	**4**	**5**	**7**
3	Zbiory danych wewnętrzne	**7**	**10**	**15**
4	Zbiory danych zewnętrzne	**5**	**7**	**10**
5	Zapytania zewnętrzne	**3**	**4**	**6**

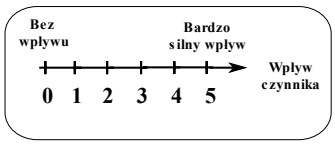
n_{ij} – ilość elementów, w_{ij} - wagi

Korekcja Punktów Funkcyjnych

- występowanie urządzeń komunikacyjnych
- rozproszenie przetwarzania
- długość czasu oczekiwania na odpowiedź systemu
- stopień obciążenia sprzętu istniejącego
- częstotliwość wykonywania dużych transakcji
- wprowadzanie danych w trybie bezpośrednim
- wydajność użytkownika końcowego
- aktualizacja danych w trybie bezpośrednim
- złożoność przetwarzania danych
- możliwość ponownego użycia programów w innych zastosowaniach
- łatwość instalacji
- łatwość obsługi systemu
- rozproszenie terytorialne
- łatwość wprowadzania zmian - pielęgnowania systemu

19 - British Mark II

Skorygowane Punkty Funkcyjne



kompleksowy współczynnik korygujący

$$VAF = \sum_{k=1}^{14} k_k$$

Punkty funkcyjne (FPs):

$$FP = (0{,}65 + 0{,}01 * VAF) * UFP$$

$$FP = (0{,}65 \ldots 1{,}35) * UFP$$

Kolejność obliczeń Punktów Funkcyjnych

- Identyfikacja systemu
- Obliczenie współczynnika korygującego
- Wyznaczenie ilości zbiorów danych i ich złożoności
- Wyznaczenie ilości i złożoności elementów funkcjonalnych (we, wy, zapytania)
- Realizacja obliczeń
- Weryfikacja
- Raport, zebranie recenzujące

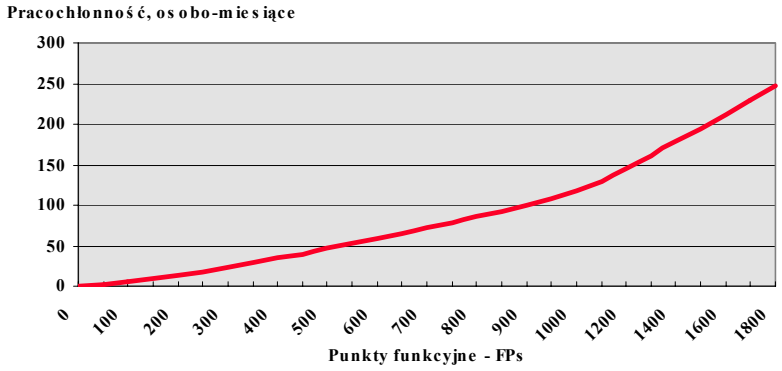
Przykład obliczania punktów funkcyjnych

Lp	Elementy	Proste	Średnie	Złożone	Razem NPF
1	Wejścia	3 x 2	4 x 5	6 x 3	*44*
2	Wyjścia	4 x 10	5 x 4	7 x 5	*95*
3	Zbiory wew.	7 x 3	10 x 5	15 x 2	*101*
4	Zbiory zew.	5 x 0	7 x 3	10 x 0	*21*
5	Zapytania	3 x 10	4 x 5	6 x 12	*122*
				Łącznie	383

Aplikacje a Punkty Funkcyjne

- 1 FP ≈ 125 instrukcji w C
- 10 FPs - typowy mały program tworzony samodzielnie przez klienta (1 m-c)
- 100 FPs - większość popularnych aplikacji; wartość typowa dla aplikacji tworzonych przez klienta samodzielnie (6 m-cy)
- 1,000 FPs - komercyjne aplikacje w MS Windows, małe aplikacje klient-serwer (10 osób, ponad 12 m-cy)
- 10,000 FPs - systemy (100 osób, ponad 18 m-cy)
- 100,000 FPs - MS Windows'95, MVS, systemy militarne

Punkty Funkcyjne a pracochłonność



Wykorzystanie punktów funkcyjnych

- Ocena złożoności realizacji systemów
- Audyt projektów
- Wybór systemów informatycznych funkcjonujących w przedsiębiorstwie do reinżynierii (wg. koszt utrzymania/FPs)
- Szacowanie liczby testów
- Ocena jakości pracy i wydajności zespołów ludzkich
- Ocena stopnia zmian, wprowadzanych przez użytkownika na poszczególnych etapach budowy systemu informatycznego
- Prognozowanie kosztów pielęgnacji i rozwoju systemów
- Porównanie i ocena różnych ofert dostawców oprogramowania pod kątem merytorycznym i kosztowym

Punkty mieszane

Próby rozszerzenia metody punktów funkcyjnych na systemy hybrydowe (wszystkie ważne struktury systemu):

- **Punkty funkcyjne (Function Points)**
- **Punkty bazodanowe (Data Points)**
- **Punkty sprzętowe (Hardware Points)**
- **Punkty serwisyjne (Service Points)**

Punkty Funkcyjne a języki baz danych

Typ języka lub konkretny język	Poziom języka wg. SPR	Efektywność LOC/FP
Access	8,50	38
ANSI SQL	25,00	13
CLARION	5,50	58
CA Clipper	17,00	19
dBase III	8,00	40
dBase IV	9,00	36
DELPHI	11,00	29
FOXPRO 2.5	9,50	34
INFORMIX	8,00	40
MAGIC	15,00	21
ORACLE	8,00	40
Oracle Developer/2000	14,00	23
PROGRESS v. 4	9,00	36
SYBASE	8,00	40

wg. Software Productivity Research

FPs a wydajność zespołu

Poziom języka wg. SPR	Średnia produktywność, FPs/osobomiesiąc
1-3	5-10
4-8	10-20
9-15	16-23
16-23	15-30
24-55	30-50
>55	40-100

wg. Software Productivity Research