MODELE CYKLU ŻYCIA OPROGRAMOWANIA

Plan prezentacji:

- Definicja procesu i procesu programowego
- Model buduj i poprawiaj
- Model kaskadowy (czysty i z nawrotami)
- Modele ewolucyjne (spiralny i przyrostowy)
- Prototypowanie
- RAD
- Montaż z gotowych komponentów
- Model formalnych transformacji

Przypomnienie - podstawowe **fazy cyklu życiowego** programu:

- Faza strategiczna
- Faza specyfikacji i analizy wymagań
- Faza projektowania
- Faza konstrukcji- implementacji
- Faza testowania
- Faza konserwacji

W pierwszej fazie (strategicznej) podejmowane są czynności poprzedzające decyzję o realizacji przedsięwzięcia. Między innymi, decyzje o sposobie organizacji i harmonogramowania.

Decyzje strategiczne:

- **Wybór modelu**, zgodnie z którym będzie realizowane przedsięwzięcie
- Wybór technik stosowanych w fazach analizy i projektowania
- Wybór środowiska (środowisk) implementacji
- Wybór narzędzia CASE
- Określenie stopnia wykorzystania gotowych komponentów
- Podjęcie decyzji o współpracy z innymi producentami

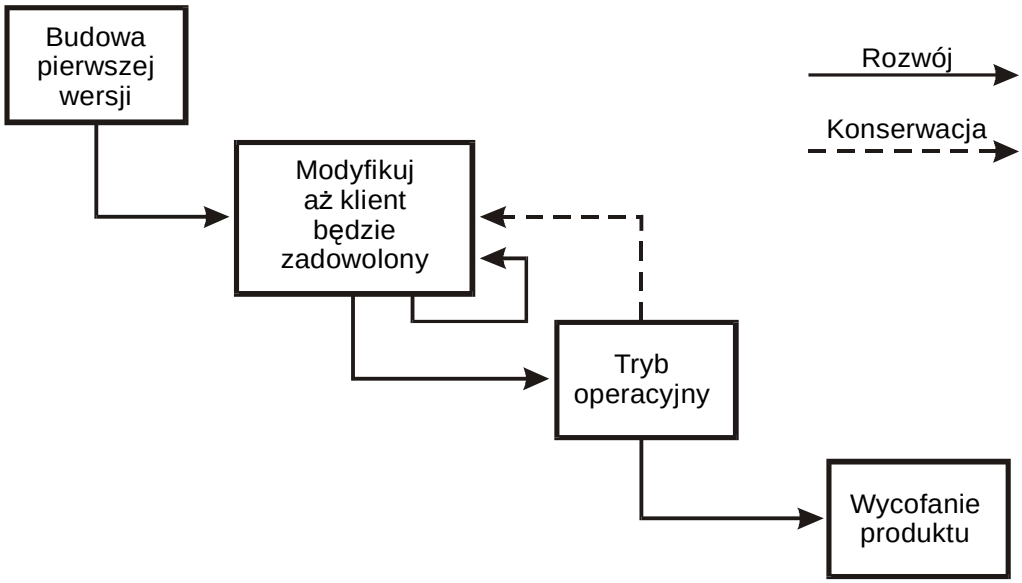
Proces programowy jest to zestaw czynności, metod, przekształceń, które stosują ludzie, by opracowywać i konserwować oprogramowanie oraz inne pokrewne produkty z nim związane.

np. plan realizacji przedsięwzięcia, projekt, kod, zestawy testów, podręczniki użytkownika, itp.

Dla lepszego zrozumienia i organizacji tego procesu, korzystne jest ustalenie jego modelu.

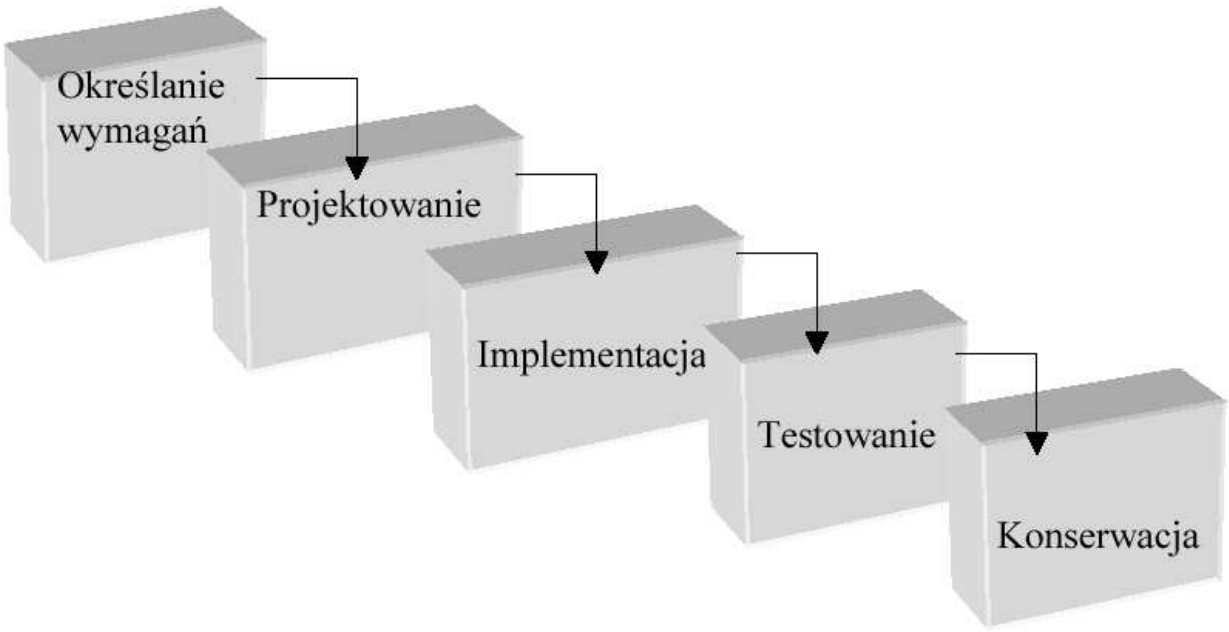
---

0) Naturalny model – „Buduj i poprawiaj”



- kiedy rozwiązywany problem jest niewielki
- kiedy nie wiemy jak się za niego zabrać

1) Model kaskadowy - wodospadowy *(ang. waterfall model)*



Zalety:

- ułatwia organizację: planowanie, harmonogramowanie, monitorowanie przedsięwzięcia
- zmusza do zdyscyplinowanego podejścia
- wymusza kończenie dokumentacji po każdej fazie
- wymusza sprawdzenie każdej fazy przez SQA

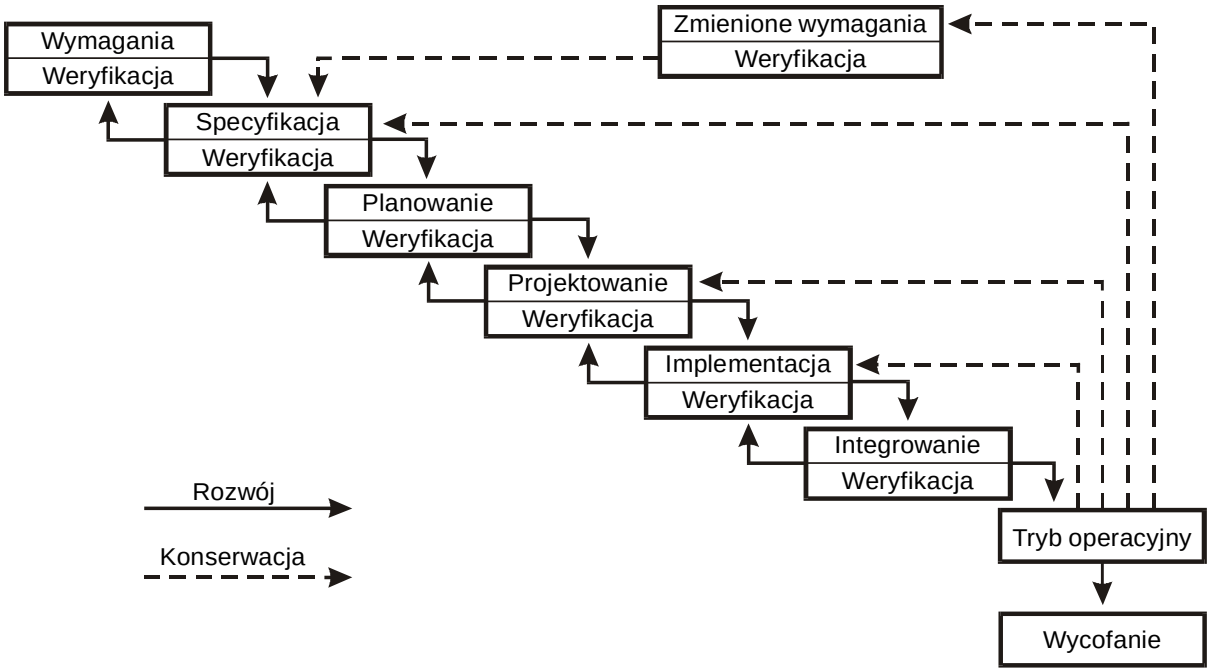
Wady:

- narzuca twórcom oprogramowania ścisłą kolejność wykonywania prac
- występują trudności w sformułowaniu wymagań od samego początku
- powoduje wysokie koszty błędów popełnionych we wczesnych fazach,
- powoduje długie przerwy w kontaktach z klientem.
- brak jest weryfikacji i elastyczności
- możliwa jest niezgodność z faktycznymi potrzebami klienta
- niedopasowanie - rzeczywiste przedsięwzięcia rzadko są sekwencyjne
- realizatorzy kolejnych faz muszą czekać na zakończenie wcześniejszych

Stosowany w projekcie o dobrze zdefiniowanych wymaganiach dla dobrze rozumianych zastosowań. Rzadko stosuje się ten model w czystej postaci, ale stanowi on bazę dla innych modeli powstałych jako jego udoskonalenia.

2) Model kaskadowy z iteracjami (z nawrotami)

Jest to odmiana modelu kaskadowego



W czystym modelu kaskadowym:

- zaplanowane fazy pracy powinny być w zasadzie realizowane po kolei;
- konieczność powrotu do wcześniejszej fazy traktuje się jako nieprawidłowość, czyli sytuację awaryjną.

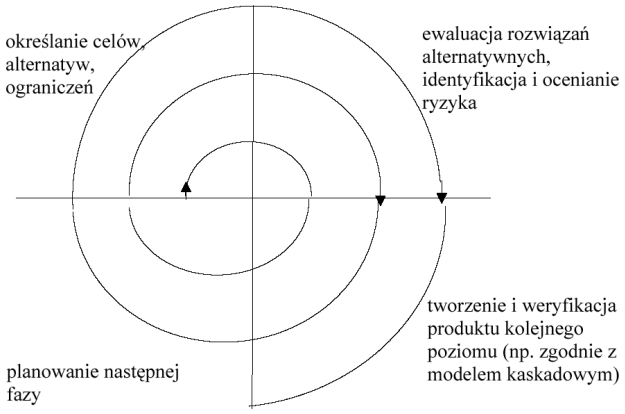
W iteracyjnym modelu kaskadowym:

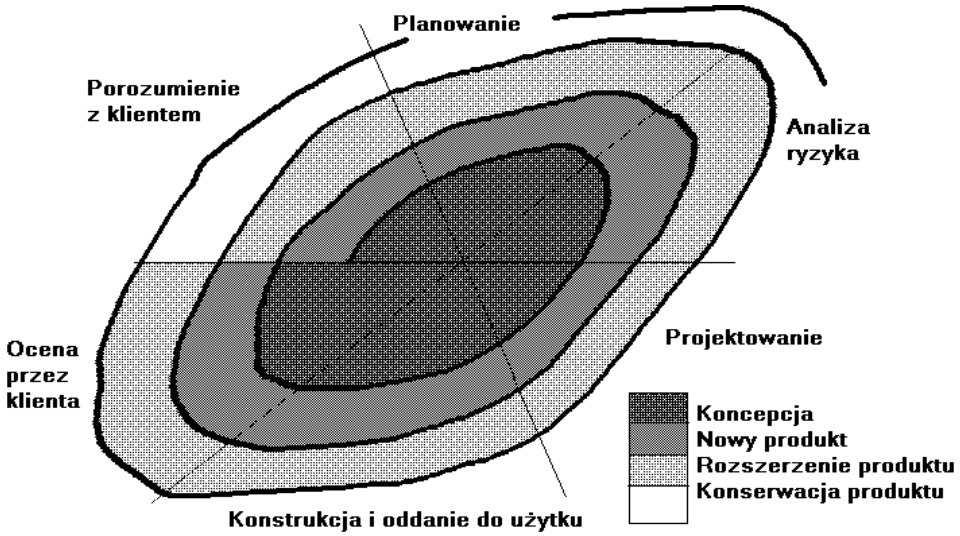
- Takie powroty z góry się przewiduje, daje to większą elastyczność, ale wydłuża czas przedsięwzięcia
- Każda faza kończy się sporządzeniem szeregu dokumentów, w których opisuje się wyniki danej fazy.
- Łatwe planowanie, harmonogramowanie oraz monitorowanie przedsięwzięcia.
- Dodatkowa zaleta: (teoretyczna) możliwość realizacji dalszych faz przez inną firmę.

Wady:

- Duży nakład pracy na opracowanie dokumentów zgodnych ze standardem
- Przerwy w realizacji niezbędne dla weryfikacji dokumentów przez klienta.

3) Model spiralny (*ang. spiral model*)





Regiony zadań modelu spiralnego:

- **Porozumienie z klientem** - ustanowienie efektywnego porozumiewania się między producentem a klientem.
- **Planowanie** - zdefiniowanie zasobów, terminów i inne ustalenia.
- **Analiza ryzyka** - ocena ryzyka technicznego i związanego z zarządzaniem projektem.
- **Projektowanie** - zadania związane z analizą i budową projektu.
- **Konstrukcja i oddanie do użytku**: konstrukcja, testowanie, instalacja i wspomaganie użytkownika
- **Ocena dokonana przez klienta**
  uzyskanie od klienta informacji nt. oceny projektu i jego implementacji.

Zalety:

- Do dużych systemów - szybka reakcja na pojawiające się czynniki ryzyka
- Połączenie iteracji z klasycznym modelem kaskadowym

Wady:

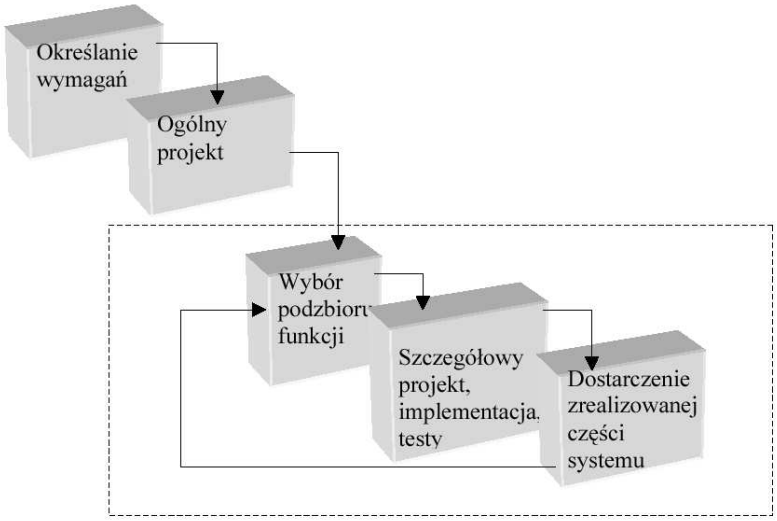
- Trudno do niego przekonać klienta
- Konieczność umiejętności szacowania ryzyka
- Problemy, gdy źle oszacujemy ryzyko

Istnieje wiele wariantów tego modelu → np. realizacja przyrostowa

---

4) Model „realizacji przyrostowej” (*ang. incremental development*)

Jest to odmiana modelu spiralnego. Wybierany jest i realizowany podstawowy zestaw funkcji. Po realizacji pewnych funkcji następuje zrealizowanie i dostarczenie kolejnych funkcji.



Zalety:

- skrócenie przerw w kontaktach z klientem
- możliwość wczesnego wykorzystania przez klienta dostarczonych fragmentów systemu
- możliwość elastycznego reagowania na powstałe opóźnienia

Wady:

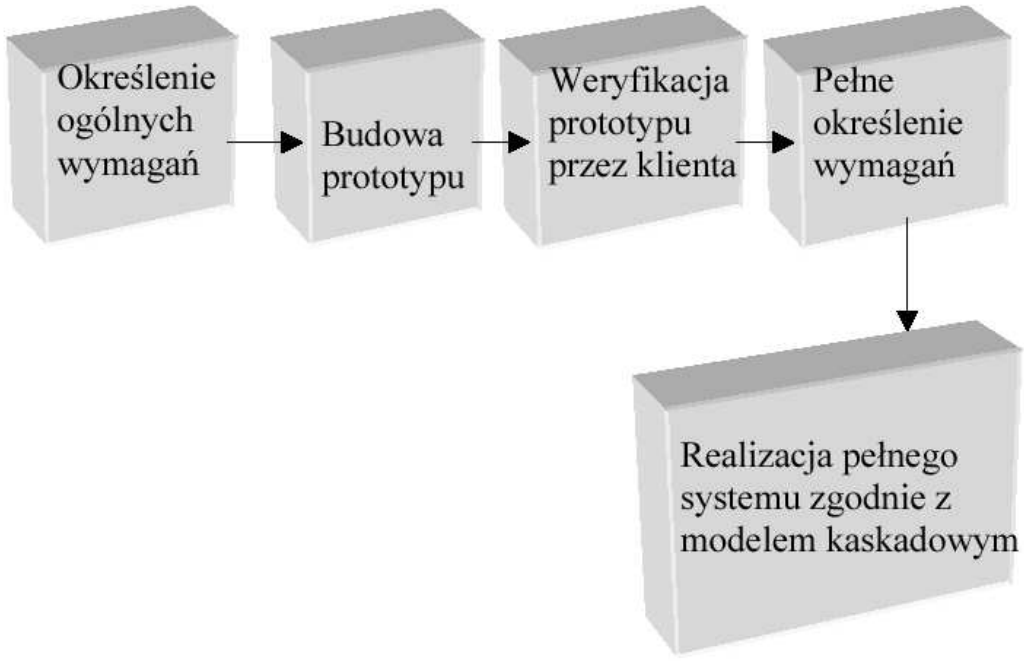
- dodatkowy koszt towarzyszący niezależnej realizacji fragmentów systemu

5) Prototypowanie - Model szybkiego prototypu

Prototypowanie - sposób na uniknięcie zbyt wysokich kosztów błędów popełnionych w fazie określania wymagań. Zalecany w przypadku, gdy określenie początkowych wymagań jest stosunkowo łatwe.

Model kaskadowy wymaga określenia wymagań na samym początku pracy. Możemy to ułatwić przez zbudowanie prototypu. Występują wówczas dodatkowe fazy projektu poprzedzające wykonanie zadań zgodnych ze zwykłym modelem kaskadowym:

- **wstępne określenie wymagań,**
- **budowa prototypu**
- **weryfikacja prototypu przez klienta**
- pełne określenie wymagań
- realizacja pełnego systemu zgodnie z modelem kaskadowym



Cele:

- wykrycie nieporozumień pomiędzy klientem a twórcami systemu
- wykrycie brakujących funkcji
- wykrycie trudnych usług
- wykrycie braków w specyfikacji wymagań

Takie podejście zwiększa koszt przedsięwzięcia, ale:

- prototyp nie musi być wykonany w pełni,
- nie musi być niezawodny ani starannie przetestowany,
- nie musi działać szybko ani mieć dobrego interfejsu użytkownika,
- może być realizowany w nieoptymalnych językach bardzo wysokiego poziomu, można nie instalować go u klienta.

Zalety prototypowania :

- lepsze poznanie potrzeb i wymagań klienta
- możliwość szybkiej demonstracji pracującej wersji systemu
- możliwość szkoleń zanim zbudowany zostanie pełny system

Wady:

- niezadowolenie klienta, który po obejrzeniu działającego prototypu musi następnie długo czekać na dostawę gotowego systemu
- (pozorna) koszt budowy prototypu

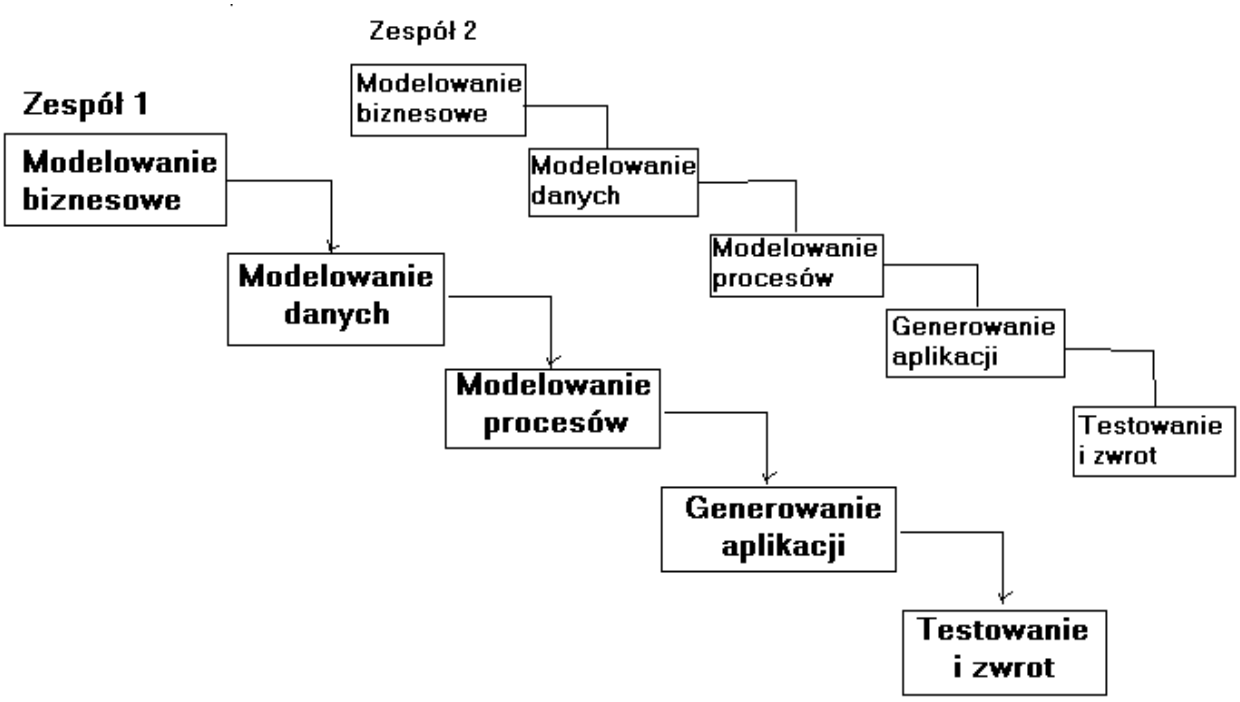
Metody prototypowania

- Niepełna realizacja: objęcie tylko części funkcji
- Języki wysokiego poziomu: Smalltalk, Lisp, Prolog, 4GL, ...
- Wykorzystanie gotowych komponentów
- Generatory interfejsu użytkownika: wykonywany jest wyłącznie interfejs, wnętrze systemu jest “podróbką”.
- Szybkie programowanie: normalne programowanie, ale bez zwracania uwagi na niektóre jego elementy, np. zaniechanie testowania

6) Model RAD (*ang. Rapid Application Development*)

Szczególna implementacja modelu kaskadowego,
można ją zastosować, gdy:

- system jest skalowalny – składa się z kilku słabo ze sobą powiązanych lub niepowiązanych głównych funkcji; każdą funkcję można przydzielić do realizacji innemu zespołowi produkcyjnemu; zespoły pracują niezależnie od siebie, a na końcu integrowane są efekty ich prac,
- zakłada się stosowanie gotowych komponentów wielokrotnego wykorzystania oraz stosowanie technik i języków 4-tej generacji



Zalety modelu RAD:

- szybkość

Wady:

- musi być skalowalny
- duże zasoby pracownicze
- intensywne zaangażowanie pracowników
- nie dla wszystkich rodzajów aplikacji

7) Montaż z gotowych komponentów (*ang. reuse*)

Kładzie nacisk na możliwość redukcji nakładów poprzez wykorzystanie podobieństwa tworzonego oprogramowania do wcześniej tworzonych systemów oraz wykorzystanie gotowych komponentów dostępnych na rynku.

Metody:

- zakup elementów ponownego użycia od dostawców
- przygotowanie elementów poprzednich przedsięwzięć do ponownego użycia

Zalety:

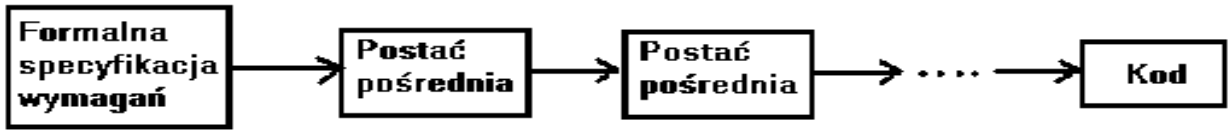
- wysoka niezawodność
- zmniejszenie ryzyka
- efektywne wykorzystanie specjalistów
- narzucenie standardów
- redukcja kosztów

Wady:

- dodatkowy koszt przygotowania elementów ponownego użycia
- dodatkowy koszt standaryzacji
- ryzyko uzależnienia się od dostawcy elementów

---

8) Model formalnych transformacji



Wady

- Trudność formalnej specyfikacji
- Wiele czasu i kosztowny
- Niewielu informatyków ma odpowiednie podstawy matematyczne
- Trudności w komunikacji z użytkownikiem
- Mała efektywność kodu

Zalety

- Oczekiwana jest duża bezbłędność kodu wynikowego

Podsumowanie zasad wyboru modelu:

- Kiedy na własny użytek chcemy rozwiązać pewien niewielki problem i nie wiemy, jak się za niego zabrać → model buduj-i-poprawiaj.
- Kiedy wymagania są dobrze zdefiniowane, produkt jest podobny do realizowanych przez nas do tej pory, mamy doświadczenie w realizacji podobnych przedsięwzięć → model kaskadowy lub montażu z gotowych elementów.
- Kiedy klient ma problemy ze wyartykułowaniem swych wymagań → prototypowanie
- Kiedy istnieje duża niepewność związana z wytwarzaniem produktu i duże ryzyko → podejście ewolucyjne (modele przyrostowy, spiralny).
- Kiedy wymagania są dość dobrze zdefiniowane, ale występuje dość duża złożoność problemu (np. bardzo duży system – dużo kodu do napisania) – podejście przyrostowe.
- Kiedy są krótkie terminy, a system jest dość duży
  → zastosować podejście RAD
  (gdy dysponujemy dużym zespołem produkcyjnym)
  → przyrostowe
  (jeżeli mamy niewielki zespół,
  a klientowi zależy głównie na najważniejszych funkcjach).
- Kiedy są krótkie terminy, system jest niewielki, a klientowi bardzo na nim zależy → model programowania ekstremalnego.
- Kiedy klient ma chwilowo małe fundusze, ale jest szansa, że w trakcie realizacji budżet się zwiększy → model spiralny.
- Kiedy system jest typu krytycznego → model formalnych transformacji.
- W każdym możliwym przypadku budować z gotowych elementów.
- W przypadku wyboru dowolnego modelu, jeśli wymagania są źle określone → zrobić prototyp