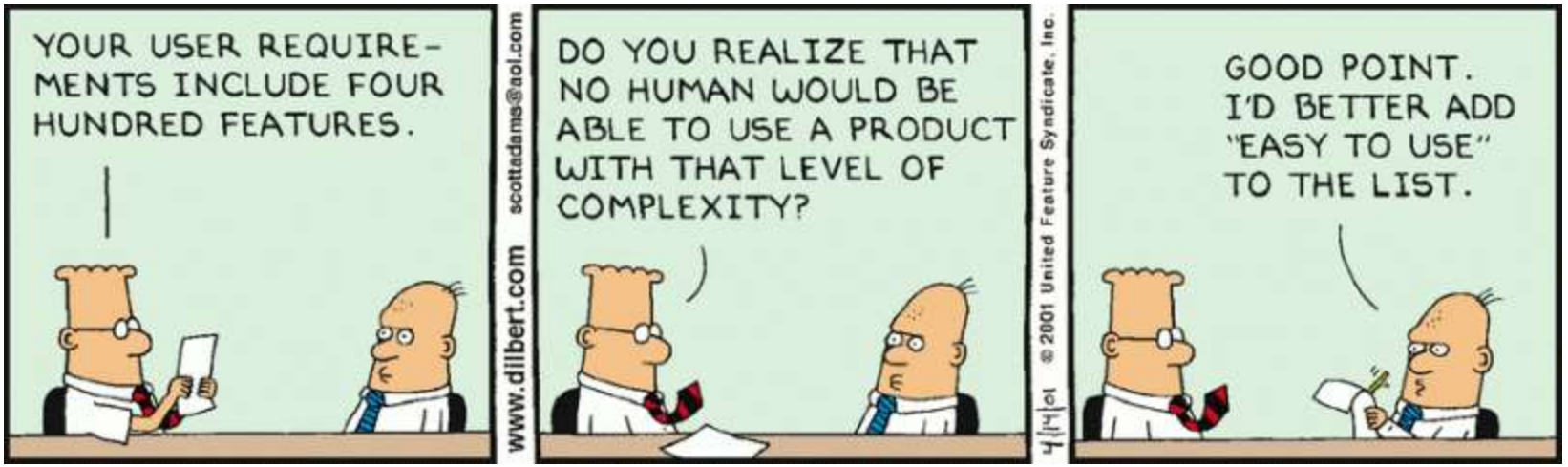
Inżynieria Oprogramowania

Proces Inżynierii Oprogramowania

Niezrozumienie potrzeb użytkowników



Nieumiejętne reagowanie na zmiany w wymaganiach

Problemy z integracją

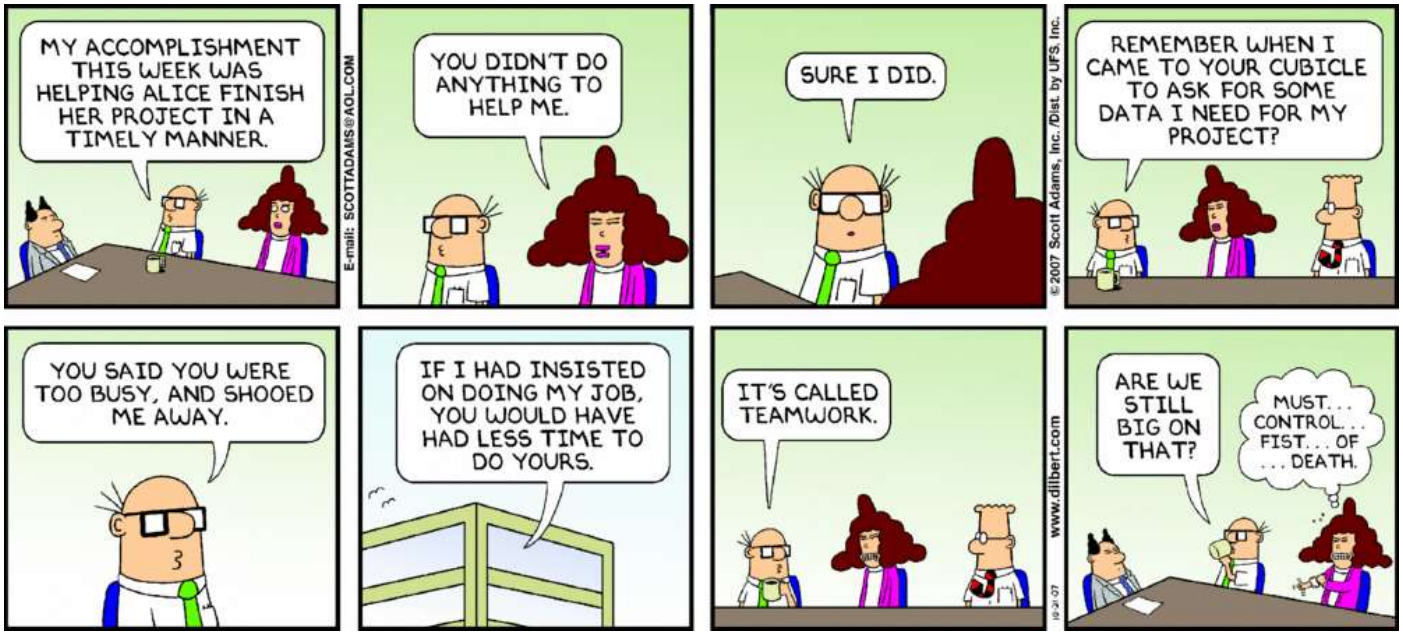
```
Module build failed (from ./node_modules/react-scripts/node_modules/babel-loader/lib/index.js):
SyntaxError: /Users/jp.marra/Sites/tutorials/learnstorybook-design-system/src/components/Avatar.stories.mdx: Unexpected token (14:9)

  12 | const makeShortcode = name => function MDXDefaultShortcode(props) {
  13 |   console.warn("Component " + name + " was not imported, exported, or provided by MDXProvider as global scope")
> 14 |   return <div {...props}/>
     |          ^
  15 | };
  16 |
  17 | const layoutProps = {
    at Parser.raise (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:6400:17)
    at Parser.unexpected (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:7728:16)
    at Parser.parseExprAtom (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8940:20)
    at Parser.parseExprSubscripts (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8507:23)
    at Parser.parseMaybeUnary (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8487:21)
    at Parser.parseExprOps (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8353:23)
    at Parser.parseMaybeConditional (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8326:23)
    at Parser.parseMaybeAssign (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8273:21)
    at Parser.parseExpression (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:8221:23)
    at Parser.parseReturnStatement (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:10301:28)
    at Parser.parseStatementContent (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:9980:21)
    at Parser.parseStatement (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:9932:17)
    at Parser.parseBlockOrModuleBlockBody (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:10508:25)
    at Parser.parseBlockBody (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:10495:10)
    at Parser.parseBlock (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:10479:10)
    at Parser.parseFunctionBody (/Users/jp.marra/Sites/tutorials/learnstorybook-design-system/node_modules/@babel/parser/lib/index.js:9523:24)
```

Trudności z konserwacją systemu

Późne wykrywanie błędów, słaba jakość

Problemy z pracą zespołową



Proces Inżynierii Oprogramowania

- Ustrukturalizowany zbiór czynności wymaganych do opracowania oprogramowania.
- W użyciu jest wiele różnych procesów, ale wszystkie uwzględniają czynności:
  - **Specyfikacja** – definiowanie tego co system powinien realizować;
  - **Rozwój** – definiowanie organizacji systemu oraz implementacja systemu;
  - **Zatwierdzanie** – sprawdzanie czy system wykonuje to czego oczekuje klient;
  - **Ewolucja** – modyfikowanie systemu w odpowiedzi na zmiany potrzeb klienta.
- Model procesu Inżynierii Oprogramowania
  - abstrakcyjna reprezentacja procesu.
  - Opis procesu z określonej perspektywy.

Procesy sterowanie planem i procesy zwinne

- Procesy sterowane planem to procesy, gdzie wszystkie czynności procesu są zaplanowane. Pomiary postępów prac odnoszone są do tego planu.

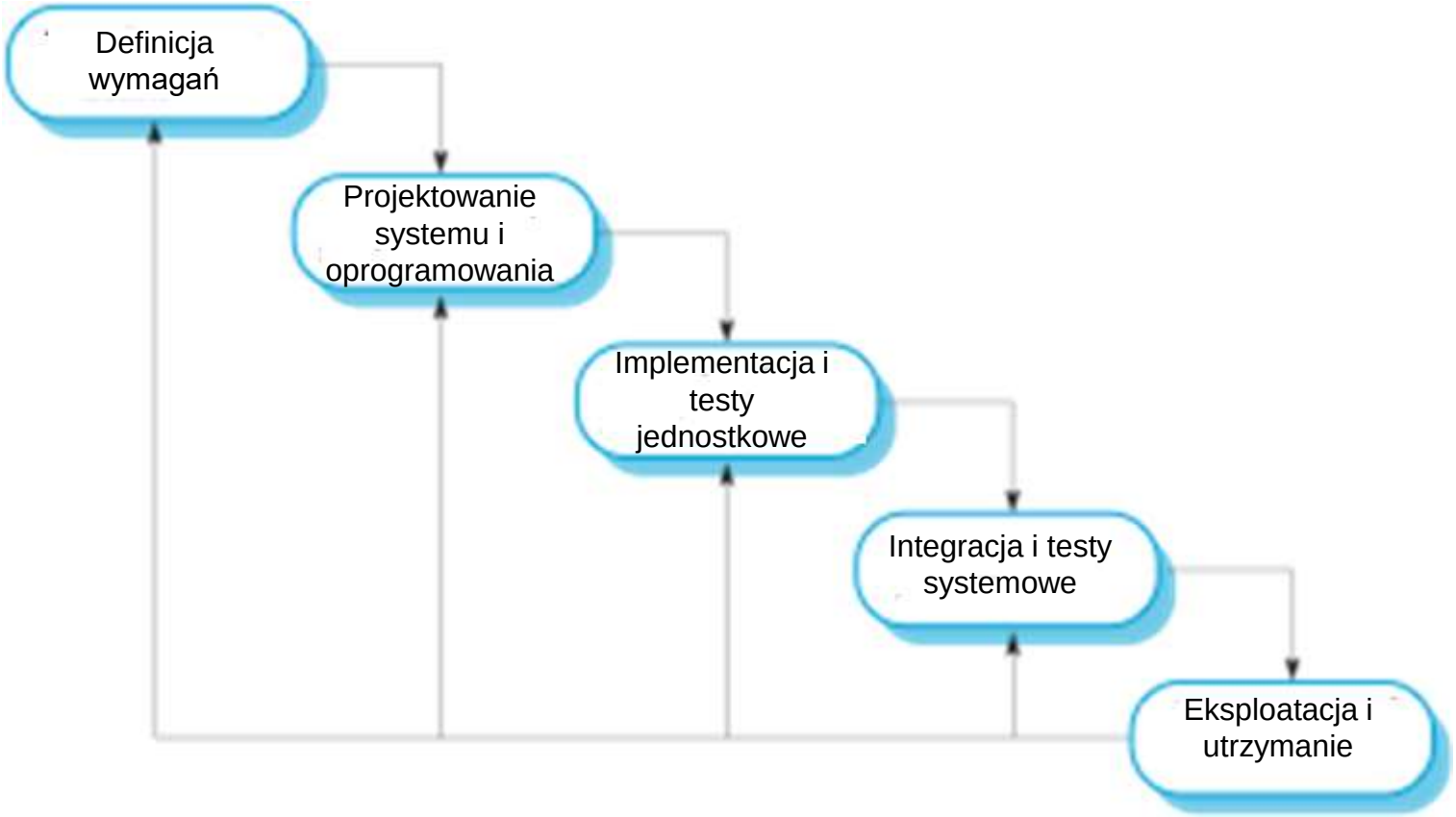
- W procesach zwinnych planowanie jest przyrostowe, co powoduje, że łatwiej jest zmienić proces w odpowiedzi na zmianę wymagań.

Modele (paradygmaty) procesu Inżynierii Oprogramowania

1. Model kaskadowy (The waterfall model)
   - Model sterowany planem. Oddzielne fazy specyfikacji i rozwoju.
2. Realizacja przyrostowa (Incremental development)
   - Specyfikacja, wytwarzanie i zatwierdzanie są przeplatane. Może być sterowana planem lub zwinna.
3. Inżynieria zorientowana na wielokrotne użycie (Reuse-oriented software engineering)
   - System jest składany z istniejących komponentów. Może być sterowana planem lub zwinna.

- W praktyce większość systemów wytwarza się z wykorzystaniem procesów, które wykorzystują elementy wszystkich tych modeli.

Model kaskadowy



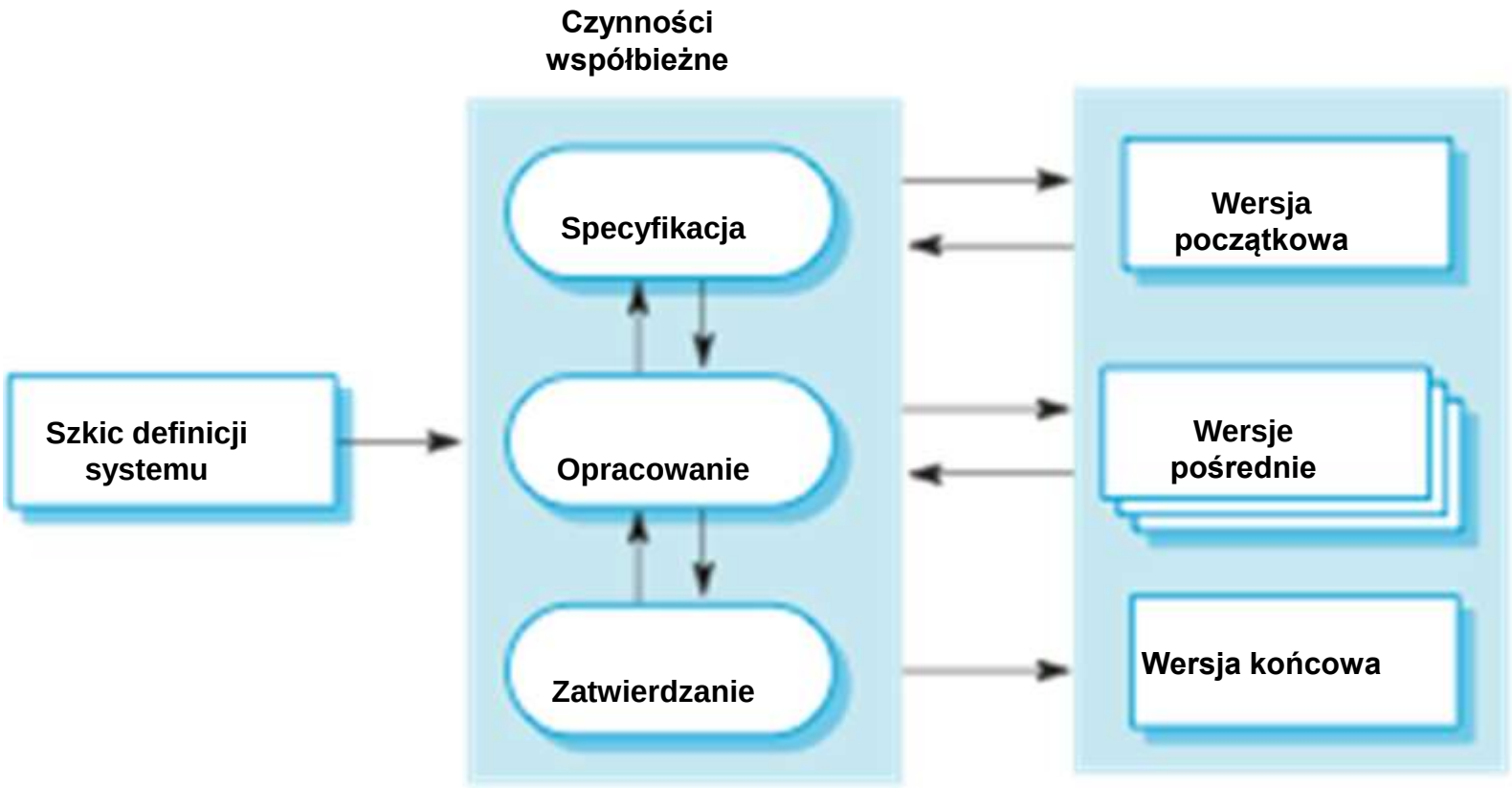
Fazy modelu kaskadowego

- Model kaskadowy składa się z wydzielonych faz:
  - Analiza i definiowanie wymagań
  - Projektowanie systemu i oprogramowani
  - Implementacja i testy jednostkowe
  - Integracja i testy systemowe
  - Eksploatacja i utrzymanie
- Podstawową wadą modelu jest trudność w adaptacji zmian pojawiających się w czasie realizacji procesu. Z zasady, faza musi być zakończona zanim rozpocznie się kolejna.

Model kaskadowy - dyskusja

- Brak elastyczności poprzez podział projektu na oddzielne etapy powoduje trudności w sytuacji, gdy pojawią zmiany wymagań.
  - Z tego powodu model ten jest odpowiedni w przypadkach gdy wymagania są dobrze zrozumiane a zmiany w trakcie procesu są mocno ograniczone.
  - Niewiele systemów biznesowych posiada stabilne wymagania.
- Model kaskadowy jest wykorzystywany dla realizacji dużych systemów, gdzie prace odbywają się w kilku miejscach.
  - Sterowana planem, natura modelu kaskadowego, pomaga w koordynacji prac.

Realizacja przyrostowa



Zalety realizacji przyrostowej

- Redukcja kosztów adaptacji zmian w wymaganiach.
  - Zakres analizy i dokumentacji, która musi zostać powtórnie wykonana jest zdecydowanie mniejsza w porównaniu z modelem kaskadowym.
- Łatwiej jest uzyskać informację zwrotną na temat dotychczas wykonanych prac.
  - Klienci mogą wypowiedzieć się na temat oprogramowania na podstawie prezentacji dotychczas zrealizowanego zakresu implementacji.
- Możliwe jest szybsze dostarczenie i wdrożenie użytecznego oprogramowania.
  - Klienci mogą wykorzystać oprogramowanie wcześnie w stosunku do tego co oferuje model kaskadowy.

Problemy realizacji przyrostowej

- Niejawność procesu
  - Klasyczne zarządzanie projektem wymaga regularnych rezultatów, pozwalających na pomiar postępów. Jeżeli system jest wytwarzany szybko nie opłaca się produkować dokumentacji odzwierciedlającej każdą wersję systemu.
- Struktura systemu ma tendencję do degradacji wraz z dodawaniem nowych przyrostów.
  - Jeżeli nie zostaną poświęcone dodatkowe środki oraz czas na restrukturyzację oprogramowania, regularne zmiany ostatecznie doprowadzą do zniszczenia struktury oprogramowania. Wprowadzanie kolejnych zmian stanie coraz bardziej kosztowne i skomplikowane.

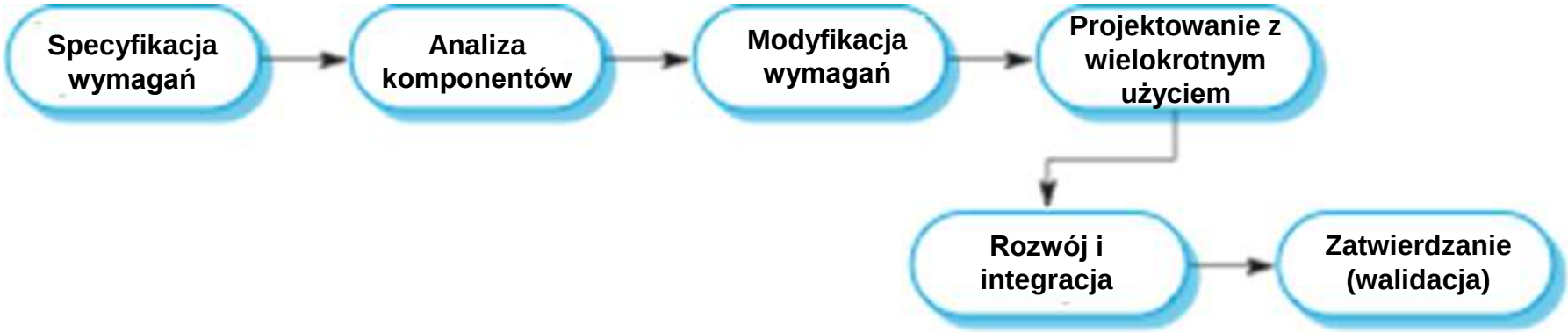
Inżynieria Oprogramowania zorientowana na wielokrotne użycie

- Opiera się na systematycznym podejściu do wykorzystania istniejącego oprogramowania. System integrowany jest z istniejących komponentów systemów COTS (Commercial-off-the-shelf) lub usług.

- Fazy procesu:
  - Analiza komponentów/usług;
  - Modyfikacja wymagań;
  - Projektowanie z wielokrotnym użyciem;
  - Rozwój i integracja.

- Wielokrotne użycie jest obecnie standardowym podejściem do budowania wielu typów systemów biznesowych.

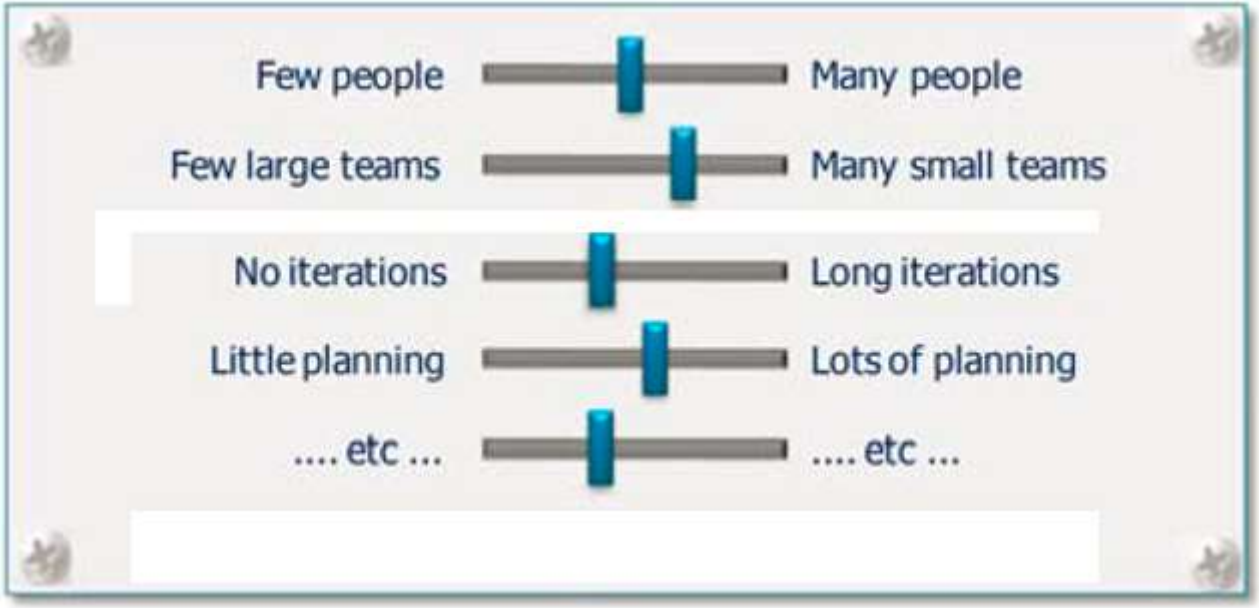
Inżynieria Oprogramowania zorientowana na wielokrotne użycie



Typy komponentów oprogramowania

- Usługi internetowe (ang. web services) wytwarzane zgodnie ze standardami usług, wywoływane zdalnie.
- Kolekcje obiektów wytwarzane w postaci pakietów, które integrowane są w ramach implementacji modelu komponentowego (frameworka) np.: .NET, Java EE.
- Autonomiczne systemy oprogramowania (COTS) konfigurowane dla działania w określonym środowisku.

Podejście empiryczne

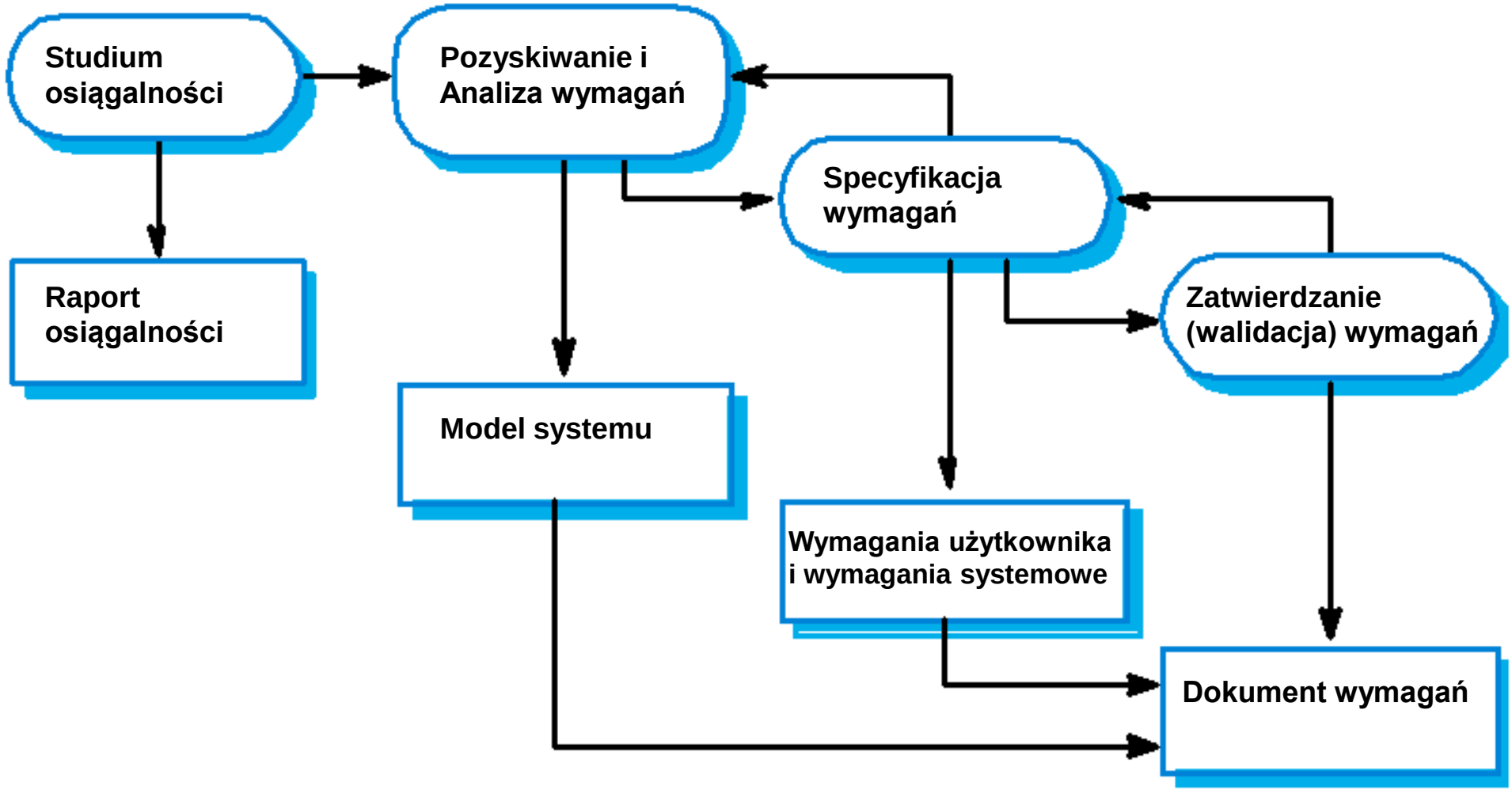


Aktywności procesu

- Rzeczywiste procesy IO to sekwencje przeplatających się zespołowych czynności technicznych i zarządczych. Ich ogólnym celem jest wyspecyfikowanie, zaprojektowanie, implementacja oraz przetestowanie systemu oprogramowania.

- Cztery podstawowe aktywności procesu:
  - Specyfikacja, rozwój, zatwierdzanie (walidacja), ewolucja
  - Są organizowane różnie, w zależności od procesu (sekwencyjnie, przeplatające się).

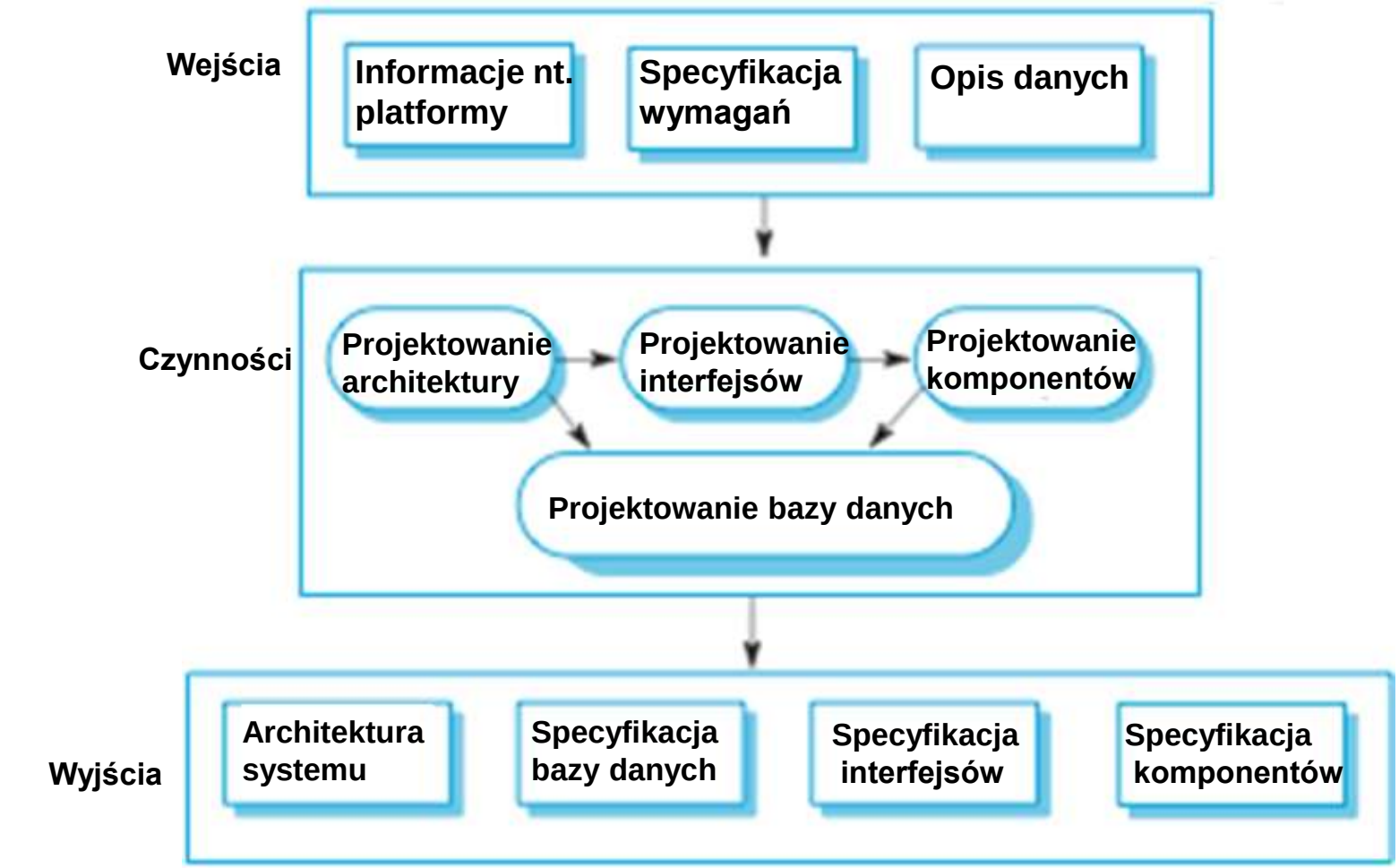
Specyfikacja oprogramowania



Projektowanie i implementacja oprogramowania

- Proces, którego celem jest przekształcenie specyfikacji w działający system
- Projektowanie
  - Określenie struktury oprogramowania, która realizuje specyfikacje.
- Implementacja
  - Translacja struktury na wykonywalny program
- Projektowanie i implementacja są ze sobą mocno powiązane i często się przeplatają.

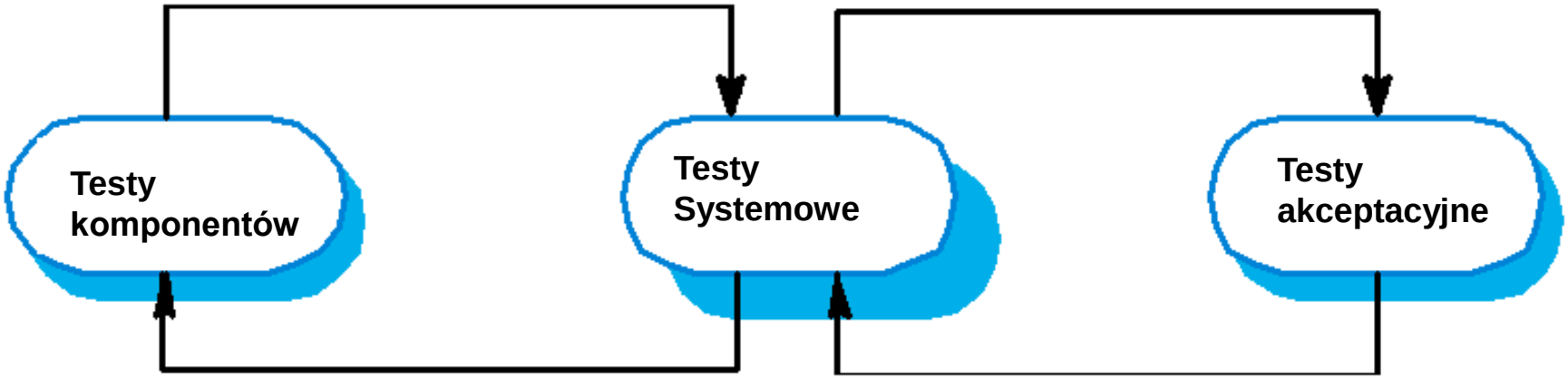
Ogólny model procesu projektowania oprogramowania



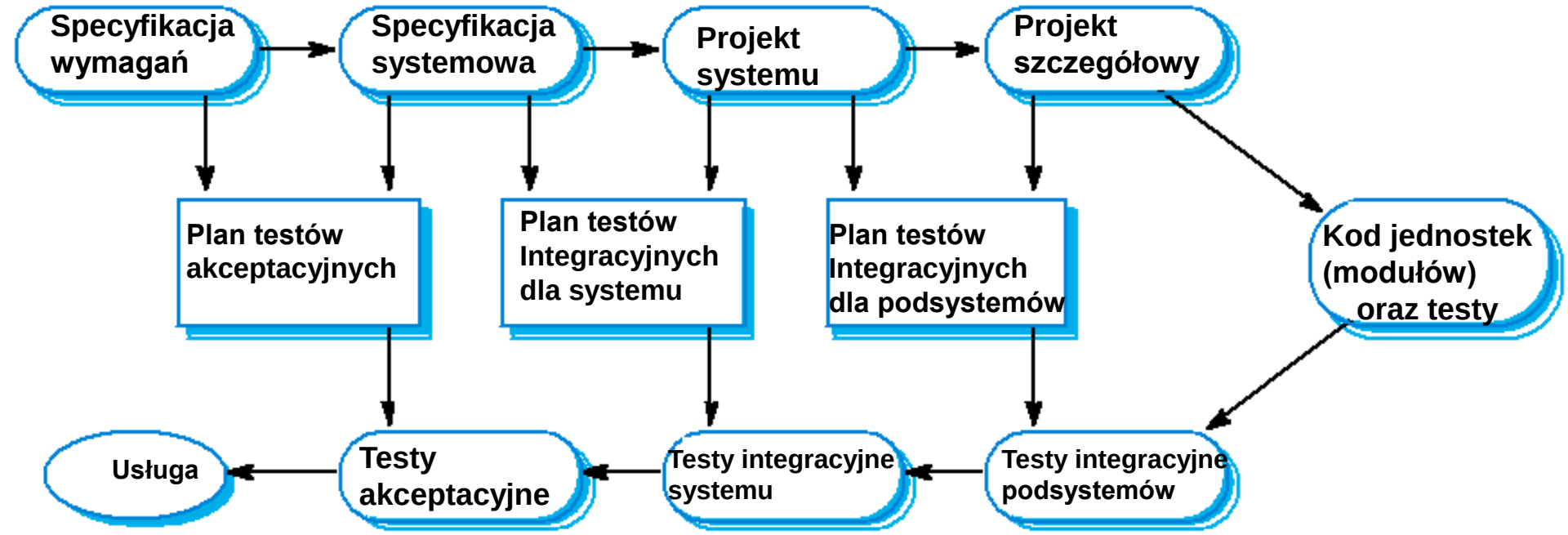
Zatwierdzanie (walidacja) oprogramowania

- Celem weryfikacji i zatwierdzania jest wykazanie, że system jest zgodny ze specyfikacją i spełnia wymagania klienta systemu.
- Obejmuje procesy kontroli i przeglądów oraz testowania systemu.
- Testowanie systemu to uruchomienie go i wykonanie przypadków testowych, które wywodzą się ze specyfikacji rzeczywistych danych, które będą przetwarzane przez system.
- Testowanie jest najczęściej wykorzystywaną techniką w ramach czynności weryfikacji i zatwierdzania.

Etapy testowania



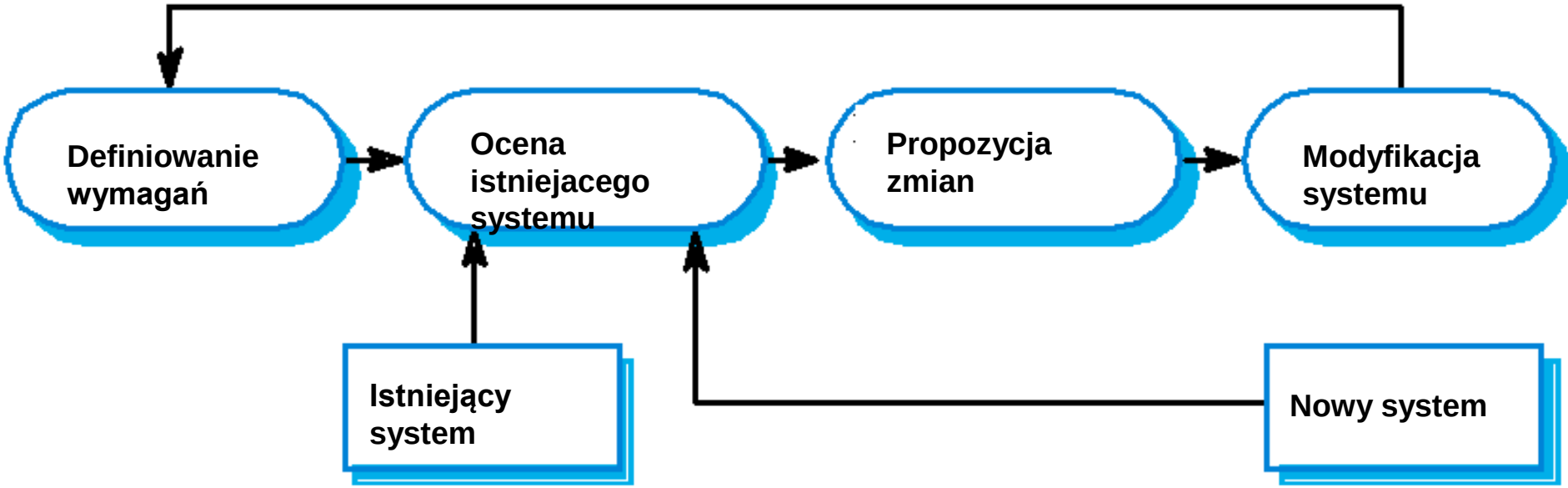
Fazy testowania w procesie IO sterowanym planem.



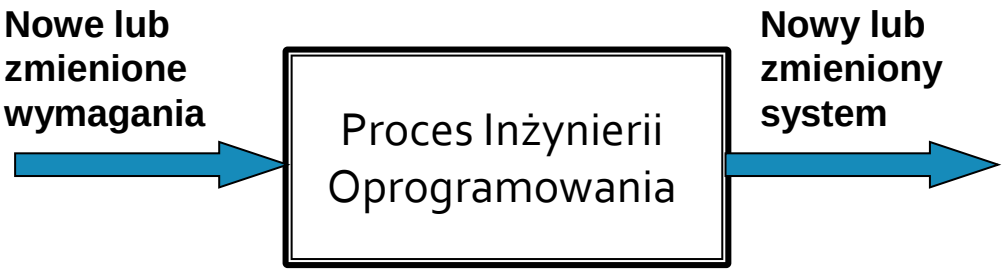
Ewolucja oprogramowania

- Oprogramowanie musi być elastyczne bo zmiany są nieuniknione
  - Zmieniają się uwarunkowania biznesowe -> zmieniają się wymagania -> ewoluuje system
- Pomimo pierwotnego wyróżnienia procesu budowy i ewolucji systemu granica ta dzisiaj coraz bardziej się zaciera (coraz mniej systemów jest zupełnie nowych).

Wytwarzanie przez ewolucję



Czyli po prostu…



Podsumowanie

- Procesy IO to czynności obejmujące produkcję oprogramowania. Modele procesów IO to abstrakcyjne ich reprezentacje.
- Ogólne modele procesu opisuję podstawową organizację procesu IO. Przykładami są: model kaskadowy, realizacja przyrostowa oraz wytwarzanie zorientowane na powtórne użycie.

Podsumowanie

- Inżynieria wymagań to proces rozwoju specyfikacji oprogramowania.
- Projektowanie i implementacja związane są z transformacją specyfikacji wymagań w wykonywalne oprogramowanie.
- Zatwierdzanie oprogramowania to proces kontroli, której celem jest sprawdzenie czy system jest zgodny ze specyfikacją i czy spełnia rzeczywiste potrzeby użytkowników systemu.
- Ewolucja oprogramowania to dostosowywanie istniejącego oprogramowania do nowych wymogów. Aby pozostać użyteczne – oprogramowanie musi ewoluować.

Inżynieria oprogramowania

Proces inżynierii oprogramowania (część 2) (aspekt zmiany …)

Radzenie sobie ze zmianą

- Zmiany są nieuniknione we wszystkich dużych projektach oprogramowania.
  - Zmiany biznesowe powoduję powstawianie nowych i zmianę istniejących wymagań
  - Owe technologie dają możliwość udoskonalania implementacji
  - Zmieniające się platformy wymuszają zmiany w aplikacjach

- Zmiany prowadzą do ponownego wykonania pewnych prac (np. ponowna analiza istniejących wymagań) oraz zaimplementowania nowych funkcjonalności. Wszystko to składa się na koszty zmian.

Redukcja kosztów modyfikacji

- **Unikanie zmian** - proces IO może obejmować działania, których celem jest przewidzenie ewentualnych zmian zanim spowodują one potrzebę znaczących przeróbek.
  - Na przykład można opracować prototyp systemu w celu zaprezentowania klientom kluczowych cech systemu.

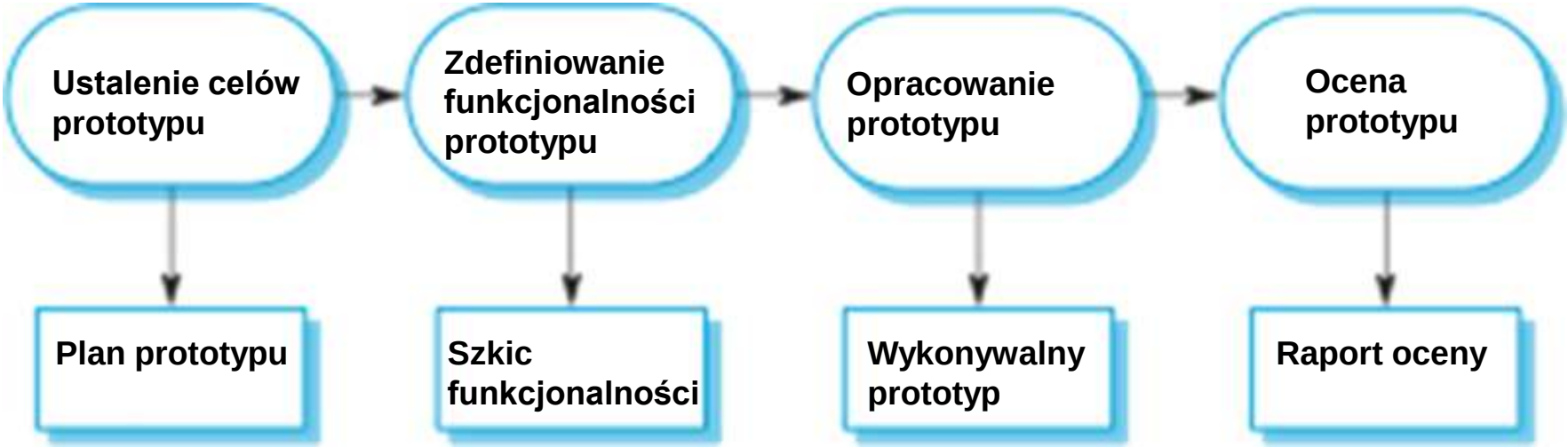
- **Tolerowanie zmian** – proces IO jest skonstruowany w taki sposób aby zmiany mogły być wprowadzone przy relatywnie niskim koszcie realizacji.
  - Takie podejście wymaga zazwyczaj jakiejś formy wytwarzania przyrostowego - iteracyjności. Proponowane zmiany mogą być wprowadzanie w ramach przyrostów, które nie zostały jeszcze zrealizowane. Jeżeli jest to niemożliwe wówczas zmiana może być uwzględniona poprzez modyfikację pojedynczego przyrostu (małego fragmentu systemu).

Prototypowanie

- Prototyp to początkowa wersja systemu, której celem jest zademonstrowanie jego koncepcji oraz wypróbowanie wariantów projektowych.

- Prototyp może być wykorzystany w:
  - Procesie inżynierii wymagań jako pomoc w pozyskiwaniu oraz zatwierdzaniu wymagań
  - Procesie projektowym w celu rozpatrzenia możliwych rozwiązań oraz rozwoju projektu interfejsu użytkownika.

Model procesu rozwoju prototypu



Rozwój prototypu

- Może wykorzystywać dedykowane języki i narzędzia
- Jest z założenia niekompletny
  - Prototyp powinien koncentrować się na nie do końca zrozumianych obszarach produktu,
  - Prototyp może pomijać kontrolę błędów.
  - Kwestie niezawodności czy bezpieczeństwa nie są zazwyczaj brane pod uwagę.

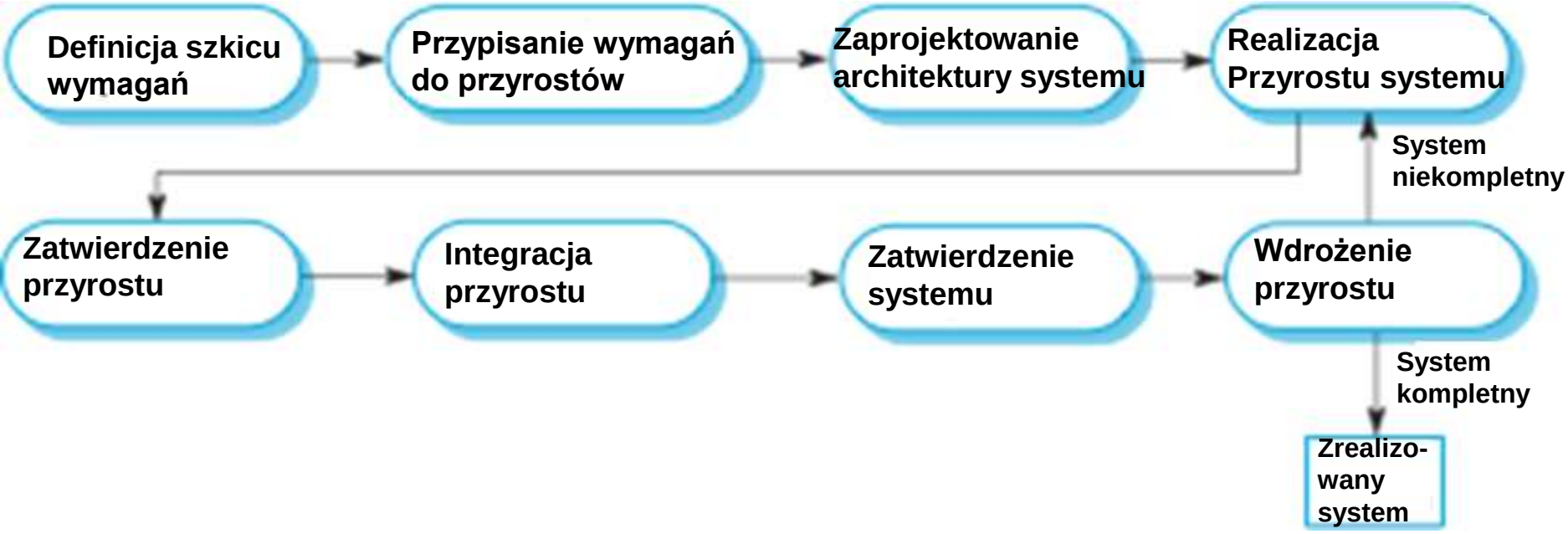
Prototyp a dalszy rozwój systemu

- Prototypy powinny być porzucane po opracowaniu ponieważ nie stanowią dobrej bazy do produkcji systemu:
  - Może być niemożliwe dostrojenie prototypu do spełnienia wymagań niefunkcjonalnych;
  - Prototypu są nieudokumentowane;
  - Z powodu szybkich zmian prototyp ma zazwyczaj mocno zdegradowaną strukturę;
  - Podczas produkcji prototypu omija się standardy jakości, które wykorzystuje organizacja.

Dostarczanie przyrostowe

- Zamiast dostarczać system w jednym kroku, rozwój i dostawy podzielone są na przyrosty. Każdy przyrost dostarcza część wymaganej funkcjonalności.
- Wymagania użytkowników podlegają priorytetyzacji – te, którym przypisano najwyższe priorytety włączane są do wczesnych przyrostów.
- W momencie rozpoczęcia realizacji przyrostu (iteracji) – wybrane wymagania są zamrażane. Natomiast wymagania przeznaczone do kolejnych przyrostów mogą cały czas podlegać ewolucji.

Dostarczanie przyrostowe



Realizacja przyrostowa a dostarczanie przyrostowe

- Realizacja przyrostowa
  - Iteracyjne Budowanie systemu (przyrosty). Ocena każdego przyrostu przed przejściem do kolejnego etapu.
  - Podejście wykorzystywane metodach zwinnych (agile).
  - Ocena wykonywana przez pełnomocnika użytkownika/klienta.

- Dostarczanie przyrostowe
  - Iteracyjne wdrażanie systemu – dostarczenie przyrostu do wykorzystania przez użytkowników.
  - Bardziej realistyczna ocena bazująca na praktycznym wykorzystaniu oprogramowania

Zalety dostarczania przyrostowego

- Każdy przyrost wprowadza elementy wartościowe dla klienta – wybrana funkcjonalność jest dostarczana wcześniej.
- Wczesne przyrosty działają jak prototypu – pomagają pozyskać wymagania dla kolejnych etapów.
- Zmniejszenie ryzyka porażki projektu.
- Usługi systemu o wyższym priorytecie są dłużej testowane.

Problemy dostarczania przyrostowego

- Większość systemów wymaga zestawu podstawowych funkcjonalności, które są wykorzystywane przez pozostałe części systemu.
  - Ponieważ wymagania nie są definiowane szczegółowo do momentu rozpoczęcia przyrostu, który ma je zrealizować, trudno jest dokonać identyfikacji wspólnych udogodnień, które będą wykorzystywane we wszystkich przyrostach..
- Istotą procesu iteracyjnego jest to, że specyfikacja jest opracowywana w połączeniu z oprogramowaniem.
  - W wielu organizacjach, pełna specyfikacja systemu jest częścią umowy dotyczącej jego rozwoju.

The Rational Unified Process

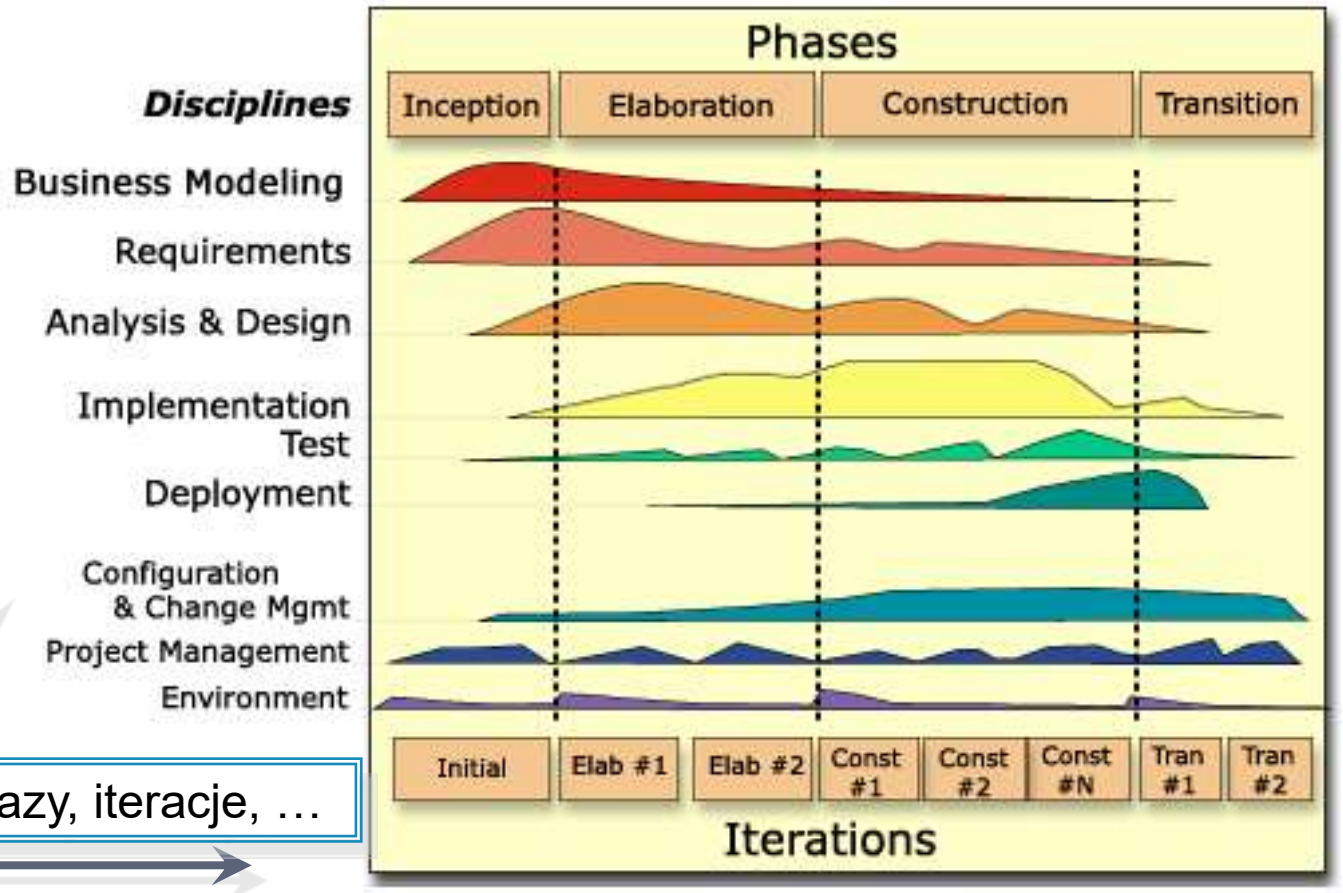
- Model procesu wywodzący się z procesów wykorzystujących język UML
- Łączy w sobie elementy 3 generycznych procesów (kaskadowy, realizacja przyrostowa i zorientowanie na wielokrotne użycie).
- Opisywany z trzech perspektyw
  - Dynamicznej – fazy w funkcji czasu
  - Statycznej – czynności procesu (procesy pracy – ang. workflow)
  - Pragmatycznej – sugestie wskazówki, dobre praktyki, narzędzia

Podejście RUP – podstawowe zasady

- Atakuj główne ryzyka od początku i przez cały proces … albo one dobiorą się do Ciebie
- Uzyskaj pewność, że spełniasz wymagania użytkowników
- Miarą postępu jest działający kod
- Przystosuj się do zmian już we wczesnych fazach projektu
- Na wstępie zdefiniuj architekturę systemu
- Wykorzystuj architekturę komponentową
- Pracuj zespołowo
- Uczyń z jakości drogę a nie rezultat

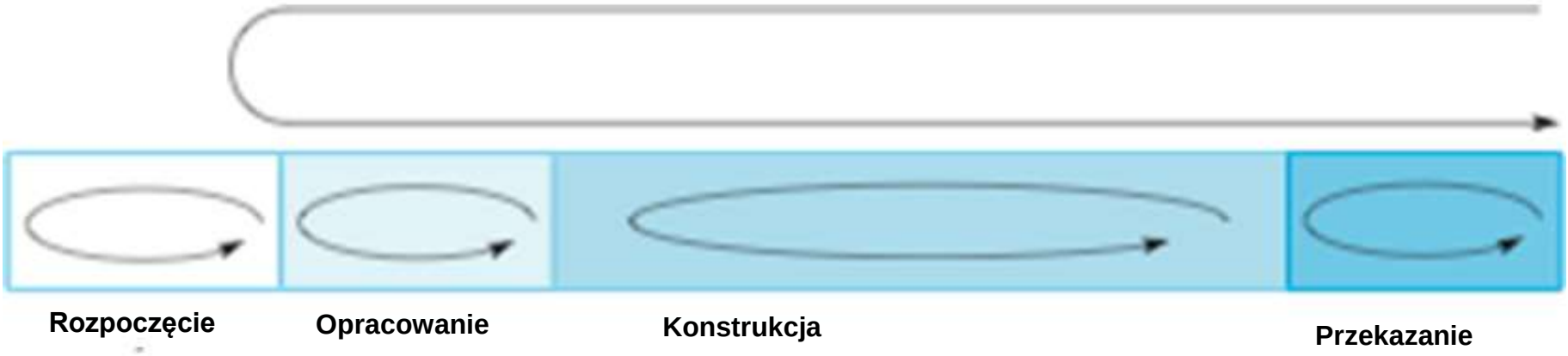
RUP – struktura dwuwymiarowa

Wymiar struktury procesu:
Jak elementy procesu (aktywności, artefakty, role, …) są logicznie pogrupowane w podstawowe dyscypliny procesu



Wymiar czasu – cykle, fazy, iteracje, …

Iteracje w procesie RUP



Dobre praktyki w RUP

- Buduj przyrostowo
- Zarządzaj wymaganiami
- Wykorzystuj architekturę komponentową
- Modeluj wizualnie
- Stale weryfikuj jakość
- Zarządzaj zmianami

Perspektywa statyczna w RUP – procesy pracy (1 z 2)

Proces pracy	Opis
Modelowanie biznesowe (Business modelling)	Procesy biznesowe modelowane są za pomocą biznesowych przypadków użycia
Wymagania (Requirements)	Aktorzy będący w interakcji z systemem oraz przypadki użycia modelujące wymagania dla systemu.
Analiza i projektowanie (Analysis and design)	Model projektowy budowany jest z wykorzystaniem modeli architektonicznych, modeli komponentowych, modeli obiektowych oraz modeli sekwencji.
Implementacja (Implementation)	Komponenty systemu są implementowane i reprezentowane w postaci podsystemów. Wykorzystuje się możliwości automatycznej generacji kodu w celu przyspieszenia procesu.

Perspektywa statyczna w RUP – procesy pracy (2 z 2)

Proces pracy	Opis
Testowanie (Testing)	Testowanie jest procesem iteracyjnym realizowanym r powiązaniu z implementacją. Po zakończeniu implementacji wykonywane są testy systemowe.
Wdrożenie (Deployment)	Tworzone jest wydanie systemu, które jest dystrybuowane do użytkowników.
Zarządzanie konfiguracją i zarządzanie zmianą (Configuration and change management)	Proces wspierający - zarządzanie zmianami w systemie
Zarządzanie projektem (Project management)	Proces wspierający -zarządzanie rozwojem systemu
Środowisko (Environment)	Proces pracy związany z zapewnianiem dostępności odpowiedniego zestawu narzędzi CASE.

Podsumowanie

- Proces musi obejmować czynności umożliwiające radzenie sobie ze zmianami. Przykładem takiej czynności jest etap prototypowania pozwalający na zmniejszenie ryzyka błędnych decyzji dotyczących wymagań i projektu systemu.
- Proces może być tak skonstruowany aby umożliwiać rozwój oraz dostarczenie przyrostowe. Dzięki iteracyjności zmiany mogą być wprowadzane przy minimalizacji ryzyka naruszenia całości systemu.
- Przykładem nowoczesnego generycznego modelu procesu IO jest Rational Unified Process. Proces ten zorganizowany jest w fazy (rozpoczęcie, opracowanie, konstrukcja, przekazanie) odseparowane od konkretnych czynności (wymagania, analiza i projektowanie,...).