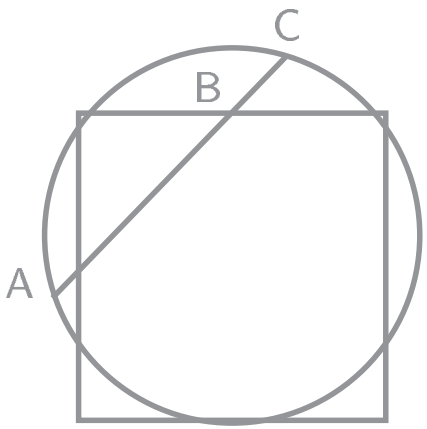
Inżynieria oprogramowania

Radosław Klimek

2015-23



http://home.agh.edu.pl/rklimek

1 Wprowadzenie do inżynierii oprogramowania

Wprowadzenie do inżynierii oprogramowania

Leonardo da VINCI: *Autoportret*

Inżynieria i podejście inżynierskie

Definicja

Inżynieria – nauka wykonywania robót inżynierskich, umiejętność projektowania, konstruowania i wznoszenia obiektów, urządzeń technicznych i różnych rodzajów budowli <fr.>
(*Słownik języka polskiego* PWN). ⌟

Inżynieria i podejście inżynierskie

Definicja

Inżynieria – nauka wykonywania robót inżynierskich, umiejętność projektowania, konstruowania i wznoszenia obiektów, urządzeń technicznych i różnych rodzajów budowli <fr.>
(*Słownik języka polskiego* PWN). ⌟

- Początkowo termin inżynieria odnosił się tylko do obiektów takich jak budowle i fortyfikacje, później wszelkie urządzenia mechaniczne i inne.
- Dziś – ogólnie – inżynieria to praktyczne stosowanie praw i zasad pewnej dziedziny. Inżynier to osoba ukierunkowana praktycznie.

Inżynieria oprogramowania – formułowanie potrzeby

Pojęcie **inżynierii oprogramowania** (ang. *software engineering*) zostało wprowadzone przez Fritza Bauera w 1968 r. na konferencji organizowanej w Garmish (Niemcy) przez NATO.

Inżynieria oprogramowania – formułowanie potrzeby

Pojęcie **inżynierii oprogramowania** (ang. *software engineering*) zostało wprowadzone przez Fritza Bauera w 1968 r. na konferencji organizowanej w Garmish (Niemcy) przez NATO.

F. Bauer

Friedrich Ludwig "Fritz" Bauer (1924–2015) – niemiecki matematyk, pionier informatyki i profesor Politechniki Monachijskiej. Żelazny Krzyż w 1944.

Bauer był członkiem niemieckiej misji Komitetu Naukowego NATO.
W 1967 NATO dyskutowało na temat "kryzysu oprogramowania", a Bauer zasugerował termin "Inżynieria oprogramowania" jako sposób na zrozumienie zarówno problemu, jak i rozwiązania.

Inżynieria oprogramowania – formułowanie potrzeby (cd.)

Taka opinia inna

"Software engineering is the part of computer science which is too difficult for the computer scientist" – Friedrich Bauer (1971): Proceedings of the IFIP Congress 1971, Ljubljana, Yugoslavia, August 23-28, 1971.

Stos

Stos został wymyślony i opracowany przez niemieckiego naukowca informatyka Friedricha Bauera w 1955 roku, a opatentowany w 1957. Za ten wynalazek Friedrich Bauer dostał w 1988 roku od IEEE nagrodę Computer Society Pioneer Award. Liczne inne osiągnięcia.

Inżynieria oprogramowania – krótka geneza

- W latach 60. natrafiono na problemy przy tworzeniu dużych systemów, głównie systemów operacyjnych oraz kompilatorów.
- Przed problemami na jakie natrafiono w latach 60. oprogramowanie było wytwarzane zazwyczaj przez użytkowników na swoje własne potrzeby.
- Jeden z pierwszych dużych systemów był system operacyjny OS 360 dla komputerów IBM 360 – jego konstruowanie unaoczniło, że tworzenie takich dużych systemów nie może odbywać się poprzez proste przeniesienie technik charakterystycznych dla małych systemów. Tworzenie oprogramowania nie może być zależne od wyjątkowej zdolności i wiedzy pojedynczych osób.
- Ze względu na rozmiary projektu i występujące szczegóły niezbędne jest wypracowanie technik komunikacji i zarządzania wykonawstwem prac.

Przyczyny „kryzysu oprogramowania”

Z gwałtownym rozwojem technik i technologii komputerowych związany jest tzw. „kryzys oprogramowania”, dotyczący przede wszystkim czasu i kosztów realizacji systemów, który ma kilka swoich źródeł:


duża i coraz większa złożoność systemów informatycznych;


niepowtarzalność kolejnych przedsięwzięć informatycznych;


mała przejrzystość procesu tworzenia oprogramowania;


pozorna łatwość modyfikowania oprogramowania („kładzenia kodu”).

Przyczyny „kryzysu oprogramowania" (cd.)

- Mała przejrzystość procesu tworzenia oprogramowania – np. trudności w ocenie stopnia zaawansowania przesięwzięcia (o czym świadczyć może pytanie uczestników o stopień zaawansowania prac).
- Pozorna łatwość „kładzenia kodu" – to nie jest tak, że jeżeli w jeden dzień stworzymy 100 wierszy kodu, to w 10 dni stworzymy 1000 wierszy.
- Chyba można powiedzieć, że tzw. „kryzys oprogramowania" trwa do dzisiaj – pojawiają się nowe potrzeby, które natrafiają na nowe trudności. Pewnym przykładem może tu być np. ciągłe odkładanie wprowadzenia na rynek kolejnych wersji systemów.

Postulaty odnośnie nowego podejścia do tworzenia oprogramowania

Na wspomnianej konferencji naukowej w 1968 r. – a także konferencjach następnych – postulowano wiele zmian w podejściu do tworzenia oprogramowania:

- wprowadzenie lepszych technik zarządzania projektem;
- inna organizacja zespołów projektowych;
- wprowadzenie nowych generacji języków i narzędzi;
- standaryzację notacji i kodowania.

W przypadku większości konferencji wnioskiem końcowym było stwierdzenie, że do tworzenia oprogramowania należy pochodzić jak do budowania złożonych obiektów przemysłowych.

Cechy podejścia inżynierskiego

Podejście inżynierskie wymaga, także w odniesieniu do tworzenia oprogramowania, uwzględnienia:

- odpowiedniego zarządzania;
- właściwej organizacji;
- dostosowanych narzędzi;
- adaptacji teorii;
- systematycznego stosowania metodologii i technik.

Wszystkie te postulaty stały u początku narodzin inżynierii oprogramowania.

Specyfika procesu programowania (1)

Programowanie może sprawiać dużą satysfakcję, przyczyny (F.Brooks: *Mityczny osobomiesiąc. Eseje o inżynierii oprogramowania*. Wydawnictwa Naukowo-Techniczne 2000) czego można zreasumować następująco:

pasja tworzenia czegoś nowego, a także podejmowania nowych wyzwań;
robienie czegoś przydatnego dla innych ludzi;
fascynacja tworzeniem złożonych i niebanalnych systemów, gdzie poszczególne elementy zazębiają się wzajemnie;
nieustanne uczenie się i poznawanie nowych rzeczy – każdy system jest w istocie niepowtarzalny;
praca nad „łatwym" do obróbki materiałem, będącym praktycznie tylko wytworem myślenia i generowania czy też modyfikowania koncepcji myślowych.

Programowanie to trochę jak tworzenie poezji.

Specyfika procesu programowania (2)

Programowanie sprawia także trudności, które można zreasumować następująco:

ludzie nie są perfekcyjni, podczas gdy komputery są „perfekcyjne” – jakikolwiek drobny/techniczny błąd może mieć daleko idące reperkusje;
w praktyce programiści są często zależni od innych, nie podejmując samemu np. decyzji o zasobach i celach;
konieczność drobiazgowej i żmudnej pracy nad wyszukiwaniem błędów i pułapek w systemie;
poprawianie błędów nie musi być funkcją liniową – znalezienie i usunięcie ostatnich błędów zajmuje dużo więcej zasobów niż w przypadku błędów pierwszych;
często produkt nad którym się tak długo pracowało okazuje się przestarzały w momencie jego oddania, albo też wkrótce po jego wdrożeniu.

Inżynieria oprogramowania – definicja

Definicja (F. Bauer, 1972)

Inżynieria oprogramowania, to ustanawianie i stosowanie solidnych zasad inżynierii w celu uzyskania w sposób ekonomiczny oprogramowania, które jest niezawodne i działa wydajnie na rzeczywistej maszynie. ⌟

Inżynieria oprogramowania – definicja

Definicja (F. Bauer, 1972)

Inżynieria oprogramowania, to ustanawianie i stosowanie solidnych zasad inżynierii w celu uzyskania w sposób ekonomiczny oprogramowania, które jest niezawodne i działa wydajnie na rzeczywistej maszynie. ⌟

- Z czasem definicja inżynierii oprogramowania ulegała uszczegółowieniu, obejmując poszczególne fazy rozwoju oprogramowania, a jako zbiór minimalny: tworzenie, ocenę i konserwację.
- Inżynierię oprogramowania określano też jako dziedzinę obejmującą wiedzę o planowaniu, zarządzaniu i organizacji przedsięwzięć informatycznych.

Inżynieria oprogramowania – inna definicja

Definicja

Inżynieria oprogramowania, to wiedza techniczna, dotycząca wszystkich faz cyklu życia/wytwarzania oprogramowania, której celem jest uzyskanie oprogramowania, jako wysokiej jakości produktu. ⌟

Zakłada się przy tym, że produkt ten, pomimo swojej specyfiki, powienien być:

- zgodny z wymaganiami klienta/użytkownika,
- niezawodny,
- efektywny,
- łatwy w konserwacji/pielęgnacji,
- ergonomiczny.

Specyfika inżynierii oprogramowania

Inżynieria oprogramowania obejmuje m.in. następujące elementy:

- techniki reprezentacji (informatyzowanego fragmentu) rzeczywistości;
- metody analizy danych i procesów;
- metody projektowania szczegółowego;
- dokumentowanie produktu;
- sposoby oceny jakości, a także inne zagadnienia.

Specyfika inżynierii oprogramowania (cd.)

Inżynieria oprogramowania odróżnia się od innych dziedzin:

- produkt/projekt jest wirtualny i został opracowany na podstawie konkretnej specyfikacji;
- zbiór dziedziny zastosowań jest zazwyczaj nieskończony w przeciwieństwie do innych dziedzin;
- projektowanie każdego wyrobu jest zazwyczaj intensywne, rzadko występuje produkcja masowa;
- standaryzacja (o ile istnieje) dotyczy procesu wytwarzania, a nie samego wyrobu.

Specyfika inżynierii oprogramowania – przykład

Inżynieria oprogramowania ma wiele podobieństw do innych rodzajów inżynierii – projektowanie przedmiotów i obiektów – ale ma swoją wyraźną specyfikę, co czyni ją zupełnie niepodobną do innych inżynierii.

Przykładowo modyfikowanie samolotu jako produktu inżynierskiego nie jest łatwe i wymaga przemyślanych zmian oraz poprawek. Modyfikowanie oprogramownia może być dokonywane niejako „od ręki”, ze względu na swoją łatwość wprowadzania takich zmian. W praktycze sytuacja ta jest bardziej skomplikowana i ma swoje reperkusje.

Zarządzanie projektem informatycznym

Przykładowe zagadnienia wchodzące w zakres profesjonalnego zarządzania projektem informatycznym, a więc także wchodzące w obszar zainteresowania nowoczesnej inżynierii oprogramowania:

- metodologie i strategie prowadzenia przedsięwzięć informatycznych, strategie;
- cykle rozwoju/życia oprogramowania;
- inżynieria wymagań systemów informatycznych, pozyskiwanie wymagań;
- szacowanie parametrów i nakładów w projekcie informatycznym, metryki oprogramowania, wymiarowanie oprogramowania;
- inspekcje i rewizje oprogramowania, zapewnienie jakości;
- zarządzanie zmianami, wersjami i konfiguracjami oprogramowania;
- analiza i zarządzanie ryzykiem projektu informatycznego;
- metody i strategie testowania, weryfikacji i walidacji oprogramowania;
- studium wykonalności projektu informatycznego;
- inne zagadnienia (np. zarządzanie zasobami ludzkimi, metody komunikacji w zespole informatycznym, inne).

Inżynieria oprogramowania (IEEE)

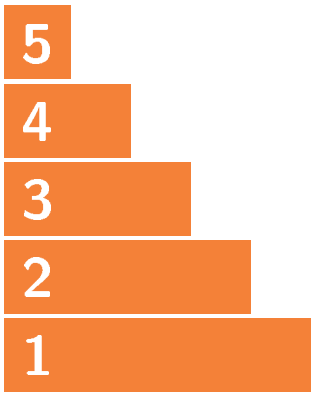
Definicja (IEEE)

Inżynieria oprogramowania, to stosowanie systematycznego, zdyscyplinowanego i wymiernego podejścia do tworzenia, użytkowania i konserwacji oprogramowania, tj. stosowanie zasad inżynierii w odniesieniu do oprogramowania oraz badanie inżynierskiego podejścia. ⌟

Obecnie nie wystarczy już zapewnienie że produkt informatyczny działa, dąży się do wykazania, że produkt spełnia wymogi jakościowe. Jakość musi być wbudowana w cały proces twórczy – dbałość o użytkowanika jest zaleceniem podstawowym, a opanowanie technik informatycznych może nawet schodzić na plan drugi.

Model CMM

Ważna rola **modelu CMM** (ang. **Capability Maturity Model**), tj. **modelu dojrzałości organizacyjnej** (firmy software'owej), zaproponowanego w 1991 r. w renomowanym **Software Engineering Institute** w Carnagie Mellon University, a zdefiniowanego poprzez pięć tzw. **poziomów**:



poziom optymalizacji
poziom zarządzania
poziom definiowania
poziom powtarzalności
poziom początkowy

Dwa nurty inżynierii oprogramowania

Można wyróżnić dwa zasadnicze nurty inżynierii oprogramowania:

1. **nurt formalny** – postulowanie stosowania metod formalnych: formalne języki specyfikacji, formalnych transformacji, ścisłe dowody poprawności;
2. **nurt praktyczny/nieformalny** – notacja nie w pełni sformalizowana, stosowanie metod graficznych, duże znaczenie ludzkiego doświadczenia i wiedzy.

Możliwe jest także łączenie obu nurtów.

Inżynieria oprogramowania odnotowuje znaczące postępy poprzez

- rozwój swoich metodyk,
- rozwój narzędzi, oraz
- edukację.

Dwa spojrzenia na rozwój inżynierii oprogramowania

Istnieją także dwa skraje punkty widzenia odnośnie wzajemnej relacji inżynierii oprogramowania oraz tradycyjnych technik programistycznych:

1. podejście tradycyjne – pojawienie się inżynierii oprogramowania, to naturalny etap rozwoju technik programowania, to kontynuacja tego co było wcześniej, jako naturalny skutek potrzeby większej abstrakcji niż tylko wynikający z poziomu danego języka programowania;
2. podejście rewolucyjne – inżyniera oprogramowania, to całkowice przeciwieństwo tego co było dotychczas, tradycyjne środki i metody okazały się niewystarczające, gdyż praca nad oprogramowaniem wymaga m.in. modelowania systemu i myślenia w kategoriach funkcjonalności i użyteczności, a nie w kategoriach kodu.

Odnośnie podejścia rewolucyjnego, to puryści informatyczni uważają, że kształcenie informatyka powinno się zaczynać od inżynierii oprogramowania, a dopiero później nauki konkretnych języków czy środowisk programowania.

Zagadnienia nowoczesnej inżynierii oprogramowania

Obecnie inżynieria oprogramowania obejmuje szereg zagadnień, takich m.in. jak:

- sposoby prowadzenia przedsięwzieć informatycznych;
- sposoby planowania, szacowania kosztów, harmonogramowania czy monitorowania przedsięwzięć informatycznych, oraz problem komunikacji wewnętrznej i zewnętrznej;
- metody analizy i projektowania systemów, problem odmiennych metodologii i standardów, oraz problem zmiennych i zmieniających się wymagań użytkownika;
- sposoby zwiększania niezawodności oprogramowania;
- sposoby testowania systemów i oceny niezawodności;
- sposoby przygotowywania dokumentacji technicznej i użytkowej;
- procedury kontroli jakości;
- metody konserwacji oprogramowania;
- techniki i rodzaje pracy zespołowej;
- narzędzia wspomagające typu CASE.

Znaczenie inżyniera oprogramowania

Znaczenie dobrze przygotowanego inżyniera oprogramowania jest trudne do przecenienia.

Oprócz takich cech jak doświadczenie, komunikatywność, znajomość algorytmów, struktur danych, języków programowania (przynajmniej jeden), powinien także:

- znać jednocześnie kilka metodyk projektowania lub przynajmniej zdawać sobie sprawę z różnic, jakie w nich występują;
- umieć „przekładać" niejasne żądania klientów i przyszłych użytkowników na precyzyjne specyfikacje;
- umieć rozmawiać z zamawiającymi oprogramowanie lub przyszłymi użytkownikami oprogramowania, posługując się terminami z dziedziny aplikacji, a nie terminami informatycznymi.

Obszary wiedzy inżyniera oprogramowania

inżynieria oprogramowania

informatyka teoretyczna

środowisko oprogramowania
(język programow., system operacyjny, środowisko projekt.)

organizacja instytucji
(funkcjonowanie admin., funkcjonowanie przedsięb.)

dziedzina aplikacji
(np. bankowość, gospod. materiałowa)

psychologia i socjologia
(m.in. etyka, komunikacja, harmonogramowanie, podejmowanie decyzji)

Inżynieria systemów

Inżynieria systemów powstała w latach 40-tych w Bell Telephone Laboratories (USA). Celem jej było sformułowanie ogólnych zasad postępowania podczas realizacji dużych przedsięwzięć technicznych.

Zadania inżynierii systemów w zakresie organizacji procesu wytwórczego:

- dostarczenie kierownictwu informacji o stanie projektu i dostrzeżonych problemach;
- formułowanie planów i zadań przypisanych do poszczególnych etapów;
- harmonizowanie działań i dążenie do optymalnego wykorzystania zasobów, w tym personelu;
- opracowanie projektów szczegółowych odpowiednio do planów;
- zapewnienie zasad i metod rozwiązań;
- wykonanie każdej czynności w zakresie inżynierii systemów z użyciem określonej techniki.

Podział zadań w inżynierii systemów

Tworzenie każdego obiektu w inżynierii systemów związane jest z podziałem na pięć faz:

1. **badania wstępne** – badanie potrzeb i otoczenia, opracowanie planu przedsięwzięcia z założeniem okresowej kontroli każdej fazy;
2. **projekt wstępny (pierwsza faza projektowania)** – sformułowanie problemu, wybór zadań, synteza (tworzenie wariantów rozwiązań), badania eksperymentalne i analiza wariantów, wybór sposobu rozwiązania;
3. **plan realizacji systemu (druga faza projektowania)** – dotyczy wybranego projektu;
4. **pierwsza faza realizacji** – projekt końcowy systemu, badania w trakcie realizacji (próby, prototypy), uwzględnienie rzeczywistych warunków pracy;
5. **druga faza realizacji** – trwa w okresie eksploatacji systemu i obejmuje kontakty ze zleceniodawcą, modyfikacje, okresowe testowanie.

Idealny inżynier wg inżynierii systemów

Cechy idealnego inżyniera wg inżynierii systemów:

- systemowy punkt widzenia (koncentrowanie się na koncepcji całości systemu);
- zdolność do rozsądnej i obiektywnej oceny potrzeb, projektu i wyrobów;
- podporządkowanie syntezy rozwiązań analizie wymagań;
- twórcze zdolności i bogata wyobraźnia (poszukiwanie nowatorskich rozwiązań);
- zręczność w kontaktach międzyludzkich oraz zdolności kierownicze i dyplomatyczne przydatne w rozwiązywaniu nieuniknionych konfliktów;
- dar przekonywania (pośrednictwo w przekazywaniu informacji współautorom projektu);
- umiejętność dokształcania się nie tylko w zakresie techniki, ale także podstaw teoretycznych, w tym matematycznych, ekonomii, filozofii, psychologii, języka ojczystego oraz języków obcych;
- umiejętność korzystania z doświadczeń własnych i cudzych.

Cechy nowoczesnego systemu informatycznego

Dobrze zaprojektowany system informatyczny powinien posiadać wiele cech, dbałość o które powinna pojawiać się już od początku prac nad systemem (stworzenie właściwego modelu fizycznego i logicznego leży u podstaw takiego systemu, możliwe jest także swoiste „optymalizowanie" niektórych cech projektowanego systemu):

- **zgodność funkcjonalna ze specyfikacją wymagań** (ang. **functional requirements specification fulfillment**) – system musi realizować zdefiniowane zadania (sic!);
- **wydajność** (ang. **performance**) – konieczność osiągania pewnych parametrów systemu, np. czas reakcji;
- **koszty projektu** – wydatki faktycznie poniesione a budżet przenaczony na budowę (i pielęgnację) systemu;
- **czas trwania projektu** – terminowość zakończenia prac, a także dotrzymanie harmonogramu prac;
- **bezpieczeństwo** (ang. **safety**) – błąd w systemie nie może powodować jego zatrzymania (zagrożenie ciągłości pracy);

Cechy nowoczesnego systemu informatycznego (cd.)

- **przyjazność** (ang. **user-friendliness**) – łatwość posługiwania się systemem przez osoby nie będące specjalistami;
- **niezawodność** (ang. **realiability**) – średni czas pracy pomiędzy awariami;
- **pielęgnowalność** (ang. **maintainability**) – może to być średni czas potrzebny na usunięcie błędu w systemie, niezbędnego do przywrócenia stanu używalności;
- **efektywność** (ang. **efficiency**) – efektywne wykorzystywanie dostępnych zasobów;
- **modyfikowalność** (ang. **modifability**) – łatwe przystosowanie systemu do zmieniającego się otoczenia, względnie nowych wymagań użytkownika;
- **elastyczność** (ang. **flexibility**) – możliwość łatwego wykonania zadań odbiegających od zadań typowych.