Wydział Informatyki PB

Inżynieria oprogramowania
i analiza biznesowa

Wykład 8:

Zarządzanie przedsięwzięciem informatycznym

Marek Krętowski
pokój 13a
e-mail: m.kretowski@pb.edu.pl
http://aragorn.pb.bialystok.pl/~mkret

Wersja 1.2

Zarządzanie przedsięwzięciem

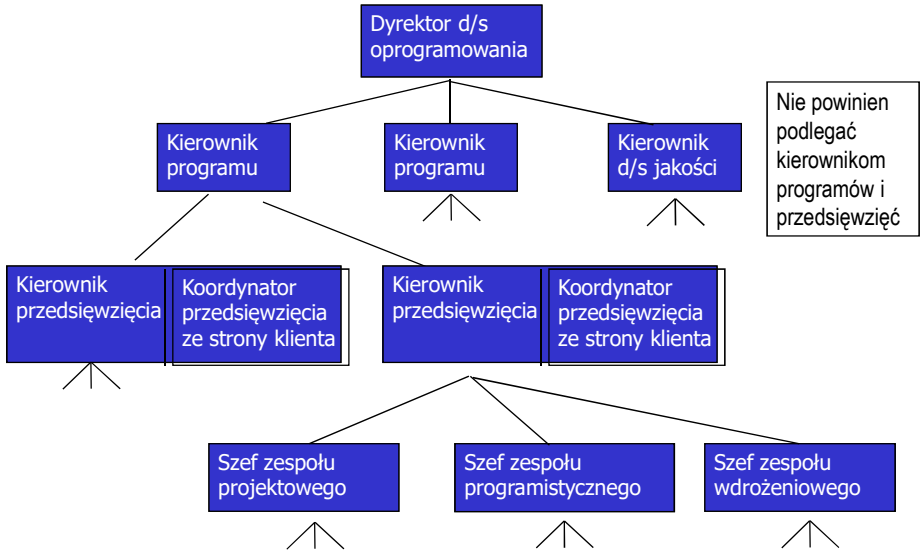
Zatrudnienie wysokiej klasy specjalistów oraz stosowanie zaawansowanych narzędzi wspomagających nie gwarantuje jeszcze sukcesu projektu; niezbędne jest właściwe zarządzanie przedsięwzięciem

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



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

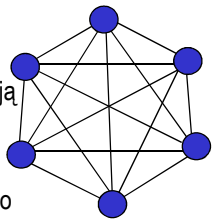
•**Rozpoznawanie i rozwijanie potencjału swoich współpracowników** - naturalnym zjawiskiem jest wymiana personelu (np. dobrze wyszkoleni i kompetentni pracownicy awansują lub zmieniają pracę); powstaje potrzeba wprowadzania do zespołu nowych ludzi, właściwego ich szkolenia, pomocy w rozwijaniu ich potencjału

•**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:

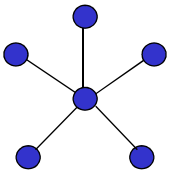


•dzięki ścisłej współpracy członkowie zespołu wzajemnie kontrolują swoją pracę; szybko osiągane są standardy jakości

•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)

Projekt / Zespół kompetencyjny	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

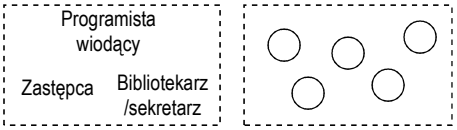
–przydziela zadania puli specjalistów

•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



Zespół programisty wiodącego

Pula specjalistów

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

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

SCRUM – zwinna metoda tworzenia oprogramowania

•Metodyka adaptacyjna stosowana w procesie wytwarzania złożonych produktów

–Daje konkretny i spójny przepis na prowadzenie projektów inf.

–Pozostawia dużą swobodę dotyczącą sposobu realizacji w konkretnej organizacji

•Wg twórców opisuje ramy metodyczne, w ramach których możliwe jest stosowanie różnych procesów i technik

–Rola Scrum-a jest wykazywanie nieefektywności stosowanych praktyk tak, aby możliwe było ich usprawnienie

•Korzyści z dobrej implementacji Scrum-a:

–zwiększenie wydajności zespołów i duża elastyczność przy realizacji

•Dotyczy **wyłącznie fazy wytwarzania**

•Nie dotyczy przygotowania wizji projektu, podejmowania decyzji strategicznych czy zarządzania kontraktem

Trzy role: Właściciel produktu, Mistrz i Zespół

•**Mistrz** (Scrum Master) – zadanie: dopilnowanie przestrzegania przez zespół wartości praktyk i reguł Scrum-a

–uczy Zespół, trenując i prowadząc członków do osiągnięcia wyższej wydajności i tworzenia produktów o wyższej jakości

–usuwanie przeszkód w środowisku, które nie jest przystosowane do wytwarzania złożonych produktów; przywództwo służebne

–współpracując z klientami i kierownictwem identyfikuje osobę, która będzie Właścicielem Produktu; następnie uczy ją wykonywania pracy w ramach Scrum (odpowiada za nauczenie)

•Samoorganizujący się i interdyscyplinarny (międzyfunkcjonalny) **Zespół**

Właściciel produktu

•**Właściciel produktu** (Product Owner) – zarządza rejestrem produktowym (ma być widoczny i zrozumiały) i czuwa na wartością pracy wykonywanej przez zespół

–jest odpowiedzialny za reprezentowanie interesów wszystkich zainteresowanych projektem i tworzonym systemem

–jedna osoba, nie komitet; mogą istnieć osoby doradzające, ale aby zmienić priorytet pozycji rejestru decyzję musi podjąć Właściciel

–aby odniósł sukces jego decyzję muszą być respektowane; nie może być innych decydentów, którzy ustawiają pracę zespołu

Członkowie zespołu

•Zespół programistów przetwarza podczas Sprintów rejestr produktowy w kolejne przyrosty funkcjonalności

–członkowie muszą posiadać wszystkie umiejętności do wytworzenia kolejnego przyrostu => mogą mieć wyspecjalizowane umiejętności,

•Ważne umiejętności wspólne dla wszystkich

–Zespołowe podjęcie zobowiązania i przekształcenie wymagań w używalny produkt

–osoby, które nie chcą kodować, a tylko np. projektować nie pasują do zespołu;

–każdy musi dołożyć cząstkę pracy, nawet kosztem nauki lub przypomnienia

–Nie istnieje hierarchia służbowa, nie ma stanowisk, nie ma podziału funkcjonalnego

–Zespół samoorganizujący się – każdy członek wykorzystuje swoje kompetencje do rozwiązania problemu; powstająca synergia podnosi wydajność

•Optymalna liczebność zespołu: 7+/-2 (bez Mistrza i Właściciela):

–Mniej niż 5: mniej interakcji => mniejsza produktywność; możliwe niedobory umiejętności

–Więcej niż 9: wymagana intensywniejsza koordynacja

Bajka o kurczaku i świnkach

[A. Koszlajda „Zarządzanie projektami IT”]

•Należy jasno rozgraniczyć osoby, które rzeczywiście pracują (poświęcają się) przy realizacji projektu (członkowie zespołu - „świnki”) od osób, które są zaangażowane, ale nie pracują bezpośrednio nad projektem („kurczaki”)

•Interesanci – odpowiedzialni za podejmowanie decyzji strategicznych

–za ich taktyczną implementację odpowiedzialny jest właściciel produktu

–mogą biernie obserwować codzienne spotkania

Podstawowe narzędzia

•**Rejestr produktowy**, dziennik zadań produktu, zaległości produktowe (Product Backlog) – zhierarchizowana lista wszelkich elementów składających się na ostateczną postać produktu

–definiuje wysokopoziomowe, biznesowe funkcje aplikacji, które mają zostać wykonane w trakcie całego projektu

–podstawowa forma dyskusyjna pomiędzy zespołem a właścicielem produktu

–żywy dokument – zmiany w wymaganiach

•**Rejestr zadaniowy**, dziennik zadań Sprinta (Sprint Backlog) – lista zadań (technicznych), których wykonanie doprowadzi do przetworzenia wybranych (w danym Sprincie) pozycji rejestru produktowego w gotowy fragment potencjalnie zbywalnego produktu

–optymalna długość trwania pojedynczego zadania od 4 do 12 godzin przy 14 dniowym Sprincie

–przy każdym zadaniu liczba godzin pozostałych do zakończenia

Sprint i rejestr zadaniowy

				Hours of work remaining upd										
Task Description	Originator	Responsible	Status (Not Started/In Progress/ Completed)	1	2	3	4	5	6	7	8	9	10	11	12
Meet to discuss the goals and features for Sprint 3-6	Danielle	Danielle/ Sue	Completed	20	0	0	0	0	0	0	0	0	0	0	0
Move Calculations out of Crystal Reports	Jim	Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Get KEG Data		Tom	Completed	12	0	0	0	0	0	0	0	0	0	0	0
Analyse KEG Data - Title		George	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Analyse KEG Data - Parcel		Tim	Completed	12	12	12	12	12	4	4	4	0	0	0	0
Analyse KEG Data - Encumbrance		Josh	In Progress							12	10	10	10	10	10
Analyse KEG Data - Contact		Danielle	In Progress	24	24	24	24	12	10	8	6	6	6	6	6
Analyse KEG Data - Facilities		Allen	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Define & build Database		Barry/ Dave	In Progress	80	80	80	80	80	80	60	60	60	60	60	60
Validate the size of the KEG database		Tim	Not Started												
Look at KEG Data on the G:\		Dave	In Progress	3	3	3	3	3	3	3	3	3	3	3	3
Confirm agreement with KEG		Sue	Not Started												
Confirm KEG Staff Availability		Tom	Not Started	1	1	1	1	1	1	1	1	1	1	1	1
Switch JDK to 1.3.1. Run all tests.		Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Store PDF files in a structure		Jacquie	Completed	8	0	0	0	0	0	0	0	0	0	0	0
TopLink: Cannot get rid of netscape parser		Richard	Completed	4	0	0	0	0	0	0	0	0	0	0	0
Build test data repository		Barry	In Progress	10	10	10	10	10	10	10	10	8	8	8	8
Move application and database to Qual (incl Crystal)		Richard	Completed	4	4	4	4	4	4	4	0	0	0	0	0
Set up Crystal environment		Josh	Completed	2	2	2	2	1	1	1	0	0	0	0	0
Test App in Qual		Sue	In Progress												20
Defining sprint goal required for solstice in 2002		Lynne	In Progress	40	40	40	40	40	40	40	38	38	38	38	38
Reference tables for import process		Josh	In Progress												
Build standard import exception process		Josh	In Progress									12	12	12	10
Handle multiple file imports on same page		Jacquie	Disregarded												
Migrate CruiseControl Servlet to iWS 6.0 (landcc_7101) server		Allen	Not Started	4	4	4	4	4	4	4	4	4	4	4	4
Create web server for Qual on PF1D8		Allen	Completed	1	0	0	0	0	0	0	0	0	0	0	0
LTCS Disk		Danielle/ George	In Progress	12	12	12	12	8	8	8	8	8	8	8	8
Follow this with questions about KEG data to Sue/Tom, re: Keg, LTO	Jacquie	Danielle	Completed	10	10	10	10	10	8	8	0	0	0	0	0
Map KEG data to Active Tables - see also #14	Jacquie	Jacquie/Allen	In Progress	50	50	50	50	50	50	50	50	50	50	50	50
Prepare SQL to import from KEG tables to Active Tables	Jacquie	George	In Progress	25	25	25	25	25	25	25	25	25	25	25	25

Rejestr zadaniowy
[K. Schwaber „Sprawne zarządzanie ...

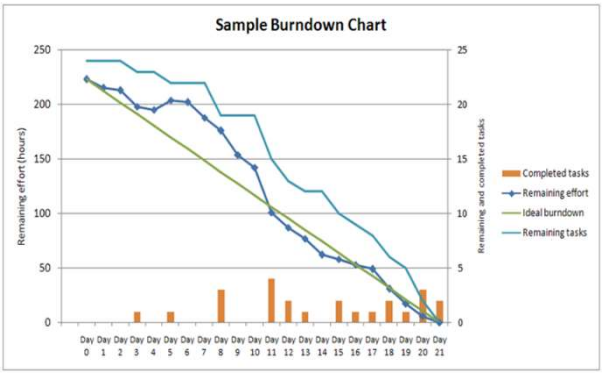
•**Sprint** – ograniczony do (zwykle) 30 kolejnych dni kalendarzowych odcinek czasu, podczas którego zespół pracuje nad zmianą wybranych zaległości produktowych (elementów rejestru produktowego) w przyrost funkcjonalności produktu możliwy do wydania

Podstawowe narzędzia (2)

•**Wykres wypalania**, diagram spalania (Burndown Chart) – ocena postępu prac w projekcie

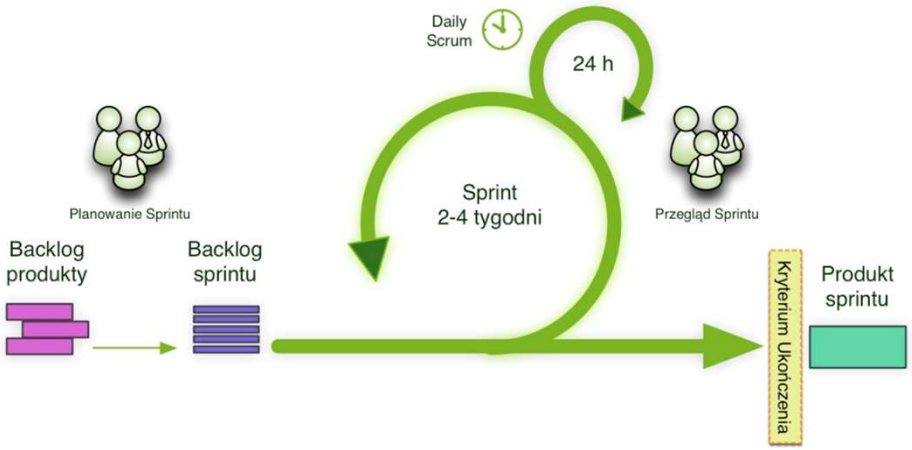
–Pomiar wypalania dla projektu (Release Burndown) – pokazuje elementy pozostające w rejestrze produktowym w stosunku do czasu zakończenia projektu

–Pomiar wypalania dla Sprintu (Sprint Burndown) – pokazuje zadania pozostające w rejestrze zadaniowym w stosunku do końca Sprintu



[Wikipedia „Burn down chart"]

Mechanika Sprintu



kryterium ukończenia (ang. definition of done) [www.scrum.org.pl]

Codzienny Scrum

•Krótkie (15 minut) spotkanie (inspekcyjno-adaptacyjne)

–zwykle o stałej porze, na stojąco

–tylko członkowie zespołu

[www.aktually.com/tag/scrum]

•Każdy z członków odpowiada na **3** pytania:

–Co zrobił od ostatniego spotkania?

–Co będzie robił do następnego spotkania?

–Jakie są przeszkody, zagrożenia w realizacji planów?

•Okazja do kontroli postępu prac ku przyjętemu celowi Sprintu:

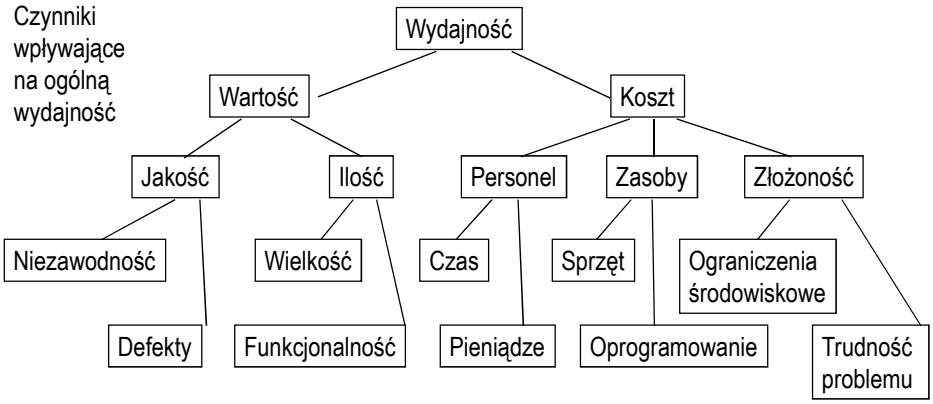
–poprawia komunikację, eliminuje potrzebę innych spotkań, identyfikuje i usuwa przeszkody, wspiera szybkie decyzje, podnosi poziom znajomości stanu prac projektowych

•Mistrz odpowiada za doprowadzenie do spotkania i je nadzoruje organizacyjnie

–zwięzłość wypowiedzi, egzekwowanie reguł

Modele i miary wydajności

Czynniki wpływające na ogólną wydajność



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) [często rozmiar kodu w KLOC] 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)

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

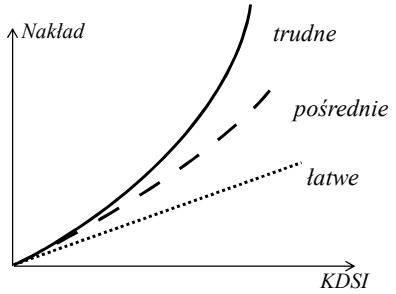
•Dla niewielkich przedsięwzięć są to zależności bliskie liniowym; wzrost jest szczególnie szybki dla przedsięwzięć trudnych (duży rozmiar kodu)

•Korekta przy wykorzystaniu czynników modyfikujących

Przedsięwzięcie łatwe:
Nakład = **2.4** * K **1.05**

Przedsięwzięcie pośrednie:
Nakład = **3.0** * K **1.12**

Przedsięwzięcie trudne:
Nakład = **3.6** * K **1.20**



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

Pracochłonność = A * (rozmiar)^{B} * 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

Wykorzystywanie PF do:

•Ocena złożoności realizacji (lub re-inżynierii) systemów

•Szacowanie liczby testów, kosztów pielęgnacji

•Porównanie i ocena różnych ofert dostawców oprogramowania

•…

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

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.
- *Zarządzanie projektami IT. Przewodnik po metodykach*, A. Koszlajda, Helion, 2010.
- *SCRUM guide – Przewodnik po metodyce SCRUM*, K. Schwaber, J. Sutherland; tłum. A. Gajewska, T. Włodarek, 2010.