

Opracowanie do egzaminu		
MPiSO	18 VI 2019	C2 s. 315 09:00

Czyli opracowanie na podstawie wykładów...



...które czasem się (nie) odbywały.

## Spis treści

---

<b>1</b>	<b>Źródła</b>	<b>2</b>
<b>2</b>	<b>Egzamin cz.1 - Praktyczna</b>	<b>3</b>
2.1	WBS . . . . .	3
2.1.1	WBS - Teoria . . . . .	3
2.1.2	WBS - Project Libre . . . . .	6
2.1.3	Słownik WBS - Teoria . . . . .	9
2.1.4	Słownik WBS - Project Libre . . . . .	9
2.1.5	WBS Project Libre - uwagi ogólne . . . . .	10
2.2	Harmonogram . . . . .	11
2.2.1	Harmonogram - Teoria . . . . .	11
2.2.2	Przypomnienie cykłów życia projektu . . . . .	11
2.2.3	Przypomnienie - uwaga ogólna . . . . .	13
2.2.4	Harmonogram - Teoria c.d. . . . .	13
2.2.5	Przypominajka relacji w harmonogramie . . . . .	14
2.2.6	Harmonogram - ProjectLibre . . . . .	15
2.2.7	Harmonogram - uwaga ogólna . . . . .	23
<b>3</b>	<b>Egzamin cz.2 - Ustna</b>	<b>24</b>
3.1	Wykład 1 - 1_mpiso_introduction_v0.1 . . . . .	24
3.2	Wykład 2 - 2_mpiso_wbs_v0.1 . . . . .	34
3.3	Wykład 3 - 3_mpiso_planowanie_v0.1 . . . . .	38
3.4	Wykład 4 - 4_mpiso_pert_delphi_v0.1 . . . . .	46
3.5	Wykład 5 - 5_mpiso_analogia_v0.1 . . . . .	53
3.6	Wykład 6 - 6_mpiso_cocomo_v0.1 . . . . .	66
3.7	Wykład 7 - 7_mpiso_functionalpoints_v0.1 . . . . .	81
3.8	Wykład 8 - mpiso_qualitymetrics_v0.1 . . . . .	86
3.9	Pytania dodatkowe wiki.stosowana opracowanie . . . . .	93

## 1 Źródła

---

Opracowanie na podstawie:

- Wykłady Mrówki
- Kazimierz Frączkowski - Zarządzanie projektem informatycznym - łatwo znaleźć w google pdf
- Project Management Body of Knowledge – an American National Standard - łatwo znaleźć w google pdf

## 2 Egzamin cz.1 - Praktyczna

---

### 2.1. WBS

Najpierw będzie wstęp teoretyczny na temat WBS, a potem tworzenie WBS w ProjectLibre.

#### 2.1.1. WBS - Teoria

Do czego służy WBS ?

W każdym projekcie Project Manager tworzy tzw. Work Breakdown Structure (WBS) (Struktura Podziału Prac). Jest to dokument, który ma głównie na celu przedstawić całą pracę jaka ma zostać wykonana w projekcie (jeśli czegoś nie ma w WBS, tzn. nie ma tego w projekcie), a także podzielić duże części projektu na małe zadania, którymi można zarządzać. (przypisanie po jednej osobie do każdego zadania, szacowanie czasu zadania itp.)

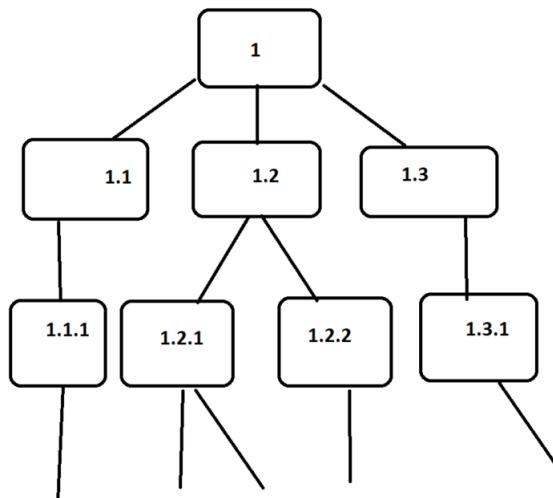
Jaka jest ogólna struktura WBS ?

WBS ma strukturę hierarchiczną, składa się z wielu poziomów. Poziomy te można przedstawić w postaci drzewa lub tabeli z numerowanymi poziomami. Zazwyczaj spotyka się ten drugi sposób i z niego korzysta się w ProjectLibre (na egzaminie). Każdy kolejny poziom musi zawierać pracę w 100 % równą pracy z poziomu do niego nadzawanego (tzw. reguła 100%).

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

Jak numerować poziomy w WBS w postaci tabeli ?

Poziomy dla WBS numerujemy oddzielając je kropką. Pierwszy poziom zawiera zawsze tylko jeden element z nazwą projektu i jest oznaczany numerem 1.



Rysunek 1: Reprezentacja numerowania poziomów dla WBS w postaci drzewa.

ID	WBS	Task Name
1	1 Projekt MegaPortal	
2	1.1 Zarządzanie projektem	
3	1.1.1 Zarządzanie jakością	
4	1.1.2 Planowanie zadań	
5	1.1.3 Kontrolowanie prac	
6	1.2 Analiza funkcjonalna	
7	1.2.1 Analiza procesów biznesowych	
8	1.2.2 Opracowanie specyfikacji wymagań	
9	1.2.3 Wymagania architektury	
10	1.3 Projekt techniczny	
11	1.3.1 Projekt architektury	
12	1.3.2 Projekt komponentów	
13	1.3.3 Projekt integracji	
14	1.3.4 Projekt szczegółowy	
15	1.4 Implementacja	
16	1.4.1 Implementacja warstwy prezentacji	
17	1.4.2 Implementacja warstwy biznesowej	
18	1.4.3 Implementacja warstwy bazy danych	
19	1.4.4 Integracja	
20	1.5 Testy oprogramowania	
21	1.5.1 Testy komponentów	
22	1.5.2 Testy integracyjne	
23	1.5.3 Testy wydajnościowe	
24	1.5.4 Testy end-end	
25	1.6 Dokumentacja	
26	1.6.1 Dokumentacja techniczna	
27	1.6.2 Dokumentacja użytkownika	
28	1.6.3 Dokumentacja powrócenia	
29	1.7 Wdrożenie	
30	1.7.1 Instalacja na środowisku testowym	
31	1.7.2 Instalacja na środowisku produkcyjnym	
32	1.7.3 Konwersja danych	
33	1.7.4 Uruchomienie systemu	

Rysunek 2: Reprezentacja numerowania poziomów dla WBS w postaci tabeli.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

---

Co powinno znajdować się na kolejnych poziomach WBS ? Rodzaje WBS.

- Poziom 1 - oznaczamy go numerem 1, na tym poziomie zawsze jest tylko jeden element, wpisujemy tutaj nazwę projektu
- Poziom 2 i kolejne... - zależy od rodzaju WBS.

To co będzie przedstawione na drugim i kolejnych poziomach WBS zależy od rodzaju WBS jaki tworzymy. Istnieją 3 rodzaje WBS (według tego co Mrówka mówił na wykładzie):

- produktowy (ten ma być podobno na egzaminie według Mrówki) - stanowi perspektywę produktów, najbardziej ogólnych komponentów, które mają być dostarczone
- fazowy - opiera się na modelu fazowym cyklu życia oprogramowania, na drugim poziomie znajdują się kolejne fazy cyklu życia oprogramowania: projektowanie, implementacja, testowanie itd., na następnym poziomie fazy te są dzielone na mniejsze aktywności lub produkty, które na następnym poziomie dekomponowane są na zadania
- mieszany = produktowy + fazowy - na pierwszym poziomie mogą znajdować się zarówno moduły produktów jak i fazy

Bardziej szczegółowo omówiony zostanie WBS produktowy, bo ten ma być na egzaminie.

Co powinno znajdować się na kolejnych poziomach dla WBS produktowego ?

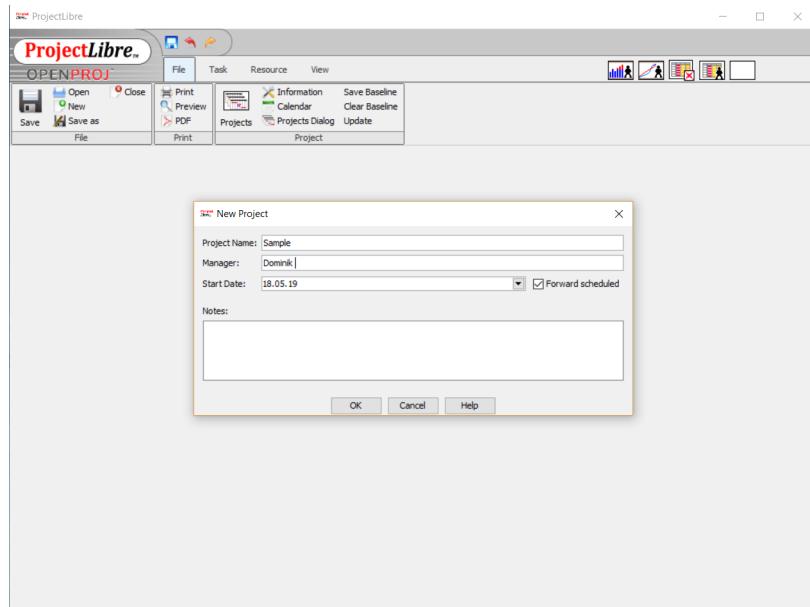
- Poziom 1 - Nazwa projektu
- Poziom 2 - Najważniejsze, ogólne komponenty projektu sformułowane w formie rzeczowników lub rzeczowników z przyniątkami
- Kolejne poziomy... - Liczba poziomów w WBS nie jest z góry ustalona, dekomponowanie należy zakończyć gdy osiągnięty zostanie żądany poziom szczegółowości określany przez ostatni poziom WBS. Na ostatnim poziomie znajdować się mają tzw. Work Package (Pakiety Pracy), czas wykonania jednego Work Package nie powinien przekraczać 2 tygodni (według wykładu, ale nie powinien być też zbyt krótki, np. 3 dni to za mało). Nie wszystkie dekompozycje muszą kończyć się na tym samym poziomie (według Project Management Body of Knowledge).
- Tak jak na poziomie 2, na kolejnych należy również formułować komponenty w formie rzeczowników dla WBS produktowego

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

### 2.1.2. WBS - Project Libre

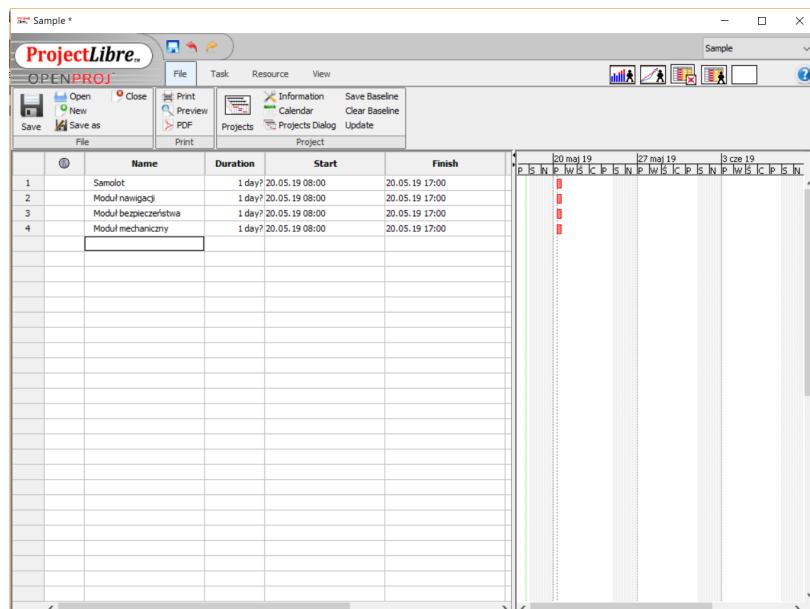
Tutaj omówione jest jak tworzymy WBS w ProjectLibre.

1. Otwieramy sobie ProjectLibre i tworzymy nowy projekt



Rysunek 3: Tworzenie nowego projekt w Project Libre.

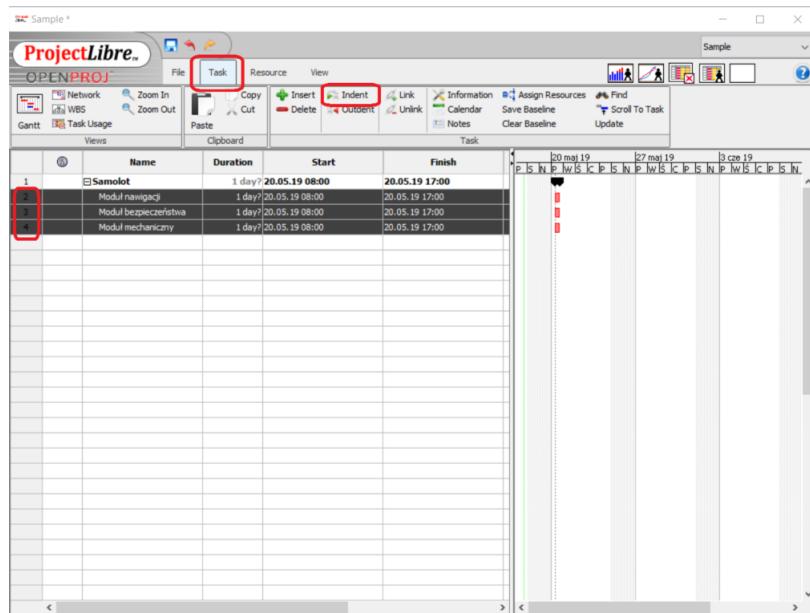
2. W kolumnie name wpisujemy kolejne części naszego WBS



Rysunek 4: Tworzenie WBS w Project Libre.

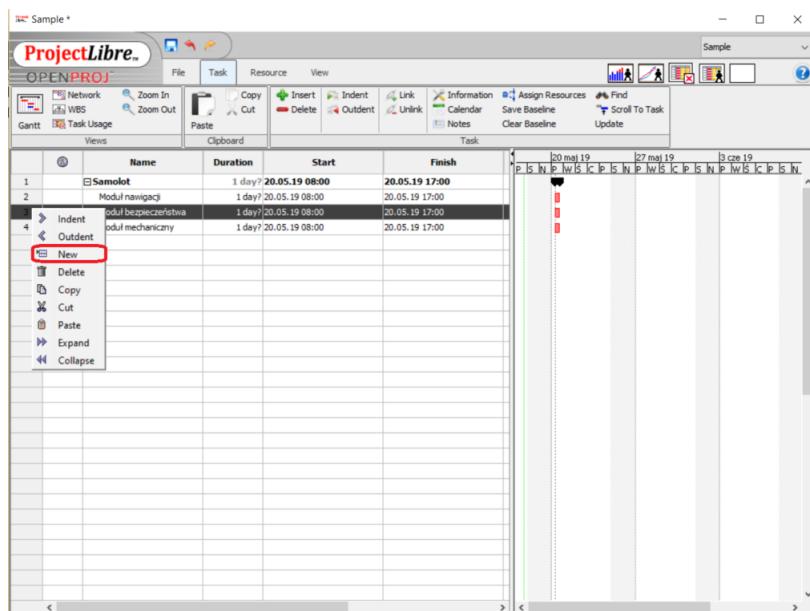
## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

3. Chcemy teraz wprowadzić poziomy do naszego WBS. Zaznaczamy po lewej stronie numery wierszy, które chcemy wciąć, przechodzimy do zakładki Task i naciskamy Indent



Rysunek 5: Tworzenie wcięć dla WBS.

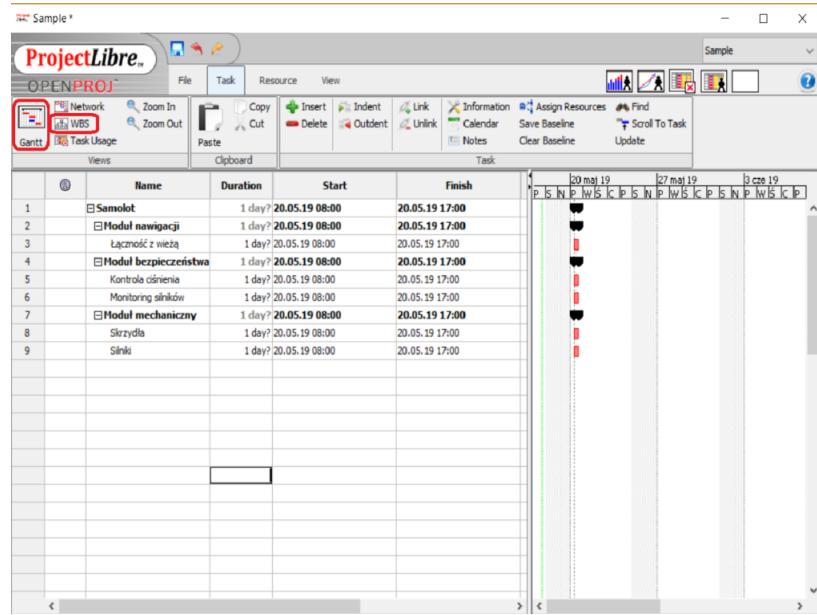
4. Jeśli teraz chcemy wprowadzić dodatkowy wiersz nad którymś z wierszy to klikamy na numer tego wiersza prawym i naciskamy New (zawsze można cofnąć CTRL+Z).



Rysunek 6: Dodawanie dodatkowego wiersza nad wybranym.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

5. Dla każdego kolejnego poziomu analogicznie dokonujemy wcięć. Widok drzewiasty naszego WBS można podglądać klikając WBS wracamy do widoku tabelkowego klikając Gantt



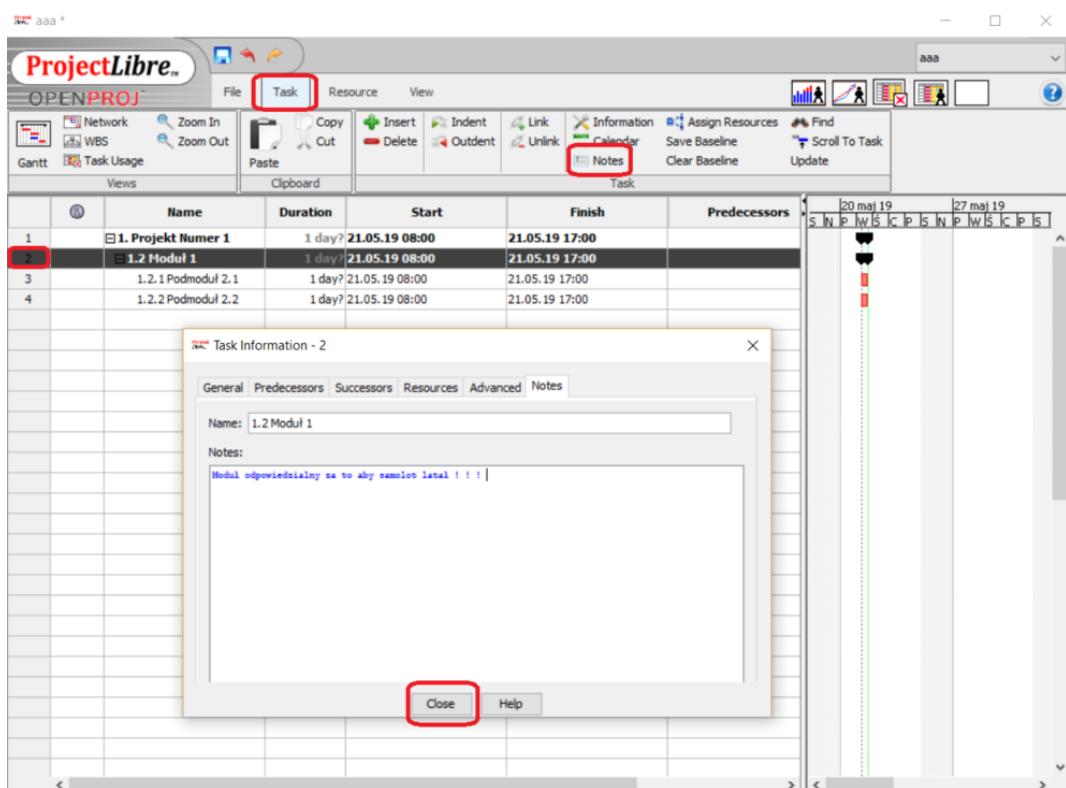
Rysunek 7: Różne możliwości wyświetlania tworzonego WBS.

### 2.1.3. Słownik WBS - Teoria

Słownik WBS to krótki opis każdej pozycji znajdującej się w WBS. W słowniku do pozycji odwołujemy się przez numery. Każda pozycja w WBS (nie tylko work package, ale wszystkie) powinna być opisana w 2-3 zdaniach.

### 2.1.4. Słownik WBS - Project Libre

Słownik WBS można stworzyć przypisując w ProjectLibre notatkę do każdej z pozycji WBS. Zaznaczamy numer wiersza do którego chcemy dodać notatkę i w zakładce task wybieramy Notes.



Rysunek 8: Tworzenie słownika WBS w ProjectLibre.

Wyskakuje okienko do wpisania notatki. Wpisujemy opis i zamykamy Close. Notatkę można teraz zobaczyć klikając ikonę notatki, która po zamknięciu pojawi się obok numeru wiersza. Jeśli chcemy usunąć notatkę, to usuwamy z niej cały tekst, wtedy sama znika.

	Name	Duration	Start	Finish	P
1	1. Projekt Numer 1	1 day?	21.05.19 08:00	21.05.19 17:00	
2	1.2 Moduł 1	1 day?	21.05.19 08:00	21.05.19 17:00	
3	1.2.1 Podmoduł 2.1	1 day?	21.05.19 08:00	21.05.19 17:00	
4	1.2.2 Podmoduł 2.2	1 day?	21.05.19 08:00	21.05.19 17:00	

Rysunek 9: Widok po dodaniu notatek do pozycji WBS.

### 2.1.5. WBS Project Libre - uwagi ogólne

- W Project Libre nie ma automatycznego numerowania jakie powinno być dla WBS, ale można je dodać ręcznie, wpisując przed każdą nazwą numer zadania zgodnie z numeracją WBS
- Na egzaminie (termin II 2018 z wiki.stosowana) opis produktu jest już podzielony i w zasadzie można tylko przepisywać do WBS to co w nim jest (?):

3. *Moduł Prospekty i leady* – moduł służy do identyfikacji potencjalnych okazji handlowych. Poszczególni pracownicy na bieżąco wprowadzają informacje o prowadzonych akcjach sprzedawczych. Na podstawie zaawansowania poszczególnych lead'ów, prezentowana jest prognoza sprzedaży na najbliższy okres rozliczeniowy z uwzględnieniem prawdopodobieństwa realizacji.
4. *Moduł Projekty* - narzędzie wspomagające zespołową realizację długoterminowych projektów. W ramach danego projektu pracownicy wyznaczeni do jego realizacji wymieniają informacje, delegują zadania, wprowadzają informacje o sprzedaży, zgłaszają absencje oraz generowane koszty itd. Moduł oferuje również mechanizmy workflow, przeglądu historii projektu oraz analizy zysków oraz poniesionych kosztów. W ramach modułu Projektu możemy wyszczególnić następujące funkcjonalności:
- Definiowanie projektów
  - Wzorce projektów
  - Definiowanie procesu projektu wraz z zadaniami

Rysunek 10: Fragment opisu systemu z egzaminu 2018.

- Ze starszych lat: Podobno patrzy głównie na harmonogram + zasoby, a WBS nawet czasem nie otwiera

## 2.2. Harmonogram

Najpierw będzie wstęp teoretyczny na temat harmonogramu, a potem tworzenie harmonogramu w ProjectLibre.

UWAGA: Dodać notkę o wzorze  $\frac{mandays}{duration}$

### 2.2.1. Harmonogram - Teoria

Harmonogram to określony w czasie porządek realizacji zadań w projekcie. Głównymi składowymi harmonogramu są zadania, zależności między nimi, czas trwania oraz alokacja zasobów do poszczególnych zadań.

Tworzenie harmonogramu składa się z kilku etapów, wejściem do całego procesu jest stworzony wcześniej WBS.

1. Stworzenie listy aktywności i zadań na podstawie WBS - Listę aktywności tworzymy na podstawie najniższego poziomu WBS, czyli na podstawie Work Package.

Work Package określa co ma być dostarczone w projekcie. Aktywność określa konkretne działania, które mają służyć do realizacji danego Work Package, czyli działanie, które jest konieczne do dostarczenia danej części projektu.

Do danego work package można przypisać jedno albo więcej aktywności.

Aktywności dalej mogą być dzielone na zadania, aktywność jest zbiorem zadań, w szczególności aktywność może być równa zadaniu (tak mówił Mrówka). Tak więc w naszym harmonogramie tak naprawdę będziemy wypisywać zadania, które w szczególnych przypadkach mogą być aktywnościami. Czym się różni aktywność od zadania... chyba tylko wymaganym nakładem prac, zadanie może być mniejsze.

2. Cykl życia projektu - Okej założmy, że mamy już zdefiniowane zadania na podstawie WBS i chcemy zrobić sobie harmonogram, jaka będzie jego struktura? Jaka będzie kolejność wykonywania poszczególnych zadań?

To zależy od cyklu życia projektu w którym działamy. Przykładowo dla waterfall będziemy dzielić nasz harmonogram na fazy takie jak m.in. projektowanie, testowanie, itd. Dla iteracyjnego na iteracje. W związku z tym najpierw sobie przypomnijmy cykle życia projektu. (Na pierwszym terminie w 2018 był w jednej grupie iteracyjny lub waterfall, a w drugiej iteracyjny spiralny, bez wyboru -> wiki.sotosowana)

### 2.2.2. Przypomnienie cykłów życia projektu

#### Waterfall (wodospadowy, klasyczny, liniowy)

Model w którym prace wykonywane są kolejno w następujących po sobie fazach. Każda z faz wykonywana jest tylko raz. Kolejna faza rozpoczyna się dopiero po zakończeniu poprzedniej. Jeśli mamy kilka modułów w projekcie, np. silnik i hamulce, to daną fazę (dajmy na to testowanie) można jeszcze rozbić na Silnik oraz Hamulce i opisać tam czynności związane z testowaniem silnika i hamulców. Typowe fazy w kolejności ich wykonywania to:

- Określenie wymagań - określane są ogólne oraz szczegółowe wymagania wobec projektu
- Analiza - budowa logicznego modelu systemu
- Projektowanie - powstaje szczegółowy projekt systemu

- Implementacja - rozwijanie funkcjonalności
- Testowanie - testowanie funkcjonalności
- Wdrożenie - uruchomienie aplikacji w środowisku produkcyjnym
- (Utrzymanie) - utrzymywanie oprogramowania podczas jego użytkowania

Nie wszystkie fazy muszą występować w każdym modelu.

Stosując ten cykl życia projektu, nasz harmonogram dzielimy na takie właśnie fazy, w każdej z nich definiujemy co ma być zrobione.

### **Iteracyjny (przyrostowy)**

Polega na iteracyjnym wykonywaniu wszystkich (lub wybranych) faz cyklu Waterfall. Każda iteracja prowadzi do pewnego rozszerzenia powstającego produktu. Na przykład, w pierwszym rozszerzeniu edytora tekstu mogą się znaleźć podstawowe mechanizmy obsługi plików, edytowanie i możliwość tworzenia dokumentów. Bardziej zaawansowane edytowanie i możliwości tworzenia dokumentów można umieścić w drugim rozszerzeniu, w trzecim - mechanizmy sprawdzania pisowni i poprawności gramatycznej, w czwartym zaawansowane możliwości projektowania wyglądu stron.

Stosując ten cykl życia produktu, nasz harmonogram dzielimy na iteracje (np. Iteracja 1 - podstawowe funkcjonalności, Iteracja 2 - Rozszerzenie komunikacji między serwisami itp.), każdą z Iteracji dzielimy z kolei na fazy Analiza, Projektowanie, Kodowanie, Testowanie. Jeśli mamy kilka modułów w projekcie, np. silnik i hamulce, to daną fazę (dajmy na to testowanie) można jeszcze rozbić na Silnik oraz Hamulce i opisać tam czynności związane z testowaniem silnika i hamulców.

### **Model oparty na prototypowaniu**

Stosowany jest gdy mamy do czynienia z projektem w którym klient nie jest pewny np. co do formatu danych wejściowych / wyjściowych albo gdy programista nie do końca wie czy zastosowane algorytmy będą działać itp. W tym cyklu życia wyróżniamy fazy":

- Uzyskanie informacji od klienta
- Przygotowanie i udoskonalanie prototypu
- Sprawdzenie prototypu przez klienta

Wykonywanie tych czynności iteracyjnie pozwala na "trafienie" w oczekiwania klienta. Po zaakceptowaniu prototypu przez klienta można przejść do implementacji właściwego systemu.

Stosując ten cykl życia projektu, nasz harmonogram dzielimy na fazy wymienione powyżej dla poszczególnych modułów, po zaakceptowaniu wersji przez klienta przechodzimy do implementacji.

### **Iteracyjny Spiralny**

W modelu spiralnym pewne podstawowe czynności są wykonywane cyklicznie. Zwykle rozważa się od 3 do 6 takich czynności. Są to kolejno:

- Kontakt z klientem - omówienie wymagań projektowych
- Planowanie - określenie potrzebnych zasobów, harmonogramów, kosztów związanych z przebiegiem prac w obecnej iteracji
- Analizowanie ryzyka - ocena ryzyka związanego z pracami technicznymi
- Inżynieria - Rozwijanie oprogramowania
- Konstruowanie i wdrożenie - testowanie i wdrożenie u klienta
- Ocena klienta - uzyskanie informacji zwrotnej od klienta

Stosując ten cykl życia projektu, nasz harmonogram dzielimy na fazy wymienione powyżej, jeśli mamy kilka modułów w projekcie, np. silnik i hamulce, to fazę inżynierii można jeszcze rozbić na Silnik oraz Hamulce, analogicznie można zrobić dla innych faz jeśli mamy wiele czynności do wykonania w nich dla poszczególnych modułów. Jeśli tych czynności nie jest dużo to można je wymieniać bezpośrednio pod daną fazą.

### 2.2.3. Przypomnienie - uwaga ogólna

- Jak widać w harmonogramie mogą występować również czynności charakterystyczne tylko dla danego cyklu życia oprogramowania, przykładowo dla spiralnego cyklu życia oprogramowania występuje analiza ryzyka, które nie jest obecna w innych cyklach oprogramowania.

### 2.2.4. Harmonogram - Teoria c.d.

Mamy już zadania i cykle życia, wiemy jak stworzyć ogólną strukturę harmonogramu oraz jak wpisać w nią zadania. Kolejną sprawą jest przypisywanie czasu do zadań.

Każdemu zadaniu przypisujemy czas wykonania w przyjętych jednostkach, mogą to być np. mandays (roboczodni) albo roboczogodziny (z tego korzysta się w ProjectLibre).

Przypisanie zadaniu czasu wykonania nie oznacza jednak, że będzie ono tyle trwało. Wpływ na czas trwania zadania mają przypisane do niego zasoby. Jeśli mamy przypisane do zadania programistę na pół etatu, to zadanie wykonane będzie wolniej niż gdyby był to programista na cały etat.

Rzeczywisty czas trwania zadania obliczany jest ze wzoru:

$$Duration = \frac{roboczogodziny}{przypisaneZasoby}$$

Przykładowo mając przypisane do zadania 40 roboczogodzin oraz 50% programisty, czas trwania zadania to:

$$Duration = \frac{40}{\frac{50}{100}} = 80roboczogodzin$$

Zasoby w projekcie określane są właśnie przy pomocy procentów. Np. programista[200%] oznacza, że mamy do dyspozycji dwóch programistów na pełny etat. Analityk[50%] oznacza, że mamy do dyspozycji jednego analityka na pół etatu.

Nie ma zatem sensu przypisywanie do zadania np. 85% programisty, bo co miało by to znaczyć ?

Jeśli chcemy sprawdzić czy poprawnie przypisane zostały w harmonogramie zadania i zasoby, to powinniśmy skorzystać ze wzoru (Mrówka mówił, że z tego wzoru sprawdza harmonogramy na egzaminie):

$$resources = \frac{work}{duration}$$

gdzie work oznacza czas jaki przypisaliśmy na trwanie zadania, a duration, to rzeczywisty czas trwania zadania (zależny od przypisanych zasobów).

Przykładowo:

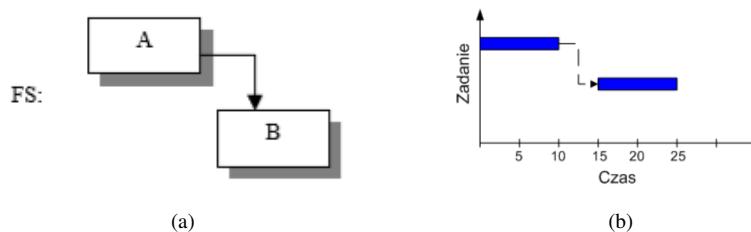
Jeśli w projekcie mamy np. 3,5 programisty, to chcemy z tego wzoru uzyskać wartość w przedziale mniej więcej 3-3,5. work będzie to suma czasów trwania zadań przypisanych do programistów, a duration to suma ich rzeczywistego czasu trwania.

Ostatnim elementem harmonogramu jest wprowadzenie relacji pomiędzy zadaniami.

### 2.2.5. Przypominajka relacji w harmonogramie

1. **Finish to start** Koniec – Start (ang. Finish-to-Start FS) – zadanie B nie może rozpoczęć się przed ukończeniem zadania A.

Oznaczenie na harmonogramie:



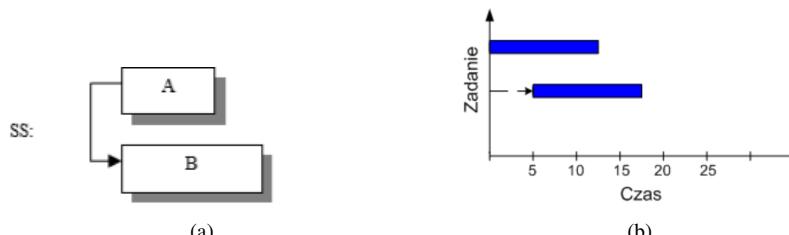
Rysunek 11: Finish to start

Przykłady z projektu informatycznego:

- Projekt interfejsu / prototyp interfejsu
- Napisanie user guide / wydrukowanie user guide
- Zebranie wymagań / podpisanie umowy dotyczącej wymagań

2. **Start to start** Start – Start (ang. Start-to-Start SS) – zadanie B nie może rozpoczęć się przed rozpoczęciem zadania A.

Oznaczenie na harmonogramie:



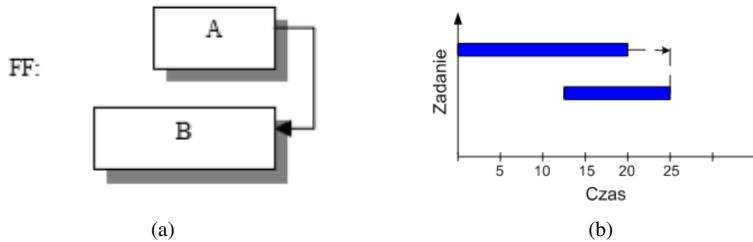
Rysunek 12: Start to start

Przykłady z projektu informatycznego:

- Pisanie dokumentu HTML / pisanie pliku CSS
- Uzyskanie danych z REST API / obróbka danych z REST API
- Pisanie kodu / Pisanie dokumentacji

3. **Finish to finish** Koniec – Koniec (ang. Finish-to-Finish FF) – zadanie B nie może zakończyć się dopóki nie zakończy się zadanie A.

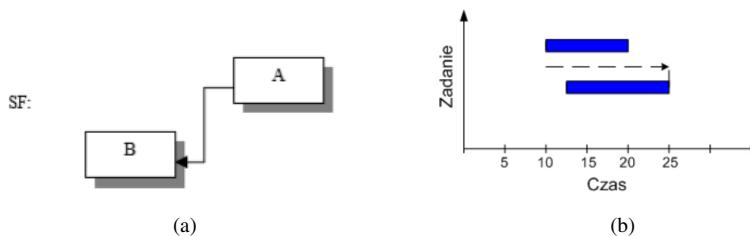
Oznaczenie na harmonogramie:



Rysunek 13: Finish to finish

Przykłady z projektu informatycznego:

- Napisanie kodu dla modułu X / testowanie jednostkowe modułu X
  - Lutowanie połączeń / Testowanie połączeń
  - Wdrożenie systemu / Zarządzanie projektem
4. **Start to finish** Start – Koniec (ang. Start-to-Finish SF) – zadanie B nie może zakończyć się dopóki nie rozpoczęcie się zadanie A.



Rysunek 14: Start to finish

Przykłady z projektu informatycznego:

- Wypuszczenie nowej wersji systemu / Likwidacja starego systemu
- Publiczna reklama systemu / Opracowanie strategii marketingowej
- Dostosowanie do potrzeb niepełnosprawnych / Otrzymanie certyfikatu bezpieczeństwa

#### 2.2.6. Harmonogram - ProjectLibre

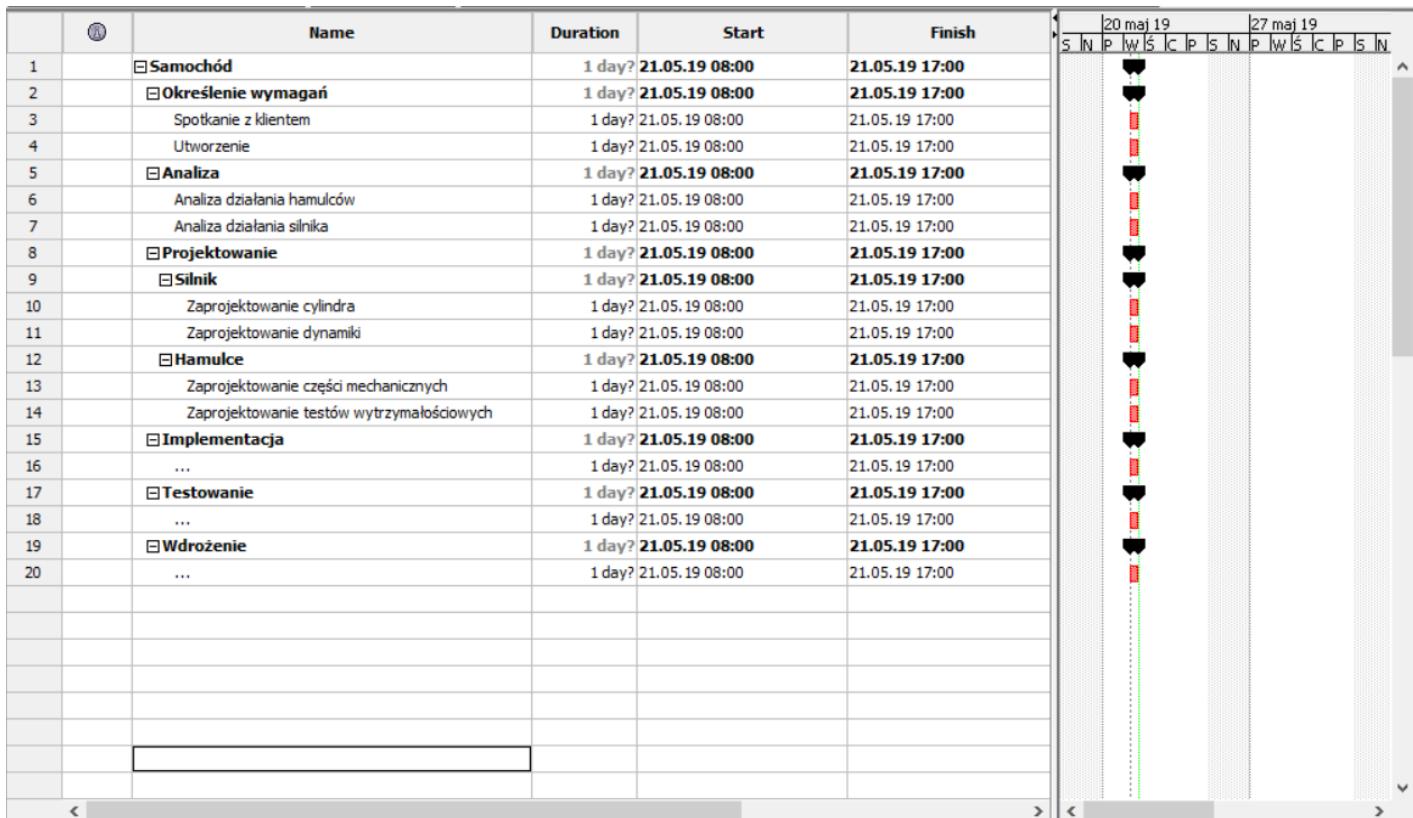
Dobrze jest przy tworzeniu harmonogramu w ProjectLibre zachować taką kolejność

- Wpisanie zadań do dokumentu
- Przypisanie zadaniom wymaganej pracy
- Wprowadzenie relacji pomiędzy zadaniami
- Przypisanie zasobów

W tej kolejności sobie to przedstawimy.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

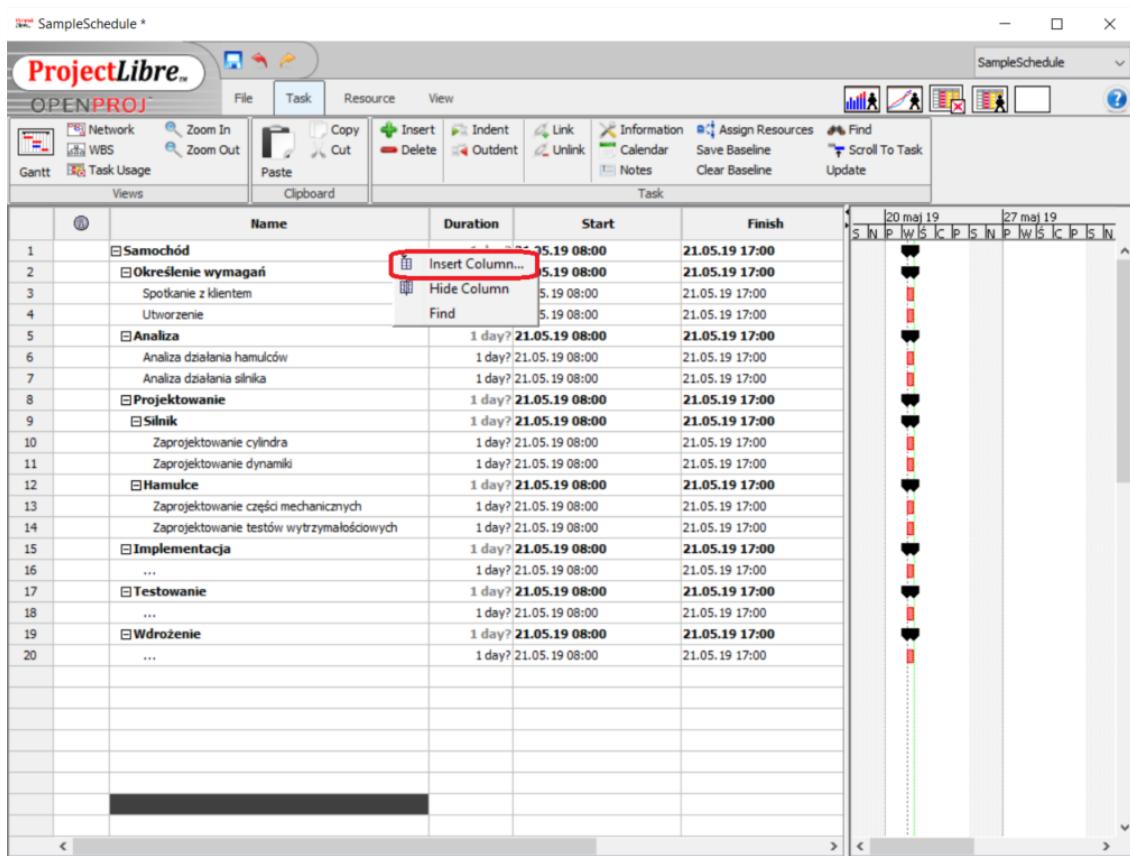
- Znów otwieramy nowy projekt i wpisujemy w kolumnie Name nasze zadania na podstawie WBS, pamiętamy o tym, że trzeba uwzględnić jaki mamy cykl projektu. Wcięcia robimy tak samo jak przy WBS. Przykładowo dla modelu waterfall moglibyśmy mieć taką strukturę:  
(Zauważmy jak projektowanie zostało rozdzielone na moduły).  
Pamiętajmy też, że tutaj nie formułujemy elementów jako rzeczniki - to nie WBS - tutaj korzystamy w większości z czasowników w różnej formie, bo przedstawiamy zadania.



Rysunek 15: Tworzenie harmonogramu w ProjectLibre.

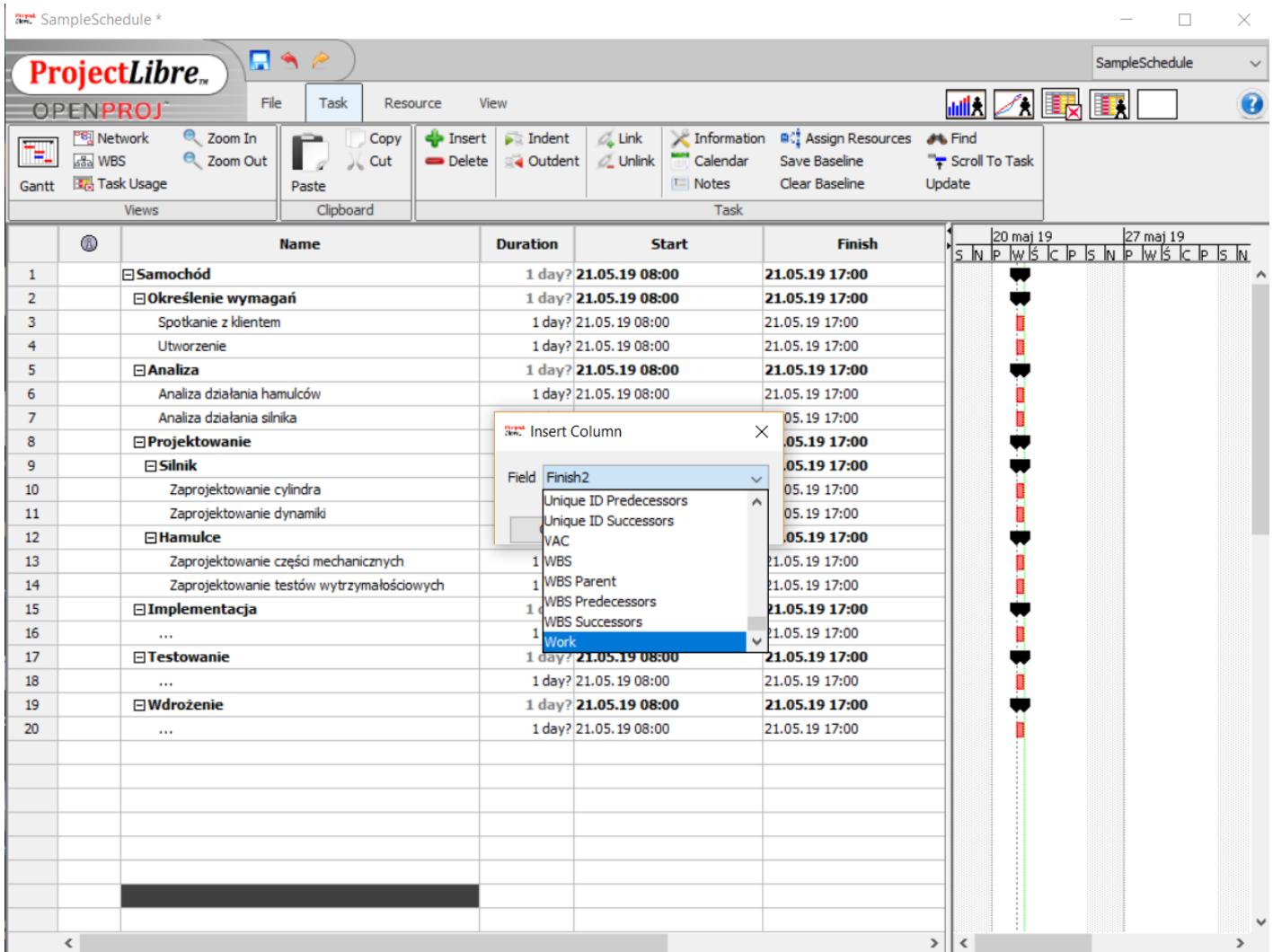
- Następnie chcemy dodać czas w roboczogodzinach. Nie jest to jednak kolumna Duration. Duration odnosi się do czasu uwzględniającego już zasoby. Musimy sami dodać kolumnę o nazwie Work. Kolumnę tą dodajemy klikając prawym klawiszem myszy w miejscu gdzie pokazane są wszystkie nazwy kolumn i wybierając Insert column.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA



Rysunek 16: Wstawianie kolumny w ProjectLibre.

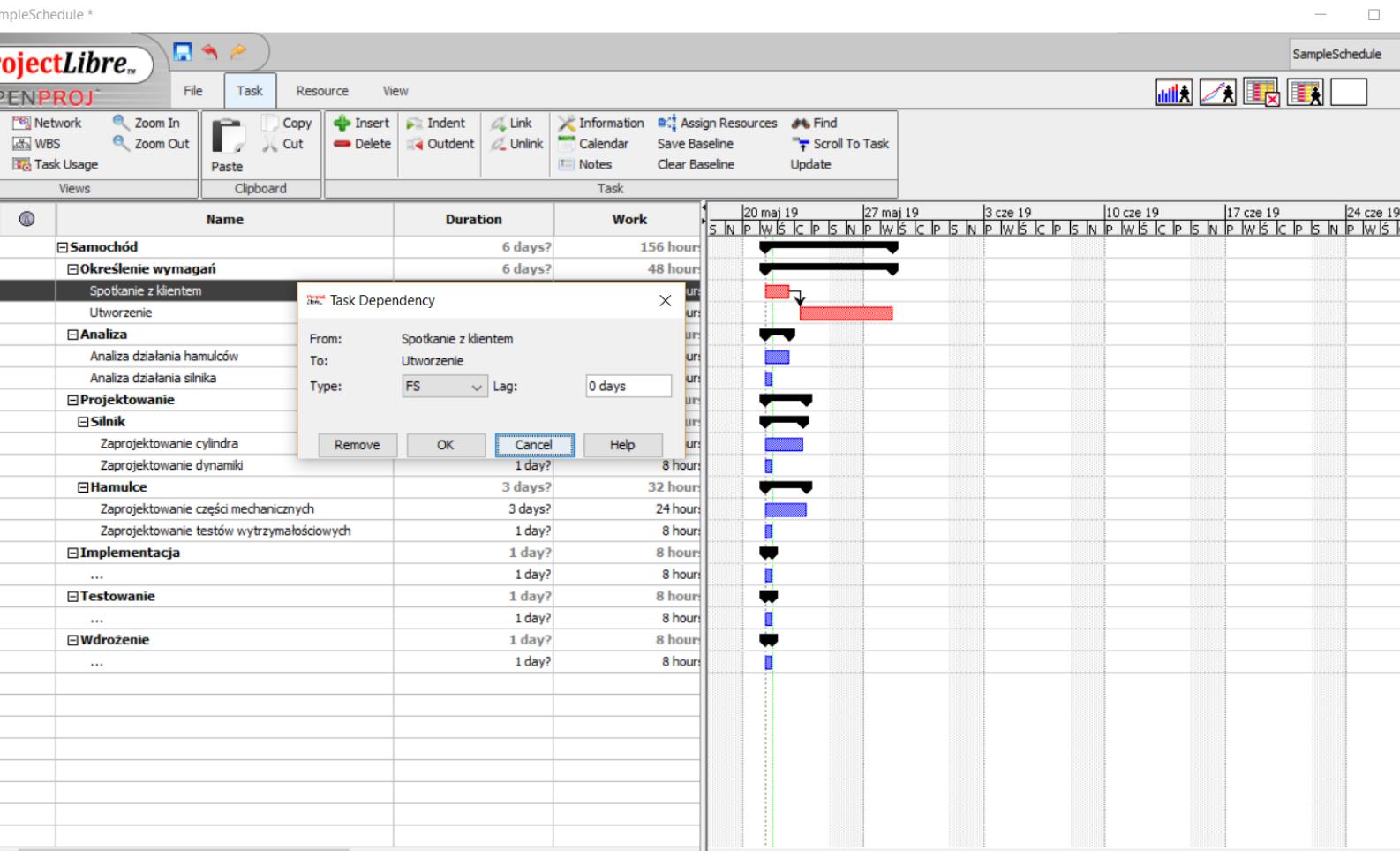
## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA



Rysunek 17: Wybieranie kolumny odpowiedzialnej za czas zadania.

3. Jak już mamy nową kolumnę to wpisujemy w niej czas jaki potrzebny jest na wykonanie zadania. Kolumna duration będzie pokazywać rzeczywisty czas trwania zadania w projekcie zależny od przypisanych mu zasobów. Domyślnie pokazuje tak jakby przypisane było 100% zasobu. Jeśli chcemy usunąć kolumnę, to klikamy na nią i zaznaczamy hide column. W kolumnie duration mamy na końcu pytajniki, można je usunąć klikając na komórkę, wybierając Information z menu w zakładce Task i odznaczając Estimated, ale dużo z tym zabawy więc niekoniecznie.
4. Jak już mamy przypisaną pracochłonność zadań to możemy ją zacząć łączyć relacjami. Aby stworzyć relacje pomiędzy dwoma zadaniami, znajdujemy je na harmonogramie po prawej, naciskamy na jeden z bloczków i trzymając przeciągamy do drugiego. Tworzy to deafultową relację F2S, aby zmienić relację klikamy na strzałkę i możemy wybrać z listy jaką relację chcemy.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA



Rysunek 18: Dodawanie relacji pomiędzy zadaniami.

Warto zwrócić uwagę, że relacje można tworzyć nie tylko pomiędzy dwoma elementami z najniższego poziomu. Przykładowo można by było stworzyć relację pomiędzy elementem Analiza i Projektowanie (czarne bloczki na harmonogramie), ma to sens bo Projektowanie może się rozpocząć dopiero po zakończeniu analizy więc relacja F2S jest tu na miejscu.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

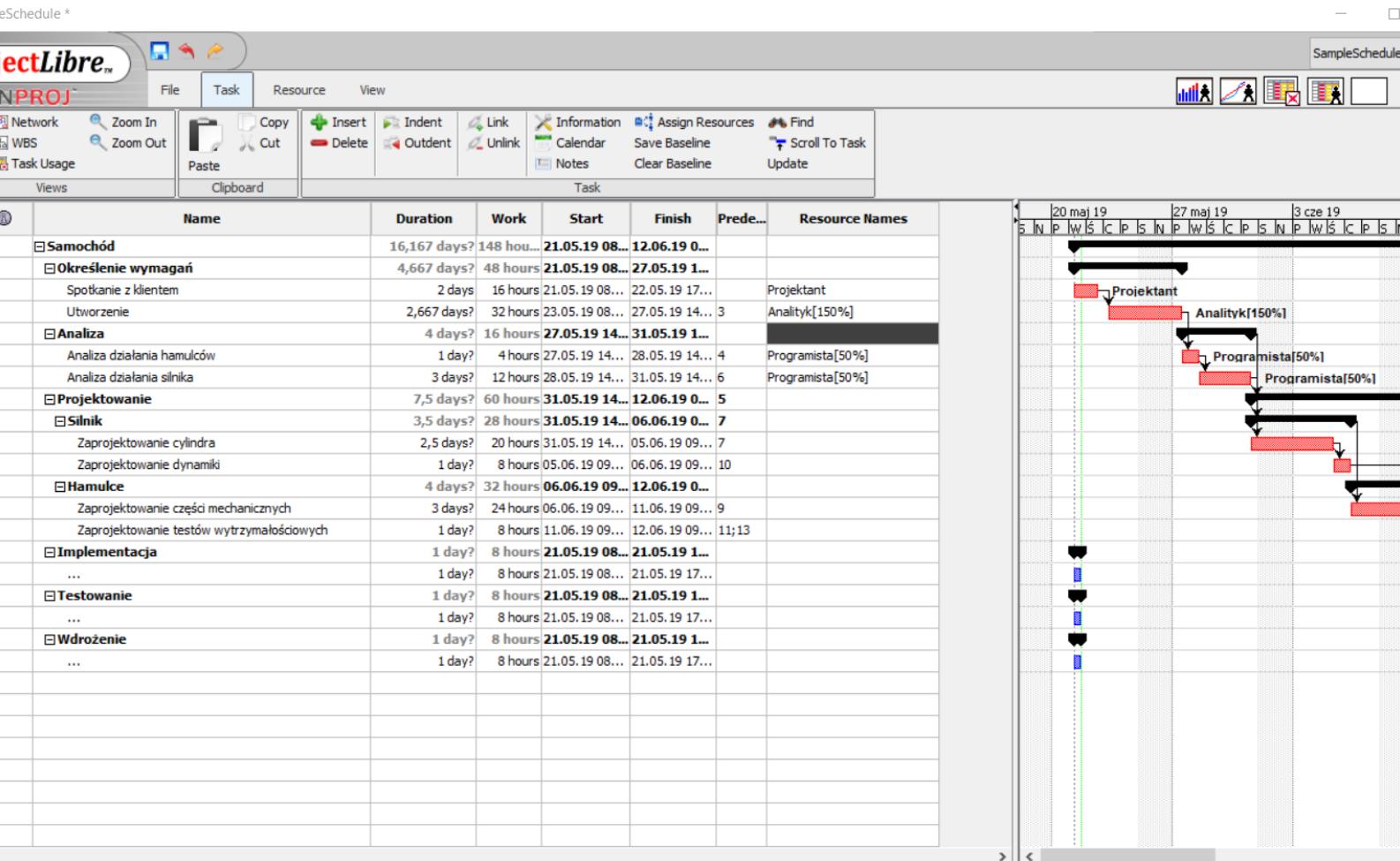
5. Jak już to mamy to dodajemy zasoby do naszego harmonogramu. Przechodzimy do zakładki Resorce i klikamy ikonkę Resources.  
Tam wypełniamy w kolumnie Name nazwę zasobu, po jej wypełnieniu naciskamy Enter i reszta powinna się wypełnić sama. Nas interesuje tylko kolumna Max. Units, w tej kolumnie określamy ile zasobów mamy w projekcie.

	Name	RBS	Type	E-mail Address	Material Label	Initials	Group	Max. Units
1	Projektant		Work			P		100%

Rysunek 19: Dodawanie zasobów do projektu.

6. Wracamy do widoku harmonogramu klikając w ikonę Gantt w zakładce Task. Teraz w kolumnie ResourceName wpisujemy nazwę zasobu oraz procentową wartość jaką chcemy przypisać do zadania.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA



Rysunek 20: Przypisywanie zasobów do zadań.

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

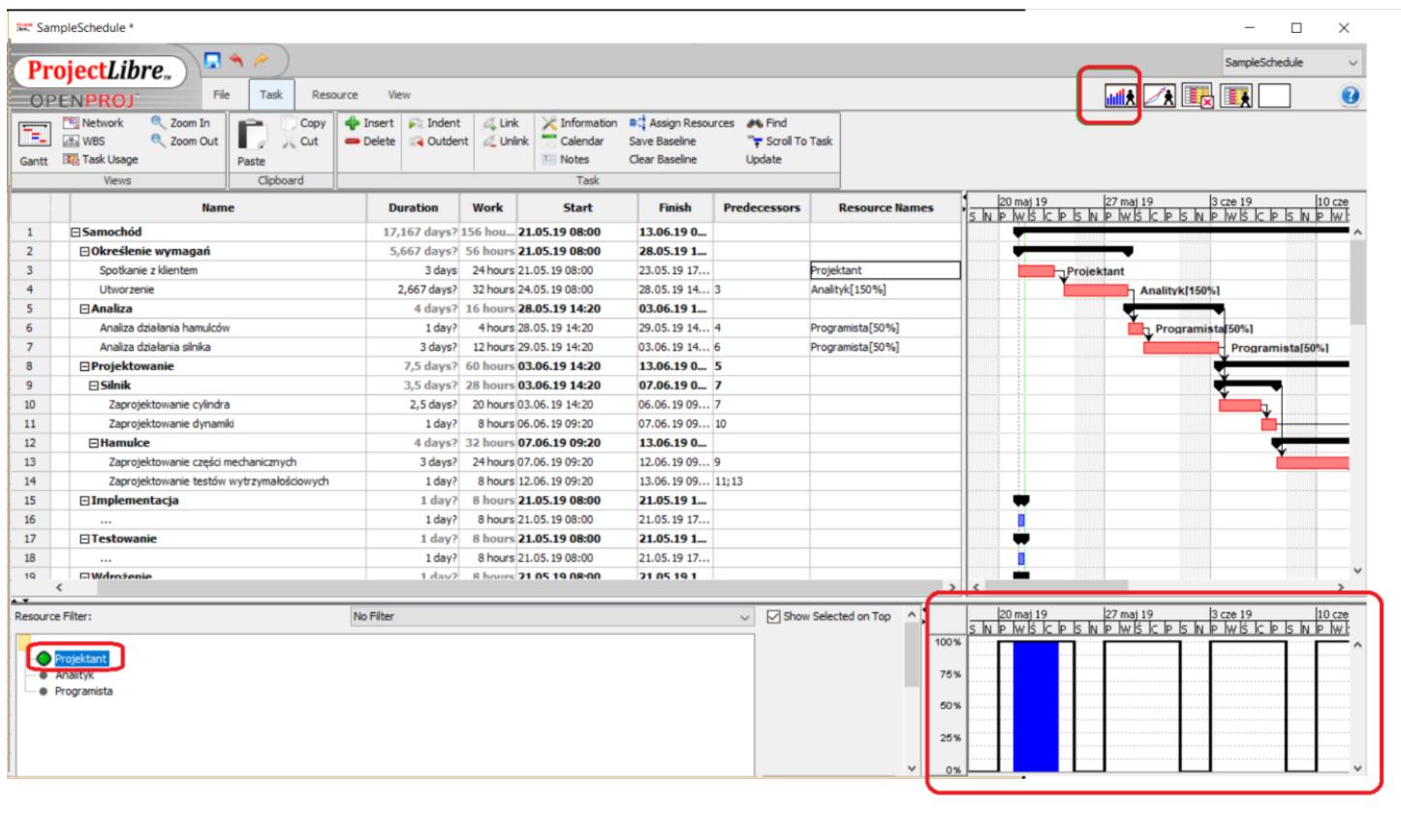
7. Po przypisaniu zasobów możemy sobie wyświetlić histogram przypisanych zasobów. Klikając ikonę w prawym górnym rogu. Zaznaczmy zasób dla którego chcemy zobaczyć histogram. Na histogramie widzimy zaznaczone kolorem czarnym dostępność zasobu oraz kolorem zielonym wykorzystanie zasobu w danym dniu.

Histogram może przekraczać 20-50 % w którąś stronę, ale nie więcej.

Mrówka podał wzór na sprawdzenie czy zasoby są dobrze przypisane:

$$resources = \frac{work}{duration}$$

work to kolumna work, a duration kolumna duration. Jeśli sprawdzamy czy np. zasoby się zgadzają dla programistów, to bierzemy wszystkie ich zadania i sumujemy work oraz osobno duration. Następnie dzielimy sumę work przez sumę duration (pamiętamy o tych samych jednostkach, np. godzin, dni) i w rezultacie oczekujemy, że uzyskamy liczbę odpowiadającą temu ile zasobów rzeczywiście mamy w projekcie. Przykładowo dla 3,5 programistów prawidłowa będzie liczba z przedziału 3-3,5.



Rysunek 21: Histogram z zasobami.

Jeśli wyjdzie nam gdzieś brak zasobów, to wprowadzamy pomiędzy zadaniami relacje F2S (wykład Mrówka), aby rozłożyć obciążenie zasobów.

#### **2.2.7. Harmonogram - uwaga ogólna**

- Może być tak, że z jednego bloczka wychodzą dwie lub więcej relacji...
- Defaultowo harmonogram jest robiony na zasadzie 8 godzinnego dnia pracy (08:00-12:00 + 13:00-17:00) i pięciodniowego tygodnia pracy (pon.-pt.), dlatego czasem jeśli coś ma czas trwania dwa dni, a zaczyna się w piątek, to będzie przechodzić przez sobotę i niedzielę i skończy się w poniedziałek.

I to chyba tyle w temacie... :)

### 3 Egzamin cz.2 - Ustna

---

#### 3.1. Wykład 1 - 1\_mpiso\_introduction\_v0.1

##### Pytanie

Do czego służą metody pomiaru i szacowania oprogramowania ?

Są to metody stosowane aby odpowiedzieć na pytania:

- Jak długo będzie trwał projekt ?
- Ile będzie kosztował projekt ?
- Jak poprowadzić projekt aby zmieścić się w zadanym czasie i kosztach ?

##### Pytanie

Co oznacza TOC (Total cost of ownership) ?

Jest to suma wszystkich kosztów rozwiązania informatycznego począwszy od jego zakupu, poprzez użytkowanie, aż do likwidacji. W skład TCO wchodzą koszty zakupu sprzętu i licencji, wydatki poniesione na szkolenia, wdrożenie, administrację, usuwanie awarii i likwidację. Jest to model pomagający rozdzielić i zrozumieć podział kosztów związanych bezpośrednio i pośrednio z użytkownikiem końcowym każdego przedsiębiorstwa.

##### Pytanie

Co można mierzyć/szacować ?

- Zakres funkcjonalny (funkcjonalność, wielkość) systemu – punkty funkcyjne, punkty obiektowe, przypadki użycia, liczba linii kodu itd.
- Jakość oprogramowania – ilość błędów na moduł/plik/komponent, poziom trudności utrzymania, itd.
- Koszt oraz czas wytworzenia oprogramowania – ilość osobodni, osoby tygodni (mandays, manweeks), czas trwania projektu, ilość niezbędnych zasobów

O tym czym są punkty funkcyjne i obiektowe będzie w pytaniach do wykładów 7/8.

##### Pytanie

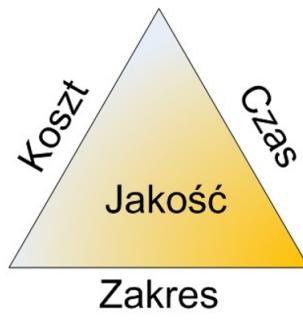
Co oznacza mandays, manweeks ?

Manday to jeden idealny dzień pracy programisty, a manweek to jeden idealny tydzień programisty.

Pytanie

Jakie są parametry definiujące projekt ? Jaka jest między nimi zależność ?

- Parametry definiujące projekt:
  - zakres
  - czas
  - koszt
  - jakość
- Zmiana jednego z nich pociąga zmianę pozostałych
- Najczęściej tracimy na jakości



Przy czym nie mówimy NIE gdy klient chce zmniejszyć jeden z boków trójkąta (wykład).

Pytanie

Jak rośnie ryzyko dla projektów informatycznych wraz ze wzrostem projektu ? Jaki jest bezpieczny próg dla projektu informatycznego ?

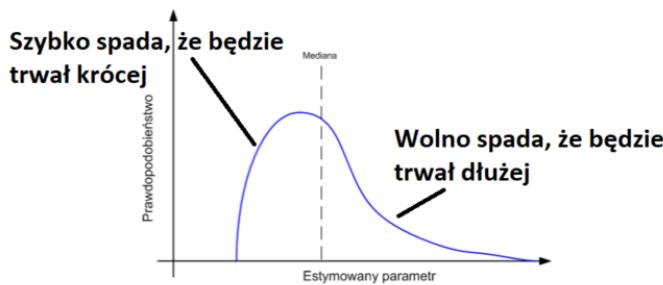
### Ryzyko projektu informatycznego

- Rośnie wraz z wielkością projektu (nieliniowo!!!)
- Heurystycznie ustalony prób bezpieczny dla projektów to 1500 FP
- Jeżeli to możliwe to duże projekty należy dzielić na mniejsze pod-projekty
- Produkty poszczególnych podprojektów powinny być realizowane i odbierane niezależnie

### Pytanie

Jak nie powinno się szacować projektu informatycznego jeśli chodzi o prawdopodobieństwo ? Czym jest estymacja jednopunktowa ? Czym jest rozkład  $\beta$  ?

Nie powinno się mówić, że coś będzie trwało tyle i tyle na 100% (estymacja jednopunktowa). Zawsze mówimy, że coś będzie trwało tyle i tyle z pewnym prawdopodobieństwem. Przy czym nie posługujemy się tutaj modelem rozkładu normalnego, w którym mamy jakąś wartość najbardziej prawdopodobną (medianę), a wartość mniejsza lub większa może pojawić się z mniejszym, ale równym sobie prawdopodobieństwem. Zamiast tego posługujemy się modelem rozkładu  $\beta$  w którym prawdopodobieństwo, że projekt będzie trwał krócej szybko spada, a że będzie trwał dłużej wolno spada:



### Pytanie

Jakie są negatywne czynniki (skutki) zawyżonego oszacowania ? (Czyli gdy estymata jest większa niż rzeczywista wartość parametru)

- Prawo Parkinsona – realizacja zadania zawsze wypełnia cały dostępny czas i dostępne zasoby
- Syndrom studenta – zwlekanie na ostatnią chwilę z rozpoczęciem zadania
- Kierownik typu X – pracownicy są nieodpowiedzialni, okłamują szefów, zawyżają oszacowania i wymagają ciągłej kontroli

Pytanie

Jakie są negatywne czynniki zaniżania oszacowania ? (Czyli gdy estymata jest mniejsza niż rzeczywista wartość parametru)



### Negatywne czynniki zaniżania oszacowania

- Brak możliwości rzeczywistego zaplanowania prac w projekcie
- Mniejsze prawdopodobieństwo na terminowe wykonanie projektu
- „Ucinanie kantów” na szkoleniu, dokumentacji, projekcie technicznym prowadzi do zmniejszenia jakości produktu
- Dodatkowe czynności i zasoby związane z gaszeniem pożaru (spotkania, eskalacje, raporty, wersje beta, itd.)
- **Ucieczka pracowników** wykwalifikowanych i znających problematykę

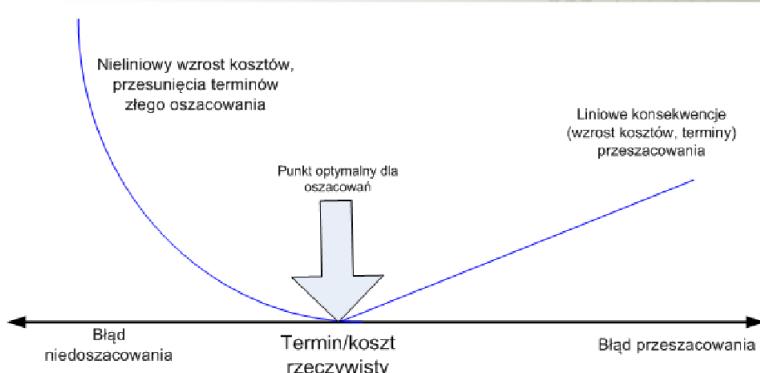
33

Pytanie

Jak rosną koszty w przypadku niedoszacowania i przeszacowania projektu ?



### Konsekwencje złego oszacowania



Uwaga: Kolejne trzy pytania dotyczą błędów, pytanie o te błędy może się pojawić również w formie:

### Pytanie

Jak nie należy szacować oprogramowania ? (czyli tak jak określają błędy)  
lub

Jak powinno się szacować oprogramowanie ? (czyli nie tak jak określają błędy<sup>②</sup>)

W powyższych pytaniach można też zaznaczyć o prawdopodobieństwo, czyli, że szacujemy nie jednopunktowo, ale z pewnym prawdopodobieństwem , a także, że trzeba dostosować metodę do fazy projektu (stożek niepewności - kilka slajdów później)

### Pytanie

Jakie błędy popełniamy przy szacowaniu oprogramowania ? Opis + przykłady



### Błędy metod

- Brak odniesienia do poprzednich projektów
- Stosowanie tylko jednej metody bez możliwości weryfikacji z wynikami innych metod
- Brak ciągłego uaktualniania estymacji i porównania z rzeczywistymi wielkościami po zakończeniu projektu
- Brak stosowania jakiejkolwiek metody estymacji
- Szacowanie w celu dostosowania wielkości do oczekiwania sponsora projektu, klienta, udziałowców, „price to win” itd.

36



### Błędy społeczne

- Nadmierny optymizm w stosunku do dostępności i kompetencji zespołu projektowego
- Przecenianie wpływu narzędzi i nowych technologii
- Brak odniesieni do krzywej uczenia się zespołu projektowego
- Utajnianie złych informacji przed przełożonymi



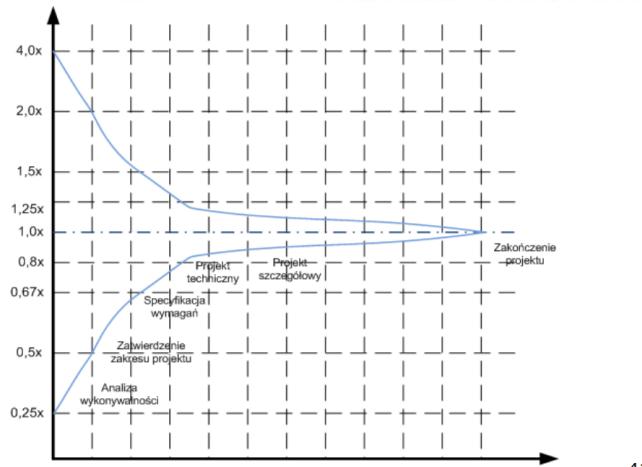
### Błędy organizacyjne

- Brak doświadczonych kierowników projektów
- Rozproszenie odpowiedzialności w strukturze organizacyjnej, brak jasnych ścieżek raportowania
- Rozproszenie zespołu projektowego
- Nieodpowiednie narzędzia i technologia

Pytanie

Od czego zależy niepewność w procesie estymacji ? Czym jest stożek niepewności ?

### Dokładność estymacji od fazy projektu – stożek niepewności



### Niepewność w procesie estymacji

- Estymacja wykonana na późniejszym etapie projektu jest bardziej wiarygodna
- W przypadku oszacowań jednopunktowych stosowane są przedziały predefiniowane dla poszczególnych etapów projektów – stożek niepewności
- W przypadku projektów iteracyjnych, każda iteracja powinna być traktowana jako pełny projekt

Pytanie

Czym jest problem niestabilnych wymagań ? Jak wpływa na stożek niepewności ?



## Problem niestabilnych wymagań

- Zmienna lub nieokreśloność wymagań jest najczęstszym powodem problemów w projekcie
- Najczęściej problem pojawia się w momencie prezentacji pierwszej wersji klientowi
- Możliwe środki zaradcze:
  - przygotowywać prototyp na wczesnych fazach projektu
  - angażować użytkownika końcowego w proces analizy wymagań
  - dokładać do oszacowań 40% zapasu na zmienność wymagań (NASA)

43

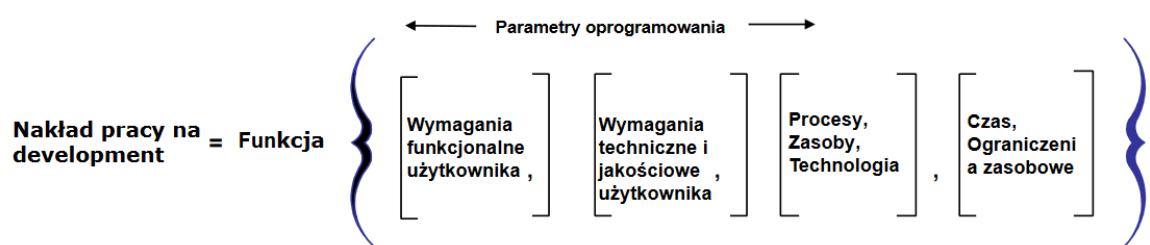
Pytanie

Jakie metody estymacji powinno się stosować w zależności od fazy projektu ?



## Zmiana metod estymacji w zależności od fazy

**Metody wczesnej estymacji (Early Life-Cycle), 'Top-down' (np. Punkty Funkcyjne lub metoda COCOMO)**



**Późna estymacja (Later Life-Cycle), metoda 'Bottom-up'**

Nakład pracy na development = Funkcja  $\times$   $\sum_i$  [Ilość kroków w procesie typu i]  $\times$  [Standardowa ilość godzin dla kroku w procesie typu i]

Pytanie

Czym jest tzw. Wskaźnik Fantazji ? Skąd się bierze ?

 „Wskaźnik fantazji”

- Kierownicy projektów i menadżerowie mają tendencję do fantazjowania – zaniżają oszacowania o około 30%
- Programiści także zaniżają estymację o około 20-30%
- Błędne założenia:
  - wzrost wydajności w stosunku do poprzedniego projektu
  - w poprzednim projekcie wszystko poszło źle – teraz będzie inaczej
  - wiemy znacznie więcej niż poprzednio

Pytanie

Czym jest oszacowanie ad hoc ?

 Oszacowania ad hoc

- Oszacowanie bez uprzedniej analizy ilościowej są obarczone większym błędem
- Człowiek będzie podawał wartości z poprzednich projektów – przez analogię
- Może doprowadzić do niekorzystnego zobowiązania
- Nie należy być zbyt precyzyjnym przy oszacowaniu zgrubnym

### 3.2. Wykład 2 - 2\_mpiso\_wbs\_v0.1

Pytanie

Czym jest WBS ?

Patrz opracowanie do cz.1 egzaminu

Pytanie

Jakie są rodzaje WBS ?

Patrz opracowanie do cz.1 egzaminu

Pytanie

Czym jest zasada 100% w kontekście WBS ?

Patrz opracowanie do cz.1 egzaminu

Pytanie

Czym jest słownik WBS ?

Patrz opracowanie do cz.1 egzaminu

Pytanie

Czym jest Work Package ?

Patrz opracowanie do cz.1 egzaminu.

Pytanie

Jak dużą pracę powinien zawierać work package ?

Na poziomie 1000 roboczogodzin, realizacja nie powinno przekraczać 2 tygodni.

Pytanie

Czym są aktywności i zadania ?

Patrz opracowanie do cz.1 egzaminu

Pytanie

Czym jest scope of work, a czym statement of work ?



## Definicja wyjściowa

- Scope of work - opisuje jaką pracę należy wykonać w projekcie
- Statement of work - określa i specyfikuje wymagania funkcjonalne i pozafunkcjonalne *deliverables*

Pytanie

Czym jest metoda ekspercka ?

### ↳ Szacowanie eksperckie

- Najczęściej stosowana metoda szacowania (Jorgensen 2002) – około 80% oszacowań
- Ocena eksperta jest obarczona największym błędem !!!
- Stosowana najczęściej w początkowych etapach projektów
- **Doświadczenie w projektach lub technologii nie czyni nikogo ekspertem**
- Eksperci w większości przypadków używają prostych strategii porównawczych

### ↳ Strukturalna ocena eksperta

- Oszacowanie bazujące na dekompozycji zadań lub pakietów prac
- Koncentruje się na oszacowaniu konkretnego zadania lub funkcji
- Najczęściej takie oszacowanie wykonują osoby, które będą realizować dane zadanie – osoby niezaangażowane podają statystycznie mniej dokładne oszacowania
- Częsty błąd – omijanie niezrozumiałych lub trudnych zadań lub funkcji

Pytanie

Jakie są etapy tworzenia WBS ?

■ Cztery kroki procesu tworzenia WBS:

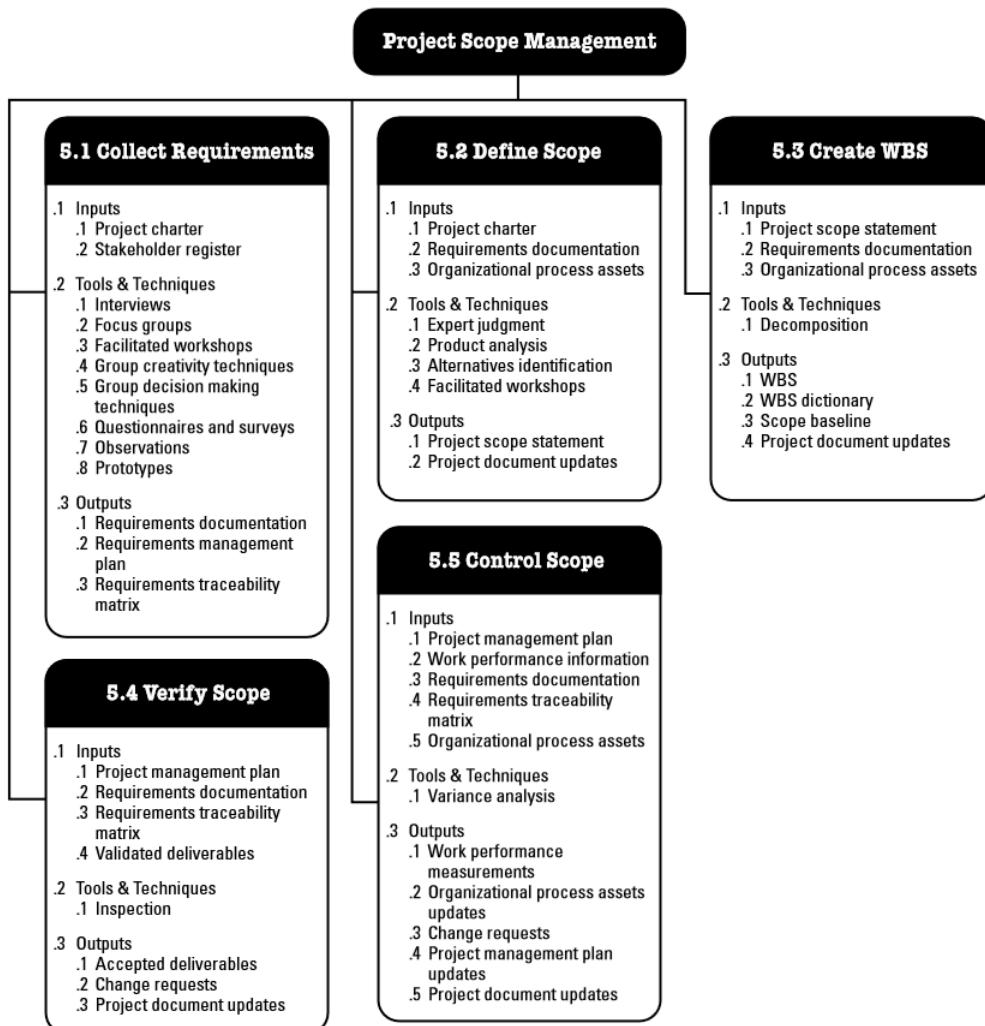
- Określenie celów projektu
- Specyfikacja głównych *deliverables* dla klienta (produkty, usługi, dokumentacja, itd.)
- Identyfikacja obszarów pracy, które prowadzą do wytworzenia *deliverables*
- Podział każdego elementu z korku 2 i 3 na podkategorie do momentu w którym złożoność i koszty wytworzenia będą zarządzalne i kontrolowalne

### 3.3. Wykład 3 - 3\_mpiso\_planowanie\_v0.1

#### Pytanie

Omów proces planowania projektu.

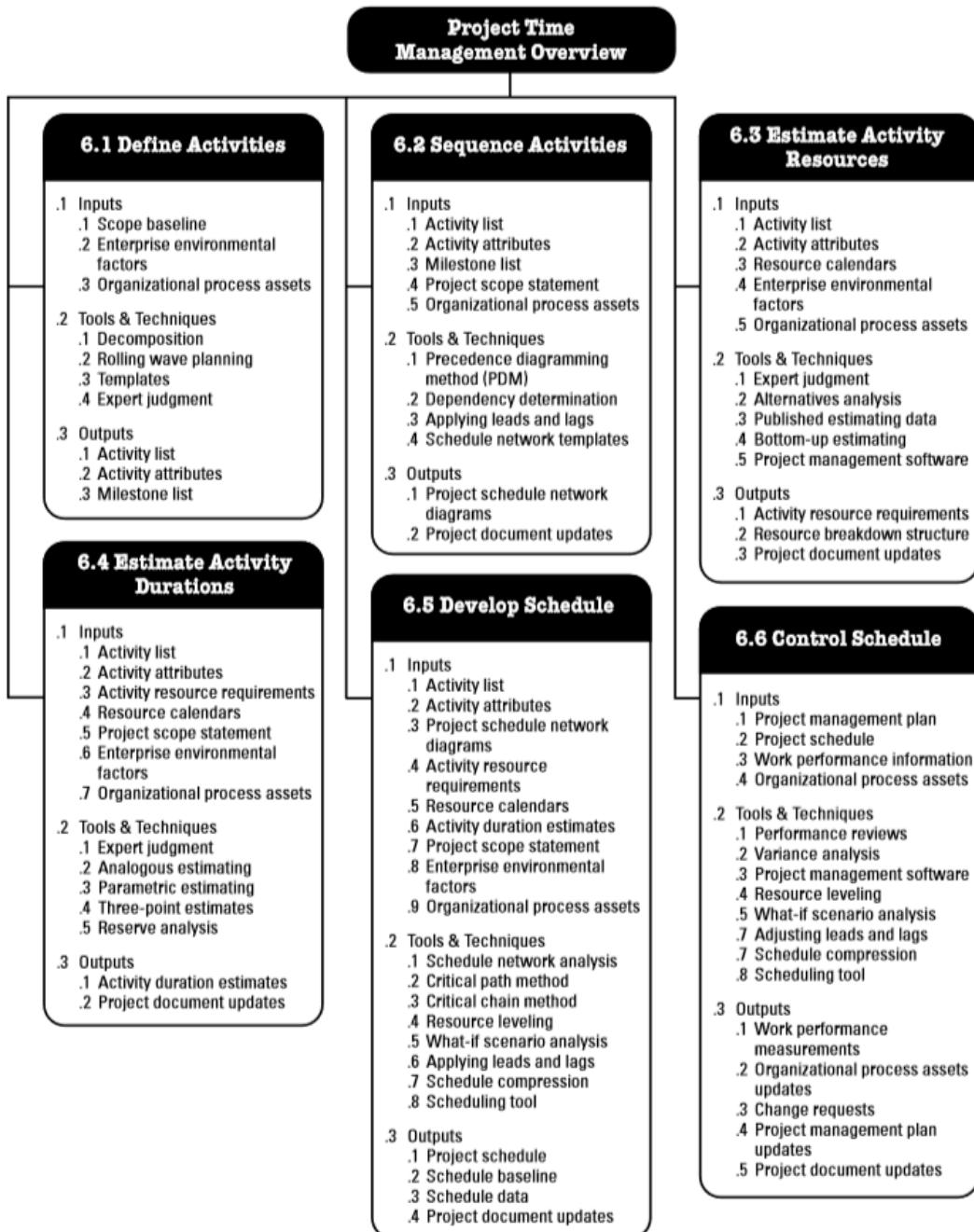
Tutaj u Mrówki jest slajd numer 2 w tym wykładzie, ale dokładniejsze i bardziej zrozumiałe wydają się być diagramy z PMBOK:



Czyli w uproszczeniu na podstawie wymagań tworzymy sobie WBS, a dalej...

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

... na podstawie WBS listę aktywności, następnie wprowadzamy relacje pomiędzy aktywnościami, a później estymujemy zasoby i czas aktywności (zadań), po to aby na końcu stworzyć ze wszystkiego harmonogram.



Pytanie

Z czego wynika rodzaj relacji pomiędzy aktywnościami (zadaniami) ?

Wynika z logiki, że coś robimy najpierw, a coś później, np. najpierw baza danych, a później login itp. Po za tym wykład dodaje, że:

- Zidentyfikowane aktywności WBS muszą zostać połączone relacjami
- Rodzaj relacji wynika z charakteru zależności pomiędzy wykonywanymi zadaniami
- Źródłem zależności jest opis produktu lub przyjęty model realizacji projektu: kaskadowy, iteracyjny, V, itd.
- Przyjęte założenia dotyczące zależności wpływają na rejestr ryzyk

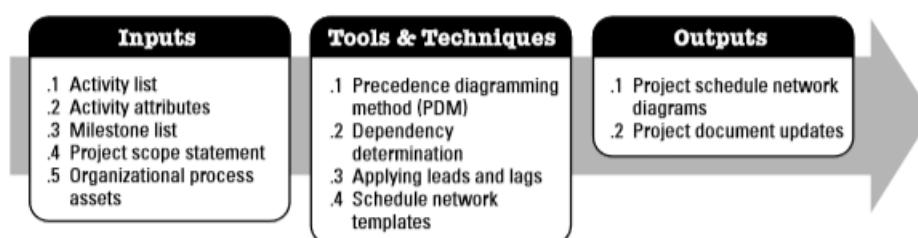
Pytanie

Czym jest lead and lag ?

Generalnie jest to narzędzie/technika stosowane przy wprowadzaniu logicznych relacji pomiędzy aktywnościami.

## 6.2 Sequence Activities

Sequence Activities is the process of identifying and documenting relationships among the project activities. Activities are sequenced using logical relationships. Every activity and milestone except the first and last are connected to at least one predecessor and one successor. It may be necessary to use lead or lag time between activities to support a realistic and achievable project schedule. Sequencing can be performed by using project management software or by using manual or automated techniques. See Figure 6-5 and Figure 6-6.

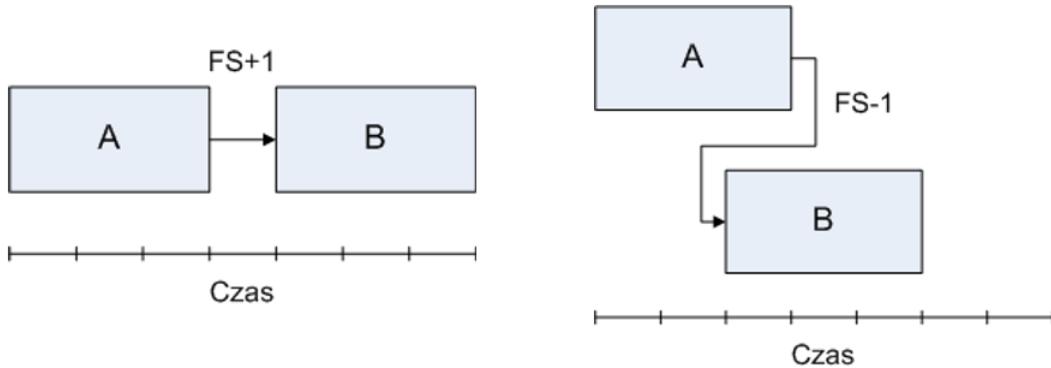


Generalnie chodzi w tej technice o to, że dane zadanie definiujemy w relacji do jakiegoś innego zadania, np. mówimy, że zadanie B powinno rozpoczęć się z 15 dniowym opóźnieniem w stosunku do zadania A

(lag). Albo mówimy, że zadanie B powinno rozpoczęć się 5 dni przed zakończeniem zadania A. Dodatkowo z PMBOK i z wykładów:

A lead allows an acceleration of the successor activity. For example, on a project to construct a new office building, the landscaping could be scheduled to start 2 weeks prior to the scheduled punch list completion. This would be shown as a finish-to-start with a 2-week lead.

A lag directs a delay in the successor activity. For example, a technical writing team can begin editing the draft of a large document 15 days after they begin writing it. This could be shown as a start-to-start relationship with a 15-day lag.



**Opóźnienie** – modyfikacja logicznej zależności pomiędzy zadaniami prowadząca do opóźnienia następnika (PMBOK 2000)

**Wyprzedzenie** – modyfikacja logicznej zależności pomiędzy zadaniami pozwalająca na wcześniejszego rozpoczęcia następnika(PMBOK 2000)

4

Pytanie

Jakie są kategorie zależności pomiędzy zadaniami ? Czym jest soft logic, a czym hard logic ?

### ↙ Kategorie zależności

- **Obowiązkowe** – zależności wynikające z rodzaju prac, fizyczne lub logistyczne ograniczenia – *hard logic*
- **Wymagane** – zależności wynikające z optymalizacji zadań, wydajności, wzorców projektowych – *soft logic*
- **Zewnętrzne** – zależności wynikająca z powiązań aktywności projektowych z poza projektowymi: kampania reklamowa, wybory, pory roku itd.

Pytanie

Jakie mogą występować relacje pomiędzy aktywnościami (zadaniami) ?

S2S, itd. -> Patrz opracowanie do cz.1

Pytanie

Jakie parametry przyporządkowujemy zadaniom w sieci ?

- **Najwcześniejszy start (ES – early start)**  
– czas, od którego zadanie może zostać najwcześniej rozpoczęte
- **Najwcześniejsze zakończenie (EF – early finish)** – czas, w którym zadanie może zostać najwcześniej zakończone
- **Najpóźniejszy start (LS – late start)** – czas, od którego zadanie może zostać najpóźniej rozpoczęte
- **Najpóźniejsze zakończenie (EF – late finish)** – czas, w którym zadanie może zostać najpóźniej zakończone



Pytanie

Czym jest ścieżka krytyczna w sieci ?

Pamiętamy, że ścieżka krytyczna określa najdłuższą drogę w grafie. Jeśli zadanie na ścieżce krytycznej opóźni się to cały projekt się opóźni.



## Ścieżka krytyczna

- Ścieżka krytyczna sieci pozwala na określenie najkrótszego czasu trwania projektu
- Określa daty rozpoczęcia oraz zakończenia zadań oraz ewentualną rezerwę na zadanie
- Opóźnienie w sieci krytycznej prowadzi do opóźnienia w projekcie
- Skrócenie harmonogramu możliwe jest jedynie poprzez skrócenie aktywności na ścieżce krytycznej

Pytanie

Czym jest float (slack) ?

Zadanie na ścieżce krytycznej mają float równe 0.



## Parametry dodatkowe

- Opóźnienie (float lub slack) – ilość czasu, o który zadanie może zostać opóźnione bez wpływu na całkowity czas trwania projektu
- Rezerwa zadania (free slack) – ilość czasu, o który zadanie może być opóźnione, bez wpływu na czas rozpoczęcia następnika
- Rezerwę zadania oblicza się jako różnica:

$$\text{LF-EF lub LS-ES}$$

Pytanie

Techniki skracania harmonogramu

Generalnie są dwie:

- Crashing - polega na tym, że dodajemy sobie do projektu jakieś dodatkowe zasoby, np. co by było gdybyśmy zatrudnili dwóch programistów więcej... i badamy jak to wpłynie na ścieżkę krytyczną, może to jednak prowadzić do wzrostu ryzyka projektowego w zakresie przekroczenia kosztu.

Z PMBOK:

**Crashing.** A schedule compression technique in which cost and schedule tradeoffs are analyzed to determine how to obtain the greatest amount of compression for the least incremental cost. Examples of crashing could include approving overtime, bringing in additional resources, or paying to expedite delivery to activities on the critical path. Crashing only works for activities where additional resources will shorten the duration. Crashing does not always produce a viable alternative and may result in increased risk and/or cost.

- Fast tracking - polega na tym, że coś co powinno się wykonywać po kolei robimy równolegle, np. robimy fundamenty równolegle z rysunkami architektonicznymi. Takie rozwiązanie prowadzi do ryzyka związanego z tym, że coś będziemy robić jeszcze raz.

Z PMBOK:

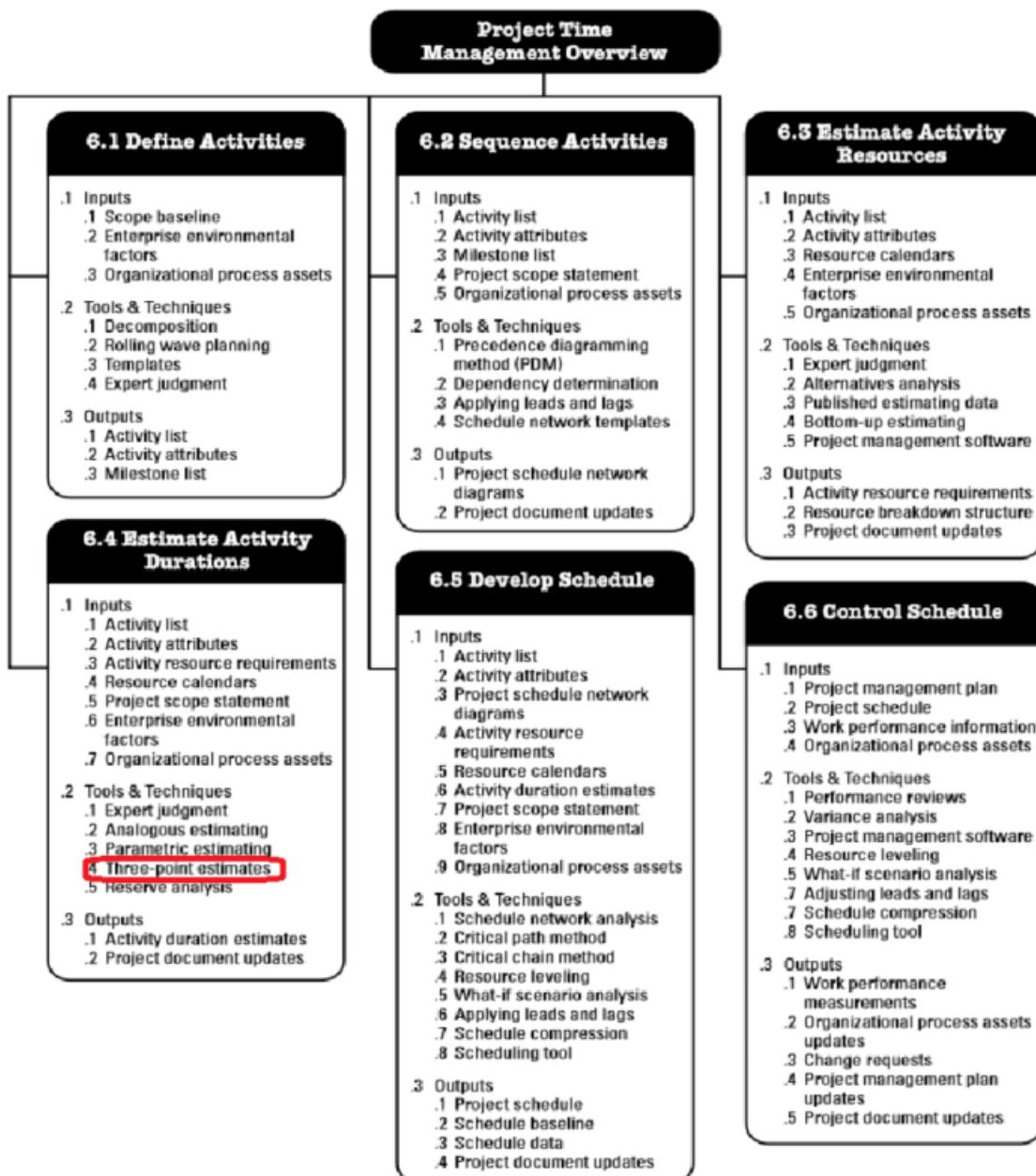
- **Fast tracking.** A schedule compression technique in which phases or activities normally performed in sequence are performed in parallel. An example is constructing the foundation for a building before completing all of the architectural drawings. Fast tracking may result in rework and increased risk. Fast tracking only works if activities can be overlapped to shorten the duration.

### 3.4. Wykład 4 - 4\_mpiso\_pert\_delphi\_v0.1

Pytanie

Czym jest metoda PERT ?

Generalnie jest to metoda służąca do szacowania czasu wykonania aktywności, czyli na naszej mapie jesteśmy tutaj:



Ogólnie rzecz biorąc PERT polega na tym, że każdej aktywności przypisujemy trzy czasy wykonania:

- Najbardziej prawdopodobny
- Najbardziej optymistyczny
- Najbardziej pesymistyczny

Tak jak przykładowo tutaj mamy:

ID	WBS	Task Name	Optimistic Dur.	Expected Dur.	Pessimistic Dur.	Duration
1	1	Projekt MegaPortal	358 days	447 days	604 days	458,33 days
2	1.1	Zarządzanie projektem	60 days	66 days	100 days	70,67 days
3	1.1.1	Zarządzanie jakością	20 days	22 days	35 days	23,83 days
4	1.1.2	Planowanie zadań	20 days	22 days	30 days	23 days
5	1.1.3	Kontrolowanie prac	20 days	22 days	35 days	23,83 days
6	1.2	Anaiza funkcjonalna	45 days	58 days	78 days	59,17 days
7	1.2.1	Analiza procesów biznesowych	10 days	15 days	25 days	15,83 days
8	1.2.2	Opracowanie specyfikacji wymagań	20 days	23 days	28 days	23,33 days
9	1.2.3	Wymagania architektury	15 days	20 days	25 days	20 days
10	1.3	Projekt techniczny	64 days	84 days	106 days	84,33 days
11	1.3.1	Projekt architektury	10 days	12 days	17 days	12,5 days
12	1.3.2	Projekt komponentów	15 days	18 days	25 days	18,67 days
13	1.3.3	Projekt integracji	14 days	16 days	24 days	17 days



## Analiza PERT

- PERT – Program Evaluation and Review Technique
- Pozwala na wyznaczenie oczekiwanej przypadku w oszacowaniu trzypunktowym: optymistycznym, oczekiwany i pesymistycznym
- Dodatkowo technika pozwala na opracowanie sieci prac i ścieżki krytycznej CPM
- Metoda zakłada wykorzystanie rozkładu  $\beta$  oraz trójkątnego do oszacowania wartości oczekiwanej

PERT analysis calculates an **Expected** ( $t_E$ ) activity duration using a weighted average of these three estimates:

$$t_E = \frac{t_0 + 4t_M + t_p}{6}$$

No i możemy sobie z tego wzorku policzyć czas oczekiwany dla zadania na podstawie tych trzech czasów.

Ale nie to jest istotą metody PERT. Istotą jest to, że traktujemy czas trwania projektu jako zmienną losową, czyli możemy powiedzieć, że projekt się skończy w tyle i tyle wynosi tyle i tyle. Natomiast czas oczekiwany posłuży nam jako parametr dla rozkładu prawdopodobieństwa.

Jak już wiemy w projektach informatycznych lepiej od rozkładu normalnego sprawdza się rozkład  $\beta$ . Krótkie przypomnienie o rozkładzie beta:

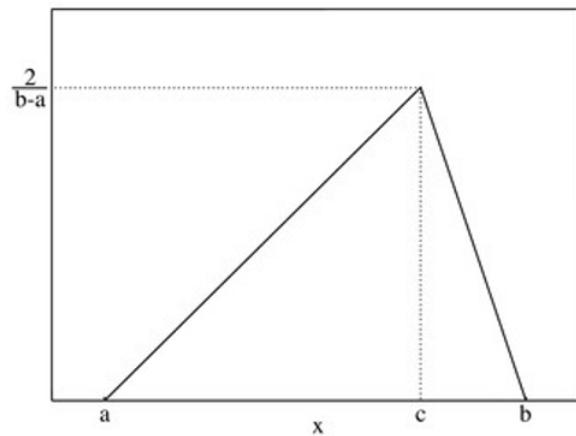


## Rozkład $\beta$

- Rozkład  $\beta$  jest wykorzystywany w statystyce i sieciach bayes'nowskich
- Wykorzystywana w modelach zdarzeń, których realizacja odbywa się w wyznaczonych przedziałach
- Bazuje na funkcji specjalnej – poprawnej całce Eulera
- Wykorzystana w metodyce PERT ze względu na rozkład prawdopodobieństwa poprawnego oszacowania

Z tym, że rozkład  $\beta$  ma dość skomplikowane wzory (całka Eulera !) więc przeciętny manager może nie ogarnąć dlatego przybliża się go rozkładem trójkątnym prawdopodobieństwa, który nie jest tak dokładny, ale daje radę.

## Funkcja gęstości rozkładu trójkątnego



## Rozkład trójkątny

### ■ Funkcja gęstości:

$$f(x|a, b, c) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)} & c \leq x \leq b \\ 0 & \text{pozostale} \end{cases}$$

### ■ Momenty:

$$E(X) = \frac{a+b+c}{3} \quad V(X) = \frac{a^2 + b^2 + c^2 - ab - ac - bc}{18}$$



## Metoda PERT

- Na podstawie aproksymacji PERT obliczamy wartość oczekiwana:

$$E(X) = \frac{x_{min} + x_{max} + \lambda x_{sr}}{\lambda + 2}$$

- oraz odchylenie standardowe:

$$\delta(X) = \frac{x_{max} - x_{min}}{\lambda + 2}$$

- Na podstawie powyższych parametrów możemy określić wartość współczynników kształtu  $v, w$ .

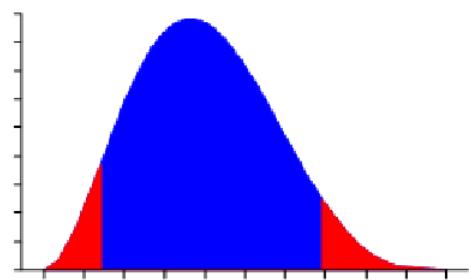
Czyli liczymy sobie sumaryczny czas zadań optymistyczny, pesymistyczny i średni i obliczamy  $E(X)$  i  $V(X)$  jak we wzorcach wyżej, a na tej podstawie można sobie wyrysować rozkłady:



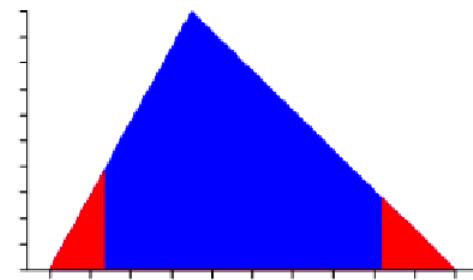
## Wartości współczynników kształtu

$$v = \frac{E(X) - x_{min}}{x_{max} - x_{min}} \left( \frac{(E(X) - x_{min})(x_{max} - E(X))}{\delta(X)^2} - 1 \right) \quad w = \frac{x_{max} - E(X)}{E(X) - x_{min}} v$$

$v=3,05 \quad w=4,57$



$\text{min}=10 \quad \text{moda}=13,5 \quad \text{max}=20$



Pytanie

Jak zastosować analizę PERT dla ścieżki krytycznej (CPM - Critical Path Method) ?

Chodzi o to, że dla zadań na ścieżce krytycznej obliczamy średnią długość trwania z tego wzorku z czasem optymistycznym, pesymistycznym, oczekiwany a później wariancję dla tego zadania analogowo. No i średnia długość trwania całego projektu to suma średnich, a wariancja to suma wariancji (zakładamy, że zdarzenia są niezależne).

Pytanie

Jak obliczyć prawdopodobieństwo zakończenia projektu ?



## Prawdopodobieństwo zakończenia projektu

- Wykorzystuje się rozkład normalny do estymacji prawdopodobieństwa zdarzenia zakończenia projektu
- Prawdopodobieństwo obliczane jest dla długości ścieżki krytycznej
- Dla kilku ścieżek krytycznych, prawdopodobieństwo wyznacza się jako iloczyn prawdopodobieństw dla poszczególnych ścieżek (uproszczenie – ścieżki krytyczne jako zdarzenia niezależne)

Pytanie

Czym jest metoda Delphi ?

- **Delphi technique.** The Delphi technique is a way to reach a consensus of experts. Project risk experts participate in this technique anonymously. A facilitator uses a questionnaire to solicit ideas about the important project risks. The responses are summarized and are then recirculated to the experts for further comment. Consensus may be reached in a few rounds of this process. The Delphi technique helps reduce bias in the data and keeps any one person from having undue influence on the outcome.



## **Szacowanie w grupach ekspertów – *Wideband Delphi***

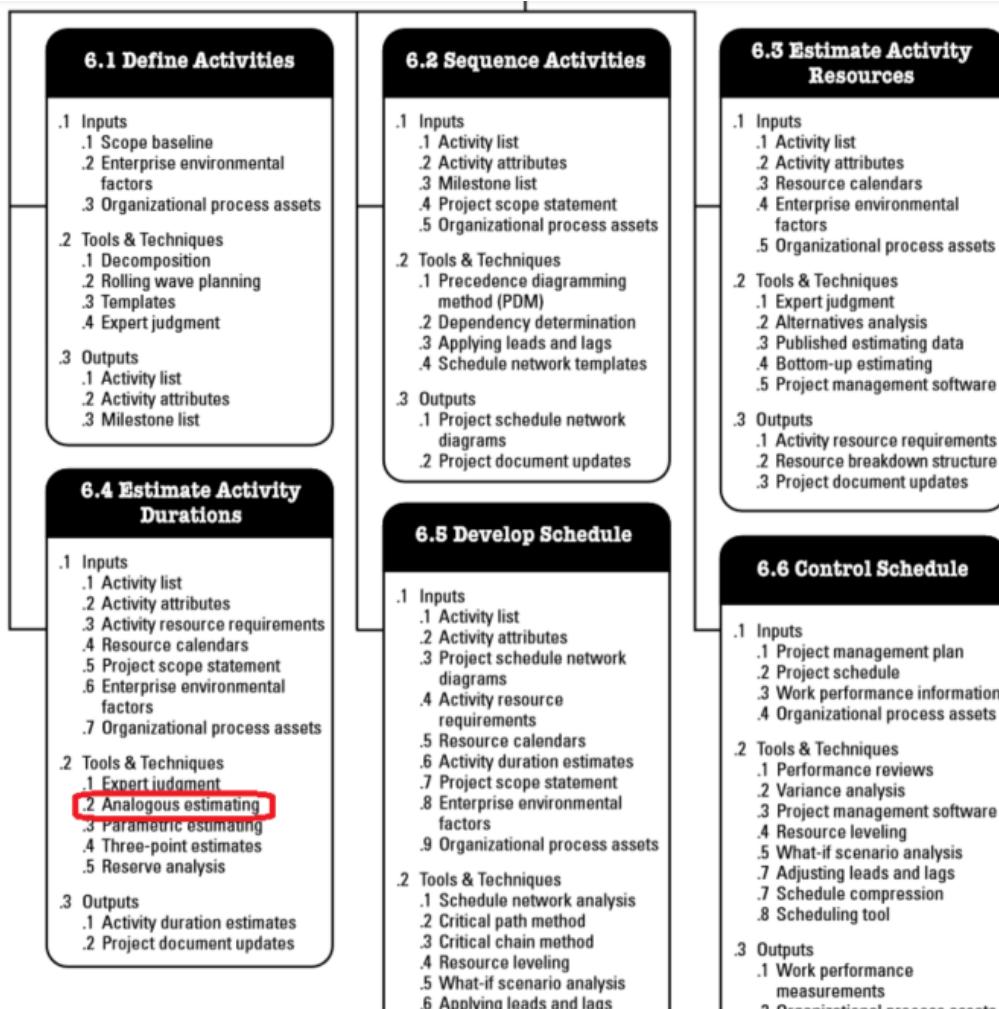
- Stosowane w początkowych fazach projektu
- Zaangażowanie większych zasobów uzasadnione dla dużych projektów
- Zakłada się poprawę dokładności oszacowania poprzez „inteligencję zbiorową”
- Badania wykazały, iż grupa 3-5 ekspertów jest wystarczająca (Jorgensen 2002)

### 3.5. Wykład 5 - 5\_mpiso\_analogia\_v0.1

Pytanie

Do czego służy szacowanie przez analogię ?

Służy do estymacji czasu wykonania aktywności (zadań), czyli na naszej mapie PMBOK jesteśmy tutaj:



Przy czym projekty muszą być do siebie podobne aby to stosować, np. nie ma sensu stosować szacowania przez analogię dla projektu android i systemu wbudowanego.

Pytanie

Jakie są najważniejsze założenia szacowania przez analogię ?

## Szacowanie przez analogię

- Metoda wykorzystywana przede wszystkim we wczesnych etapach projektu
- **Konieczne jest posiadanie bazy danych wykonanych projektów**
- Zakładamy zbliżone warunki realizacji nowego i poprzednich projektów
- Szacowanie może odbyć się po dekompozycji projektu na co najmniej 5 części
- Otrzymane dane należy traktować jako mało dokładne oszacowanie ( $\sim 50-100\%$ )
- **Prawo wielkich liczb !!!**

Pytanie

Czym jest prawo wielkich liczb (Bernoulliego) ?

Chodzi ogólnie o to, że przy dużej liczbie prób wszystko się uśrednia.

Pytanie

Czym są metryki oprogramowania ?

Metryki oprogramowania służą do oceny oprogramowania pod względem ilościowym, jakościowym itp. Są stosowane w szacowaniu przez analogię, ponieważ chcemy jakoś porównać starszy projekt z tym który mamy realizować.

Pytanie

Jakie znasz metryki oprogramowania ?

Metryki oprogramowania zależą od tego co chcemy mierzyć dlatego można wyróżnić ich kilka rodzajów, m.in.:

• Jakościowe

Metryki jakości należą do grupy metryk procesu i produktu.

Podstawowe metryki jakości produktu:

- częstość defektów
- średni czas bezawaryjnej pracy (MTTF)
- problemy klienta (zgłoszenia klienta)
- satysfakcja klienta

Wewnętrzprocesowe metryki jakości:

- częstość defektów podczas testów maszynowych
- efektywność usuwania defektów

Metryki serwisowe oprogramowania:

- opóźnienie napraw i wskaźnik zarządzania opóźnieniami (BMI)
- czas odpowiedzi na zgłoszenie
- odsetek źle wykonanych napraw
- jakość naprawy

• Ilościowe

Tutaj dobrze jest wziąć pod uwagę co liczymy, przykładowo podstawową metryką ilościową są linie kodu, ale w przypadku baz danych może to być mało miarodajne, dlatego dobrze jest dopasować metrykę do zagadnienia.

- Linie kodu
- Liczba klas (dla programowania obiektowego)
- Liczba tabel (bazy danych)
- Liczba widoków (Front-end, GUI)
- itd.

Czasem pod konkretne zastosowanie trudno coś wymyślić, np. logika biznesowa, wtedy w szacowaniu przez analogię przyjmuje się współczynnik zgodnie z intuicją.

Pytanie

Czym są logiczne (SLOC), a czym fizyczne linie kodu ?

## Zliczenie linii kodu

### ■ Rozróżniania się wiele metod zliczania linii kodu źródłowego:

- TLOC – wszystkie linie w plikach (synonim total LOC)
- fizyczne LOC – wszystkie linie kodu, łącznie z komentarzami oraz do 25% (danej sekcji) linii pustych
- LOC (SLOC) – logiczne linie kodu źródłowego, czyli linii, które zawierają jeden *statement* (instrukcję), zależnie od języka programowania (np. w C to linia do ';' bez *for*)
- CLOC – ilość linii komentarza
- BLOC – ilość linii pustych

4

Pytanie

Jakie są kroki szacowania przez analogie ?

1. Dekompozycja - dzielimy projekt na części takie jak występowały w przeszłych projektach (np. baza danych, logika biznesowa, GUI, itp.)
2. Ustalenie wartości odniesienia - bierzemy jeden lub kilka starych projektów i liczymy logiczne linie kodu dla każdej wydzielonej warstwy, możemy też zastosować inne wielkości, takie jak liczba tabel dla baz danych, liczba ekranów dla GUI itd.
3. Porównujemy wielkości i wyznaczamy współczynniki



### Krok 3 – porównanie wielkości

Lp.	Moduł	Wielkość poprzednia	Nowa wartość	Współczynnik
1	Baza danych	25 tabel	30 tabel	1,2
2	Logika biznesowa	nieokreślone	nieokreślone	1,7*
3	Warstwa prezentacji (GUI)	50 ekranów	45 ekranów	0,9
4	Raporty	10 raportów +2 Wykresy	15 raportów + 3 wykresy	1,5
5	Framework (klasy narzędziowe)	35 klas	50 klas	1,45

Pamiętamy, że



### Krok 3 – wyniki silnie zależne od założeń

- Porównanie odbywa się na wysokim poziomie abstrakcji
- Nie uwzględnia się złożoności poszczególnych modułów (klas, tabel, raportów, wykresów, itd.)
- Zakłada się, że realizować nowych projekt będzie ten sam zespół
- Często współczynnik należy odgadnąć (np. logika biznesowa)
- Nie są uwzględniane takie elementy jako technologia

4. Porównujemy wielkości

#### **Krok 4 – porównanie wielkości**

Lp.	Moduł	Wielkość LOC	Współczynnik	Nowa wielkość LOC
1	Baza danych	3 000	1,2	3 600
2	Logika biznesowa	35 000	1,7	59 500
3	Warstwa prezentacji (GUI)	12 000	0,9	10 800
4	Raporty	5 500	1,5	8 250
5	Framework (klasy narzędziowe)	10 000	1,45	14 500

#### **Krok 4 – inne możliwości jednostek**

- Porównanie może odbywać się dla innych jednostek np. punkty funkcyjne, tabele, ekranы, przypadki użycia
- Przyjęte jednostki powinny posiadać przelicznik do jednostek pracy dla danej organizacji
- Można stosować różne jednostki dla poszczególnych modułów

5. Wyliczamy pracochłonność



## Krok 5 – wyznaczenie pracochłonności

- Na podstawie wielkości z kroku 4 obliczamy:

Parametr	Wielkość
Wielkość poprzedniego projektu	65 500 LOC
Szacowana wielkość nowego projektu	96 650 =~ 100 000 LOC
Współczynnik porównania	1,52
Pracochłonność poprzedniego projektu	35 osobomiesiący
<b>Szacowania pracochłonność nowego projektu</b>	<b>53,2 =~ 55 osobomiesiący</b>

6. Sprawdzamy założenia - jak duży współczynnik studenta zastosować ?



## Krok 6 – sprawdzamy założenia

- Uwzględnić nieekonomiczność skali
- Zmodyfikować współczynniki ze względu na technologię
- Uwzględnić doświadczenie zespołu projektowego
- W niektórych przypadkach niezbędne jest uwzględnienie zmian w architekturze starego i nowego projektu (dwuwarstwowa i trójwarstwowa)

Pytanie

Kiedy stosujemy szacowanie przez zastąpienie ?

Np. gdy szacujemy apkę mobilną na podstawie strony www, ale w stronie www jest jsp, a w android nie ma. Wtedy umawiamy się, że np. zastępujemy widok apki mobilnej dwoma plikami .jsp.

Pytanie

Czym jest metoda szacowania zwana logiką rozmytą ?



## Logika rozmyta

- Metoda oryginalnie zaproponowana przez Putnam'a oraz Meyers'a w 1992
- Bazuje na naturalnych kategoriach oceny przez umysł ludzki
- Wprowadza się klasyfikacje jakościowe, a nie ilościowe: mało, dużo, średnio, itd.
- W metodzie istotna jest odpowiednia kalibracja poszczególnych klas – niezbędne dane historyczne
- Wymaga większej ilości danych historycznych (min 20 projektów)

Ogólnie chodzi o to, że bierzemy moduły z poprzednich projektów i przypisujemy im kategorie bardzo małe, mały, średni, duży, bardzo duży itp. i mamy policzoną średnią liczbę linii kodu dla danego modułu o danej etykiecie. Później dopasowujemy to do liczby i wielkości modułów w naszym projekcie.



## Kalibracja danych – na danych historycznych

Rozmiar modułu	Liczba modułów	Łączna LOC na wszystkie moduły	Średnia LOC na moduł
Bardzo duży	123	285 852	2324
Duży	34	25 262	1309
Średni	53	39 379	743
Mały	76	18 468	243
Bardzo mały	135	13 230	98

Pytanie

Czym jest metoda składników standardowych ?



## Składniki standardowe

- Metoda bazuje na zliczaniu elementów architektury systemu
- Zakłada się, że metoda wytworzenia danego rodzaju modułu jest podobna w przypadku porównywanych projektów
- Oszacowanie w odniesieniu do danych historycznych
- Można wprowadzić metodę PERT w wyznaczaniu wartości oczekiwanej

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA



### Przykład szacowania – składniki standardowe

Składnik	LOC / skt.	Min	Naj. prawdo-podobna	Max	Oczekiwana	Szacowana LOC
Logika biznesowa	5439	-	13	-	13	70 707
Tabele	2253	23	34	56	36	81 108
Raporty	987	12	23	54	26	25 662
Szablony WWW	324	32	45	52	44	14 256
Formularze WWW	1387	56	65	78	66	91 542
<b>RAZEM</b>						<b>283 275</b>

#### Pytanie

Czym są percentile i gdzie się z nich korzysta ?

Percentyle określają położenie danego wyniku względem innych wyników, jeśli mówimy, np. że coś ma 25% percentilei, to znaczy 25% wyników było poniżej tego wyniku, jeśli coś ma 50% tzn. 50% wyników było poniżej tego wyniku.

Percentyle można zastosować w metodzie składników standardowych do wyznaczenia przedziałów dla etykiet:



### Składniki standardowe z percentylami

Składnik	Bardzo mało (10%)	Mało (25%)	Srednio (50%)	Dużo (75%)	Bardzo dużo (90%)
Logika biznesowa	5439	6578	7565	8103	9324
Tabele	2253	2546	3453	4342	4902
Raporty	987	1234	1664	1953	2102
Szablony WWW	1543	2432	2675	2984	3124
Formularze WWW	5342	5860	6345	7432	8321

Pytanie

Jak szacowane są historyjki w XP ?

Szacowanie historyjek korzysta właśnie z logiki rozmytej, ponieważ wprowadzane są etykiety dla oszacowania zadań.



## Punktacja historyjek

- Wariant logiki rozmytej wykorzystywany w programowaniu ekstremalnym – Cohn 2006
- Oszacowania opera się na opisie procesów biznesowych (historyjek), będących realizacją wymagań funkcjonalnych
- Grupa realizująca implementację przypisuje wagę (numeryczne) poszczególnym historyjką
- Wykorzystuje się różne skale wag:
  - potęga 2: 1, 2, 4 ,8, 16, ...
  - ciąg Fibonacciego: 1, 2, 3, 5, 8, 13, ...

3

Zazwyczaj pomiędzy kolejnymi etykietami jest coraz większa odległość, ponieważ im większe zadania tym trudniej je oszacować.

Pytanie

Jest też coś o Agile na tym wykładzie...

 **Charakterystyka podejścia agile 1/2**

- Metodyka podkreśla krytyczny warunek komunikacji bezpośredniej
- Komunikacja bezpośrednią dotyczy członków zespołu, jak również klientów
- Metryką postępu projektu jest wielkość działającego systemu (nie LOC, punkty, itd.)
- Satysfakcja klienta jest osiągana przez ciągłe dostarczanie wartościowego oprogramowania
- Dokumentacja jest drugorzędną w stosunku do działającego oprogramowania

 **Charakterystyka podejścia agile 2/2**

- Zmiany w wymaganiach można wprowadzać kiedykolwiek
- Zespół powinien składać się z dobrze zmotywowanych jednostek (indywidualności), którym ufamy
- Programiści i osoby biznesowe powinny współpracować bardzo blisko (co-location) i często się komunikować
- Samo organizujący się zespół
- Tendencja do upraszczania
- Adaptacja do zmian jest czynnikiem stałym w projekcie

Pytanie

Czym jest teoria ograniczeń (TOC)?



## Teoria ograniczeń

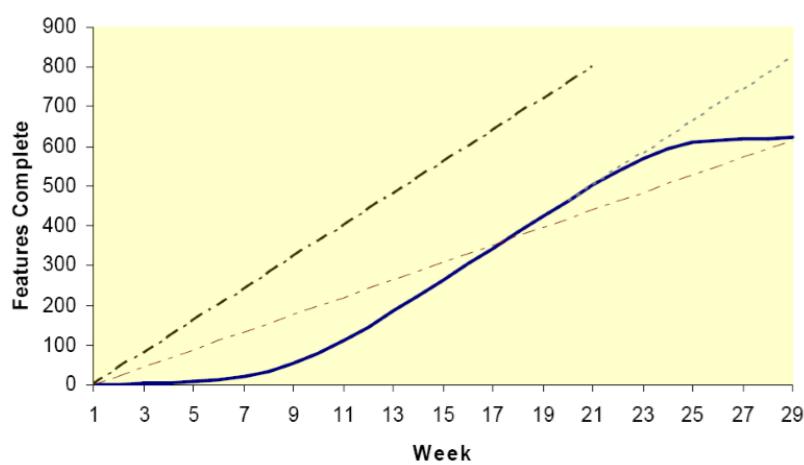
- Eli Goldratt ogłosił teorię ograniczeń – TOC (1990):
  - ogólne rozwiązanie związane z „wąskim gardłem” – ograniczeniem
  - wydajność ograniczenia decyduje o wydajności całego systemu
  - maksymalizacja przychodów, zysków, ROI poprzez inwestycje i zarządzanie ograniczeniami wzrostu
  - proces realizuje dynamikę zgodnie z krzywą-S

Pytanie

Czym jest krzywa S ?



## Krzywa S – [www.agilemanagement.net](http://www.agilemanagement.net)



Przedstawia ile featurów zostało zaimplementowanych w zależności od czasu. Na początku słabo, bo się rozgrzewamy, później już lepiej, a na końcu znów troszkę słabiej.

### 3.6. Wykład 6 - 6\_mpiso\_cocomo\_v0.1

#### Pytanie

Czym są algorytmiczne i analityczne metody szacowania oprogramowania ?

Modele algorytmiczne rozpoczyna się od opisania danego przedsięwzięcia za pomocą szeregu atrybutów. Atrybuty te mogą oddawać bardzo wiele aspektów projektu, przykładowo powszechność technologii, wsparcie narzędzi developerских, ograniczoność czasową itd.

Pytanie

Jakie znasz metody algorytmiczne i analityczne szacowania oprogramowania ?



## Metody algorytmiczne i analityczne

- COCOMO (COCOMO 81) – *constructive cost model*, Barry Boehm dla TWR Inc.
- COCOMO2 – rozszerzenie COCOMO o czynniki fazy, zaktualizowana baza danych projektów 1997
- Punkty funkcyjne – metryka funkcjonalności
- Punkty obiektowe – metryka obiektów funkcyjnych i wizualnych (nie klas)
- Przypadki użycia – metryka scenariuszy biznesowych
- Specjalne

COCOMO to skrót od COst COnstrucion MOdel.

Pytanie

Czym jest COCOMO ?

To metoda szacowania kosztów i czasu trwania projektu opierająca się na istnieniu zależności:

- wielkość projektu <-> koszt projektu
- koszt projektu <-> czas trwania projektu

Zależności te są wyrażane wzorami matematycznymi.

Metodę COCOMO można stosować w trzech wariantach (trzy poziomy hierarchii):

- Podstawowy
- Pośredni
- Szczegółowy

Pytanie

Na czym polega model podstawowy COCOMO ?



### Podstawowy COCOMO

- Model podstawowy jest statyczny
- Rozdziela projekty informatyczne na trzy grupy:
  - Łatwe (*organic*) – małe, proste projekty z małym zespołem (2-4 osoby), z dobrze określonymi, stałymi wymaganiem
  - Średnie (*semi-detached*) – średniozaawansowany zakres i średnia złożoność, brak wiedzy o części wymagań, brak wiedzy o części technologii
  - Trudne (*embedded*) – silne ograniczenia technologiczne i procesowe, duży poziom nieoznaczoności, state of art



### Obliczenie kosztów w modelu podstawowym

- Koszt projektu wyznacza się z równania:

$$\text{Effort} = a_b (\text{KLOC})^{b_b}$$

- Czas trwania projektu:

$$\text{Duration} = c_b \text{Effort}^{d_b}$$

- Ilość niezbędnych zasobów:

$$\text{Resource} = \frac{\text{Effort}}{\text{Duration}}$$

Jak widać potrzebne jest obliczenie linii kodu aby można było sobie wyznaczyć koszt projektu, a później czas trwania i ilość niezbędnych zasobów. Zauważmy, że model ten bazuje w zasadzie tylko na jednym parametrze, którym są linie kodu. Wszystkie pozostałe obliczane są na podstawie tego parametru.

## Parametry modelu podstawowego

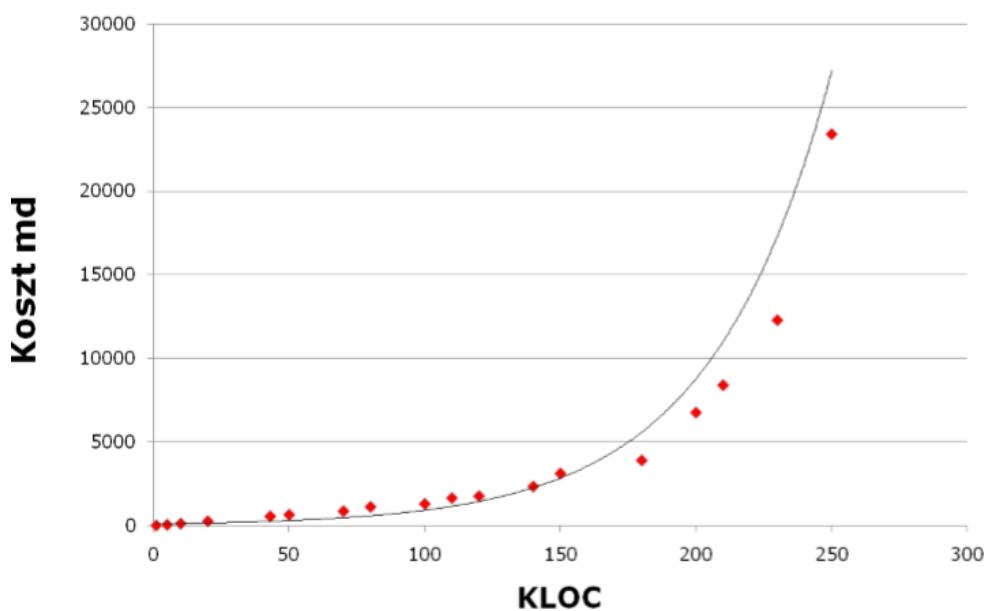
Rodzaj projektu	$a_b$	$b_b$	$c_b$	$d_b$
Łatwy	2,4	1,05	2,5	0,38
Średni	3,0	1,12	2,5	0,35
Trudny	3,6	1,20	2,5	0,32

## Pytanie

Do czego służy metoda regresji ?

W kontekście COCOMO, chodzi o to, że defaultowe parametry metody COCOMO ( $a_b, b_b, c_b, d_b$ ) mogą nie pasować do naszej organizacji. Wtedy bierzemy bazę danych historycznych i robimy regresję z tych danych.

### Dane projektów historycznych



Parametry a i b można odczytać z regresji, aby obliczyć pozostałe trzeba przejść na skalę logarytmiczną i zrobić jakieś tam obliczenia.

Pytanie

Na czym polega model pośredni COCOMO ? Czym jest EAF (Effort Adjustment Factor) ?

### Model pośredni COCOMO

- Wprowadza czynniki kosztowe (*cost drivers*):

- Atrybuty produktu:

- wymagana niezawodność
    - wielkość bazy danych
    - złożoność produktu

- Atrybuty sprzętu:

- ograniczenia czasowe (RTS)
    - ograniczenia pamięci operacyjnej
    - środowisko (zmienność) maszyny wirtualnej
    - czas przejścia do nowego systemu (migracja, itd.)



### Parametry modelu COCOMO cd.

- Atrybuty zasobów

- zdolności analityczne
    - zdolności inżynierii systemowej
    - doświadczenie w budowie aplikacji
    - doświadczenie w zakresie maszyny wirtualnej
    - doświadczenie w języku programowania

- Atrybuty projektu

- wsparcie narzędzi developerskich
    - wykorzystanie metod inżynierskich
    - harmonogram projektu

- Każdy atrybut ma określoną wagę

- Iloczyn wszystkich wag daje – **EAF** (*effort adjustment factor*)

Czyli przykładowo bierzemy sobie jakiś atrybut i ustalamy na jakim poziomie jest on wymagany, a następnie na tej podstawie dobieramy wagę z tabelki:

Atrybut	Bardzo mały	Mały	Normalny	Wysoki	Bardzo wysoki	Ekstra wysoki
zdolności analityczne	1,46	1,19	1	0,86	0,71	
zdolności inżynierii systemowej	1,29	1,13	1	0,91	0,82	
doświadczenie w budowie aplikacji	1,42	1,17	1	0,86	0,70	
doświadczenie w zakresie maszyny wirtualnej	1,21	1,10	1	0,90		
doświadczenie w języku programowania	1,14	1,07	1	0,95		
wsparcie narzędzi developerskich	1,24	1,10	1	0,91	0,82	
wykorzystanie metod inżynierskich	1,24	1,10	1	0,91	0,83	
harmonogram projektu	1,23	1,08	1	1,04	1,10	

Jak mamy wszystkie wagi, to obliczamy sobie EAF (effort adjustment factor) jako ich iloczyn, a później ze wzroków podobnie jak w modelu podstawowym:

### ↳ Obliczenie kosztów w modelu pośrednim

- Koszt projektu wyznacza się z równania:

$$\text{Effort} = a_i(\text{KLOC})^{b_i} * EAF$$

- Pozostałe parametry zgodnie z modelem podstawowym

Rodzaj projektu	$a_i$	$b_i$
Łatwy	3,2	1,05
Średni	3,0	1,12
Trudny	2,8	1,20

Pytanie

Na czym polega szczegółowy model COCOMO ?

Generalnie tak samo jak dla pośredniego, ale rozważamy fazy projektu.

### Szczegółowy model COCOMO

- Model szczegółowy stosowany jest do poszczególnych faz projektów
- Wykorzystuje te same parametry i atrybuty co model pośredni
- Wartość kosztów oraz czasu trwania oblicza się poprzez sumę odpowiednich składników
- Przykładowo dla podejścia kaskadowego:

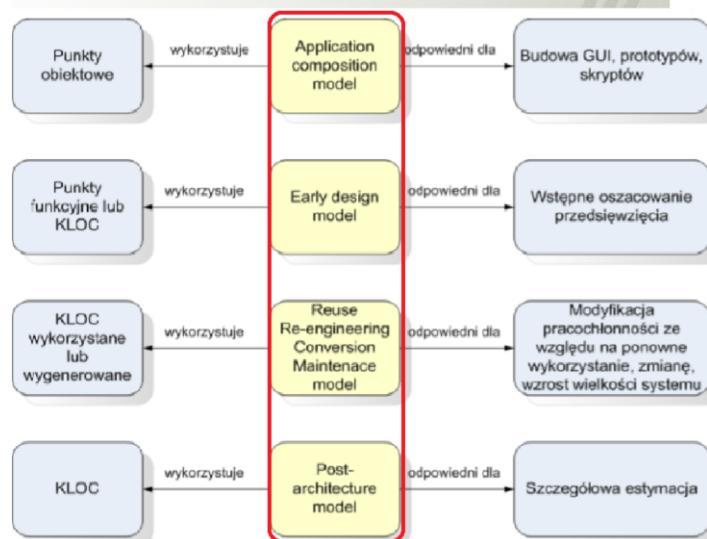
$$\text{Effort} = \sum_{i \in phases} \text{Effort}_i$$

Pytanie

Czym jest model COCOMO II ? Co nowego wprowadza w stosunku do COCOMO ?

Jest rozwinięciem modelu COCOMO. W istocie, COCOMO II nie jest pojedynczym modelem, ale hierarchią modeli podaną poniżej. To który z modeli wybieramy zależy np. od fazy projektu, tego co robimy itp. (przykłady na obrazku)

### Przegląd modeli COCOMO2



### Wstęp do COCOMO 2

- COCOMO 2 został opracowany na bazie projektów prowadzonych w latach 90-tych XX wieku
- Główne wytyczne:
  - Pozostaje otwarty jak COCOMO 81
  - Struktura modeli nakierowana na nowe rynki informatyki
  - Dane wejściowe i wyjściowe modeli dostosowane do posiadanych informacji
  - Możliwość dostosowania modeli do procesu produkcyjnego

Warto na tym obrazku zwrócić uwagę na to co jest wejściem, a co wyjściem z danego modelu. Na przykładzie bardziej szczegółowo są omówione Early design model oraz Post-architecture model.

Pytanie

Podaj kilka informacji o modelu Early Design Model z COCOMO II.

 **The Early Design Model**

- ▶ Pozwala na oszacowanie pracochłonności na wstępnych etapach projektu lub na początku iteracji
- ▶ Bazuje na metryce punktów funkcyjnych lub KLOC (punkty funkcyjne są przeliczane na KLOC)
- ▶ Może być wykorzystany we wszystkich typach projektów (Application generator, System integration, Infrastructure)
- ▶ Model wykorzystuje 7 współczynników pracochłonności

Pytanie

Jak obliczana jest pracochłonność w Early Design Model w COCOMO II ?

### Model matematyczny

- Pracochłonność nominalna:

$$\text{Effort}_{\text{nominal}} = A(\text{Size})^B$$



- Parametry:

- $A=2,94$  – współczynnik przeliczeniowy np.: KLOC-manday, FP-manday

- $B = \text{od } 1.1 \text{ do } 1.24$  – współczynnik ekonomiczny skali

- Pracochłonność zmodyfikowana:

$$\text{Effort}_{\text{adjusted}} = A(\text{Size})^B \prod_{i=1}^7 \text{EM}_i$$

Pytanie

Czym są współczynniki modyfikujące ?



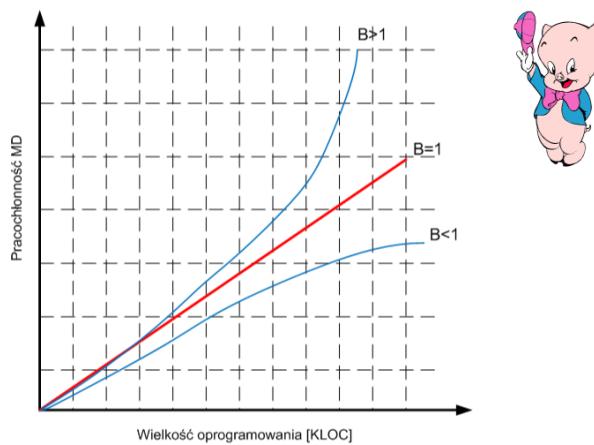
## Współczynniki modyfikujące

- Współczynniki odpowiadają wymaganiom poza-funkcjonalnym, doświadczeniu zespołu projektowego, dojrzałość środowiska i technologii:
  - RCPX – złożoność i niezawodność produktu
  - RUSE – wymagania ponownego wykorzystania
  - PDIF – złożoność środowiska i technologii
  - PREX – doświadczenie zespołu
  - PERS – potencjał zespołu
  - SCED - harmonogram
  - FCIL – rozmieszczenie i wsparcie zespołu projektowego

## Pytanie

Czym jest współczynnik ekonomii skali ? Jaki efekt jest z nim związany ?

- Współczynnik skali  $B$  reprezentuje zależność wzrostu pracochłonności wraz ze wzrostem wielkości oprogramowania
- W przypadku małych, przeciętnych projektów nie występuje efekt skali
- W dużych projektach efekt skali może występować:
  - efekt ekonomii – automatyzacja często powtarzanych czynności, stworzenie narzędzi, itd.
  - efekt dysekonomii – zwiększone koszty integracji, zarządzania komunikacją itd.

Wpływ wartości  $B$  na model

25

Mrówka natomiast na wykładzie zwrócił uwagę na tą linię pod czerwoną kreską, bo przykłady takie, że wraz z wielkością oprogramowania rośnie pracochłonność są oczywiste. Do tej kreski poniżej dla  $B < 1$  podał takie przykłady:

- Biznesowy - tzn. ekonomia skali, czyli zrobić tak aby każda sprzedaż aplikacji kosztowała nas coraz mniej

- Techniczny - zaprojektować sobie szablony aplikacji w taki sposób aby móc je później łatwo wykorzystywać (wzorce projektowe, itp.)

### Efekt skali w modelu COCOMO 2

- Współczynnik B wyznaczamy jako:

$$B = 0,91 * 0,01 \sum_{i=1}^5 W_i$$

- Parametry:  $W_i$  – współczynniki skali
- Współczynniki skali reprezentują obszary projektu mające wpływ na efekt skali
- Dla modelu *Early Desing* oraz *Post-Architecture* przyjmuje się 5 współczynników

#### Pytanie

Podaj kilka ogólnych informacji o modelu Post-Architecture Model w COCOMO II

### Post-Architecture Model

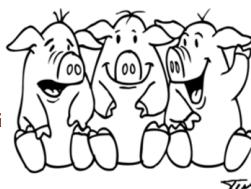
- Post-Architektura jest najbardziej szczegółowym modelem COCOMO2
- Stosowany w przypadku, gdy znana jest architektura systemu
- Model pozwala na oszacowanie wytworzenia oraz utrzymania systemu
- Stosowalny dla wszystkich trzech obszarów rynku
- Wykorzystuje 17 współczynników kosztów



## Post-Architecture – model matematyczny

- Długość harmonogramu:

$$\text{Effort}_{\text{adjusted}} = A(\text{Size})^B \prod_{i=1}^{17} \text{EM}_i$$



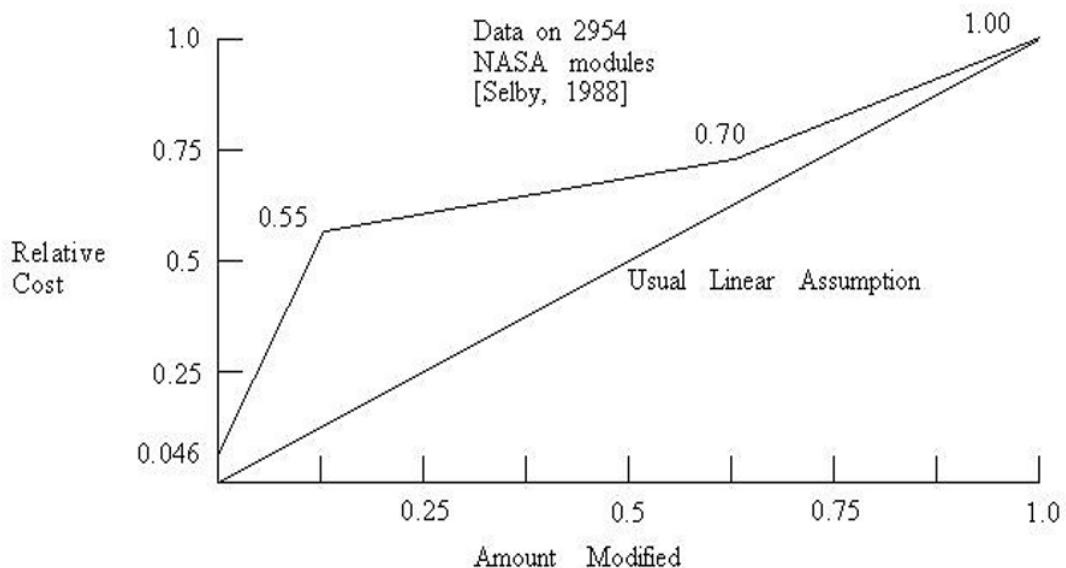
- Parametry: SCED – współczynnik kompresji harmonogramu (jeden z 17 współczynników kosztów)

$$TDEV = [3,67 * \overline{PM}^{0,28+0,2*(B-1,01)}] * \frac{\text{SCED}\%}{100}$$

Pytanie

Czy ponowne wykorzystanie kodu przynosi oszczędności?

Często nie, ponieważ np. potrzebujemy czasu na zaznajomienie się z kodem. To jest obrazowane na wykresie poniżej jako takie kolanko:



### 3.7. Wykład 7 - 7\_mpiso\_functionalpoints\_v0.1

#### Pytanie

Dla jakich aplikacji będzie działać metoda punktów funkcyjnych ?

Dla małych prostych aplikacji, gdzie nie mamy skomplikowanego back endu i algorytmów. Klasyczne aplikacji typu widok + baza danych. Nie używa się punktów funkcyjnych dla liczeniu kosztów utrzymania. Punkty funkcyjne to metoda pomiaru wielkości oprogramowania

#### Pytanie

Jak mierzmy produktywność w metodzie punktów funkcyjnych ?

$$\frac{FP}{mandays}$$

#### Pytanie

Wymień i omów kroki metody punktów funkcyjnych

1. Określenie rodzaju FP

- Implementacyjne - budowane od zera
- Rozszerzeń - rozbudowa istniejącego systemu
- Aplikacji - mam gotową apkę i liczę ile jest tam punktów funkcyjnych np. po to aby obliczyć produktywność

#### Pytanie

Podaj rodzaje punktów funkcyjnych

2. Zdefiniowanie zakresu systemu - czy coś wchodzi w zakres prac nad systemem czy nie, np. otwierając okno wybierania pliku w word wyświetla się natywne wyszukiwanie windows i nie jest to częścią systemu, który budujemy

3. Identyfikacja kategorii funkcjonalnych

#### Pytanie

Czym są EI, EO, EQ ?

#### Pytanie

Czym są ILF, EIF ?

- EI - wprowadzanie danych wejściowych w formularzach
- EO - wyświetlanie danych
- EQ (Kwerendy zewnętrzne)- pełni obie funkcje, wprowadzanie i wyświetlanie

## METODY POMIARU I SZACOWANIA OPROGRAMOWANIA

- ILF - upraszczając to liczba tabel w bazie danych lub odpowiednik tabeli
- EIF - tak samo jak ILF, ale nie jest w granicy systemu, np. jakiś web service

Wszystkie powyższe kategorie mogą mieć złożoność niską, średnią lub wysoką. Złożoność określana jest na podstawie FTR oraz DET dla EI, EO oraz EQ, natomiast dla ILF oraz EIF złożoność obliczana jest na podstawie DEF oraz RET.

### Pytanie

Czym są DET, FTR, DEF, RET ?

- DET - kontrolka na GUI, link, radio button, zakładka, itp. coś w co można kliknąć
- FTR - liczba ILF, które są przechowywane w IE, EO, EQ, czyli do ilu tabel sięgamy
- DEF - odpowiednik liczby kolumn w tabeli
- RET - odpowiednik relacji pomiędzy FTR, np. jeden do wielu

### Pytanie

Negatywne przykłady EI (jest w opracowaniu na wiki pytanie z tego dlatego zamieszczam)

### Przykłady negatywne

- ▶ Dane nie utrzymywane przez system (pobierane jako EIF)
- ▶ Dane selekcji z zewnętrznej kwerendy
- ▶ Menu nawigacyjne
- ▶ Ekran logowania, który umożliwia zalogowanie do systemu, ale nie utrzymuje ILF

### Pytanie

Przykłady EQ

### Przykłady EQ

- Wyszukiwanie danych na podstawie danych wejściowych (kryteriów)
- Ekran logowania do systemu liczymy jako EQ (zapytanie o login i hasło)
- Lista „ostatnio otwarte” w menu
- Dynamiczne pop-menu
- System pomocy kontekstowej (wyszukiwanie po słowach kluczowych)

Następnie sobie sięgamy do tabelek takich jak ta aby określić czy element ma złożoność niską, średnią czy wysoką:

### Klasyfikacja EI (złożoności)

FTR	DET		
	1-4	5-15	>15
<2	Niska(3)	Niska(3)	Średnia(4)
2	Niska(3)	Średnia(4)	Wysoka(6)
>2	Średnia(4)	Wysoka(6)	Wysoka(6)

### Klasyfikacja EIF (złożoności)

RET	DEF		
	1-19	20-50	>50
1	Niska(5)	Niska(5)	Średnia(7)
2-5	Niska(5)	Średnia(7)	Wysoka(10)
>5	Średnia(7)	Wysoka(10)	Wysoka(10)

4. Zliczenie funkcji danych

### Krok 4 – Zliczenie funkcji danych

- Zliczamy ILE oraz EIF zidentyfikowanych w kroku 3 w systemie
- Niekiedy warto wspomóc się diagramem encji (w przypadku istniejących systemów)
- Dla każdej encji (typu danych) należy podjąć decyzję w zakresie utrzymywania tej encji – wewnętrz (ILE), na zewnątrz (EIF)

5. Zliczanie funkcji transakcyjnych



## Krok 5 – Zliczenie funkcji transakcyjnych

- Zliczenie EI, EO oraz EQ
- EI jest elementem funkcyjnym, a nie częścią fizycznego przepływu danych
- EO określa proces, który może obejmować: kalkulacje, wyznaczenie nowych danych, aktualizacja ILF lub bezpośrednie działanie aplikacji
- EQ zwrot danych na podstawie ILF oraz EIF

6. Obliczenie współczynnika VAF



## Krok 6 - Współczynnik modyfikujący VAF

- Wyrażenie określa współczynnik modyfikacji:

$$VAF = 0,65 + \left( \sum_{i=1}^{14} C_i \right) / 100$$

- $C_i$  – wartości wynikające z charakterystyki systemu



## Współczynniki $C_i$

Lp.	Nawa	Opis
1	Komunikacja	Ile ośrodków należy połączyć w celu wymiany danych
2	Przetwarzanie rozproszone	Ile danych rozproszonych i funkcji należy przetwarzać
3	Wydajność	Czy użytkownik wymaga maksymalnych czasów odpowiedzi?
4	Obciążenie środowiska	Jak bardzo wykorzystane jest obecne środowisko sprzętowe
5	Ilość transakcji	Jak często transakcje są przetwarzane

7. Obliczanie UFP



## Krok 7 - Obliczanie UFP

Moduł	Złożoność			Suma
	Niska	Średnia	Wysoka	
EI	_x3=_	_x4=_	_x6=_	
EO	_x4=_	_x5=_	_x7=_	
EQ	_x3=_	_x4=_	_x6=_	
ILF	_x7=_	_x10=_	_x15=_	
ELF	_x5=_	_x7=_	_x10=_	
Całkowita wartość nieskorygowanych FP				

Jak już mamy punkty UFP, to teraz w zależności od typu punktów funkcyjnych (jak w kroku pierwszym) obliczamy FP z różnych wzorów, przykładowo dla aplikacji pisanej od zera to:



## Deweloperskie punkty funkcyjne

- Dla nowych projektów ilość skorygowanych punktów funkcyjnych wyznaczamy jako:

$$\mathbf{FP = (UFP + CFP)*VAF}$$

- CFP – liczba skorygowanych punktów funkcyjnych dla aplikacji konwertującej

### 3.8. Wykład 8 - mpiso\_qualitymetrics\_v0.1

#### Pytanie

Czym jest jakość oprogramowania ?

Istnieją formalne definicje jakości oprogramowania: wąska (klasyczna) w dwóch wersjach i szeroka

- wąska (klasyczna), wersja I - jakość oprogramowania to częstość defektów na liczbę możliwości pełnienia tego defektu. Jakość definiujemy jako  $\frac{LD}{M}$ , gdzie licznik to Liczba Defektów, a mianownik M dobieramy tak aby mieć jakąś metrykę wielkości systemu, np. punkty funkcyjne, KLOC itp.
- wąska (klasyczna), wersja II - definiuje jakość w sensie niezawodności, czyli mówimy np., że system będzie działał bez defektów przez 10000 godzin, gdzie przez defekty rozumiemy jakiekolwiek, nawet najmniejsze usterki.
- szeroka - stosowana jest współcześnie, skupia się na satysfakcji klienta, obejmuje całokształt użytkowania systemu przez klienta, a więc dokumentację, wygląd, wygodę użytku, jakość ta mierzona jest zazwyczaj przez ankietyzację, analizę komentarzy/lajków itp.

#### Pytanie

Czym jest podejście klienckie ?

To podejście, które skupia się na wymaganiach klienta oraz na jego ogólnym zadowoleniu. Zakłada ono zapewnienie sprawnego serwisu i dystrybucji produktu, a także ciągłe nastawienie na doskonalenie jakości.

#### Pytanie Z wykładu - nie ma na slajdach

Jak coś sprzedawać ? Jak ustawić cenę ?

Sprzedaż opierać się powinna na przedstawieniu klientowi, że rzeczywiście potrzebuje produktu. Cena powinna być zdefiniowana na podstawie tego jaką wartość ma produkt dla klienta, a nie na podstawie kosztu wytworzenia.

#### Pytanie

Czym jest TQM ? (Kompleksowe zarządzanie jakością)

W ogólności jest to proces tworzenia produktu w taki sposób aby dopilnować, że produkt będzie miał wysoką jakość i klient będzie zadowolony (orientacja na potrzeby klienta, usprawnianie procesu).

A bardziej od praktycznej strony (to na co Mrówka kładł największą uwagę) to chodzi o to aby w organizacji wytworzyć taką atmosferę żeby wszyscy dzielili się swoimi pomysłami względem poprawy jakości procesów, żeby starali się zrobić wszystko aby firma się lepiej rozwijała. Można to osiągnąć przez motywowanie pieniędzmi, ale nie może to być jedyny czynnik, bo podwyżki szybko się nudzą, trzeba więc zbadać motywację ludzi z którymi pracujemy.

### Pytanie

Jaki jest podział metryk jakości oprogramowania ?

Metryki jakości oprogramowania dzielimy na:

- Metryki jakości produktu - mierzą jakość produktu końcowego
- Metryki jakości procesu - pomiar jakości samego procesu wytwarzania, np. jaka jest efektywność usuwania bugów pomiędzy kolejnymi fazami projektowymi
- Metryki serwisowe - czyli jak dobry jest serwis zapewniany do produktu, np. jakie jest opóźnienie napraw defektów

### Pytanie

Podaj przykłady metryk jakości oprogramowania dla każdej z metryk.

- Jakości produktu
  1. częstość defektów
  2. średni czas pracy bezawaryjnej
  3. problemy klienta (zgłoszenia klienta)
  4. satysfakcja klienta
- Jakości procesu
  1. częstość defektów podczas testów maszynowych
  2. efektywność usuwania defektów
- Metryki serwisowe
  1. opóźnienie napraw

### Pytanie bez sensu...

Czym różnią się błąd, usterka, defekt i awaria ?

- Błąd - jest wynikiem nieprawidłowej działalności człowieka
- Usterka - przypadkowy warunek
- Defekt - anomalia w produkcie, defekt to jakikolwiek Błąd lub Usterka
- Awaria - gdy system w ogóle lub częściowo przestaje działać

Pytanie

Omów metryki produktowe jakości oprogramowania

- Metryka częstości defektów

### Metryka częstości defektów

- Bazuje na relacji:

*Liczba defektów / Liczba sposobności do popełnienia błędów (OFE)*

- OFE możemy zdefiniować jako:

- Liczba linii kodu (KLOC, f-LOC, itd.)
- Punkty funkcyjne

- Przykładowo dla punktów funkcyjnych podaje się następujące wartości dla modelu CMM:

- Poziom 1: 0,75
- Poziom 2: 0,44
- Poziom 3: 0,27
- **Poziom 4: 0,14**

- Metryka zgłoszeń klienta

### Metryka problemów (zgłoszeń) klienta

- Polega na zliczaniu wszystkich zgłoszeń (nie tylko defektów) klienta

- Problemy mogą wynikać:

- Brak/niejasność dokumentacji
- Trudność obsługi
- Brak znajomości obejść defektów
- Błędy użytkownika

- Wprowadzamy PUM (Problems per user month):

$$PUM = \frac{\text{Liczba zgłoszeń}}{(\text{Liczba wydanych licencji} \times \text{liczba miesięcy w danym okresie})}$$

- Metryki zadowolenia klienta

### Metryki zadowolenia klienta

- Najczęściej opiera się na 5-stopniowej skali:
  - Bardzo zadowolony
  - Zadowolony
  - Neutralny
  - Niezadowolony
  - Bardzo niezadowolony
- Bada się wiele aspektów oprogramowania:
  - Funkcjonalność
  - Łatwość obsługi
  - Wydajność
  - Niezawodność
  - Serwis

#### Pytanie

Omów metryki procesowe jakości oprogramowania

### Wewnętrzprocesowe metryki jakości

- Częstość defektów w testach maszynowych
  - Opiera się na wyznaczaniu ilość defektów na liczbę linii kodu w testach automatycznych
  - Zmiany wartości metryki w kolejnych iteracjach prowadzi do analizy zjawiska:
    - Jeżeli liczba defektów się zmniejszyła, to zadajemy pytanie: czy testy były tak samo dokładne? Jeżeli odpowiedź jest „tak” → test pozytywny, „nie” → konieczne dodatkowe testy
    - Jeżeli liczba defektów jest większa, pytanie brzmi: czy rzeczywiście poprawiliśmy efektywność testów?: jeżeli „nie” → prognoza jakości jest negatywna – dodatkowe testy, „tak” → utrzymać aktualne testy



## Efektywność usuwania defektów

- Definiujemy zależność:

$$\text{Efektywność usuwania defektów} = \frac{\text{defekty usunięte w fazie produkcji}}{\text{defekty ukryte w produkcie}} \cdot 100\%$$

- Mianownik najczęściej szacujemy jako:

*defekty usunięte w czasie trwania fazy + defekty znalezione później*



## Macierz danych o defektach

		Faza wprowadzenia defektu						
		Analiza	Projekt wstępny	Projekt szczegółowy	Testy Implementacja wewnętrzne	Testy akceptacyjne	Utrzymanie	SUMA
	Analiza	-						-
	Projekt wstępny	30	130					160
	Projekt szczegółowy	5	42	234				281
Faza wykrycia	Implementacja	12	37	543	932			1524
	Testy wewnętrzne	19	23	104	230	3		379
	Testy akceptacyjne	15	41	34	112	-	2	204
	Utrzymanie	8	16	40	72	-	-	140
	SUMA	89	289	955	1346	3	2	4

Pytanie

Czym jest reguła 1:10 w usuwania defektów jakości oprogramowania ?

### Koszty usuwania defektów

- Ogólna reguła: wcześniejsze usuwanie defektów jest tańsze (regułą 1:10)
- Bada IBM wykazały, że czasem usunięcie błędu w następnej fazie jest tańsze (projekt ogólny – projekt szczegółowy)
- Należy uwzględnić koszty tworzenia raportu defektów (malej wraz z późniejszą fazą)

Pytanie

Omów metryki serwisowe jakości oprogramowania

### Metryki serwisowe oprogramowania

- Opóźnienie naprawy i wskaźnik zarządzania opóźnieniem

$$BMI = \frac{\text{Liczba problemów rozwiązywanych w ciągu miesiąca}}{\text{Liczba problemów zgłoszonych w ciągu miesiąca}} \cdot 100\%$$

- Jeżeli BMI > 100% to opóźnienia są redukowane
- Jeżeli BMI < 100% to opóźnienia narastają
- Definiuje się limity kontrolne dla BMI: UCL oraz LCL



## Pozostałe metryki serwisu

- Czas odpowiedzi na zgłoszenie (średni czas rozwiązania problemu)
- Odsetek źle wykonanych napraw:  
$$\text{odsetek źle wykonanych napraw} = \frac{\text{liczba napraw z przekroczonym czasem naprawy}}{\text{liczba wszystkich napraw}} \cdot 100\%$$
- Jakość naprawy – liczba źle wykonanych napraw (powracających problemów)

### Pytanie

Czym jest model niezawodności ?



## Model niezawodności

- Wprowadza się w celu oceny niezawodności lub w celu **oszacowania ukrytych defektów**
- Oszacowanie ukrytych defektów może być podstawa do wyliczenia kosztów serwisu
- Badana zmienna: liczba błędów w danym przedziale czasu
- Dwie kategorie modeli: statyczne i dynamiczne:
  - Statyczne – parametry projektu
  - Dynamiczne – wykorzystuje statystyczne dane jako wzorce na podstawie już wykrytych błędów

### **3.9. Pytania dodatkowe wiki.stosowana opracowanie**

Warto też zająrzeć do opracowania na wiki.stosowana z przykładowymi pytaniami.