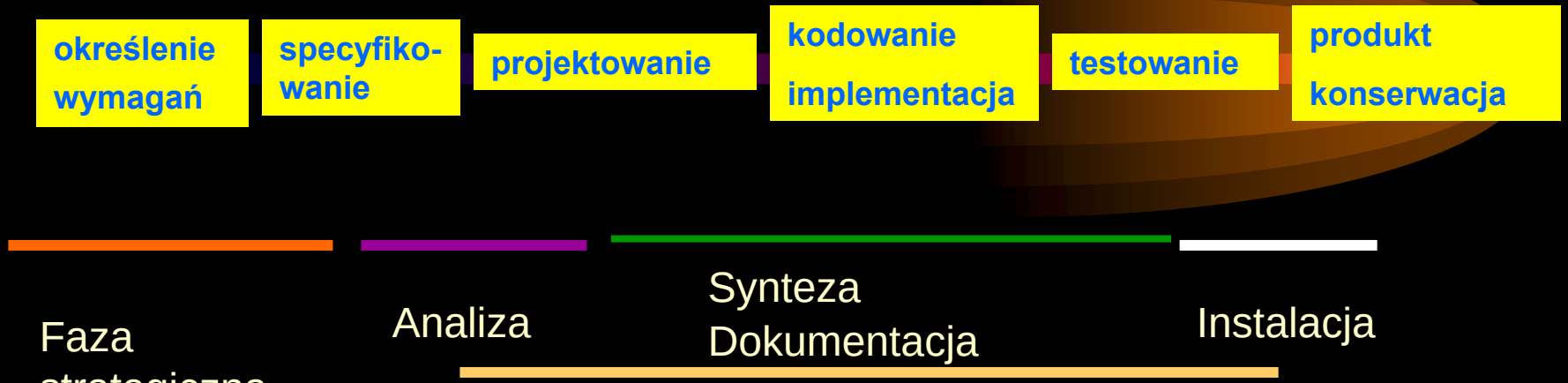
Błędy procesu tworzenia oprogramowania

(Badania firmy Rational Software Corporation)

- Zarządzanie wymaganiami *Ad hoc* (najczęściej brak zarządzania nimi)
- Niejednoznaczna, nieprecyzyjna komunikacja
- Architektura oprogramowania nieodporna na obciążenia (ang. Brittle architecture)
- Zbytnia, niepotrzebna złożoność oprogramowania
- Niewykryte niespójności w wymaganiach, projekcie, oraz implementacji
- Brak, lub niewystarczające testowanie
- Subiektywna ocena postępu projektu
- Brak zarządzania ryzykiem, brak atakowania ryzyka
- Brak automatyzacji prowadzenia projektu

! w każdym projekcie inny zestaw czynników niepowodzenia
! brak stosowania standaryzacji wymaganej przez UML

FAZA STRATEGICZNA



- rozmowy z klientami (przedstawicielami)
- cele przedsięwzięcia z pkt. widzenia klienta
- zakres i kontekst przedsięwzięcia
-ogólne określenie wymagań – wykonanie wstępnej analizy i projektu systemu
-propozycja kilku sposobów realizacji systemu
- szacowanie kosztów oprogramowania
-analiza rozwiązań
-prezentacja wyników analizy klientowi – korekta
-budowa wstępnego harmonogramu
-określenie standardów wg których będzie realizowane przedsięwzięcie

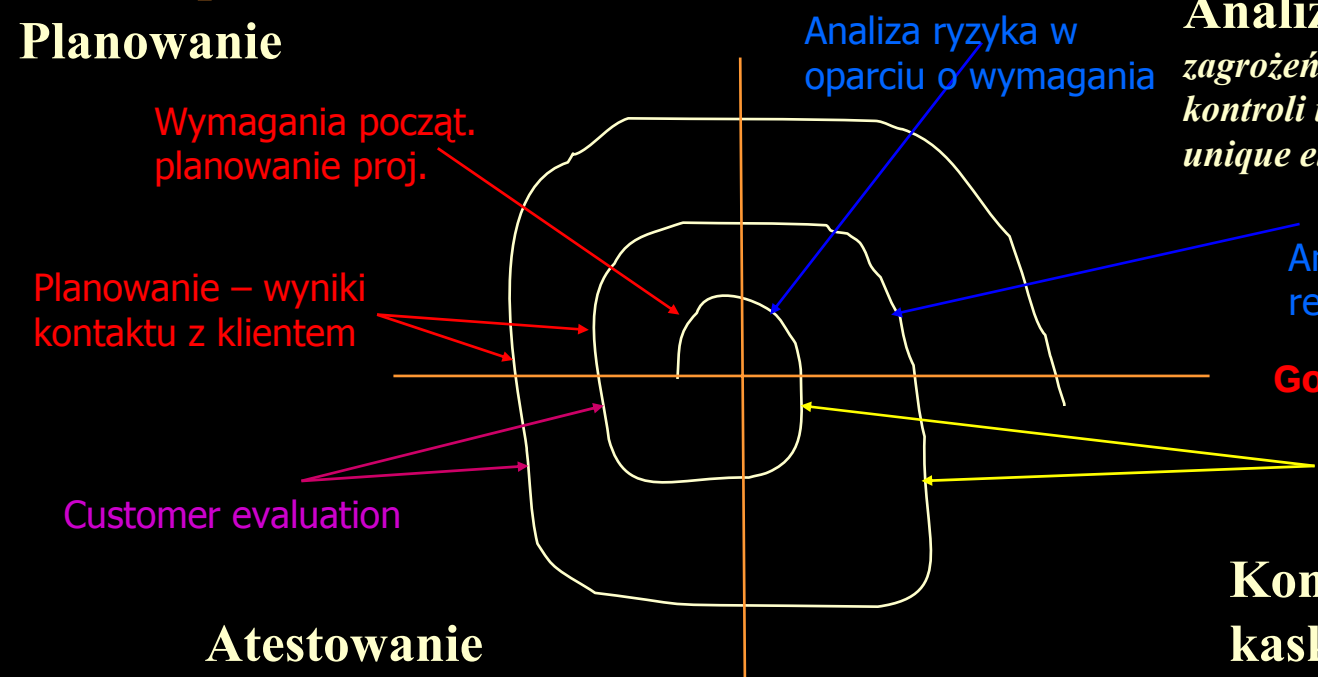
MODELE CYKLU ŻYCIA OPROGRAMOWANIA

Model spiralny – model interaktywny

(Boehma – 1988) – cztery główne fazy cyklu życia oprogramowania wykonywane cyklicznie: analiza ryzyka, konstrukcja, atestowanie, planowanie.

Planowanie

Analiza ryzyka ***(identyfikacja zagrożeń wewnętrznych – z możliwością kontroli i zewnętrznych bez kontroli - unique element)***



Konstrukcja (model kaskadowy)

Atestowanie

Model spiralny jest raczej tzw modelem referencyjnym albo inaczej metamodelem dla innych model
Model spiralny jest tak naprawdę „sposobem myślenia „ o tworzeniu oprogramowania na rynek

Baza RUP

(Przyczyny powstania Najczęstsze błędy)

- Zarządzanie wymaganiami *Ad hoc* (najczęściej brak zarządzania nimi)
- Niejednoznaczna, nieprecyzyjna komunikacja
- Architektura oprogramowania nieodporna na obciążenia (ang. Brittle architecture)
- Zbytnia, niepotrzebna złożoność oprogramowania
- Niewykryte niespójności w wymaganiach, projekcie, oraz implementacji
- Brak, lub niewystarczające testowanie
- Subiektywna ocena postępu projektu
- Brak zarządzania ryzykiem, brak atakowania ryzyka
- Brak automatyzacji prowadzenia projektu

Baza RUP

(Zasady IO + najlepsze praktyki programistyczne)

- Iteracyjnym wytwarzaniu oprogramowania (Iterative Development)
- Zarządzaniu wymaganiami (Requirement Management)
- Używaniu architektury bazującej na komponentach (Component-based architecture)
- Graficznym projektowaniu oprogramowania
- Kontroli jakości oprogramowania (Quality Assurance)
- Procesu kontroli zmian w oprogramowaniu (Change Management)

Iterative development

(Zasady IO + najlepsze praktyki programistyczne)

Przyczyny zmiany wymagań: - ograniczenia architektury
- zmiany potrzeb użytkowników
- lepsze zrozumienie problemu

Dlatego podejście iteracyjne i przyrostowe pozwalające na

- Integracja oprogramowania robiona krok po kroku podczas wytwarzania oprogramowania, ograniczając go do mniejszej liczby elementów
- Integracja jest prostsza i mniej kosztowna
- Składowe oprogramowania są projektowane oddzielnie i łatwiej poddają się reużywalności
- Łatwiej wykrywać zmiany wymagań i łatwiej nimi zarządzać
- Ryzyka identyfikowane i atakowane są wcześnie ponieważ każda iteracja pozwala wykryć kolejne ryzyka
- W iteracjach ulepszana jest architektura oprogramowania

MODELE CYKLU ŻYCIA OPROGRAMOWANIA

Rational Unified Process (RUP-1998) *(2000 – IBM)*

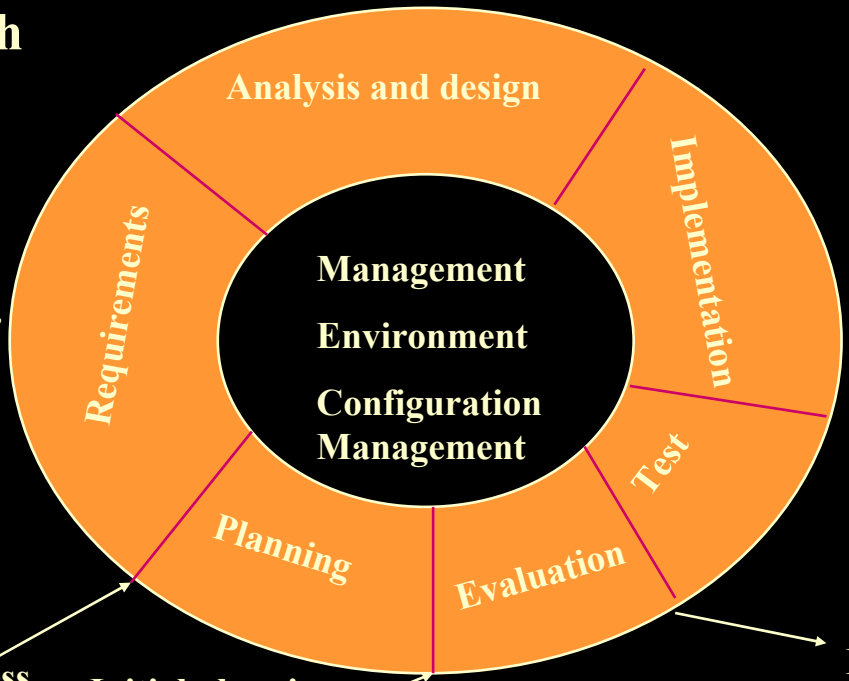
(IBM – 2003) – coś więcej niż model cyklu życia oprogramowania. Środowisko *(RUP platform)* dla programistów obejmujące szereg narzędzi asystujących i prowadzących twórców programowania w dwóch wymiarach

WYMIAR WERTYKALNY

sześć przepływów aktywności:

- ***modelowanie biznesowe***
- ***identyfikacja wymagań***
- ***analiza i projektowanie***
- ***implementacja***
- ***testowanie***
- ***osadzenie systemu***

trzy wspomagające:



WYMIAR HORYZONTALNY (*aspekt dynamiczny cyklu – upływ czasu):*

Inception - ***powstanie, rozpoczęcie (analiza wykonalności)***

Elaboration – ***szczegółowe dopracowanie (analiza dziedziny problemu)***

Construction – ***budowa, konstrukcja***

Transition – ***przejście, (przekazanie systemu)***

Metodyka wytwarzania oprogramowania – proces iteracyjnego wytwarzania oprogramowania