ZARZĄDZANIE CYKLEM ŻYCIA APLIKACJI

INŻYNIERIA OPROGRAMOWANIA I SDLC

WPROWADZENIE

INŻYNIERIA OPROGRAMOWANIA

INŻYNIERIA OPROGRAMOWANIA

ZASTOSOWANIE SYSTEMATYCZNEGO, ZDYSCYPLINOWANEGO, ILOŚCIOWEGO PODEJŚCIA DO ROZWOJU, EKSPLOATACJI I UTRZYMANIA OPROGRAMOWANIA.

IEEE Standard Glossary of Software Engineering Terminology

INŻYNIERIA OPROGRAMOWANIA

- Wymagania
- Projektowanie
- Procesy
- Zarządzanie
- Ewolucja
- Walidacja
- Narzędzia
- API

INŻYNIERIA OPROGRAMOWANIA

- Wymagania
- Projektowanie
- Procesy
- Zarządzanie
- Ewolucja
- Walidacja

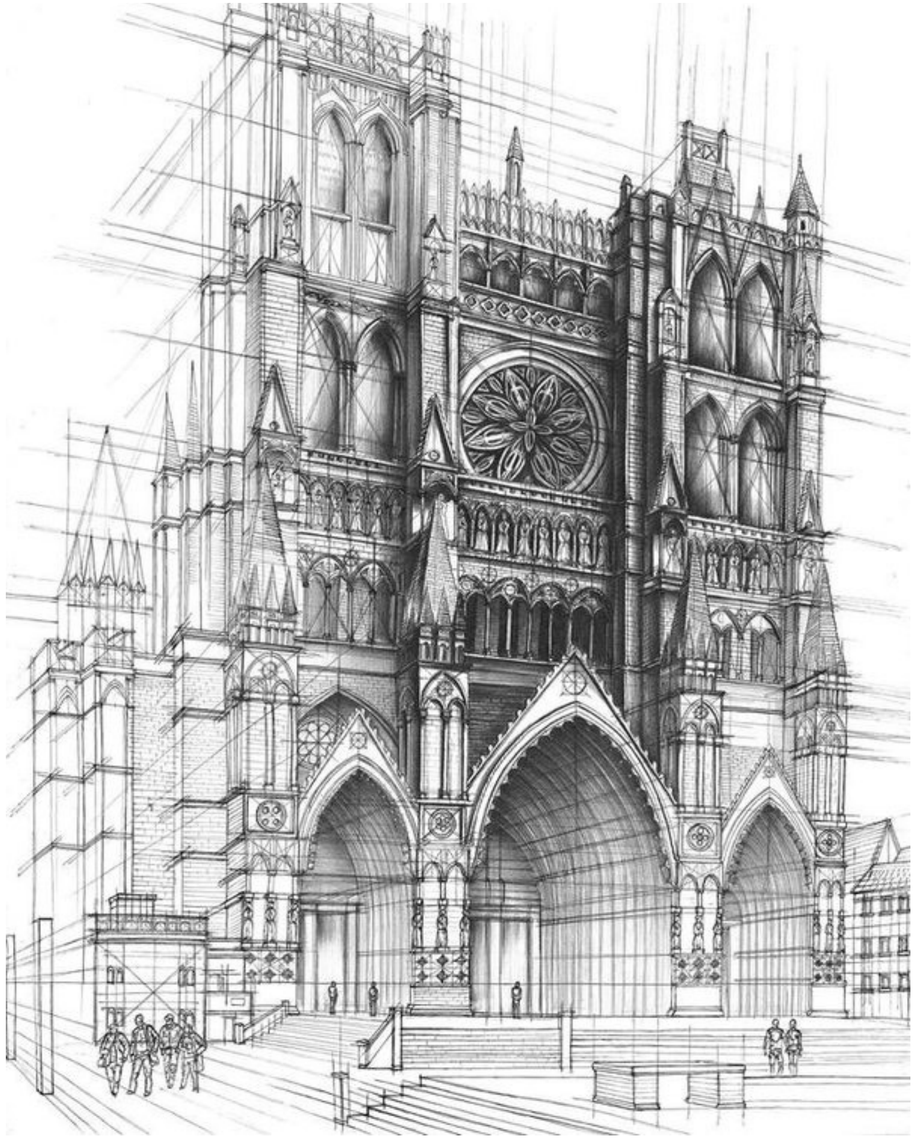
- Inżynieria wymagań
- Projektowanie oprogramowania
- Zarządzanie procesem produkcyjnym
- Klasyczne Metodyki wytwarzania oprogramowania
- Zwinne metodyki wytwarzania oprogramowania
- Testowanie

INŻYNIERIA WYMAGAŃ

- Czym jest Inżynieria Wymagań?
- Wymagania funkcjonalne, pozafunkcjonalne
- Model FURPS
- Podejścia do opisu wymagań
- Ćwiczenia: Przypadki użycia, User Stories

PROJEKTOWANIE OPROGRAMOWANIA

- Funkcja i znaczenie UML
- Podstawy notacji języka UML
- Diagram przypadków użycia



ZARZĄDZANIE PROCESEM PRODUKCYJNYM

- PMBOK i kluczowe obszary zarządzania projektem
- Procesy i produkty zarządcze
- Macierz RACI
- Rejestr Ryzyk

KLASYCZNE METODYKI

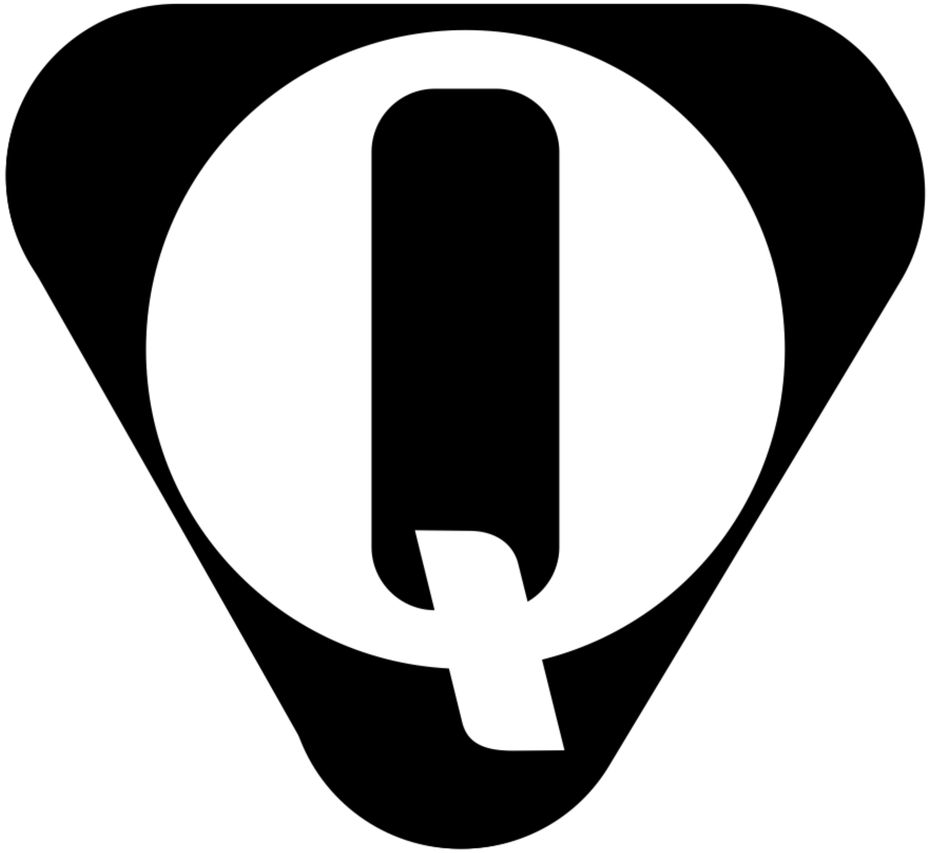
- Wprowadzenie do metodyki Prince2
- Role i odpowiedzialności w Prince 2
- Fazy projektu i produkty zarządcze w Prince 2
- Metoda Ścieżki Krytycznej
- PERT - szacowanie pracochłonności
- Harmonogramowanie i kontrola przebiegu prac przy użyciu wykresu Gantta

ZWINNE METODYKI

- SCRUM - definicja, filary wartości
- Zespół SCRUMowy
- Wydarzenia w SCRUM
- Artefakty e SCRUM

TESTOWANIE OPROGRAMOWANIA

- Testowanie jako weryfikacja i walidacja
- Statyczna i dynamiczna weryfikacja
- Testy czarno- i biało skrzynkowe
- Podział testów ze względu na przedmiot lub cel testowania
- Scenariusze i przypadki testowe
- TDD - Test Driven Development



WPROWADZENIE

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

FRAMEWORK DEFINIUJĄCY ZADANIA WYKONYWANE NA KAŻDYM ETAPIE PROCESU TWORZENIA OPROGRAMOWANIA.

ISO/IEC/IEEE 12207:2017 (Listopad 2017)

SOFTWARE DEVELOPMENT LIFE CYCLE

1. PLANOWANIE I ANALIZA WYMAGAŃ

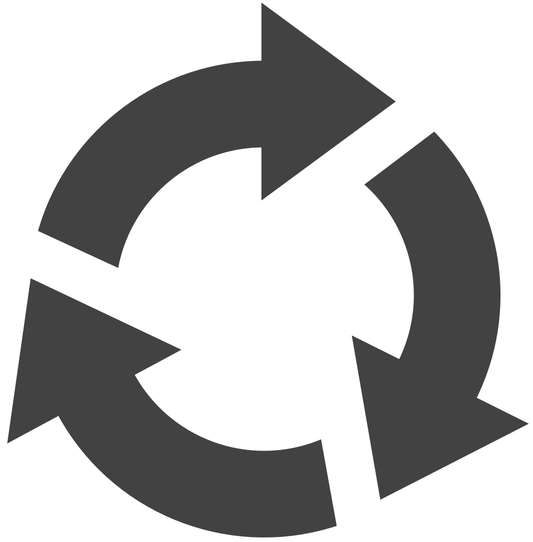
2. DEFINIOWANIE WYMAGAŃ

3. PROJEKTOWANIE SYSTEMU

4. ROZWÓJ SYSTEMU

5. INTEGRACJA I TESTOWANIE

6. WDROŻENIE I UTRZYMANIE



1. PLANOWANIE I ANALIZA WYMAGAŃ

- Zbieranie i analiza wymagań
- Źródła wymagań:
  - wywiady z interesariuszami, użytkownikami
  - analiza istniejących rozwiązań
- Studium wykonalności
- Plan testów
- Identyfikacja ryzyk i planowanie reakcji

2. DEFINIOWANIE WYMAGAŃ

- Jasne zdefiniowanie wymagań
- Udokumentowanie wymagań
  - PRD
  - Product Backlog
- Akceptacja wymagań przez klienta, interesariuszy

3. PROJEKTOWANIE

- Projekt architektury aplikacji
- Projekt infrastruktury serwerowej
- Projekt bazy danych
- Diagram przepływu danych, komunikacji z wewnętrznymi i zewnętrznymi modułami
- Analiza i zatwierdzenie projektu przez interesariuszy

4. ROZWÓJ PRODUKTU

- Rozwój i budowa modułów produktu
  - Wybór języka programowania, bibliotek
  - Pisanie kodu
  - Kompilowanie
- Unit testy

5. INTEGRACJA I TESTY

- Integracja modułów w system
- Testy integracyjne
- Testy funkcjonalne
- Testy akceptacyjne
- Testy regresyjne

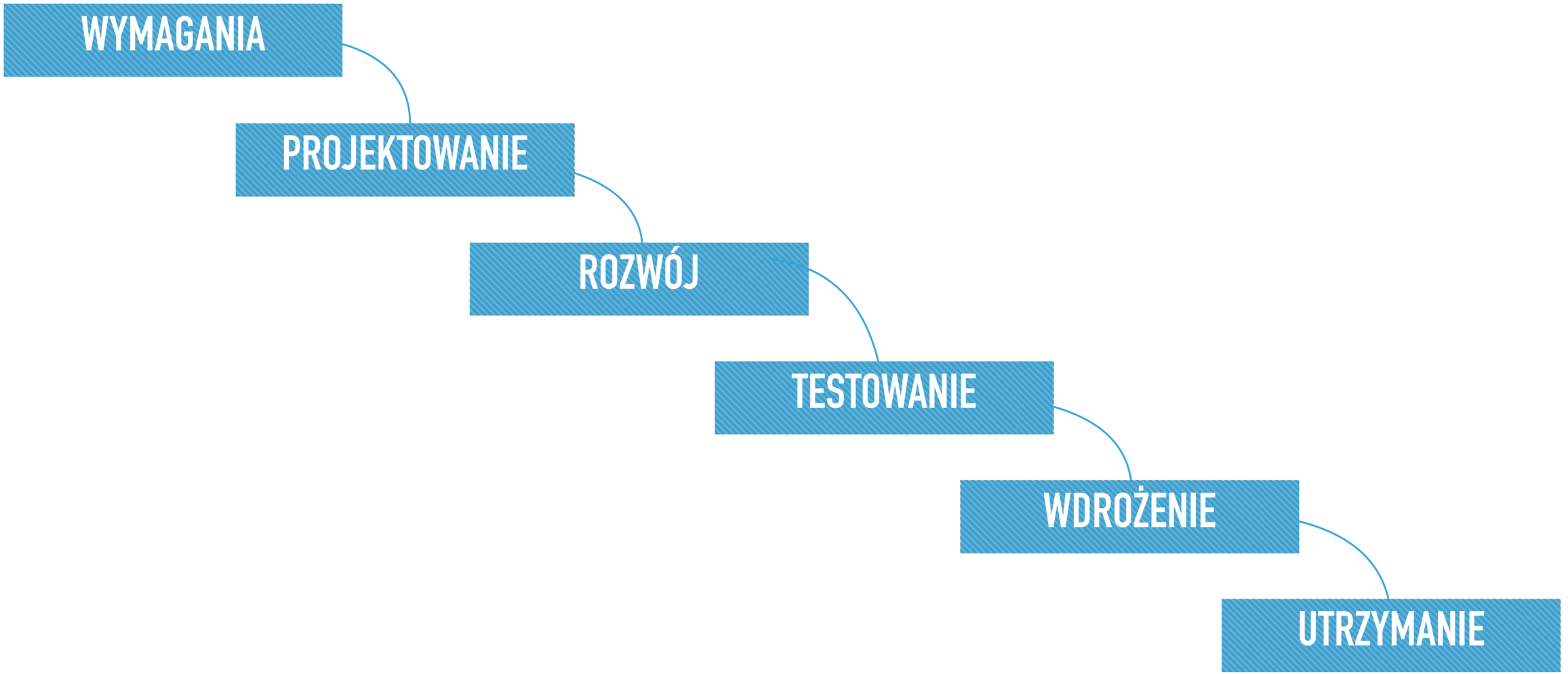
6. WDROŻENIE I UTRZYMANIE

- Plan deployment
- Wdrożenie (deployment) na środowiska produkcyjne
- Publikacja produktu
- Wdrożenie etapami:
  - Publikacja produktu o ograniczonym zakresie
  - Testy użytkownika UAT
  - Decyzja o wdrożeniu lub dalszym rozwoju w oparciu o wyniki UAT
- Utrzymanie

MODELE WYTWARZANIA OPROGRAMOWANIA

- Model Waterfall (Kaskadowy)
- Model Iteracyjny
- Model V (Weryfikacja i Walidacji)

MODEL WATERFALL (KASKADOWY)



KIEDY STOSOWAĆ WATERFALL?

- Wymagania są bardzo dobrze udokumentowane, jasne i niezmienne.
- Definicja produktu jest stabilna
- Technologia jest znana i nie ewoluuje
- Nie ma niejednoznacznych wymagań
- Projekt jest krótki

ZALETY WATERFALL

- Łatwy zrozumienia i używania
- Każda z faz ma jasno określone produkty, które ma dostarczyć oraz proces ich weryfikacji
- Tylko jedna faza na raz
- Jasno zdefiniowane etapy oraz „kamienie milowe”
- Łatwość w organizacji zadań
- Procesy i rezultaty są dobrze udokumentowane
- Sprawdza się na małych projektach z dobrze zdefniowanymi wymaganiami

WADY WATERFALL

- Działające oprogramowanie nie jest udostępnione, aż do późnych faz procesu
- Jest to powodem niepewności i zwiększonego ryzyko w projekcie
- Nierekomendowany do złożonych i długoterminowych projektów
- Niewskazany w projektach o wysokim prawdopodobieństwie zmian w wymaganiach produktowych
- Nie uwzględnia zarządzanie zmianą
- Trudność w mierzeniu postępu prac w ramach danej fazy

MODEL ITERACYJNY

KIEDY STOSOWAĆ MODEL ITERACYJNY?

- Oczekiwania względem kompletnego systemu są jasne i zrozumiałe
- Główne wymagania są zdefiniowane, natomiast niektóre funkcje lub rozszerzenia mogą wyewoluować z czasem
- Do budowy systemu zostanie użyta nowa technologia, której zespół projektowy musi się nauczyć w trakcie trwania projektu
- Członkowie zespołu o wymaganych kompetencjach nie są dostępni od reki, lecz ich udział jest zaplanowany na zasadach kontraktu w konkretnych iteracjach
- W zakresie projektu występują funkcje lub cele wysokiego ryzyka, które z czasem mogą ulec zmianie

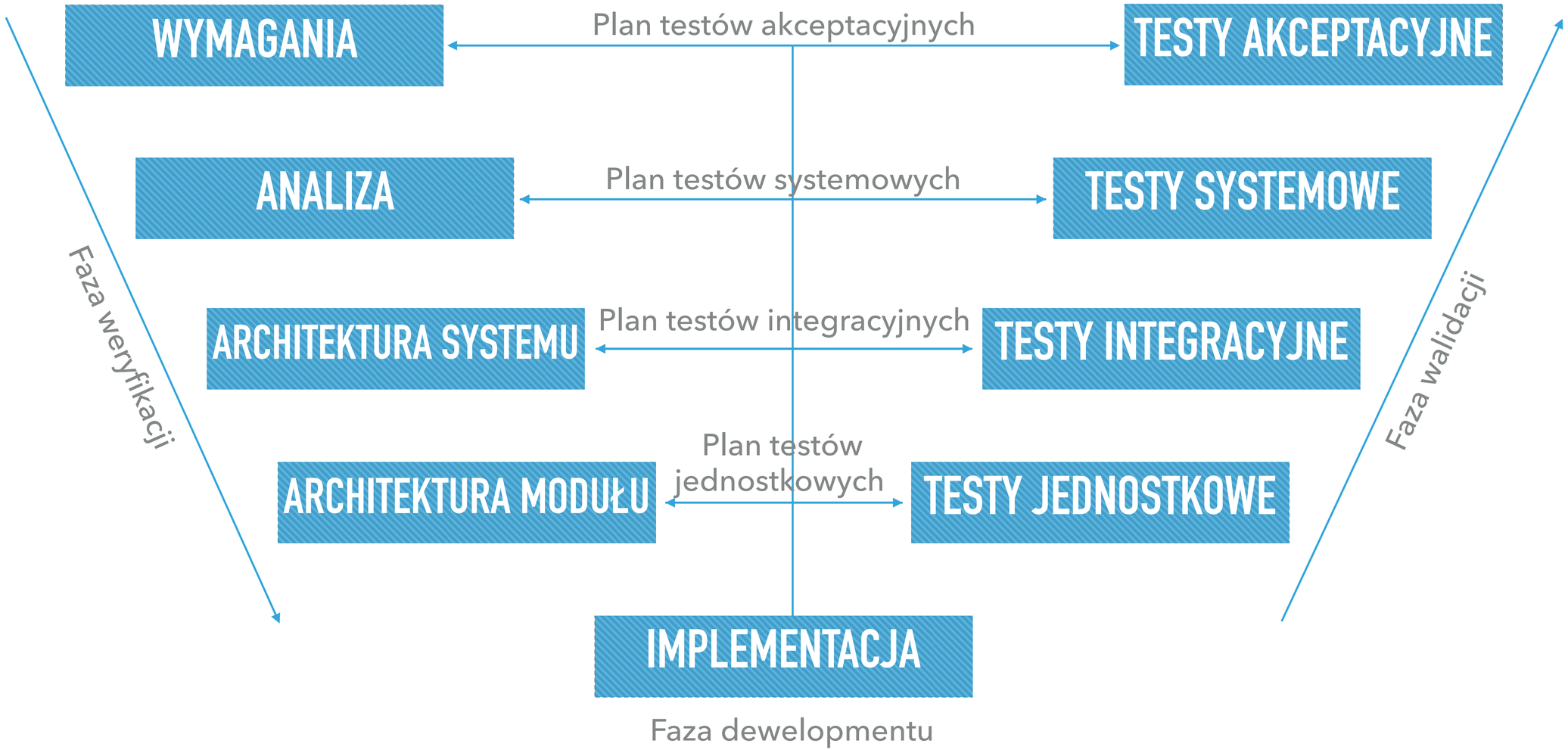
ZALETY MODELU ITERACYJNEGO?

- Rezultaty są uzyskiwane wcześnie i cyklicznie
- Z każdym przyrostem dostarczany jest działający produkt
- Testowanie i debugowanie niewielkiego zakresu danej iteracji jest łatwiejsze
- Dostępność działającego modelu systemu na wczesnym etapie cyklu produkcji oprogramowania, co wspiera ewaluację przez klienta na wczesnym etapie i zbieranie feedbacku
- Możliwość zrównoleglenia prac deweloperskich
- Łatwiejszy pomiar postępu
- Zmiany w zakresie są mniej kosztowne łatwiejsze do zarządzenia
- Ryzyka są identyfikowane i rozwiązywane w ramach iteracji, elementy wysokiego ryzyka są rozwiązywane w pierwszej kolejności
- Rekomendowany do dużych projektów o wysokim priorytecie dla organizacji

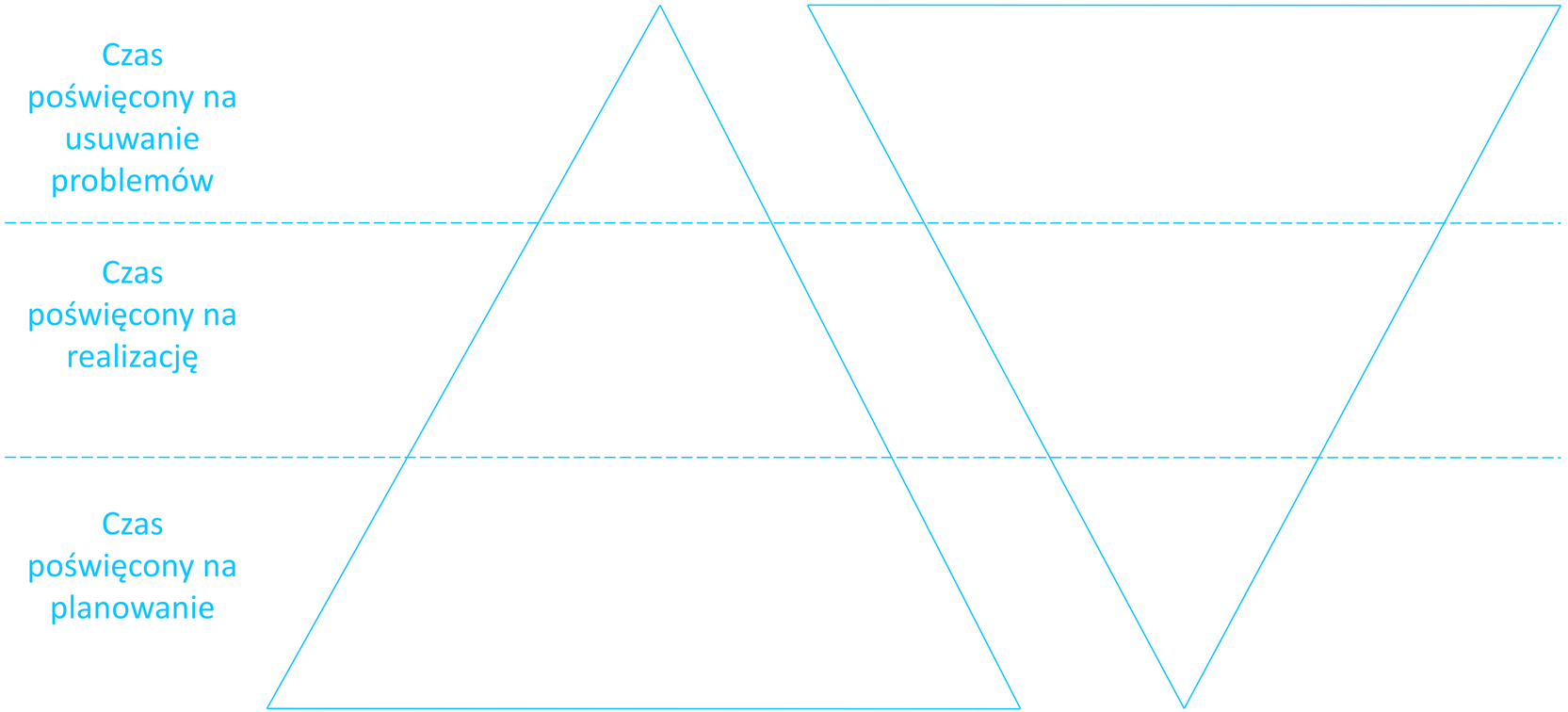
WADY MODELU ITERACYJNEGO?

- Często wymaga większej liczby zasobów
- Wymaga większej uwagi kadry zarządzającej, zarządzanie jest bardziej złożone
- Problemy związane z architekturą lub projektem systemu mogą wystąpić się na późniejszym etapie projektu
- Nie odpowiedni do małych projektów
- Ocena ryzyka wymaga zaangażowania osób o wysokich kompetencjach
- Postęp projektu jest uzależniony od analizy ryzyka

MODEL V (WALIDACJA I WERYFIKACJA)



MODEL V – ZASADA 2 TRÓJKĄTÓW



ZALETY MODELU V

- Wysoce zdyscyplinowany model - tylko jedna faza na raz może być realizowana
- Łatwość planowania, harmonogramowania, monitorowania
- Eliminacja zagrożeń na wczesnych etapach projektu
- Obniżenie kosztów usuwania usterek
- Wyczerpująca, szczegółowa specyfikacja
- Zaangażowanie zespołu w cały proces - wysoka jakość produktu
- Zarządzanie zmianą

WADY MODELU V

- Czasochłonność
- Wysoki koszt wytwarzania
- Wymaga zaangażowania dużej liczby zasobów
- Dokumentowanie na każdym z etapów