Budowa i integracja systemów informacyjnych

Wykład 1

Wprowadzenie do inżynierii oprogramowania

dr inż. Włodzimierz Dąbrowski

Polsko Japońska Wyższa Szkoła Technik Komputerowych

Katedra Systemów Informacyjnych, pokój 310

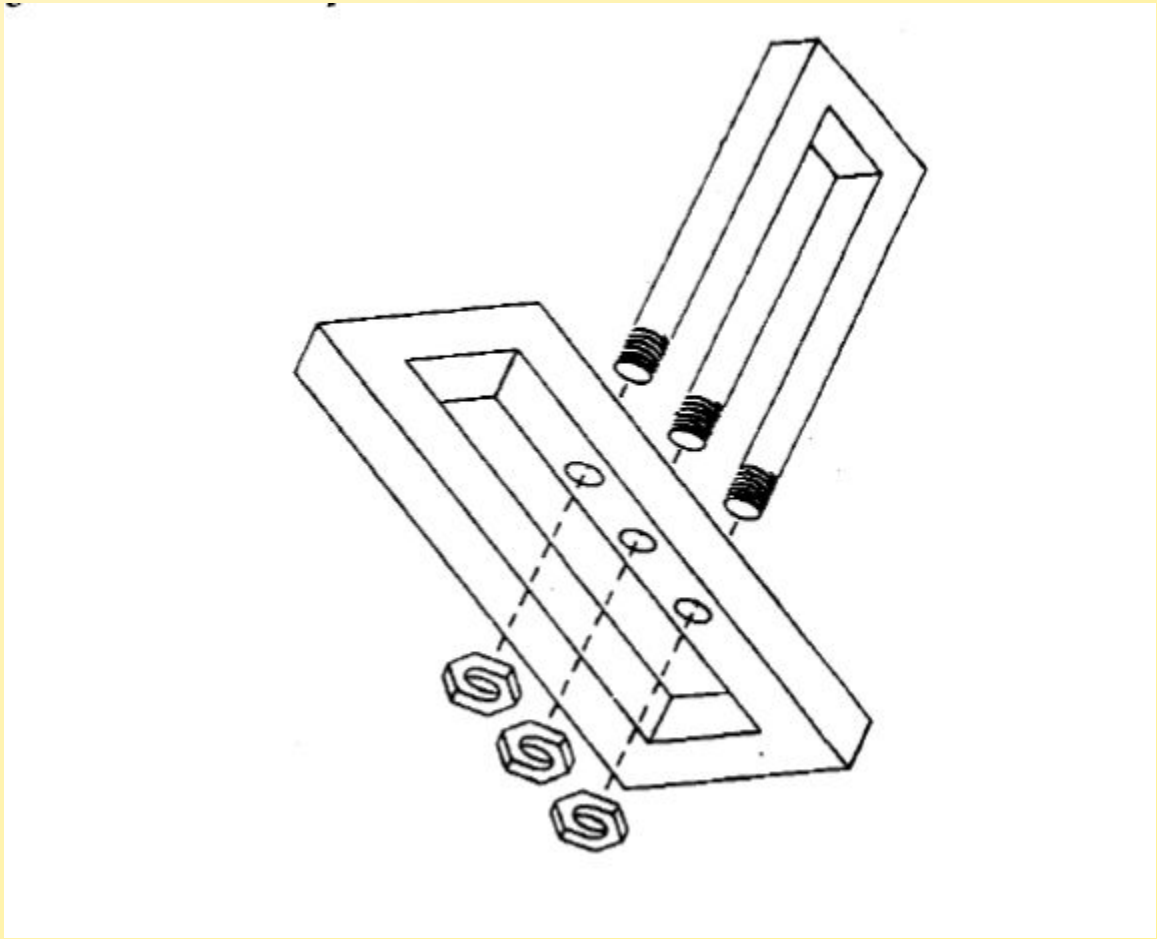
e-mail: Wlodek@pjwstk.edu.pl

Wersja PC

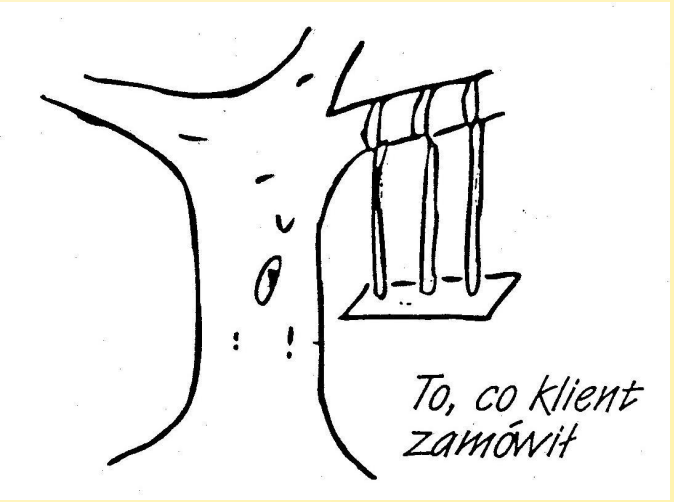
Plan wykładu

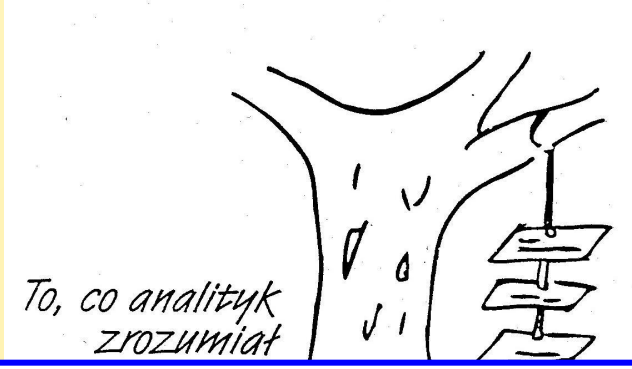
- Jak się uczyć?
- Jak zdać egzamin?
- O czym *to* jest?
- Czy projekty IT to „dobry interes”?
- Modele – co to takiego?
- … i po co …?

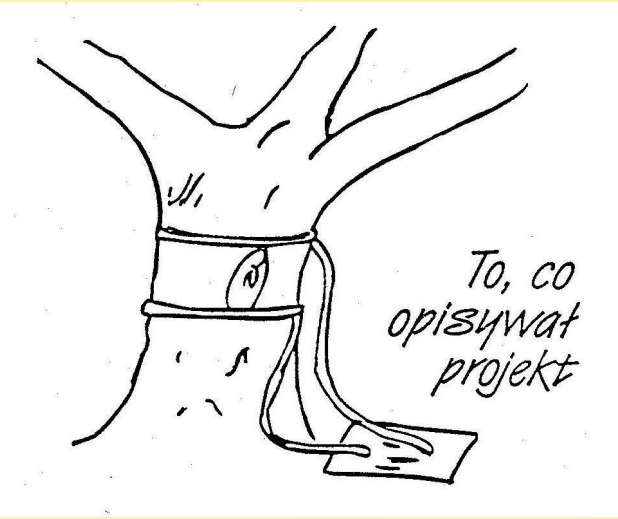
Czy potrafisz ..... ????

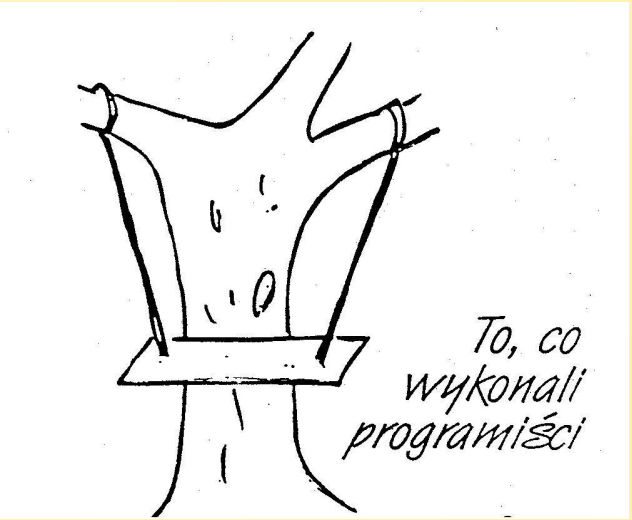


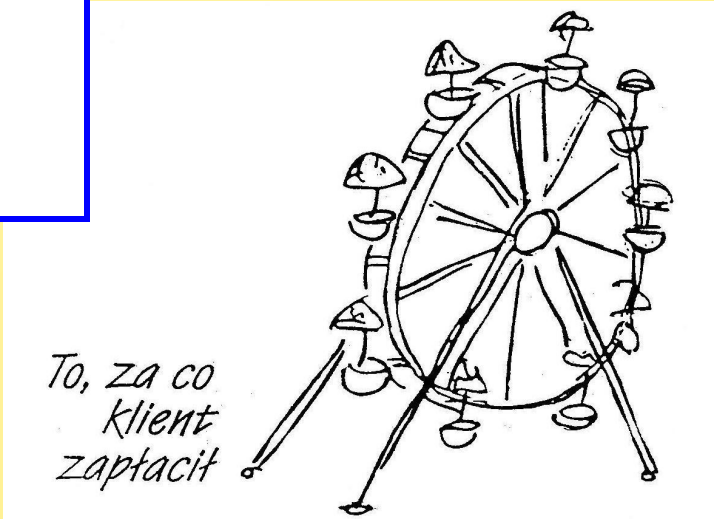
Etapy rozwoju systemu informatycznego









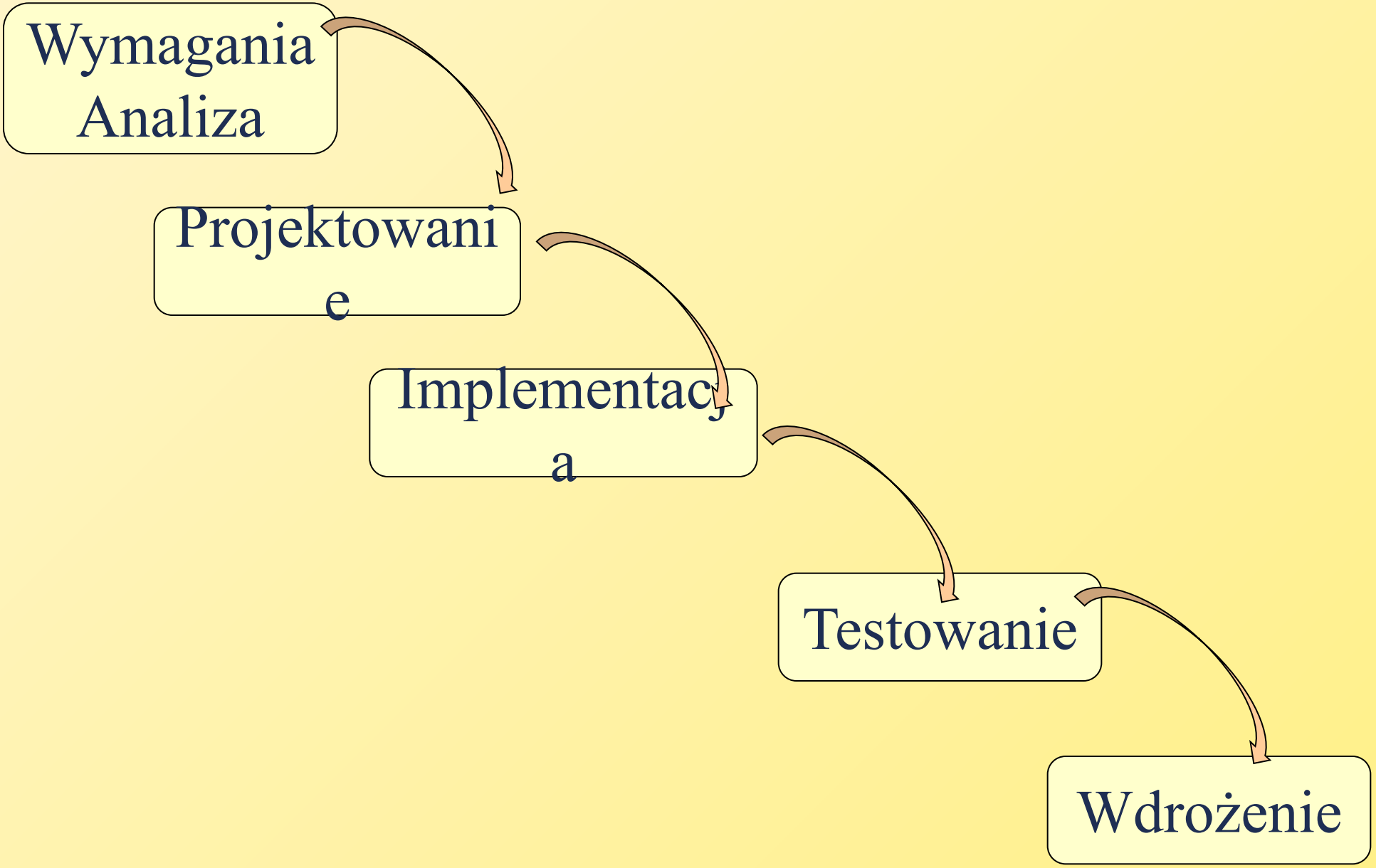


Czego oczekujemy??

Wymagania

Software

Plan ataku – teoria (w uproszczeniu)



A jak jest w rzeczywistości?

Wymagania
a
Analiza

Softwerek

Sukcesy projektów IT

Koszt: $10*10^{12}$ $

Czas: 3 lata opóźnienia

Jakość: pierwszy start Columbii odłożony z powodu problemów synchronizacyjnych z piątym komputerem pokładowym

- Źródłem błędów była zmiana wykonana 2 lata wcześniej przez programistę (współczynnik opóźnienia w procedurze zmieniony z 50 ms na 80 ms)
- Mimo tysięcy testów błąd ten nie został wykryty

Sukcesy projektów IT

- **POJAZD**
- **CEPiK**

Koszt: 200 000 0000 PLN

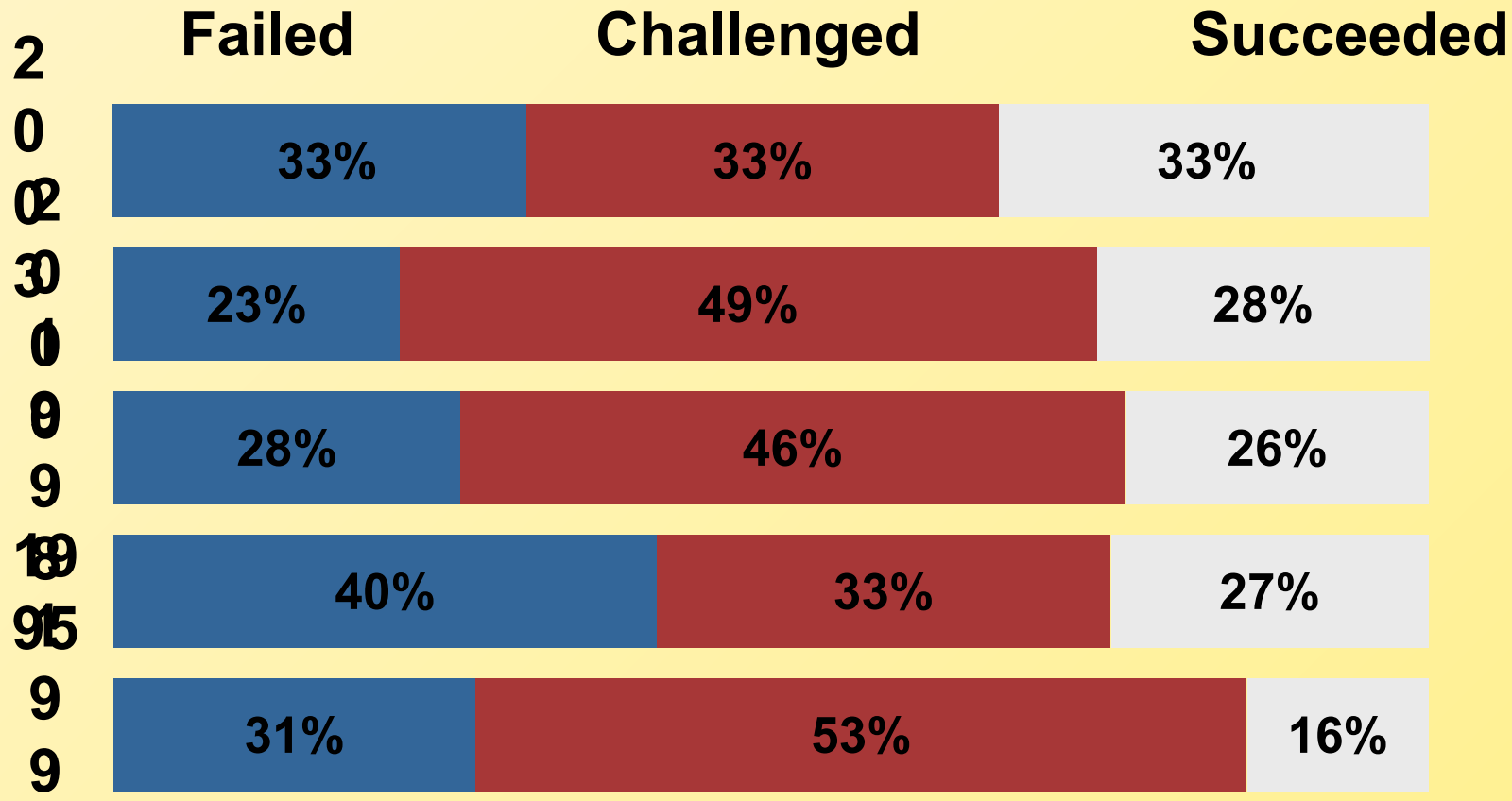
Czas: nieznany

Jakość: wydłużenie czasu rejestracji pojazdu z 15 do 45 minut

konieczność ręcznego przenoszenia danych

Wykonawca: Face Technologies - RPA

Sukcesy projektów informatycznych

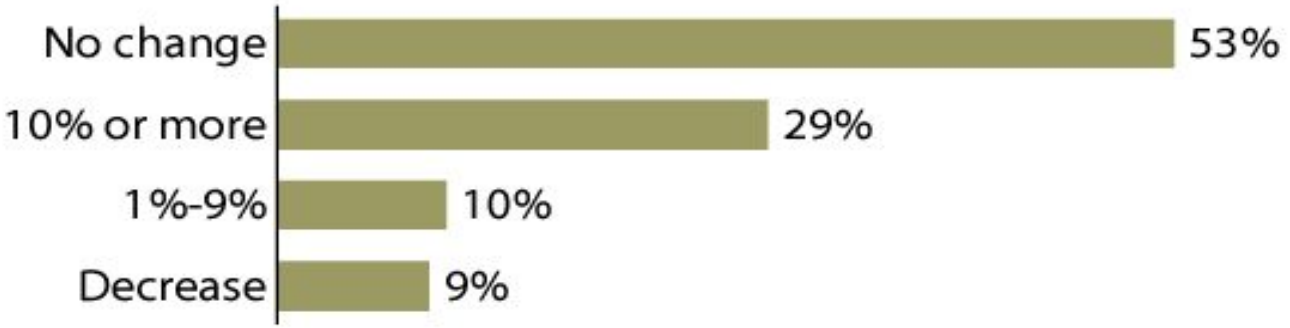


This chart depicts the outcome of the 30,000 application projects in large, medium, and small cross-industry U.S. companies tested by The Standish Group since 1994.

Source: The Standish Group International, *Extreme Chaos, The Standish Group International, Inc., 2004*

Budżet IT - 2004

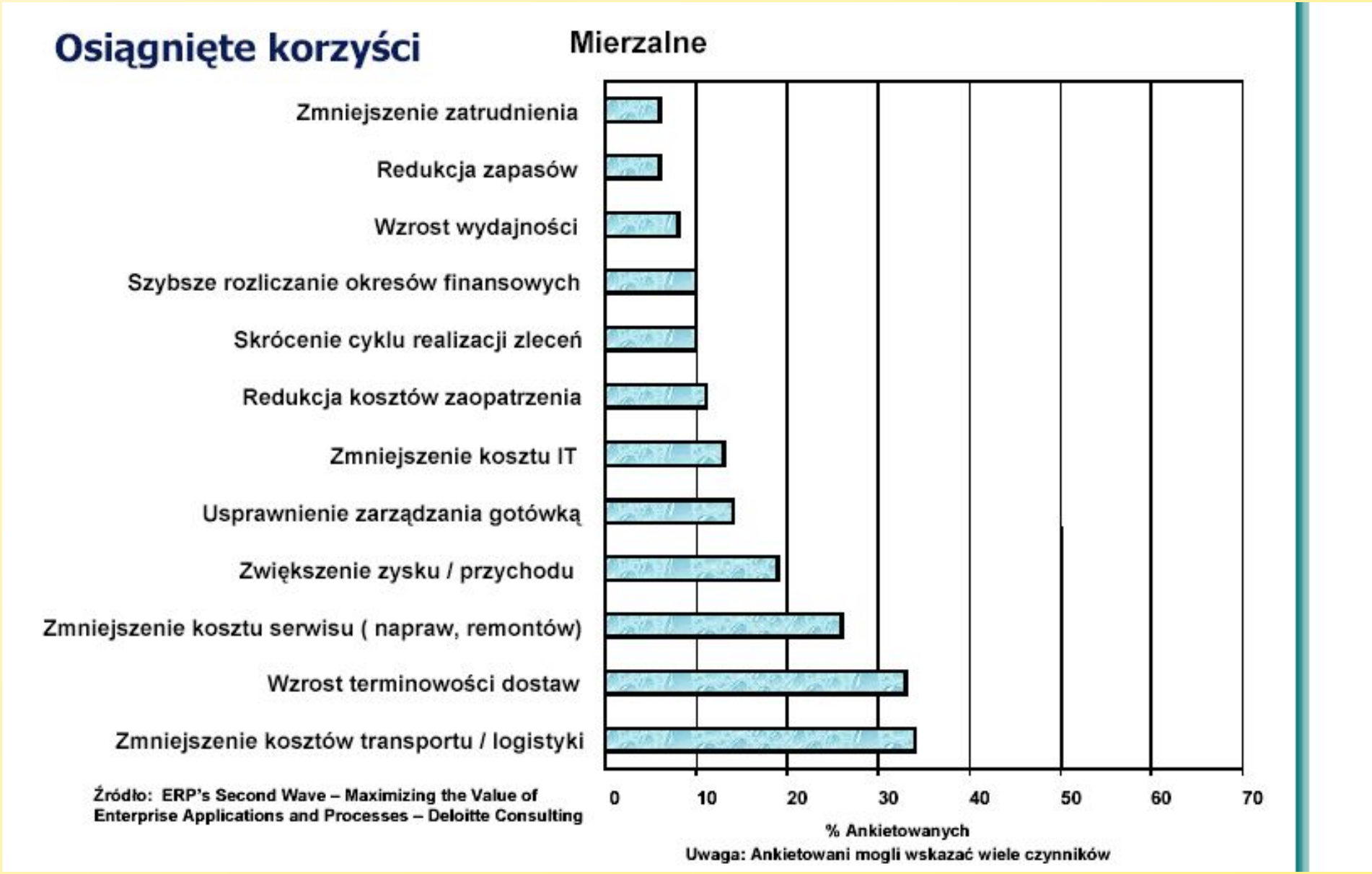
"By what percent do you estimate your IT budget will change in 2004?"*



*Although the survey base is 52 retail IT professionals, the base for this question is fewer than 40 respondents.

Source: Forrester's Business Technographics November 2003 North American Benchmark Study

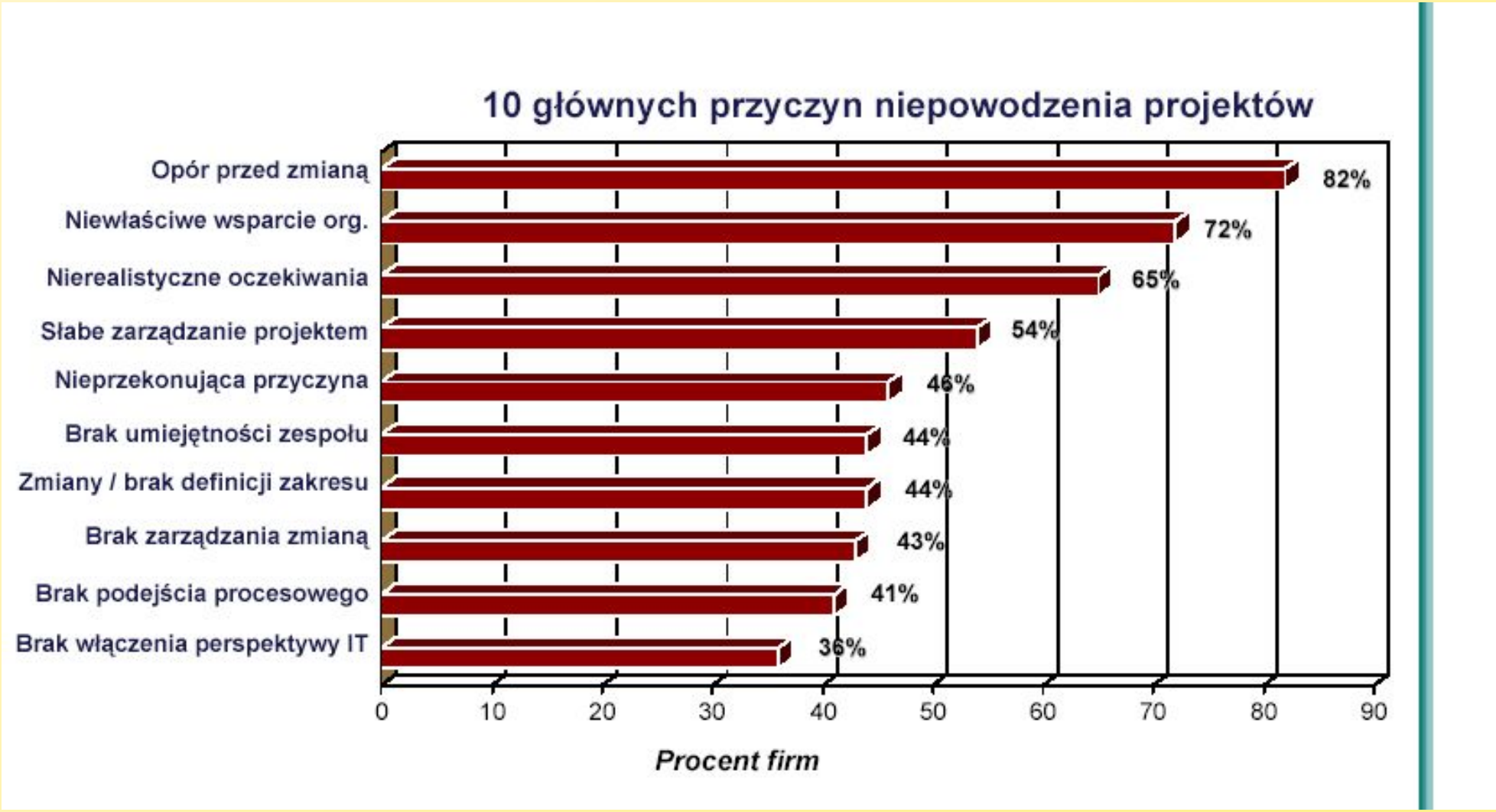
Czy warto? 1/2


Osiągnięte korzyści
Mierzalne
Zmniejszenie zatrudnienia
Redukcja zapasów
Wzrost wydajności
Szybsze rozliczanie okresów finansowych
Skrócenie cyklu realizacji zleceń
Redukcja kosztów zaopatrzenia
Zmniejszenie kosztu IT
Usprawnienie zarządzania gotówką
Zwiększenie zysku / przychodu
Zmniejszenie kosztu serwisu (napraw, remontów)
Wzrost terminowości dostaw
Zmniejszenie kosztów transportu / logistyki
0
10
20
30
40
50
60
70
% Ankietowanych
Źródło: ERP's Second Wave – Maximizing the Value of Enterprise Applications and Processes – Deloitte Consulting
Uwaga: Ankietowani mogli wskazać wiele czynników

Czy warto? 2/2


Osiągnięte korzyści
Niemierzalne
Lepszy dostęp do informacji
Nowe / usprawnione procesy
Poprawa komunikacji z klientem
Standaryzacja
Integracja
Poprawiona struktura kosztów
Większa elastyczność
Globalizacja
Poprawa wyników biznesowych
Zwiększenie wydajności łańcucha logistycznego
Nowy model biznesowy
0
10
20
30
40
50
60
70
% Ankietowanych
Źródło: ERP's Second Wave – Maximizing the Value of Enterprise Applications and Processes – Deloitte Consulting
Uwaga: Ankietowani mogli wskazać wiele czynników

Drobne trudności projektów



Zaliczenie

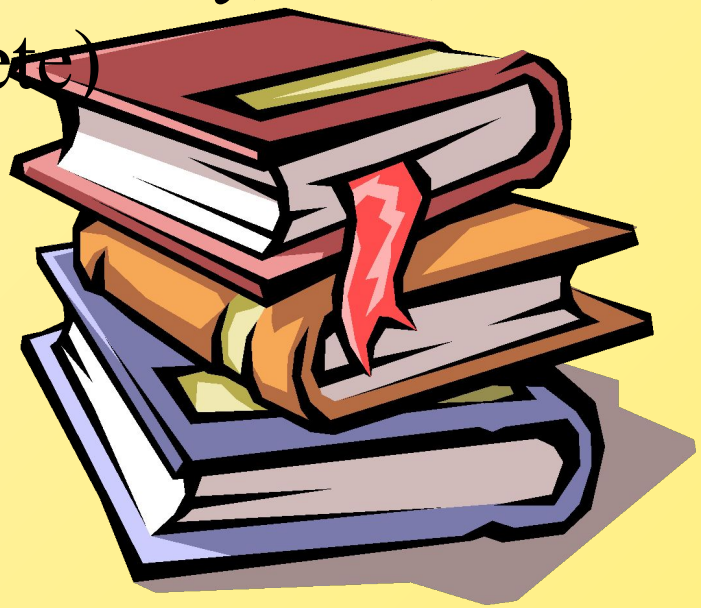
Literatura

[1] Kazimierz Subieta, Wprowadzenie do inżynierii oprogramowania, PJWSTK 2002

[2] Ian Sommerville, Inżynieria oprogramowania, WNT, Warszawa 2003

[3] Steve McConnell, Programista doskonały, LTP, Warszawa 2003 (ang. Code Complete)

[4] www.pjwstk.edu.pl/wlodek



Przedmiot inżynierii oprogramowania

Inżynieria oprogramowania jest wiedzą techniczną dotycząca wszystkich faz cyklu życia oprogramowania. Traktuje oprogramowanie jako produkt, który ma spełniać potrzeby techniczne, ekonomiczne lub społeczne.

Dobre oprogramowanie powinno być:

- zgodne z wymaganiami użytkownika,
- niezawodne,
- efektywne,
- łatwe w konserwacji,
- ergonomiczne.

Produkcja oprogramowania jest procesem składającym się z wielu faz.
Kodowanie (pisanie programów) **jest tylko jedną z nich**, niekoniecznie najważniejszą.

Inżynieria oprogramowania jest wiedzą empiryczną, syntezą doświadczenia tysięcy ośrodków zajmujących się budową oprogramowania.
Praktyka pokazała, że w inżynierii oprogramowania nie ma miejsca stereotyp „od teorii do praktyki”. Teorie, szczególnie zmatematyzowane teorie, okazały się dramatycznie nieskuteczne w praktyce.

Projekty

„Nie twierdzę, że kontrolowałem wydarzenia, wręcz przeciwnie – przyznaję otwarcie, że to one kontrolowały mnie.”

Abraham Lincoln

Zagadnienia inżynierii oprogramowania

- Sposoby prowadzenia przedsięwzięć informatycznych.
- Techniki planowania, szacowania kosztów, harmonogramowania i monitorowania przedsięwzięć informatycznych.
- Metody analizy i projektowania systemów.
- Techniki zwiększania niezawodności oprogramowania.
- Sposoby testowania systemów i szacowania niezawodności.
- Sposoby przygotowania dokumentacji technicznej i użytkowej.
- Procedury kontroli jakości.
- Metody redukcji kosztów konserwacji (usuwania błędów, modyfikacji i rozszerzeń)
- Techniki pracy zespołowej i czynniki psychologiczne wpływające na efektywność pracy.

Kryzys oprogramowania (1)

- Sprzeczność pomiędzy odpowiedzialnością, jaka spoczywa na współczesnych SI, a ich zawodnością wynikającą ze złożoności i ciągle niedojrzałych metod tworzenia i weryfikacji oprogramowania.
- Ogromne koszty utrzymania oprogramowania.
- Niska kultura ponownego użycia wytworzonych komponentów projektów i oprogramowania; niski stopień powtarzalności poszczególnych przedsięwzięć.
- Długi i kosztowny cykl tworzenia oprogramowania, wysokie prawdopodobieństwo niepowodzenia projektu programistycznego.
- Długi i kosztowny cykl życia SI, wymagający stałych (często globalnych) zmian.
- Eklektyczne, niesystematyczne narzędzia i języki programowania.

Kryzys oprogramowania (2)

- Frustracje projektantów oprogramowania i programistów wynikające ze zbyt szybkiego postępu w zakresie języków, narzędzi i metod oraz uciążliwości i długotrwałości procesów produkcji, utrzymania i pielęgnacji oprogramowania.
- Uzależnienie organizacji od systemów komputerowych i przyjętych technologii przetwarzania informacji, które nie są stabilne w długim horyzoncie czasowym.
- Problemy współdziałania niezależnie zbudowanego oprogramowania, szczególnie istotne przy dzisiejszych tendencjach integracyjnych.
- Problemy przystosowania istniejących i działających systemów do nowych wymagań, tendencji i platform sprzętowo-programowych.

Walka z kryzysem oprogramowania

- Stosowanie technik i narzędzi ułatwiających pracę nad złożonymi systemami;
- Korzystanie z metod wspomagających analizę nieznanych problemów oraz ułatwiających wykorzystanie wcześniejszych doświadczeń;
- Usystematyzowanie procesu wytwarzania oprogramowania, tak aby ułatwić jego planowanie i monitorowanie;
- Wytworzenie wśród producentów i nabywców przekonania, że budowa dużego systemu wysokiej jakości jest zadaniem wymagającym profesjonalnego podejścia.

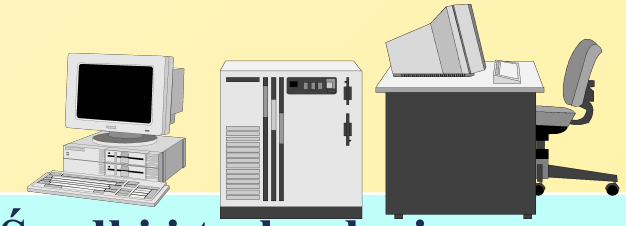
Podstawowym powodem kryzysu oprogramowania jest złożoność produktów informatyki i procesów ich wytwarzania.

Źródła złożoności projektu oprogramowania

Dziedzina problemowa, obejmująca ogromną liczbę wzajemnie uzależnionych aspektów i problemów.

Zespół projektantów podlegający ograniczeniom pamięci, percepcji, wyrażania informacji i komunikacji.

Oprogramowanie:
decyzje strategiczne,
analiza,
projektowanie,
konstrukcja,
dokumentacja,
wdrożenie,
szkolenie,
eksploatacja,
pielęgnacja,
modyfikacja.



Środki i technologie informatyczne:
sprzęt, oprogramowanie, sieć, języki, narzędzia, udogodnienia.

Potencjalni użytkownicy:
czynniki psychologiczne, ergonomia, ograniczenia pamięci i percepcji, skłonność do błędów i nadużyć, tajność, prywatność.

Jak walczyć ze złożonością ?

- ***Zasada dekompozycji*:**
  rozdzielenie złożonego problemu na podproblemy, które można rozpatrywać i rozwiązywać niezależnie od siebie i niezależnie od całości.

- ***Zasada abstrakcji*:**
  eliminacja, ukrycie lub pominięcie mniej istotnych szczegółów rozważanego przedmiotu lub mniej istotnej informacji; wyodrębnianie cech wspólnych i niezmiennych dla pewnego zbioru bytów i wprowadzaniu pojęć lub symboli oznaczających takie cechy.

- ***Zasada ponownego użycia*:**
  wykorzystanie wcześniej wytworzonych schematów, metod, wzorców, komponentów projektu, komponentów oprogramowania, itd.

- ***Zasada sprzyjania naturalnym ludzkim własnościom*:**
  dopasowanie modeli pojęciowych i modeli realizacyjnych systemów do wrodzonych ludzkich własności psychologicznych, instynktów oraz mentalnych mechanizmów percepcji i rozumienia świata.

Modelowanie pojęciowe

- Projektant i programista muszą dokładnie wyobrazić sobie problem oraz metodę jego rozwiązania. Zasadnicze procesy tworzenia oprogramowania zachodzą w ludzkim umyśle i nie są związane z jakimkolwiek językiem programowania.

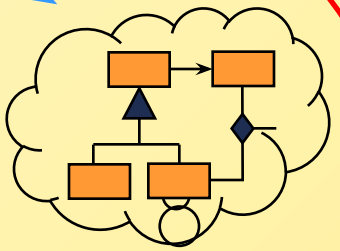
- Pojęcia ***modelowania pojęciowego*** (*conceptual modeling*) oraz ***modelu pojęciowego*** (*conceptual model*) odnoszą się procesów myślowych i wyobrażeń towarzyszących pracy nad oprogramowaniem.

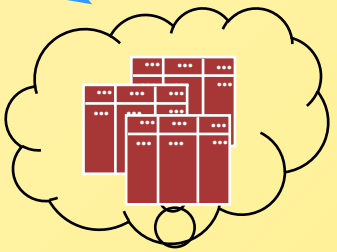
- Modelowanie pojęciowe jest wspomagane przez środki wzmacniające ludzką pamięć i wyobraźnię. Służą one do przedstawienia rzeczywistości opisywanej przez dane, procesów zachodzących w rzeczywistości, struktur danych oraz programów składających się na konstrukcję systemu.

Modelowanie systemów

odwzorowanie

odwzorowanie





Percepcja rzeczywistego świata

Analityczny model rzeczywistości

Model struktur danych i procesów SI

Trwałą tendencją w rozwoju metod i narzędzi projektowania oraz konstrukcji SI jest dążenie do minimalizacji luki pomiędzy myśleniem o rzeczywistym problemie a myśleniem o danych i procesach zachodzących na danych.

Co to jest metodyka (metodologia)?

Metodyka jest to zestaw pojęć, notacji, modeli, języków, technik i sposobów postępowania służący do analizy dziedziny stanowiącej przedmiot projektowanego systemu oraz do projektowania pojęciowego, logicznego i/lub fizycznego.

Metodyka jest powiązana z **notacją** służącą do dokumentowania wyników faz projektu (pośrednich, końcowych), jako środek wspomagający ludzką pamięć i wyobraźnię i jako środek komunikacji w zespołach oraz pomiędzy projektantami i klientem.

Metodyka ustala:

- fazy projektu, role uczestników projektu,
- modele tworzone w każdej z faz,
- scenariusze postępowania w każdej z faz,
- reguły przechodzenia od fazy do następnej fazy,
- notacje, których należy używać,
- dokumentację powstającą w każdej z faz.

Podsumowanie