Inżynieria oprogramowania

Zastosowanie systematycznego, zdyscyplinowanego, mierzalnego podejścia do rozwoju, eksploatacji i utrzymania oprogramowania. *(za IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology)*

Inżynieria oprogramowania zajmuje się wszystkimi fazami życia oprogramowania. Oprogramowanie traktowane jest jako produkt, który powinien spełniać określone wymagania.
Inżynieria oprogramowania jest dyscypliną empiryczną. Nie ma jedynego słusznego modelu czy metodologii.

Cykl życia oprogramowania

1. Planowanie projektu
2. Implementacja
3. Wdrożenie
4. Pielęgnacja i rozwój

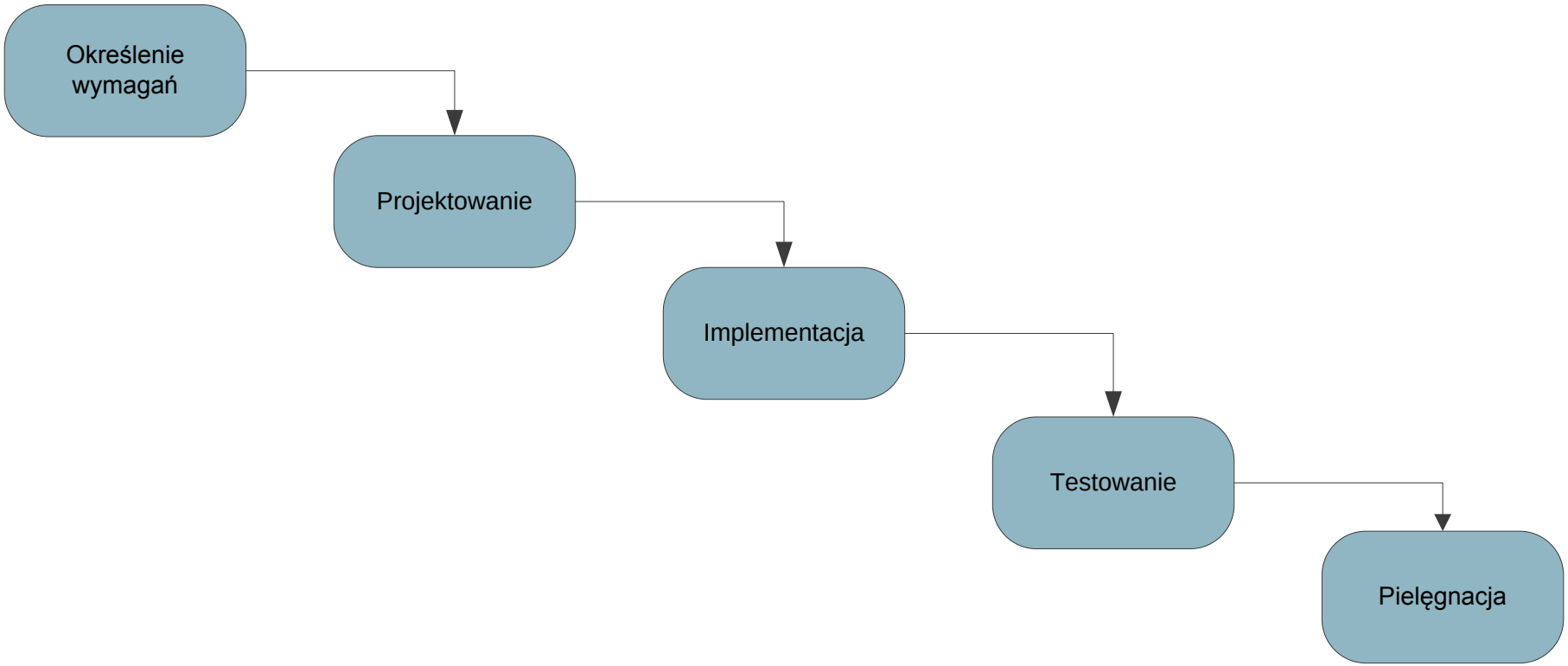
Cykl życia oprogramowania

1. Określenie wymagań
2. Projektowanie
3. Implementacja
4. Testowanie
5. Wdrożenie
6. Pielęgnacja i rozwój

Cykl życia oprogramowania

1. Określenie celów, definicja projektu
2. Określenie wymagań
3. Analiza wymagań i zasobów
4. Projektowanie
5. Implementacja
6. Testowanie
7. Dokumentacja
8. Instalacja
9. Szkolenie użytkowników
10. Pielęgnacja i rozwój

Model kaskadowy



Model kaskadowy

Model kaskadowy

Założenia modelu

- Sekwencyjne wykonywanie faz
- Precyzyjnie określone wymagania
- Wymagania nie zmieniające się w czasie

Zalety

- Łatwość zarządzania projektem
- Łatwość księgowego rozliczenia projektu

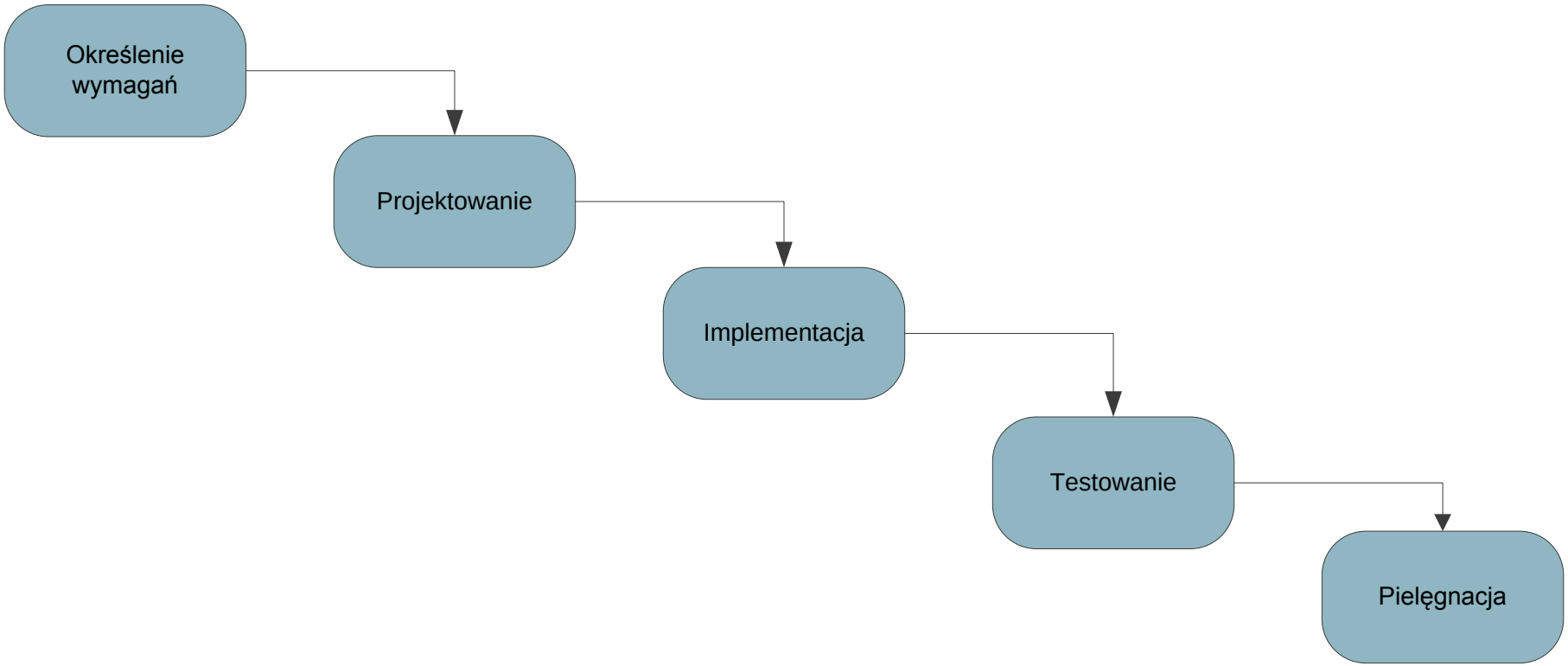
Wady

- Narzucona ścisła kolejność
- Mały kontakt z klientem
- Wysoki koszt błędów początkowych faz
- Ograniczone zastosowanie w praktyce

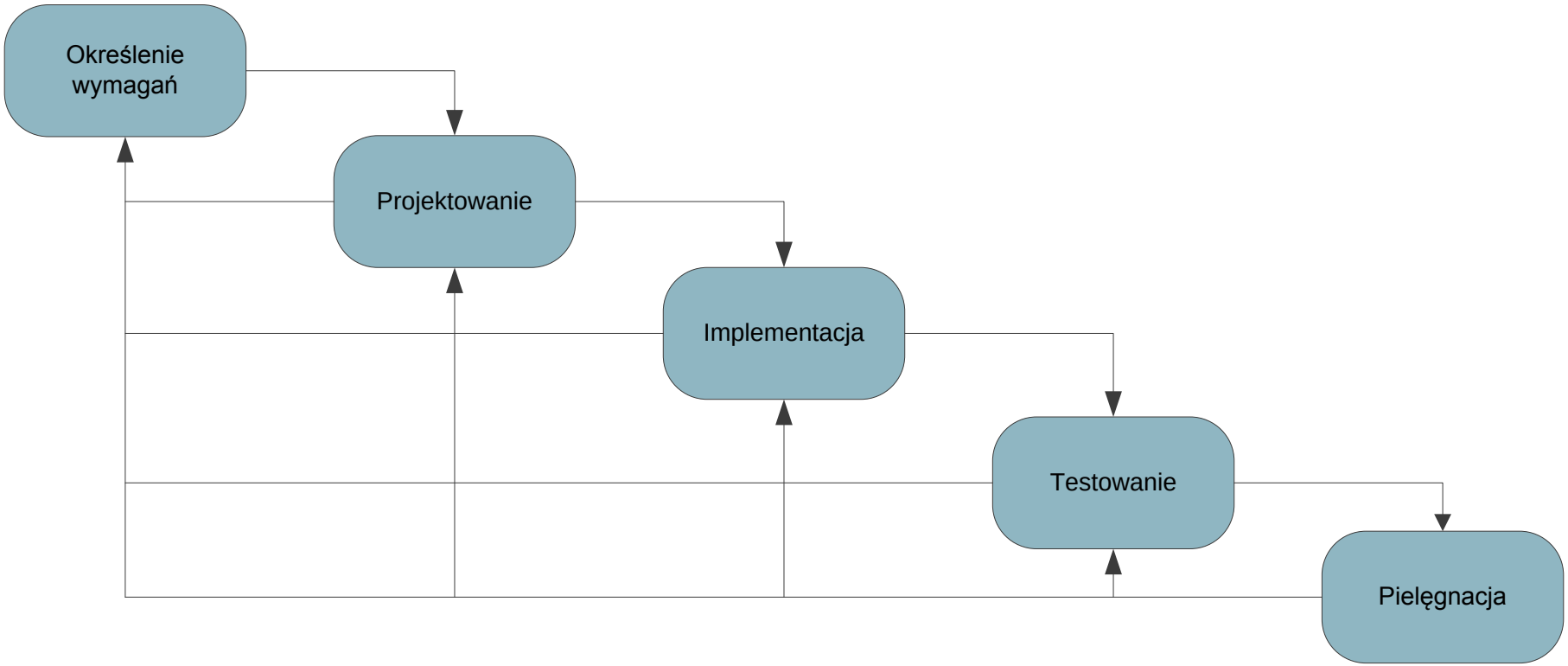
Model kaskadowy

W praktyce projekt nie przebiega sekwencyjnie. Wymagania zmieniają się w czasie i konieczne są nawroty do poprzednich faz.

Zmodyfikowany model kaskadowy



Zmodyfikowany model kaskadowy



Realizacja kierowana dokumentami

Założenia modelu

- Odmiana modelu kaskadowego
- Przyjęty przez armię amerykańską
- Każda faza kończy się stworzeniem dokumentów, w których opisane są wyniki danej fazy

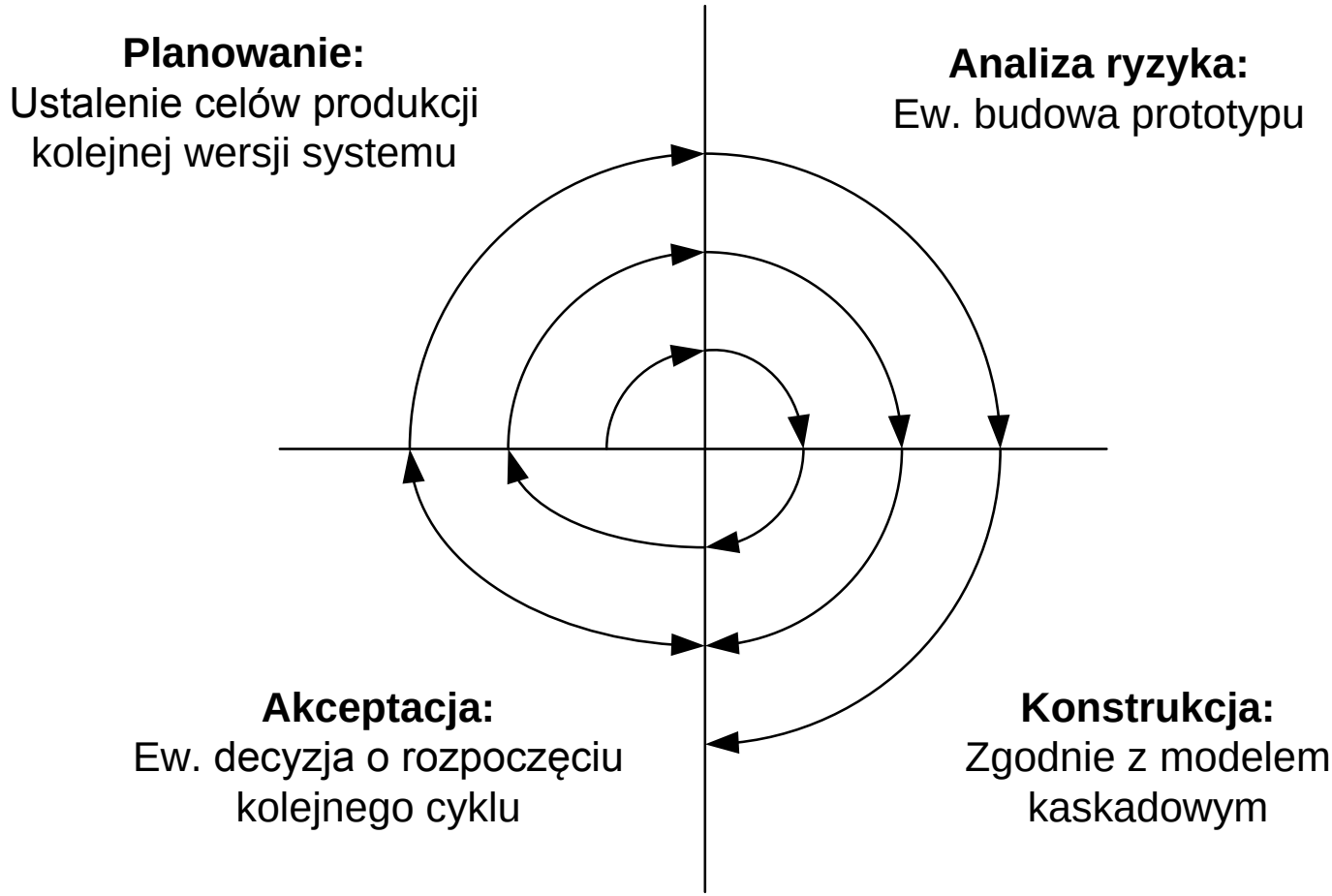
Zalety

- Łatwość zarządzania projektem
- Możliwość przekazania realizacji innemu wykonawcy

Wady

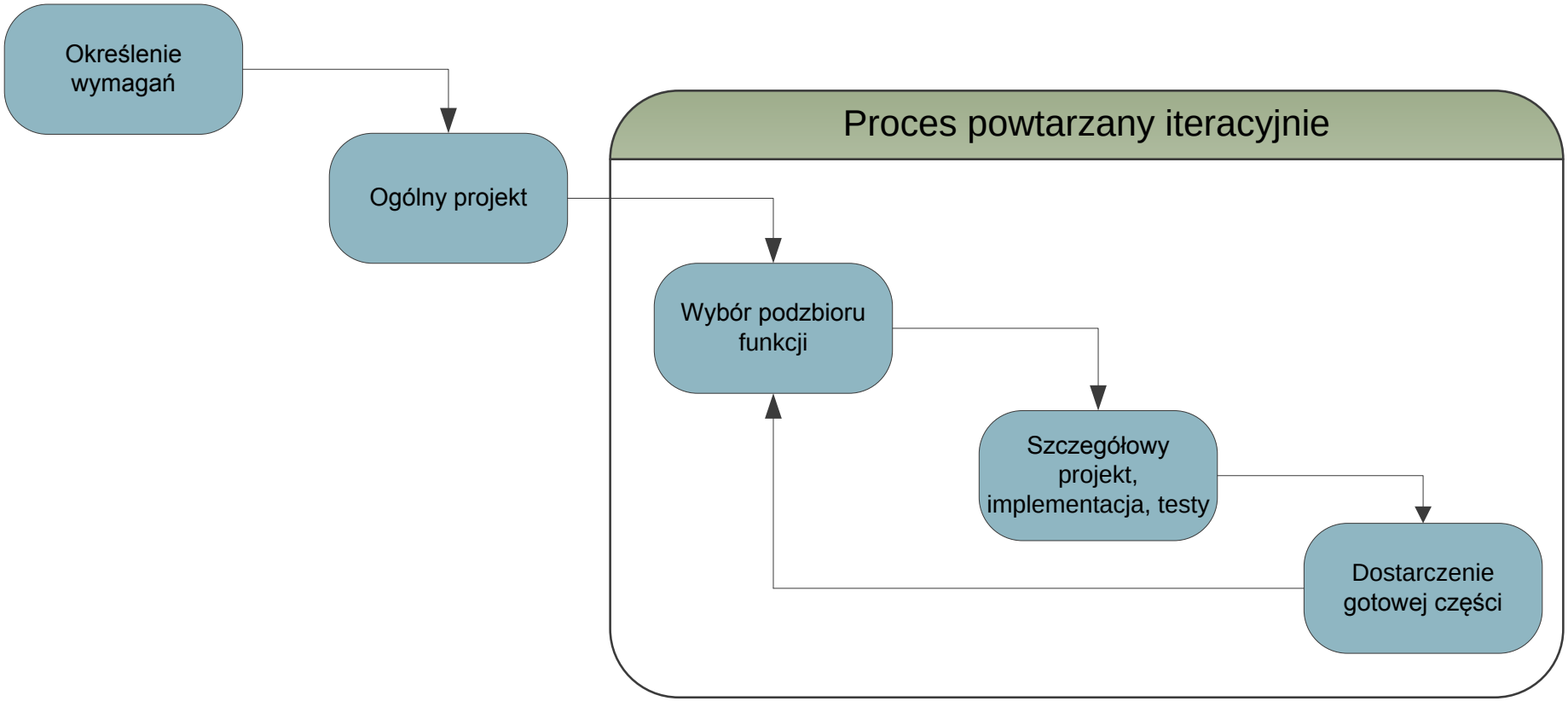
- Duży nakład pracy (i kosztów) na opracowanie dokumentów zgodnych ze standardem – ponad 50% nakładów
- Przerwy w realizacji projektu na czas weryfikacji dokumentów przez klienta

Model spiralny



Model ten ma wiele odmian.

Model przyrostowy



Prototypowanie

Należy wyraźnie rozdzielić tworzenie prototypu od tworzenia właściwego systemu.

Metody prototypowania:

- Niekompletna realizacja
- Nieoptymalna realizacja (np. za pomocą języków wyższego poziomu)
- Wykorzystywanie gotowych komponentów
- Automatyczne generowanie interfejsu (bez funkcjonalności wewnątrz)
- Zaniechanie testów

Modele przyrostowe

Założenia modelu

- Wstępny projekt, następnie doprecyzowywany
- Projekt podzielony na małe moduły
- Moduły rozwijane niezależnie, następnie integrowane
- Klient musi być cały czas dostępny

Zalety

- Możliwość równoległej pracy nad wieloma modułami
- Częsty kontakt z klientem – szybkie wykrywanie nieporozumień i braków w wymaganiach
- Wcześniejsze powstanie najważniejszych części systemu
- Możliwość szkoleń użytkowników przed ukończeniem systemu

Wady

- Więcej czasu potrzebnego na integrację modułów
- Możliwość nadmiernego rozrostu funkcjonalności systemu

Tworzenie z gotowych komponentów

Redukcja kosztów osiągana jest przez maksymalne wykorzystywanie gotowych komponentów (pochodzących z innych projektów lub dostępnych na rynku).

Zalety

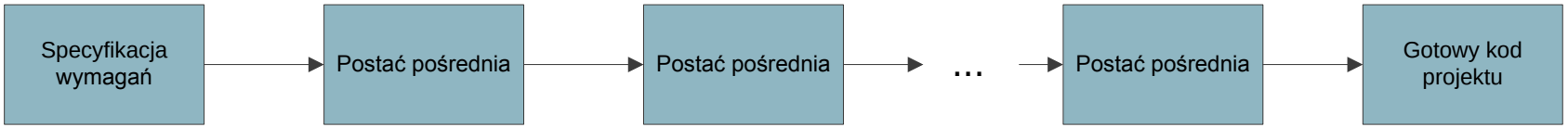
- Wysoka niezawodność modułów
- Zgodność ze standardami
- Lepsze wykorzystanie zespołu

Wady

- Koszt uzyskania i przygotowania komponentów
- Konieczność zapoznania z interfejsem komponentów
- Ryzyko uzależnienia od dostawcy komponentów

Formalne tworzenie systemów

Wymagania formułowane są w formalnym języku, a następnie poddawane szeregom transformacji aż do uzyskania wersji finalnej.



Transformacje dokonywane są bez bezpośredniej ingerencji ludzi.

Ze względu na nikłe możliwości zastosowań praktycznych, metodyka ta pozostaje raczej ciekawostką.