

INŻYNIERIA OPROGRAMOWANIA

wykład 1: zakres wiedzy

dr inż. Leszek Grocholski
pok. 236

Zakład Inżynierii Oprogramowania
Instytut Informatyki
Uniwersytet Wrocławski

INŻYNIERIA OPROGRAMOWANIA

SWEBOK 2014

***Guide to the
Software Body of Knowledge
(<http://www.swebok.org>)***

***A project of the IEEE Computer Society
Professional Practices Committee***

**IEEE Computer Society (<http://computer.org>)
(Institute of Electrical and Electronics Engineers)**

Przedmiot inżynierii oprogramowania

INŻYNIERIA OPROGRAMOWANIA jest praktyczną wiedzą techniczną dotyczącą wszystkich procesów dotyczących: wytwarzania, wdrażania i utrzymania oprogramowania.

Traktuje oprogramowanie jako produkt, który ma spełniać potrzeby techniczne, ekonomiczne lub społeczne.

Wiedza praktyczna

- tworzona przez praktyków a nie naukowców,
- przeznaczona dla praktyków,
- wiedza interdyscyplinarna – socjologia, psychologia itd.,
- jej wynikiem są najlepsze praktyki, standardy, normy, metodyki, które przeznaczone są do bardzo konkretnych zastosowań.

Przedmiot inżynierii oprogramowania

■ ■

Produkcja oprogramowania jest procesem składającym się z wielu etapów. **Kodowanie (pisanie programów) jest tylko jednym z nich, niekoniecznie najważniejszą.**

W USA spada zapotrzebowanie na programistów (z 17% do 9%) a rośnie rynek pracy dla inżynierów oprogramowania (z 28 % do 6%) dane pochodzą z lat 2018 – 2019.

Inżynieria oprogramowania jest wiedzą empiryczną, syntezą doświadczenia tysięcy ośrodków zajmujących się budową oprogramowania.

Praktyka pokazała, że w inżynierii oprogramowania nie ma miejsca stereotyp „od teorii do praktyki”. Teorie, szczególnie „zmatematyzowane” teorie, okazały się dramatycznie nieskuteczne w praktyce.

Inżynieria oprogramowania

- główny powód stosowania

Głównym powodem stosowania wiedzy dostarczanej przez inżynierię oprogramowania jest problem rosnącej złożoności oprogramowania !!!

Trochę historii:

- 43 lat temu nie było komputerów osobistych - PC,
- 30 lat temu nie było internetu,
- 20 lat temu nie było smartfonów, Google, Facebook

Co już się rozpoczęło i co nas czeka:

- internet rzeczy małych (np. palety) i dużych (samochody),
- coraz bardziej inteligentne miasta,
- klienci potrzebują rozwiązań a nie oprogramowania, komputerów itd.,
- ogromna ilość transmitowanych i gromadzonych informacji - co w 2030r ?.

Zasługą inżynierii oprogramowania jest to, że od lat procent przedsięwzięć informatycznych które kończą się sukcesem jest od stały i wynosi ok 30 %

INŻYNIERIA OPROGRAMOWANIA

HISTORIA INŻYNIERII OPROGRAMOWANIA

- **software** 1958 – John Tukey, słynny statystyk użył pierwszy raz słów **software** i **hardware**.
- **software engineering** - tytuł konferencji NATO, która odbyła się w Niemczech w roku 1968.
- W roku 1972 IEEE Computer Society (Instytut inżynierów Elektryków i Elektroników) zaczęło wydawać czasopismo **Transactions on Software Engineering**.
- Komitet IEEE Computer Society **odpowiedzialny za stanowienie standardów inżynierii** oprogramowania – 1976 r.
- Pierwsza norma – IEEE Std 730 – dotycząca **zapewnienia jakości oprogramowania** – 1979 r.

INŻYNIERIA OPROGRAMOWANIA

Normy IEEE dot. SE do roku 1999:

- Customer and Terminology Standards - 9
- Process standards – 14
- Product standards – 5
- Resource and technique standards -12

Po roku 1999 – kolejnych 100 norm

INŻYNIERIA OPROGRAMOWANIA

ACM – Association for Computing Machinery

1995 r. - IEEE i ACM zauważyły konieczność opracowania odpowiedniego kompedium wiedzy dla działalności związanej z inżynierią oprogramowania - SWEBOOK

Kompedium wiedzy i regulacje prawne, które stanowiły by podstawę: decyzji przemysłowych, certyfikatów zawodowych i programów edukacyjnych.

INŻYNIERIA OPROGRAMOWANIA

Wspólny komitet IEEE i ACM postanowił dla inżynierii oprogramowania:

- Zdefiniować podstawowy zakres wiedzy i rekomendowanych zasad działania (recommended practice).
- Zdefiniować zawodowe standardy etycznego, profesjonalnego postępowania (code of ethical and professionals practice).
- Zdefiniować programy nauczania (educational curricula) dla szkół zawodowych i wyższych.

INŻYNIERIA OPROGRAMOWANIA

- Pierwsze wydanie SWEBOK - 1998 roku. Aktualnie wersja 3 z 2014 r.
- Kodeks dot. etycznych zasady postępowania zawodowego został opracowany i zaaprobowany przez IEEE i ACM w roku 1998.
- Wytyczne dla celów nauczania oraz wymagane zakresy wiedzy (curriculum guidelines) zostały opracowane w roku 2004 - Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (<http://www.computer.org/portal/>)

INŻYNIERIA OPROGRAMOWANIA

- Steve McConnell, Roger Pressman, Ian Sommerville - światowe autorytety, autorzy podręczników inżynierii oprogramowania (wydanych również po polsku)
stanowili panel ekspertów biorących udział w opracowywaniu przewodnika SWEBOOK.
- Z Polski w pracach komisji IEEE/ACM uczestniczył prof. Janusz Górski z Wydziału Elektroniki, Telekomunikacji i Informatyki Katedra Inżynierii Oprogramowania, Politechniki Gdańskiej.

INŻYNIERIA OPROGRAMOWANIA

10 obszarów wiedzy (Knowledge Areas - KA) SE. Każdy z obszarów wiedzy został opisany w poszczególnych rozdziałach przewodnika:

1. Zarządzanie wymaganiami dot. oprogramowania (inżynieria wymagań)
 2. Projektowanie oprogramowania
 3. Wytwarzanie oprogramowania (w tym programowanie)
 4. Testowanie oprogramowania
 5. Wdrożenie i eksploatacja (utrzymywanie) oprogramowania
-
6. Zarządzanie zmianami i konfiguracjami
 7. **Zarządzanie wytwarzaniem oprogramowania**
 8. Procesy w życiu oprogramowania
 9. Metody i narzędzia inżynierii oprogramowania
 10. **Jakość oprogramowania**

INŻYNIERIA OPROGRAMOWANIA

Co stanowi źródło wiedzy inżynierii oprogramowania wg SWEBOK ? :

- normy i standardy : IEEE, ISO/IEC, SEI,...
zawierają m in definicje
- podręczniki,
- publikacje,
- strony www.

SWEBOK rekomenduje podstawową literaturę.

INŻYNIERIA OPROGRAMOWANIA

Przykład:

Zarządzanie wymaganiami dot. oprogramowania

Wymaganie jest zdefiniowane jako CECHA, która musi być ustalona w celu rozwiązania pewnego problemu dotyczącego rzeczywistego świata.

W skład obszaru wiedzy dot. zarządzania wymaganiami oprogramowania wchodzi następujące podobszary:

- Podstawy definiowania wymagań na oprogramowanie;
- Proces definiowania wymagań;
- Uzyskiwanie wymagań od zainteresowanych stron;
- Analiza wymagań (np. ważność);
- Specyfikacja wymagań;
- Ocena wymagań;
- Rozważania praktyczne dotyczące inżynierii wymagań.

INŻYNIERIA OPROGRAMOWANIA

Przykład: Zarządzanie wytwarzaniem oprogramowania

Zarządzanie wytwarzaniem oprogramowania odwołuje się do wiedzy dot. zarządzania przedsiębiorstwem i pomiarów procesów inżynierii oprogramowania. W skład obszaru wiedzy wytwarzanie oprogramowania wchodzi następujące podobszary:

- Rozpoczęcie i definiowanie zakresu przedsięwzięcia;
- Planowanie przedsięwzięcia;
- Dokumentowanie postępów prac;
- Przeglądy i ocena;
- Zakończenie przedsięwzięcia;
- Miary związane z wytwarzaniem oprogramowania.

INŻYNIERIA OPROGRAMOWANIA

SWEBOK – określa poziom wiedzy inżyniera oprogramowania zdobytej podczas 3, 4 letniej edukacji.

Do określania niezbędnego poziomu wiedzy wykorzystano tzw. taksonomie celów edukacyjnych Blooma [5].

Oceniono, że dla skutecznego korzystania z metod inżynierii oprogramowania opisanych w SWEBOK niezbędny jest pewien minimalny poziom znajomości zagadnień.

W załączniku SWEBOK podano, jakie są wymagane minima dla poszczególnych obszarów. Wytyczne te mogą pomóc w przygotowaniu materiałów szkoleniowych, opisie obowiązków, planowaniu rozwoju pracownika, szkoleniach zawodowych, jak również (przede wszystkim?) w tworzeniu programów edukacyjnych na uczelniach.

INŻYNIERIA OPROGRAMOWANIA

Cele nauczania wg Blooma to:

znajomość zagadnień, zrozumienie, stosowanie wiedzy, analiza, synteza oraz ocena.

Cele nauczania 221 podobszarów wiedzy SE wg SWEBOK

• Stosowanie	109	49,32%
• Zrozumienie	80	36,20%
• Analiza	31	14,03%
• Znajomość	1	0,45%
• Synteza	0	0,0 %
• Ocena	0	0,0 %

• Razem	221	100 %
---------	-----	-------

INŻYNIERIA OPROGRAMOWANIA

**Przykład 1: Podobszar wiedzy: projektowanie obiektowe w dziedzinie (wiedzy)
projektowanie architektury.**

Autorzy SWEBOOK sugerując cel nauczania na poziomie analizy. Stwierdzają, że niższy poziom nauczania (znajomość, zrozumienie lub stosowanie) są niewystarczające, natomiast wyższe (synteza lub ocena) są niepotrzebne.

INŻYNIERIA OPROGRAMOWANIA

Przykład 2: Konstruowanie testów w dziedzinie (wiedzy) wytwarzanie oprogramowania

Wymagania wiedzy na poziomie stosowania.

Wiedza na niższym poziomie (znajomość, zrozumienie) jest niewystarczająca, a na wyższym (analiza, synteza lub ocena) zbędna lub koszt jej zdobycia jest nieadekwatny do pożytku.

INŻYNIERIA OPROGRAMOWANIA

**Może zwracać uwagę fakt, że
implementacja zaleceń SWEBOK z
reguły nie wymaga wiedzy na
najwyższych poziomach
- analizy i syntezy !**

INŻYNIERIA OPROGRAMOWANIA

Ostatni rozdział SWEBOK określa dziedziny związane z inżynierią oprogramowania, wymieniając wśród nich:

- inżynierię komputerową
- inżynierię systemów
- informatykę teoretyczną (computer science)
- zarządzanie zespołami ludzkimi
- matematykę
- zarządzanie przedsięwzięciem (projektem)
- zarządzanie jakością
- ergonomię oprogramowania

INŻYNIERIA OPROGRAMOWANIA

- Przykładowo w zakresie informatyki teoretycznej SWEBOK wymienia obszary:

struktury dyskretne, podstawy programowania, algorytmy i złożoność, organizacja i architektura, systemy operacyjne, obliczenia sieciowe, języki programowania, interfejs człowiek – maszyna, wizualizacja i grafika, systemy inteligentne, metody i modele numeryczne.

INŻYNIERIA OPROGRAMOWANIA

- Jak nauczać inżynierię oprogramowania ?
Oczywiście na podstawie SWEBOK

Główna korzyść:

- Zgodność ze standardami (czytaj np.. programami studiów i normami przemysłowymi) światowymi.

INŻYNIERIA OPROGRAMOWANIA - INACZEJ

- Sposoby prowadzenia przedsięwzięć informatycznych.
- Techniki planowania, szacowania kosztów, harmonogramowania i monitorowania przedsięwzięć informatycznych.
- Metody analizy i projektowania systemów.
- Techniki zwiększania niezawodności oprogramowania.
- Sposoby testowania systemów i szacowania niezawodności.
- Sposoby przygotowania dokumentacji technicznej i użytkowej.
- Procedury kontroli jakości.
- Metody redukcji kosztów konserwacji (usuwania błędów, modyfikacji i rozszerzeń)
- Techniki pracy zespołowej i czynniki psychologiczne wpływające na efektywność pracy.

PROBLEMY IO (1)

Sprzeczność pomiędzy odpowiedzialnością, jaka spoczywa na współczesnych SI, a ich zawodnością wynikającą ze złożoności i ciągle niedojrzałych metod tworzenia i weryfikacji oprogramowania.

Ogromne koszty utrzymania oprogramowania.

Niska kultura ponownego użycia wytworzonych komponentów projektów i oprogramowania; niski stopień powtarzalności poszczególnych przedsięwzięć.

Długi i kosztowny cykl tworzenia oprogramowania, wysokie prawdopodobieństwo niepowodzenia projektu programistycznego.

Długi i kosztowny cykl życia SI, wymagający stałych (często globalnych) zmian.

Eklektyczne, niesystematyczne narzędzia i języki programowania.

PROBLEMY IO (2)

Frustracje projektantów oprogramowania i programistów wynikające ze zbyt szybkiego postępu w zakresie języków, narzędzi i metod oraz uciążliwości i długotrwałości procesów produkcji, utrzymania i pielęgnacji oprogramowania.

Uzależnienie organizacji od systemów komputerowych i przyjętych technologii przetwarzania informacji, które nie są stabilne w długim horyzoncie czasowym.

Problemy współdziałania niezależnie zbudowanego oprogramowania, szczególnie istotne przy dzisiejszych tendencjach integracyjnych.

Problemy przystosowania istniejących i działających systemów do nowych wymagań, tendencji i platform sprzętowo-programowych.

Walka z problemami

Stosowanie technik i narzędzi ułatwiających pracę nad złożonymi systemami;

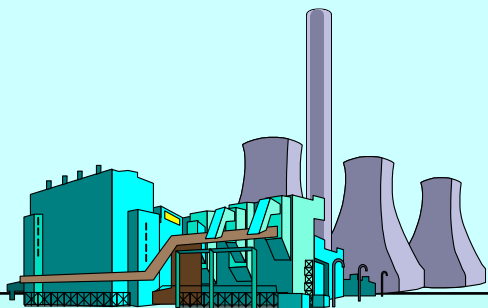
Korzystanie z metod wspomagających analizę nieznanych problemów oraz ułatwiających wykorzystanie wcześniejszych doświadczeń;

Usystematyzowanie procesu wytwarzania oprogramowania, tak aby ułatwić jego planowanie i monitorowanie;

Wytworzenie wśród producentów i nabywców przekonania, że budowa dużego systemu wysokiej jakości jest zadaniem wymagającym profesjonalnego podejścia.

Podstawowym powodem problemów oprogramowania jest złożoność produktów informatyki i procesów ich wytwarzania.

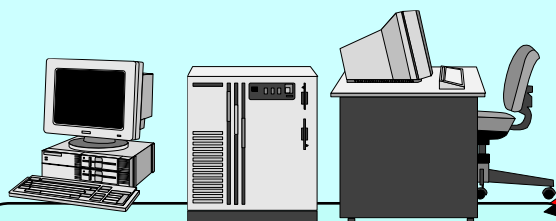
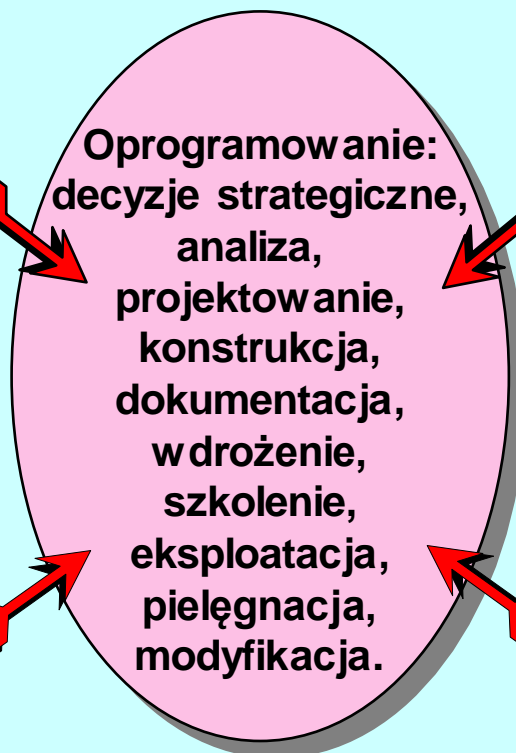
Źródła złożoności projektu oprogramowania



Dziedzina problemowa,
obejmująca ogromną liczbę
wzajemnie uzależnionych
aspektów i problemów.



Zespół projektantów
podlegający ograniczeniom
pamięci, percepcji, wyrażania
informacji i komunikacji.



Środki i technologie informatyczne:
sprzęt, oprogramowanie,
sieć,
języki, narzędzia,
udogodnienia.



Potencjalni użytkownicy:
czynniki psychologiczne,
ergonomia, ograniczenia
pamięci i percepcji, skłonność
do błędów i nadużyć, tajność,
prywatność.

JAK WALCZYĆ ZE ZŁOŻONOŚCIĄ ?

Zasada dekompozycji:

rozdzielenie złożonego problemu na podproblemy, które można rozpatrywać i rozwiązywać niezależnie od siebie i niezależnie od całości.

Zasada abstrakcji:

eliminacja, ukrycie lub pominięcie mniej istotnych szczegółów rozważanego przedmiotu lub mniej istotnej informacji; wyodrębnianie cech wspólnych i niezmiennych dla pewnego zbioru bytów i wprowadzaniu pojęć lub symboli oznaczających takie cechy.

Zasada ponownego użycia:

wykorzystanie wcześniej wytworzonych schematów, metod, wzorców, komponentów projektu, komponentów oprogramowania, itd.

Zasada sprzyjania naturalnym ludzkim własnościom:

dopasowanie modeli pojęciowych i modeli realizacyjnych systemów do wrodzonych ludzkich własności psychologicznych, instynktów oraz mentalnych mechanizmów percepcji i rozumienia świata.

ZROZUMIENIE TEGO CO TRZEBA ZROBIĆ

Projektant i programista muszą dokładnie wyobrazić sobie problem oraz metodę jego rozwiązania. Zasadnicze procesy tworzenia oprogramowania zachodzą w ludzkim umyśle i nie są związane z jakimkolwiek językiem programowania.

Pojęcia **modelowania pojęciowego** (*conceptual modeling*) oraz **modelu pojęciowego** (*conceptual model*) odnoszą się do procesów myślowych i wyobrażeń towarzyszących pracy nad oprogramowaniem.

Modelowanie pojęciowe jest wspomagane przez środki wzmacniające ludzką pamięć i wyobraźnię. Służą one do przedstawienia rzeczywistości opisywanej przez dane, procesów zachodzących w rzeczywistości, struktur danych oraz programów składających się na konstrukcję systemu.

METODYKA (METODOLOGIA)

Metodyka jest to zestaw pojęć, notacji, modeli, języków, technik i sposobów postępowania służący do analizy dziedziny stanowiącej przedmiot projektowanego systemu oraz do projektowania pojęciowego, logicznego i/lub fizycznego.

Metodyka jest powiązana z **notacją** służącą do dokumentowania wyników faz projektu (pośrednich, końcowych), jako środek wspomagający ludzką pamięć i wyobraźnię i jako środek komunikacji w zespołach oraz pomiędzy projektantami i klientem.

Metodyka ustala:

- fazy projektu, role uczestników projektu,
- modele tworzone w każdej z faz,
- scenariusze postępowania w każdej z faz,
- reguły przechodzenia od fazy do kolejnej fazy,
- notacje, których należy używać,
- dokumentację powstającą w każdej z faz.

INŻYNIERIA OPROGRAMOWANIA

Dziękuję za uwagę