SPECYFIKACJA WYMAGAŃ

Technologie Obiektowe

Cykl życia systemu informatycznego

Inżynieria oprogramowania traktuje oprogramowanie jako produkt. Cykl życia dowolnego produktu obejmuje min.

3 fazy:

1. Zebranie wymagań dotyczących systemu,
2. Opracowanie projektu systemu (np. język UML)
3. Wykonanie kodu w oparciu o projekt

\+ Sprawdzenie, czy system nie zawiera defektów i czy spełnia wymagania klienta (kontrola jakości)

Określenie wymagań – cel

- **Cel:** zebranie wymagań dotyczących systemu
- Opiera się na komunikacji pomiędzy klientem, a informatykami
- Stanowi główną część kontraktu między klientem, a dostawcą

Określenie wymagań -cel

Faza określania wymagań

Pytanie: *Co i przy jakich ograniczeniach system ma robić?*

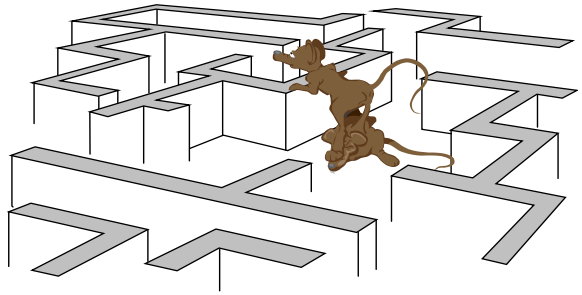
Wynik: *specyfikacja*, czyli zbiór wymagań funkcjonalnych i niefunkcjonalnych

Spisanie wymagań ⟺ szansa ⟺ klient dostanie dokładnie to czego potrzebuje

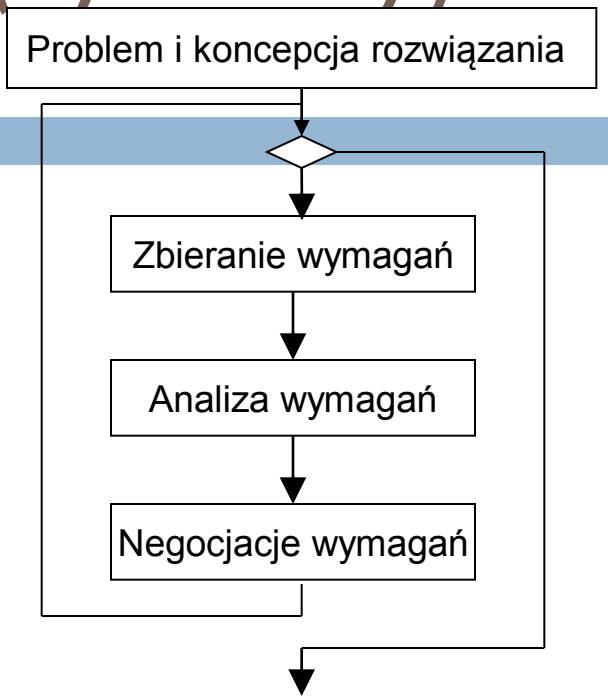
Proces zbierania i opracowania wymagań zazwyczaj jest cykliczny

Wymaga komunikacji z użytkowaniem!!!

Wykonanie specyfikacji wymagań przy użyciu narzędzi metod strukturalnych lub obiektowych



Proces zbierania wymagań (cykliczny)



I. Problem i koncepcja rozwiązania

sformułowanie problemu(ów), które system ma rozwiązać
+ ogólna wizja rozwiązania, etap poprzedzający

II. Proces zbierania wymagań

1. Zbieranie wymagań
   (często od różnych osób + przepisy prawa)
2. Analiza wymagań
   (w celu eliminacji sprzeczności, niejednoznaczności)
3. Negocjacje wymagań
   (w przypadku wykrycia defektów)

Uwagi:

- Klient może nie być w pełni świadomy każdego wymagania, stąd nie jest w stanie przekazać wszystkiego analitykowi.
- Analityk powinien rozumieć klienta (rozumieć terminy).
- Spisywanie wymagań jest „sztuką” , brak jest uniwersalnego sposobu, ale można wykorzystać metody, które się sprawdziły.

Problem

- **Wymagania** – opisy usług i ograniczeń stawianych systemowi
- **Dwa aspekty**
  - Formułowanie wymagań
  - Wynajdowanie, analizowanie, dokumentowanie, weryfikacja wymagań – inżynieria wymagań

Podstawowe zagadnienia

- Wymagania funkcjonalne i niefunkcjonalne
- Wymagania użytkownika
- Wymagania systemowe
- Dokumentacja wymagań stawianych oprogramowaniu

Co to jest wymóg?

- Pojęcie stosowane niekonsekwentnie.
- Zapisane na wysokim poziomie, abstrakcyjne określenie usług, które system powinien oferować, albo ograniczenie działania systemu.
- Niektórzy określają wymaganie jako szczegółową, matematycznie formalną definicję funkcji systemu.

Specyfikacja wymagań

- **Wymaganie** – opis tego co system ma robić
- **Specyfikacja wymagań** – opis, zbiór wymagań

- **Metody rozpoznania wymagań:**
  wywiady, przeglądy, analiza istniejącego oprogramowania, prototypowanie, itp.

Opis specyfikacji:

- język naturalny,
- tablice, formularze,
- diagramy blokowe (cykl przetwarzania),
- diagramy kontekstowe (system i jego powiązania z otoczeniem, DFD),
- diagramy przypadków użycia (funkcje użytkownika).

Pożądane cechy specyfikacji wymagań

- **Ścisłość**
  - Programista zinterpretuje tak, żeby uprościć sobie życie
- **Kompletność**
  - Wszystkie wymagania uwzględnione
- **Spójność**
  - Nie ma sprzeczności pomiędzy różnymi wymaganiami

Te cechy są niekiedy trudne do osiągnięcia, zwłaszcza kompletność i spójność.

Potrzeba różnych poziomów wymagań

- Firma chce podpisać kontrakt na budowę systemu
  - Musi określić swoje wymagania w odpowiednio abstrakcyjny sposób
  - Żeby nie narzucać z góry rozwiązania!
  - → wymagania użytkownika
- Kontrakt jest podpisany, wykonawca wybrany
  - Wykonawca musi sporządzić definicję systemu
  - Obie strony muszą zrozumieć i zgodzić się co ma być zrobione
  - → wymagania systemowe

Typy wymagań

- **Wymagania użytkownika**
  - To wyrażenia w języku naturalnym oraz diagramy o usługach oczekiwanych od systemu oraz o ograniczeniach, w których system ma działać.
- **Wymagania systemowe**
  - Szczegółowo ustalają usługi systemu i ograniczenia. Dokumentacja wymagań systemowych, zwana czasem specyfikacją funkcjonalną, powinna być precyzyjna.
- **Specyfikacja projektu oprogramowania**
  - Jest abstrakcyjnym opisem projektu oprogramowania, który jest podstawa bardziej szczegółowego projektu i implementacji.

Wymagania użytkownika systemu

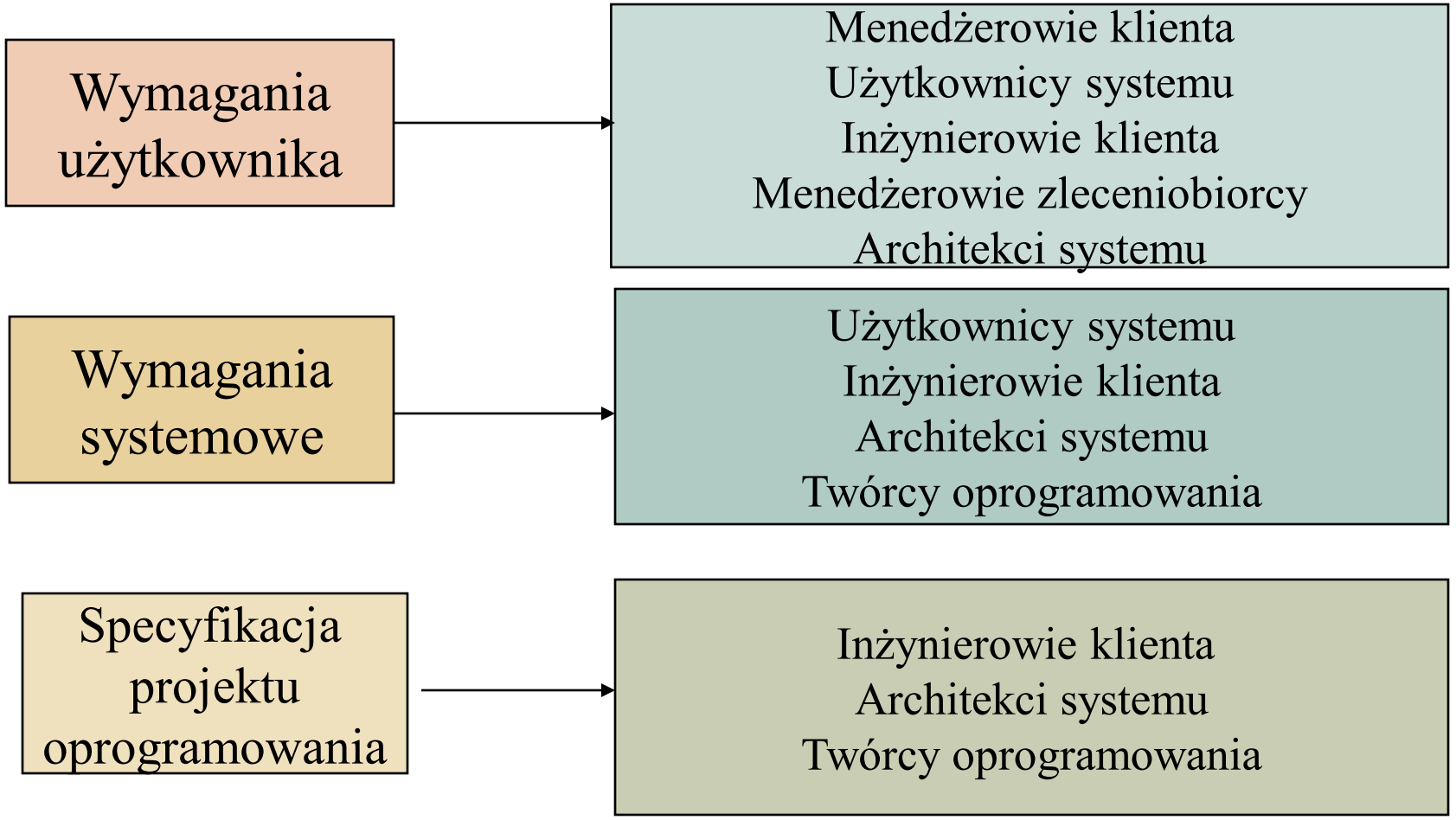
Definicja wymagań użytkownika

Oprogramowanie musi zapewniać mechanizmy reprezentowania i dostępu do plików zewnętrznych tworzonych przez inne narzędzia.

Specyfikacja wymagań systemowych

1.1 Użytkownik powinien mieć możliwość definiowania typów plików zewnętrznych.
1.2 Każdy typ pliku zewnętrznego może mieć przypisane narzędzie do obróbki takich plików.
1.3 Każdy typ pliku zewnętrznego może być przedstawiony w postaci charakterystycznej ikony na ekranie użytkownika.
1.4 Należy zapewnić udogodnienia do definiowania przez użytkownika ikon odpowiadających typom plików zewnętrznych.
1.5 Gdy użytkownik wybierze ikonę powiązaną z plikiem zewnętrznym, następuje zastosowanie do tego pliku narzędzia skojarzonego z typem tego pliku.

Czytelnicy różnych rodzajów specyfikacji



Wymagania funkcjonalne i niefunkcjonalne

- **Wymagania funkcjonalne**
  - Jakie usługi ma oferować system
  - Jak ma reagować na określone dane wejściowe
  - Jak ma się zachowywać w określonych sytuacjach
  - Czego nie powinien robić
- **Wymagania niefunkcjonalne**
  - Ograniczenia usług i funkcji systemu
  - Np. ograniczenia czasowe, ograniczenia dotyczące procesu tworzenia, standardy itd.
- **Wymagania dziedzinowe**
  - Pochodzą z dziedziny zastosowania
  - Mogą być funkcjonalne lub niefunkcjonalne.

Wymagania funkcjonalne

- **Jaką funkcjonalność / usługi system powinien oferować?**
- Zależą od rodzaju tworzonego oprogramowania, spodziewanych użytkowników oprogramowania i rodzaju wytwarzanego systemu.
- Gdy mają postać wymagań użytkownika, ich opis jest zwykle bardziej ogólny, natomiast wymagania funkcjonalne systemowe szczegółowo definiują funkcje systemu, jego wejścia, wyjścia, wyjątki itd.

Wymagania funkcjonalne – metody

- Wymagania funkcjonalne- to funkcje systemu widoczne od strony użytkownika

Metody opisu wymagań funkcjonalnych (podejścia)

1. „system powinien…"

Historyczne podejście, lata 80 (XX)

zaleta: łatwość spisywania,

wada: słaba czytelność, trudności w sprawdzeniu kompletności, spójności, duża ilość luźnych zdań, perspektywa systemu

przykład: System powinien generować zestawienie dzienne faktur sprzedaży
System powinien umożliwić wprowadzenie danych nabywcy

2. „Funkcje systemu"

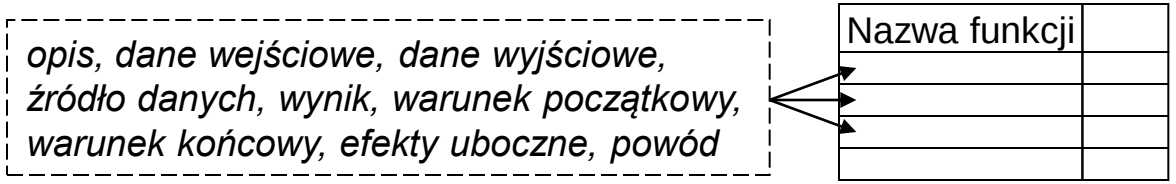
opis funkcji systemu (operacji) z określeniem wejść, wyjść, efektów ubocznych, hierarchia wymagań funkcjonalnych (HFD)

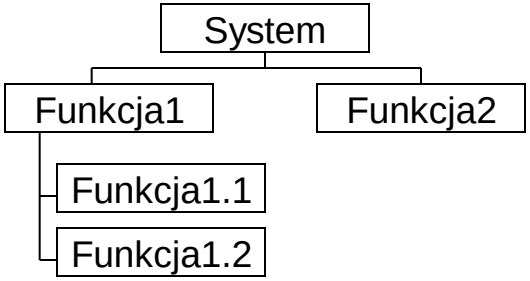
zaleta: łatwość spisywania, jednoznaczna interpretacja

podejście strukturalne

wada: perspektywa systemu, trudności w zrozumieniu

przykład: Wystawienie faktury (wejście: pozycje faktury; wyjście:wydruk; efekt: zapis w rejestrze)





Podejście obiektowe, współczesne

3. Przypadki użycia (ang. *use cases*)

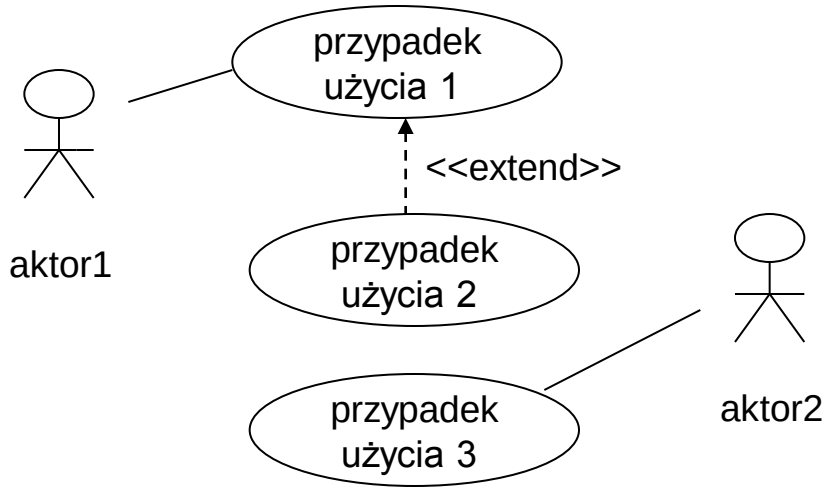
zaleta: łatwość spisywania, łatwość zrozumienia sposobu korzystania z systemu, czytelność, bez zbędnych szczegółów, ukazuje strukturę systemu w taki sposób w jaki widzą go użytkownicy,

uwagi: opisują interakcję między użytkownikiem a systemem (a nie jak system powinien się zachować),

elementy: scenariusz + diagram przypadków użycia (aktor, przypadek, związki)

UC1. Nazwa Przypadku
Główny scenariusz
Krok 1
...
Krok n
Rozszerzenie
Kroku k.1
...

Diagram przypadków użycia



Aktor –to byt, który inicjuje przypadek użycia (użytkownik, grupa, rola)

Przykłady wymagań funkcjonalnych

- Użytkownik będzie mógł przeszukać zbiór wszystkich baz danych lub wybrać tylko ich podzbiór.
- System udostępni odpowiednie narzędzia do oglądania, aby użytkownik mógł czytać dokumenty z magazynu.
- Każde zamówienie będzie oznaczone unikatowym identyfikatorem (ID), który będzie można skopiować do pamięci trwałej konta użytkownika.

Wymagania niefunkcjonalne

- Cechy i ograniczenia systemu nie związane z jego funkcjonalnością, np.
  - Wymagania pamięciowe (np. system nie może zajmować więcej niż 4Mb pamięci)
  - Efektywność
  - Skalowalność
  - Bezpieczeństwo
  - Niezawodność
  - ...

Wymagania niefunkcjonalne

opisują ograniczenia, przy których system ma realizować swoje funkcje

- wymagania dotyczące *produktu*,
  np. musi istnieć możliwość operowania z systemem wyłącznie za pomocą klawiatury,
- wymagania dotyczące *procesu*,
  np. proces realizacji harmonogramowania zleceń musi być zgodny ze standardem opisanym w dokumencie nr 20013/11/A,
- wymagania *zewnętrzne*,
  np. system harmonogramowania musi współpracować z bazą danych systemu komputerowego działu marketingu opisaną w dokumencie nr 2013/1. Niedopuszczalne są jakiejkolwiek zmiany w strukturze tej bazy.

Wymagania niefunkcjonalne powinny być weryfikowalne
(czyli powinna być możliwość sprawdzenia lub zmierzenia, czy system je faktycznie spełnia).

„*System ma być łatwy w obsłudze*” „... *niezawodny*” „ ... *szybki*”
- cechy są nieweryfikowalne, trzeba je ukonkretnić.

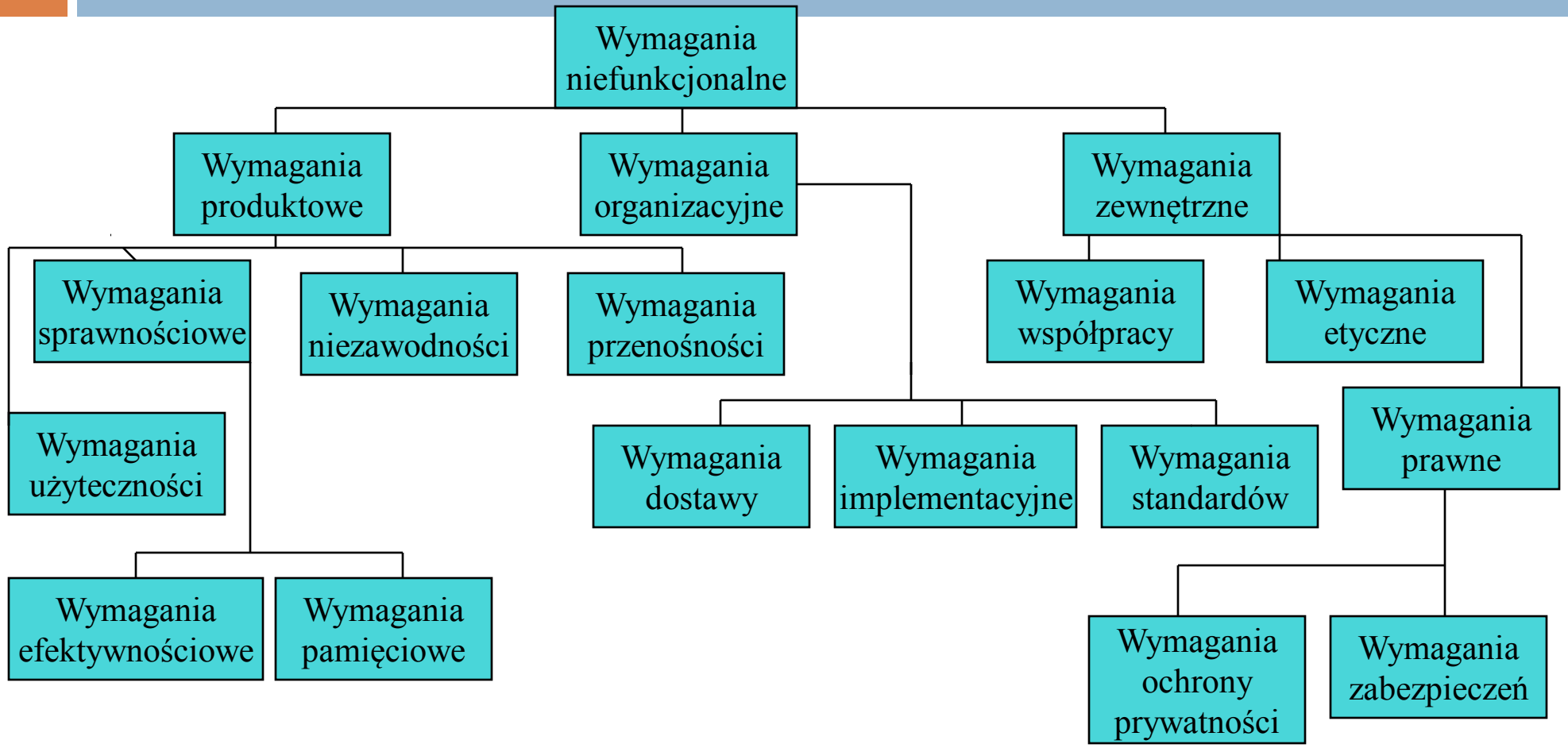
Wymagania niefunkcjonalne cd -czynniki

- Czynniki uwzględniane przy konstrukcji wymagań niefunkcjonalnych:
  - sprzęt
  - oprogramowanie
  - liczebność, objętość: ilość użytkowników pracujących jednocześnie, ilość danych przechowywanych, ilość stanowisk itp.
  - szybkość: liczba operacji na jednostkę czasu
  - zabezpieczenia
  - odporność na awarie
  - standardy: określenie dokumentów standaryzacyjnych
  - zasoby ludzkie, czasowe, finansowe

Klasyfikacja wymagań niefunkcjonalnych

- **Wymagania produktowe**
  - Określają zachowanie produktu. efektywnościowe dotyczące szybkości działania systemu i jego zapotrzebowania na pamięć, wymagania niezawodności.
- **Wymagania organizacyjne**
  - Wynikają ze strategii i procedur w firmie- kliencie i w firmie-wytwórcy.
- **Wymagania zewnętrzne**
  - Wynikające z czynników zewnętrznych dla systemu i procesu jego tworzenia. Np. wymagania współpracy z innymi systemami, wymagania prawne.

Typy wymagań niefunkcjonalnych



Trudności weryfikacji wymagań niefunkcjonalnych

- **Cel systemu**
  - „System powinien być łatwy w użyciu dla doświadczonych kontrolerów, a sposób jego organizacji powinien zmniejszać liczbę błędów użytkownika.”
- **Jak weryfikować spełnienie tego wymagania?**
- **Inne sformułowanie**
  - „Doświadczeni kontrolerzy powinni móc używać wszystkich funkcji systemu po szkoleniu trwającym dwie godziny. Po tym szkoleniu średnia liczba błędów robionych przez doświadczonych użytkowników nie powinna przekroczyć dwóch dziennie.”

Cechy systemu i ich weryfikowalne miary

- **wydajność:** liczba transakcji, czas odpowiedzi, ...
- **rozmiar:** liczba rekordów, wymagana pamięć dyskowa,
- **łatwość użytkowania:** czas niezbędny dla przeszkolenia użytkowników, dokumentacje, ...
- **Niezawodność:** prawdopodobieństwo zniszczenia danych w czasie awarii, dostępność systemu (czas, procent czasu itp.) w czasie awarii, itp.
- **Przenaszalność:** ilość platform docelowych, koszt przeniesienia na inną platformę

Miary do specyfikowania wymagań niefunkcjonalnych

Właściwość	Miara
Szybkość	Liczba przetworzonych transakcji na sekundę Czas reakcji na zdarzenie wywołane przez użytkownika Czas odświeżenia ekranu
Rozmiar	Kilobajty Liczba układów pamięci
Łatwość użycia	Czas szkolenia Liczba okien pomocy
Niezawodność	Średni czas bez awarii Prawdopodobieństwo niedostępności Częstość błędów Dostępność
Solidność	Czas uruchamiania po awarii Ułamek zdarzeń powodujących awarie Prawdopodobieństwo zniszczenia danych po awarii
Przenaszalność	Procent poleceń zależnych od platformy Liczba docelowych systemów

Wymagania dziedzinowe

- Wynikają z dziedziny zastosowanie systemu
- Mogą być funkcjonalne lub niefunkcjonalne
- **Przykład:**
  - Tempo zmniejszania prędkości pociągu jest wyznaczane następująco:

    Dpociągu = Dsterowania + Dnachylenia

    przy czym Dnachylenia wynosi 9,81m/s2 * wyrównane nachylenie/alfa, a 9,81m/s2 /alfa są znane dla różnych typów pociągów .
- **Problem:** formułowane są w języku ekspertów, który może być niezrozumiały dla twórców systemu

Przykład: wymaganie użytkownika stawiane siatce edytora

Udogodnienia siatki w edytorze tekstu.
Przez opcje panelu sterowania użytkownik może uaktywnić siatkę w centymetrach lub w calach, która będzie pomagała w umieszczaniu bytów na diagramie. Siatka może być włączona i wyłączona w dowolnej chwili sesji edycji; to samo dotyczy przełączania między calami i centymetrami. Opcja siatki będzie dostępna w widoku „zmniejsz, aby dopasować", ale liczba linii siatki będzie wówczas zmniejszona, aby uniknąć zapełnienia małego diagramu liniami siatki.

Co jest złego w tym wymaganiu?

Co jest złego w tym wymaganiu?

- Trzy rodzaje wymagań w jednym
  - System powinien udostępniać siatkę – wymaganie funkcjonalne
  - Wymóg co do jednostek siatki – wymaganie niefunkcjonalne
  - Określenie jak użytkownik włącza i wyłącza siatkę – wymaganie niefunkcjonalne
- Wymaganie to niejasno sugeruje, że na początku siatka jest wyłączona
  - Ale nie mówi już, jaka jednostka powinna być uaktywniona na początku
- Zbyt wiele informacji! Ogranicza innowacyjność twórców

Wymaganie dobrze sformułowane – tylko główne cechy

2.6 Siatka

2.6.1 Edytor będzie udostępniał siatkę, tzn. matrycę linii pionowych jako tło okna edytora. Ta sama siatka powinna być pasywna, tzn. za układanie bytów odpowiada użytkownik.

Uzasadnienie: Siatka pomaga użytkownikowi w tworzeniu schludnego diagramu ze starannie poukładanymi bytami. Chociaż siatka aktywna, przy której byty przeskakują do linii siatki, może być użyteczna, jednak wówczas układ diagramu jest nieprecyzyjny. Użytkownik jest najwłaściwszą osobą do decydowania o położeniu bytów.

Specyfikacja: ECLIPSE/WS/Tools/DE/FS Punkt 5.6

Bardziej szczegółowe wymaganie użytkownika – dodawania węzłów

3.5.1 Dodawanie węzłów do projektu

3.5.1.1 **Edytor będzie udostępniał użytkownikom udogodnienia do dodawania do swoich projektów węzłów określonego typu**

3.5.1.2 Sekwencja czynności, które prowadzą do dodania węzła, powinna być następująca:

1. Użytkownik powinien wybrać typ węzła, jaki należy dodać
2. Użytkownik powinien przesunąć wskaźnik do przybliżonego miejsca nowego węzła na diagramie i zalecić dodanie symbolu węzła w tym punkcie
3. Użytkownik powinien następnie przeciągnąć węzeł do jego ostatecznego położenia

Uzasadnienie: Użytkownik jest najwłaściwszą osobą do decydowania o położeniu węzłów na diagramie. Takie podejście daje użytkownikowi całkowite panowanie nad wyborem typu węzła i jego umiejscowieniem

Specyfikacja: ECLIPSE/WS/Tools/DE/FS Punkt 3.5.1

Wymagania systemowe

- Bardziej szczegółowe opisy wymagań użytkownika
- Mogą być podstawą kontraktu na implementację systemu
  - Powinny być pełną i niesprzeczną specyfikacją całego systemu
- Punkt wyjścia do projektowania systemu
- Specyfikacja wymagań systemowych może zawierać różne modele systemu

Wymagania a projekt

- W dokumentacji wymagań można zdefiniować wstępną architekturę systemu
  - Nadaje specyfikacji odpowiednią strukturę
  - Wymagania systemowe są zorganizowane zgodnie z podziałem na podsystemy wchodzące w skład systemu
- W większości wypadków systemy muszą współpracować z innymi istniejącymi systemami
  - Ograniczają one projekt, co implikuje dodatkowe wymagania stawiane nowemu systemowi
- Użycie specyficznego projektu może być zewnętrznym wymaganiem systemowym.

Formularz do definiowania wymagań

- Opis specyfikowanej funkcji lub bytu
- Opis jej danych wejściowych i źródło ich pochodzenia
- Opis jej danych wyjściowych i miejsce ich przeznaczenia
- Określenie, z których innych bytów się korzysta (część wymaga)
- Jeśli użyto podejścia funkcjonalnego, to jest wywołany warunek początkowy, który musi być prawdziwy przed wywołaniem tej funkcji, oraz warunek końcowy, który musi być prawdziwy po wywołaniu funkcji.
- Opis efektów ubocznych operacji (jeśli występują)

Dokument wymagań

- Wymagania powinny być udokumentowane – tzw. opis wymagań.
- Dokument powinien być podstawą szczegółowego kontraktu między klientem a producentem oprogramowania.
- Dokument powinien pozwalać na weryfikację stwierdzającą, czy wykonany system rzeczywiście spełnia postawione wymagania.
- Dokument powinien być zrozumiały dla obu stron.

Specyfikacja wymagań systemu z użyciem standardowego formularza

ECLIPSE/Workstation/Tools/DE/FS/3.5.1
Funkcja. Dodaj węzeł **Opis.** Dodaje węzeł do istniejącego projektu. Użytkownik wybiera typ i położenie węzła po dodaniu do projektu węzeł jest zaznaczony. Użytkownik wybiera miejsce węzła przesuwając wskaźnik na obszar, w którym dodano węzeł **Dane wejściowe.** Typ węzła, Położenie węzła, Identyfikator projektu **Źródło.** Typ węzła i Położenie węzła pochodzą od użytkownika, a Identyfikator projektu z bazy danych **Dane wyjściowe.** Identyfikator projektu **Przeznaczenie.** Baza danych projektów. Projekt jest utrwalany w bazie danych po zakończeniu operacji **Wymagania.** Korzeniem grafu projektu musi być dany identyfikator projektu **Warunek początkowy.** Projekt jest otwarty i wyświetlony na ekranie użytkownika **Warunek końcowy.** Projekt nie uległ zmianie z wyjątkiem dodania węzła zadanego typu o zadanym położeniu **Efekty uboczne.** Nie ma

Specyfikacja interfejsów

- Większość systemów musi współdziałać z innymi systemami, które już znajdują się w środowisku
- Interfejsy istniejących systemów muszą być wyspecyfikowane
- Trzy typy informacji
  - Interfejsy proceduralne
  - Struktury danych, które są przekazywane między podsystemami
  - Reprezentacje danych (np. porządek bitów)

Każdy interfejs użytkownika powinien być:

- **użyteczny** (spełniać to do czego został stworzony),
- **intuicyjny** (jak najbardziej przyjazny użytkownikowi),
- **spójny** (z innymi dialogami aplikacji),
- **prosty** (pozbawiony zbędnych opcji i przycisków),
- **bezpieczny** (chroniący dane przed błędami lub uszkodzeniem),
- **wyrozumiały** umożliwiać wycofanie się z operacji
- **estetyczny**

Opis interfejsu serwera drukowania za pomocą PDL opartego na Javie

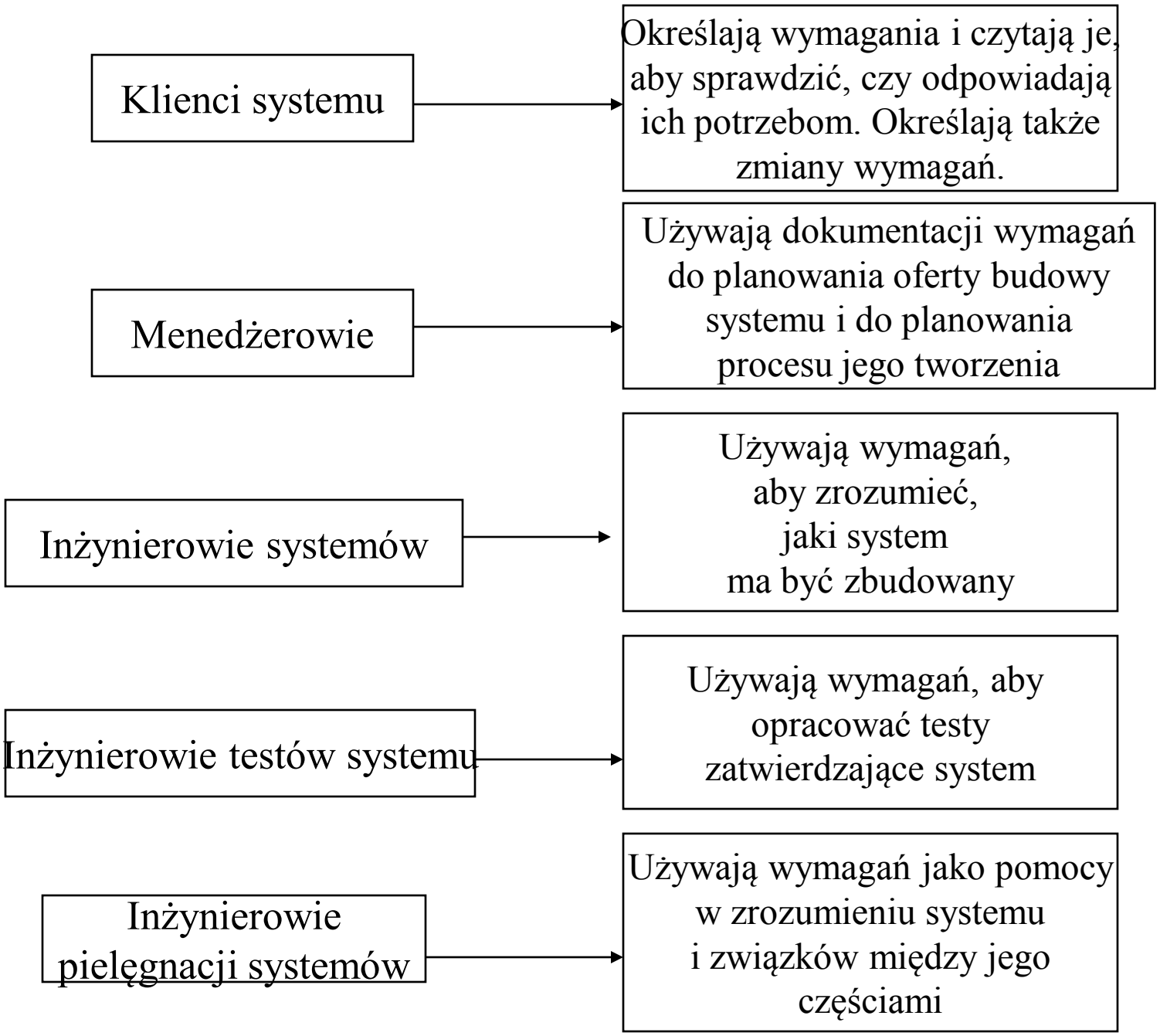
```
     interface SerwerDrukowania {

     // definiuje abstrakcyjny serwer drukowania
     // wymaga: interface Drukarka, interface DokumentDoWydruku
     //      udostepnia:      inicjuj,      drukuj,      wyświetlKolejkeZadań,
anulujZadanieDrukowania,
     // zmieńDrukarkę

       void inicjuj ( Drukarka d ) ;
       void drukuj ( Drukarka d, DokumentDoWydruku w) ;
       void wyświetlKolejkęZadań ( Drukarka d ) ;
       void anulujZadanieDrukowania (Drukarka d, DokumentDoWydruku w) ;
       void zmieńDrukarkę(Drukarka d1, Drukarka d2, DokumentDoWydruku w) ;
     } //SerwerDrukowania
```

Dokumentacja wymagań stawianych oprogramowaniu

- Software Requirements Specification (SRS)
  - oficjalna deklaracja tego, co jest wymagane od wytwórców systemu
- Powinna zawierać zarówno wymagania użytkownika stawiane systemowi, jak i szczegółowe specyfikacje wymagań systemowych
- Nie jest projektem
  - Opisuje co system ma robić, a nie jak to robić



Użytkownicy dokumentacji wymagań

Standard wymogów IEEE

1. Wstęp
    - 1.1. Przeznaczenie tej dokumentacji wymagań
    - 1.2. Zakres produktu
    - 1.3 Definicje, akronimy i skróty
    - 1.4. Odnośniki
    - 1.5. Przegląd pozostałej części dokumentu
2. Ogólny opis
    - 2.1 Wizja produktu
    - 2.2 Funkcje produktu
    - 2.3 Charakterystyka użytkowników
    - 2.4 Ogólne ograniczenia
    - 2.5 Założenia i zależności
3. Szczegółowe wymagania
4. Dodatki
5. Skorowidz

Struktura dokumentacji wymagań

- Przedmowa
- Wstęp
- Słownik
- Definicja wymagań użytkownika
- Architektura systemu
- Specyfikacja wymagań systemowych
- Modele systemu
- Ewolucja systemu
- Dodatki
- Skorowidz

Literatura

- Jaszkiewicz A., Inżynieria oprogramowania, Helion, Gliwice 1997 (lub nowsze)
- Subieta K., Wprowadzenie do inżynierii oprogramowania, PJWSTK, Warszawa 2002
- Wrycza S., Język UML 2.0 w modelowaniu obiektowym, Helion 2006
- Schmuller J., UML dla każdego, Helion, Gliwice 2003
- http://wazniak.mimuw.edu.pl
- http://www.ipipan.waw.pl/~subieta/wyklady/wyklady.htm
- Ian Sommerville, Inżynieria Oprogramowania, r. 5

Przykład

DZIĘKUJĘ ZA UWAGĘ