Inżynieria oprogramowania

Wykład 8:

Zarządzanie przedsięwzięciem informatycznym

Marek Krętowski
pokój 206
e-mail: m.kretowski@pb.edu.pl
http://aragorn.pb.bialystok.pl/~mkret

Wersja 1.12

Zarządzanie przedsięwzięciem

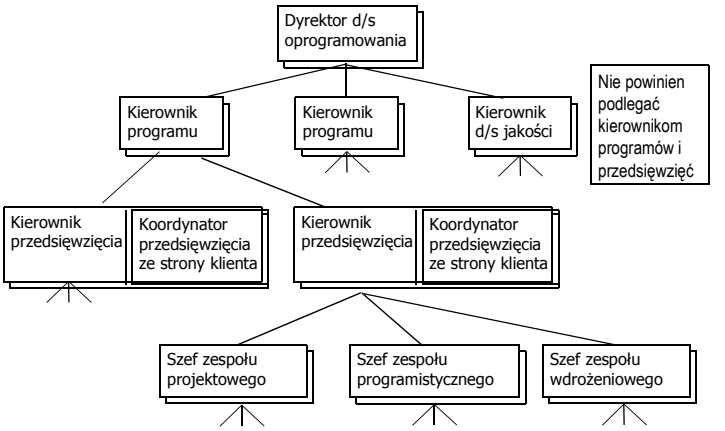
Zatrudnienie wysokiej klasy specjalistów oraz stosowanie zaawansowanych narzędzi wspomagających nie gwarantuje jeszcze sukcesu projektu; niezbędne jest właściwe zarządzanie przedsięwzięciem

Podstawowe zadania kierownictwa przedsięwzięcia programistycznego:

- opracowanie propozycji dotyczących sposobu prowadzenia przedsięwzięcia,
- kosztorysowanie przedsięwzięcia i jego wycena
- planowanie i harmonogramowanie przedsięwzięcia,
- monitorowanie i kontrolowanie realizacji przedsięwzięcia,
- dobór i ocena personelu,
- opracowanie i prezentowanie sprawozdań dla kierownictwa wyższego szczebla.

Sposoby zarządzania przedsięwzięciem programistycznym w wielu aspektach nie różnią się od zarządzania innymi przedsięwzięciami, ale muszą brać pod uwagę specyfikę procesu budowy oprogramowania (np. nieprzejrzystość procesu)

Struktura zarządzania firmą programistyczną



Funkcje osób pracujących nad oprogramowaniem

- Kierownik przedsięwzięcia
- Analityk - osoba bezpośrednio kontaktująca się z klientem, której celem jest określenie wymagań i budowa modelu systemu
- Projektant - osoba odpowiedzialna za realizację oprogramowania; może posiadać wyspecjalizowane funkcje:
  - projektant interfejsu użytkownika
  - projektant bazy danych
- Programista (implementacja)
- Tester (testowanie)
- Twórca dokumentacji użytkownika
- Ekspert metodyczny - osoba szczególnie dobrze znająca stosowaną metodykę
- Ekspert techniczny - osoba szczególnie dobrze znająca sprzęt i narzędzia

W mniejszych przedsięwzięciach jedna osoba może pełnić wiele funkcji, często rozważane są następujące modele:

- **analityk/projektant + programista**: funkcje analizy i projektu w jednych rękach; zakłada wysoki poziom projektanta, natomiast nie wymaga zbyt wiele od programistów (funkcje programisty dość niskiego poziomu); w warunkach polskich model nie zdaje raczej egzaminu
- **analityk + projektant/programista**: model bardziej realistyczny; zakłada znacznie wyższy poziom przygotowania programisty

Kierownik projektu

Zadaniem szefa nie jest wykonywanie pracy za podległych mu pracowników, lecz dbanie aby wykonywali oni swoją pracę; zadaniem szefa nie jest zmuszanie ludzi do pracy, ale umożliwienie im tego

Pożądane cechy kierownika:

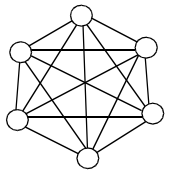
- **Zdolność do przewidywania** - umiejętność dostrzegania drobnych spraw, które mogą być zalążkiem poważniejszych problemów, przewidywania skutków, podejmowanie zawczasu akcji naprawczych
- **Umiejętność motywowania** - zdolność do pobudzania poczucia przynależności do grupy, zainteresowania wykonywaną pracą, utożsamiania się z projektem, atmosfery współpracy w zespole (przywództwo i budowanie zespołu)
- **Zdolność do przystosowania się do zmieniającej się sytuacji** - umiejętność podjęcia niezbędnych działań mających na celu przystosowanie zespołu i jego metod pracy do zmienionych warunków

Pożądane cechy kierownika (cd)

- **Zdolność do przekonania otoczenia do własnych możliwości i wartości** - kierownik powinien wzbudzać zaufanie; sukces zespołu zależy w dużej mierze od cech przywódczych szefa i poważania i zaufania jakim się cieszy wśród członków grupy
- **Rozpoznawanie i rozwijanie potencjału swoich współpracowników** - naturalnym zjawiskiem jest wymiana personelu (np. dobrze wyszkoleni i kompetentni pracownicy awansują lub zmieniają pracę); powstaje potrzeba wprowadzania do zespołu nowych ludzi, właściwego ich szkolenia, pomocy w rozwijaniu ich potencjału
- **Komunikatywność** - łatwość komunikownia się z szerokim wachlarzem ludzi - podwładni, kierownictwo, klienci, dostawcy, ...
- **Terminowe podejmowanie decyzji dostosowanych do bieżącej sytuacji i potrzeb** - kierownik powinien precyzyjnie określić co ma być zrobione i w jakim terminie; wiąże się to często z podejmowaniem decyzji w sytuacji niepełnej wiedzy.

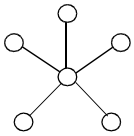
Organizacja zespołu

Struktura sieciowa - każdy komunikuje się i współpracuje z pozostałymi (nie powinna być zbyt liczna); zespoły o podobnym doświadczeniu i stopniu zaawansowania; zalety:



- dzięki ścisłej współpracy członkowie zespołu wzajemnie kontrolują swoją pracę; szybko osiągane są standardy jakości
- umożliwia realizację idei wspólnego programowania
- ponieważ praca członków zespołu jest znana dla innych członków, łatwo mogą oni przejąć obowiązki pracownika, który opuścił zespół

Struktura gwiaździsta - szef zespołu jest jedyną osobą ściśle współpracującą z pozostałymi osobami; przydziela zadania i kontroluje efekty; komunikacja pomiędzy członkami zespołu poprzez szefa; jest przydatna wtedy, gdy w skład zespołu wchodzi wielu niedoświadczonych pracowników; wielkość zespołu może być znacznie większa niż w strukturze sieciowej (ogranicza ją jedynie zdolność szefa do równoczesnego kierowania dużą grupą); najpoważniejszą wadą jest trudność zastąpienia szefa w momencie jego odejścia



Inne struktury organizacyjne

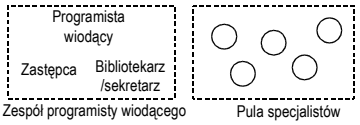
Struktura macierzowa

- specjaliści pogrupowani w zespoły kompetencyjne i przydzielani do wykonania konkretnych zadań w ramach projektów
- ekonomiczne wykorzystanie zasobów ludzkich
- podstawowa wada: dwupodległość (szef zespołu i szef projektu)

Zespół kompetencyjny \ Projekt	P1	P2	P3
Analitycy	2	2	1
Zespół baz danych	1	1.5	0.5
Zespół internetowy	3	3	-
Testerzy	0.5	1	0.5

Zespół programisty wiodącego:

- programista wiodący (wybitne kwalifikacje):
  - wymyśla koncepcję i specyfikuje zadania
  - sam realizuje najważniejsze zadania
  - przydziela zadania puli specjalistów
- zastępca (dobrze wykwalifikowana osoba)
  - na bieżąco, biernie uczestniczy w pracy szefa (w każdej chwili może go zastąpić)
  - może przygotowywać testy
- bibliotekarz/sekretarz - odpowiada za dokumentację i komunikację w zespole



Pożądane cechy inżyniera oprogramowania

- **Umiejętność pracy w stresie** - w procesie budowy i konserwacji oprogramowania często zdarzają się okresy wymagające szybkiego wykonania złożonych zadań; dla większości osób niewielki stres działa mobilizująco, ale po przekroczeniu pewnego progu następuje spadek możliwości danej osoby (próg ten jest bardzo różny dla różnych osób); należy pamiętać, że w stresie nie pracuje się lepiej a jedynie szybciej.
- **Zdolności adaptacyjne** - informatyka jest jedną z najszybciej zmieniających się dziedzin; ocenia się, że 7-9 miesięcy przynosi w informatyce zmiany, które w innych bardziej tradycyjnych dziedzinach zajmują 5-7 lat; narzuca to konieczność stałego dokształcania się (poznawanie nowych narzędzi, sprzętu, oprogramowania, technologii, metod, sposobów pracy), gdyż stosunkowo szybko można „wypaść z gry o najwyższe stawki"(uśpienie - zajmowanie się jednym problemem w jednym środowisku przez lata może mieć nieprzyjemne konsekwencje); z drugiej strony wiele osób nie wytrzymuje zawrotnego tempa (tzw. wypalanie się osób)

Nastawienie do pracy w zespole

Czynniki psychologiczne mają zasadniczy wpływ na efektywność pracy zespołu. Wyróżnia się następujące typy osób:

- **Zorientowani na zadanie** (ang. *task-oriented*) - osoby samowystarczalne, zdolne, zamknięte, agresywne, lubiące współzawodnictwo, niezależne; są zwykle efektywne, o ile pracują w pojedynkę, natomiast zespół złożony tylko z takich osób może być jednak nieefektywny (zbyt wiele indywidualności - rywalizacja); jeśli nie są doceniani mogą przeorientować się na siebie
- **Zorientowani na siebie** (ang. *self-oriented*) - osoby niezgodne, dogmatyczne, agresywne, zamknięte, lubiące współzawodnictwo, zazdrosne; mogą być efektywne w zespole pod warunkiem odpowiedniego motywowania przez kierownictwo
- **Zorientowani na interakcję** (ang. *interaction-oriented*) - osoby nieagresywne, o niewielkiej potrzebie autonomii i indywidualnych osiągnięć, pomocne, przyjazne; zespoły złożone z tego typu osób pracują najefektywniej; szczególnie potrzebni we wstępnych fazach projektu podczas kontaktów z klientami

Ergonomia pracy

- Zamiast dużej hali („amerykańskie biuro"), lepsze wyniki daje umieszczenia 2-3 stanowisk pracy w wielu mniejszych pomieszczeniach
- Umożliwienie personalizacji stanowiska pracy
- Zdrowe stanowisko pracy (prawidłowe oświetlenie - okna, ...)
- Pokój zebrań dla organizowania formalnych spotkań pracowników; zakaz dyskusji w pokojach pracy
- Miejsce dla spożywania posiłków, przerw relaksujących w pracy, spotkań nieformalnych (np. omówienie spraw przy kawie)
- Poczucie pracy na nowoczesnym sprzęcie (wydajność i chęć ludzi do pracy gwałtownie spada, jeżeli odczuwają oni, że pracują na przestarzałym sprzęcie - nawet wtedy, gdy wymiana sprzętu jest merytorycznie nieuzasadniona)
- Komfort psychiczny, właściwa atmosfera w pracy, eliminacja napięć i zadrażnień, nie dopuszczanie do rozmycia odpowiedzialności, sprawiedliwa ocena wyników pracy poszczególnych członków zespołu, równomierny rozkład zadań.

Harmonogramowanie przedsięwzięć

Układanie planu realizacji przedsięwzięcia polega na:

- ustaleniu kalendarza prac:
  - daty rozpoczęcia przedsięwzięcia
  - dni roboczych i wolnych w przewidywanym okresie realizacji przedsięwzięcia
  - czasu pracy w poszczególnych dniach
- podziale przedsięwzięcia na poszczególne zadania,
- określenie parametrów zadań,
- określenie zasobów niezbędnych do realizacji poszczególnych zadań,
- ustaleniu dostępności zasobów,
- ustaleniu kolejności i czasów wykonania poszczególnych zadań.

Przedsięwzięcie powinno być podzielone na stosunkowo małe zadania (realizacja do kilku dni), których parametry można łatwo określić; harmonogramowanie można wykonywać z różnym stopniem dokładności (najbliższe zadania szczegółowo; późniejsze bardziej ogólnie)

Harmonogramowanie (2)

Po ustaleniu zadań konieczne jest określenie parametrów czasowych:

- czasu wykonania,
- najwcześniejszy możliwy termin rozpoczęcia (np. niezbędne są pewne dane, które mogą być dostarczone dopiero ...),
- pożądany czas zakończenia,

oraz innych ograniczeń kolejności wykonywania prac (np. aby rozpocząć nowe zadanie niezbędne jest zakończenie pewnych innych zadań); można wykorzystać do tego diagramy typu PERT

Pewne zadania mogą zostać opóźnione bez wpływu na termin zakończenia całego przedsięwzięcia, inne natomiast tzw. zadania krytyczne nie mogą zostać opóźnione; ich ciąg definiujący termin zakończenia nazywany jest ścieżką krytyczną

Układając harmonogram należy brać pod uwagę dostępność zasobów niezbędnych do realizacji poszczególnych zadań (głównie zasoby ludzkie, ale może być również sprzęt i oprogramowanie)

Monitorowanie

Monitorowanie polega na śledzeniu przebiegu realizacji przedsięwzięcia oraz reagowaniu na pojawiające się problemy; następujące czynności:

- obserwacja rzeczywistych terminów rozpoczęcia i zakończenia zadań oraz porównywanie ich z zaplanowanymi,
- identyfikacja przyczyn niezgodności z planem,
- modyfikacja harmonogramu, jeżeli różnice pomiędzy harmonogramem a rzeczywistym przebiegiem realizacji są zbyt duże,
- obserwacja wykorzystania zasobów oraz rozstrzyganie konfliktów zasobowych,
- przesuwanie zasobów pomiędzy zadaniami oraz przedsięwzięciami.

Szczególną uwagę należy zwracać na zadania krytyczne i na zadania ze stosunkowo niewielkim luzem

Ekonomiczne aspekty działalności firmy

- O ile celem inżynierii oprogramowania jest tworzenie dobrego oprogramowania, o tyle celem firmy programistycznej jest po prostu zarabianie pieniędzy
- Jakość produktu jest tylko jednym z czynników wpływających na wynik ekonomiczny firmy; inne istotne aspekty:
  - reklama i promocja produktu,
  - renoma i zaufanie do producenta,
  - rodzaj i zakres gwarancji oraz innych usług dla klientów,
  - przyzwyczajenia klientów,
  - sposób wyceny rozmaitych wersji produktu,
  - sposób rozwoju produktu, polityka uaktualnień,
  - efektywność sposobu pozyskiwania klientów lub dystrybucji produktu.
- Czynniki te pozwalają redukować wpływ niższej jakości produktów danej firmy; wydaje się jednak, że wiele firm zwraca zbyt wielką uwagę na działalność marketingową zaniedbując kwestie podstawowe

Zarządzanie projektami

- Projekt – niepowtarzalne, (zwykle) złożone przedsięwzięcie do zrealizowania w ograniczonym czasie
  - zaangażowanie ograniczonych środków (środki, ludzie, ...)
  - związana z ryzykiem (technicznym, ekonomicznym, organizacyjnym)
- Popularne metodyki/metody/narzędzia zarządzania projektami:
  - PRINCE2 (PRoject IN Controlled Environment)
  - PMI/PMBoK (Project Management Body of Knowledge)
  - PCM (Project Cycle Management)
  - Wytyczne kompetencyjne IPMA (International Project Management Association)
  - ...

PRINCE2 – Project IN Controlled Environment

- Koncentruje się na sposobach podejmowania decyzji w projekcie i zarządzaniu realizacją
- Struktura PRINCE2 obejmuje:
  - **zasady** (pryncypia) – nakazy przewodnie i dobre praktyki
  - **tematy** – kluczowe aspekty zarządzania, odpowiadające na pytania:
    „*Dlaczego?"* (uzasadnienie),
    „*Kto?"* (organizacja),
    „*Co?"* (jakość),
    „*Jak? Za ile? Kiedy?"* (plany),
    „Co, jeżeli?" (ryzyko),
    „Jaki jest wpływ" (zmiana),
    „Gdzie jesteśmy i dokąd zmierzamy?" (postępy)
  - **procesy** – opisują krok po kroku działania w ramach cyklu życia projektu. Każdy proces dostarcza listy kontrolne zalecanych czynności, produkty zarządcze oraz związane z nimi obowiązki
  - **dostosowanie środowiska** – elastyczna struktura umożliwia dostosowanie do konkretnego (kontekstu, rodzaju, czy wielkości) projektu
- Właścicielem znaku towarowego jest brytyjskie Ministerstwo Skarbu

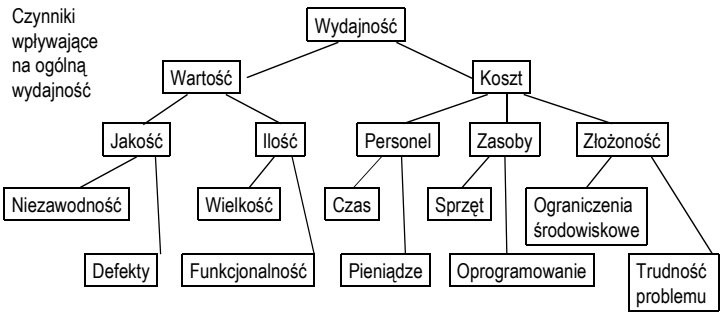
PMBoK

- Kompendium wiedzy o zarządzaniu projektami – szeroki zbiór dobrych praktyk zarządczych
  - zarządzanie integracją projektu, zakresem, czasem, kosztami, jakością, komunikacją, zasobami ludzkimi, ryzykiem, zamówieniami
  - można wykorzystywać wybiórczo
- Dostarcza szeregu technik dla Kierowników projektów **jak** zrealizować działania
- Obejmuje umiejętności interpersonalne
- Grupy procesów (5 faz realizacji):
  - Inicjowanie projektu
  - Planowanie projektu
  - Wykonywanie projektu
  - Monitorowanie i kontroling
  - Zamykanie projektu
- Łącznie 42 procesy

PCM

- PCM jest zestawem (koncepcji, zadań lub technik):
  - koncepcji cyklu życia projektu,
  - analizy zainteresowanych,
  - narzędzi planowania – matrycy logicznej (identyfikowanie: celów, rezultatów, wskaźników osiągnięcia celów, źródeł weryfikacji i czynników ryzyka),
  - kluczowych czynników jakościowych,
  - harmonogramów czynności i zasobów,
  - zestandaryzowanych, spójnych struktur kluczowych dokumentów projektowych
- Zaadaptowany w 1992 przez Komisję Europejską
- Cykl projektu:
  - faza programowania
  - faza identyfikacji
  - faza oceny
  - faza finansowania
  - faza wdrożenia
  - faza ewaluacji

Modele i miary wydajności



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

Techniki szacowania nakładów pracy

- Modele algorytmiczne - opis przedsięwzięcia za pomocą charakteryzujących go atrybutów (liczbowych) i na tej bazie wyliczenie (np. prosta formuła matematyczna) nakładów
- Ocena przez ekspertów - bazują na własnym doświadczeniu zdobytym przy realizacji innych projektów
- Ocena przez analogię - przy założeniu gromadzenia informacji o uprzednio wykonanych projektach; wyszukuje się podobne systemy poprzednio zrealizowane i wykorzystuje nakłady poniesione na ich stworzenie
- Wycena dla wygranej (ang. *pricing to win*) - na podstawie oceny możliwości klienta oraz przewidywanych działań konkurencji; opiera się na prawie Parkinsona - przedsięwzięcia są wykonywane przy założonych kosztach (jakie by one nie były)
- Szacowanie wstępujące - dzieli się przedsięwzięcie na mniejsze zadania, które łatwiej oszacować; koszt całości jest ich sumą (ew. z narzutem na integrację)

Modele algorytmiczne szacowania nakładów

- Większość modeli przyjmuje, że jednym z istotnych atrybutów jest rozmiar systemu, mierzony np. liczbą instrukcji (linii) kodu (ang. *Lines Of Code* - LOC), KLOC (ang. *Kilo*-LOC), KDSI (ang. *Kilo (thousand) of Delivered Source code Instructions*)
- Specjalizacja metryk w kierunku konkretnej klasy oprogramowania powinna dawać lepsze i bardziej adekwatne oceny niż metryki uniwersalne
- Najlepiej jest stosować zestawy metryk, co pozwala zmniejszyć błędy pomiaru
- Metryki powinny być wykorzystywane jako metody wspomagania ekspertów (stosowane formalistycznie mogą być groźne)
- Pomimo pochodzenia empirycznego, metryki skutecznie pomagają w szybkiej i mniej subiektywnej ocenie oprogramowania
- Żadna metoda przewidywania kosztów nie jest doskonała i jest oparta na szeregu arbitralnych założeń; niemniej dla celów planowania tego rodzaju metody stają się koniecznością

Metoda COnstructive COst MOdel (COCOMO 81)

Powstała w oparciu o dane z rzeczywistych projektów, realizowanych tradycyjnie (np. bez narzędzi CASE) w oparciu o model kaskadowy, autor Boehm, 1981 r.

Wymaga oszacowania liczby instrukcji, z których będzie składał się system (co może być tak samo trudne jak oszacowanie nakładów)

Wyróżnia się trzy klasy przedsięwzięć:

- **Łatwe** (ang. *organic*) - wykonywane przez stosunkowo małe zespoły, złożone z osób o podobnych wysokich kwalifikacjach; dziedzina oraz wykorzystywane metody i narzędzia są dobrze znane
- **Pośrednie** (ang. *semi-detached*) - członkowie zespołu różnią się stopniem zaawansowania; pewne aspekty dziedziny problemu lub wykorzystywane narzędzia nie są dobrze znane
- **Trudne** (osadzone, ang. *embedded)* - przedsięwzięcia realizujące systemy o bardzo złożonych wymaganiach; dziedzina problemu, stosowane narzędzia i metody mogą być w dużej mierze nieznane; członkowie zespołu nie mają doświadczenia w realizacji podobnych zadań

Podstawowy model COCOMO

- Podstawowy wzór dla oszacowania nakładów w osobomiesiącach (zależność wykładnicza):

$$\text{Nakład} = A * K^{b}$$

- K oznacza rozmiar kodu źródłowego mierzony w tysiącach linii (KDSI); nie obejmuje kodu, który nie został wykorzystany w systemie
- Wartości stałych A i b zależą od klasy, do której zaliczono przedsięwzięcie
- Dla niewielkich przedsięwzięć są to zależności bliskie liniowym; wzrost jest szczególnie szybki dla przedsięwzięć trudnych (duży rozmiar kodu)

Przedsięwzięcie łatwe:

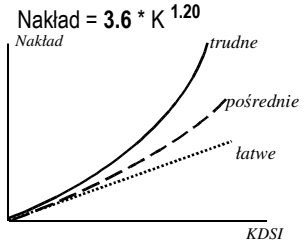
$$\text{Nakład} = \mathbf{2.4} * K^{\mathbf{1.05}}$$

Przedsięwzięcie pośrednie:

$$\text{Nakład} = \mathbf{3.0} * K^{\mathbf{1.12}}$$

Przedsięwzięcie trudne:

$$\text{Nakład} = \mathbf{3.6} * K^{\mathbf{1.20}}$$



Model COCOMO (2)

- Znając nakład można oszacować czas (w miesiącach) realizacji, z czego wynika przybliżona wielkość zespołu
- Z obserwacji wiadomo, że dla każdego przedsięwzięcia istnieje optymalna liczba członków zespołu wykonawców
- Zwiększenie tej liczby może nawet wydłużyć czas realizacji

Przedsięwzięcie łatwe:

$$\text{Czas} = 2.5 * \text{Nakład}^{0.32}$$

Przedsięwzięcie pośrednie:

$$\text{Czas} = 2.5 * \text{Nakład}^{0.35}$$

Przedsięwzięcie trudne:

$$\text{Czas} = 2.5 * \text{Nakład}^{0.38}$$

Otrzymane w ten sposób oszacowania powinny być skorygowane przy pomocy tzw. czynników modyfikujących

Atrybuty przedsięwzięcia brane pod uwagę:

- wymagania wobec niezawodności systemu
- rozmiar bazy danych w stosunku do rozmiaru kodu
- złożoność systemu (złożoność struktur danych i algorytmów, komunikacja z innymi systemami, stosowanie obl. równoległych)
- wymagania co do wydajności systemu
- ograniczenia pamięci
- zmienność sprzętu i oprogramowania systemowego tworzącego środowisko pracy systemu

Wady metody COCOMO

- Liczba linii kodu jest znana dokładnie dopiero wtedy, gdy system jest napisany a szacunki mogą być (i zwykle są) obarczone bardzo poważnym błędem (niekiedy ponad 100%)
- Określenie "linii kodu źródłowego" inaczej wygląda dla każdego języka programowania (np. 1 linia w Smalltalk'u jest równoważna 10-ciu linii w C; dla języków 4GL ten stosunek może być nawet 1000:1)
- Koncepcja oparta na liniach kodu źródłowego natomiast całkowicie nie przystaje do współczesnych narzędzi programistycznych, np. opartych o programowanie wizyjne
- Opiera się tylko na długości kodu i nie bierze w ogóle pod uwagę funkcjonalności ani złożoności produktu
- Kwalifikacja przedsięwzięcia do predefiniowanych klas oraz dobór czynników modyfikujących jest trudny, a ewentualne błędy mogą prowadzić do znacznych rozbieżności pomiędzy oczekiwanym i rzeczywistym kosztem

COCOMO II

- Uwspółcześniona wersja COCOMO opublikowana w 2000 r.
- Zakłada, że pracochłonność przedsięwzięcia zależy od wielu czynników, które ujawniają się w czasie realizacji
  - opracowanie architektury jest kluczowym momentem
- Dwa różne schematy obliczeń:
  - uproszczony model (wczesnego projektu) (ang. *early design model*)
  - dokładny model (gotowej architektury) (ang. *post-architecture model*)
- Postać formuły wyliczania pracochłonności w funkcji rozmiaru jest analogiczna, różne są tylko uwzględniane czynniki

$$\textit{Pracochłonność} = A * (\text{rozmiar})^{B} * \textit{Korekta}$$

- współczynnik A jest stały (A=2,94)
- parametr B decyduje o skalowalności $B = 0{,}91 + 0{,}01 * \sum Si$ (i=1..5)
- *Korekta* = iloczyn czynników: 7 w uproszczonym, 17 w pełnym modelu

COCOMO II (2)

- Współczynniki *Si* są ocenami następujących cech projektu :
  - innowacyjność projektu
  - elastyczność wymagań
  - jakość planowania i zarządzania ryzykiem
  - zgodność celów udziałowców projektu
  - dojrzałość procesu wytwórczego
- W efekcie *B* może zmieniać się w zakresie od 0,91 do 1,23
- Pełna wersja modelu COCOMO II jest bardzo złożona
  - zawiera ok. 160 parametrów umożliwiających obliczanie rozmiaru programu (uwzględniając np. kod generowany automatycznie, dostosowywany z innych projektów) oraz reguły wyliczania współczynników korygujących

Analiza Punktów Funkcyjnych

- Metoda analizy punktów funkcyjnych (ang. *function points* - FP), została opracowana przez Albrechta (początek lat 80-tych); łączy własności metody badającej rozmiar projektu programu z możliwościami metody badającej produkt programowy (jego funkcjonalność)
- Najpierw ocenia się zobiektywizowaną złożoność problemu, wynikającą ze złożoności danych i algorytmów przetwarzania realizowanych przez aplikację
  - koncentruje się na perspektywie użytkowników; nie bierze się pod uwagę technologii implementacji
  - liczbę pierwotnych (nieskorygowanych) punktów funkcyjnych (ang. *Unadjusted FP*) wylicza się korzystając z 5 kategorii elementów składowych modelu
  - każdy zidentyfikowany element modelu przetwarzania jest oceniany w 3 stopniowej skali opisowej
- W drugim etapie uwzględnia się dodatkowe czynniki wpływające na złożoność przedsięwzięcia

$$FP = UFP * VAF$$

Kategorie elementów modelu przetwarzania

- Zbiory wewnętrzne – modelują dane przechowywane i utrzymywane przez oceniane oprogramowanie
  - zbiór odpowiada grupie logicznie powiązanych danych opisujących jakiś istotny aspekt przetwarzania
- Zbiory zewnętrzne – modelują współpracę z innymi systemami
  - zbiór reprezentuje logicznie powiązane dane otrzymywane z innego systemu
- Wejścia użytkownika (zewnętrzne) - modelują elementarne operacje przetwarzające dane dostarczane z zewnątrz (przez użytk. lub inne systemy)
  - operacje opisują działania istotne z punktu widzenia użytkownika => efekt zmiana zawartości zbiorów wewnętrznych lub zmiana zachowania systemu
- Wyjścia użytkownika (zewnętrzne) - modelują elementarne operacje przetwarzające dane i przekazujące wyniki do otoczenia
  - działanie musi obejmować coś więcej niż tylko proste odczytanie danych
- Zapytania – modelują operacje prostego wyszukania i przedstawienia danych zgromadzonych w zbiorach aplikacji (brak obliczeń i zmian wartości w zbiorach)

Ocena złożoności elementów modelu

- Reguły oceny złożoności elementów modelu są ściśle określone; brane są pod uwagę następujące charakterystyki:
  - liczba typów rekordu zbioru (liczba różnych encji reprezentowanych przez zbiór)
  - liczba pól zbioru (liczba różnych atrybutów występujących w encjach zbioru; uwzględnia się też atrybuty łączące zbiory – klucze obce)
  - liczba wykorzystywanych zbiorów (zarówno wewnętrznych jak i zewnętrznych w operacji)

Czynnik złożoności	Prosty	Średni	Złożony
Wejścia użytkownika	3	4	6
Wyjścia użytkownika	4	5	7
Zbiory danych wewnętrzne	7	10	15
Zbiory danych zewnętrzne	5	7	10
Zapytania zewnętrzne	3	4	6

Wagi przypisywane poszczególnym elementom modelu zależą od ich złożoności

Pierwotne i skorygowane punkty funkcyjne

- Złożoność oprogramowania (pierwotna liczba punktów funkcyjnych) – ważona suma ilości elementów modelu przetwarzania
  - wagi wynikają ze złożoności poszczególnych elementów modelu
- Aby otrzymać finalną (skorygowaną) liczbę punktów funkcyjnych należy uwzględnić dodatkowe czynniki złożoności przedsięwzięcia:
  - pod uwagę branych jest 14 czynników korygujących i przypisuje się im wagi (skala 6-stopniowa: od 0 - „brak wpływu" do 5 -„bardzo duży wpływ")
  - suma ich ocen oraz odpowiednie wyskalowanie pozwalają wyliczyć współczynnik korekcyjny (ang. *Value Adjustment Factor*)
  - pozwala to skorygować liczbę punktów funkcyjnych o 35% w obie strony
- Można pokusić się o przeliczenie otrzymanej złożoności projektu na pracochłonność albo koszt wykonania
  - dostępne są uśrednione tabele i wykresy (np. pracochłonności) dla różnych języków
  - należy jednak być ostrożnym; zaleca się wykorzystanie własnych danych historycznych

Czynniki korygujące

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

Wykorzystanie punktów funkcyjnych

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

1 FP ≈ 125 instrukcji w C

10 FPs - typowy mały program tworzony samodzielnie przez klienta (1 m-c)

100 FPs - większość popularnych aplikacji; wartość typowa dla aplikacji tworzonych przez klienta samodzielnie (6 m-cy)

1,000 FPs - komercyjne aplikacje w MS Windows, małe aplikacje klient-serwer (10 osób, ponad 12 m-cy)

10,000 FPs - systemy (100 osób, ponad 18 m-cy)

100,000 FPs - MS Windows'95, MVS, systemy militarne

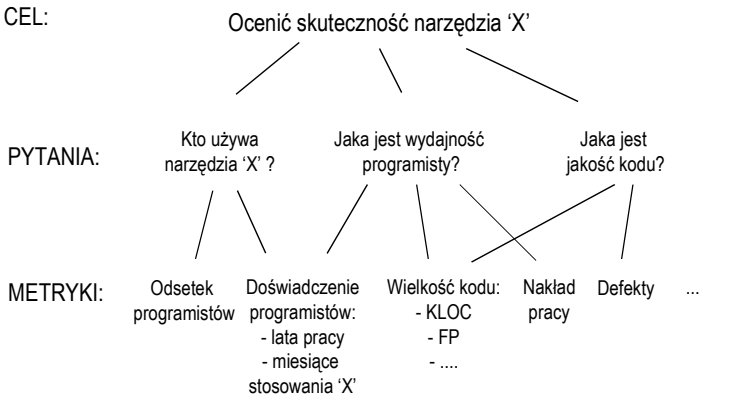
Punkty aplikacyjne (obiektowe)

- Alternatywna metoda w stosunku do punktów funkcyjnych zaproponowana na początku lat 90-tych przez Bankera i współpracowników
- Wykorzystywana do języków programowania baz danych, 4GL
  - obiektowe w nazwie nie ma tutaj związku z programowaniem obiektowym
- Liczba punktów obiektowych jest ważoną sumą:
  - liczby różnych formatek ekranowych (prosty ekran - 1 punkt, średnio złożony - 2 punkty, bardzo złożony - 3 punkty)
  - liczba generowanych raportów (prosty raport - 2 punkty, średnio skomplikowany - 5 punktów, a potencjalnie najtrudniejsze do utworzenia - 8 p.)
  - liczba modułów w językach 3GL (takich jak C++; Java) , które należy utworzyć w celu uzupełnienia kodu 4GL - każdy moduł 10 punktów
- Zaletą punktów obiektowych w porównaniu z punktami funkcyjnymi jest to, że łatwiej je obliczyć na podstawie specyfikacji oprogramowania wysokiego poziomu

Cel, Pytanie, Metryka
(Goal, Question, Metric - GQM)

- Wiele zamierzeń związanych z metrykami oprogramowania zawiodło, gdyż cele były zwykle słabo zdefiniowane; aby temu przeciwdziałać na Uniwersytecie Maryland (Vic Basili *et al.*) opracowano metodę GQM
- Trzy podstawowe kroki:
  - ustalenie celów w kategoriach *zamiaru*, *perspektywy* i *środowiska*
  - ponowne wypowiedzenie celów w terminach odpowiedzi na łatwe do "skwantyfikowania" pytania
  - określenie metryk i danych, które należy zgromadzić, aby udzielić odpowiedzi na postawione pytania
- Dzięki takiemu ułożeniu koncentrujemy się tylko na metrykach, które odnoszą się do interesującego nas celu
- Kilka pomiarów, aby odpowiedzieć na jedno pytanie; pojedynczy pomiar może mieć zastosowanie do kilku pytań

Przykład wykorzystania GQM



Dojrzałość procesów wytwórczych

Niedojrzałość

- ☹ Improwizacja podczas procesu wytwórczego
- ☹ Proces jest wyspecyfikowany, ale specyfikacja nie jest stosowana
- ☹ Doraźne reagowanie w sytuacji kryzysów
- ☹ Harmonogram i budżet są przekraczane
- ☹ Funkcjonalność jest stopniowo okrajana
- ☹ Jakość produktu jest niska
- ☹ Brak obiektywnych kryteriów oceny

Dojrzałość

- ☺ Zdolność do budowy oprogramowania jest cechą organizacji a nie personelu
- ☺ Proces jest zdefiniowany, znany i wykorzystywany
- ☺ Proces jest obserwowany i ulepszany
- ☺ Prace są planowane i monitorowane
- ☺ Role i odpowiedzialności są zdefiniowane
- ☺ Obiektywna, ilościowa ocena

Model dojrzałości procesu wytwórczego

- Model dojrzałości procesu wytwórczego (ang. *Capability Maturity Model* - CMM) został opracowany w latach 80-tych przez amerykański Instytut Inżynierii Oprogramowania na zlecenie rządu USA
- Wykorzystywany był w procedurach oceny (klasyfikacji) potencjalnych wykonawców oprogramowania dla Departamentu Obrony
- Opiera się w znacznym stopniu na koncepcjach TQM (ang. *Total Quality Management*)
- Ocenia wiele różnych atrybutów wytwarzania oprogramowania, obejmujących użycie narzędzi i standardowych praktyk
- Model ten szybko zyskał szeroką akceptację jako wzorzec do usprawniania procesu programowania
- Wywarł znaczny wpływ na uświadomienie znaczenia miar i ich stosowanie, gdyż w CMM miary są uważane za ważne dla osiągnięcia wyższych poziomów usprawnienia procesu

Poziomy dojrzałości wytwórców

Wyróżniono 5 poziomów dojrzałości wytwórców (poczynając od p. najniższego):

- początkowy (1) - proces chaotyczny, nie istnieją żadne standardy, decyzje podejmowane *ad hoc*; może dotyczyć nawet firm o dobrym zaawansowaniu technicznym
- powtarzalny (2) - proces zindywidualizowany;przedsięwzięcia wykonywane w podobny sposób (standardy *de facto*); standardy nie są udokumentowane i nie istnieją ścisłe procedury kontroli
- zdefiniowany (3) - proces zinstytucjonalizowany; standardy postępowania są zdefiniowane, sformalizowane i ich stosowanie jest kontrolowane
- zarządzany (4) - proces nie tylko podlega kontroli ale jest również mierzony w sposób ilościowy; informacje zwrotne wykorzystywane są do sterowania procesem
- optymalizujący (5) - standardy są ciągle uaktualniane; informacje zwrotne wpływają na ulepszenie procesu; standardy zawierają elementy pozwalające na dostrojenie procesu do aktualnych potrzeb
- Początkowo niewiele firm uzyskiwało poziom 3-ci, umożliwiający dostarczanie oprogr. dla Dep. Obrony; tylko IBM w zakresie oprogr. promu kosmicznego dla NASA uzyskał poziom 5-ty

Przygotowano na podstawie:

- *Wprowadzenie do inżynierii oprogramowania*, K. Subieta, Wydawnictwo PJWSTK, 2002.
- *Inżynieria oprogramowania w projekcie informatycznym*, red. J. Górski, Mikom 2000.
- *Inżynieria oprogramowania*, K. Sacha, PWN, 2010.