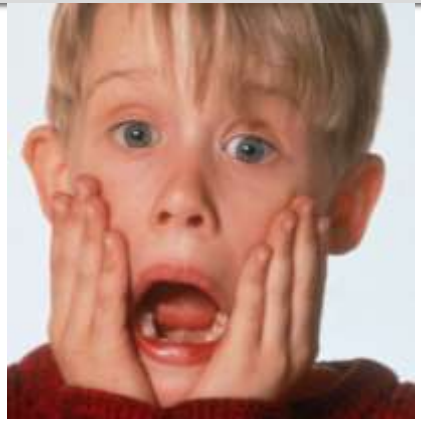
Podstawy Inżynierii Oprogramowania

- Tomasz Kowalski, PhD
- email: tomasz.kowalski@p.lodz.pl
- Instytut Informatyki Stosowanej (www.iis.p.lodz.pl)
- bieżące materiały są na wikamp
  - archiwalne slajdy, instrukcje:
    - www: radamus.kis.p.lodz.pl,
    - username: pio
    - password: student

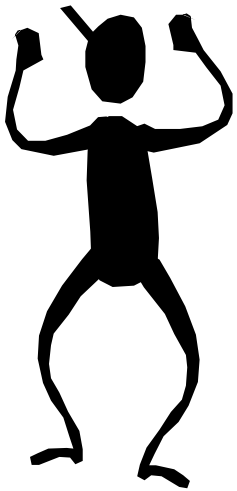
Bibliografia

- Ian Sommerville „Software Engineering”, 9th edition, Pearson Education, 2011
- Robert C. Martin,
  - Czysty kod. Podręcznik dobrego programisty, Helion 2008
  - Mistrz czystego kodu. Kodeks postępowania profesjonalnych programistów, Helion 2011
- Robert L. Glass, Facts and Fallacies of Software Engineering, Pearson Education, 2002
- Blogi techniczne, screencasty, publikacje, doświadczenia osobiste …

Trzej przyjaciele ... ze studiów



Andrzej



Krzysiek

Maciek

Zadanie z … programowania (proceduralnego? obiektowego?)

- **Napisz program, który wczyta 100 liczb naturalnych, posortuje je w porządku rosnącym, wyświetli listę tak posortowanych liczb wraz z obliczoną wartością średnią.**

- Nasi studenci nie mieli większych problemów z rozwiązaniem powyższego zadania.

Andrzej i Krzysiek zaczęli od napisania algorytmu na kartce papieru…

…Maciek od razu podszedł do komputera i zaczął kodowanie…

Andrzej ...

- ... pracuje w dużej firmie przemysłowej ...

- Pewnego dnia Andrzej oraz pozostali pracownicy wydziału inżynierii oprogramowania, na specjalnie zwołanym zebraniu, otrzymali, od wiceprezesa do spraw sprzedaży i marketingu następujące zadanie:

> *Opracujcie zautomatyzowany system, który pozwoli nam przetwarzać zamówienia przynajmniej 24 godziny szybciej niż obecna średnia oraz dostarczać nasze produkty przynajmniej trzy dni wcześniej niż obecnie.*

Krzysiek …

- … znalazł zatrudnienie w firmie zajmującej się budowaniem systemów informatycznych dla potrzeb lotnictwa …

- Jego zespół otrzymał następujące zadanie:

> *Opracujcie oprogramowanie, dla nowego samolotu pasażerskiego, które pozwoli mu lądować na większości lotnisk bez pomocy pilota.*

Maciek ...

- ... rozpoczął pracę dla firmy specjalizującej się w tworzeniu oprogramowania dla komputerów osobistych, oraz urządzeń mobilnych ...

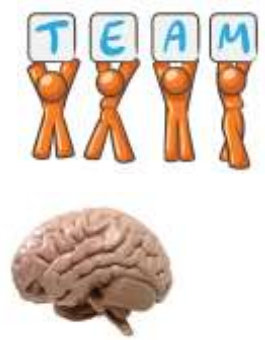
- Szef firmy wezwał wszystkich nowych inżynierów oprogramowania i zlecił im następujące zadanie:

> *Opracujcie oprogramowanie, które sprzeda się przynajmniej w milionie kopii i którego detaliczna cena będzie wynosiła przynajmniej 200$.*

Opracujcie zautomatyzowany system, który pozwoli nam przetwarzać zamówienia przynajmniej 24 godziny szybciej niż obecna średnia oraz dostarczać nasz produkty przynajmniej trzy dni wcześniej niż obecnie

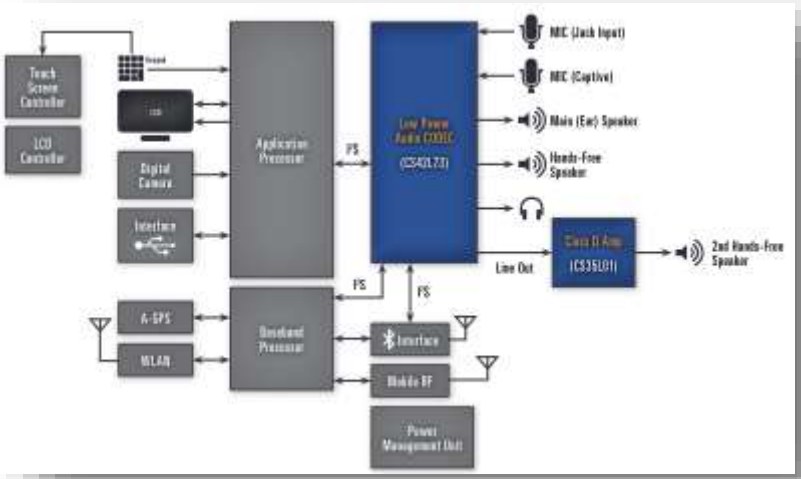
Opracujcie oprogramowanie, dla nowego samolotu pasażerskiego Z-686, które pozwoli mu lądować na większości lotnisk bez pomocy pilota.

Opracujcie oprogramowanie, które sprzeda się przynajmniej w milionie kopii i którego detaliczna cena będzie wynosiła przynajmniej 200$.



Wyobraźmy sobie...

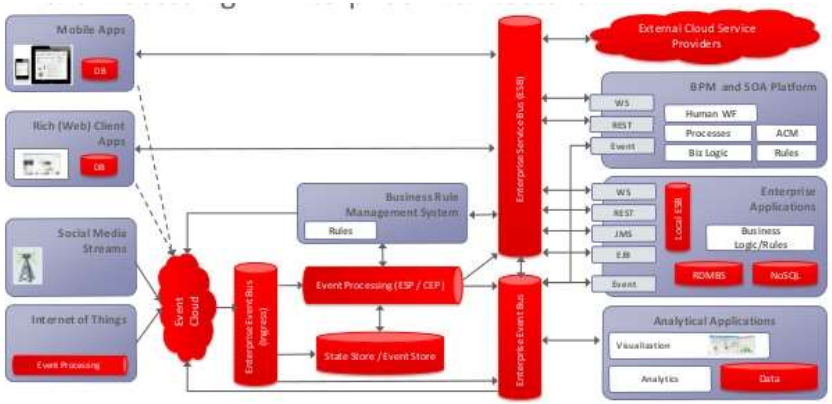
- Produkt



Wyobraźmy sobie...

- Oprogramowanie jako produkt

```
package org.apache.spark.streaming

import java.util.concurrent.TimeUnit
import java.util.concurrent.locks.ReentrantLock

private[streaming] class ContextWaiter {

  private val lock = new ReentrantLock()
  private val condition = lock.newCondition()

  // Guarded by "lock"
  private var error: Throwable = null

  // Guarded by "lock"
  private var stopped: Boolean = false

  def notifyError(e: Throwable): Unit = {
    lock.lock()
    try {
      error = e
      condition.signalAll()
    } finally {
      lock.unlock()
    }
  }

  def notifyStop(): Unit = {
    lock.lock()
    try {
      stopped = true
      condition.signalAll()
    } finally {
      lock.unlock()
```



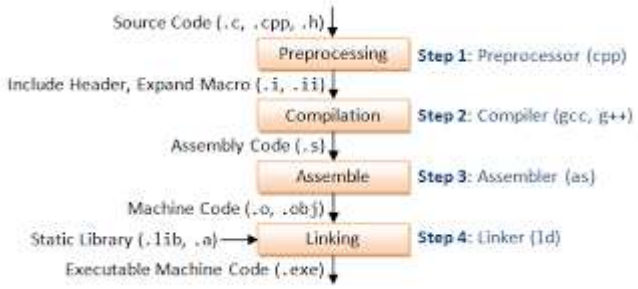
Wyobraźmy sobie...

- wytwarzanie jako proces

Wyobraźmy sobie...

- wytwarzanie oprogramowania jako proces

```
# Not intended for manual invocation.
# Invoked if automatic builds are enabled.
# Analyzes only on those sources that have changed.
# Does not build executables.
autobuild:
   $(GNATMAKE) -gnatc -c -k  -P "$(GPRPATH)"

# Clean the root project of all build products.
clean:
   $(GNATCLEAN) -P "$(GPRPATH)"

# Clean root project and all imported projects too.
clean_tree:
   $(GNATCLEAN) -P "$(GPRPATH)" -r

# Check *all* sources for errors, even those not changed.
# Does not build executables.
analyze:
   $(GNATMAKE) -f  -gnatc -c -k  -P "$(GPRPATH)"

# Build executables for all mains defined by the project.
build:
   $(GNATMAKE) -P "$(GPRPATH)"

# Clean, then build executables for all mains defined by the project.
rebuild: clean build
```



Czy można produkować oprogramowanie?

- Oprogramowanie
  - Jest nieuchwytne
  - Jego wytwarzanie jest procesem
    - twórczym (jak projektowanie mostu)
    - wytwórczym (jak budowa mostu)
    - iteracyjnym (jak „dobudowywanie" mostu)
- Oprogramowania się nie produkuje
- Oprogramowanie się rozwija (ang. develop)

Inżynieria Oprogramowania

Wprowadzenie

Inżynieria Oprogramowania

- Ekonomia wszystkich krajów rozwijających się opiera się na wykorzystaniu oprogramowania.
- Coraz większa liczba systemów jest kontrolowana za pomocą oprogramowania.
- Systemy oprogramowania są abstrakcyjne i nieuchwytne.
- Inżynieria Oprogramowania związana jest z teoriami, metodami i narzędziami do profesjonalnego wytwarzania oprogramowania.

Eighty percent of software work is intellectual. A fair amount of it is creative. Little of it is clerical.

Robert L. Glass, Facts and Fallaces of Software Engineering

Koszty oprogramowania

- Koszty oprogramowania są często dominujące – przewyższają koszty sprzętu.
- Koszty wsparcia oprogramowania są wyższe niż koszty jego produkcji
  - Dla systemów długoterminowych mogą być wielokrotnie wyższe.
- Efektywne wytwarzania oprogramowania, jest istotą Inżynierii Oprogramowania.

The most important factor in software work is not the tools and techniques used by the programmers, but rather the quality of the programmers themselves.

Robert L. Glass, Facts and Fallaces of Software Engineering

Oprogramowanie jako produkt

- **Oprogramowanie generyczne/ogólne** (ang. generic)
  - Odrębne systemy wytwarzane i sprzedawane szerokiemu zakresowi klientów.
  - Przykłady – Programy graficzne, oprogramowanie wspierające zarządzanie projektami, systemy CAD, systemy specyficzne dla określonej dziedziny np. system obsługi apteki.
- **Oprogramowanie dopasowane** (ang. custom)
  - wyprodukowane dla konkretnego klienta zgodnie z dostarczoną specyfikacją
  - Przykłady – systemy wbudowane, systemy kontroli ruchu lotniczego, systemy monitoringu i sterowania ruchem ulicznym.

Specyfikacja produktu

- Oprogramowanie generyczne
  - Specyfikacja tego, jak oprogramowanie powinno działać, jest własnością **wykonawcy**,
  - decyzje związane ze zmianami podejmuje **wykonawca**.
- Oprogramowanie dopasowane
  - Specyfikacja tego, jak oprogramowanie powinno działać, jest własnością **klienta**,
  - decyzje związane ze zmianami podejmuje **klient**.

Model dostarczania

1. Instalacja po stronie klienta
   - System uruchomiony na infrastrukturze klienta (własnej lub dzierżawionej)
   - Klient -> właściciel
2. Udostępnianie zdalne
   - System uruchomiony na infrastrukturze dostawcy
   - Oprogramowanie jako usługa (ang. Software as a Service SaaS)
   - Dostawca -> właściciel, Klient -> Dzierżawca

Projekty software'owe

- Oprogramowanie różni się od innych produktów
  - Koszty koncentrują się na rozwoju
  - Konserwacja to poprawianie błędów i udoskonalanie oraz dodawanie nowych funkcjonalności
  - Postęp projektu trudno jest mierzyć

Często zadawane pytania...

- Co to znaczy dobre oprogramowanie?
- Co to jest Inżynieria Oprogramowania?
- Jakie czynności stanowią fundament pracy inżynierów oprogramowania?
- Jaka jest relacja pomiędzy Inżynierią Oprogramowania a informatyką czy inżynierią systemów?
- Co należy do podstawowych wyzwań współczesnej Inżynierii Oprogramowania?
- Jakie są koszty produkcji oprogramowania?
- Jakie metody i techniki są stosowane w Inżynierii Oprogramowania?
- Jak na Inżynierię Oprogramowania wpłynął rozwój technologii Internetowych?
- ...

Atrybuty dobrego oprogramowania

- Oprogramowanie powinno dostarczać użytkownikom wymaganą funkcjonalność i wydajność.
- Oprogramowanie powinna cechować:
  1. **Zarządzalność**: musi ewoluować w odpowiedzi na zmieniające się wymagania
  2. **Wiarygodność i bezpieczeństwo**: można na nim polegać
  3. **Sprawność**: nie powinno marnować zasobów systemu
  4. **Użyteczność**: musi być użyteczne dla użytkowników, dla których zostało zaprojektowane

Jakość to zbiór atrybutów!!

Inżynieria Oprogramowania

- Inżynieria Oprogramowania to **dyscyplina inżynieryjna** dotycząca **wszystkich aspektów związanych z wytwarzaniem** oprogramowania
  - poczynając od wstępnych zarysów specyfikacji
  - kończąc na wspieraniu systemu w czasie jego działania.
- Dyscyplina inżynieryjna
  - Wykorzystanie adekwatnych teorii i metod do rozwiązania problemów z uwzględnieniem ograniczeń organizacyjnych i finansowych.
- Wszystkie aspekty związane z wytwarzaniem
  - Nie tylko techniczny proces produkcji ale również zarządzanie projektem oraz rozwój narzędzi, metod itp. Wspierających rozwój oprogramowania.

Znaczenie inżynierii oprogramowania

- Coraz więcej jednostek i całych społeczeństw polega na rozwoju systemów oprogramowania. Musimy być zdolni do **szybkiego** i **ekonomicznego** wytwarzania **wiarygodnych** i **niezawodnych** systemów.

- W szerokim horyzoncie czasowym, wykorzystanie **metod i technik Inżynierii Oprogramowania** (projekt zespołowy)**,** w miejsce po prostu **pisania programów** (projekt indywidualny), jest po prostu bardziej opłacalne.
  - Dla większości typów systemów najwyższe koszty umiejscowione są w obszarze wprowadzania zmian w działającym oprogramowaniu.

Czynności procesu Inżynierii Oprogramowania

- Specyfikacja oprogramowania
  - Klient i inżynierowie definiują oprogramowanie, które ma zostać wyprodukowane oraz określają ograniczenia przy których ma realizować swoje zadania.
- Rozwój (produkcja) oprogramowania
  - Oprogramowanie jest projektowane i implementowane.
- Zatwierdzanie (walidacja) oprogramowania
  - oprogramowanie jest sprawdzane pod kątem spełniania wymagań klienta.
- Ewolucja oprogramowania
  - Oprogramowanie jest modyfikowane w celu uwzględnienia zmian w wymaganiach klienta oraz środowiska (np. rynkowego, prawnego, sprzętowego, ...).

Ogólne problemy mające wpływ na oprogramowanie

- Heterogeniczność
  - Coraz częściej systemy muszą działać w środowiskach rozproszonych, które obejmują różne rodzaje systemów komputerowych i urządzeń mobilnych.
- Zmiany środowiska biznesowego i społecznego
  - Biznes i społeczeństwo podlegają szybkim zmianom wymuszanym przez rozwój gospodarczy i dostępność nowych technologii. Powoduje to potrzebę wprowadzania zmian w istniejącym oraz szybką produkcję nowego oprogramowania.
- Bezpieczeństwo i zaufanie
  - Ponieważ oprogramowanie splata się z niemal wszystkimi aspektami naszego życia bardzo istotne jest to czy możemy mu ufać.

Inżynieria oprogramowania a inne dyscypliny

- **Informatyka** (ang. computer science) związana jest z podstawami teoretycznymi; inżynieria oprogramowania związana jest z praktyką budowania i dostarczania użytecznego oprogramowania.

- **Inżynieria systemów** (ang. system engineering) jest związana z wszelkimi aspektami budowo i ewolucji złożonych systemów
  - w których oprogramowanie może grać kluczową rolę

Niejednolitość Inżynierii Oprogramowania

- Istnieje wiele różnych typów oprogramowania
  - Nie istnieje zestaw uniwersalnych technik i narzędzi, który można by zastosować w produkcji.
- Metody i narzędzia wykorzystywane do produkcji oprogramowania będą się różnić w zależności od:
  - Typu rozwijanej aplikacji
  - Wymagań klienta
  - Umiejętności, wiedzy i doświadczenia zespołu projektowego.
  - Dostępności narzędzi

Typy aplikacji

- Aplikacje samodzielne (lokalne)
  - Aplikacje działające na lokalnym komputerze i posiadające wszystkie wymagane funkcjonalności bez potrzeby łączenia się z siecią.
- Interaktywne aplikacje transakcyjne
  - Aplikacje wykonywane na zdalnym komputerze, do których użytkownicy mają dostęp za pośrednictwem własnego komputera. (aplikacje webowe np. e-commerce).
- Wbudowane systemu kontrolne
  - Oprogramowanie kontrolujące i zarządzające działaniem urządzeń. W ujęciu ilościowym tego typu systemów jest prawdopodobnie więcej niż wszystkich innych typów razem wziętych.

Typy aplikacji

- Systemy przetwarzania wsadowego
  - Systemy biznesowe przeznaczone do przetwarzania dużych danych ilości danych (wsadów). Przetwarzają duży zestaw pojedynczych danych wejściowych w celu utworzenia odpowiadającego im rezultatu.
- Systemy rozrywkowe
  - Systemy, przede wszystkim do użytku osobistego, przeznaczone do rozrywki.
- Systemy do modelowania i symulacji
  - Systemy opracowywane przez naukowców i inżynierów dla celów modelowania fizycznych procesów lub sytuacji, które obejmują wiele odrębnych, ale będących ze sobą w interakcji, obiektów.

Typy aplikacji

- Systemy gromadzenia danych
  - Systemy zbierające, za pośrednictwem czujników, dane ze środowiska i przesyłające je do innych systemów dla celów dalszego przetwarzania.
- Systemy systemów
  - Systemy złożone z wielu innych systemów.

Fundamenty inżynierii oprogramowania

- Niezależnie od typu oprogramowania i techniki jego opracowania istnieje zbiór podstawowych zasad, które należy stosować:
  - System powinien być wytwarzany przy użyciu zarządzalnego i zrozumiałego procesu (inżynierii oprogramowania). Oczywiście do różnych typów systemów stosuje się różne procesy.
  - Niezawodność i wydajność są ważne.
  - Zrozumienie wymagań i zarządzanie specyfikacją oprogramowania są ważne.
  - Tam gdzie jest to właściwe, zamiast budowania nowego, powinno się powtórnie wykorzystywać oprogramowanie, które zostało już opracowane.

Inżynieria oprogramowania a Internet

- Internet (Web) jest obecnie platformą wykonawczą aplikacji. Organizacje coraz intensywniej angażują się w rozwój systemów bazujących na technologiach internetowych (Webowych) kosztem systemów lokalnych.
- Usługi Internetowe (ang. Web Services) pozwalają na dostęp do funkcjonalności za pośrednictwem Internetu.
- Chmury obliczeniowe (ang. Cloud computing) to podejście do dostarczania usług komputerowych w którym aplikacje wykonywane są zdalnie w „chmurze”.
  - Użytkownik nie kupuje oprogramowania tylko płaci za jego użycie.

Inżynieria systemów internetowych

- **Internet doprowadził do znaczącej zmiany w sposobie organizacji oprogramowania biznesowego.**
- **Dominującym podejściem w konstrukcji oprogramowania webowego jest wielokrotne użycie.**
  - Budując tego typu systemy skupiamy się na wymyśleniu w jaki sposób można złożyć system z istniejących komponentów oraz systemów.
- **Systemy webowe powinny być rozwijane i dostarczane przyrostowo.**
  - Obecnie powszechnie uznaje się, że nie jest możliwe z góry określenie wszystkich wymagań dla takich systemów.
- **Interfejsy użytkownika są ograniczane możliwościami przeglądarek Internetowych.**
  - Technologie takie jak AJAX pozwalają na realizację rozbudowanych i szybkich interfejsów w ramach przeglądarki. Na szeroką skalę wykorzystywane są formularze webowe oraz skrypty wykonywane lokalnie.
  - Obecnie systemy internetowe zaczynają być konkurencyjne z systemami PC w zakresie dziedzinie interfejsów użytkownika.

Podsumowanie

- Inżynieria Oprogramowania to dyscyplina inżynierii związana z wszystkim aspektami produkcji oprogramowania.
- Najważniejsze cechy produktu będącego oprogramowaniem to zdatność do konserwacji, niezawodność i bezpieczeństwo, efektywność oraz akceptowalność.
- Podstawowe czynności każdego procesu Inżynierii Oprogramowania to specyfikacja, rozwój, zatwierdzanie oraz ewolucja.
- Istnieje wiele różnych typów systemu, a każdy wymaga odpowiednich narzędzi Inżynierii Oprogramowania oraz techniki rozwoju.
- Podstawowe pojęcia Inżynierii Oprogramowania są powszechnie stosowane do rozwoju wszystkich typów systemów.