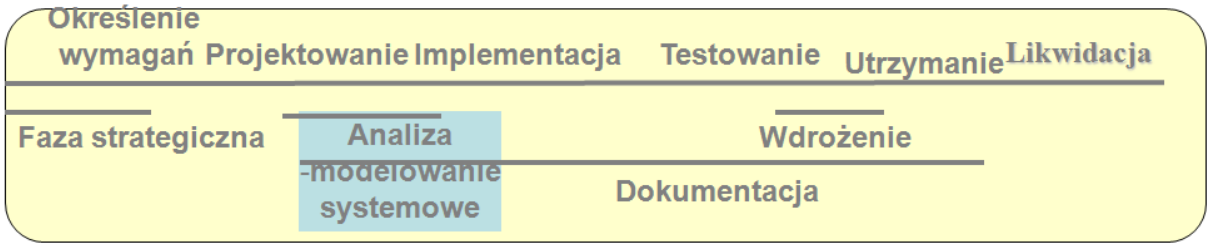
Modele cyklu życia oprogramowania.

Cykl życia oprogramowania – ciąg etapów (faz) w życiu oprogramowania – od powstania potrzeby istnienia do zaprzestania jego użytkowania. Obejmuje wiele etapów składających się na projektowanie, programowanie (kodowanie), testowanie i wdrażanie oprogramowania oraz czas jego użytkowania.

Fazy cyklu życia oprogramowania:

- faza strategiczna.
- określanie wymagań.
- analiza - modelowanie systemowe.
- projektowanie.
- implementacja.
- integracja i testowanie.
- wdrożenie.
- utrzymanie.
- likwidacja.



Modele cyklu życia oprogramowania odpowiadają na pytanie JAK powinien przebiegać proces wytwarzania oprogramowania.

Cechy modeli:

- porządkują przebieg prac.
- ułatwiają planowanie zadań.
- ułatwiają monitorowanie realizacji zadań.

Podstawowe czynności związane z tworzeniem oprogramowania:

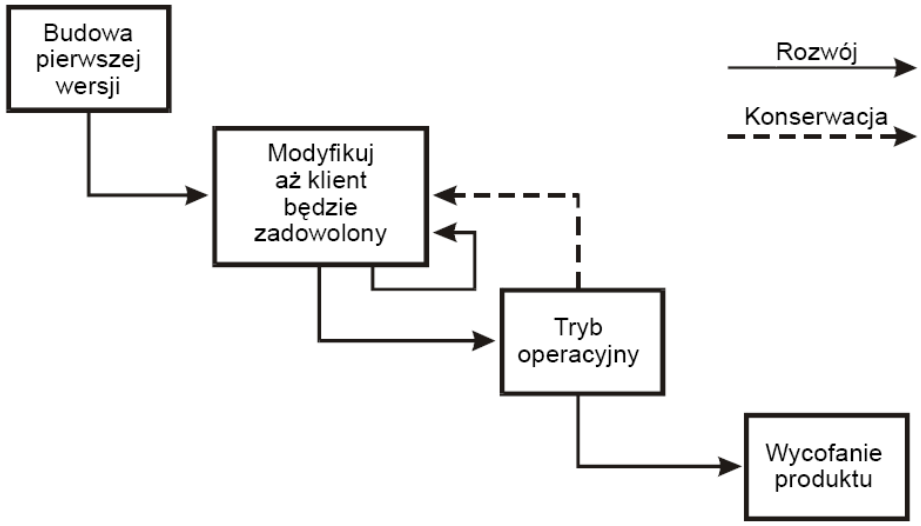
- Określanie wymagań i specyfikacji
  - Zdefiniowanie oczekiwań wobec wszystkich elementów systemu (rzeczywistego)
  - Wybranie tych oczekiwań, w spełnieniu których ma pomóc oprogramowanie.
  - Analiza wymagań: określenie dziedziny informacyjnej produktu, jego oczekiwane funkcje, zachowanie, efektywność, interfejsy.
- Projektowanie
  - Struktury danych, architektura, interfejsy użytkownika, szczegóły algorytmiczne.
- Implementacja

    - Generowanie kodu na podstawie projektu.
- Testowanie
    - Walidacja i weryfikacja
- Konserwacja (pielęgnacja)
    - Poprawianie
    - Adaptowanie
    - Rozszerzenie
    - Zapobieganie

Modele cyklu życia oprogramowania:

0. Naturalny – „Pisz i poprawiaj”
1. Kaskadowy (Waterfall, Sekwencyjny model liniowy)
    a. Tradycyjny.
    b. Z iteracjami.
    c. Model V.
2. Prototypowy
3. RAD (Model szybkiej rozbudowy aplikacji)
4. Przyrostowy (Iteracyjny)
5. Spiralny
6. Równoległy
7. Programowanie odkrywcze.
8. Tworzenie oprogramowania opartego na komponentach.

Ad. 0 Model Naturalny – „Pisz i poprawiaj”

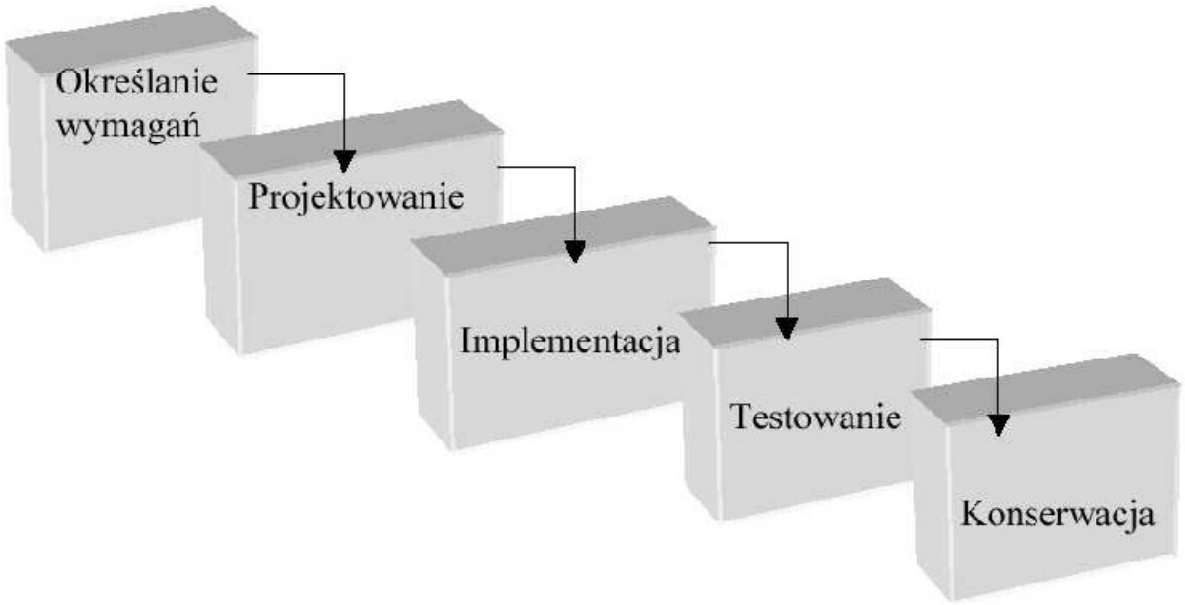


Stosowany gdy:

- Rozwiązywany problem jest niewielki
- Kiedy nie wiemy jak się za niego zabrać.

Ad. 1 Model kaskadowy

Polega on na wykonywaniu podstawowych czynności jako odrębnych faz projektowych, w porządku jeden po drugim. Każda czynność to kolejny schodek (kaskada).



Zalety:

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

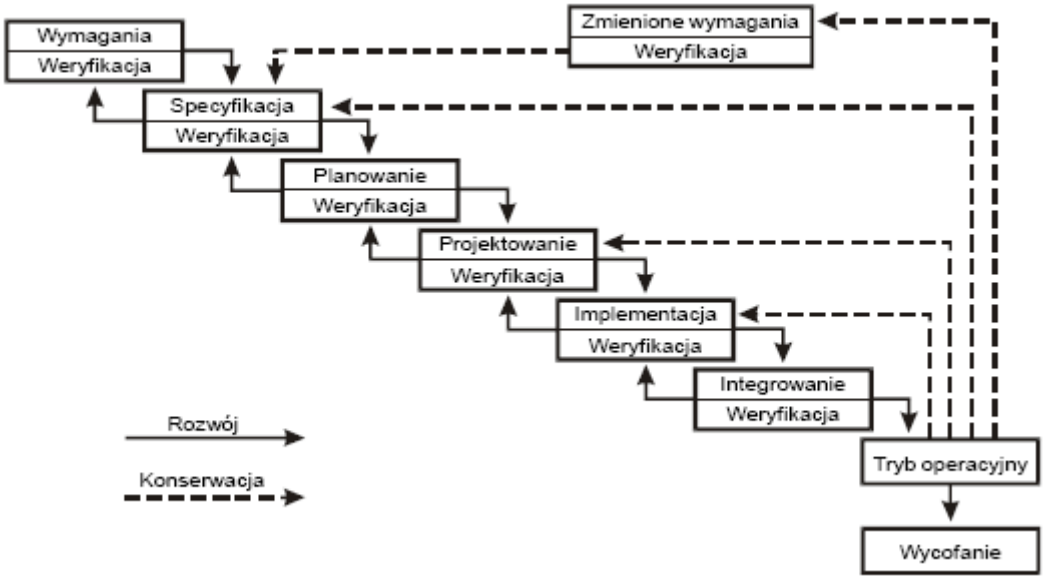
Wady:

- 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.

Model Kaskadowy z iteracjami.

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.

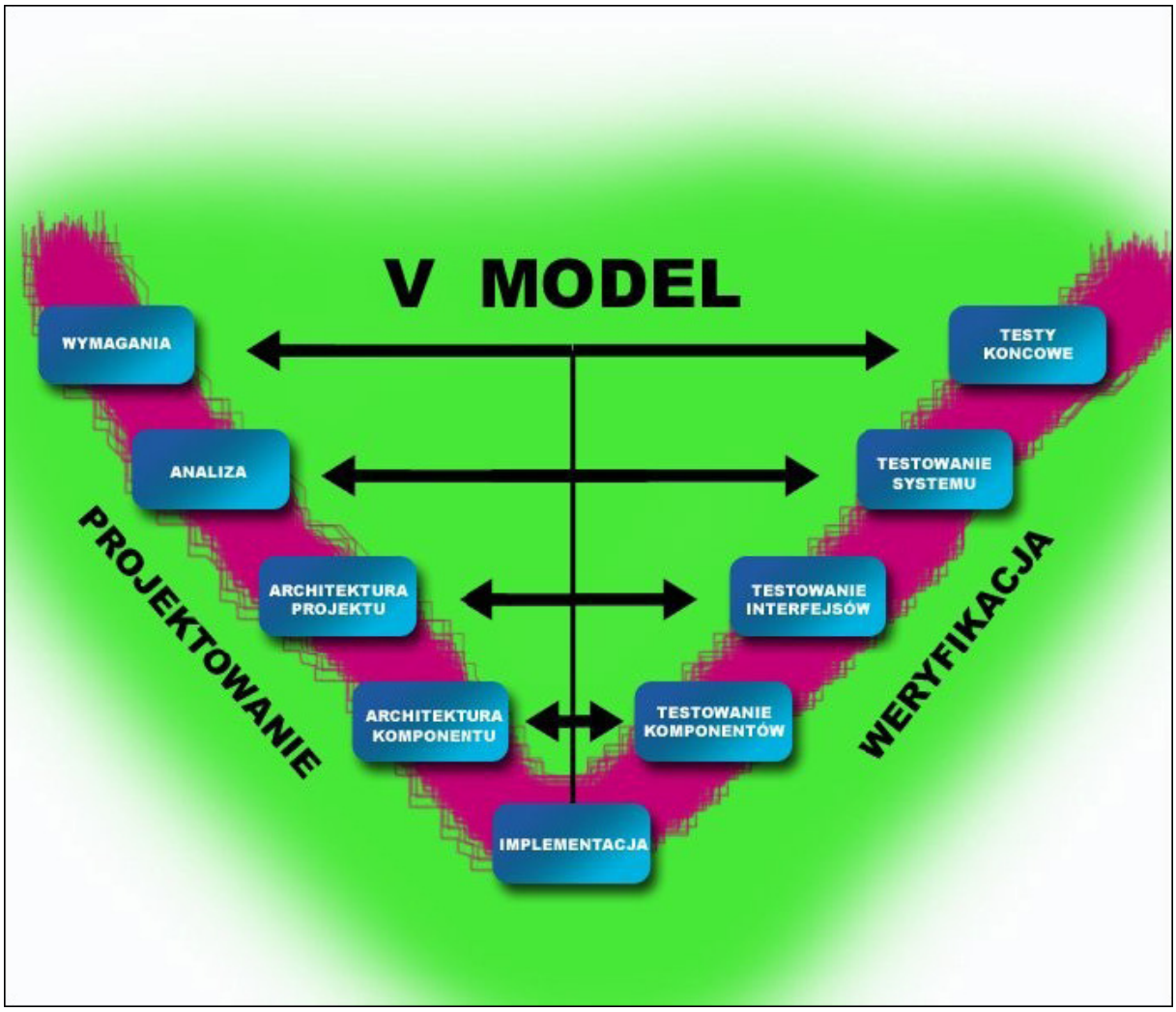
Wady:

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

Model V.

Odmiana modelu kaskadowego, definiuje główne procesy projektu jako etapy realizowane sekwencyjnie - jeden po drugim.

- Rozpoczęcie kolejnego etapu następuje po zakończeniu etapu poprzedniego.
- Podczas realizacji każdego z etapów (lewej strony) opracowuje się program metodyki i testy do badań jakości wyników etapu.
- Sprzężone są z nim procesy weryfikacyjne i walidacyjne, rozmieszczone na drugim ramieniu litery V, obrazują strukturę całego modelu cyklu wytwarzania oprogramowania.

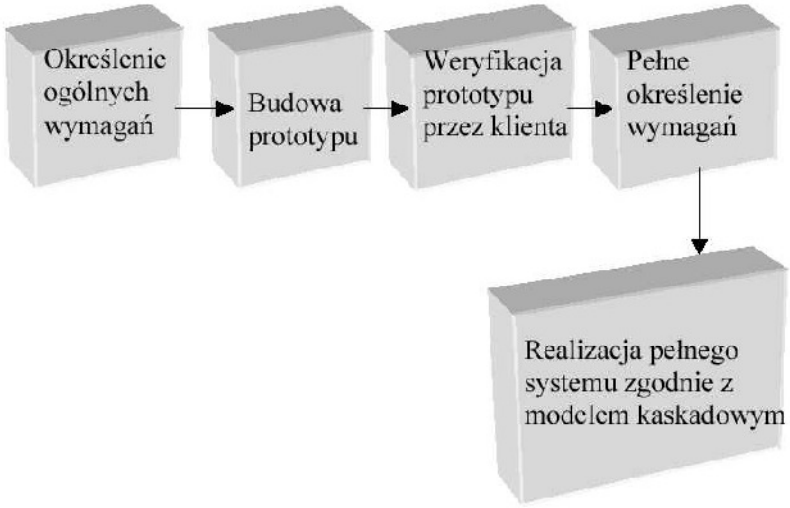


Zalety:

- ✓ Model V precyzyjnie pokazuje zależności, jakie powinny łączyć kolejne etapy projektu.
- ✓ Każdy etap projektowania kończy się stworzeniem danych wejściowych dla następnej fazy oraz do korespondującej z nim fazy weryfikacji.
- ✓ Zachęca do jak najwcześniejszego rozpoczęcia procesu tworzenia planów testów, specyfikacji testowej i samego testowania.
- ✓ Dzięki analizie grafu możemy łatwo zidentyfikować obszary, gdzie stworzona dokumentacja powinna być sprawdzona.

Ad. 2 Model prototypowy

- ➢ Stosowany, gdy są trudności w jednoznacznym ustaleniu wymagań.
- ➢ Jest to sposób na uniknięcie zbyt wysokich kosztów wskutek błędów popełnionych z powodu niedostatecznej znajomości pożądanych właściwości oprogramowania.



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

Ad. 3 Model RAD (Model szybkiej rozbudowy aplikacji)

Szczególna implementacja modelu kaskadowego.

- Dąży się do bardzo szybkiego stworzenia kompletnego produktu.

Fazy:

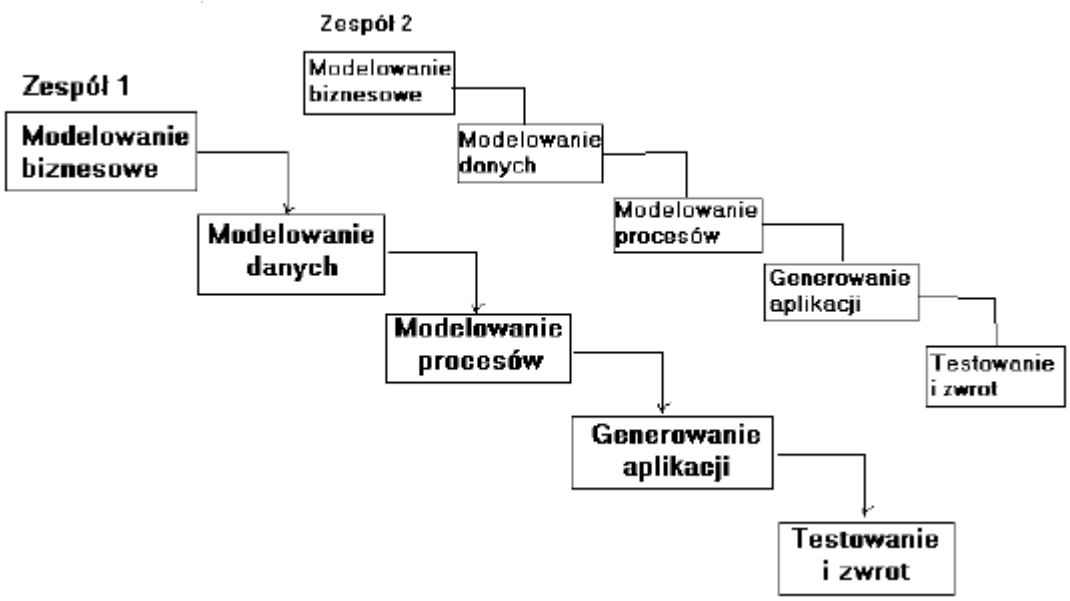
1. Modelowanie biznesowe:
   - opis przepływu informacji z i do projektowanego systemu
   - odp. Na pytania: jakie informacje sterują realizacją tego procesu, którego usprawnieniu posłuży system? Jakie informacje powstają w tym procesie i kto je tworzy? Gdzie te informacje są wysłane?

2. Modelowanie danych:
   - określenie postaci obiektów danych.
   - Dla każdego obiektu określa się zestaw jego charakterystycznych cech (atrybutów) i jego związki z innymi obiektami danych.

3. Modelowanie procesów:
   - Tworzenie procedur tworzenia, modyfikowania i usuwania dla każdego rodzaju obiektu.
4. Generowanie aplikacji:
   - Stosowanie technik czwartej generacji (nie używa się konwencjonalnych języków programowania trzeciej generacji)
   - Korzystanie z wcześniej stworzonych komponentów
   - Stosowanie zautomatyzowanych narzędzi programistycznych.
5. Testowanie i wdrożenie:
   - Testowanie jest skrócone ponieważ wykorzystywane są fragmenty oprogramowania, którego komponenty były już wcześniej przetestowane.

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

Ad. 4 Model przyrostowy.

Model ewolucyjny.

- Połączenie modelu liniowego z powtarzalnością obecną w modelu RAD

- Wielokrotne i w pewnym stopniu jednoczesne wykonywanie procesów liniowych.

- Każdy z procesów prowadzi do powstania pewnego rozszerzenia powstającego produktu.

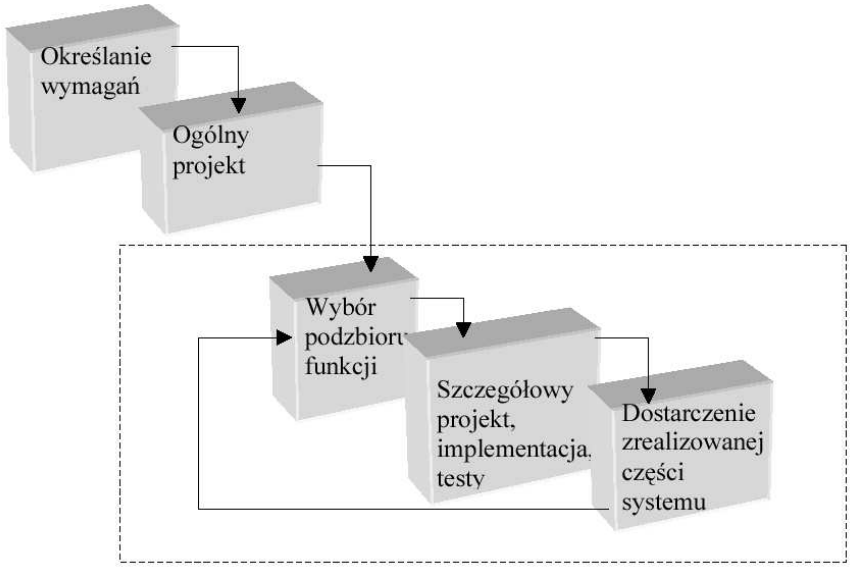
Różnica między prototypowaniem:

- nacisk na uzyskanie działającego produktu po przygotowaniu każdego rozszerzenia (prototyp jest z kolei atrapą).

Etapy:

- określenie ogólnych wymagań.
- wykonanie projektu całości z wyróżnieniem przyrostów funkcjonalności oprogramowania i określeniem kolejności ich realizacji.

- iteracyjna realizacja przyrostów właściwości użytkowych oprogramowania użytkowego (funkcjonalności) zgodnie z modelem kaskadowym.



Zalety:

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

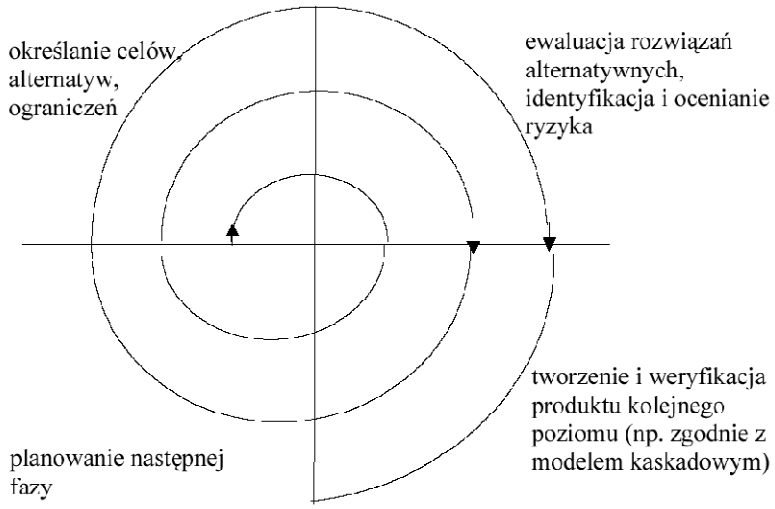
Wady:

- dodatkowy koszt towarzyszący niezależnej realizacji fragmentów systemu.

Ad.5 Model Spiralny.

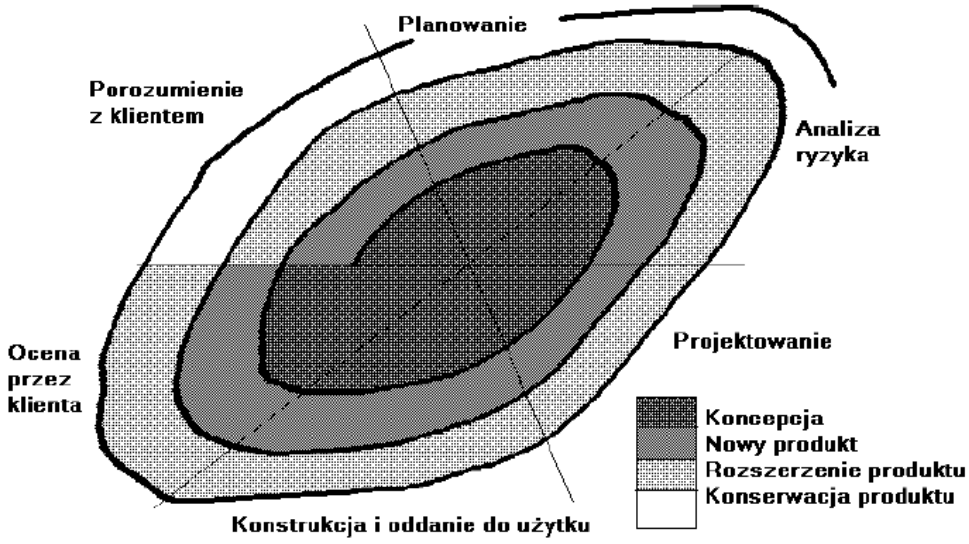
Model ewolucyjny.

- ➢ Połączenie powtarzalności z modelu partego o prototypowanie a systematycznością klasycznego modelu liniowego.
- ➢ Oprogramowanie jest wynikiem serii kolejnych rozszerzeń.
- ➢ Wersje z początkowych cykli mogą być modelami zapisanymi na papierze lub prototypami.
- ➢ W kolejnych cyklach powstają coraz bardziej kompletne wersje działającego systemu.



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.



- Każde przejście przez strefę planowania wiąże się z uzupełnianiem i modyfikowaniem planu, harmonogramu, budżetu na podstawie informacji uzyskanych dzięki współpracy z klientami.
- W odróżnieniu od modeli klasycznych, które przewidują zakończenie procesu wytwórczego w chwili przekazania produktu klientom, model spiralny obejmuje cały czas istnienia oprogramowania.

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

Ad. 6. Model równoległy.

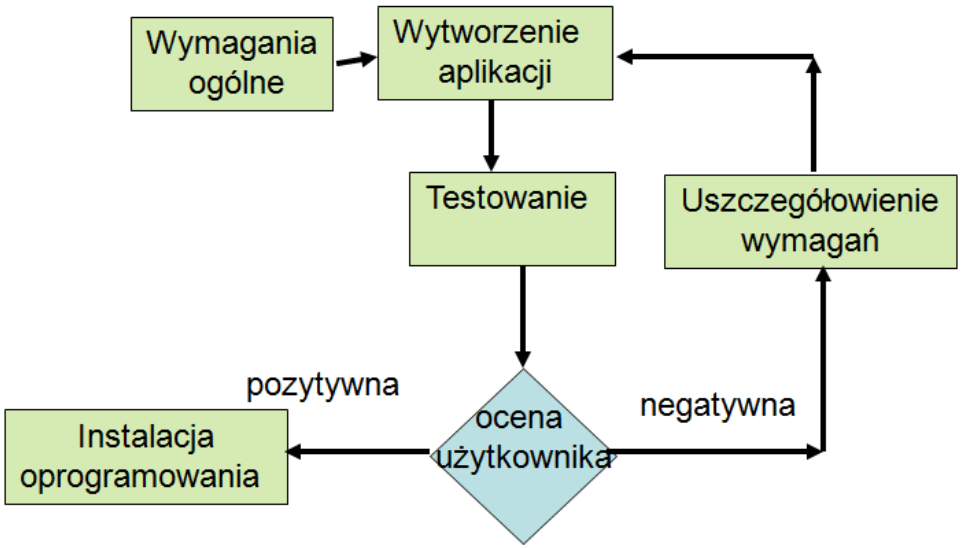
Model ewolucyjny.

- Stosowany do wytwarzania systemów o architekturze klient-serwer
- Praca równoległa nad kilkoma komponentami
- Proces wytwórczy traktowany jako sieć zadań połączonych zależnościami, a nie jako ciąg zdarzeń.
- Każde zadanie można wykonywać równolegle z innymi zadaniami.
- Wykonanie niektórych zadań wiąże się z powstaniem zdarzeń, które powodują zmiany stanów procesu wykonywania innych zadań.
- Zadania mają swoje określone stany (np. w przygotowaniu, oczekiwanie na zmiany, poprawki, wykonano itp.)
- Model równoległy obejmuje opis zestawu zdarzeń, które powodują zmiany stanów.

Ad. 7 Programowanie odkrywcze.

Model stosowany gdy są trudności w ustaleniu nawet wstępnych wymagań – gdy zastosowanie prototypowania nie pozwala ustalić wymagań na oprogramowanie.

- Ustalane są bardzo ogólne wymagania – jednakże o maksymalnym, możliwym do określenia, stopniu szczegółowości. Po czym budowany jest projekt i jego implementacja programowa, która udostępniana jest użytkownikowi do oceny.
- Uwagi użytkownika stanowią podstawę do uściślenia wymagań i powtórzenia cyklu.



Ad. 8 Tworzenie oprogramowania opartego na komponentach.

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

- Podobny do spiralnego – ewolucyjny i przewiduje realizację powtarzalnych sekwencji czynności. (Fazy jak w spiralnym)
- Oprogramowanie składane z gotowych komponentów programowych.

Metody:

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

Faza inżynierii (konstrukcji w spiralnym) rozpoczyna się od określenia klas, które mogą wejść w skład produktu.
- Definiowanie danych, które będą przetwarzane w produkcie i algorytmów, które będą zastosowane w tym celu.
Struktury danych i metody operowania na nich stanowią klasę. Klasy przygotowane już dla wcześniejszych produktów przechowuje się w bibliotece klas lub w repozytorium. Po określeniu, które klasy mogą wejść w skład produktu, szuka się ich w bibliotece klas. Jeśli tam są, to pobiera się je do ponownego użycia.
Faza inżynierii:
Identyfikacja dopuszczalnych komponentów -> Sprawdzanie dostępności komponentów w bibliotece -> wybranie dostępnych komponentów -> Dodawanie nowych komponentów do biblioteki -> Konstrukcja n-tej iteracji systemu.

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.

Podsumowanie zasad wyboru modelu:

- Kiedy na własny użytek chcemy rozwiązać pewien niewielki problem i nie wiemy, jak się za niego zabrać -> model pisz-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 klient ma chwilowo małe fundusze, ale jest szansa, że w trakcie realizacji budżet się zwiększy -> model spiralny.
- W każdym możliwym przypadku budować z gotowych elementów.
- W przypadku wyboru dowolnego modelu, jeśli wymagania są źle określone -> zrobić prototyp