Inżynieria oprogramowania

na podstawie:
https://zasoby.open.agh.edu.pl/zasob/inzynieria-oprogramowania/

Oprogramowanie (software)

Do oprogramowania włącza się:

- pliki binarne
- pliki źródłowe
- dokumentację
- pliki konfiguracyjne
- przykładowe dane
- itp.

Inżynieria oprogramowania (Software Engineering)

- dziedzina inżynierii związana z wytwarzaniem oprogramowania we wszystkich jego fazach cyklu życia
- jako dziedzina inżynierii, IO rozważa w kontekście praktycznym rozmaite aspekty wytwarzania oprogramowania (techniczny, organizacyjny, finansowy itp.)

Cele inżynierii oprogramowania

- Cel podstawowy:

  Dostarczenie zasad organizacji procesu wytwarzania oprogramowania (sofware development) w oparciu o istniejące teorie, modele, metody i narzędzia (formalne lub nieformalne).
- W końcowym efekcie proces wytwarzania oprogramowania ma w sposób efektywny doprowadzić do powstania produktu wysokiej jakości

Problemy inżynierii oprogramowania

- Złożoność programów
- Złożoność systemów, w ramach których funkcjonują programy (ludzie, sprzęt, instytucje, procesy produkcyjne, itp.)
- Zmienność systemów (wynikająca np. ze zmiany przepisów)
- Złożoność procesów składających się na wytwarzanie oprogramowania
- Łatwość i praktyczna nieuniknioność popełniania błędów przy wytwarzaniu oprogramowania

Cechy dobrego oprogramowania

Wg. Sommerville'a:

- Poprawność, zgodność z wymaganiami użytkowników
- Łatwość pielęgnacji (konserwacji) i dokonywania zmian
- Niezawodność (dostępność i pewność)
- Bezpieczeństwo (w obu znaczeniach – safety, security)
- Wydajność, efektywne korzystanie z zasobów
- Łatwość stosowania, ergonomiczność

Historia IO: 1945 – 1965
narodziny komputerów i oprogramowania

- Komputery, wraz z oprogramowaniem, są stworzone do konkretnych celów i rodzaju obliczeń.
- Ciągła potrzeba przepisywania programów doprowadza do powstania języków programowania
- Oprogramowanie było darmowe
- Na uniwersytetach nie istnieje taki przedmiot jak Informatyka

Historia IO: 1965 – 1985
kryzys oprogramowania

- Przekraczanie budżetu – o miliony dolarów (OS/360)
- Przekraczanie czasu – na przykład 10 lat
- Uszkodzenie mienia – w wyniku błędów programu (Mariner 1)
- Zagrożenie zdrowia i życia – np.: śmierć w wyniku napromieniowania (Therac-25), błędu systemu kierowania ogniem (MIM-104 Patriot)

Historia IO: 1968 – 1969
powstaje Software Engineering

- **The NATO Software Engineering Conferences** were held in 1968 and 1969 (Garmisch, Rome)
- The term "software engineering" was deliberately chosen as the conference title to provoke thought regarding the need for disciplined approaches in software development.
- The conferences popularized the term "software engineering" and brought attention to the critical issues facing the software industry at the time. approaches in software development.

https://en.wikipedia.org/wiki/NATO_Software_Engineering_Conferences

Historia IO: 1986
brak panaceum

- Fred Brooks, *No Silver Bullet - Essence and Accident in Software Engineering*, 1986

„there is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity.”

https://en.wikipedia.org/wiki/No_Silver_Bullet

Propozycje rozwiązania problemów

- Narzędzia: różne sposoby programowania (strukturalne, OO), CASE, dokumentacja, standardy
- Dyscyplina pracy programistów
- Metody formalne z inżynierii
- Etyka pracy, licencje, profesjonalizm
- Unified Modeling Language
- Nowe języki programowania (np. Java)

Czynności w czasie produkcji oprogramowania

Co można robić w czasie tworzenia oprogramowania?

- Analiza
- Planowanie
- Projektowanie
- Implementacja
- Testowanie
- Dokumentowanie
- Wdrożenie
- Utrzymanie (konserwacja)

Faza: Analiza (ang. Analysis)

Cel: Zrozumieć **co** system ma robić, **dla kogo**, i **dlaczego**.
Innymi słowy — poznać potrzeby użytkowników i przekształcić je w jasno opisane wymagania.

Najważniejsze zadania:

- Zbieranie wymagań od klienta i użytkowników końcowych
- Analiza procesów biznesowych i problemów, które system ma rozwiązać
- Określenie **wymagań funkcjonalnych** (co system robi) i **niefunkcjonalnych** (jak dobrze to robi – np. wydajność, bezpieczeństwo)
- Tworzenie modeli i diagramów (np. przypadków użycia, przepływów danych)
- Wykrywanie sprzeczności, niejasności lub braków w wymaganiach

Faza: Planowanie (Planning)

Cel: Określić **jak**, **kiedy** i **kim** zrealizować projekt — na podstawie wyników analizy.

Najważniejsze zadania:

- Szacowanie **czasu**, **kosztów** i **zasobów** potrzebnych do realizacji projektu
- Określenie **zakresu projektu** (co będzie, a czego nie będzie w wersji końcowej)
- Ustalanie **harmonogramu** i **kamieni milowych**
- Podział projektu na **zadania** i przypisanie ich członkom zespołu
- Identyfikacja **ryzyk** i opracowanie planów awaryjnych
- Wybór **metodyki wytwarzania** (np. Agile, Waterfall, Spiral, itp.)

Faza: Projektowanie (Design)

Cel: Określić **jak system będzie zbudowany** — zaprojektować jego strukturę, architekturę i sposób działania, aby spełniał wymagania z analizy.

Najważniejsze zadania:

- Opracowanie **architektury systemu** (np. klient–serwer, mikroserwisy)
- Projektowanie **bazy danych** (modele ERD, schematy tabel)
- Definiowanie **interfejsów** między modułami i komponentami
- Projektowanie **interfejsu użytkownika**
- Wybór **technologii**, bibliotek, frameworków
- Tworzenie **modeli projektowych** (diagramy klas, sekwencji, komponentów)
- Przygotowanie **prototypów** (jeśli potrzebne)

Faza: Implementacja (Implementation)

Cel: Zamienić **projekt** w **działający kod źródłowy** — zbudować system zgodnie z dokumentacją projektową.

Najważniejsze zadania:

- Programowanie poszczególnych modułów zgodnie z projektem
- Integracja kodu w repozytorium (np. Git)
- Tworzenie **testów jednostkowych** i **modułowych**
- Pisanie **komentarzy** i **dokumentacji technicznej** kodu
- Weryfikacja zgodności implementacji z wymaganiami
- Regularne **code review** (przeglądy kodu)
- Integracja komponentów w spójną całość

Faza: Testowanie (Testing)

Cel: Sprawdzić, czy system **działa poprawnie**, spełnia wymagania i jest **wolny od błędów**.

Najważniejsze zadania:

- Przygotowanie **planów testów** i **scenariuszy testowych**
- Wykonywanie różnych rodzajów testów:
  - **Jednostkowe (Unit Tests)** – testowanie pojedynczych modułów
  - **Integracyjne (Integration Tests)** – sprawdzanie współdziałania modułów
  - **Systemowe (System Tests)** – testowanie całego systemu
  - **Akceptacyjne (Acceptance Tests / UAT)** – sprawdzenie, czy system spełnia wymagania użytkownika
  - **Wydajnościowe, bezpieczeństwa, regresji** (opcjonalnie, w zależności od projektu)
- Zgłaszanie i śledzenie **błędów / defektów**
- Weryfikacja poprawek i retesty

Faza: Dokumentowanie (Documentation)

Cel: Utworzyć **kompletną dokumentację** systemu, aby ułatwić jego **użytkowanie, utrzymanie i rozwój** w przyszłości.

Najważniejsze zadania:

- Tworzenie **dokumentacji użytkownika** (instrukcje, podręczniki, FAQ)
- Tworzenie **dokumentacji technicznej** (opis architektury, struktury kodu, konfiguracji, API)
- Dokumentowanie **procedur instalacji, konfiguracji i wdrożenia**
- Spisywanie **raportów testów i znanych błędów**
- Aktualizacja dokumentacji po każdej zmianie systemu (wersje, poprawki, nowe funkcje)

Faza: Wdrożenie (Deployment)

Cel: Udostępnić system użytkownikom końcowym w środowisku produkcyjnym, zapewniając jego poprawne działanie.

Najważniejsze zadania:

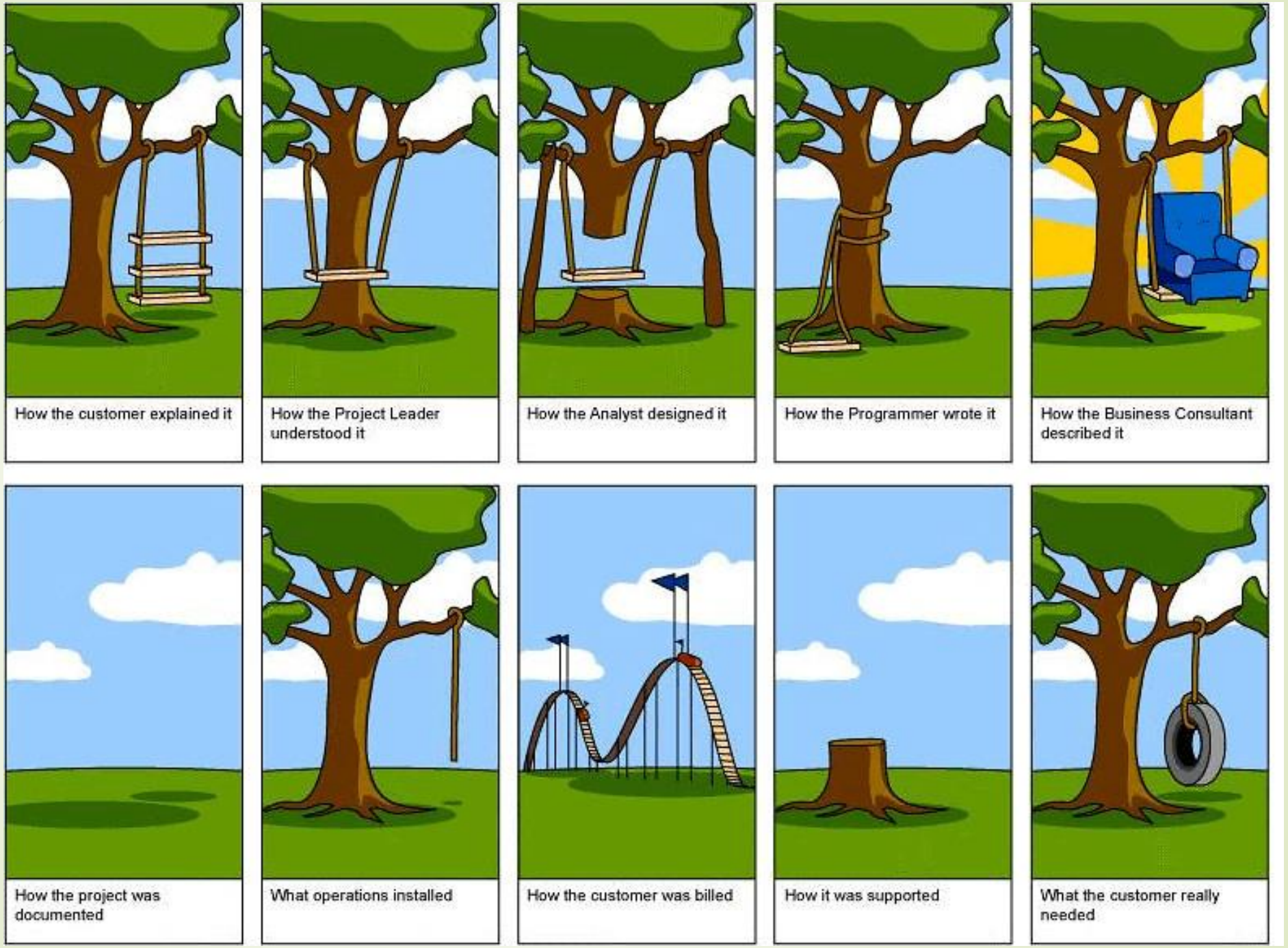
- Przygotowanie **środowiska produkcyjnego** (serwery, bazy danych, sieć)
- Instalacja i konfiguracja systemu w docelowym środowisku
- Migracja danych z poprzednich systemów (jeśli konieczne)
- Weryfikacja poprawności działania w środowisku produkcyjnym
- Szkolenie użytkowników końcowych (opcjonalnie)
- Rozwiązywanie problemów pojawiających się w trakcie pierwszego użycia

Faza: Utrzymanie (Maintenance)

Cel: Zapewnić **ciągłe działanie systemu**, jego **aktualizację**, poprawki błędów oraz rozwój funkcji w miarę potrzeb użytkowników.

Najważniejsze zadania:

- Naprawa **błędów wykrytych po wdrożeniu**
- Aktualizacja systemu w odpowiedzi na **zmieniające się wymagania** lub przepisy
- Optymalizacja działania i wydajności systemu
- Monitoring i wsparcie użytkowników (helpdesk, FAQ, zgłoszenia)
- Wdrażanie **poprawek bezpieczeństwa**
- Czasami rozwój nowych funkcji i wersji systemu


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed