

Budowa i integracja systemów informatycznych

dr hab. inż. Marta Łabuda
labudama@pjwstk.edu.pl



Materiały i zaliczenie

- **Udostępnienie materiałów**

<http://gakko.pjwstk.edu.pl>

- **Zaliczenie: teoria (trzy sprawdziany) + zadania projekt**
- **Ocena końcowa przedmiotu (wykładu)**
 - egzamin pisemny (lub zaliczenie pozytywnie wejściówek)
 - **warunek dopuszczenia do egzaminu**
: pozytywna ocena z projektu
 - **warunek zdania**
: pozytywna ocena z egzaminu pisemnego
: ocena \Leftarrow $\frac{1}{2}$ sumy ocen za egzamin i za projekt, 40/60

Osoby, które nie zaliczyły ITN z BYT powinny powtarzać zajęcia projektowe

Treść przedmiotu BYT

Wykład stanowi omówienie podstawowych zagadnień inżynierii oprogramowania, w tym faz rozwoju oprogramowania oraz metod podwyższenia jego jakości; stanowi rozwinięcie wykładu „Projektowanie systemów informacyjnych” (PRI)

❏ Wykład

- Motywacje i pojęcia IO; treść przedmiotu; Faza *Planowania projektu*
- Cykle życia oprogramowania - podstawowe modele; dobór strategii
- Procesy analizy i projektowania; wzorce, szablony i komponenty; integracja
- Zapewnianie jakości oprogramowania; bezpieczeństwo systemów
- Testowanie i walidacja oprogramowania; dokumentowanie testów.
- Wdrożenie i pielęgnacja oprogramowania

❏ Projekt/laboratorium

Konstrukcja aplikacji w oparciu o przyjętą metodykę, wzorce (analityczne, projektowe) frameworki i komponenty + dokumentacja projektu

Literatura

Literatura podstawowa:

- Bruegge B., Dutoit A.H.: Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java. Wyd. 3, Helion, 2011.
- Gamma E., Helm R., Johnson R., Vlissides J.: Wzorce projektowe, WNT, 2008
- **Górski J. (red.): Inżynieria oprogramowania w projekcie informatycznym, Wyd. II, MIKOM, 2000.**
- Szejko S. (red): Metody wytwarzania oprogramowania, MIKOM 2002

Literatura uzupełniająca:

- Dąbrowski W., Subieta K.: Podstawy inżynierii oprogramowania. Wyd. PJWSTK, 2005.
- Fowler M.: Analysis patterns. Addison-Wesley, 1996
- Shalloway A., Trott J.: Projektowanie zorientowane obiektowo. Wzorce projektowe, Helion 2002.
- Sommerville I.: Inżynieria oprogramowania, WNT 2003.
- Trzaska M.: Modelowanie i implementacja systemów informatycznych. Wyd. PJWSTK, 2008

Treść zajęć PRO

Praktyczne zagadnienia planowania i prowadzenia projektów informatycznych

- Pojęcia wstępne
- Ustanowienie i planowanie projektu
- Zagadnienia ryzyka i dobór strategii prowadzenia projektu
- Praktyka analizy i projektowania; dokumentowanie
- Narzędzia wspierające zarządzanie projektem; MS Project
- Wykorzystanie wzorców projektowych
- Implementacja i integracja oprogramowania
- Testowanie i walidacja
- Dobre praktyki oraz retrospekcja projektu i dokumentacji
- Instalacja, wdrożenie i pielęgnacja oprogramowania
- *Praktyka inżynierii oprogramowania*
- Raport końcowy projektu

• *Projekt/laboratorium*

- ćwiczenia wprowadzające do zagadnień analizy projektów, doboru strategii, harmonogramowania (MS Project) oraz integracji oprogramowania
- realizacja i udokumentowanie projektu wytwarzanego według przyjętego planu i metodyki;

• *Zintegrowanie z BYT*

Motywacja

Odpowiedzieć na podstawowe pytania / problemy inżynierii oprogramowania, m.in.:

- Jak zbudować system / aplikację, jak dopasować do potrzeb, jaką wybrać drogę? Zespołowo?
- Aby było efektywnie, relatywnie tanio, a przede wszystkim – by system był dobrej jakości? I co to znaczy *dobrej jakości*?
- Jak zaplanować realizację systemu, co taki plan powinien zawierać?
- Jakże winny być kolejne kroki procesu analizy wymagań? Projektowania? Testowania? Integracji komponentów?
- Co obejmuje *wdrożenie systemu*?
- Co z dokumentacją?

.....



Sytuacja

Oprogramowanie nie jest takie, jakiego oczekuje klient !

Główne „osiągi” krytyczne:

- niepełna funkcjonalność, kiepska użyteczność, słabe dostosowanie do potrzeb
- za długo trwa przygotowanie oprogramowania
- za dużo kosztuje



© Scott Adams, Inc./Dist. by UFS, Inc.

Potrzeba skutecznych, efektywnych metod konstrukcji (...) oprogramowania wysokiej jakości

Remedium:

profesjonalizm – umiejętności,
kompetencja, etyka – w działaniach
inżynierii oprogramowania



Od problemu do realizacji

○ IO podsuwa wzorce metodyk budowy oprogramowania

Tradycyjne...

Kaskadowy (klasyczny)

Model V

Prototypowanie

Model przyrostowy

Model spiralny

Ich hybrydy
i adaptacje

Podejście (paradygmat) – przyjęty sposób widzenia rzeczywistości w danej dziedzinie



Od problemu do realizacji

... i nowsze - **bardziej lub mniej udane** -

Wielokrotne użycie oprogramowania (*ang. software reuse*)

Model komponentowy

RUP – Rational Unified Process

Adaptacyjne (lekkie, *agile*) – SCRUM,
extreme programming, *test-driven programming*,...

Przekształcenia modeli – MDD, MDA
i przekształcenia formalne

Ponowna inżynieria (*ang. re-engineering*)

Dlaczego? Skąd taka potrzeba?



Od problemu do realizacji

- ❑ IO oferuje gotowe
- ✓ Komponenty
- ✓ wzorce
- konstrukcji
- oprogramowania
- ✓ szablony
- (*frameworki*)

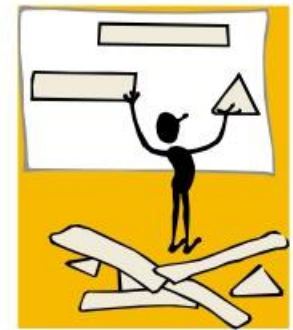


Jak poprowadzić przedsiębiorstwo, jak dobrać strategię postępowania?

Nie ma „jednej prawdy”, są tylko lepsze lub gorsze rozwiązania konkretnych sytuacji

Czynniki wpływające na dobór strategii

- sytuacja problemowa
- modele, ich przydatność w tej sytuacji
- minimalizacja ryzyka
- sytuacja firm
- technologia
-



W praktyce modele (wzorce) rzadko występują 'w stanie czystym' – są dostosowywane do potrzeb / sytuacji projektów lub firm je realizujących

Procesy Klienta i Dostawcy

Faza	Projekt A (Kupującego)	Projekt B (Dostawcy)
Planowanie	Identyfikacja potrzeb i wymagań biznesowych Analiza wykonalności Podejmowanie decyzji Wstępne planowanie projektu Przydział zasobów	Lobbing Ewentualna analiza wykonalności - na zamówienie Kupującego
Negocjowanie	Organizacja procesu selekcji Kontraktacja	Ocena zasobów Ocena korzyści Podejmowanie decyzji Planowanie Udział w przetargu i negocjacje kontraktu
Wytworzenie	Uszczegółowienie wymagań Nadzór Walidacja i testowanie akceptacyjne	Analiza wymagań Projektowanie Kodowanie i testowanie Weryfikacja, walidacja i testowanie
Wdrożenie (Rozpowszechnienie)	Absorpcja systemu Eksploatacja	Procesy utrzymaniowe i ewolucji oprogramowania

Główne obszary prowadzenia projektu informatycznego (zarządzania projektem)

- ➡ koordynowanie prac, reagowanie na problemy i zmiany
- ➡ zarządzanie czasem i zakresem projektu
- ➡ zarządzanie kosztami
- ➡ zarządzanie infrastrukturą
środowiskiem, ludźmi, komunikacją,
wersjami, dokumentowaniem,
- ➡ zapewnienie jakości
- ➡ identyfikacja i przeciwdziałanie
zagrożeniom (ryzyku)
- ➡ zarządzanie konfiguracją

*Przewodnik Komitetu
Standaryzacyjnego PMI
(Project Management
Institute)*

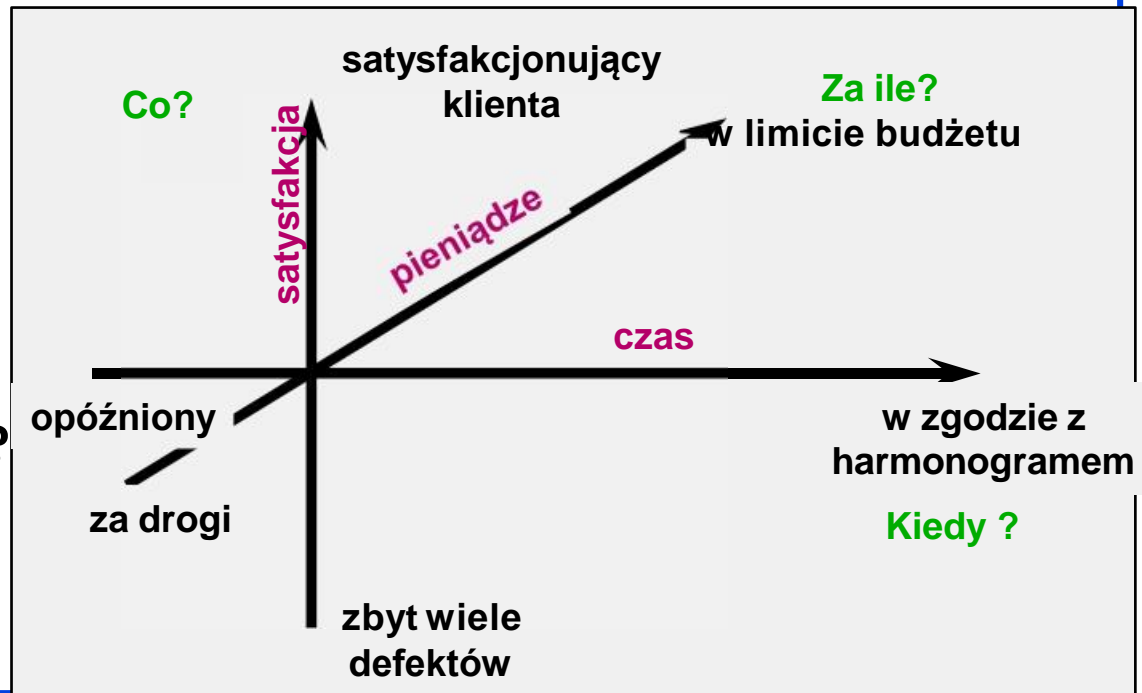


IEEE Std. 1490-98

Miary sukcesu

Projekt osiąga swoje cele w wyznaczonym czasie i w ramach przewidzianego budżetu

- cele (co chcemy osiągnąć?)
 - satysfakcja klienta
- harmonogram (w jakim czasie?)
- budżet (ile jesteśmy skłonni zapłacić?)
- wpływ na udziałowców? (skutki społeczne)



Czynniki sukcesu i niepowodzeń projektów informatycznych

za *Standish Group Reports*

Czynnik sukcesu	Punktacja
Wsparcie zarządzających, sponsorskie	20
Zaangażowanie użytkowników	15
Dostosowanie strategii realizacji	15
Kwalifikacje realizatorów	13
Umiejętne zarządzanie projektem	12
Lekka (<i>agile</i>), dopasowana strategia realizacji	10
Jasno określone cele	6
Dojrzałość emocjonalna zespołu i środowiska projektu	5
Wykonanie, planowość realizacji	3
Narzędzia i infrastruktura	1

Czynniki niepowodzeń	% głosów
Brak współpracy z użytkownikami	12,8
Błędy, braki specyfikacji wymagań	12,3
Zmiany wymagań, ich specyfikacji	11,8
Brak wsparcia od zarządzających	7,5
Niewiedza, braki kompetencji technicznych	7,0
Niedostatek zasobów	6,4
Nierealistyczne oczekiwania	5,9
Niejasno wyznaczone cele	5,3
Nierealistyczne ramy czasowe	4,3
Nowa technologia	3,7
Inne	23

Zagrożenie obniżenia poziomu sukcesu przedsięwzięcia = *ryzyko*



Co to jest „ryzyko”?

- możliwość zaistnienia niechcianego zdarzenia
- niepożądane konsekwencje



Zagrożenia w projekcie informatycznym - przykłady [J. Górski]

Dla klienta i wykonawcy

- przekroczenie budżetu
- przekroczenie terminu realizacji

Dla użytkownika końcowego

- niewłaściwa funkcjonalność
- “trudny” interfejs użytkowy
- niska wydajność systemu
- wysoka zawodność systemu

Dla wdrożeniowca

- niska jakość oprogramowania
- trudność w dopasowaniu systemu do środowiska docelowego

Co zrobić z ryzykiem?

pasywna akceptacja ⇔ plan aktywnego przeciwdziałania



□ Wizja systemu

cele, wymagania, rozwiązania projektowe

✓ **Cele** wyjaśniają **dlaczego potrzebujemy** systemu w kontekście organizacji

- jakie są spodziewane korzyści
- utracone korzyści gdy go nie będzie
- ... (**cele dalsze**)

- Poprawa obsługi klienta (nowe usługi)
- Zwiększenie konkurencyjności własnych produktów, poprawa jakości
- Lepsze szacowanie własnych zasobów
- Dodatkowe argumenty marketingowe
- Obniżka kosztów obsługi
- Zwiększenie własnej płynności finansowej
- ...

oczekiwane produkty i usługi (**cele bliższe**)



✓ **Rozwiązania** projektowe określają

- **zakres systemu** i wykorzystanie dla realizacji wymagań
- budowę i działanie systemu
- drogę dojścia do systemu docelowego

Wymagania przekazują uzasadnienie dla potrzeby wytworzenia systemu - określają pośrednio cele systemu i wyjaśniają na ile i dlaczego są ważne

✓ **Wymagania** względem systemu definiują:

- **usługi** dostarczane przez system jego otoczeniu,
- wymaganą **jakość** tych usług,
- określają **ograniczenia**, w ramach których system jest realizowany oraz użytkowany

Projekt informatyczny a projekt PRO

Projekt informatyczny:

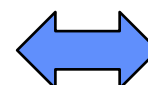
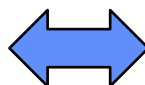
planowa działalność, obejmująca zestaw zadań mających wspólny cel – wytworzenie systemu informatycznego lub produktu (usługi) programowego; w warunkach ograniczeń

każda!

Sukces projektu:


*osiągnięcie zakładanego celu w zakładanym czasie
i przy zakładanych kosztach
+ satysfakcja klienta,
+ pozytywne skutki dla społeczności, jednostek*

- systematyczna realizacja
- metodyczna
- udokumentowanie
- znajomość wszystkich aspektów



Klient

Podstawowe charakterystyki projektów informatycznych

- Rodzaj projektu (analityczny, wytwórczy,...)
 - Innowacyjność / typowość projektu
- Cel projektu 
- Dziedzina
- Znajomość i stopień stabilności wymagań
- Wielkość, złożoność projektu
 - Technologia
- Czas na wykonanie projektu
- Zespół; jego wielkość i umiejętności poszczególnych osób

Cele projektów

➤ Cele dalsze

- związane z korzyściami jakie chcemy uzyskać poprzez informatyzację (użycie / poprawę środków informatyki), np

Klient: zwiększenie wydajności produkcji, jakości obsługi, kompletności informacji albo zasięgu klientów naszego biznesu

Zamawiający: zdobycie renomy, realizacja kontraktu

➤ Cele bliższe

- Określające jaki produkt lub usługę należy dostarczyć poprzez przeprowadzenie projektu

np. analiza celowości..., aplikacja internetowa (z bazą danych), system sterowania, komunikacja pomiędzy stacjami nad/odb.,...

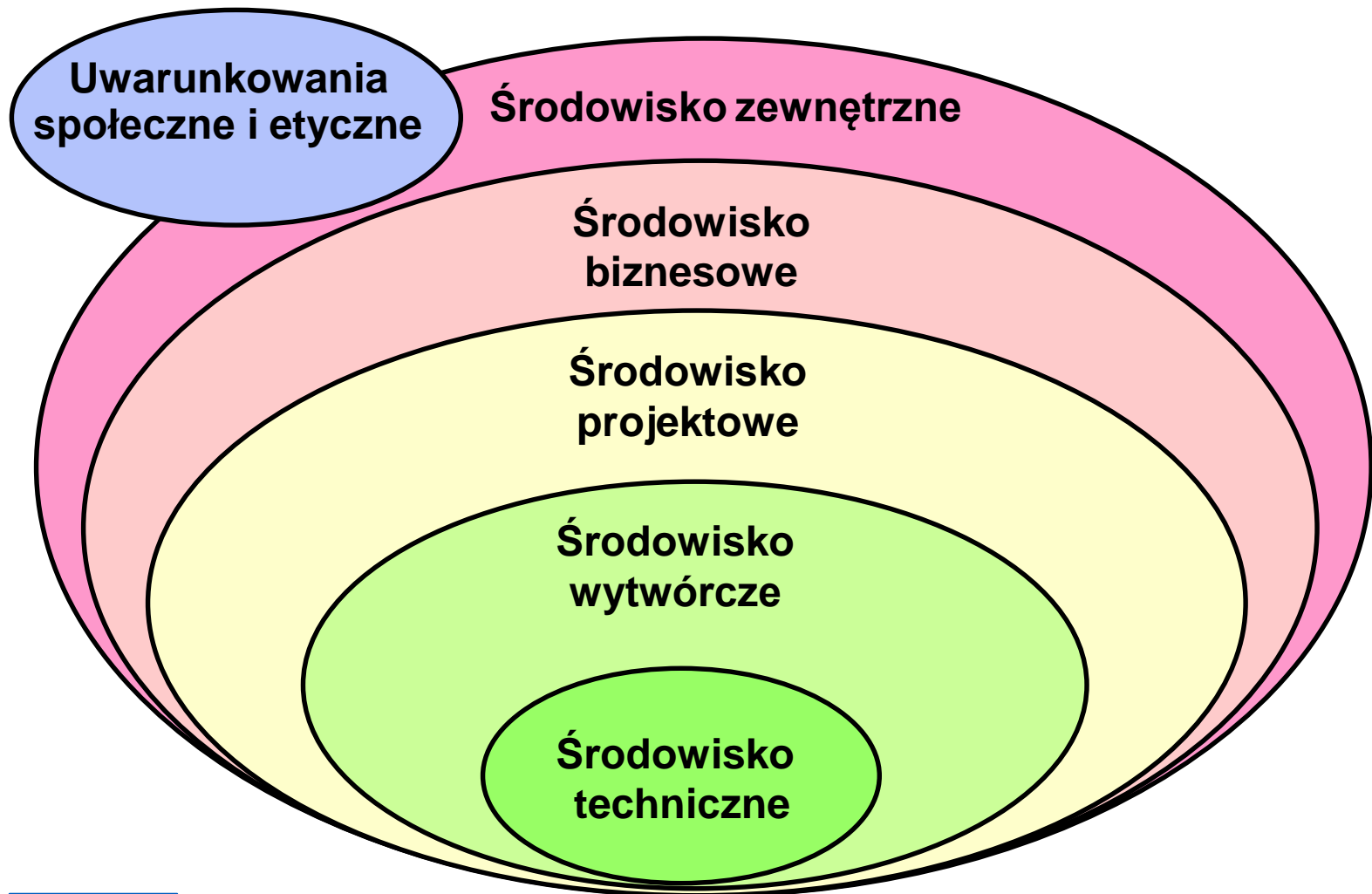
Osiągnięcie celów bliższych prowadzi do realizacji celów dalszych

Wybrane charakterystyki projektów informatycznych

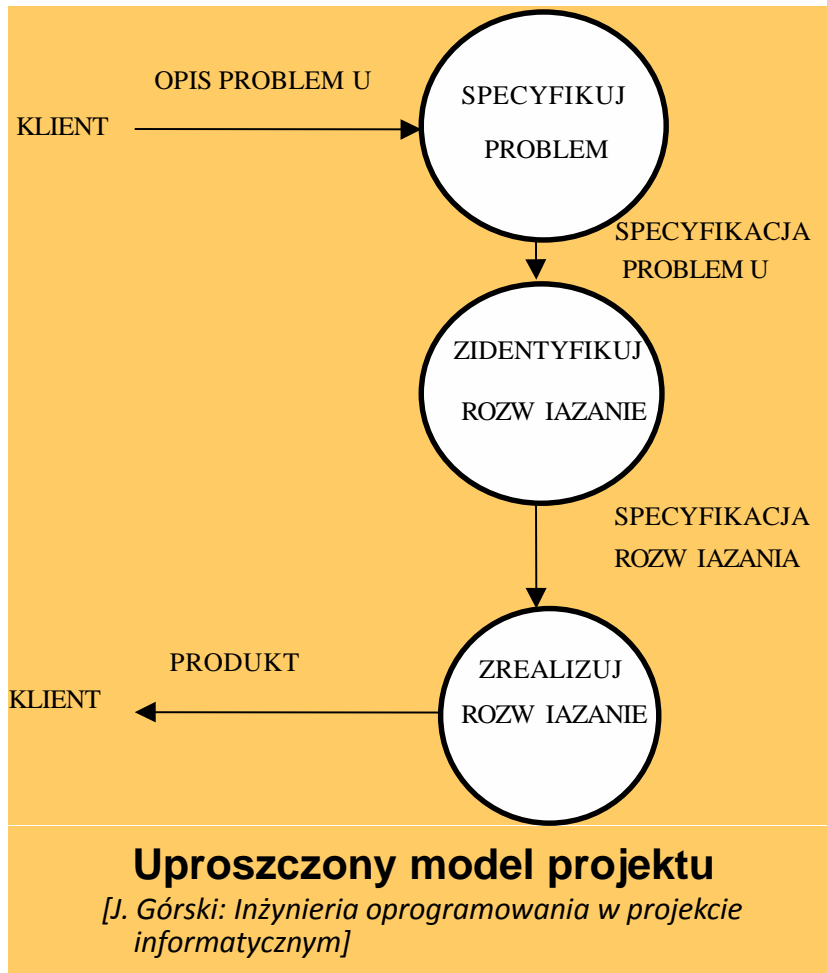
- **Dziedzina**
- **Wielkość, złożoność projektu**
- **Innowacyjność / typowość projektu**
- **Czas na wykonanie projektu**
- **Znajomość i stopień stabilności wymagań**
- **Wielkość zespołu i umiejętności
poszczególnych osób**
- **Dalszy rozwój systemu**
- **...**

Projekt w szerszym kontekście

J. Górski



Model i ważniejsze atrybuty projektu informatycznego



- **początek, koniec**
 - *rola Klienta!*
- **fazy / etapy**
- **produkty**
 - etapowe, finalne
- **zadania**
 - techniczne i menedżerskie
 - zwykle dzielone na podzadania i czynności
- **czas trwania, koszt**
- **zasoby**
 - ludzie, infrastruktura,
 - czas
 - pieniądze, ...

Właściwości projektu (zapewne każdego)

- **jednostkowość i złożoność**
- **ograniczoność środków**
- **wymaga pogodzenia pracy ludzkiej, zasobów, kapitału i czasu**
- **włącza w swój obręb różnych ludzi, funkcje i organizacje**
- **wymaga planowania, koordynacji i sterowania**

Właściwości projektu informatycznego

- konsekwencje *specyfiki produktu programowego*
- tendencja do jak najwcześniejszego podjęcia implementacji
- rozległość dziedzin problemów wymagająca specjalistycznej wiedzy
- niestabilność wymagań

- trójkąt wymiarów projektu



warto zajrzeć: <https://support.office.com/pl-pl/article/Tr%C3%B3jk%C4%85t-projektu-8c892e06-d761-4d40-8e1f-17b33fdcf810>

- realizacja w układzie klient - wykonawca

Udziałowcy (interesariusze) projektu

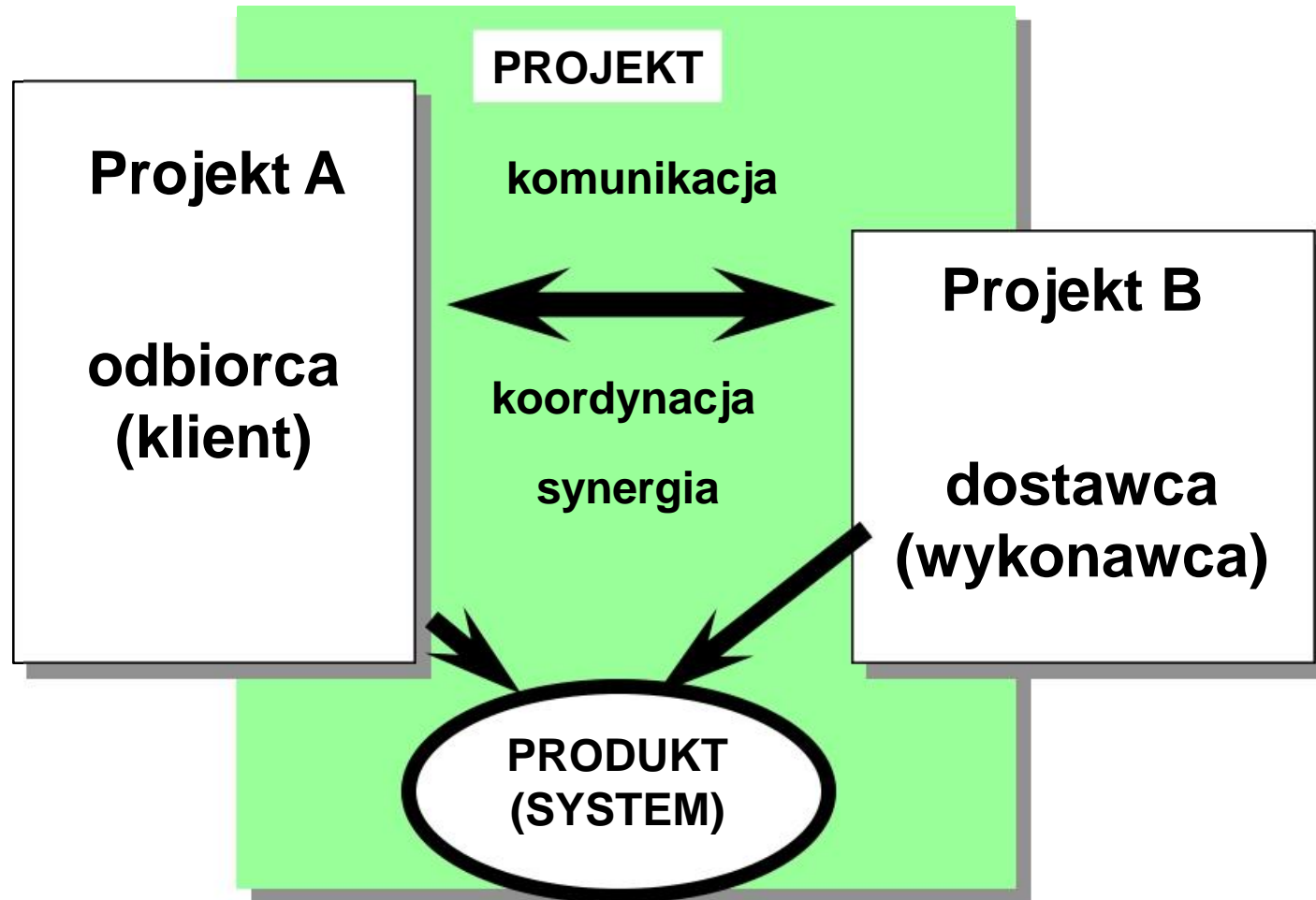
Udziałowcem projektu jest każdy podmiot (niekoniecznie ożywiony), który ma uzasadnione prawo wywarcia (pośrednio lub bezpośrednio) wpływu lub może znaleźć się pod wpływem rozpatrywanego projektu

Punkt widzenia - perspektywa widzenia projektu lub systemu; koncentracja uwagi na wybranym aspekcie



- **klient**
 - zamawiający, użytkownik, operator, ...
- **wykonawca, dostawca**
 - zespół projektowy
 - wykonawcy: analitycy, programiści, testerzy,...
 - poddostawcy, eksperci...
- **oprogramowanie, urządzenie współpracujące**
- **udziałowcy pośredni**
 - klienci klienta, firma i klienci wykonawcy, zewnętrzni beneficjenci lub ofiary systemu, ...

Przedsięwzięcie informatyczne to są dwa projekty! *J. Górski*



Zakres projektu

Zakres projektu - zakres pracy koniecznej, by dostarczane oprogramowanie i usługi (**produkty projektu**) posiadały wymagane cechy

- **Zakres projektu** — **zadania i ich produkty**
- **Zakres projektu** — **pokrycie cyklu życia**
- **Zakres projektu a zakres systemu**



Cykl życia projektu

Cykl życia projektu – zbiór (zwykle sekwencja) faz projektu, od jego rozpoczęcia do zakończenia, wyodrębnianych z uwagi na potrzeby sterowania nim przez organizację (organizacje) zaangażowaną w projekt.

- **faza projektu** grupuje pewne zadania
- z fazą projektu związany jest zbiór jej *produktów*
- zwykle kończy się *przeglądem końcowym fazy*, robionym w celu
 - stwierdzenia wykonania jej produktów, wykrycia i usunięcia usterek
 - zdecydowania, czy projekt powinien przejść do fazy następnej

Cykl życia projektu określa

- ⇒ początek i koniec projektu
 - akcje związane z rozpoczęciem (*ustanowienie*) i zakończeniem projektu
- ⇒ wyodrębniane fazy, ich powiązanie, przekazywane produkty
- ⇒ zakres prac poszczególnych faz
 - często związany z technologią
- ⇒ wykonawców faz
- ⇒ sposób zarządzania projektem

Nie należy utożsamiać cyklu życia projektu i cyklu życia oprogramowania (produktów programowych)!

**Ale model wytwarzania lub
ewolucji silnie wpływa na cykl
(kształt, strategię prowadzenia)
projektu wytwórczego, ewolucji
oprogramowania**



Projekt na PRO

1. Dobór projektu

- dziedzina i projekt
- kontekst, cele bliższe i produkty
- skąd wymagania, klient
- charakter wytwórczy, a zakres projektu ?
- stopień trudności i technologia



2. Zakres projektu

- zbiór funkcji? podsystemy użytkowe?
- zawartość informacyjna systemu (bazy danych)?
- dodatkowe wymagania, np. wobec frontendu?
- co poza wytwarzaniem – walidacja? wypełnienie bd? wdrożenie?..

Projekt na PRO

3. Zaplanowanie

- zagrożenia ?
- wizja *technologiczno – architektoniczna*,
- metodyka realizacji
 - etapy, produkty etapowe, harmonogram
- zespół, podział pracy
- kierownik?
- komunikacja
 - zapewnianie (kontrola) jakości,
 - dystrybucja produktów etapowych, wersjonowanie
 - środowisko (technolog., komunikacyjne, dokumentacyjne,...)
 - dokumentowanie



Projekt na PRO

4. Realizacja

- kończenie i inicjowanie etapów ?
- rozstrzyganie sporów
- zmiana obciążeń członków zespołu
- retrospekcja
- postępowanie w przypadku odstępstw od:
 - harmonogramu
 - jakości, zakresu, funkcji
 - zmian technologii
 - braków / opóźnień dokumentacji



Wskazówki do analizy projektów na ćwiczeniach

- Cel, kontekst projektu
- Udziałowcy
- Produkty (finalne, większe etapowe)
- Zakres projektu
- Zagrozenia
- Metodyka realizacji, etapy, ich omówienie; na ile dokumentacja jest z tym zgodna
- Technologia i narzędzia
- Główne rozwiązania architektoniczne
- Testy
- Co było sukcesem? Co się nie udało?
- Inne uwagi, oceny, sugestie,...



Inżynier oprogramowania (software engineer)

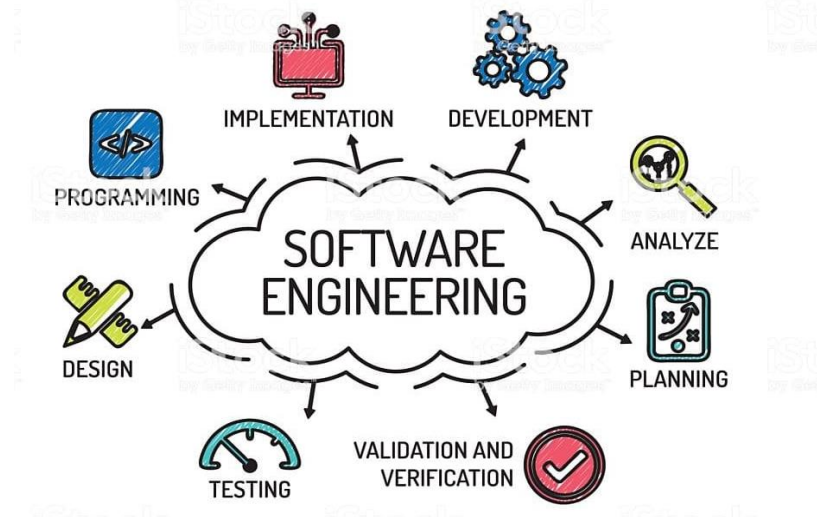
- zajęcie wpisane na oficjalną listę zawodów w roku 1990 (za granicą)
- według *Money Magazine* i *Salary.com* – zawód uznany w 2006 roku za najlepszy w Stanach Zjednoczonych, pod względem możliwości rozwoju, wysokości pensji, poziomu stresu, elastyczności godzin pracy i środowiska pracy
- wg Careercast: 11 miejsce na liście najmniej stresujących zawodów w 2019 r. (spokojna praca przed komputerem z bardzo dobrymi perspektywami zatrudnienia)



- W 2023 Application Software Developer nr. 1 dla maturzystów.

Znaczenie inżynierii oprogramowania

- gospodarki wszystkich rozwiniętych krajów zależą od oprogramowania
- ponadto, samo wytwarzanie oprogramowania jest też poważną gałęzią gospodarki narodowej każdego rozwiniętego kraju
- coraz więcej i więcej systemów wymaga niezawodnego oprogramowania
- istnieje potrzeba systematyzacji i uporządkowania procesu wytwarzania oprogramowania celem ułatwienia tworzenia oprogramowania wysokiej jakości



Stanowiska pracy

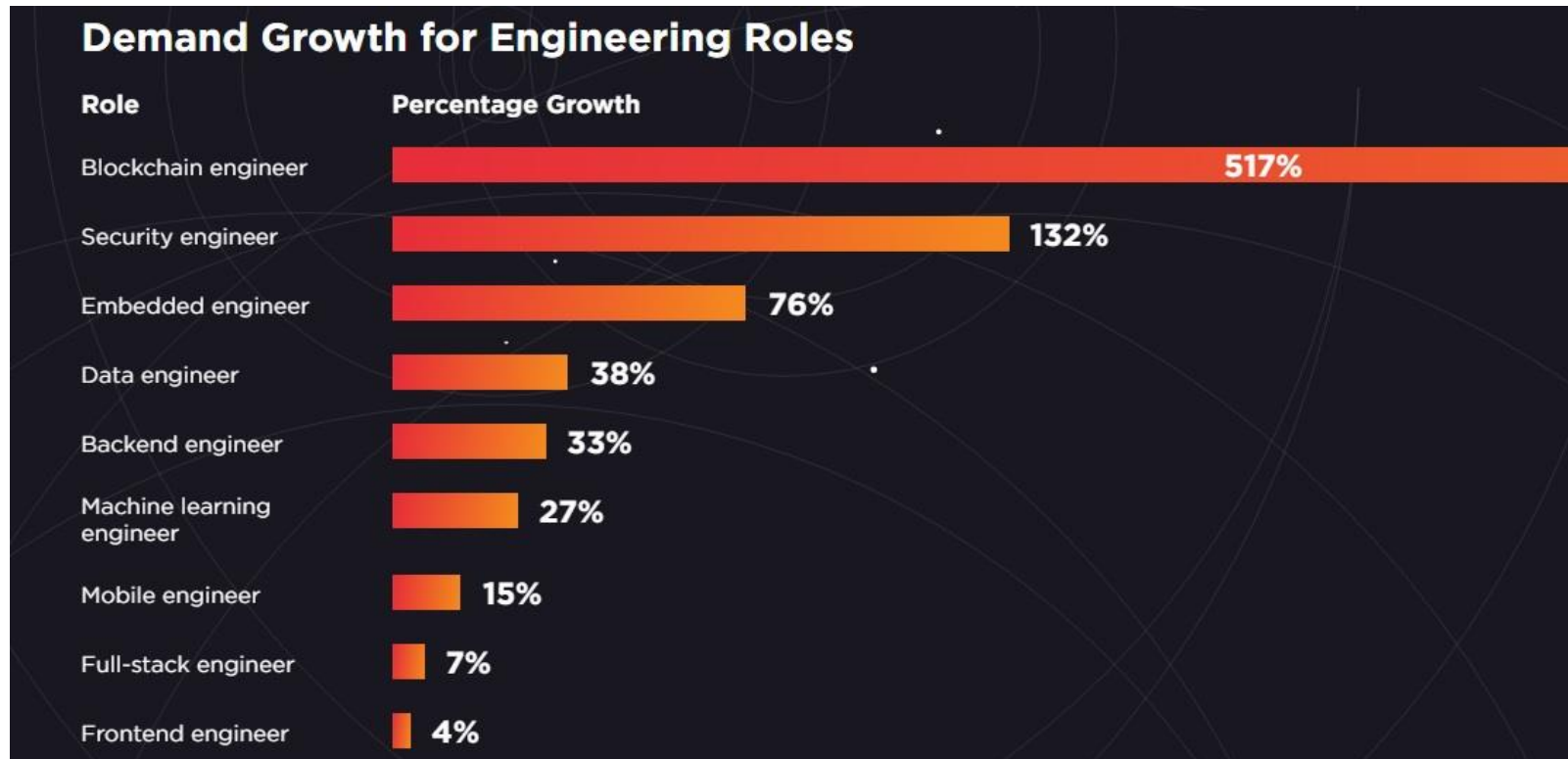
<https://news.codecademy.com/what-does-a-software-engineer-do/>

- Software engineer
- Project manager
- Database architect
- Software architect
- Project manager
- Team leader
- Data Scientist
- Programista ORACLE, SQL, .NET etc.
- Administrator hurtowni danych
- Wszelkie inne rodzaje (z danych ok. 36000 ofert pracy na całym świecie)





Software engineer jobs



<https://www.houseofbots.com/news-detail/11583-4-9-hottest-software-engineering-jobs-in-demand-&-the-high-salaries-they-command-across-world>



POLSKO-JAPOŃSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

Wynagrodzenie

Według danych portalu pracuj.pl przeciętne zarobki na stanowisku inżyniera oprogramowania wynosiły 11.000 brutto.

Ciekawostki:

- 93% inżynierów oprogramowania to mężczyźni
- 86% praca na umowę o pracę
- Praca mobilna
- Duże benefity (w tym medyczne)

A Day in the Life of a Software Engineer

Entry-Level Software Engineer



Implement
new
features

React
Ruby
Python

learn new
languages &
frameworks



Maintain
existing code



Pair
programming



Resolve simple
bugs / errors



Refactor code



Practice test driven
development (TDD)

Decide which
framework &
libraries to use



Architect API
endpoints &
data models



Mentor junior
engineers



Attend sprint planning
meetings with product
managers & designers



Code
reviews



Refactor legacy code



Architect scalable
systems



Senior Software Engineer

