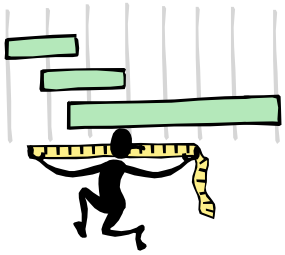
Zaawansowana inżynieria oprogramowania

Szacowanie rozmiaru oprogramowania



Koncepcja wykładu: **Jerzy Nawrocki/Łukasz Olek**
Slajdy/Lektor/Montaż: **Łukasz Olek**

UCZELNIA ONLINE

Witam Państwa na kolejnym wykładzie z cyklu „Zaawansowana inżynieria oprogramowania” poświęconemu szacowaniu rozmiaru.

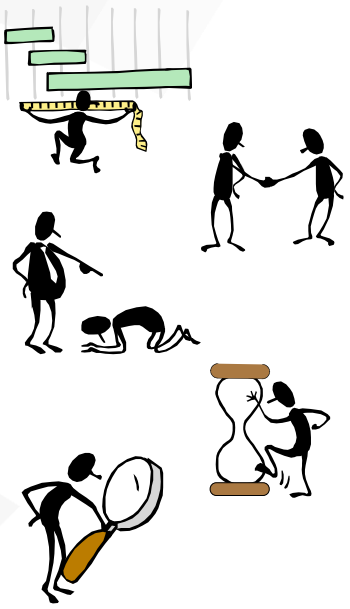
Plan wykładów

Standardy serii ISO 9000
Model dojrzałości CMMI
Zarządzanie przedsięwzięciami i PRINCE2, cz. I
Zarządzanie przedsięwzięciami i PRINCE2, cz. II
Personal Software Process, cz. I
Personal Software Process, cz. II
Metodyki programowania: TSP i RUP
Pozyskiwanie i dokumentowanie wymagań (IEEE 830)
Wymagania pozafunkcyjne i ISO 9126
Zarządzanie ryzykiem
Systemy krytyczne i HAZOP
Szacowanie rozmiaru oprogramowania
Szacowanie pracochłonności

Jest to przedostatni wykład z tej serii. Problem poruszany na tym wykładzie jest dużym problemem wielu współczesnych firm i aspektem, który jest niestety często zaniedbywany w projektach informatycznych.

Wprowadzenie

- Po co szacować rozmiar?
  - Wycena oferty
  - Ocena wielkości zadań
    - Mierzenie wydajności
  - Kontrola postępów projektu
  - Mierzenie jakości



Nie każdy zdaje sobie sprawę z korzyści, jakie przynosi kierownikom projektów.

Po pierwsze, szacowanie rozmiaru to konieczna czynność do ustalenia budżetu i harmonogramu. W związku z tym przydaje się podczas początkowych rozmów z klientami - aby wycenić ofertę.

Po drugie, aby mierzyć wydajność poszczególnych pracowników, niezbędne jest ocenianie wielkości zadań, jakie im przydzielamy. W związku z tym, iż stosunek wydajności najszybszego i najwolniejszego programisty wynosi 10:1, warto znać wydajność poszczególnych pracowników, aby za wszelką cenę zachować przy sobie tych najlepszych.

Po trzecie, w sytuacji kiedy nie wiemy jak duży jest projekt, nie jesteśmy w stanie powiedzieć w jakim stopniu jest on już skończony.

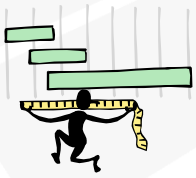
Po czwarte, znajomość rozmiaru oprogramowania przydaje się do mierzenia jakości.

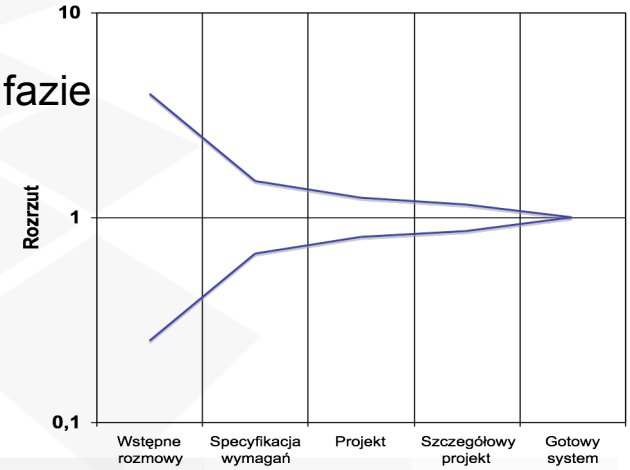
Możemy liczyć liczbę defektów/projekt. Lecz projekty mają różną wielkość, więc porównywanie tych wartości byłoby niepoprawne. W momencie kiedy znamy rozmiar możemy policzyć liczbę defektów na 1000 linii kodu (LOC) - co jest dużo bardziej obiektywną miarą.

Wprowadzenie - mity

- ~~Na podstawie wymagań można dokładnie oszacować koszt projektu~~

- Szacowanie w tej fazie ma ograniczoną dokładność





Na samym początku należy rozwiać pewne mity związane z szacowaniem wielkości oprogramowania. Po pierwsze - nigdy nie jesteśmy w stanie podać dokładnych szacunków. Literatura mówi o widełkach różnej szerokości - w zależności od etapu projektu. I tak:

-na etapie wstępnych wymagań możemy się pomylić 4x (koszt może wyjść 4x większy, jak również 4x mniejszy)

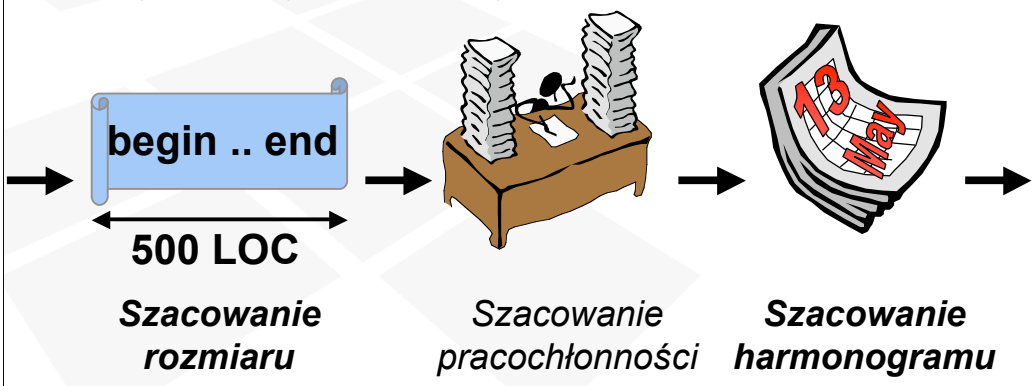
-na etapie dokładnej specyfikacji wymagań statystycznie można się pomylić 1,5x

-na etapie projektu architektury rozrzut maleje do 1,25

Dopiero tuż przed końcem projektu jesteśmy w stanie powiedzieć ile projekt na prawdę kosztował (lecz wtedy taki „szacunek” nie jest nikomu potrzebny)

Wprowadzenie - mity

- ~~Można od razu na podstawie wymagań powiedzieć jak długo projekt będzie trwał~~
- Systematyczne podejście do szacowania:



Szacowanie rozmiaru *Szacowanie pracochłonności* ***Szacowanie harmonogramu***

Nie można również od razu na podstawie wymagań powiedzieć jak długo projekt będzie trwał. Szacowanie to czynność złożona - najpierw należy oszacować rozmiar oprogramowania, następnie pracochłonność, a dopiero wtedy nanieść wszystko na kalendarz i powiedzieć jak długo projekt potrwa.

Plan wykładu

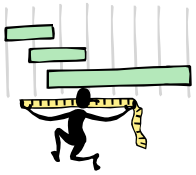
- Systematyczne podejście do planowania
  - Szacowanie rozmiaru:
    - Metoda punktów funkcyjnych
  - Szacowanie pracochłonności
  - Planowanie, a kalendarz
- Wykorzystanie szacowania rozmiaru:
  - Śledzenie postępów prac
- Podsumowanie

Plan wykładu wygląda następująco. Najpierw przedstawimy metodę punktów funkcyjnych, która służy do szacowania rozmiaru oprogramowania na podstawie wymagań. Następnie po krótce przedstawimy kolejne „ogniwa” systematycznego podejścia do planowania: szacowanie pracochłonności oraz szacowanie harmonogramu.

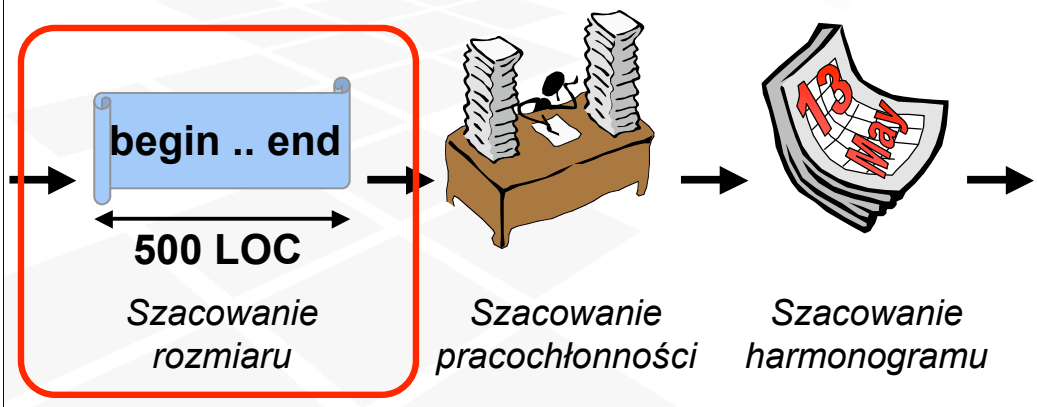
W dalszej części przedstawimy metodę wartości zarobionej, która służy do śledzenia postępów prac nad projektem.

Plan wykładu

- Systematyczne podejście do planowania
  - **Szacowanie rozmiaru:**
    - **Metoda punktów funkcyjnych**
  - Szacowanie pracochłonności
  - Planowanie, a kalendarz
- Wykorzystanie szacowania rozmiaru:
  - Śledzenie postępów prac
- Podsumowanie



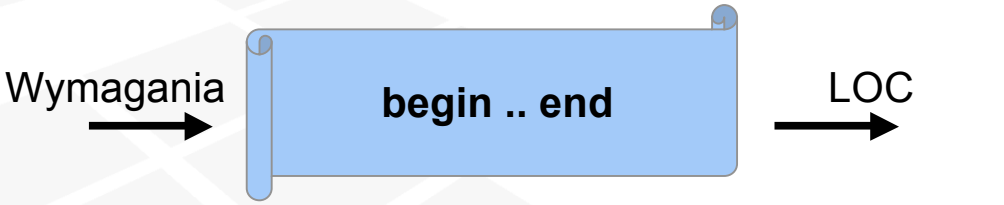
Metoda punktów funkcyjnych



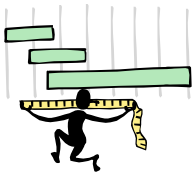
Metoda punktów funkcyjnych to pierwsza czynność systematycznego podejścia do planowania.

Metoda punktów funkcyjnych

Metoda punktów funkcyjnych
(ang. Function Point Analysis, FPA)



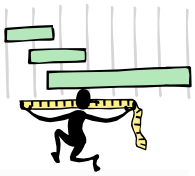
Szacowanie rozmiaru



Dzięki niej możemy na podstawie wymagań systemu informatycznego uzyskać miarę wielkości takiego systemu, w punktach funkcyjnych, lub nawet w liniach kodu.

Metoda punktów funkcyjnych

- Sposób mierzenia funkcjonalności
- Funkcjonalność, czyli system:
  - z punktu widzenia użytkownika
  - niezależnie od technologii
- Punkt funkcyjny:
  - miara „jednostki” funkcjonalności
  - 324 punkty funkcyjne (*fp*)



Metoda punktów funkcyjnych pozwala mierzyć funkcjonalność, czyli oceniać wielkość systemu z punktu widzenia użytkowników. Użytkownik rzadko kiedy zdaje sobie sprawę z zastosowanej technologii, czy architektury, tak więc metoda ta musi od tych rzeczy abstrahować.

No dobrze, ale jak tu abstrahować od technologii, skoro każdy z nas wie, że stworzenie systemu w różnych technologiach zajmuje przecież różną ilość czasu. Tworzenie czegoś bezpośrednio w assemblerze na pewno będzie trwało więcej, niż w języku wyższego poziomu. Zgadza się, lecz tutaj otrzymujemy tylko liczbę punktów funkcyjnych, którą możemy przekształcić w liczbę linii kodu stosując odpowiedni mnożnik. Tak więc metoda jest niezależna od technologii.

Liczba punktów funkcyjnych, ocenia „ilość funkcjonalności”, którą dostaje użytkownik.

Dzięki temu można powiedzieć, że pewien system ma 324 punkty funkcyjne, a inny np. 543.

Wprowadzenie

- Alan Albrecht, IBM, 1977
- Podstawowe funkcje:
  - wejścia (EI)
  - wyjścia (EO)
  - zapytania (EQ)
  - wewnętrzne pliki danych (ILF)
  - zewnętrzne interfejsy (EIF)

Twórcą metody punktów funkcyjnych jest Alan Albrecht, który wymyślił ją w 1977 roku, kiedy pracował w firmie IBM.

Wyodrębnił on podstawowe funkcje, jakie są istotne dla użytkownika:

-wejścia

-wyjścia

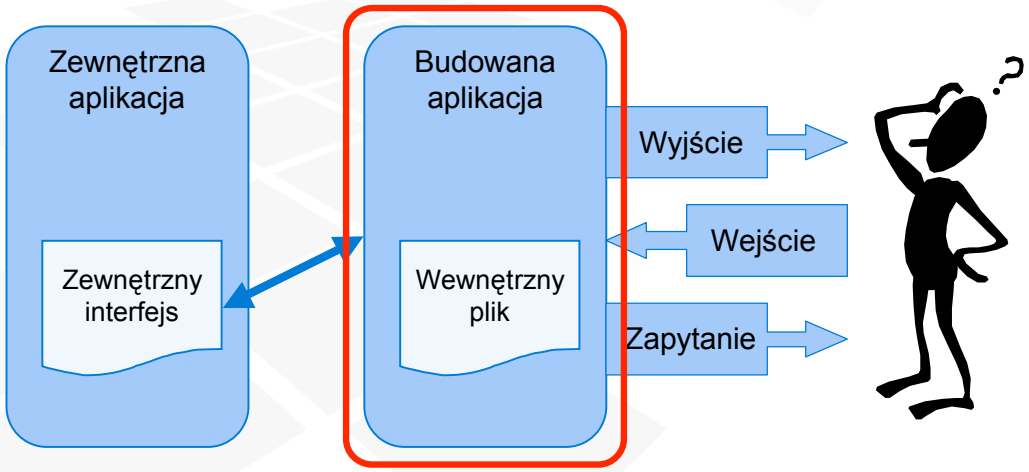
-zapytania

-wewnętrzne pliki danych, oraz

-zewnętrzne interfejsy

Funkcje te składają się na „model” aplikacji, czyli każdy system można rozbić na te elementy. Robi się to na podstawie wymagań systemu.

Metoda punktów funkcyjnych



Obrazowo można to przedstawić na następującym diagramie. Szacowany system jest zaznaczony na czerwono.

Poszczególne elementy to:

- plik wewnętrzny - permanentnie przechowuje dane istotne dla użytkownika. System informacyjny zarządza tymi danymi

- interfejs zewnętrzny - również przechowuje permanentnie dane. System informacyjny je odczytuje, lecz są one zarządzane z poziomu innej aplikacji.

- wejście - na podstawie bodźca zewnętrznego, informacja w pliku wewnętrznym jest uaktualniana

- wyjście - sposób prezentacji danych z plików i interfejsów zewnętrznych.

- zapytanie - proste wejście/wyjście

Wyjście≠Zapytanie?

- **Wyjście**: raport, ekran, komunikat o błędzie. Pojedyncze dane w raporcie nie są liczone osobno.
- **Zapytanie**: bezpośrednie wej. skutkujące bezpośrednim wyj. Zapytanie nie może modyfikować żadnego pliku wewnętrznego (stanu).

Większość z tych elementów można intuicyjnie pojąć. Jedyny problem może się pojawić z rozróżnieniem wyjścia i zapytania.

Generalnie: zapytanie to bardzo prosta forma wejścia/wyjścia. Nie może ono modyfikować stanu systemu (żadnego pliku wewnętrznego), jest to pojedyncze wejście, skutkujące prostym wyjściem. Przykładowym zapytaniem jest wyświetlenie danych o statusie przesyłek w firmie kurierskiej: wchodzimy na stronę internetową, gdzie podajemy jedynie numer przesyłki, a system w odpowiedzi daje nam listę wszystkich czynności związanych z przesyłką (np. odebranie od nadawcy, dostarczenie do adresata, itp.)

Metoda punktów funkcyjnych

Typ fun.	Proste	Średnie	Złożone	Razem
Wejście	x 3	x 4	x 6	
Wyjście	x 4	x 5	x 7	
Zapytanie	x 3	x 4	x 6	
Pliki wewn.	x 7	x 10	x 15	
Interf. zewn.	x 5	x 7	x 10	
		Wstępne oszacowanie		

Problem: proste, średnie, czy złożone?

Czytając specyfikację wymagań tworzonego systemu należy zidentyfikować poszczególne funkcje oraz zaklasyfikować do jednej z trzech kategorii:

-proste

-średnie

-złożone

O szczegółowych kryteriach dotyczących przypisania poszczególnych funkcji do kategorii można przeczytać w literaturze.

Różne typy funkcji każdej kategorii otrzymują pewną liczbę punktów (szczegóły w tabeli).

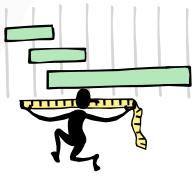
Jedyne co należy zrobić, to przemnożyć liczbę funkcji danej kategorii, przez liczbę punktów oraz podsumować.

Metoda punktów funkcyjnych

Typ fun.	Proste	Średnie	Złożone	Razem
Wejście	2 x 3	2 x 4	2 x 6	26
Wyjście	3 x 4	3 x 5	3 x 7	48
Zapytanie	2 x 3	1 x 4	0 x 6	10
Pliki wewn.	2 x 7	1 x 10	0 x 15	24
Interf. zewn.	0 x 5	1 x 7	0 x 10	7
		Wstępne oszacowanie		115

W rezultacie otrzymujemy liczbę punktów wstępnego oszacowania.

Metoda punktów funkcyjnych



FP = UT * CM

- FP – Punkty funkcyjne (*Function points*)
- UT – Wstępne oszacowanie (*Unadjusted total*)
- CM – Mnożnik złożoności (*Complexity multiplier*): 0.65 .. 1.35
- CM = 0.65 + 0.01 * Σ Współczynniki_wpływu (Influence_factors)
- 14 współczynników wpływu, 0 - 5 punktów każdy

Punkty wstępnego oszacowania musimy następnie przetworzyć na punkty funkcyjne. W tym celu należy należy punkty z wstępnego oszacowania przemnożyć przez mnożnik złożoności (CM) , który jest przeskalowaną sumą współczynników wpływu.

Współczynniki skalujące we wzorze:

CM=0.65 + 0.01 * Σ Współczynniki_wpływu

Czyli: 0.65 i 0.01 zostały dobrane na podstawie badań statystycznych przez twórców metody.

Twórcy wyróżnili również 14 współczynników wpływu, z których każdy jest punktowany od 0 – 5 punktów.

Metoda punktów funkcyjnych

- Ocena współczynników wpływu
  - 0 – Brak wpływu
  - 1 – Bardzo słaby
  - 2 – Raczej słaby
  - 3 – Średni
  - 4 – Istotny
  - 5 – Zasadniczy

Dla każdego współczynnika:

0 - oznacza całkowity brak wpływu, natomiast

5 - zasadniczy wpływ tego współczynnika.

Metoda punktów funkcyjnych

1. Czy jest wymagane **przesyłanie danych**?
2. Czy są funkcje **przetwarzania rozproszonego**?
3. Czy **wydajność** ma kluczowe znaczenie?
4. Czy system ma działać w **mocno obciążonym środowisku operacyjnym**?
5. Czy system wymaga **wprowadzania danych on-line**?
6. Czy wewnętrzne **przetwarzanie jest złożone**?
7. Czy kod ma być **re-używalny**?

Lista wszystkich 14 współczynników jest zaprezentowana na kolejnych slajdach.

Warto zwrócić uwagę na ich różnorodność:

- Czy jest wymagane przesyłanie danych?
- Czy są funkcje przetwarzania rozproszonego?
- Czy wydajność ma kluczowe znaczenie? - jest to więc pytanie z natury wymagań pozafunkcjonalnych
- Czy system ma działać w mocno obciążonym środowisku operacyjnym?
- Czy system wymaga wprowadzenia danych on-line?
- Czy wewnętrzne przetwarzanie jest złożone?
- Czy kod ma być re-używalny? – to pytanie dotyczy bardziej kodu i przyjmowanej architektury.

Przykładowe wartości współczynników (dla aplikacji internetowych):

1. Czy wymagane jest przesyłanie danych? - Aplikacja, która pozwala na modyfikację plików wewnętrznych poprzez Internet otrzymuje wagę 5. Aplikacja, która tylko przesyła parametry kontrolne otrzymuje wagę 3.
2. Czy są funkcje przetwarzania rozproszonego? - Aplikacja odczytująca dane za pośrednictwem Internetu otrzymuje 3, natomiast taka, która również zapisuje dane otrzymuje 4.
3. Czy wydajność ma kluczowe znaczenie? - Aplikacja internetowa wymaga pewnego poziomu wydajności, ze względu na interakcyjną pracę z użytkownikami, więc otrzymuje 3.

5. Czy system wymaga wprowadzania danych on-line? - W aplikacji internetowej umożliwiającej wprowadzanie danych współczynnik ten powinien wynieść minimum 4.

Metoda punktów funkcyjnych

8. Czy wejścia, wyjścia, pliki i zapytania są **złożone**?
9. Czy wprowadzanie danych on-line wymaga **transakcji** obejmujących **wiele ekranów** lub operacji?
10. Czy **pliki** główne są **aktualizowane on-line**?
11. Czy system ma mieć automatyczne konwersje i **instalacje**?

Kolejne współczynniki:

-Czy wejścia, wyjścia, pliki i zapytania są złożone?

-Czy wprowadzanie danych on-line wymaga transakcji obejmujących wiele ekranów lub operacji?

-Czy pliki główne są aktualizowane on-line?

-Czy system ma mieć automatyczne konwersje i instalacje? - pytanie o coś, co jest jakby „poza” tworzoną funkcjonalnością, lecz należy to uwzględnić w rozmiarze oprogramowania

Przykładowe wartości dla aplikacji internetowych:

11. Czy system ma mieć automatyczne konwersje i instalacje? - Aplikacje internetowe nie wymagają najczęściej wyrafinowanych instalatorów (są instalowane jednokrotnie na serwerze), tak więc ten współczynnik może wynosić 1-2

Metoda punktów funkcyjnych

12. Czy system wymaga mechanizmu **kopii zapasowych i odtwarzania**?
13. Czy system jest projektowany dla **wielu instalacji** w różnych organizacjach?
14. Czy aplikacja jest projektowana aby wspomagać **zmiany i być łatwą w użyciu** przez użytkownika?

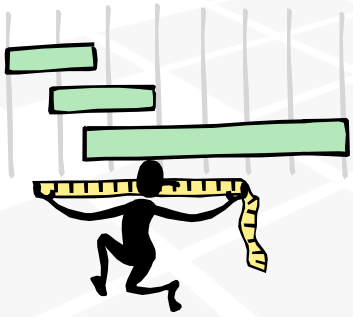
•Czy system wymaga mechanizmu kopii zapasowych i odtwarzania?

•Czy system jest projektowany dla wielu instalacji w różnych organizacjach?

•w końcu pytanie o użyteczność oprogramowania: „Czy aplikacja jest projektowana aby wspomagać zmiany i być łatwą w użyciu przez użytkownika?”

Tak więc mamy wiele pytań dotyczących wielu aspektów tworzonego systemu informatycznego. Każdy z tych aspektów na pewno ma wpływ na wielkość systemu, więc należy je wszystkie uwzględnić w metodzie szacującej rozmiar oprogramowania.

Metoda punktów funkcyjnych



Język	LOC/FP
Język asemblera	320
C	128
Cobol	105
Fortran	105
Pascal	90
C++/Java	53
Arkusze kalkulacyjne	6

Otrzymaną liczbę punktów funkcyjnych możemy następnie przemnożyć przez pewien współczynnik aby uzyskać liczbę linii kodu (LOC) tej aplikacji.

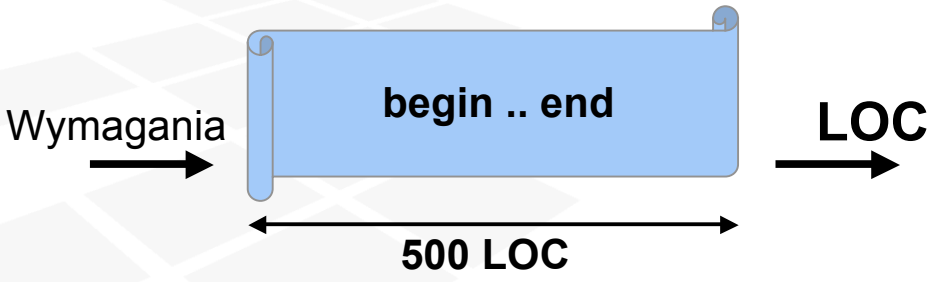
Liczba linii kodu mocno zależy od użytego języka programowania, przykładowo język asemblera potrzebuje 320 linii kodu na 1 punkt funkcyjny.

Jeżeli jakąś funkcjonalność można uzyskać tworząc szablon arkusza kalkulacyjnego, wtedy potrzebujemy jedynie 6 linii kodu/1 punkt funkcyjny.

Natomiast współczesne języki obiektowe potrzebują 53 linii kodu/ 1 punkt funkcyjny.

Oczywiście liczba linii kodu nie przekłada się bezpośrednio na pracochłonność systemu. Z jednej strony dlatego, iż jedną linię kodu pisze się z różną szybkością w różnych językach programowania (przykładowo szybciej jest napisać linię w assemblerze niż w C++). Z drugiej strony zaś, nawet w przypadku jednego języka programowania nie jest to zależność liniowa, dlatego potrzebujemy kolejnych metod, które wyliczą nam pracochłonność na podstawie liczby linii kodu. Wspomnimy o tych metodach w dalszej części wykładu, natomiast na ostatnim wykładzie Zaawansowanej inżynierii oprogramowania metoda taka będzie dokładniej przedstawiona.

Metoda punktów funkcyjnych



Szacowanie rozmiaru

W rezultacie dzięki metodzie punktów funkcyjnych udało nam się na podstawie wymagań systemu (liczonych w liczbie poszczególnych funkcji użytkownika) uzyskać szacowaną liczbę linii kodu rozpatrywanej aplikacji.

Wykorzystanie metody punktów funkcyjnych

- Wczesne szacowanie kosztów i budżetu
- Pomiary wydajności programistów
- Lepsza kontrola projektu
- Komunikacja
- Mierzenie jakości

Zastosowania metody punktów funkcyjnych są różnorakie. Pierwsze zastosowanie jakie przychodzi do głowy, to szacowanie rozmiaru, które jest dalej wykorzystywane do oszacowania kosztów projektu. Dzięki metodzie punktów funkcyjnych można to zrobić już podczas fazy analizy wymagań.

Po drugie, punkty funkcyjne stanowią obiektywną miarę rozmiaru, tak więc łatwiej na tej podstawie mierzyć wydajność poszczególnych programistów.

Rozmiar można w stosunkowo prosty sposób przekształcić na pracochłonność zadania, a dzięki temu możemy sprawować lepszą kontrolę nad postępami projektu.

Metoda punktów funkcyjnych wykonywana przez niezależnych ekspertów powinna dać te same rezultaty dla tych samych wymagań. Jeżeli jest inaczej, to znaczy, że eksperci nie zrozumieli w wymagań ten sam sposób. Dzięki wykryciu takich różnic można zatem wskazać fragmenty specyfikacji wymagań, które nie są do końca jasne.

Rozmiar oprogramowania może też stanowić część mierzenia jakości oprogramowania, wtedy można mówić np. o liczbie defektów/LOC, lub /PF

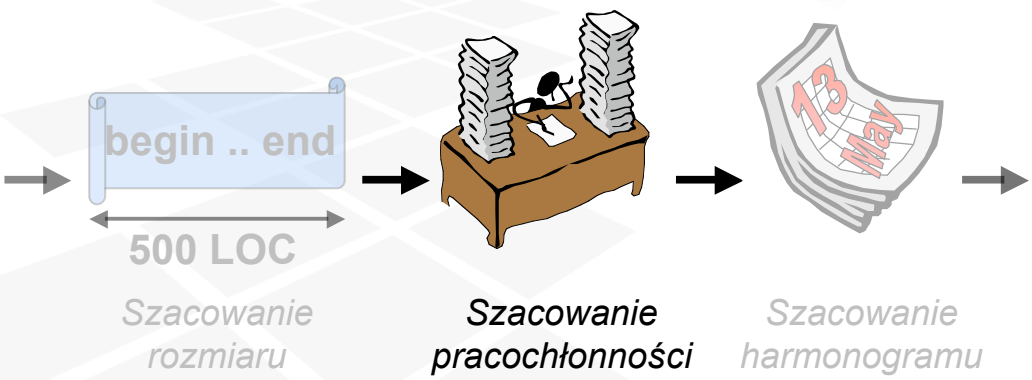
UCZELNIA ONLINE

Plan wykładu

- Systematyczne podejście do planowania
  - Szacowanie rozmiaru:
    - Metoda punktów funkcyjnych
  - **Szacowanie pracochłonności**
  - Planowanie, a kalendarz
- Wykorzystanie szacowania rozmiaru:
  - Śledzenie postępów prac
- Podsumowanie

Czas na poznanie dalszych elementów systematycznego podejścia do planowania.

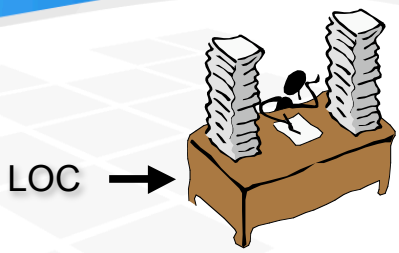
Systematyczne podejście do planowania



Szacowanie rozmiaru | **Szacowanie pracochłonności** | Szacowanie harmonogramu

Najpierw spójrzmy na szacowanie pracochłonności, które jest drugim krokiem systematycznego podejścia do planowania.

Szacowanie pracochłonności



LOC → Osobogodziny

Szacowanie pracochłonności

- COCOMO/COCOMO II

Parametrem wejściowym tej czynności jest rozmiar oprogramowania, wyrażany najczęściej w liczbie linii kodu (czasem również punktach funkcyjnych). Na wyjściu otrzymujemy liczbę osobogodzin potrzebnych do wykonania projektu.

Jedną z najbardziej znanych metod szacowania pracochłonności jest metoda COCOMO/COCOMO II, która zostanie omówiona na kolejnym wykładzie.

Na potrzeby tego wykładu możemy założyć, że czynność ta to czarna skrzynka, która przetwarza liczbę linii kodu na liczbę osobogodzin, uwzględniając szereg aspektów, które mogą mieć wpływ, np.: wielkość zespołu, doświadczenie zespołu, charakter klienta, itp.

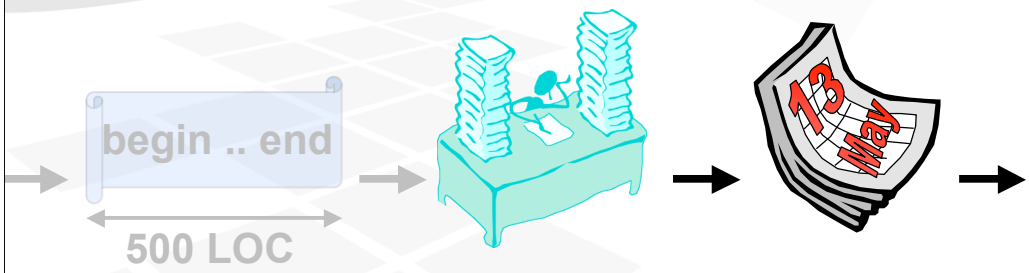
Plan wykładu

- Systematyczne podejście do planowania
  - Szacowanie rozmiaru:
    - Metoda punktów funkcyjnych
  - Szacowanie pracochłonności
  - **Planowanie, a kalendarz**

- Wykorzystanie szacowania rozmiaru:
  - Śledzenie postępów prac
- Podsumowanie

I ostatni element systematycznego podejścia do planowania…

Planowanie, a kalendarz

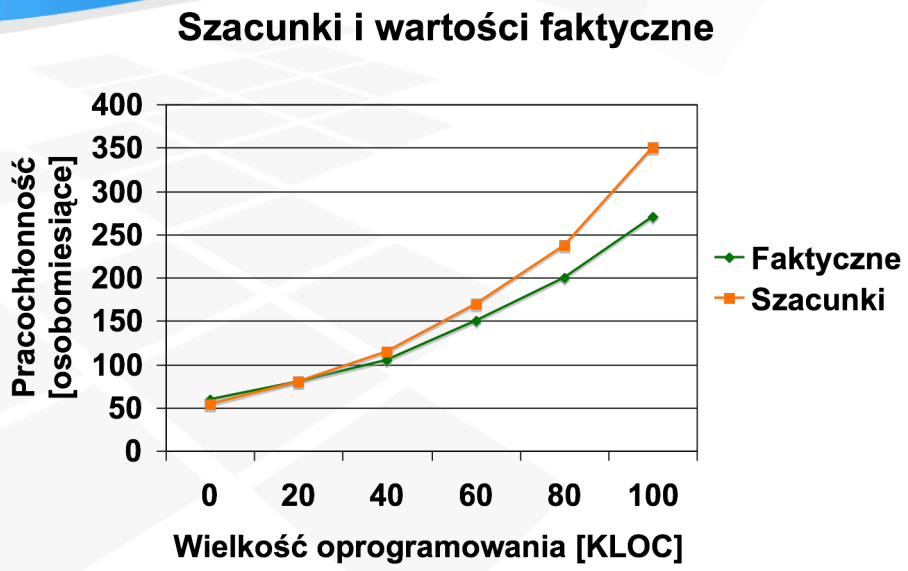


Szacowanie rozmiaru | *Szacowanie pracochłonności* | ***Szacowanie harmonogramu***

… to szacowanie harmonogramu. Zatrzymamy się nad nim na chwilę, gdyż mimo iż jest to najprostszy, jest najbardziej zaniedbywanym elementem.

Tutaj parametrem wejściowym jest liczba osobogodzin, natomiast na wyjściu otrzymujemy harmonogram prac.

Planowanie, a kalendarz

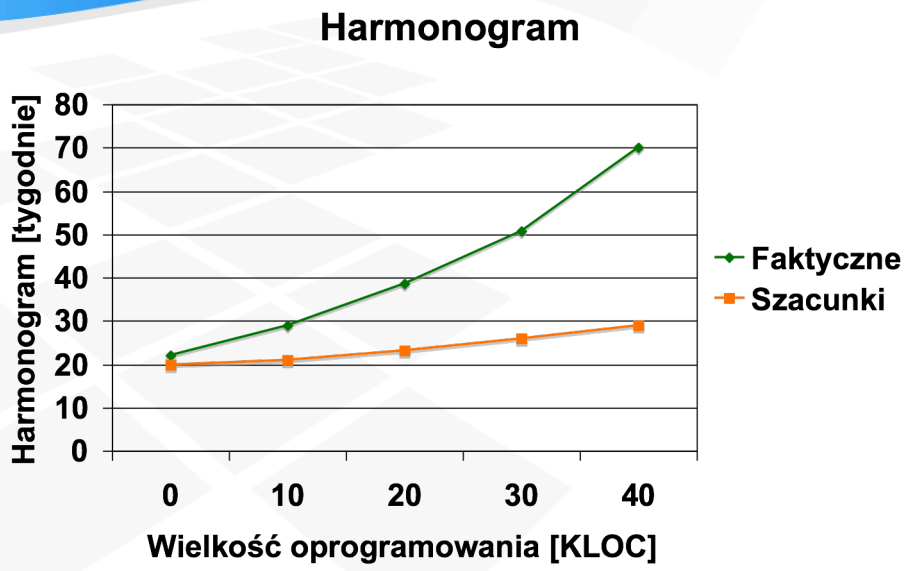


źródło: Mahil Carr: Software Effort and Schedule Estimation: A Case Study

Czynność ta wydawałaby się prosta. Niestety w praktyce nastręcza często wielu problemów.

Wykres zaprezentowany na slajdzie pokazuje skuteczność metod szacowania pracochłonności oprogramowania (drugi krok w systematycznym podejściu do planowania) w pewnej firmie konsultingowej. Okazuje się, że szacunki są bardzo dobre - jedynie przy bardzo małych projektach występuje lekkie niedoszacowanie rozmiaru, natomiast przy większych obserwujemy lekkie przeszacowanie (nie jest to złe, gdyż dzięki temu pozostaje pewna rezerwa na nieprzewidziane problemy).

Planowanie, a kalendarz



źródło: Mahil Carr: Software Effort and Schedule Estimation: A Case Study

Takie dobre szacunki pracochłonności stanowią parametr wejściowy dla trzeciego kroku - szacowania harmonogramu. Niestety tutaj obserwujemy dosyć duże problemy (wykres). Jak widać faktyczne wartości tym razem są dużo większe od szacowanych, przez co projekty mogły być bardzo zagrożone. Z czego mogła wynikać taka duża rozbieżność w harmonogramie, przy założeniu tak dobrych szacunków pracochłonności?

Planowanie, a kalendarz

- Problemy z klientem:
  - Testowanie akceptacyjne
  - Dostępność klienta
- Problemy z personelem:
  - Trudności ze skupieniem się na zadaniu
  - Współczynnik dostępności (*ang. Availability Factor*)
  - Rotacja pracowników
- Problemy infrastrukturalne:
  - Awarie sieci
  - Wolne kanały komunikacyjne

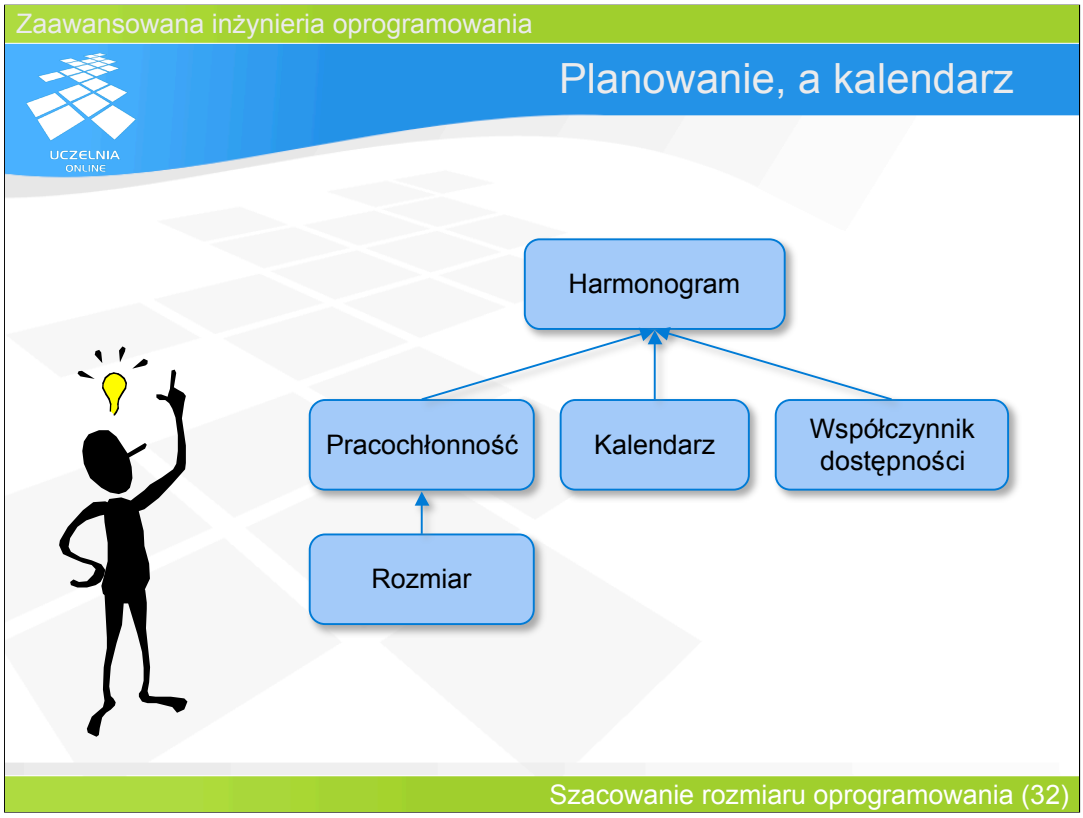
Po pierwsze - przyczyną wielu problemów jest klient. Menadżerowie projektu często zakładają bezproblemową pracę z klientem. Po czym okazuje się, że projekt musi czekać parę tygodni na testy akceptacyjne, bo klient jest zajęty ważniejszymi sprawami. Również są problemy z dostępnością klienta, w momencie kiedy pojawiają się pytania/wątpliwości w trakcie implementacji projektu.

Po drugie - często zakładamy bezproblemową pracę członków zespołu. Okazuje się natomiast, że:

-pracownicy nie są w stanie spędzić 8h dziennie tylko na pracy - część czasu zabiera im korespondencja, przygotowywanie herbaty, czy też rozmowy niemerytoryczne z innymi osobami. W dzisiejszych czasach powszechnego Internetu oraz komunikatorów jest to jeszcze trudniejsze. Dlatego warto mierzyć w firmie (dla każdego pracownika osobno) tzw. Współczynnik Dostępności - procent czasu pracy, który pracownik poświęca na wykonywanie zadań związanych z projektem. Współczynnik ten można następnie uwzględnić podczas przygotowywania harmonogramu.

-w niektórych firmach rotacja pracowników jest bardzo duża - jeżeli stracimy jednego pracownika, a jesteśmy zmuszeni zatrudnić kolejnego okazuje się, że zmarnujemy dużo czasu - aby tego nowego pracownika wdrożyć

Po trzecie - czasem (na szczęście coraz rzadziej) zdarzają się problemy infrastrukturalne, które blokują prace, przykładowo: awarie sieci energetycznej, lub problemy z dostępem do Internetu.



Aby poprawić ten stan rzeczy, podczas szacowaniu harmonogramu należy uwzględnić następujące elementy:

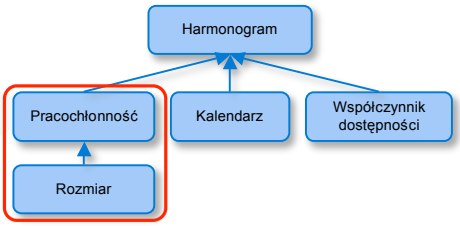
-pracochłonność zadań – to, co otrzymaliśmy z drugiego kroku systematycznego podejścia do planowania

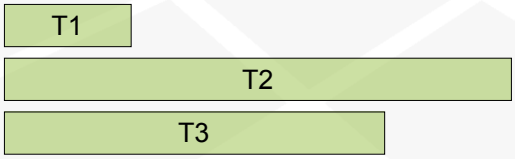
-kalendarz poszczególnych pracowników (urlopy, inne projekty)

-współczynnik dostępności

Kalendarz - przykład

- Krok 1:
  - Oszacuj wielkość i pracochłonność zadań:
    - Zadanie T1: 10h
    - Zadanie T2: 40h
    - Zadanie T3: 30h





Najlepiej zobaczyć to na przykładzie.

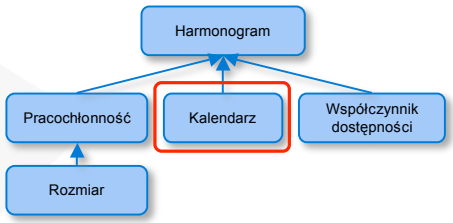
Załóżmy, że mamy 3 zadania do wykonania. W pierwszym kroku szacujemy pracochłonność tych zadań (chociażby na podstawie wcześniejszych metod). Otrzymujemy następujące wyniki:

- Zadanie T1: 10h
- Zadanie T2: 40h
- Zadanie T3: 30h

Kalendarz - przykład

- Krok 2:
  - Sprawdź dostępny czas pracownika

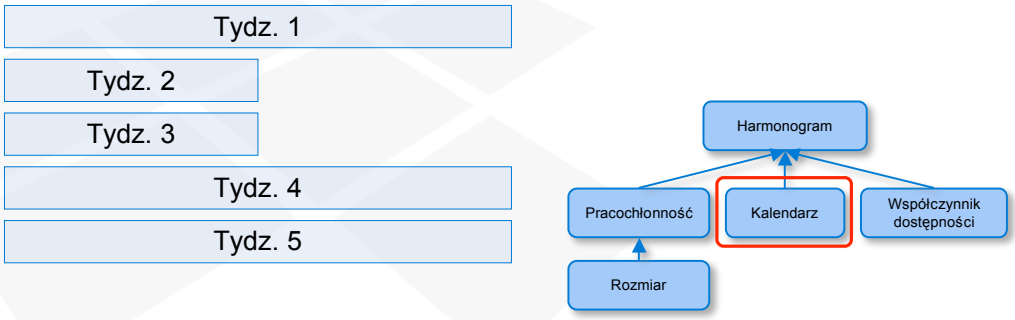
Tydz. 1
Tydz. 2
Tydz. 3
Tydz. 4
Tydz. 5



Następnie sprawdzamy kalendarz pracownika.

Kalendarz - przykład

- Krok 2:
  - Sprawdź dostępny czas pracownika
    - odejmij urlopy, czas zabrany przez inne projekty

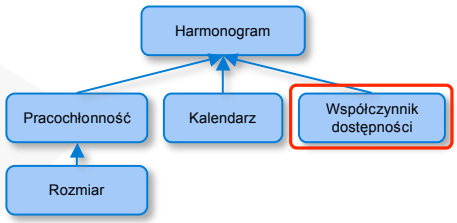


Okazuje się, że pracownik ten nie ma żadnych urlopów w ciągu najbliższych kilku tygodni, lecz w tygodniu 2 i 3 połowę swojego czasu musi poświęcić na inny projekt.

Kalendarz - przykład

- Krok 3:
  - pomnóż przez współczynnik dostępności:
    - przykładowo 75%

Tydz. 1
Tydz. 2
Tydz. 3
Tydz. 4
Tydz. 5

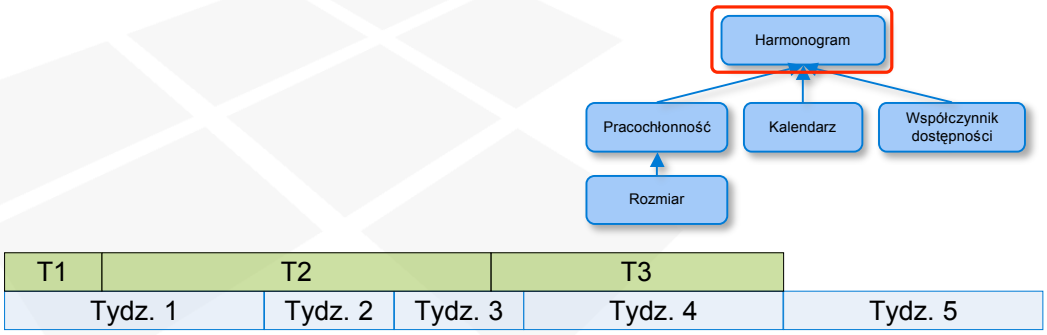


Następnie liczbę godzin pracy w poszczególnych tygodniach mnożymy przez współczynnik dostępności. Dla naszego przykładowego pracownika wynosi on 75%.

Jak widać na slajdzie, słupki obrazujące ilość czasu w kolejnych tygodniach zmniejszyły nam się o 25%.

Kalendarz - przykład

- Krok 4:
  - nałóż zadania na tygodnie, aby otrzymać harmonogram
    - w rezultacie: harmonogram o długości **4 tygodni**



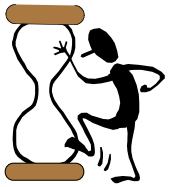
Mając te trzy elementy można przystąpić do przygotowania harmonogramu. W tym celu wystarczy nałożyć zadania na godziny dostępne w poszczególnych tygodniach.

Można to zrobić na jednej, wspólnej osi czasu, czyli uszeregować, w podobny sposób jak na slajdzie.

W przykładzie okazało się, że do wykonania trzech zadań potrzebne będą 4 tygodnie, mimo iż sumaryczny czas zadań wynosi 80h, czyli teoretycznie 2 tygodnie.

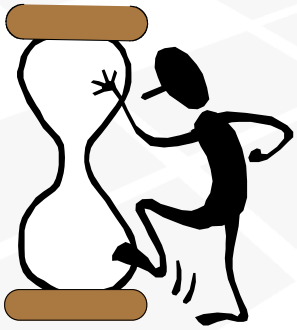
Plan wykładu

- Systematyczne podejście do planowania
  - Szacowanie rozmiaru:
    - Metoda punktów funkcyjnych
  - Szacowanie pracochłonności
  - Planowanie, a kalendarz
- Wykorzystanie szacowania rozmiaru:
  - **Śledzenie postępów prac**
- Podsumowanie



W momencie kiedy mamy opracowany harmonogram oraz szacunki czasu trwania poszczególnych zadań, możemy przejść do śledzenia postępów prac w projekcie.

Śledzenie postępu prac



- **Metoda wartości zarobionej (*ang. EV - Earned Value*)**:
  - zadania mają przypisaną pewną liczbę punktów
  - punkty odzwierciedlają pracochłonność i są znormalizowane do 1000 punktów
  - aby zarobić punkty związane z zadaniem należy zakończyć to zadanie

Jedną z najpowszechniejszych metod jest metoda wartości zarobionej. Jest to bardzo prosta metoda, która doczekała się obecnie wielu wariantów. Na wykładzie przedstawimy jedynie najprostszy wariant.

W metodzie wartości zarobionej:

-każde zadanie ma przypisaną pewną liczbę punktów

-punkty odzwierciedlają pracochłonność i są znormalizowane do 1000 punktów

-aby zarobić punkty, należy w całości wykonać to zadanie

Metoda wartości zarobionej

- 3 zadania:
  - T1 - 3h
  - T2 - 5h
  - T3 - 2h

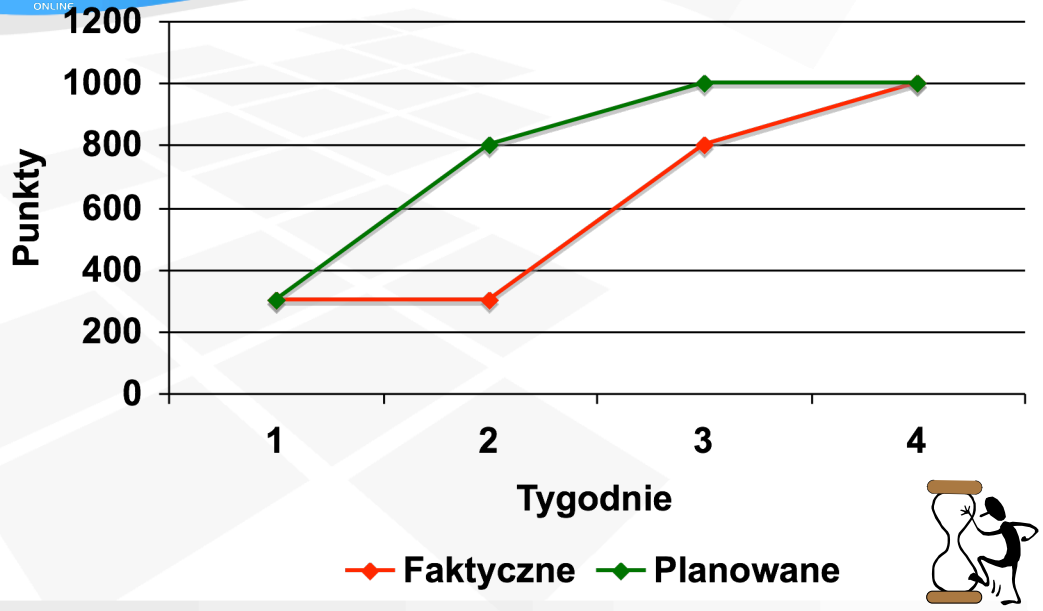
$$WZ_i = \frac{czas_i}{\sum_j czas_j} \cdot 1000$$

- Wartości zarobione:
  - WZ_{T1}=3h/10h*1000 = 300
  - WZ_{T2}=500
  - WZ_{T3}=200

Liczbę punktów otrzymujemy na podstawie pokazanego wzoru: szacunkowy czas trwania zadania dzielimy przez sumę czasów trwania wszystkich zadań oraz mnożymy razy 1000 – czyli wartość zarobiona zadania stanowi ułamek, jaki to zadanie stanowi w całym projekcie pomnożony x1000.

Przykładowo, mając dane zadania: 3h, 5h i 2h, otrzymamy: 300, 500 i 200 punktów.

Metoda wartości zarobionej

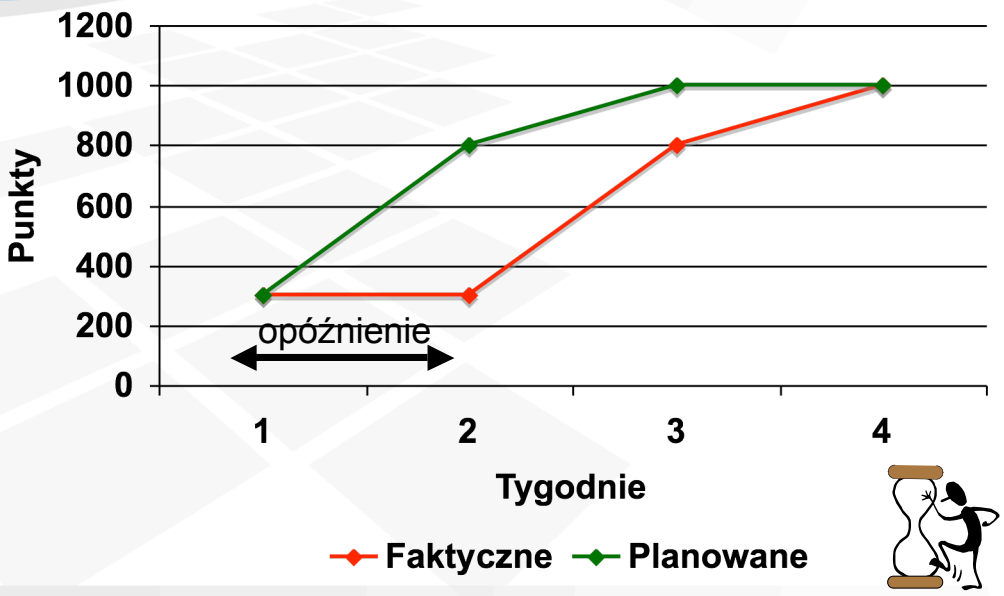


Na etapie tworzenia harmonogramu przygotowujemy wykres liczby zarobionych punktów na podstawie harmonogramu projektu. Na slajdzie jest to linia zielona. Stanowi ona linię bazową, odnośnik do wykresu aktualnych punktów zarobionych.

W trakcie wykonywania projektu na ten wykres nanosimy punkty faktycznie zarobione w danym okresie czasu.

Najlepiej jest, gdy te dwie linie nam się pokrywają. Znaczy to, że projekt idzie zgodnie z planem. Często jednak tak nie jest…

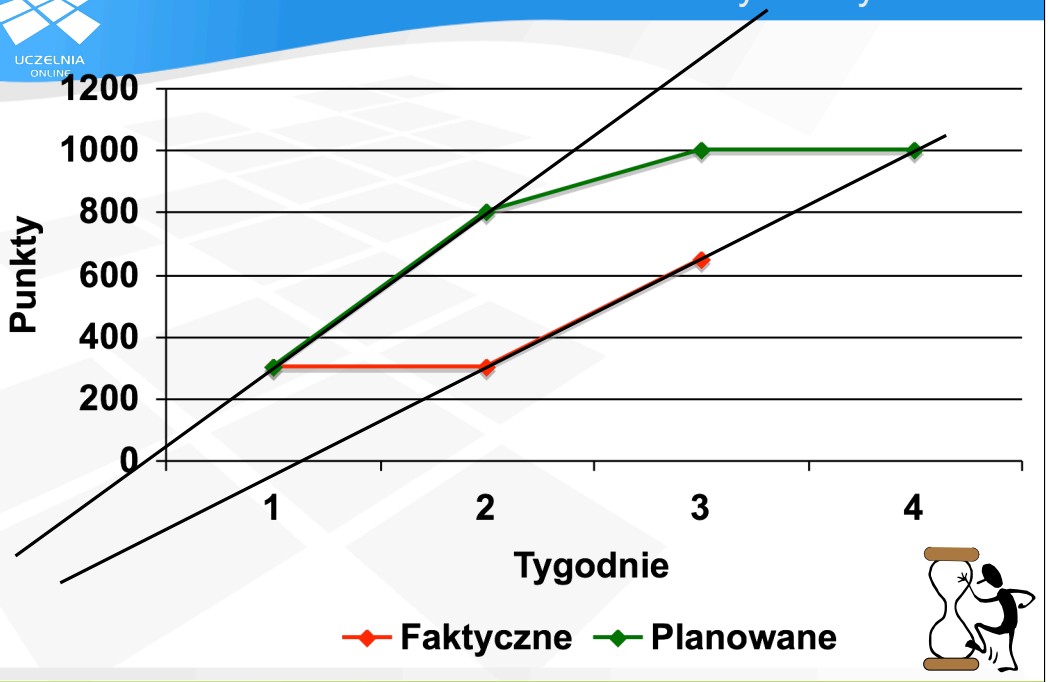
Jak czytać wykres?



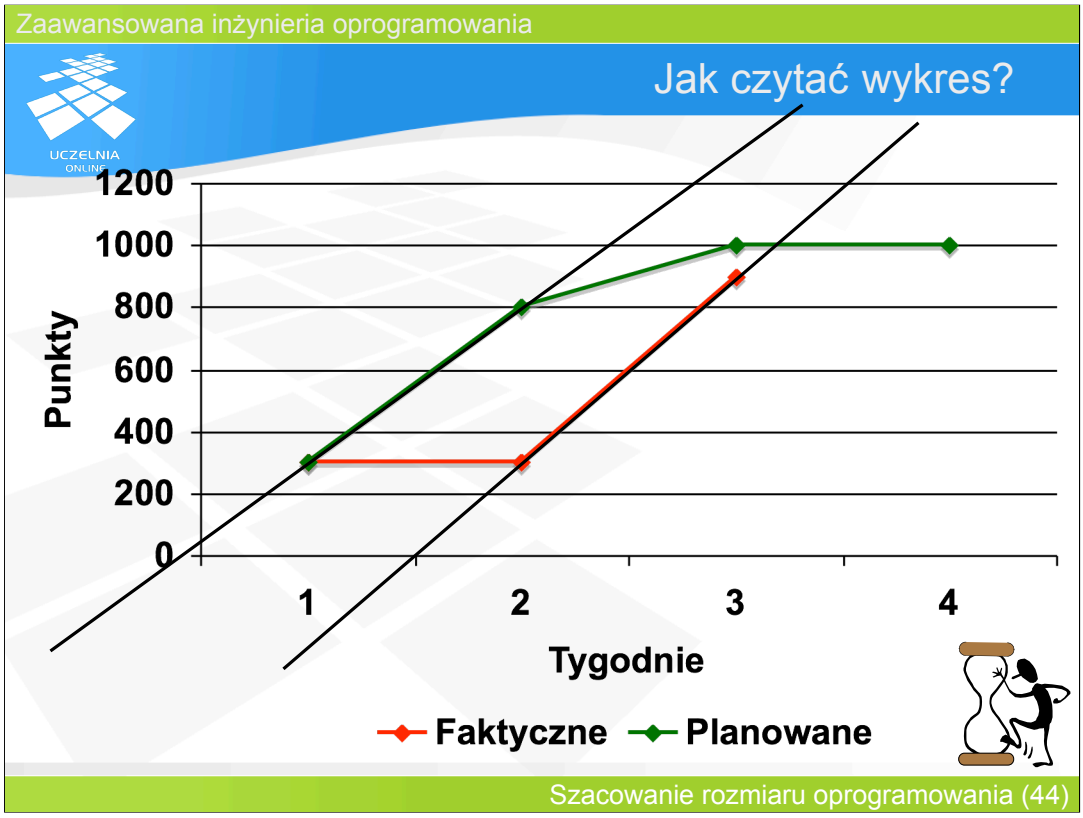
Jeżeli linia czerwona jest przesunięta na prawo od linii zielonej, oznacza to opóźnienie w projekcie. Wielkość tego opóźnienia można w bardzo prosty sposób zmierzyć, patrząc na odległość pomiędzy odpowiadającymi sobie punktami. Ponieważ oś pozioma jest osią czasu, więc odległość ta również będzie czasem.

Na przykładowym wykresie widzimy opóźnienie tygodniowe.

Jak czytać wykres?



To jeszcze nie wszystkie informacje, jakie możemy wyciągnąć z takiego wykresu. Można również przeanalizować nachylenie poszczególnych odcinków. Jeżeli proste przeprowadzone przez odpowiadające sobie odcinki przecinają się u dołu wykresu, to możemy się spodziewać zwiększenia się opóźnienia w projekcie – jest to sytuacja groźna dla każdego projektu.



Natomiast jeżeli proste te przecinają się u góry, to prawdopodobnie opóźnienie w projekcie zmniejszy się.

Podsumowanie

- Szacowanie rozmiaru to pierwszy krok systematycznego podejścia do planowania
- Szacowanie rozmiaru pomaga:
  - Wycenić kontrakt
  - Mierzyć postęp prac
  - Kontrolować wydajność pracowników
- Nie ma sposobów na dokładne oszacowanie rozmiaru

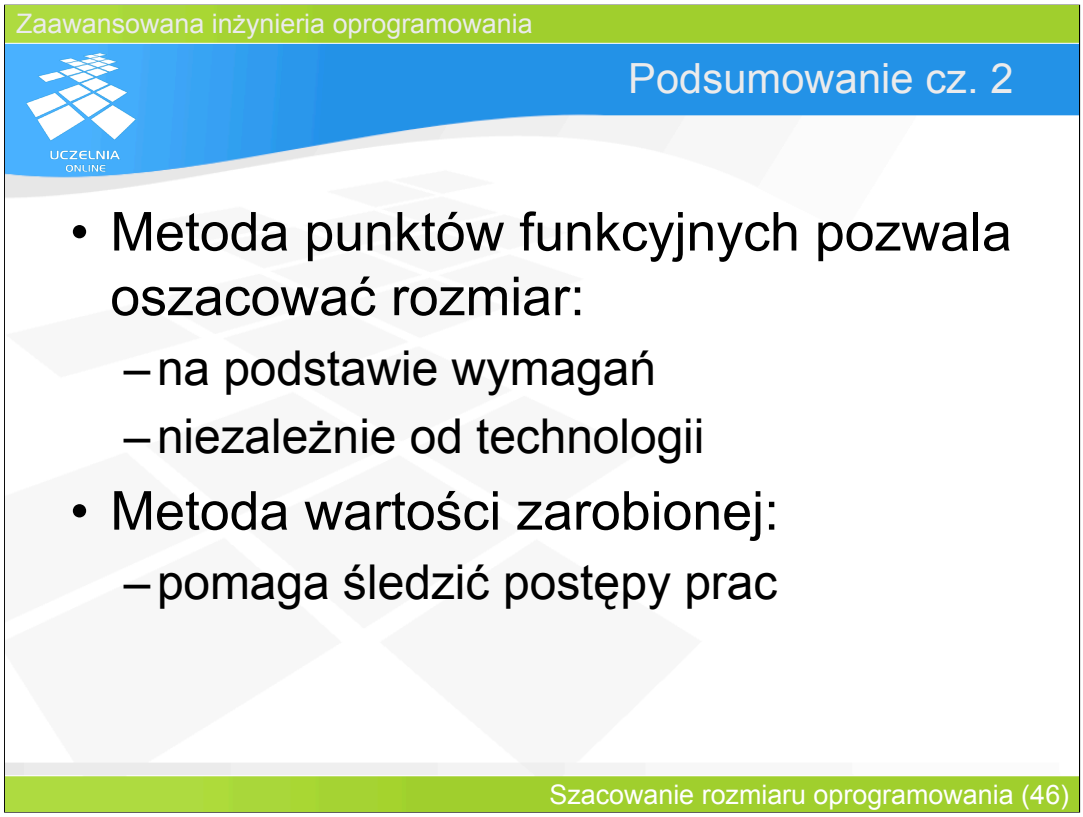
Podsumowując:

-planując projekt warto zrobić to w sposób systematyczny (szacowanie rozmiaru, pracochłonności i harmonogramu)

-szacowanie rozmiaru to pierwszy krok systematycznego podejścia do planowania

-dzięki oszacowaniu rozmiaru możemy lepiej wycenić kontrakt, mierzyć postęp prac, kontrolować wydajność pracowników

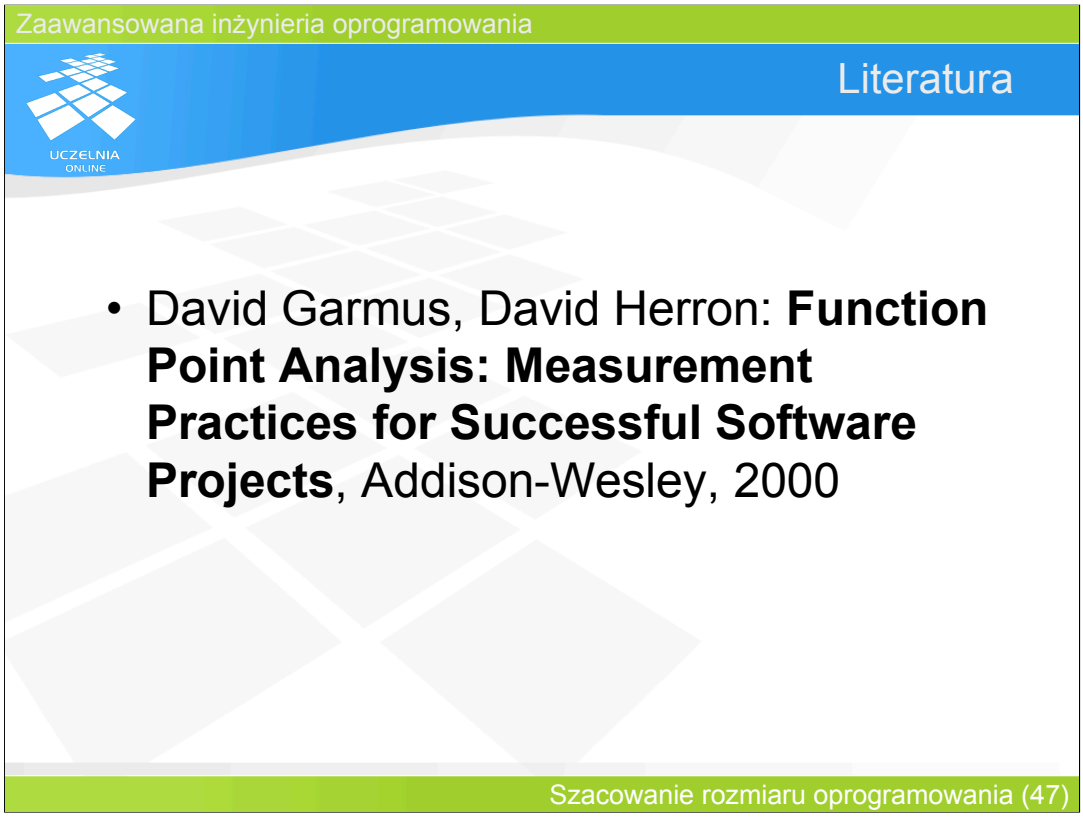
Nie ma idealnej metody szacowania rozmiaru - każda pozwala jedynie na oszacowanie z pewnym prawdopodobieństwem, tak więc może się okazać, iż faktyczne wartości będą się różnić.



Metoda punktów funkcyjnych pozwala na szacowanie rozmiaru projektu na podstawie wymagań.

Mając daną pracochłonność poszczególnych zadań, możemy w prosty sposób śledzić postępy prac za pomocą metody wartości zarobionej.

Jednak rzadko która polska firma informatyczna wykorzystuje metodę punktów funkcyjnych. Tłumaczą to faktem, iż muszą podać dokładny koszt projektu już na etapie przystępowania do przetargu, natomiast wtedy nie znają wymagań na tyle, żeby zastosować metodę. W rezultacie obserwujemy niestety podawanie szacunkowych kosztów z powietrza, a następnie realizowanie projektu, który wpasowuje się w te koszty. Na takim podejściu najbardziej traci jakość systemu. Niestety problem ten wynika również z naszego prawa, więc dopóki nie ulegnie ono zmianie, nie należy się spodziewać znacznego wzrostu wykorzystania tych metod w praktyce… A szkoda…



Osoby chętne zgłębić wiedzę przedstawioną w trakcie tego wykładu zachęcam do lektury książki: „Function Point Analysis: Measurement Practices for Successful Software Projects” autorstwa panów Garmusa i Herrona.

Dziękuję Państwu za uwagę!