INŻYNIERIA OPROGRAMOWANIA

(ang. Software Engineering, Engineering of Software)

Definicje:

1. **Powszechnie przyjęta nazwa procesu tworzenia oprogramowania.**
2. **„Inżynieria oprogramowania” stanowi zbiór umiejętności, pojęć i procedur, mających pomóc ludziom dobrze wykonywać pracę w tworzeniu oprogramowania.**
3. **„Software engineering is the introduction of formal engineering principles to the creation and production of software.**

Inżynieria oprogramowania ma za zadanie doprowadzić do produkcji wysokiej jakości oprogramowania, lecz to wcale nie jest takie pewne!!!

INŻYNIERIA OPROGRAMOWANIA

(ang. Software Engineering, Engineering of Software)

Definicje:

4. ***Wiedza techniczna***, **dotycząca wszystkich *faz cyklu życia oprogramowania*, której celem jest uzyskanie wysokiej jakości *produktu* – *oprogramowania.***

Czyli oprogramowanie to produkt, a więc oprogramowanie powinno być:

- **zgodne z wymaganiami i oczekiwaniami użytkownika**
- **niezawodne**
- **efektywne**
- **łatwe w konserwacji**
- **ergonomiczne**

Historia (Każdy wielki postęp w dziedzinie software'u dokonał się dzięki błędom programowym – w każdym programie są błędy....)

1950-60 – drobne oprogramowanie – cele naukowe

1960-80 – liczniejsze zastosowania komputerów – rozwój języków programowania wyższego poziomu

1980-2014 – gwałtowny rozwój sprzętu i znaczne zwiększenie jego możliwości – próby nadążenia z oprogramowaniem – kompletne fiasko(wiele przedsięwzięć informatycznych nie zakończono ze względu na ich zbyt wielką złożoność bądź też przekroczenie założonego czasu lub budżetu.

„KRYSYS OPROGRAMOWANIA !!!!"

(Rozwój technik programowania nie nadąża za rozwojem sprzętu)

(Software to udana próba optymalizacji błędów komputerowego hardware'u i zwiększanie zasobu błędów poprzez jego doskonalenie...)

Historia („Kryzys oprogramowania”)

„Programista nie może panować na raz nad więcej niż jednym ekranem tekstu programu”

„Komputer wykona to, co zaprogramujesz, a nie to czego oczekujesz”

„Błędy w programie ujawniają się dopiero po przeprowadzeniu kontroli programu”

„Testowanie wskazuje błędy w oprogramowaniu, lecz nie wskazuje jego bezbłędności”

„Nie ma programów bezbłędnych, znalezienie w nich błędów jest tylko kwestią czasu użytkowania”

(Software to udana próba optymalizacji błędów komputerowego hardware’u i zwiększanie zasobu błędów poprzez jego doskonalenie…)

Historia (2) (Każdy wielki postęp w dziedzinie software'u dokonał się dzięki błędom programowym – w każdym programie są błędy....)

lata	programy	błędy	Inżynieria oprogramowania
1950-1960	progr. dla siebie – komputery w nauce	niewielkie błędy	nie istnieje
1960-1970	duże systemy – początki informatyzacji	kosztowne błędy, początki kryzysu	rodzi się
1970-1990	olbrzymie systemy informatyczne – komputery dla mas	wyniszczające błędy, uznanie błędów za rzecz normalną	Inżynieria oprogramowania w rozkwicie
1990-2014	komputer jak powietrze	kryzys oprogramowania	Inżynieria opr. nadal w rozkwicie
przyszłość	granice sztucznej inteligencji	czy myli się człowiek czy maszyna?	granice inż. opr. I genetycznej

Historia (3)

Szacowanie ilości niewykrytych błędów:

1. **wprowadzamy do systemu 1000 sztucznych błędów;**
2. **Każemy programistom wyszukiwać błędy w systemie;**
3. **Zakładając, że znaleziono n błędów w tym k sztucznych oraz n-k prawdziwych ilość błędów prawdziwych:**

   il.bł.praw/1000=(n-k)/k

Szacowanie ilości błędów:

- **nie wykazuje gdzie są błędy;**
- **Zlicza tylko lokalne błędy typowe;**
- **Nie wskazuje przyczyn błędów**

Historia (4)

Analiza przyczyn *kryzysu oprogramowania* wskazała następujące przyczyny:

- **Duża złożoność systemów informatycznych**
- **Niepowtarzalności poszczególnych przedsięwzięć**
- **Brak przejrzystych procedur opisu procesu budowy oprogramowania pozwalających na precyzyjną ocenę stopnia zaawansowania projektu (wpadki: ZUS-Płatnik, NFZ- Świadczeniodawca, PESEL, system rejestracji pojazdów, CELNIK itd.)**
- **Założenie *liniowości* w procesie tworzenia kodu programu**

Propozycje wyjścia z *kryzysu oprogramowania* *zaowocowały powstaniem nowego działu informatyki*

INŻYNIERIA OPROGRAMOWANIA

ZAKRES (1)

Co to jest inżynieria i produkt zwany *oprogramowaniem*

- **analogia do innych specjalności – wiedza techniczna , zawód!!!**

Pojęcia dotyczące tworzenia oprogramowania (odrobina filozofii)

- **zespół czy indywiduum**
- **modele cyklu życia oprogramowania**
- **Szacowanie kosztów oprogramowania**

Zasady tworzenia oprogramowania

- **kontrola intelektualna nad projektem, dziel i zwyciężaj, odbiorcy? od ogółu do szczegółu, dokumentuj!, wszystko to we-wy, co ze zmianami, nie wyważaj otwartych drzwi, weź odpowiedzialność.**

Technologie wytwarzania oprogramowania

- **podejście strukturalne i obiektowe**

***Cykl życia* oprogramowania**

- rola fazy strategicznej, etapy tworzenia oprogramowania (wymagania, specyfikowanie, projektowanie, kodowanie, testowanie, konserwacja)

INŻYNIERIA OPROGRAMOWANIA

ZAKRES (2)

Wymagania

- analiza problemu, interpretacja potrzeb użytkownika (zrozumienie), dzielenie wejść i wyjść, prototypowanie
- właściwości dobrych wymagań: brak elementów nieistotnych, poprawność, kompletność, zwięzłość, precyzja, jasność, jednoznaczność, spójność, możliwość śledzenia, łatwość modyfikacji, możliwość testowania (weryfikacji), wykonalność!!
- przypadki użycia, uczestnicy, warunki początkowe, rozszerzenia

Specyfikowanie

- odbiorcy i ich potrzeby, precyzowanie wymagań-formalizm (precyzja jest celem)
- języki specyfikacji – metody formalne i nieformalne
- oprogramowanie niebezpieczne
- Programowanie specyfikacji – wprowadzenie do języka Prolog

Zarządzanie wymaganiami i specyfikacja systemu

INŻYNIERIA OPROGRAMOWANIA

ZAKRES (3)

Projektowanie
- ogólne zasady projektowania jako etap fazy analizy
- etapy projektowania (wysokiego poziomu i szczegółowe)
- tworzenie modeli obiektowych i strukturalnych,
- Sztuka tworzenia interfejsów
- notacja formalna

Kodowanie
- języki programowania i narzędzia wspomagające (analizatory, skrypty)

Projektowanie z użyciem pseudokodu

Abstrakcyjne typy danych, hermetyzacja i język C

Projektowanie obiektowe
- Obiekty rzeczywiste i programowe, wymagania obiektowe
- Języki obiektowe – Smalltalk, C++, Java - wprowadzenie

Diagramy przepływu danych

Kierowanie etapami projektowania i kodowania

INŻYNIERIA OPROGRAMOWANIA

ZAKRES (4)

Testowanie oprogramowania

- plan testów w trakcie cyklu życia
- Inspekcja a testowanie
- testowanie funkcjonalne i oparte na błędach
- testowanie jednostek i systemu
- testowanie losowe, automatyczne, systematyczne, regresywne
- Zarządzanie testowaniem

Dokumentacja systemu informatycznego

Wdrożenie systemu informatycznego

Zarządzanie projektem informatycznym

- konteksty procesu (interpersonalny, zachowawczy, organizacyjny)
- rola menagera, definiowanie projektu, zespół, sieć krytyczna projektu, ryzyko i niepewność, kontrola projektu, komunikacja i wymiana informacji w projekcie.

Narzędzia CASE w Inżynierii Oprogramowania.

INŻYNIERIA OPROGRAMOWANIA

Literatura

1. Dick Hamlet, Joe Maybee, Podstawy techniczne inżynierii oprogramowania, seria: Inżynieria oprogramowania, wyd. WNT Warszawa 2003.
2. Ian Sommerville, Inżynieria oprogramowania, WNT Warszawa 2003 - Klasyka Informatyki.
3. L.A.Maciaszek, B.L.Liong, Practical Software Engineering – A case Study Approach, Pearson Education Limited 2005.
4. M.Kliszewski, Inżynieria Oprogramowania Obiektowego, WKT-Respekt, 1994.
5. Andrzej Jaszkiewicz, Inżynieria oprogramowania, wyd. Helion Gliwice 1997.
6. Derek Partridge, Artificial Intelligence and Software Engineering, Understanding the Promise of the Future, Glenlake Publishing Company Chicago 1998.
7. J.Graf, Komputerowe Prawa Murphy'ego (czyli zasada, że to co może się nie udać, nie uda się na pewno), wyd. Mark&Technik 1993
8. INTERNET – hasła: Inżynieria Oprogramowania, Software Enginering i pokrewne.

INŻYNIERIA OPROGRAMOWANIA

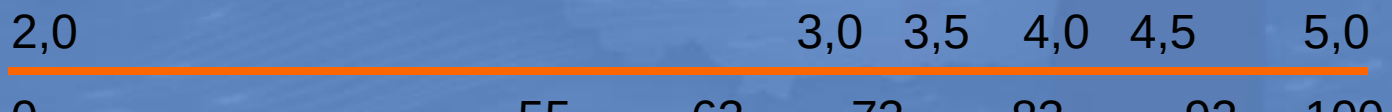
„zasady gry”

Kontakty:

- **dr hab.inż. Marek STANUSZEK – marek.stanuszek@pk.edu.pl (pok.140a)**
  www.pk.edu.pl/~mareks
- **dr inż. Paweł Jarosz – pjarosz@pk.edu.pl**
- **Mgr inż. Filip Krużel – fkruzel@pk.edu.pl**

Warunki zaliczenia

- **zaliczone laboratorium/projekt (zadania i kolokwia)**
- **test 100 pkt. + (obecności)**
- **dodatkowe punkty za aktywność w trakcie wykładu (3/wykład)**
- **Skala ocen/punktów**



INŻYNIERIA OPROGRAMOWANIA

„zasady gry”(2)

1. **Nie jestem „guru” w Inżynierii oprogramowania i moje poglądy mogą się różnić od poglądów specjalistów.**

2. **W tym przedmiocie Państwo macie oczywiście prawo do swoich poglądów, musicie jedynie umieć je wyartykułowć i obronić.**

INŻYNIERIA OPROGRAMOWANIA

(ang. Software Engineering, Engineering of Software)

Inżynieria oprogramowania jest zbudowana na podstawie idei racjonalnego myślenia niezmiernie oczywistej:

Myśląc o czymś bardzo skomplikowanym, nie próbuj robić wszystkiego jednocześnie. Podziel to na mniej złożone części i skoncentruj się kolejno na każdej z nich.

Tak więc

Inżynieria oprogramowania jest to wiedza polegająca m.in. Na umiejętności pracy w zespole, tworzeniu nie tylko kodu, ale specyfikacji i architektury projektu, umiejętności modelowania, weryfikacji i testowania, a także opracowania użytecznych metryk i pielęgnowania wszystkich wytworzonych składników oprogramowania i dokumentacji, oraz umiejętności planowania i zarządzania całym przedsięwzięciem programistycznym.

INŻYNIERIA OPROGRAMOWANIA

Zespół tworzący oprogramowanie

KLIENT

1. **Zarządzanie produktem**
2. **Zarządzanie projektem**
3. **Programowanie**
4. **Testowanie**
5. **Doświadczenie użytkowników**
6. **Wdrażanie**

Paradygmaty Organizacyjne Zespołu

- **HIERARHCIA**
- **ROZPROSZENIE**

PRODUKT → **UŻYTKOWNICY**

Największe błędy powstają na stykach: klient – firma programistyczna, projektant – programista, programista - tester

INŻYNIERIA OPROGRAMOWANIA

Klient

1. Klientowi nigdy nie przyjdzie na myśl ile kosztuje produkt programistyczny (projekt), tylko ile można na tym produkcie zarobić lub zaoszczędzić
2. Żaden klient tak do końca nie wie czego chce
3. Każdy klient wie czego nie chce
4. Żaden klient nie chce tego co mamy już gotowe
5. Nie wie tak do końca co chciałby zamiast tego
6. Jeżeli udało ci się wprowadzić w programie wymagane przez klienta poprawki, wtedy często on z nich rezygnuje
7. Klient żąda większych zmian dokładnie wtedy, kiedy produkt jest już gotowy
8. Nie ma pełnej satysfakcji z produktu przy braku aktywności Klienta

INŻYNIERIA OPROGRAMOWANIA

Użytkownicy

1. Z statystyki rynku wynika iż 80% użytkowników programów stosuje tylko 20% ich funkcji
2. Z kolei 20% użytkowników wymaga 80 procent funkcji których program nie posiada
3. Kiedy opanujesz już dany program, to pojawi się jego nowa wersja czyniąca poprzednią zupełnie przestarzałą
4. Podstawowa wada użytkowników produktów informatycznych to przyzwyczajenia
5. Nie istnieje pojęcie użytkownika zadowolonego (bo nie ma programów bezbłędnych – wyjątek stanowią użytkownicy będący projektantami
6. Ilość wykrytych błędów w oprogramowaniu jest proporcjonalna do ilości użytkowników - wniosek????

INŻYNIERIA OPROGRAMOWANIA

Sukces projektu

1. Zadowolenie klientów – zarządzanie produktem
2. Nie przekraczanie ograniczeń – zarządzanie projektem
3. Zgodność ze specyfikacją – projektant i programista
4. Odpowiednia jakość produktu - tester
5. Dbałość o ułatwienia dla użytkowników (user frendly) – doświadczenia użytkowników
6. Łatwość wdrożenia i pielęgnacja (customer service)

INŻYNIERIA OPROGRAMOWANIA

Ekonomika twórców oprogramowania (1)

1. **Garaż** (twórca działa nieformalnie i ma zamiar sprzedać opracowany produkt bez żadnej formalnej struktury gospodarczej – skrajnie bezpłatna produkcja gratyfikowana uznaniem)
2. **Mała firma programistyczna** (kilka osób sprzedających niewielki zakres produktów – wąska specjalizacja
3. **Duże firmy programistyczne** (podejmują się dużych zleceń z zakresu zaspakajania potrzeb społeczeństwa informacyjnego na których z reguły się wykładają (Computerland, Prokom) – przerzucenie winy na zleceniodawców – adwokaci (ZUS, POJAZD)
4. **Nie wiadomo co?** (Optimus, JTT)
5. **Microsoft** ( )

INŻYNIERIA OPROGRAMOWANIA

Ekonomika twórców oprogramowania (2)

1. **Czas do wprowadzenia produktu na rynek (jego szacowanie zależy od dobrze sporządzonego harmonogramu. Z reguły ograniczenie zbyt ambitnych planów (rezygnacja z pewnych funkcji) pozwala na dotrzymanie terminu.**

2. **Postrzegana jakość oprogramowania – lista „ość”**
   (funkcjonalność – *reklamowane funkcje*, niezawodność – *średni czas awarii*, pewność – *oprogramowanie realizuje to czego oczekujemy i jest odporne na awarie krytyczne*, łatwość użycia – *bez studiowaniu dziesiątków manuali*, komunikatywność – *we-wy do innych programów*, zdolność do pielęgnacji – *łatwość usuwania błędów i dodawania funkcji.)*

   - wydajność (szybkość i efektywność) jest pomijana

   - nie ma możliwości spełnienia wszystkich kryteriów np. funkcjonalność a niezawodność (Microsoft, oprogramowanie sprzętu medycznego)

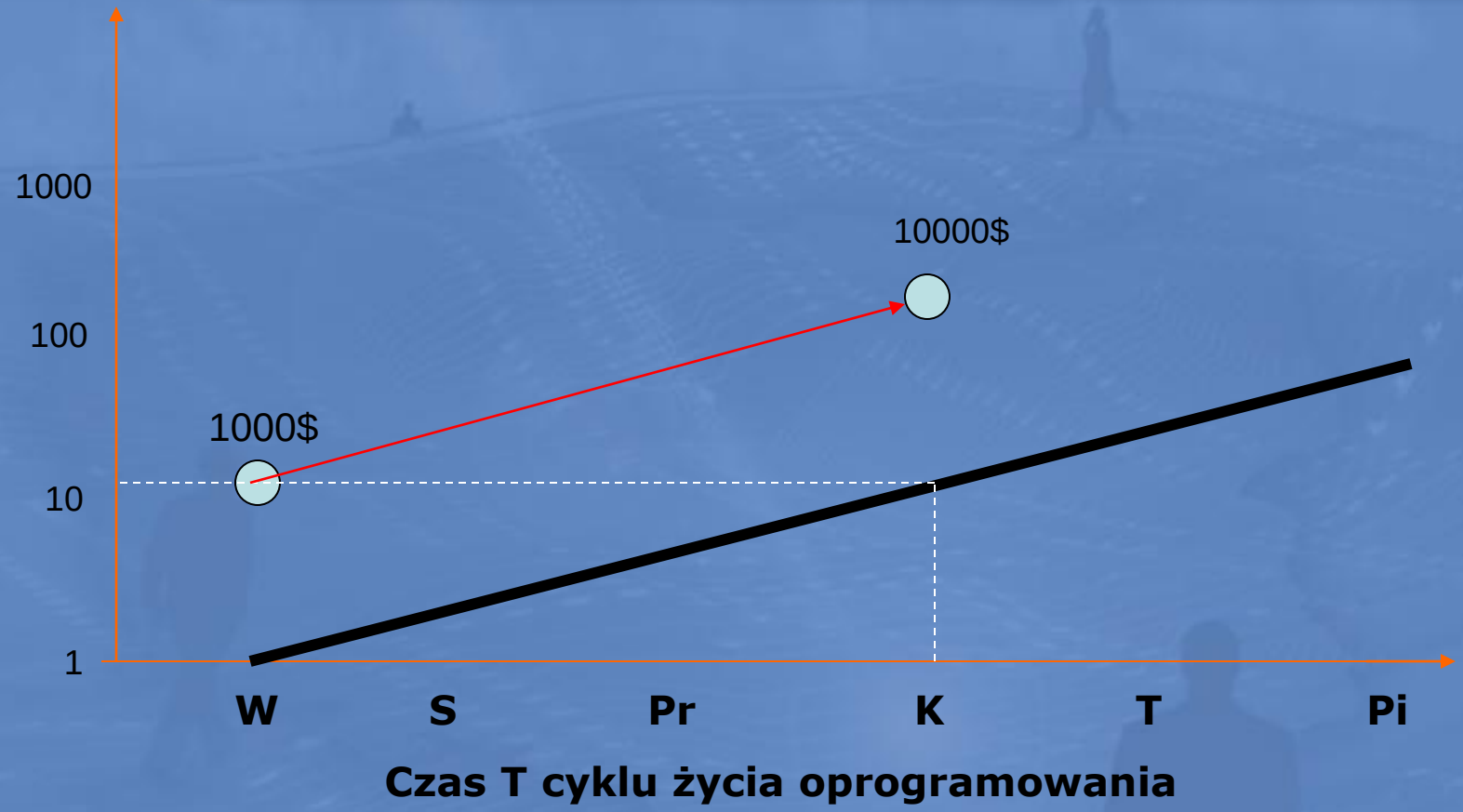
„zwiększenie personelu w opóźnionym przedsięwzięciu informatycznym powoduje zwiększenie opóźnienia”

INŻYNIERIA OPROGRAMOWANIA

Kaskadowy model życia oprogramowania

INŻYNIERIA OPROGRAMOWANIA

Względny koszt C poprawy problemu



„logarytmiczna zależność kosztu usunięcia błędu od etapu jego wykrycia”

INŻYNIERIA OPROGRAMOWANIA

Pytania?

1. Czy czas wprowadzenia oprogramowania jest istotny?
2. Wyjaśnij różnicę pomiędzy niezawodnością a pewnością?
3. Jak to może być, że program jest łatwy w pielęgnacji ale ma małą funkcjonalność?
4. Jeśli inżynier błędnie wymawia „kamień milowy” (milestone) jako „kamień młyński” (millstone) o czym może on nieświadomie myśleć?
5. Opisz etapy kaskadowego życia oprogramowania
6. Czy plan testów może stanowić jeden z elementów kaskady?
7. Porównaj menadżerskie i inżynierskie spojrzenie na tworzenie oprogramowania.