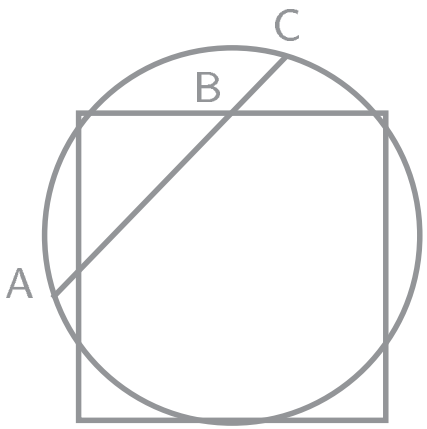
Inżynieria oprogramowania

Radosław Klimek

2015-23



http://home.agh.edu.pl/rklimek

Cykle życia oprogramowania

Henri MATISSE: *Taniec*

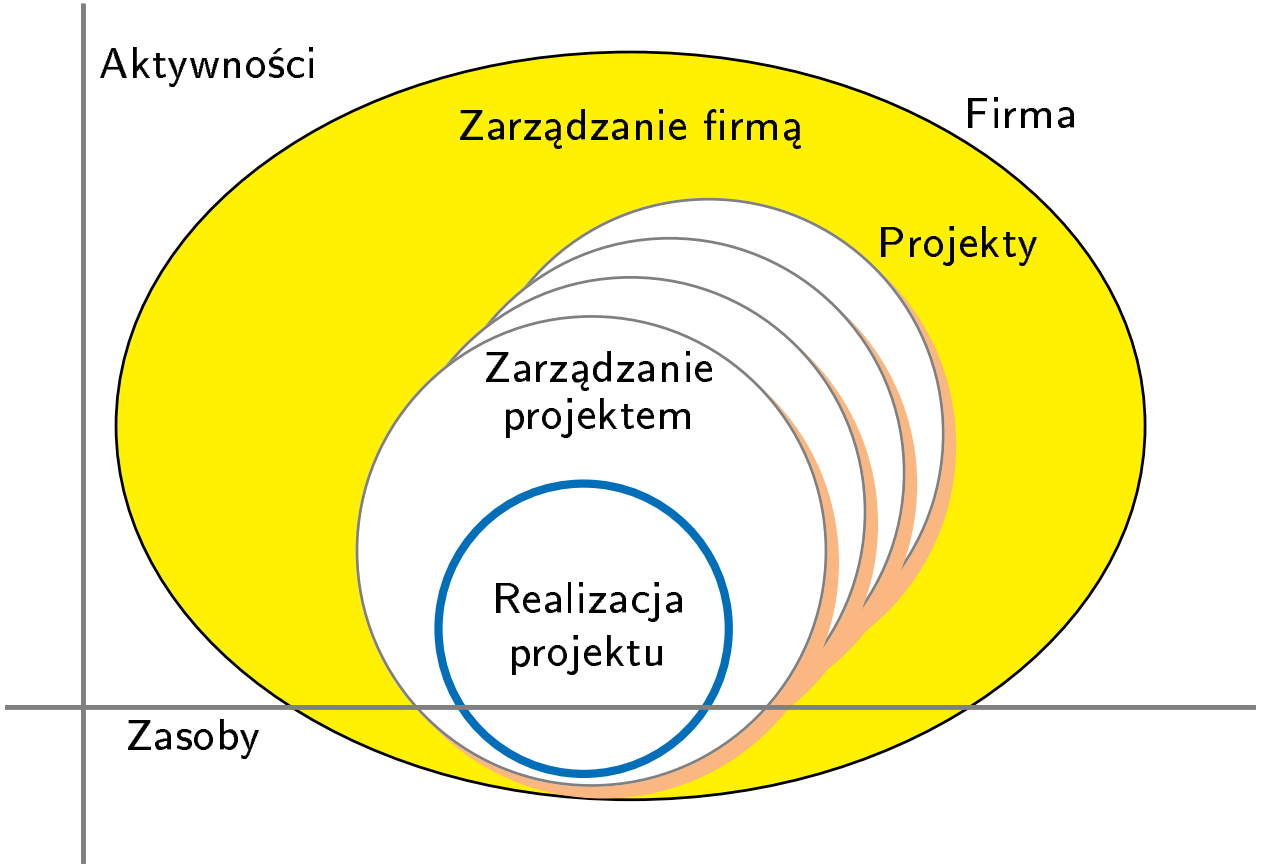
Projekt i jego kontekst

Projekt – także informatyczny – będzie tutaj oznaczał wszelką aktywność techniczną, której celem jest przejście od wymagań/oczekiwań użytkownika do fizycznego produktu spełniającego jego oczekiwania.

Każdy projekt charakteryzują po pierwsze jego cele oraz po drugie jego sposób realizacji i postęp z tym związany.

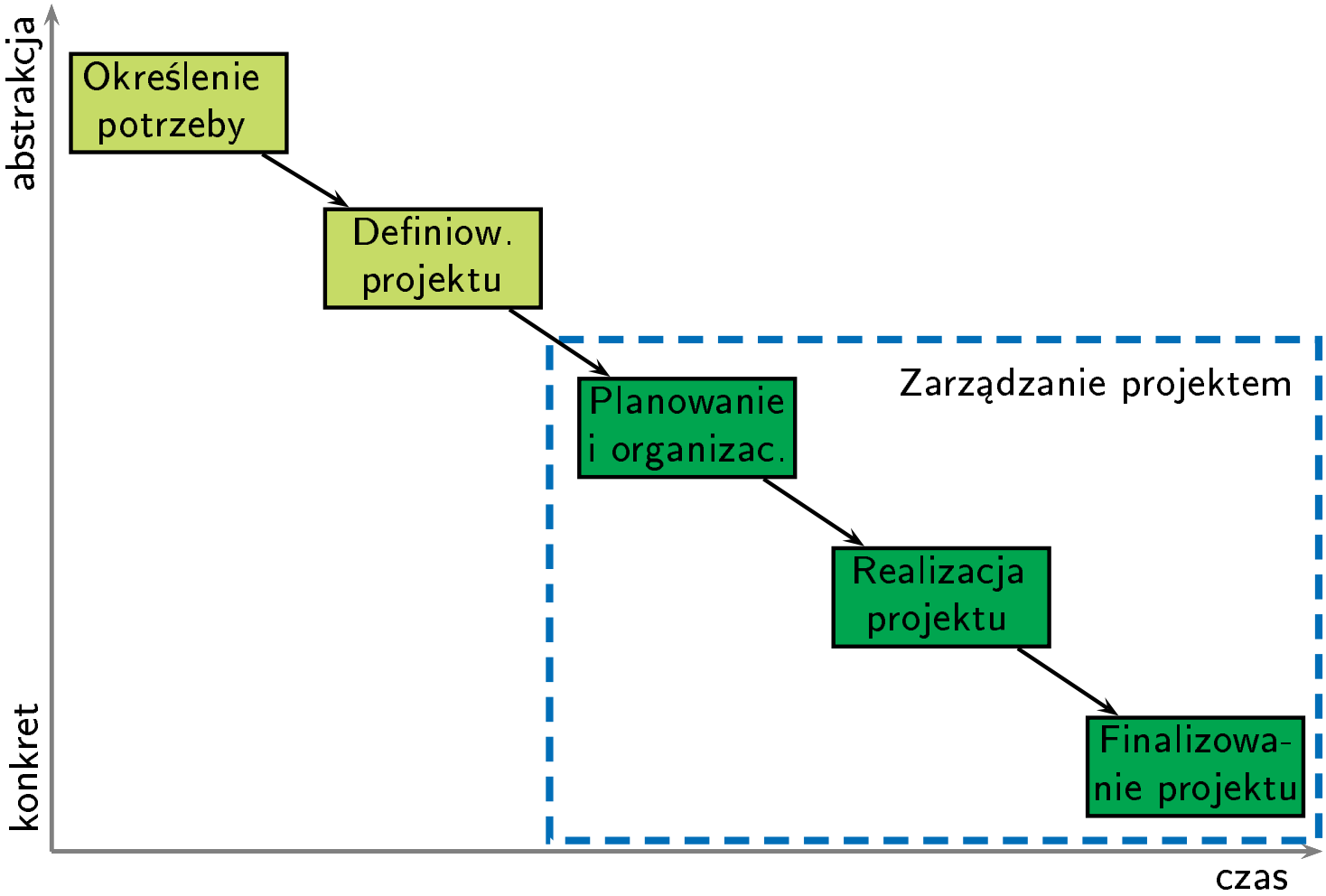
W przypadku każdego projektu nie należy zapominać, że w firmie może być realizowanych niezależnie wiele projektów – firma stanowi jakby „zbiór projektów”.

Projekt i jego kontekst (cd.)



Opisy osi „Zasoby” oraz „Aktywności” nie należy odczytywać wprost na wykresie, raczej jest to zaznaczenie o zmienności tych wielkości w zależności od podejmowanych w firmie projektów.

Fazy projektu a zarządzanie projektem



Na projekt (informatyczny) można popatrzeć jak na związany (w czasie) z pewnymi aktywnościami zarówno przed, jak i w trakcie.

Fazy projektu a zarządzanie projektem (cd.)

- **Określenie potrzeby** – wyrażenie potrzeby i zapotrzebowania na system wraz z określeniem celu;
- **Definiowanie projektu** – wstępna analiza, studium wykonalności, możliwości rozwiązań, kryteria decyzji, budżet, terminy;
- **Planowanie i organizacja** – po decyzjach startegicznych: identyfikowanie zadań projektu (czas, jakość, budżet), zasoby;
- **Realizacja projektu** – techniczna realizacja całego projektu ze wszystkimi aspektami tej fazy;
- **Finalizowanie projektu** – konfrontacja projektu z przyjętymi celami, formalna akceptacja, zwolnienie zasobów, raporty.

Procesy tworzenia oprogramowania

W trakcie prac nad projektem można wyróżnić i zdefiniować kilka typowych procesów projektowych:

- **inicjacja** – związana z podjęciem decyzji strategicznych i przydzieleniem zasobów do projektu;
- **specyfikacja** – zazwyczaj uznawana jako pierwszy etap prac, umożliwia stworzenie formalnego opisu systemu, typowe i przykładowe zadania składające się na specyfikację to: wywiady, analiza danych, specyfikacja wymagań, itd;
- **projektowanie** – przedstawienie schematycznej reprezentacji składników systemu i ich wzajemnych powiązań; zazwyczaj projektowanie obejmuje wiele procesów, także odmiennych w swoim charakterze;

Uwaga: Opisane procesy projektowe nie odnosimy tutaj jeszcze do żadnego cyklu życia oprogramowania – są to procesy ogólne.

Procesy tworzenia oprogramowania (cd.)

- **kodowanie** – napisanie przez programistę pojedynczego modułu, wraz z przetestowaniem tego modułu;
- **integracja** – łączenie modułów w większe całości (podsystemy), które z kolei są łączone w cały system;
- **weryfikacja** – ocena zgodności produktu z jego specyfikacją – produkt ma być wiernym „tłumaczeniem" specyfikacji;
- **walidacja** – sprawdzenie czy otrzymano właściwy system, wyspecyfikowany z udziałem klienta i użytkownika;

- **Weryfikacja** polega na ustaleniu relacji pomiędzy produktem a specyfikacją, tzn. czy budujemy produkt w odpowiedni sposób względem specyfikacji.
- **Walidacja** polega na ustaleniu relacji pomiędzy zaplanowanymi wynikami a wytworzonymi procedurami i czynnościami systemu, tzn. potwierdzenie działania w odpowiedni sposób zgodny z oczekiwaniami klienta.

Procesy tworzenia oprogramowania (cd.)

- **instalacja** – zainstalowanie systemu w środowisku docelowym wraz z przeprowadzeniem testów akceptacyjnych;
- **pielęgnacja/utrzymywanie** – bieżące użytkowanie systemu wraz z usuwaniem błędów oraz modyfikowaniem (rozwijaniem) systemu zgodnie z potrzebami.

Cele dekompozycji procesów

W trakcie tworzenia oprogramowania dokonuje się dekompozycji całego procesu na procesy mniejsze. Zasadnicze cele takiego postępowania można ująć w trzech punktach:

1	**ułatwienie rozwiązania problemu** – poszczególne etapy mają mniejszy zakres i są mniej skomplikowane, w efekcie łatwiejsze do wykonania;
2	**lepsze zarządzanie** – dekompozycja przekłada się bezpośrednio na łatwiejsze zarządzanie całym przedsięwzięciem;
3	**łatwiejsza kontrola jakości** – na końcu każdego etapu może być wykonana właściwa analiza i kontrola.

Immanentne problemy projektowania systemów

Z projektowaniem oprogramowania związanych jest szereg istotnych, nieodłącznych problemów, a tylko jako niektóre z nich można wymienić:

- **problem złożoności wymagań**
  klient często nie zna „z góry" wszystkich wymagań funkcjonalnych projektowanego systemu, nie wspominając już o wymaganiach niefunkcjonalnych;
- **problem zmiany wymagań**
  wymagania często ulegają zmianie, także nowe technologie mogą mieć wpływ na wymagania niefunkcjonalne i możliwości ich opanowania;

Immanentne problemy projektowania systemów (cd.)

- **trudność zarządzania częstymi zmianami**
  przy częstych zmianach istnieją duże trudności w definiowaniu punktów kontrolnych przedsięwzięcia oraz szacowaniu nakładów projektu;
- **problem kompatybilności i obecności innych systemów**
  nowy system bardzo często musi być kompatybilny ze starym systemem, a także współpracować z systemami istniejącymi.

Cykl życia oprogramowania – pierwsza definicja

Definicja

Cyklem życia oprogramowania, lub **systemu**, (ang. *software life cycle*, *system life cycle*) nazywamy okres czasu od początku prac nad stworzeniem oprogramowania do końca jego eksploatacji. **Modelem cyklu życia** (ang. *model life cycle*) nazywamy strukturę procesu związanego z realizacją takiego systemu. ⌟

Cykl życia oprogramowania – pierwsza definicja

Definicja

Cyklem życia oprogramowania, lub **systemu**, (ang. *software life cycle*, *system life cycle*) nazywamy okres czasu od początku prac nad stworzeniem oprogramowania do końca jego eksploatacji. **Modelem cyklu życia** (ang. *model life cycle*) nazywamy strukturę procesu związanego z realizacją takiego systemu. ⌟

- Cykl życia systemu pokrywa wszystkie jego fazy, tj. od studium wykonalności, analizę, specyfikację, projekt, implementację, ale także użytkowanie czy pielęgnację, tj. fazy które następują już po zaakceptowaniu systemu przez użytkownika końcowego.
- Modele cyklu życia oprogramowania pozwalają uporządkować przebieg prac, ułatwiają planowanie czy monitorowanie.

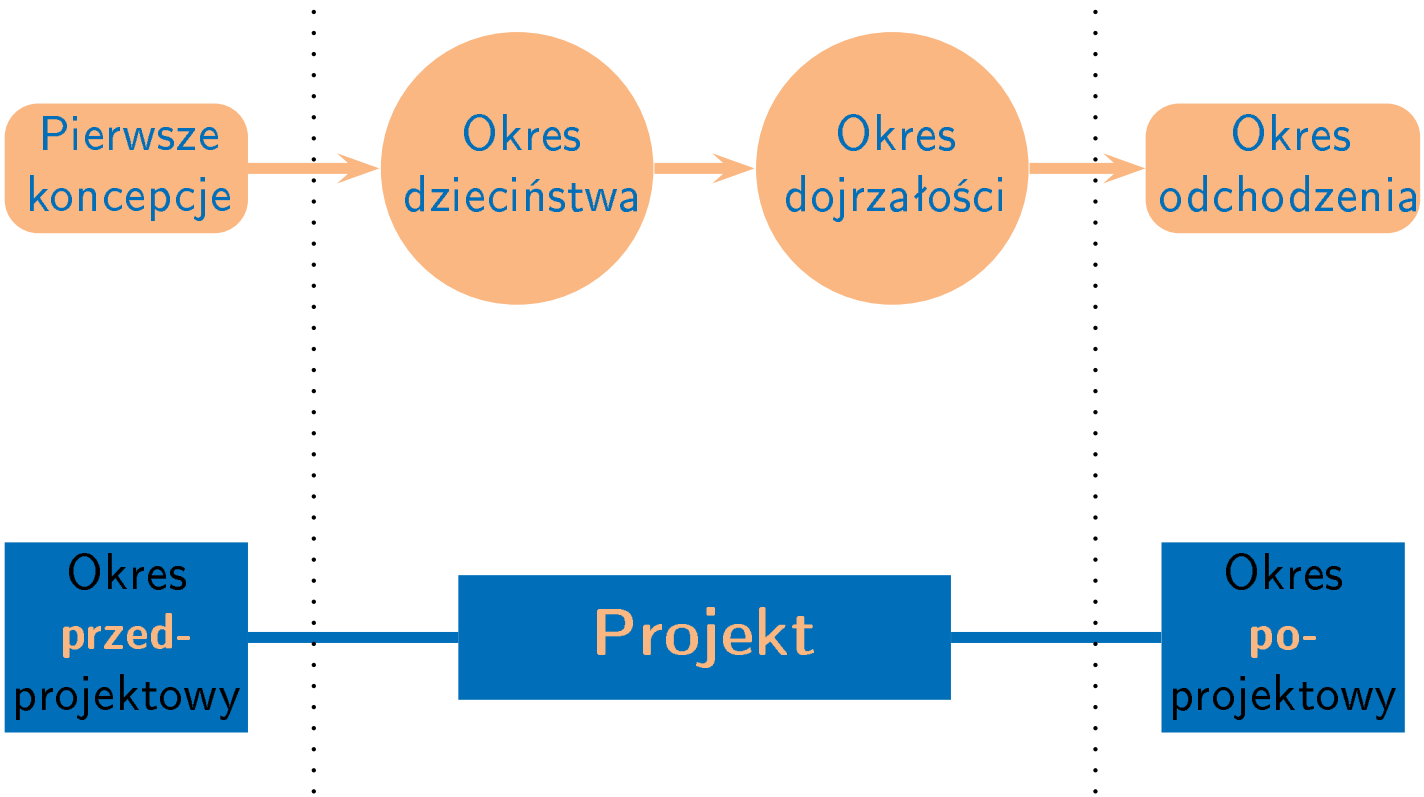
Cykl życia systemu a projektu

cykl życia systemu $\neq$ cykl życia projektu

Podstawowe różnice:

1. cykl życia projektu często „zawiera” się w cyklu życia systemu, choć nie jest to regułą;
2. model cyklu życia systemu koncentruje się raczej na dostarczeniu systemu, podczas gdy model cyklu życia projektu na wszystkich aspektach prowadzących do osiągnięcia celów projektu.

Cykl życia a następstwo stanów



Następstwo stanów – uwagi

- „Projekt” w znaczeniu „rozwój projektu”, działaność projektowa.
- „Okres odchodzenia” obejmuje także normalne użytkowanie systemu, „odchodzenie” w znaczeniu że po projekcie. Sam okres użytkowania może trwać nawet kilkadziesiąt lat.

Typowe problemy wyboru cyklu

Można przedstawić podstawowe, wybrane problemy związane z cyklami życia oprogramowania, ich wyborem oraz zarządzaniem samymi cyklami:

jakie czynności należy przewidzieć dla projektu/rozwoju systemu?

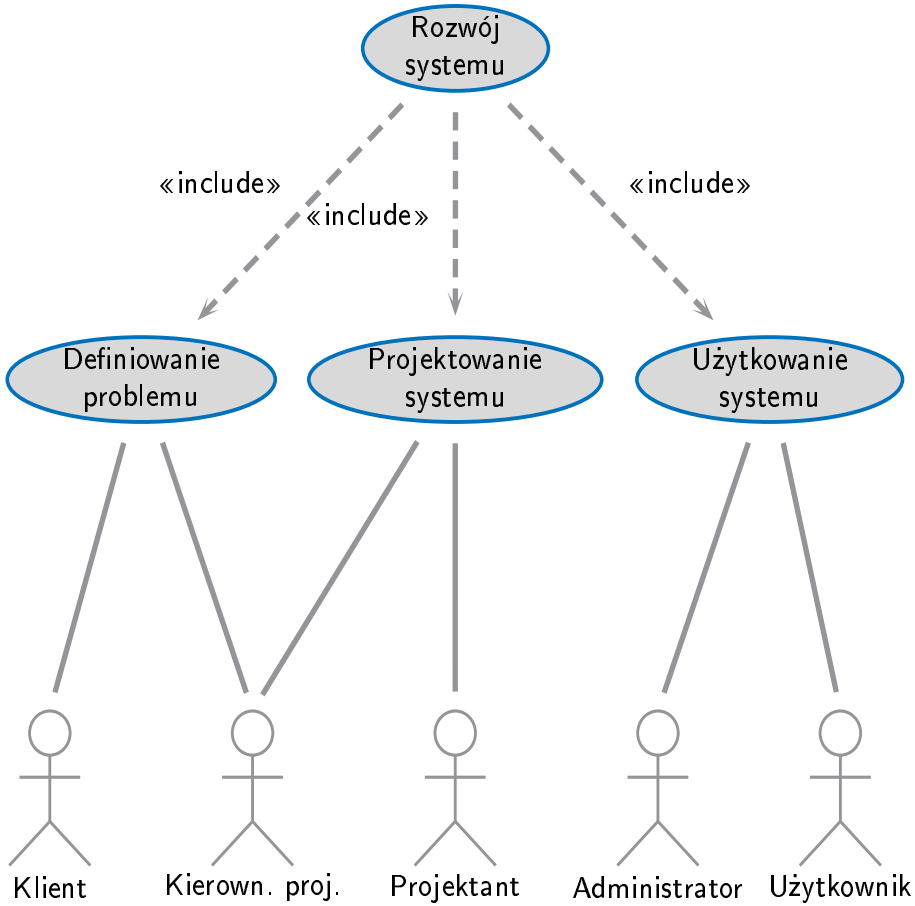
jakie są zależności pomiędzy poszczególnymi czynnościami projektu?

jak należy zaplanować czynności w całym harmonogramie?

Problemy te można wiązać z typowym działaniem zarządczym nad projektem informatycznym.

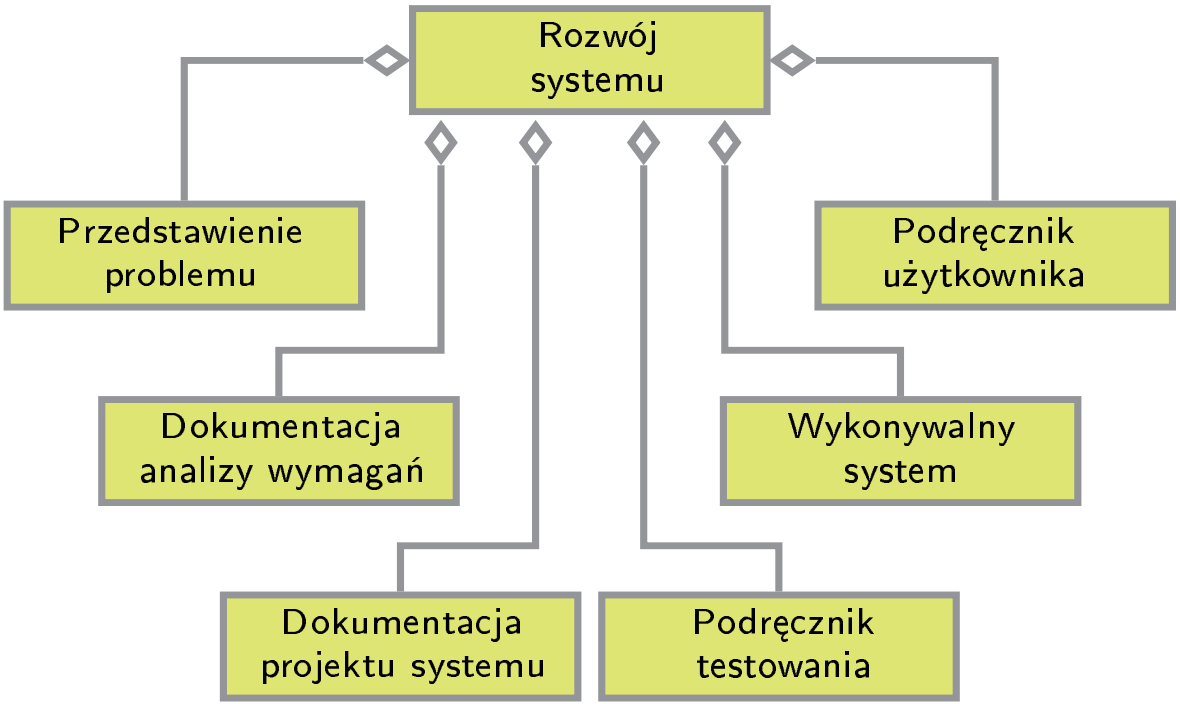
Cykl życia jako diagram przypadków użycia

Cykl życia systemu można przedstawić w dziedzinie pewnej aplikacji przy pomocy diagramu przypadków użycia języka UML.



Cykl życia – prosty model obiektowy

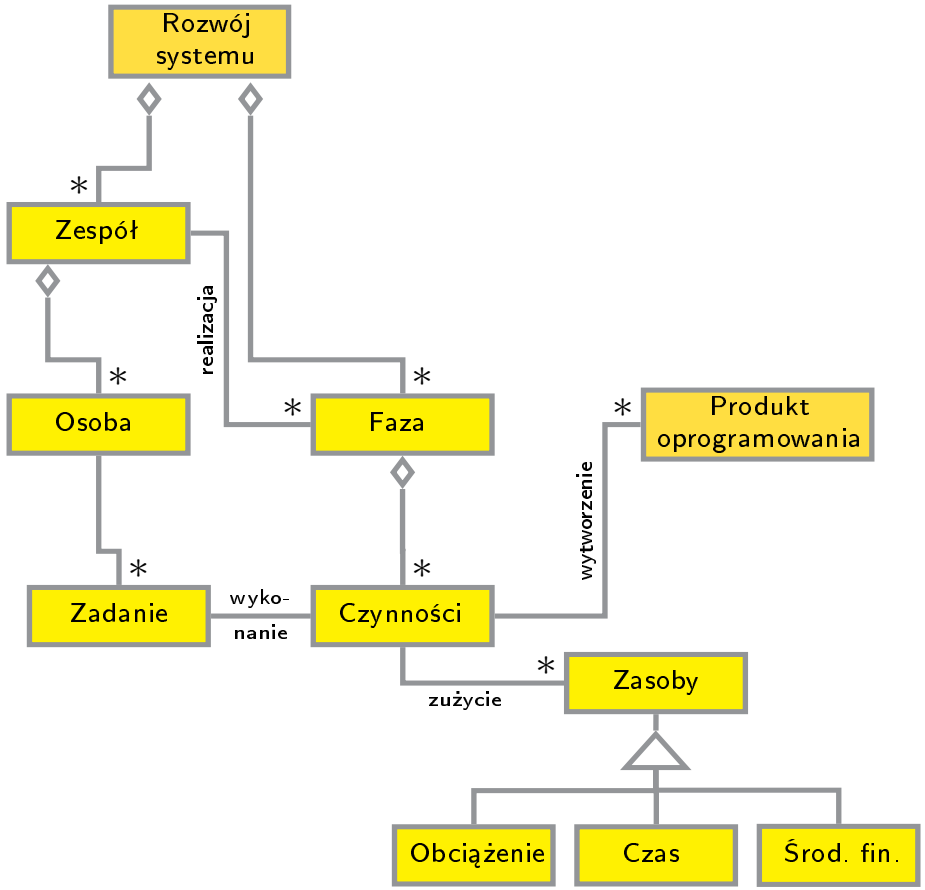
Cykl życia systemu można także przedstawić z wykorzystaniem podejścia obiektowego, przy zastosowaniu już najprostszych relacji.



Pomiędzy „Rozwój systemu” a innymi obiektami występuje relacja agregacji (tutaj bez wchodzenia w szczegóły jak relacja wygląda dokładnie).

Cykl życia – inny model obiektowy

Bardziej złożony model obiektowy obrazujący cykl życia systemu i wytwarzania oprogramowania.

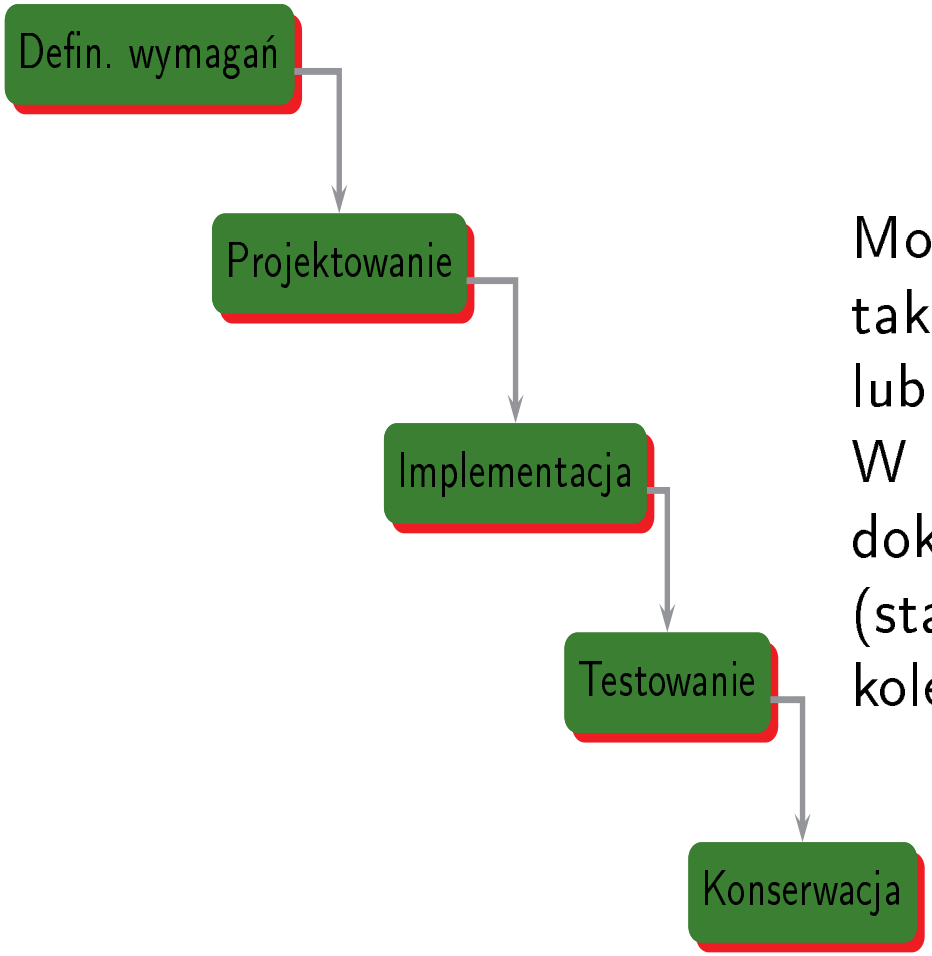


Model obiektowy – uwagi

- Przykład bardziej zaawansowanego modelu obiektowego dla cyklu wytwarzania oprogramowania.
- Można sobie wyobrazić, że istnieje potrzeba zinformatyzowania/skomputeryzowania samego cyklu wytwarzania oprogramowania.
- Modelu nie należy traktować jako wzorcowy, to tylko przykład, wszystko jest uzależnione od tego co jest dla nas ważne w samym cyklu, oraz jak ten cykl ma wyglądać – zależy to od kierownictwa.

Model kaskadowy – ujęcie historyczne

Jednym z pierwszych zdefiniowanych modeli życia oprogramowania był **model kaskadowy** (ang. *waterfall model*).



Model kaskadowy jest zwany także **modelem wodospadu** lub **modelem liniowym**. W każdej fazie wytwarzane są dokumenty będące podstawą (stanowiące wejście) dla fazy kolejnej.

Podstawowe fazy modelu kaskadowego

Fazy modelu w ujęciu klasycznym:

- **faza definiowania wymagań** (ang. *requirements*) – szczegółowe określenie wymagań wobec tworzonego systemu;
- **faza projektowania** (ang. *design*) – szczegółowy projekt systemu spełniający i precyzujący wcześniej określone wymagania;
- **faza implementacji/kodowania** (ang. *implementation/coding*) – wykonanie projektu w konkretnym środowisku programistycznym, także wstępne wykonanie **testów modułów** implementowanych w systemie;
- **faza testowania** (ang. *testing*) – integracja modułów połączona z testowaniem podsystemów i całości oprogramowania;

Podstawowe fazy modelu kaskadowego (cd.)

- **faza konserwacji** (ang. *maintenence*) – wykonywanie zmian i rozszerzeń zainstalowanego i działającego oprogramowania: usuwanie błędów, zmiany i rozszerzenia funkcjonalności.

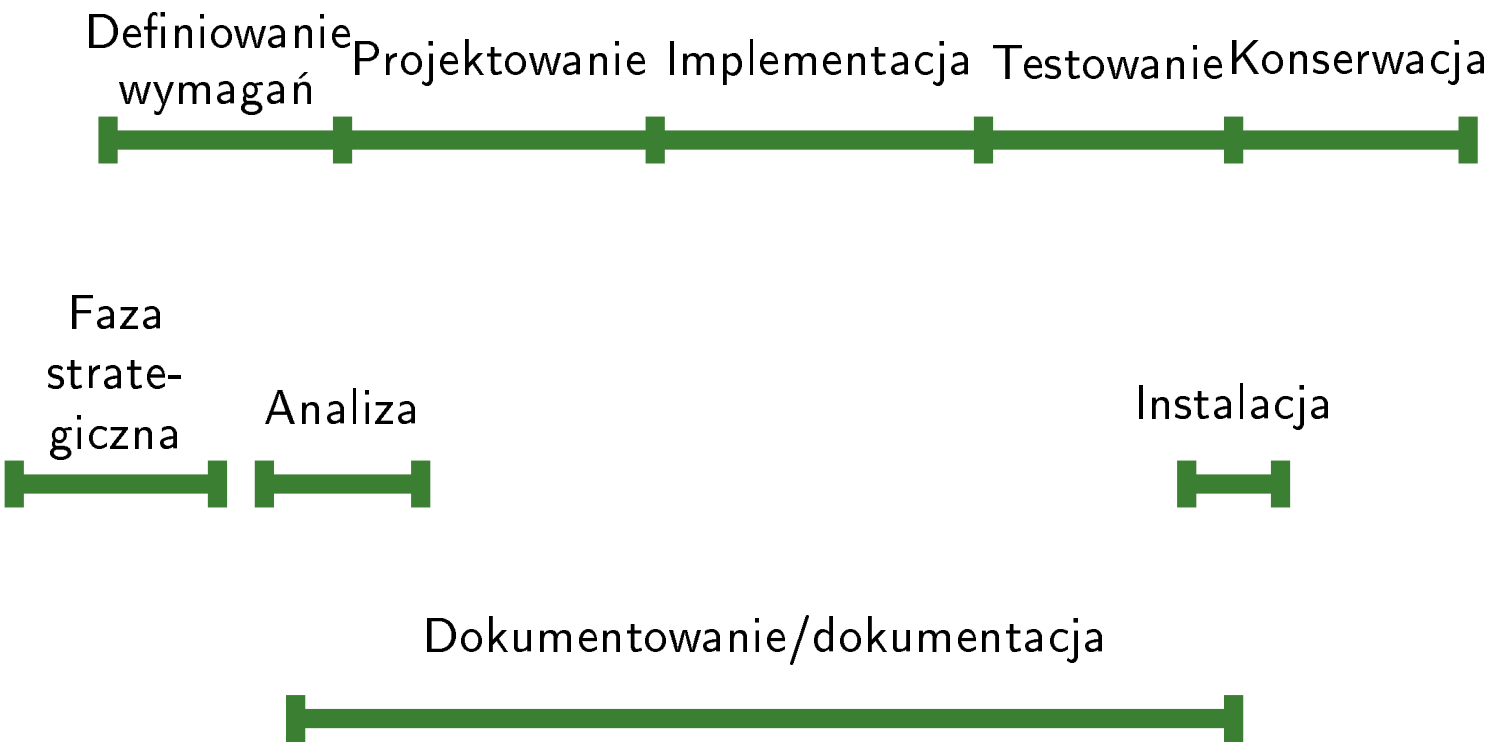
Istotą modelu kaskadowego jest spojrzenie na niego jak na proces liniowy, gdzie poszczególne fazy realizowane są sekwencyjnie.

Winston Royce – trochę historii

Winston Walker Royce (1929–1995) – amerykański informatyk, dyrektor Lockheed Software Technology Center in Austin, Texas. Pionier w badaniach nad rozwojem oprogramowania. Przypisano mu nieco błędnie termin model kaskadowy w rozwoju oprogramowania.

- Podejście wodospadowe zostało po raz pierwszy opsane przez przez Winstona Royce'a w 1970, jako coś co nie działa, ale autor nie użył tego terminu, (W.Royce: *Managing the Development of Large Software Systems*, Proceedings of IEEE WESCON 26 (1970), 1–9).
- Pierwszymi osobami, które go użyły byli T.E. Bell i T.A. Thayer w 1976, (T.E. Bell, T. A. Thayer: *Software requirements: Are they really a problem?* Proceedings of the 2nd International Conference on Software Engineering. IEEE Computer Society Press, 1976).
- Branża Agile stworzyła chłopca do bicia, aby łatwo było pokazać zalety Agile szczególnie tym, którzy nie wiedzą zbyt wiele o historii tworzenia oprogramowania.

Fazy dodatkowe modelu kaskadowego

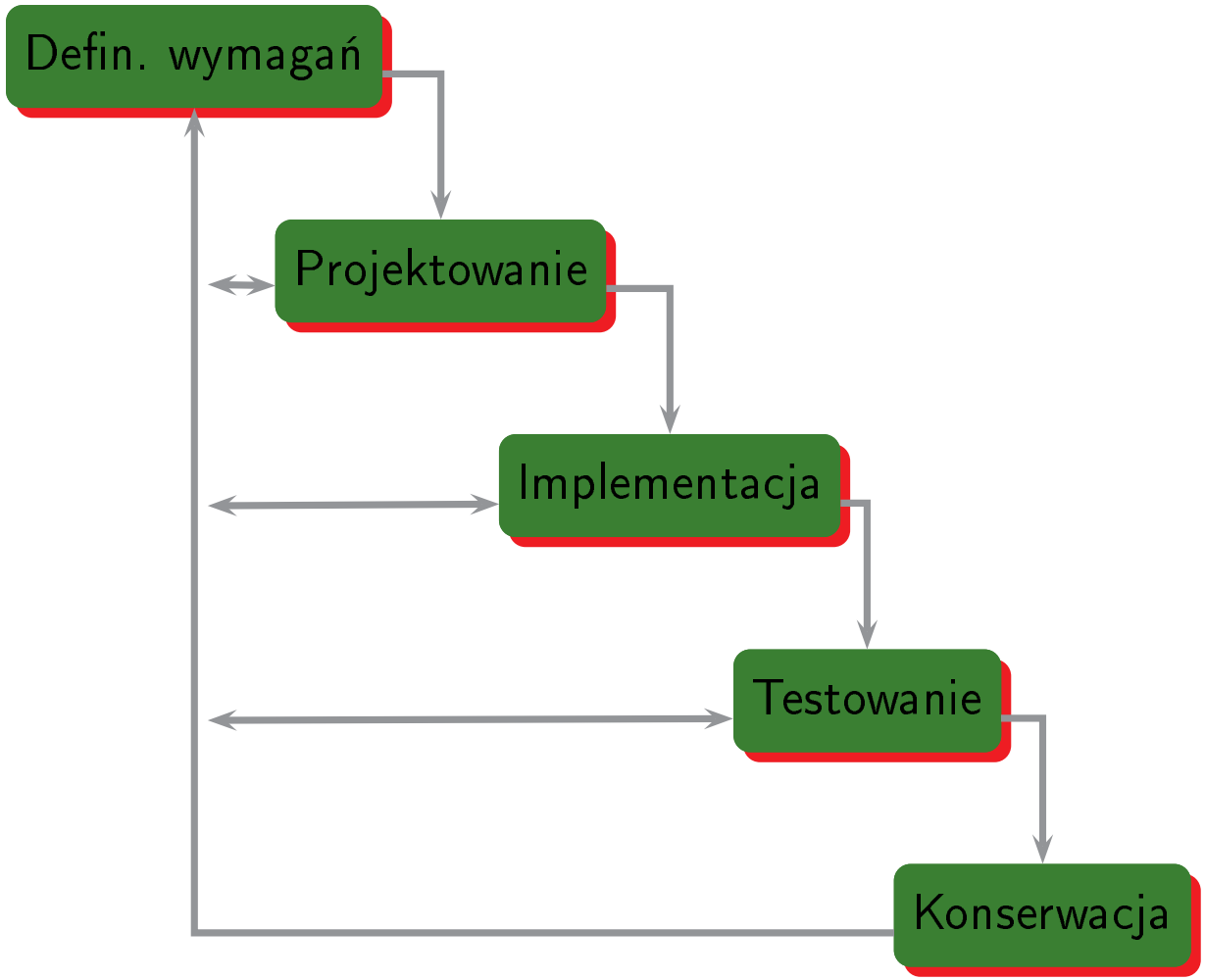


Fazy dodatkowe modelu kaskadowego (cd.)

Dodatkowe fazy cyklu:

- **faza strategiczna** (ang. *strategy*) – podjęcie strategicznych decyzji odnośnie przedsięwzięcia;
- **faza analizy** (ang. *analysis*) – budowanie modelu logicznego systemu;
- **faza instalacji** – przekazanie systemu użytkownikowi;
- **faza dokumentacji** – wytwarzanie dokumentacji użytkownika.

Nawroty w modelu kaskadowym

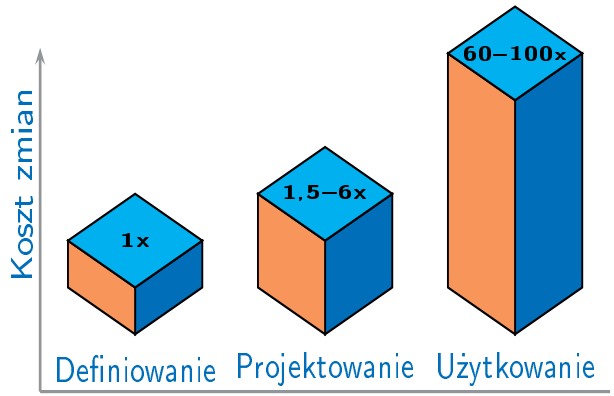


Model kaskadowy – pierwsze podsumowanie

Model został zdefiniowany na początku lat siedemdziesiątych (Winston Royce), w związku z tzw. kryzysem oprogramowania, jako potrzeba formalizacji procesów projektowych, model został przyjęty entuzjastycznie (już wtedy była dostrzeżona potrzeba istnienia pętli sprzężenia zwrotnego);
Względnie duża łatwość zarządzania przedsięwzięciem (lepsze planowanie, harmonogramowanie, monitorowanie, itd.), łatwość znajdowania punktów kontrolnych, ściśle określona kolejność działań, każdy etap musi być zakończony przed rozpoczęciem kolejnego, łatwość stwierdzenia postępu prac;
Wysoki koszt błędów popełnionych we wstępnych fazach – błędy mogą być wykrywane nawet w fazach końcowych lub już po dostarczeniu systemu;

.....

Wg. E.Yourdona:



Model kaskadowy – pierwsze podsumowanie (cd.)

Nie jest możliwe wytworzenie od razu – już na początku i za pierwszym razem – poprawnej i pełnej specyfikacji;
Klient uczestniczy tylko w fazach wstępnych i fazach końcowych – słabość modelu i zła praktyka;
Brak procesu (jawnej) weryfikacji pomiędzy poszczególnymi etapami;
Duża przerwa w kontaktach z klientem/odbiorcą (cierpliwość ze strony klienta), trudności w wydobywaniu wymagań;
Tak naprawdę proces produkcji oprogramowania ma raczej charakter iteracyjny;
Różnorodne i nieraz bardzo rozbieżne interpretacje modelu kaskadowego;
Model kaskadowy jest zalecany przede wszystkim do projektów krótkich – z dobrze określonymi wymaganiami nad którymi możemy „zapanować”;
Model kaskadowy powstał na początku lat siedemdziesiątych i na jego bazie powstało szereg innych.

Model kaskadowy – interpretacje

Odmienne **interpretacje**:

- np. „wszystkie projekty powstają wg tego modelu”,
- „żaden projekt nie powstaje wg tego modelu”

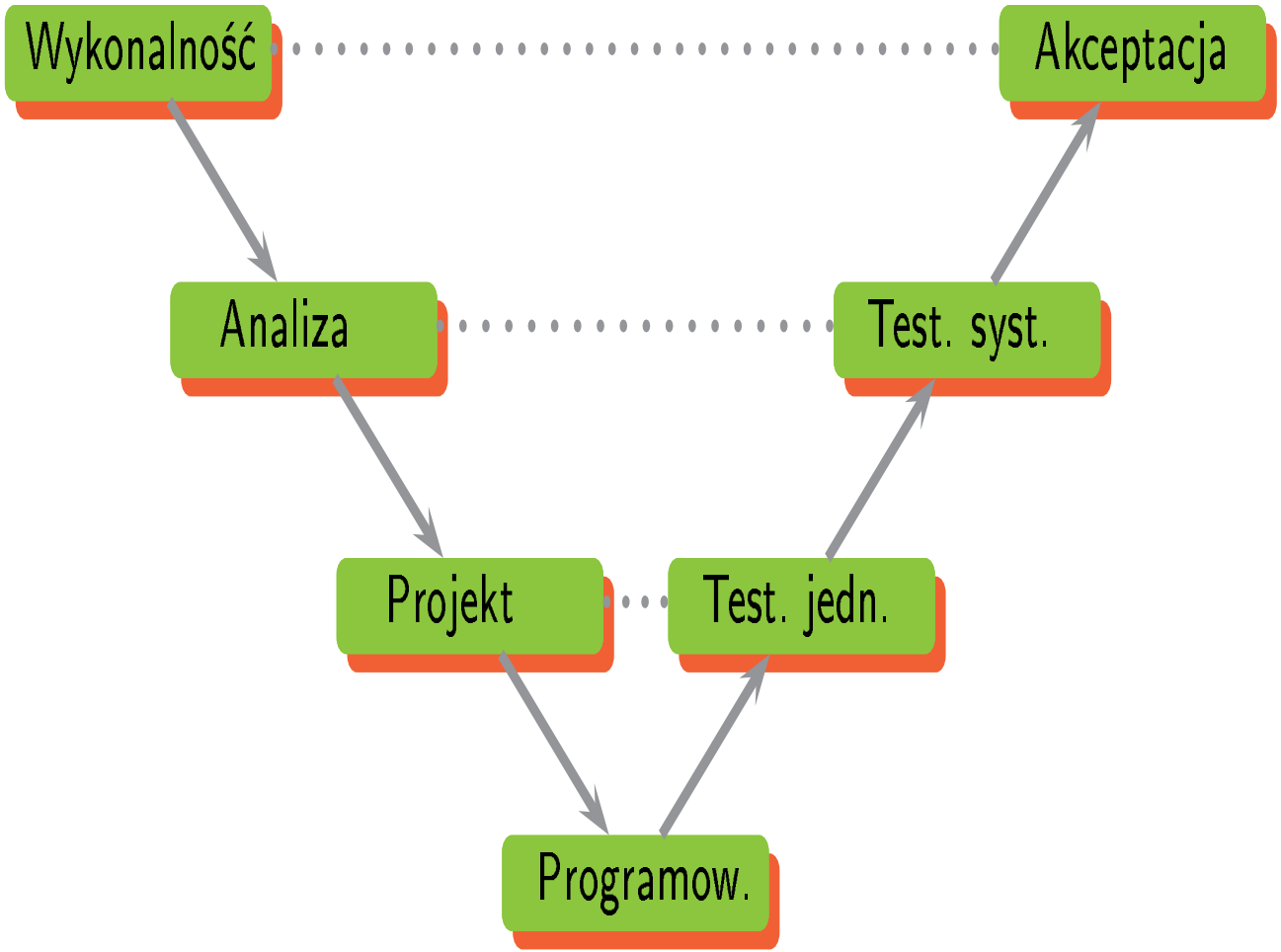
biorą się z faktu, że nie dopuszcza się – lub też odwrotnie dopuszcza się – nakładania się poszczególnych faz. W przypadku dopuszczania, przyjmuje się że fazy mają znaczenie raczej koncepcyjne, a poszczególne fragmenty systemu mogą być w tym samym czasie na różnych etapach realizacji.

Model V

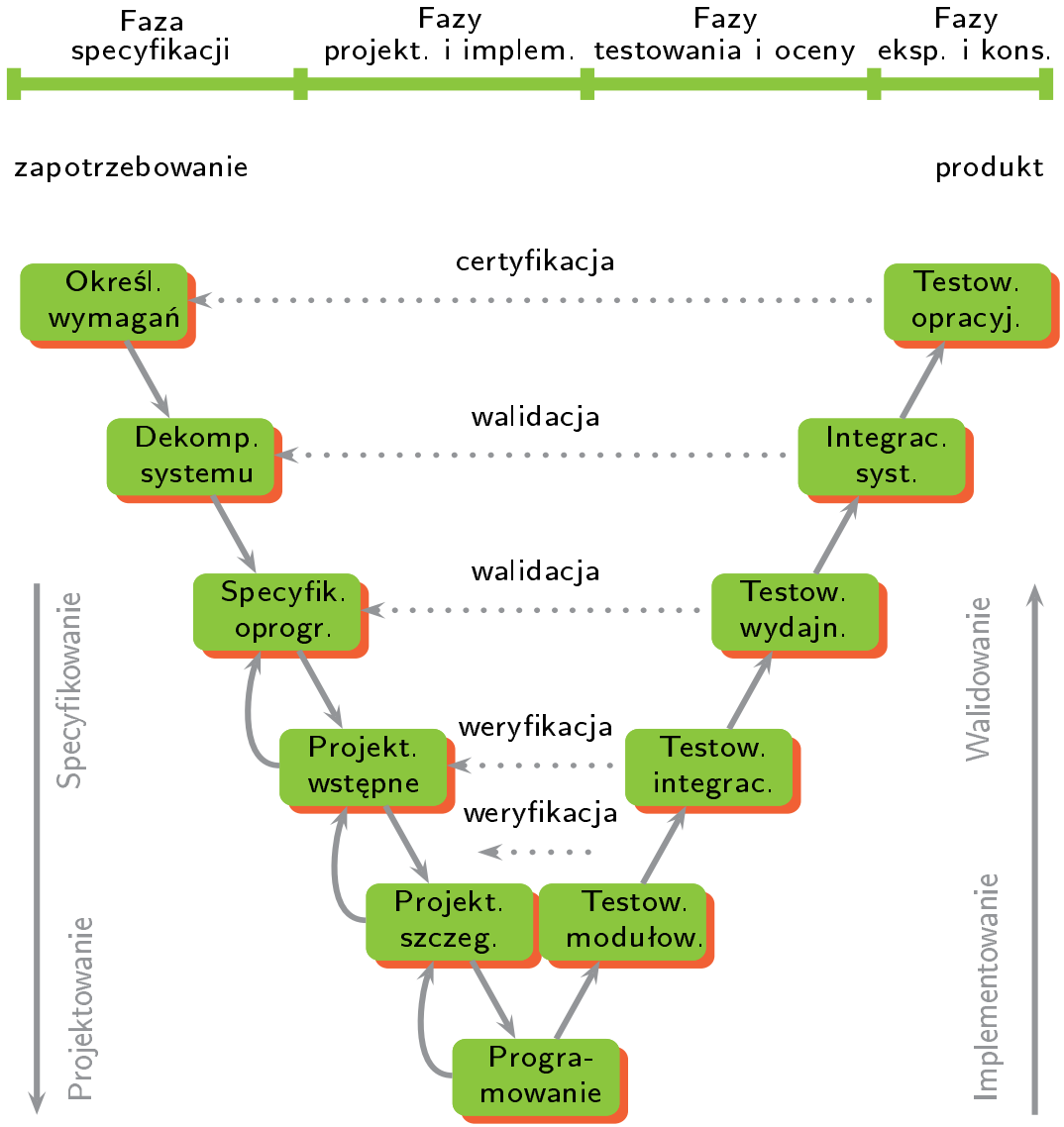
Podstawowe założenia modelu:

- zmiany poziomu abstrahowania projektu;
- rozwijanie oprogramowania ma charakter zstępujący, a jego testowanie i inspekcje charakter wstępujący;
- fazy weryfikacji są przewidziane i sprzężone z każdym krokiem rozwijania;
- proces zstępujący jest komplementarny ze wstępującym, i **vice versa**.

Model V – ilustracja



Przykład szczegółowy dla modelu V

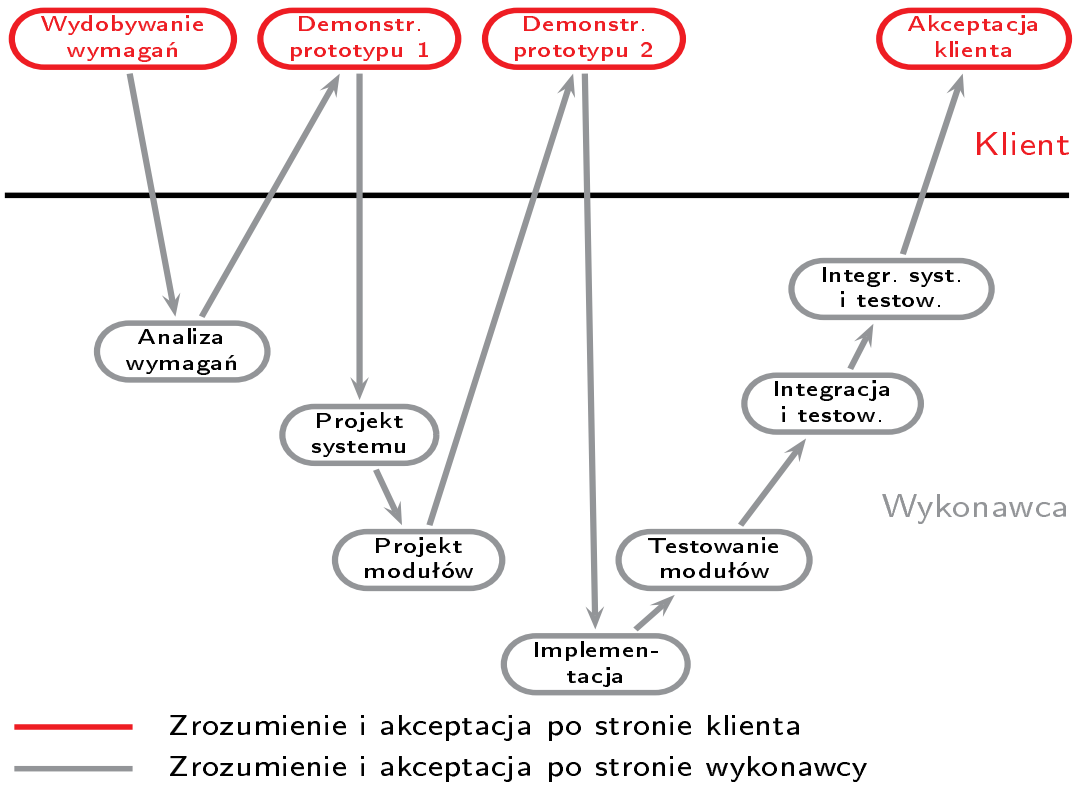


Model V – uwagi

- Widać wyraźnie, że specyfikowanie i projektowanie mają charakter zasadniczo zstępujący, a testowanie i walidacja charakter zasadniczo wstępujący.
- Oś pozioma pokazuje poszczególne fazy projektu w upływającym czasie. Osie pionowe zmiany poziomy abstrakcji/szczegółowości realizowanego projektu.
- Z programowaniem jest już związane wstępne (nieformalne) testowanie przez programistów!
- „Testowanie wydajnościowe" odnosi się raczej do kontroli spełniania wszystkich założeń oprogramowania.
- **Weryfikacja** polega na ustaleniu relacji pomiędzy produktem a specyfikacją, tzn. czy budujemy produkt w odpowiedni sposób.
- **Walidacja** polega na stwierdzeniu czy produkt spełnia oczekiwania, tzn. czy budujemy właściwy produkt.

Model „ząb piły”

Model określany jako „ząb piły” został zaproponowany w 1990 przez R. Rowena i był konsekwencją spostrzeżenia, że percepcja systemu przebiega po odmiennych trajektoriach w przypadku klienta/użytkownika oraz wykonawcy.



Model „ząb piły" (cd.)

- Uczestnicy budowania systemu informatycznego odmiennie postrzegają proces jego budowy i mają odmienne potrzeby. Ich percepcja przebiega po różnych trajektoriach, a np. w modelu V przebiega po tej samej trajektorii. Klient i projektant są zmuszeni „oglądać" system razem na całym etapie jego powstawania.
- Obie trajektorie przebiegają odmiennie ale mają te same punkty początkowy i końcowy.
- Odległość pomiędzy obu trajektoriami, w niektórych miejscach modelu, pokazuje odmienne postrzeganie systemu przez strony.
- Wprowadza się punkty kontrolne, gdzie na krótko obie trajektorie spotykają się.

R. B. Rowen: Software project management under incomplete and ambiguous specifications. **IEEE Transaction on Engineering Management**, 1990, vol. 37, no. 1.

Prototypt w modelu

Należy zwrócić uwagę na odmienny charakter obu prototypów występujących w modelu „zęba piły”, zwanych, odpowiednio, **rewolucyjnym** (ang. *revolutionary prototype*) oraz **ewolucyjnym** (ang. *evolutionary prototype*):

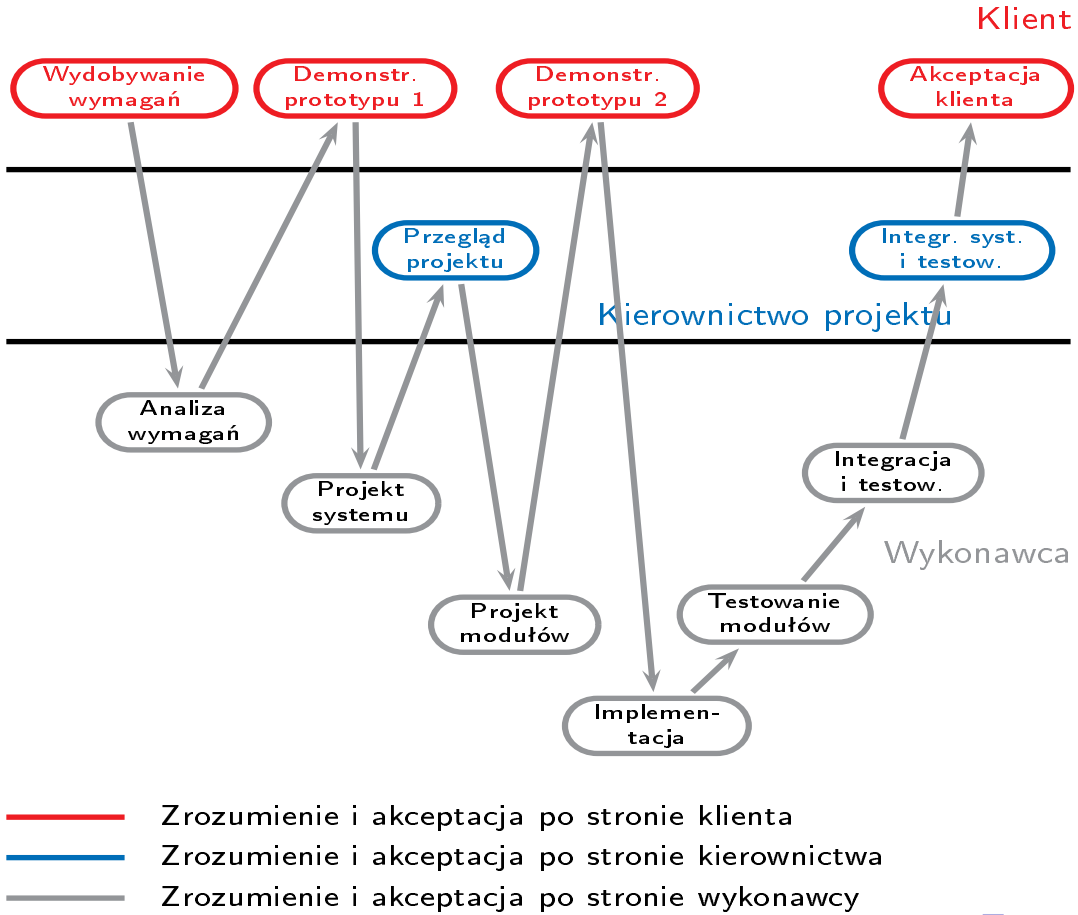
- Prototyp rewolucyjny ma charakter ilustracyjny.
- Prototyp ewolucyjny ma charakter jakby rozliczeniowy.

Prototyp rewolucyjny a ewolucyjny modelu

rewolucyjny	szybka budowa prototypu w celu niezwłocznej, jak najszybszej demonstracji funkcjonalności systemu, funkcjonalność tego prototypu będzie raczej fragmentem całej funkcjonalności systemu i powinna obejmować najważniejsze „miejsca" budowanego systemu; standardowe tutaj pytanie wykonawców: czy ten/taki prototyp zostanie zaakceptowany?
ewolucyjny	prototyp realizowany w dalszej fazie budowy systemu, gdy duża część jego funkcjonalności została być może już zaimplementowana, a na pewno wykonany jest cały projekt systemu; obecnie można wymagać demonstracji złożonych (także hierarchicznych) funkcjonalności realizowanej całości.

Model „ząb rekina”

Model określany jako „ząb rekina” jest w sposób oczywisty rozszerzeniem i uściśleniem poprzedniego modelu, przede wszystkim poprzez wprowadzenie nowego poziomu jakim jest kierownictwo.



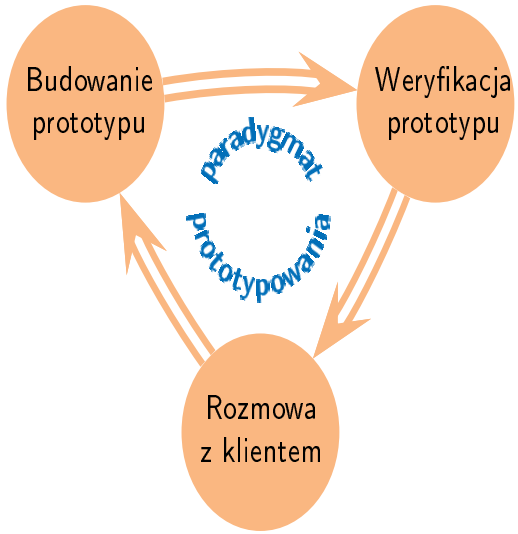
Model „ząb rekina" – uwagi

- W modelu wprowadza się nowy poziom jakim jest kierownictwo projektu i w konsekwencji mamy doczynienia z tzw. małym zębem oraz dużym zębem.
- Mały ząb jest utworzony przez przegląd projektu dokonywany wobec kierownictwa. Duży ząb dotyczy drugiego prototypu okazywanego klientowi. Oba prototypy tworzą ząb rekina.
- Także przesunięcie fazy „Integracji i testowania systemu" jest bardziej zrozumiałe jako związane z uczestnictwem managementu – oczywiście dobrym odbiorcą tej fazy raczej nie może być klient.
- Management może/powinien sobie pozwolić na głębsze wejście/zrozumienie budowanego systemu niż klient.

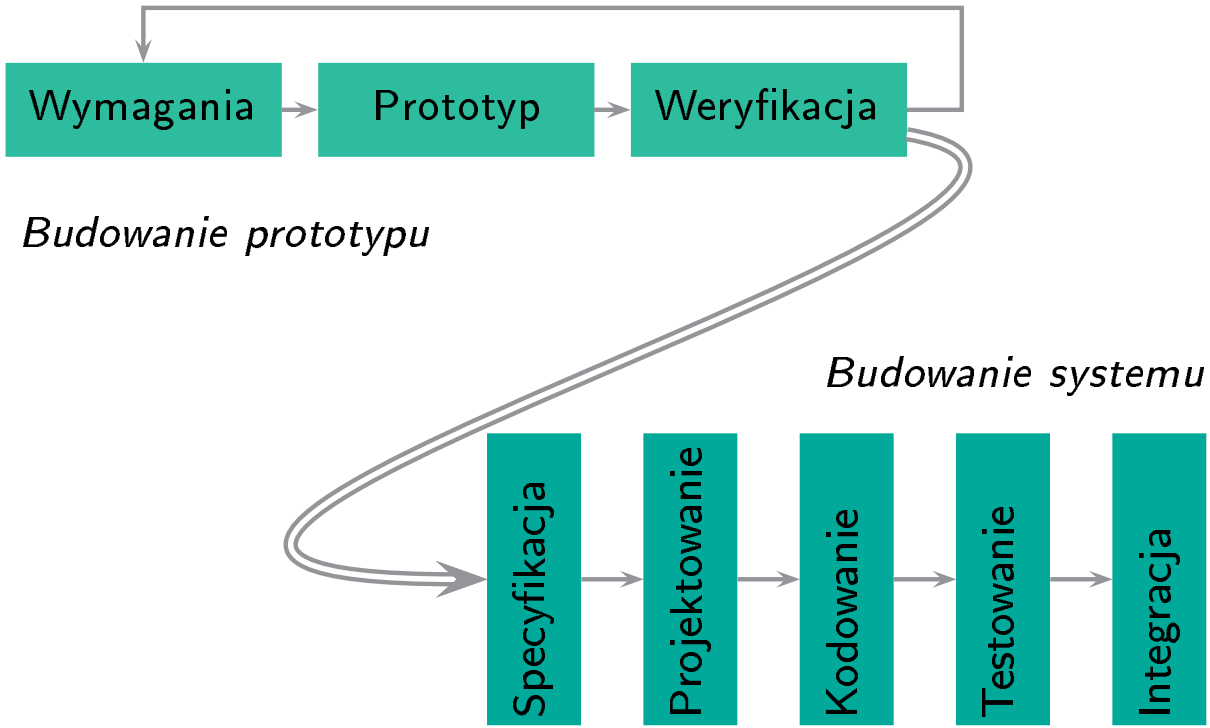
Model prototypowania

Prototypowanie (ang. *prototyping model*), lub **makietowanie**, jest stosowane wtedy, gdy istnieje trudność uzyskania (pełnej) informacji o wymaganiach systemu, względnie informacje te nie są pewne. Sam model przewiduje:

1	„szybką” budowę prototypu celem skonfrontowania go z klientem (jego wyobrażeniami o systemie);
2	rozwijanie pełnego systemu na podstawie prac nad prototypem.



Fazy modelu prototypowania



Prototypowanie – wybrane aspekty

Prototyp – wybrane aspekty:

- niepełna realizacja,
- języki wysokiego poziomu,
- technologie komponentowe,
- generatory interfejsów,
- inne, proste środki.

Prototypowanie można uznać za bardzo specyficzną, rozbudowaną formę analizy wymagań. ?

Model prototypowania – uwagi

- Należy silnie podkreślić, że budowany prototyp nie jest częścią przyszłego systemu. Prototyp po akceptacji przez klienta jest porzucany, a budowa samego systemu rozpoczyna się od nowa.
- Oczywiście w praktyce prototyp może być bardzo udany i możliwe jest wykorzystanie – np. w dużych fragmentach – prototypu w pracach właściwych.
- W budowie prototypu należy dążyć do szybkiej jego realizacji, np. kosztem jego efektywności czy niezawodności. Być może dlatego w literaturze – chyba niesłusznie – mówi się tutaj o tym modelu jako o **szybkim prototypowaniu**.
- Prototypowanie można uznać za pewnego rodzaju mutację cyklu kaskadowego – różnicą jest tu odmienne podejście poprzez budowanie makiety systemu, sam system może być tworzony z wykorzystaniem modelu wodospadowego.

Model prototypowania – uwagi (cd.)

Należy podkreślić, że

- niepełna realizacja – tylko część wymagań może być trudna i związana z prototypem;
- technologie komponentowe – coraz bardziej popularne;
- generatory interfejsów – prototyp może być związany (tylko) z budową interfejsu, reszta nie wymaga prototypowania;
- inne, proste środki – nawet w pewnych sytuacjach „technologie” papier-ołówek – jest to szybka i lubiana przez klientów forma prototypowania.

Jeżeli prototypowanie można uznać za bardzo specyficzną, rozbudowaną formę analizy wymagań, to tak jabyśmy wracali do modelu kaskadowego.

Prototypowanie – podsumowanie

Podstawowymi celami tego modelu jest:

- wykrycie potencjalnych nieporozumień pomiędzy stronami projektu ze względu na wymagania;
- wykrycie brakujących funkcjonalności systemu;
- wykrycie trudnych wymagań niefunkcjonalnych;
- wykrycie wszelkich błędów w specyfikacji wymagań.

Prototypowanie – podsumowanie (cd.)

Jako zalety – poza powyższymi – można wymienić:

- możliwość szybkiej prezentacji pracującej wersji systemu;
- możliwość rozpoczęcia szkoleń jeszcze przed ukończeniem systemu.

Podstawowymi wadami modelu jest:

- konieczność budowy prototypu, powiększająca koszty ogólne przedsięwzięcia;
- niezrozumienie ze strony klienta wynikające z niewiary w konieczność budowy prototypów.

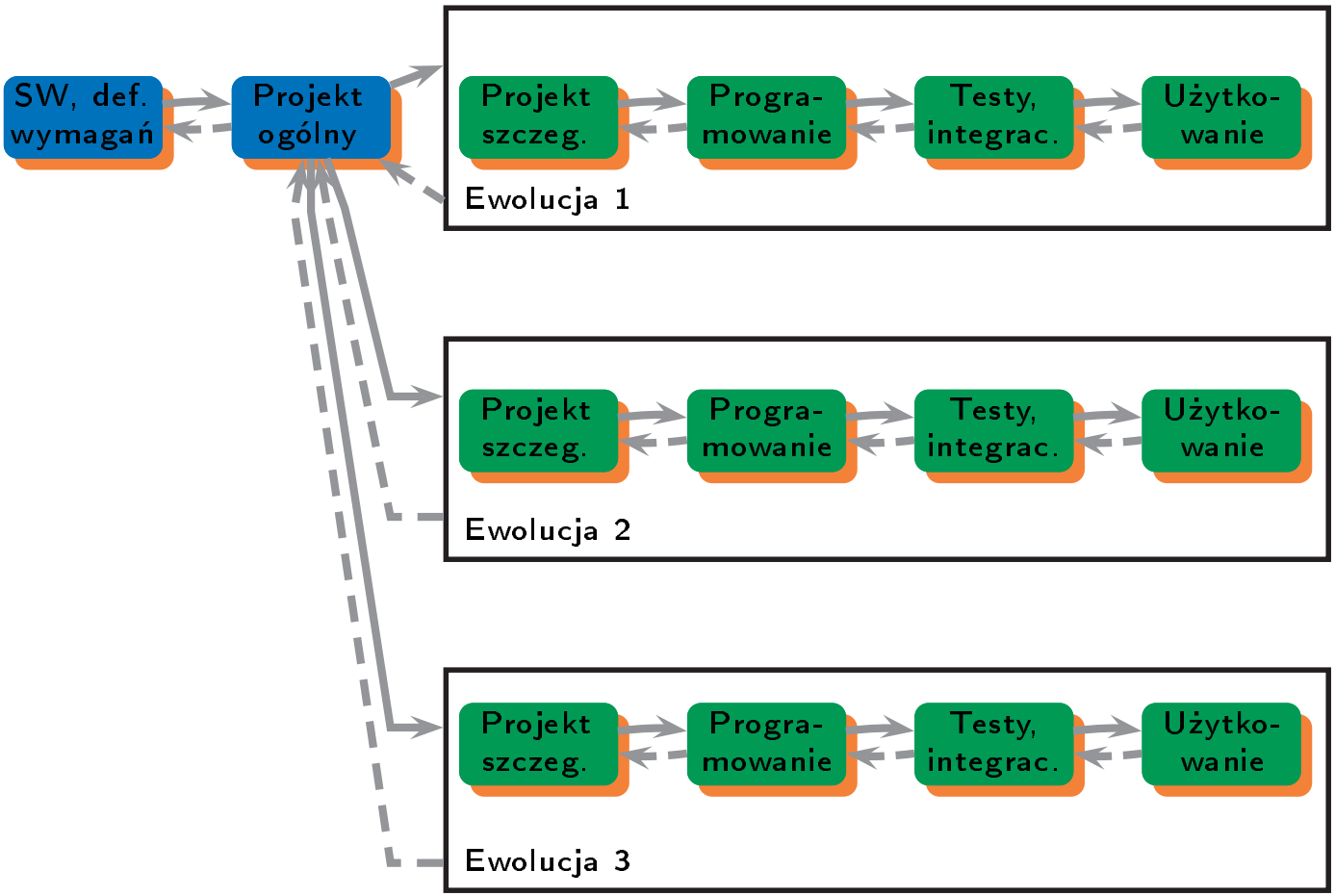
Jeśli chodzi o wady, to klient może w efekcie być bardzo zdziwionym, że tak długo czekał i musiał sporo (ze względu na prototyp) płacić za system, który został wykonany stosunkowo szybko. Jednak „szybko", o ile nie widzi się konieczności budowy prototypu.

Model przyrostowy (ewolucyjny)

Model przyrostowy – zaproponowany w 1980 przez H. Millisa i in. – jest pewną kontynuacją modelu kaskadowego:

- składa się z części początkowej, zakończonej projektem wysokiego poziomu, oraz części ewolującej;
- możliwość skrócenia przerw w kontaktach z klientem i dostarczania mu fragmentów (porcji) systemu;
- większe koszty realizacji wynikające z przyrostowego wykonywania systemu.

Model przyrostowy – ilustracja



Model przyrostowy – uwagi

- Model przyrostowy jest związany zasadniczo z wyborem pewnych funkcjonalności systemu, które są rozwijane w danej ewolucji. Każda nowa wersja jest rozwiniętą wersją poprzedniej.
- Nie jest możliwe w pełni niezależne „wycięcie" wybranych funkcjonalności, bez wpływu na pozostałe – takie postępowanie zwiększa koszty całości.
- Klient jest „wciągany" w realizację systemu. Ma także możliwość zmian w wymaganiach.
- Trudność zarządzania takim modelem. Krytycznym może się stać przekroczenie tempa napływających zmian „rosnącego" produktu.

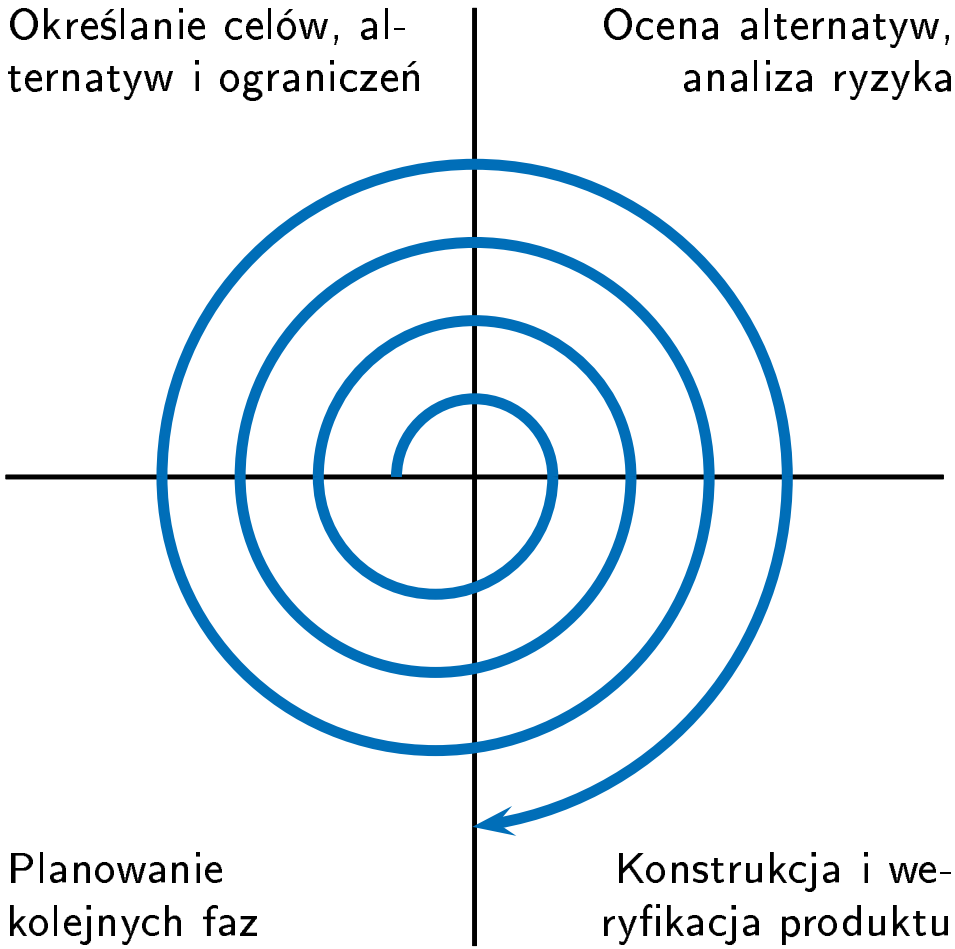
Model spiralny

Podstawowe fakty:

- **model spiralny** (ang. *spiral model*) został zaproponowany przez B. Boehma w 1988 r. i był pomyślany jako model dość ogólny;
- model składa się z czterech kolejnych faz w każdym pojedynczym cyklu;
- odległość punktu na krzywej od środka układu obrazuje akumulację dotychczas poniesionych nakładów.

B. Boehm: A spiral model for software development and enhancement. *Computer*, May 1988, vol. 21, no. 5, pp. 61–72.

Model spiralny – pierwsza ilustracja



Fazy modelu spiralnego

Na cały projekt składa się szereg pełnych obrotów rozkręcającej się spirali.

lewa, górna ćwiartka	cele bieżącego obrotu, definiowanie ograniczeń dla planowanych celów, wyznaczanie wariantów alternatywnych;
prawa, górna ćwiartka	ocena wariantów alternatywnych, identyfikowanie i szacowanie ryzyka, wybór właściwego sposobu wytworzenia produktu;
prawa, dolna ćwiartka	przejście produktu przez proces wytwarzania, a także jego weryfikacja i walidacja;
lewa, dolna ćwiartka	przygotowanie i planowanie realizacji kolejnych faz – prace nad produktem są kontynuowane.

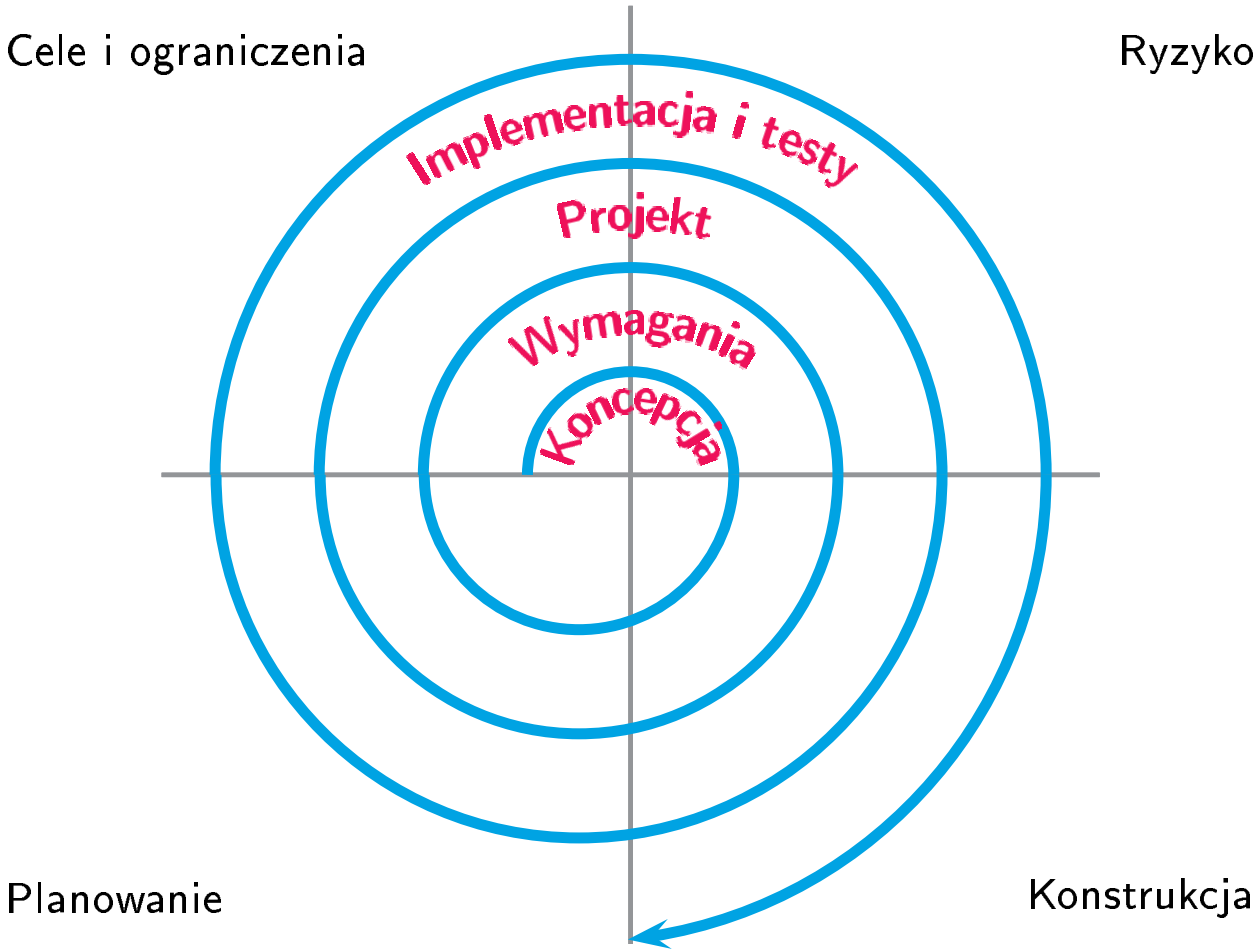
Model spirali jest dość ogólny i może być różnie interpretowany, np. kolejne obroty jako kolejne prototypy produkowanego systemu.

Model spiralny – uwagi

- Można powiedzieć, że celem modelu spiralnego jest uświadomienie zarządzającym wagi planowania prac nad projektem oraz aktywnego zarządzania ryzykiem.
- Kierownicy muszą sobie uświadomić, że sukces projektu zależy nie tyle od ścisłego przestrzegania planu, lecz od kompetentnego kierowania (nieuchronnymi) zmianami w prowadzeniu projektu – model spirali umożliwia wielokrotne powtarzanie (przemyśliwanie, poprawianie) tych czynności.
- Model spirali może być interpretowany jako budowa kolejnych prototypów. Inną interpretacją jest realizowane w kolejnych obrotach nowych funkcjonalności systemu. Generalnie jednak nie jest to istotą modelu, a jest nią – wspominane – skupienie uwagi na planowaniu przedsięwzięcia i zarządzaniu ryzykiem.

Czy model spiralny jest jakby modelem meta?

Model spiralny – prosty przykład



Model spiralny – czynności

Typowe czynności występujące w modelu spiralnym:

- koncepcja podejmowanych operacji;
- analiza wymagań;
- projektowanie systemu;
- szczegółowe projektowanie;
- kodowanie;
- testowanie modułowe;
- integracja i testowanie;
- testy akceptacyjne.

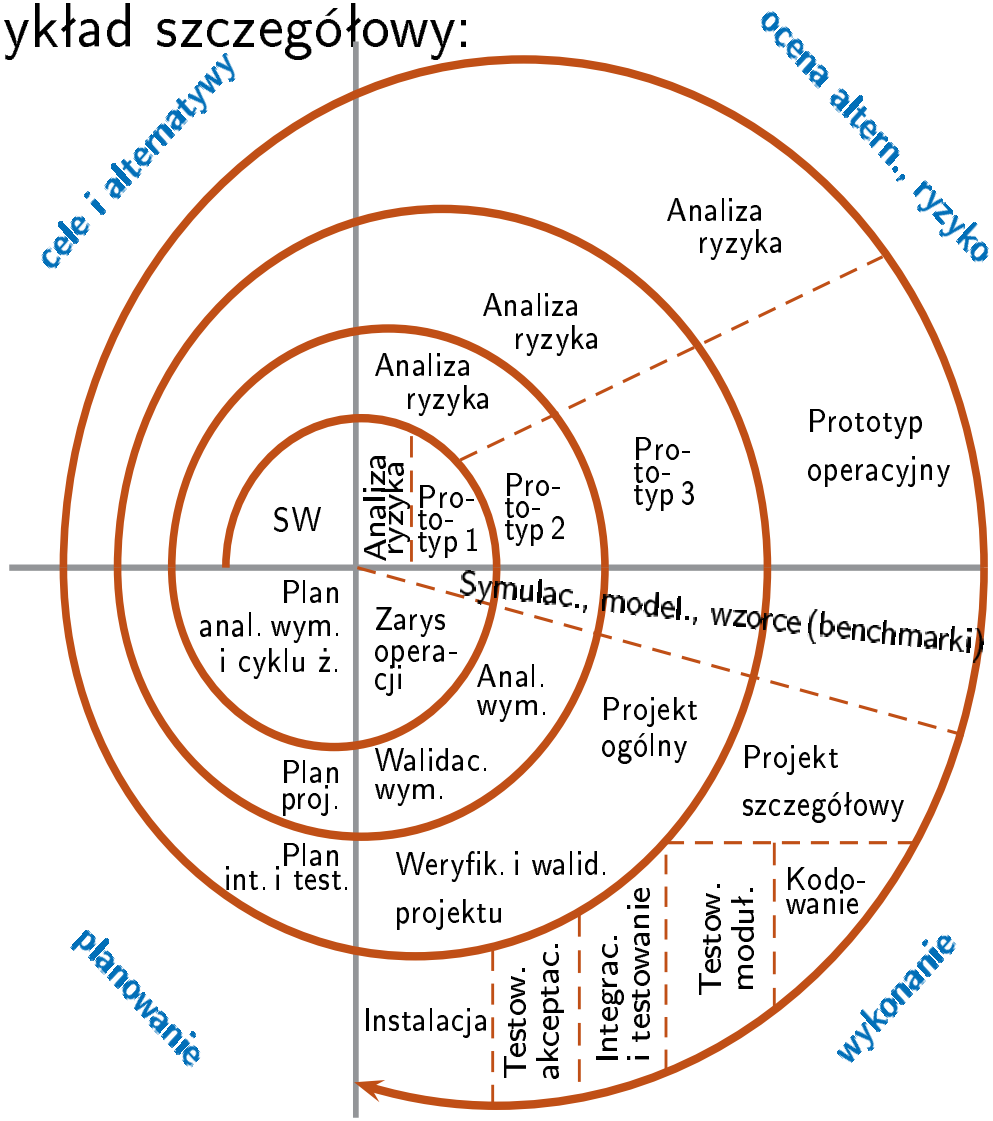
Analiza ryzyka w modelu spiralnym

Jednym z ważniejszych aspektów modelu spiralnego – odróżniającym go od innych modeli – jest analiza ryzyka, prawdopodobnie tutaj po raz pierwszy tak wyeksponowana w samym modelu cyklu życia.

Powszechnie uznaną zaletą modelu spiralnego jest analiza ryzyka;
Możliwość przypisania priorytetów do poszczególnych kategorii ryzyka;
Przewiduje się serię prototypów dla zidentyfikowanych kategorii ryzyka, począwszy od ryzyka najwyższego;
Dla każdego prototypu może być wykorzystany model wodospadowy;
Jeśli problem ryzyka zostanie rozwikłany z powodzeniem, to oceniane są rezultaty i planowana jest następna runda;
Jeśli pewien problem związany z ryzykiem nie został rozwikłany, to projekt może być zakończony.

Model spiralny – przykład szczegółowy wg. Boehma

Dobrze znany przykład szczegółowy:



Klasy prototypów modelu spiralnego

W modelu spiralnym można wyróżnić kilka kategorii występujących prototypów:

- **prototyp ilustracyjny**
  projektowanie interfejsów użytkownika, przeznaczeniem prototypu są pierwsze kontakty z klientem;
- **prototyp funkcjonalny**
  implementacja działającego systemu z minimum funkcjonalności, później następuje dodawanie nowych funkcjonalności (w kolejności wynikającej np. z analizy ryzyka);
- **prototyp eksploracyjny**
  implementacja fragmentu systemu w celu weryfikacji wymagań (wydobycia wymagań);

Klasy prototypów modelu spiralnego (cd.)

- **prototyp rewolucyjny**
  zwany także prototypem specyfikacji, przeprowadzenie użytkownika lub klienta przez cały system celem zagwarantowania poprawności wymagań;
- **prototyp ewolucyjny**
  prototyp wykorzystywany już jako podstawa implementacji systemu docelowego.

Tak więc, w zależności od sposobu prowadzenia projektu, od zaawansowania prac nad systemem możemy mówić o różnych kategoriach prototypów występujących w modelu spiralnym.

Najważniejsze wnioski i uwagi (1)

Poszczególne modele różnią się przydatnością w różnych sytuacjach – istnieją (liczne) mutacje prezentowanych cykli;
Model kaskadowy wydaje się dobry w sytuacji produkcji systemu o dobrze zdefiniowanych wymaganiach, w przypadku dobrze rozumianej dziedziny zastosowań;
Model kaskadowy może być lubiany przez zarządzających – np. łatwo wskazać punkty kontrolne (ang. *milestones*);
Model V umożliwia polepszenie zarządzania procesem rozwijania systemu i przechodzenie od abstrakcji do konkretów w połączeniu z procesami weryfikacji zawartymi w prawej gałęzi modelu;
Model V nie uwzględnia zależności czasowych i logicznych, w szczególności nie jest także modelem iteracyjnym;
Modele „ząb piły” oraz „ząb rekina” są reakcją na odmienne potrzeby stron projektu odnośnie percepcji samego projektu i systemu – dopasowanie trajektorii postrzegania;
Prototypowanie jest wskazane w przypadku budowy systemu obarczonej dużym ryzykiem;
Prototypowanie znacznie utrudnia szacowanie czasu i kosztów;
Przy prototypowaniu wadą modelu jest to, że prototyp może czasem stać się samym systemem;

Najważniejsze wnioski i uwagi (2)

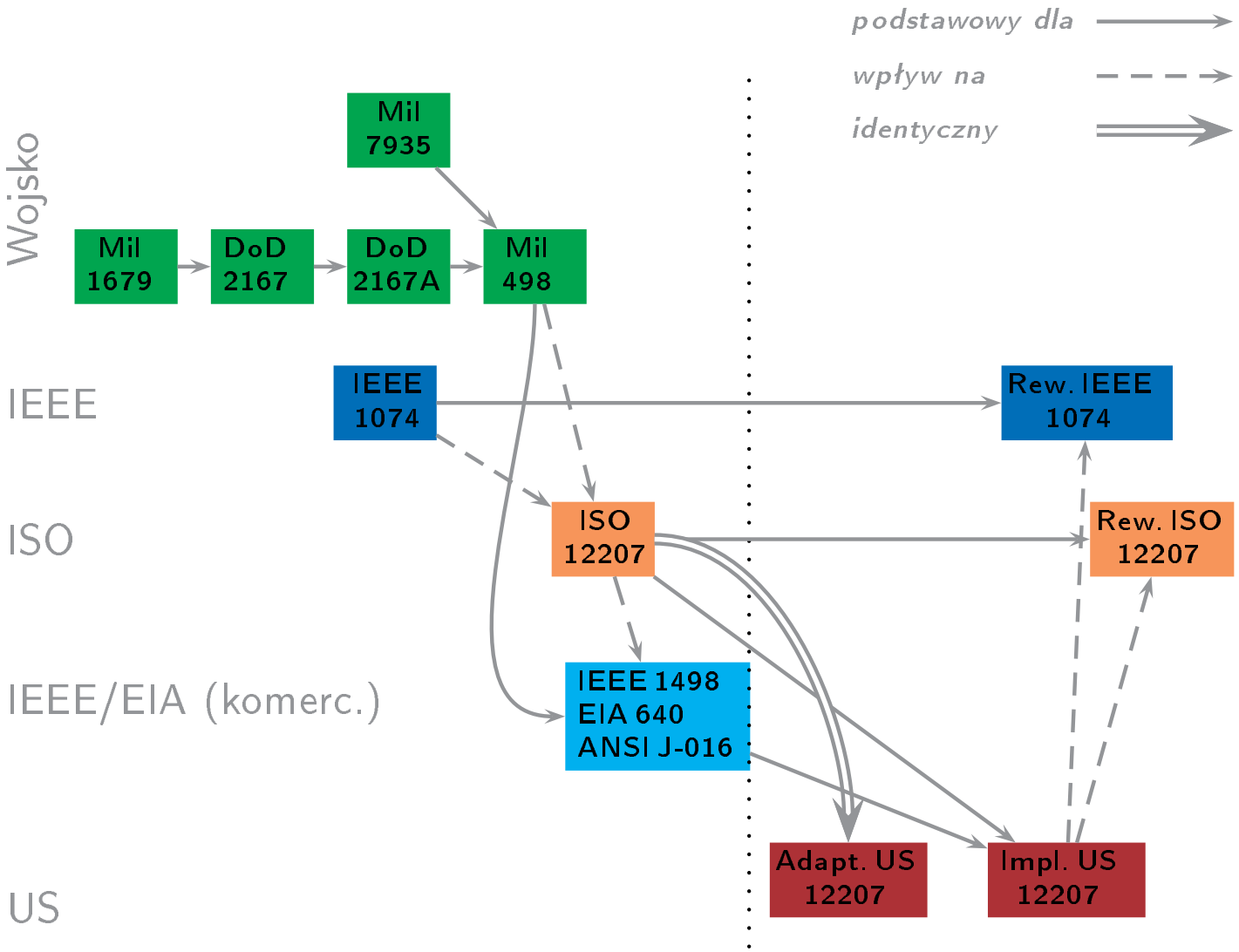
Model przyrostowy jest przydatny w sytuacji możliwości wyróżnienia i zaakceptowania początkowych funkcjonalności systemu – jako pierwszej jego wersji – z opcją dodawania kolejnych tzw. przyrostów – np. zastosowania bazodanowe;
W modelu przyrostowym poszczególne fazy cyklu są rozciągnięte na cały proces tworzenia oprogramowania;
Model przyrostowy szybko daje działający produkt, co może być mile widziane przez klienta, ale sam system i jego budowa wymaga tzw. otwartej architektury;
Model spiralny, poprzez nacisk na analizę alternatyw oraz ryzyka (często pomijaną), wskazywany jest do produkcji wieloegzemplarzowej (choć jednak nie tylko);
Analiza ryzyka (model spiralny) wymaga sporej praktyki i doświadczenia stąd też model spiralny wydaje się być wskazany dla dużych projektów;
Model spiralny, kładzie także nacisk na jakość, wyróżnia punkty kontrolne w przedsięwzięciu;
Ani modelu wodospadowy, ani model spiralny nie odnosi się do sytuacji (możliwych) częstych zmian w projekcie, w trakcie budowania systemu;

.....

Najważniejsze wnioski i uwagi (3)

Wybór właściwego cyklu życia jest zadaniem zarządczym i należy do kierownictwa projektu;
Wybór właściwego cyklu nie jest ogólnie łatwy – warto jednak zawsze wiedzieć przed rozpoczęciem prac, jaki cykl został przyjęty, z drugiej jednak strony zauważa się pewną tendencję do zbyt sztywnego trzymania się zaakceptowanego cyklu życia;
W gruncie rzeczy istnieją tylko dwa cykle życia systemu, a mianowicie model wodospadowy oraz model spiralny;
Patrząc jednak z innej strony modelami ewolucyjnymi są model przyrostowy i model spiralny, pozostałe już raczej nimi nie są.
W modelu kaskadowym zakłada się, że to co już zostało zrobione w fazie zostaje zamknięte i raczej nie jest otwierane; natomiast w modelu spiralnym zmiany pomiędzy fazami są możliwe, ale wewnątrz fazy są niedopuszczalne;
Co robić jeśli zmiany są częste? Jak zarządzać zmianami w takich sytuacjach? – pytania pozostają otwarte.

Cykl życia oprogramowania – ewolucja standardów



Z raportu Jima Moore'a.

Standard ISO 12207

Zgodnie z tym standardem:

Definicja

Model cyklu życia (ang. *life cycle model*) stanowi ramę, która zawiera procesy, czynności oraz zadania dotyczące rozwoju, użytkowania i konserwacji wyrobu programowego, a obejmuje czas życia liczony od zdefiniowania założeń systemu aż do zakończenia jego użytkowania, czyli wycofania z użycia. ⌟

International Standard ISO/IEC 12207: *Information technology – Software life cycle processess*. Genewa 1995.

Standard ISO 12207 – procesy

Standard ISO 12207 – uwagi

- Zaletą standardu jest, że „widzi" on obie strony cyklu życia, tj. **zleceniodawcę** oraz **zleceniobiorcę**.
- Razem siedemnaście procesów – można powiedzieć, że celem standardu jest ‚skrojenie" tych siedemnastu procesów do rzeczywistego procesu wytwarzania oprogramowania.
- Procesy cyklu podstawowego dzielą się na czynności (aktywności), a te na zadania.
- Trzy główne kategorie procesów stanowią bardzo szerokie tło procesu powstawania oprogramowania, obejmują wiele jego aspektów (np. dwa pierwsze procesy z procesów podstawowych), itd.

Procesy podstawowe standardu

W standardzie ISO 12207 zostało wyróżnionych pięć następujących **podstawowych procesów cyklu życia** (ang. *primary life cycle processes*):

1. **proces akwizycji** (ang. *acquisition process*) – rozpoczyna się z chwilą podjęcia decyzji o staraniu się o system informatyczny, obejmuje ogłoszenie przetargu, określenie wstępnych wymagań i uwarunkowań, przygotowanie treści kontraktu, monitorowanie dostawcy, procedura akceptacji produktu;
2. **proces dostarczania** (ang. *supply process*) – rozpoczyna się od podjęcia starań o zlecenie, obejmuje odpowiedź na ofertę, plan przedsięwzięcia, szacowane koszty, propozycje zakupów, szkoleń czy konsultacji, skład zespołu realizacyjnego, lista referencyjna, a po podpisaniu kontraktu ustalenie cyklu życia oraz sposobu zarządzania przedsięwzięciem;

Procesy podstawowe standardu (cd.)

3. proces rozwoju (ang. *development process*) – obejmuje pełny **cykl rozwojowy oprogramowania** (ang. **software development cycle**) i precyzuje czynności dostawcy po sformułowaniu kontraktu, ogólnie stanowi proces przekształcania specyfikacji wymagań nabywcy w przekazywany mu wyrób programowy;
4. proces użytkowania (ang. *operation process*) – wyszczególnione działania operatorskie, które są wykonywane w środowisku użytkownika, ustanowienie procedur informowania o błędach i problemach, procedury okresowych testów w trakcie eksploatacji;
5. proces konserwacji (ang. *maintenance process*) – wszelkie działania związane z utrzymywaniem produktu (w długim czasie), jego korygowaniem i doskonaleniem, także modyfikacje wynikające z poszerzenia zakresu funkcjonalności systemu, czy też przenoszenia go do innego środowiska sprzętowo-programowego.

Procesy podstawowe standardu – uwagi końcowe

Procesy (podstawowe) $\Rightarrow$ czynności $\Rightarrow$ zadania. Sama norma opisuje także dość precyzyjnie podstawowe fazy procesu rozwoju (budowa systemu). Pozostałe procesy – tj. procesy wspomagające oraz procesy organizacyjne – nie będą tutaj szczegółowo omawiane.

Standard ISO 12207 proponuje szereg norm i wzorców dokumentów na różnych etapach procesu powstawania oprogramowania.