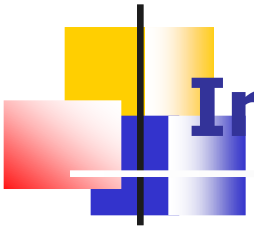


Inżynieria oprogramowania

Grzegorz Młynarczyk

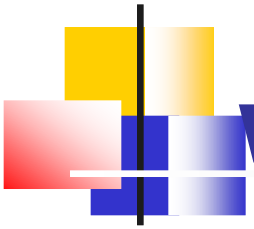
Jacek Kołodziej

Semestr IV

Wykład 30h, Laboratorium 30h

Semestr V

Wykład 30h, Projekt 30h, Egzamin



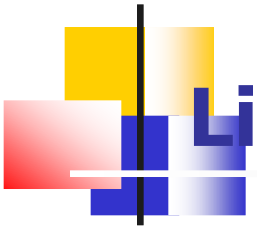
Warunki zaliczenia

- Semestr IV
  - Liczba nieusprawiedliwionych nieobecności na zajęciach lab. <=2
  - Obecność na wszystkich kolokwiach
  - Średnia z kolokwiów > 2.75
  - Przewidywany jeden termin poprawkowy dla wszystkich zainteresowanych pod koniec semestru

Niespełnienie powyższych wymagań jest równoznaczne z **brakiem zaliczenia** !

Literatura (1/3)

1. Bezier B. „Software Testing Techniques”, 2nd edition, Van Nostrand Rheinhold
2. Boehm B. W. „A spiral model of software development and enhancement”, IEEE Computer, 21, 61-72
3. Booch, G. „Object Oriented Analysis and Design with Applications”
4. Davis A. M. „Software Requirements: Analysis and Specification”, Prentice-Hall
5. Górski J. „Inżynieria Oprogramowania w projekcie informatycznym”, Mikom
6. Jackson M.A. „Requirements and Specifications”, Addison-Wesley

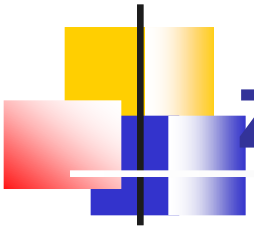


Literatura (2/3)

1. Lehman M. M. „Program Evolution: Process of Software Change”, Academic Press
2. Motet G, Fleurquin R. „Wprowadzenie do problematyki jakości oprogramowania. Normy ISO 9000 i programy jakości”, CCATIE
3. Prieto-Diaz R. „Domain Analysis and Software System Modeling”, IEEE Press
4. Wirth N. „Program development by stepwise refinement”, ACM 14
5. ISO-12207, „Information Technology – Software Engineering, Software Life Cycle Process”, ISO
6. Tom DeMarco - „Controlling Software Projects”

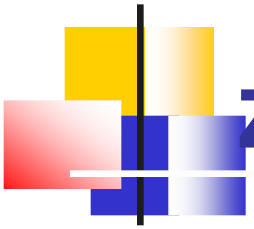
Literatura (3/3)

1. IFPUG „Function Point Counting – Practices Manual” Release 4.1.1 2000
2. Garmus D., Herron D. „Measuring the Software Process: A Pratical Guide to Functional Measurements”, Prentice Hall 1996
3. Boehem B. W. „COCOMO II Model Definition Manual” Version 1.5, University of Souther California 1998
4. Auer K., Miller R., Cunningham W. „Extreme Programming Applied: Playing to Win”, Addison-Wesley Pub Co; ISBN: 0201616408; 1st edition (October 1, 2001)
5. Hall E. M. „Managing Risk: Methods for Software Systems Development”, Addison-Wesley Pub Co; ISBN: 0201255928; 1st edition (February 1998)



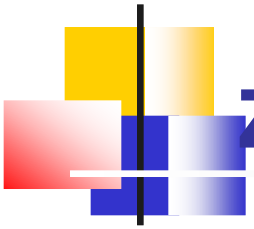
Zagadnienia (1/3)

- Pojęcie inżynierii oprogramowania - historia i obecne znaczenie, wprowadzenie terminologii (proces, projekt informatyczny, kierownik projektu itp.)
- Złożoność systemów informatycznych – oprogramowanie jako produkt, podział oprogramowania
- Proces tworzenia oprogramowania – omówienie podstawowych faz oraz ich znacznia i wpływu na końcowy produkt/oprogramowanie
- Podstawowe atrybuty dobrego oprogramowania
- Przykłady procesów wykorzystywanych w komercyjnych projektach przez liderów rynku oprogramowania



Zagadnienia (2/3)

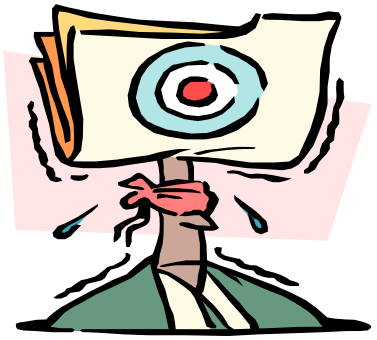
- Modele życia projektów (waterfall, V-model, model ewolucyjny, model spiralny Bohema)
- Wprowadzenie pojęcia zarządzania ryzykiem w projektach informatycznych
- Dokumentacja projektowa – podział, rodzaje dokumentów ich znaczenie z punktu widzenia członków grupy projektowej, zarządzania projektem oraz kierownictwa firmy. Wymiana informacji pomiędzy różnymi zespołami biorącymi udział w projekcie informatycznym
- Koncepcja inżynierii systemów informatycznych w odniesieniu do inżynierii oprogramowania



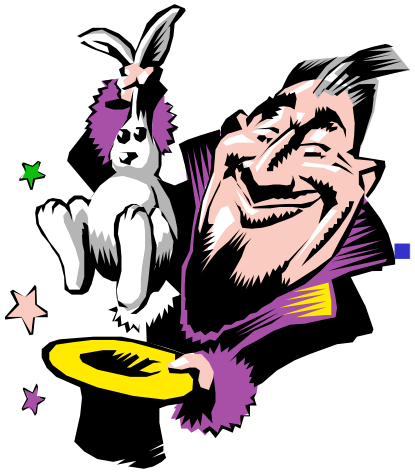
Zagadnienia (3/3)

- Wpływ otoczenia (sprzęt, oprogramowanie, ludzie) na systemy informatyczne
- Niezawodność systemów informatycznych.

Projekt informatyczny – podejście *ad hoc*



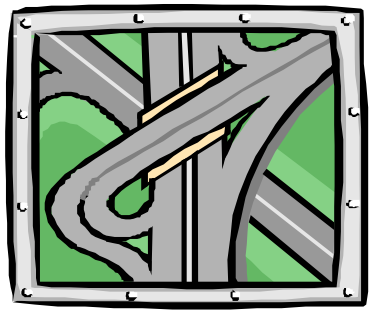
- Zbiór nieuporządkowanych działań wykonywanych przez grupę osób
- Brak wyraźnego celu oraz planu działania
- Kolejne funkcje systemu/moduły powstają w sposób przypadkowy a integracja z istniejącymi już częściami odbywa się w sposób niezaplanowany i bez informowania innych członków zespołu



- W przypadku gdy funkcjonalność systemu jest duża, grupa projektowa rozrasta się bez wyraźnie określonego celu dla poszczególnych jej członków a działania nie związane z główną czynnością w projekcie tj. **produkcją oprogramowania** gwałtownie rosną
- Projekty tworzone *ad hoc w większości przypadków kończą się z dużym opóźnieniem, z przekroczonym budżetem, nieustającą fazą testowania i poprawiania błędów, implementacji pominiętych wymagań a nikt nie jest w stanie powiedzieć w jakim właściwie stanie znajduje się **projekt** i jego **produkty***

W jaki sposób chcielibyśmy realizować projekty?

- Tak aby zaspokajać **potrzeby Klientów**
- W sposób:
  - zdyscyplinowany dający obraz aktulanego stanu projektu
  - dający się zarządzać i przewidywać (czas i budżet)
  - powtarzalny z określoną gwarancją poziomu jakości

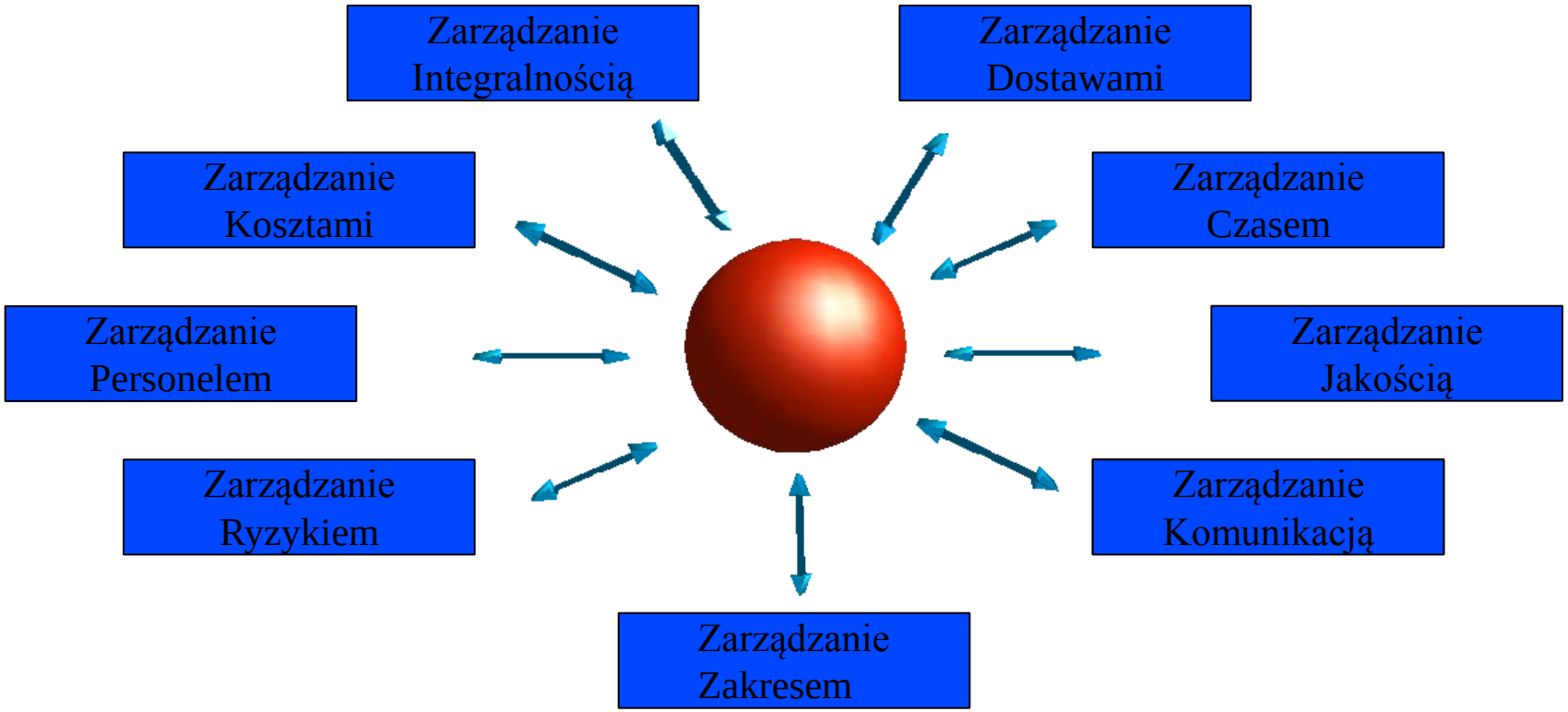


W tym celu staramy się wprowadzać i stosować **METODYKI** stanowiące próbę zastosowania pewnych sprawdzonych praktyk inżynierskich lub budowanie nowych na podstawie zgromadzonych doświadczeń i tzw. najlepszych praktyk (ang. best practice)

Inżynieria oprogramowania

- Pojęcie „Software engineering” wprowadzone pod koniec lat 60-tych
- **Inżynieria oprogramowania** – jawne wykorzystanie technik, metodyk i narzędzi do wytworzenia odpowiedniego oprogramowania (produktu) w ramach projektu informatycznego
  - Analiza
  - Implementacja
  - Zarządzanie
  - **Utrzymanie**
- **Projekt informatyczny** – zestaw zadań podlegających **zarządzaniu**, połączonych wspólnym **celem** osiaganym w ramach istniejących **ograniczeń**.

Projekt informatyczny – obszary podlegające zarządzaniu



Uproszczony model projektu – procesy

- **Klient**

Opis problemu

Specyfikuj problem

Specyfikacja problemu

Zident. rozwiąz.

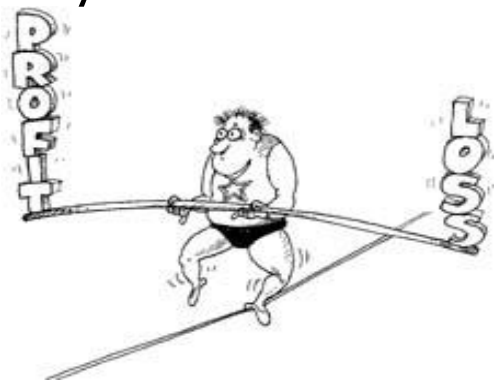
Specyfikacja rozwiązania

Zrealizuj rozwiąz.

Produkt

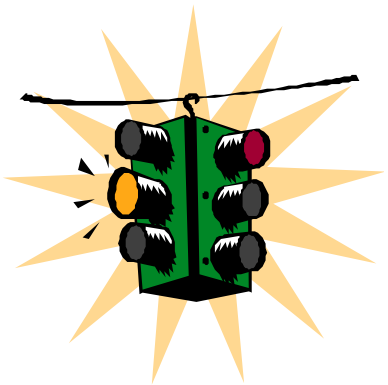
Projekt i produkt

- **Produkt** (system informatyczny) jest wynikiem realizacji **projektu**
- Koszt oprogramowania stanowi w wielu przypadkach dominujący czynniki kosztu całego systemu informatycznego, często znacznie większy niż np. koszt sprzętu
- Koszt utrzymania i zarządzania oprogramowaniem jest znacznie większy niż koszt jego wytworzenia - fazy specyfikacji i testowania mają ogromny wpływ na koszt utrzymania systemu
  - **Inżynieria oprogramowania** - sposób na efektywną budowę systemów informatycznych (w szczególności oprogramowania)



Plan projektu

- Sposób realizacji i analiza wykonalność projektu
- Zawiera informacje przekazywane udziałowcom
- Zakres prac i ich podział pomiędzy udziałowców
- Operacyjne prowadzenie projektu (harmonogram)
- Dokumentowanie założeń i wyborów alternatyw
- Oszacowanie czasu, zasobów, kosztów (budżet)
- Ocena ryzyka
- Kluczowe założenia dotyczące polityki zapewnienia jakości



W skrócie można powiedzieć, że plan projektu powinien odpowiadać na pytania: co należy zrobić?, z kim?, przy użyciu jakich zasobów?, w jaki sposób?, na kiedy?, za ile? i jakie podjąć działania, żeby wszystko się udało?

Oprogramowanie jako produkt - klasyfikacja

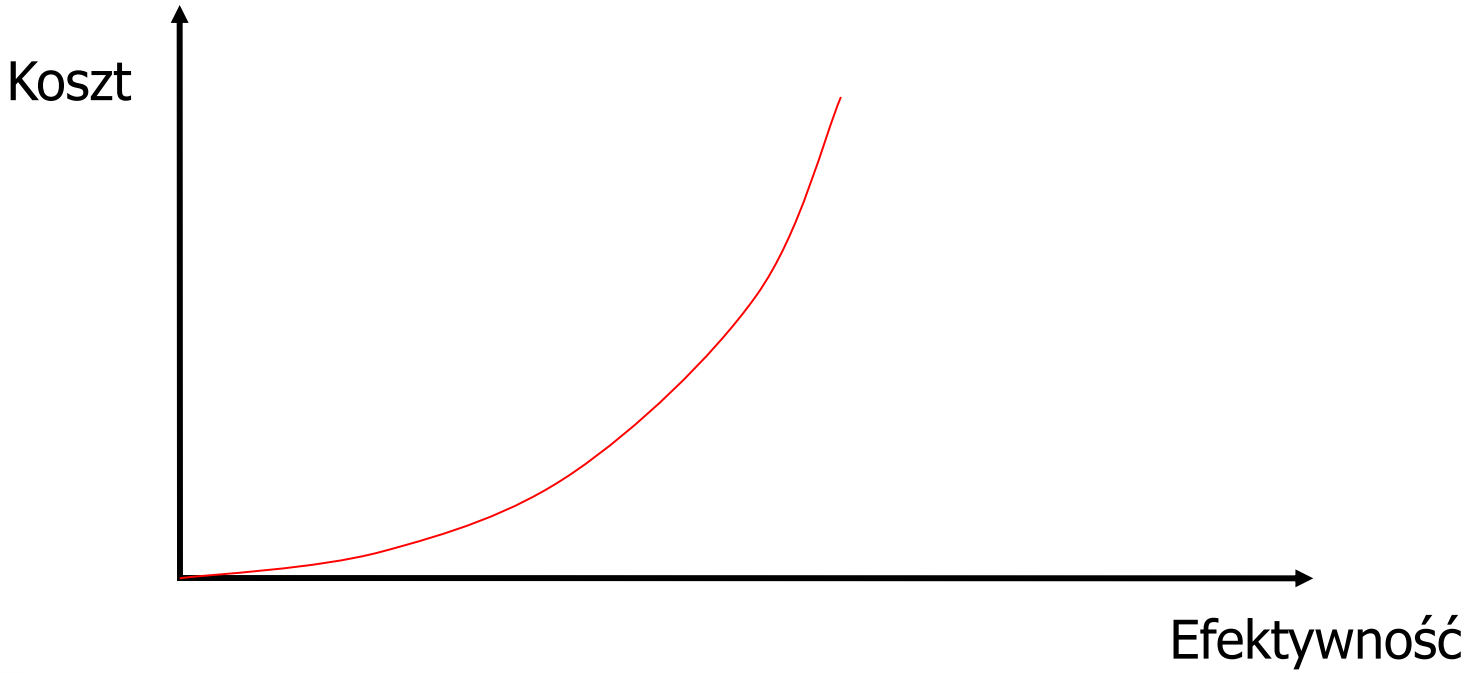
- Oprogramowanie ogólnego zastosowania tzw. produkty z półki - wytworzone i sprzedawne przez firmy dla szerokiego grona odbiorców (brak identyfikacji konkretnych potrzeb poszczególnych klientów)
- Oprogramowanie wytworzone na zamówienie i na potrzeby określonego klienta

Większość dostępnego oprogramowania należy do pierwszej kategorii ale największe nakłady przeznaczane są na produkty należące do kategorii drugiej

Atrybuty oprogramowania jako produktu

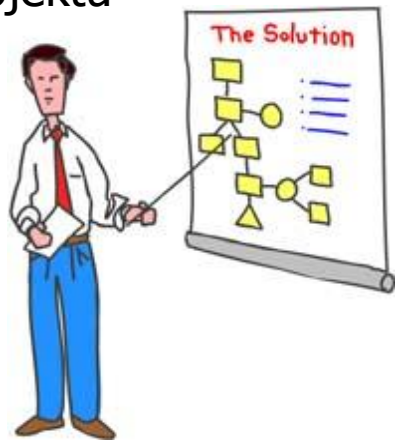
- **Zarządzalność** (ang. maintainability) - możliwość (łatwość, efektywność) dostosowania oprogramowania do zmieniających się wymagań klienta (np. możliwość pracy na innym sprzęcie, systemie operacyjnym)
- **Niezawodność** (ang. dependability) - niezawodność systemu rozumiana jako zdolność do obsługi żądań użytkowników oprogramowania, bezpieczeństwo, tolerancja awarii itp.
- **Efektywność** (ang. efficiency) - wydajność rozumiana jako użytkowanie zasobów (takich jak np. procesor, pamięć) w trakcie „normalnej" pracy oprogramowania
- **Użyteczność** (ang. usability) - łatwość korzystania z oprogramowania poprzez jego interfejs oraz odpowiednią dokumentację

Wpływ wzrostu efektywności na koszt produktu

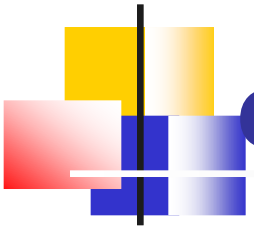


Proces budowy oprogramowania

- **Usystematyzowany**, **zorganizowany** w czasie zbiór **działań** prowadzący do wytworzenia oprogramowania w ramach projektu
- Podstawowe działania tworzące proces:
  - Analiza i specyfikacja wymagań
  - Projektowanie
  - Implementacja
  - Walidacja/Testowanie
  - Rozwój i utrzymanie



- Zakres i sposób realizacji poszczególnych działań może znacznie się różnić w zależności od firmy oraz rodzaju tworzonego oprogramowania
- Zarządzanie parametrami budowanego oprogramowania (jakość, efektywność) jak również przebiegiem realizacji projektu wymaga **jawnego** wyspecyfikowania procesu od samego początku projektu

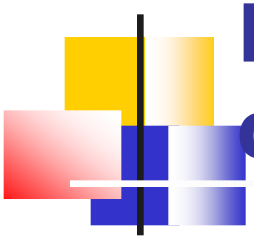


Główne atrybuty procesu

- **Zrozumiałość** – w jakim stopniu proces jest zdefiniowany i jak łatwo można zrozumieć poszczególne działania/dokumenty wchodzące w jego skład
- **Czytelność** – możliwość łatwego kontorolowania wyników poszczególnych działań relizowanych w ramach procesu oraz łatwość określania postępu/stanu projektu
- **Wsparcie** – w jaki sposób działania realizowane w ramach procesu mogą zostać wsparte poprzez wykorzystanie rożnego rodzaju narzędzi (CASE) – w jaki sposób wpływa to na jakość i efektywność oprogramowania
- **Niezawodność** – w jaki sposób proces pozwala na uniknięcie błędów przed wdrożeniem oprogramowania
- **Adaptowalność** – możliwość zmian procesu w ramach zmian organizacji lub jego poprawy wynikającej ze zgromadzonych doświadczeń
- **Efektywność** – szybkość dostarczenia produktu oraz koszt jego wytworzenia

Proces produkcji „zwykłego” produktu vs. proces budowy oprogramowania

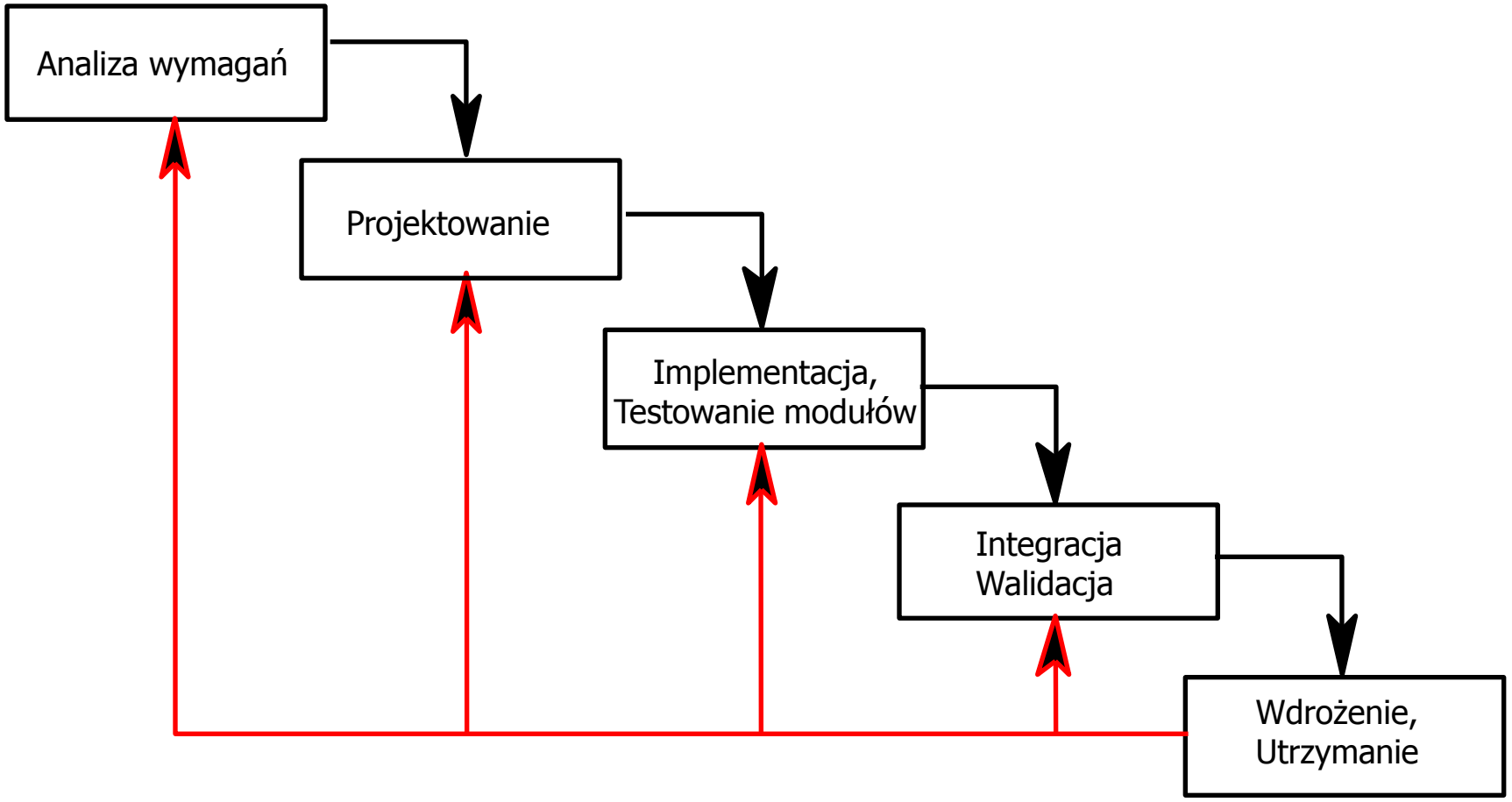
- **Specyfikacja** - określa zbiór wymagań i ograniczeń nałożonych na produkt – w przypadku oprogramowania ze względu na jego złożoność zwykle specyfikacja jest niekompletna lub wręcz błędna
- **Projektowanie** – stworzenie modelu opisującego system – w przypadku oprogramowania brak wzorców, zwykle każdy produkt posiada odmienne rozwiązania projektowe
- **Wytworzenie** – budowa systemu na podstawie projektu (w przypadku oprogramowania ciągle zmieniające się wymagania i projekt)
- **Testowanie** – sprawdzenie czy system spełnia wymagania i ograniczenia zawarte w specyfikacji – w przypadku oprogramowania brak „fizycznej” realizacji systemu praktycznie aż do momentu jego wdrożenia
- **Instalacja i utrzymanie** – dostarczenie systemu i naprawa błędu w momencie ich wystąpienia – w przypadku oprogramowania naprawa większości awarii nie może zostać zrealizowana tylko poprzez wymianę niektórych komponentów (w wielu przypadkach wymagana jest zmiana implementacji fragmentów lub nawet całości systemu)

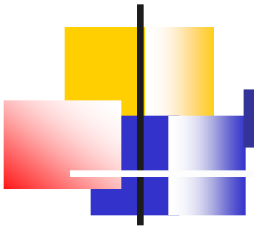


Modele procesu budowy oprogramowania (cyklu życia projektu)

- Kaskadowy (ang. waterfall)
- Model V
- Ewolucyjny (ang. evolutionary)
- Spiralny Bohema (ang. Bohem's spiral model)
- RUP
- eXtreme Programming

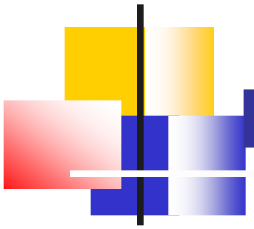
Model kaskadowy (1/3)





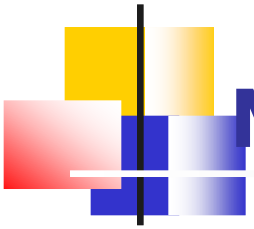
Model kaskadowy (2/3)

- **Analiza wymagań** – określenie wymagań i ograniczeń nałożonych na system przy współudziale Klienta. Przygotowanie specyfikacji stanowiącej podstawę dla etapu projektowania
- **Projektowanie** – stworzenie architektury systemu oraz wybór technologii, która zostanie wykorzystana do implementacji
- **Implementacja, Testowanie modułów** – realizacja projektu oraz weryfikacja czy stworzone moduły spełniają wymagania i założenia projektowe
- **Integracja i Walidacja** – połączenie wszystkich modułów w jeden system, testowanie czy system spełnia wszystkie wymagania zawarte w specyfikacji. Określenie czy zbudowany system spełnia oczekiwania Klienta (testy akceptacyjne)
- **Wdrożenie i Utrzymanie** – uruchomienie systemu w środowisku produkcyjnym Klienta oraz nadzór nad jego pracą (reagowanie na awarie, dostosowywanie do nowych wymagań Klienta itp.)



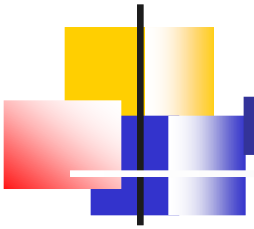
Model kaskadowy (3/3)

- **Zalety**
  - Prosta koncepcja odzwierciedlająca praktyki stosowane w innych procesach produkcji
  - Duża czytelność całości procesu – zakończenie poszczególnych etapów ułatwia raportowanie stanu projektu
- **Wady**
  - Brak jawnej weryfikacji pomiędzy poszczególnymi etapami
  - Przyjęcie założenie, że można stworzyć poprawną specyfikację już na samym początku projektu
  - Długi okres upływający od momentu stworzenia spececyfikacji do momentu dostarczenia/wdrożenia systemu
  - Brak zarządzania ryzkiem



Model kaskadowy – NASA

- Model kaskadowy stworzony został na potrzeby programów kosmicznych agencji NASA zbierając najlepsze praktyki i doświadczenia projektów informatycznych realizowanych w agencji (od 1976 r.)
- Opis metodyki zawarty został w dokumencie "Recommended Approach to Software Development" SEL-81-305
- Według metodyki NASA projekt wykonywany jest w 8 fazach: definicja wymagań, analiza wymagań, projekt wstępny, projekt szczegółowy, implementacja, testy integracyjne, testy akceptacyjne, utrzymanie
- Dla każdej fazy metodyka NASA określa:
  - Warunki rozpoczęcia i zakończenia
  - Kluczowe działania
  - Produkty poszczególnych faz i miary ich akceptowalności
  - Narzędzia wspomagające
  - Przeglądy wykonywane na zakończenie faz (wybranych)



Model V

Specyfikacja

Test. akcept.

Projektowanie

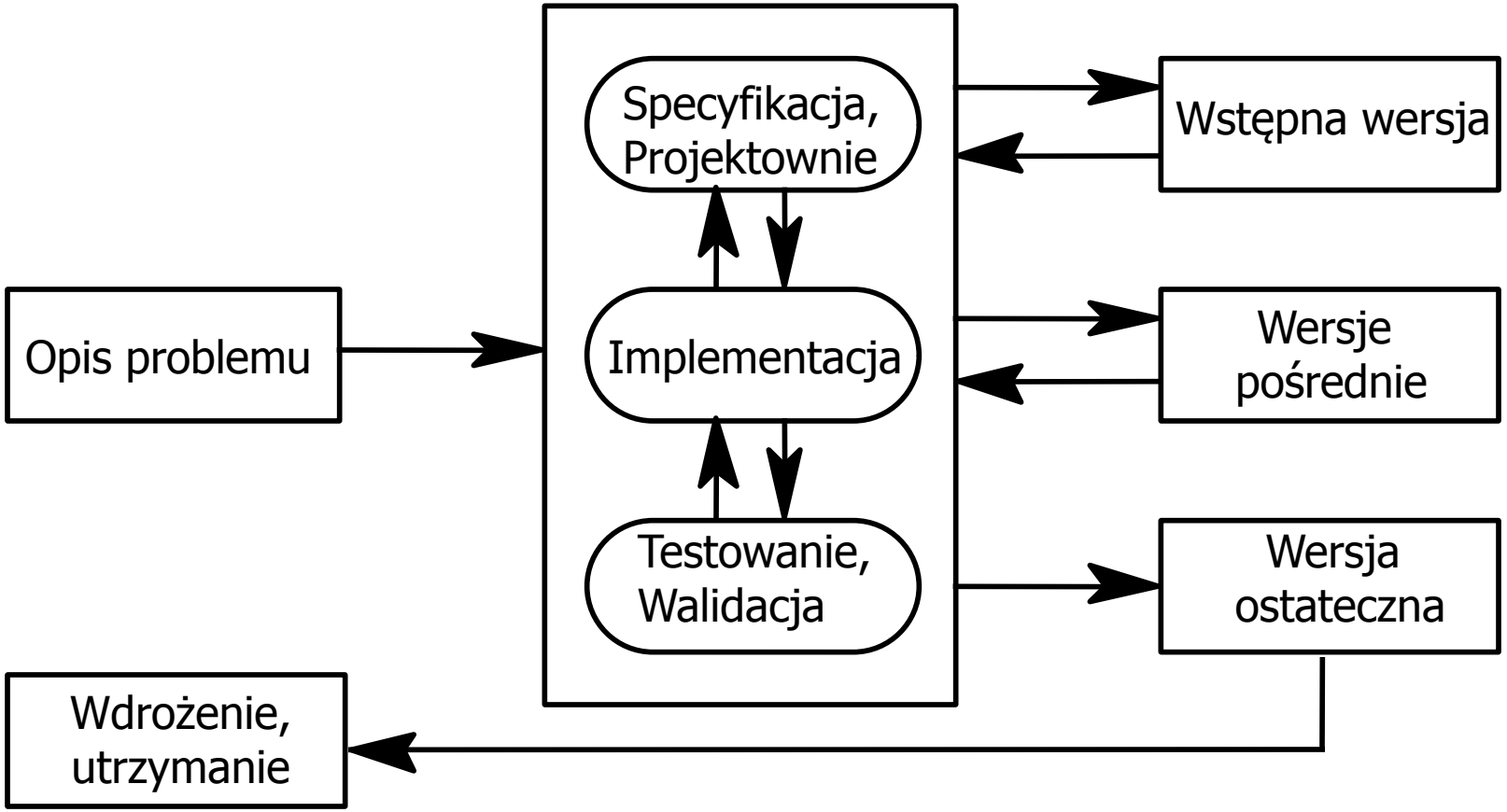
Integr. Walidacja

Projek. Modułu

Test. modułu

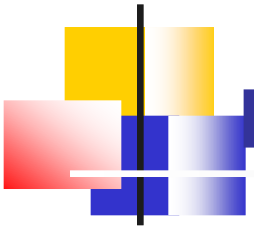
Implementacja,
wstępne testowanie

Model ewolucyjny (1/3)



Model ewolucyjny (2/3)

- **Prototypowanie** – lepsze zrozumienie wymagań, udowodnienie realizowalności założeń, modelowanie wymagań, których nie można dokładnie sprecyzować
- **Programowanie eksploracyjne** – stworzenie finalnego systemu rozpoczynając od implementacji modułów/wymagań najlepiej rozumianych w danej chwili. Realizacja całości systemu polega na implementacji kolejnych modułów/wymagań w miarę gdy zrozumienie pozostałej funkcjonalności i zadań systemu rośnie wraz z realizacją wcześniejszych funkcji. Kolejne wymagania implementowane są w systemie wraz z nowymi propozycjami Klienta. Programowanie eksploracyjne stosowane jest w przypadku, gdy dokładna definicja wymagań nie jest możliwa na początku projektu

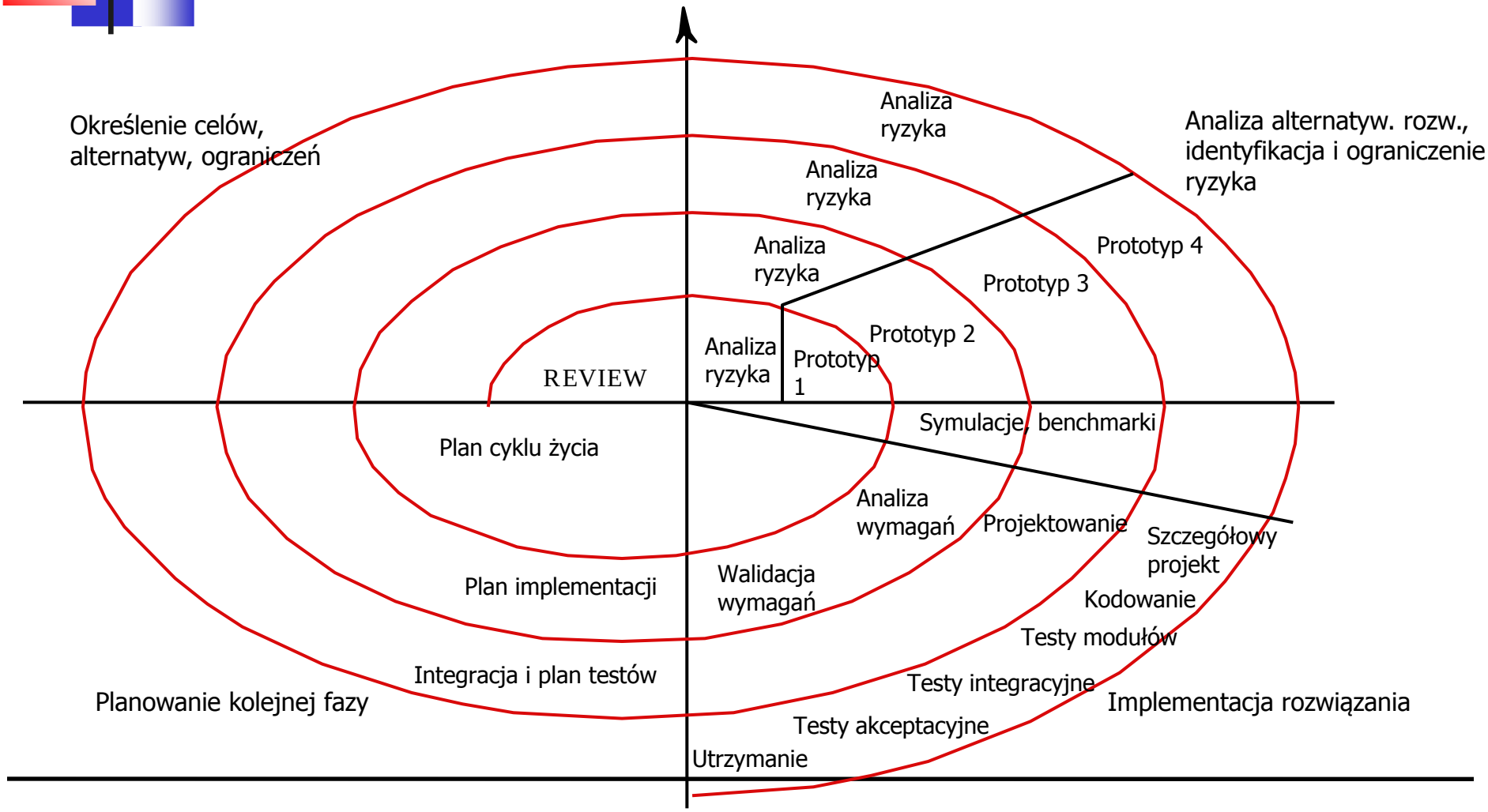


Model ewolucyjny (3/3)

- **Zalety**
  - Skrócenie czasu pomiędzy spec. systemu a wstępną walidacją wykonywaną przez Klienta
  - Zwiększenie szans powodzenia projektu poprzez spełnienie wielu wymagań Klienta, które nie zostały wprost wyrażone w spec.
  - Ograniczenie ryzyka związanego z błędną analizą wymagań

- **Wady**
  - Mała czytelność procesu
  - Duży nakład na zarządzanie zmianami wymagań
  - Trudne zarządanie całością projektu

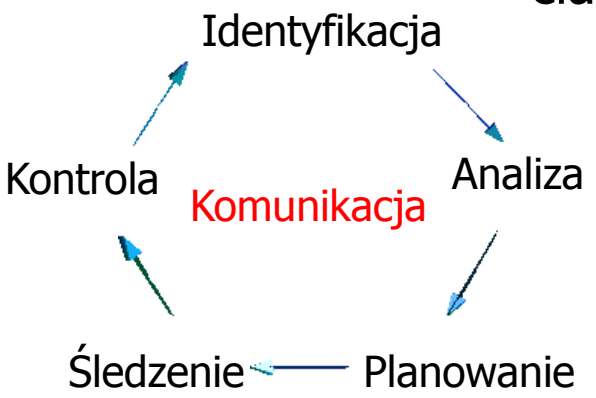
Model spiralny Bohema



Ryzyko w projekcie informatycznym

- **Ryzyko w projekcie** – jakiekolwiek działania, mogące powodować problemy w realizacji projektu
- Działania o dużym stopniu ryzyka mogą powodować znaczne opóźnienia i nieprzewidziane zwiększenie kosztów realizacji projektu
- Ryzyko związane jest z ilością i jakością dostępnej informacji. Im mniej dokładne informacje zostały zgromadzone tym większe ryzko wystąpienia problemów
- W pierwszej kolejności powinno się identyfikować problemy (ryzyko), które mogą mieć największy wpływ na realizację projektu i których prawdopodobieństwo wystąpienia jest największe

Zarządzanie ryzkiem w projekcie informatycznym



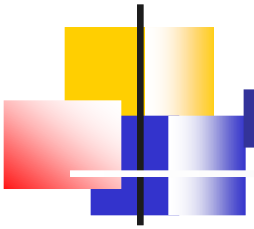
Zarządzanie ryzkiem powinno odbywać się w sposób ciagły:

- Określenie co może pójść źle w projekcie
- Określenie, które ryzyka powinny zostać uwzględnione w planie projektu
- Realizacja planu zarządzania ryzykiem (akcje podejmowane w celu ograniczenia i tłumienia skutków)

Bez odpowiedniej komunikacji zarządzanie ryzkiem w projektach informatycznych nie jest możliwe !

Zarządzanie ryzykiem w projekcie informatycznym

- **Identyfikacja – określenie możliwych zagrożeń oraz ich kategoryzacja**
- **Analiza – ocena prawdopodobieństwa wystąpienia zidentyfikowanych ryzyk oraz stopnia ich wpływu na projekt**
- **Planowanie – określenie planu umożliwiającego zmniejszenie prawdopodobieństwa wystąpienia poszczególnych ryzyk oraz skutków ich wystąpienia, definiowanie awaryjnych planów postępowania**
- **Śledzenie – monitorowanie i raportowanie aktualnego stanu projektu w odniesieniu do zdefiniowanych ryzyk**
- **Kontrola – sprawdzanie wpływu ryzyk na projekt, decydowanie o podjęciu określonych w planie działań zaradczych**



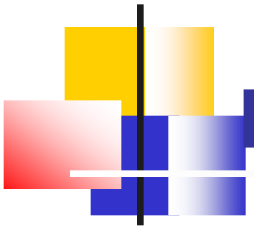
Model spiralny

- **Zalety**
  - Jawne zarządzanie ryzykiem
  - Możliwość wczesnej eliminacji błędów
  - Koncentracja na zarządzaniu jakością tworzonego systemu
  - Poszczególne fazy projektu mogą być realizowane z wykorzystaniem różnych, najbardziej odpowiednich modeli
  - Dobra czytelność

- **Wady**
  - Skomplikowany w porównaniu do np. modelu kaskadowego
  - Stosowany raczej do realizacji dużych systemów
  - Wymaga znajmości problematyki szacowania i zarządzania ryzykiem

Model spiralny – przyładowy formularz

- **Cel** – cel analizy
- **Ograniczenia** – ograniczenia nałożone na rozwiązania/system
- **Alternatywy** – alternatywne rozwiązanie prowadzące do realizacji celu
- **Ryzyko** – zidentyfikowane źródła ryzyka dla poszczególnych alterantyw
- **Metody minimalizacji ryzyka** – wszelkie działania mogące zmniejszyć prawdopodobieństwo wystąpienia problemu lub ograniczenia jego skutków
- **Rezultaty** – wnioski płynące z analizy poszczególnych alternatyw i związanego z nim ryzyka
- **Plan** – plan realizacji celu na podstawie analizy uzyskanych rezultatów
- **Zobowiązania** – decyzje kierownictwa co do dalszego postępowania



Przykładowa analiza ryzyka

- **Cel** – pozyskanie systemu katalogowania dla biblioteki
- **Ograniczenia** – w ciągu 1 roku, łatwa integracja z istniejącym syst., całkowity koszt 400 000 PLN
- **Alternatywy** – zakup istniejącego systemu, zakup systemu bazodanowego i implementacja systemu własnymi siłami
- **Ryzyko** – spec. systemu jest niekompletna, możliwość przekroczenia czasu realizacji przy spełnieniu innych ograniczeń
- **Metody minimalizacji ryzyka** – stworzyć prototyp w celu uzupełnienia i weryfikacji wymagań, zwiększyć czas realizacji, zamówić szczegółowy raport na temat istniejących systemów katalogowania zbiorów
- **Rezultaty** – istniejące roz. nie spełniają wymagań, implementacja całości systemu może nie spełnić oczekiwań ze względu na niekompletną spec., prototyp może zostać rozwijany o kolejne funkcje w miarę realizacji systemu
- **Plan** – implementacja prototypu w oparciu o wybrany system bazodanowy i narzędzia 4GL
- **Zobowiązania** – zatwierdzenie budżetu na natępne 12 miesięcy

Czytelność poszczególnych modeli procesów

Model procesu	Charakterytyka czytelność
Kaskadowy	Dobra czytelność, każdy z etapów kończy się dostarczeniem pewnej całości
Model V	Dobra czytelność, poprawiona efektywność całego procesu
Ewolucyjny	Słaba czytelność spowodowana szybkimi iteracjami; ekonomicznie nieuzasadnione tworzenie dokumentacji w trakcie szybko zmieniających się iteracji
Spiralny	Dobra czytelność, każdy obieg spirali kończy się wytworzeniem dokumentacji, dobre zarządzanie ryzykiem

Dokumentacja projektowa w modelu kaskadowym

Zadania	Dokumenty
Analiza wymagań	Definicja systemu, wstępny zbiór wymagań (Klient)
Specyfikacja systemu	Definicja wymagań, specyfikacja funkcjonalna, plan testów akcept., szkielet dokumentacji systemu
Projekt architektury	Projekt systemu, plan testów całości systemu
Projekt interfejsu	Specyfikacja interfejsu systemu, plan testów integracyjnych
Szczegółowy projekt	Projekt poszczególnych modułów systemu, plan testów modułów
Implementacja	Kod programu
Testowanie modułów	Raport z testów poszczególnych modułów
Testy integracyjne	Raport z testów integracyjnych
Testy systemu	Raport z testów całości systemu
Testy akceptacyjne	Raport z przekazania systemu, dokumentacja systemu