Wykład :

Inżynieria Wymagań

-

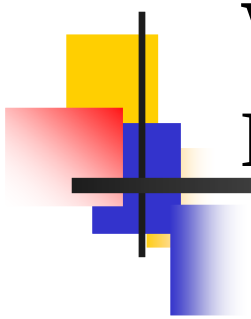
techniki specyfikacji wymagań

mgr inż. Jacek Kołodziej, mgr inż. Grzegorz Młynarczyk

Opracowano na podstawie:

Inżynieria oprogramowania – wykład : mgr inż. Grzegorz Młynarczyk, WSZIB, Kraków

Software Engineering, 7th edition, Ian Sommerville 2004



W poprzednim odcinku:
IW (RE) (1/4)

- Pozyskiwanie i analiza wymagań
- Reprezentacje wymagań na poszczególnych etapach projektu
- Najczęściej pojawiające się problemy podczas pozyskiwania wymagań oraz metody ich rozwiązywania
- Reprezentacja wymagań z punktu widzenia różnych grup użytkowników systemów

W poprzednim odcinku:
IW (2/4)

- Walidacja i weryfikacja
- Mierzalność wymagań i kryteria akceptacji
- Rodzaje dokumentów powstających w procesie analizy wymagań
- Struktura dokumentu specyfikacji wymagań
- Przykłady dokumentów specyfikacji wymagań

W poprzednim odcinku: IW (3/4)

- **Inżynieria wymagań** - proces **identyfikacji wymagań** jakich spełnienia oczekują **użytkownicy** systemu oraz **ograniczeń** nakładanych na jego realizację i użytkowanie
- Co to jest wymaganie?

W poprzednim odcinku:
Wymaganie (4/4)

- **Wymagnie (ang. requirement)** - pojęcie wymagania jest różnie interpretowane przez różne organizacje zajmujące się produkcją oprogramowania. Może oznaczać zarówno bardzo „zgrubny”, abstrakcyjny opis **funkcji** jakie powinien realizować system informatyczny lub **ograniczeń** jakim może podlegać, jak również może być matematyczną formułą, bardzo precyzyjnie określającą jego funkcjonalność

- Wymagania w wielu przypadkach mogą spełniać różne, wydawałoby się przeciwstawne, funkcje:
  - mogą stanowić podstawę dla ofert na realizację systemu - powinny być otwarte, tak aby nie narzucać formy realizacji systemu
  - mogą (a właściwie **powinny**) stanowić podstawę kontraktu na realizację systemu - powinny zatem być **dokładne**, **kompletne**, **spójne**, **realizowalne** i **weryfikowalne**.

Techniki specyfikacji oraz definiowania wymagań systemów informatycznych

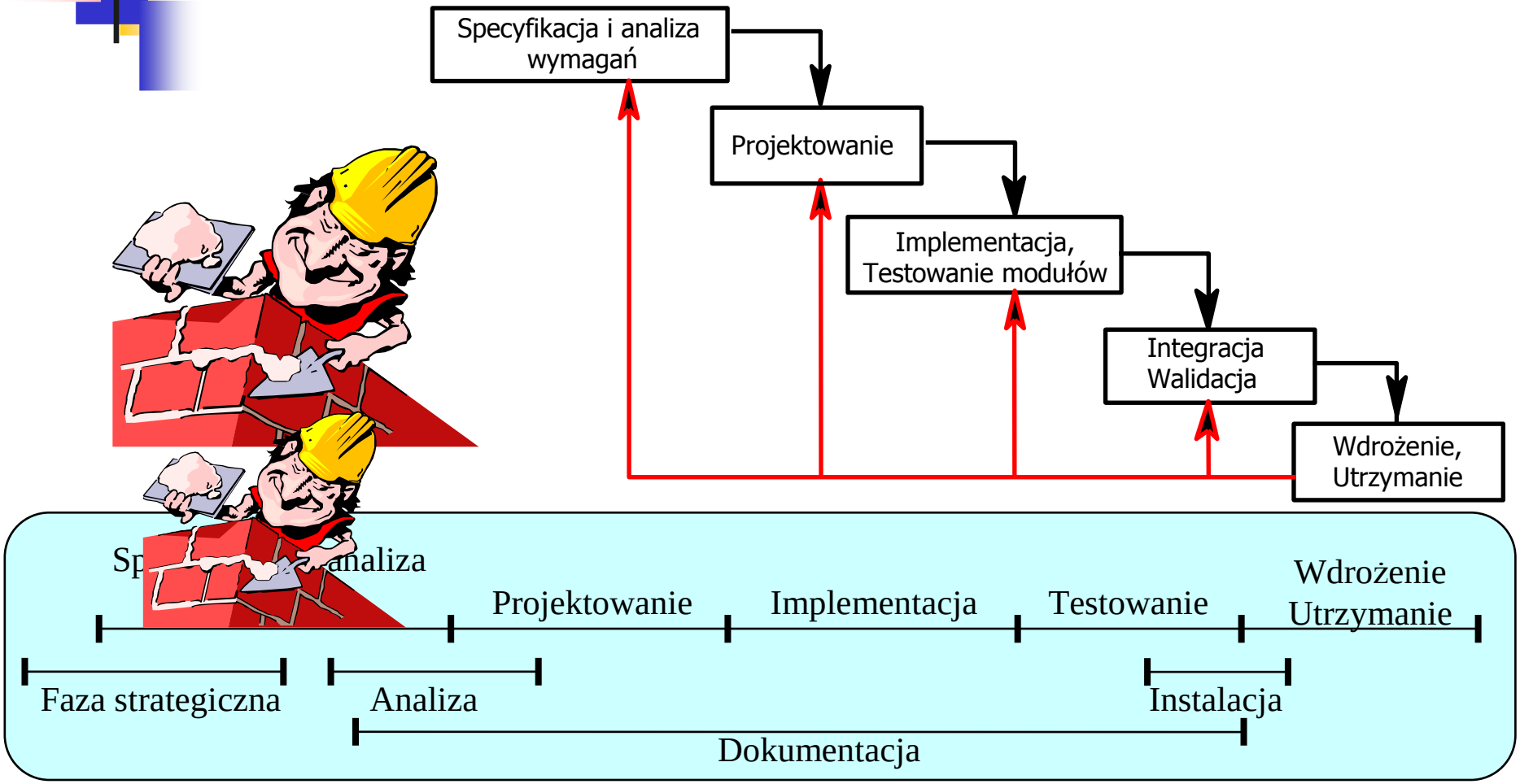
Zagadnienia (1/2)

- Opis sposobów tworzenia precyzyjnego opisu wymagań
- Omówienie istotnej roli wymagań niefunkcjonalnych
- Różne typy wymagań niefunkcjonalnych oraz opis sposobów ich specyfikacji

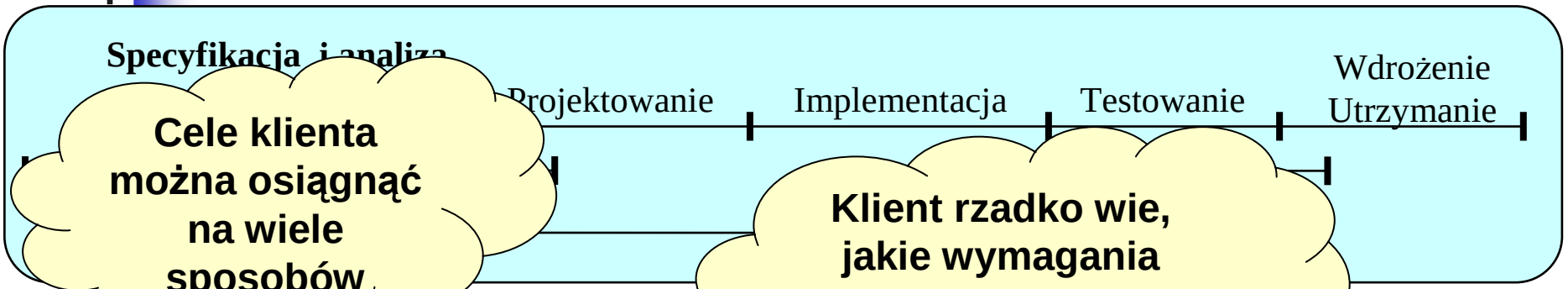
Zagadnienia (2/2)

- Definicja wymagań
- Specyfikacja wymagań
- Wymagania niefunkcjonalne
- Wymaganie systemowe
- Prezentacja standardu IEEE dokumentu specyfikacji wymagań

Specyfikacja wymagań

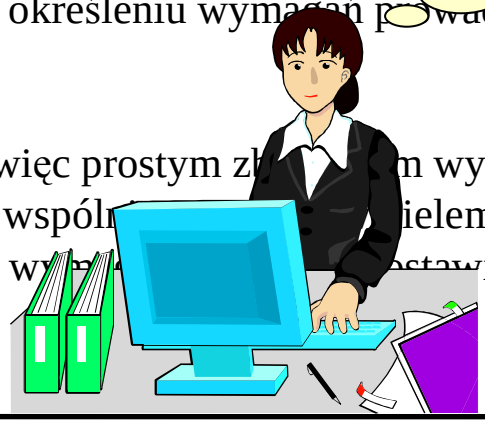


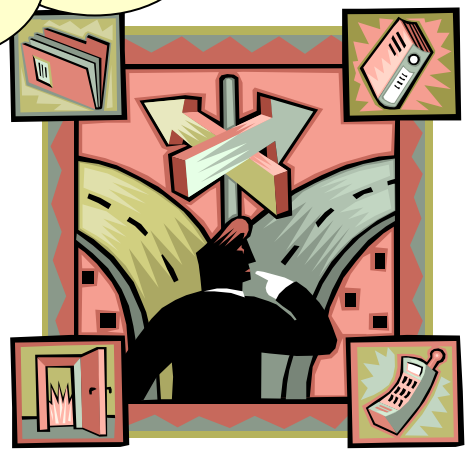
Faza analizy wymagań

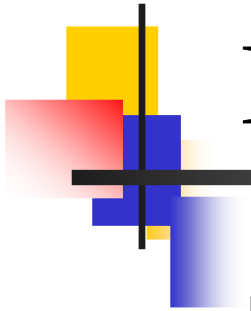


Specyfikacja i analiza wymagań ***(inżynieria w***
projektowa której celem jest precyzyjne określen
klienta wobec projektowanego systemu. Pole[illegible]nterpretacji
intencji klienta i określeniu wymagań prowadzących do ich
osiągnięcia.

Ta faza nie jest więc prostym zb[illegible]m wymagań, lecz procesem,
w którym klient wspól[illegible]ielem producenta
konstruuje zbiór w[illegible]stawionymi celami.







Definicja a specyfikacja

- Definicja wymagań
  - Widziany z perspektywy klienta opis funkcji systemu oraz ograniczeń jego funkcjonowania
- Specyfikacja wymagań
  - Precyzyjny, detaliczny opis funkcjonalności systemu oraz jego ograniczeń. W swoim zamierzeniu ma on służyć twórcom systemu – jaka funkcjonalność ma być stworzona oraz jako podstawa kontraktu dotyczącego stworzenia systemu

Definicja wymagań

- Powinna opisywać zachowanie systemu – tak więc nie powinna być tworzona w oparciu o model implementacji
  - Powinna być zrozumiała dla klienta – język naturalny, proste i intuicyjne diagramy
- Powinna zawierać zarówno wymagania funkcjonalne jak i niefunkcjonalne
  - Wymagania funkcjonalne – opis funkcjonalności zapewnianej przez system
  - Wymagania niefunkcjonalne – ograniczenia nakładane na funkcjonalność systemu

Tworzenie definicji wymagań

- Standardowy sposób postępowania – język naturalny, diagramy oraz tabele
- Daje to gwarancję zrozumiałości ale generuje trzy rodzaje problemów
  - Brak przejrzystości. Uzyskanie precyzji opisu jest trudne bez jego komplikacji
  - Brak podziału wymagań. Wymagania funkcjonalne i niefunkcjonalne mają tendencję do wzajemnego przeplatania się
  - Łączenie wymagań. Częsta sytuacja – kilka różnych wymagań jest przedstawianych jako jedno

Metody rozpoznawania wymagań

- Wywiady i przeglądy
  - przygotowane w postaci listy pytań
  - podzielone na odrębne zagadnienia, pokrywające całość zagadnień dotyczących projektu
  - przeprowadzone na reprezentatywnej grupie użytkowników
  - Pozytywny Wywiady = Kompromis i Akceptacja projektu.

Metody rozpoznawania wymagań

- Studia wymagań systemowych.
  - Analiza integracji projektowanego systemu z nadrzędnym, którego jest elementem.
- Studia osiągalności.
  - Wyznaczenie realistycznych wymagań systemu oraz wskazanie metod ich osiągnięcia.
- Studia na istniejącym oprogramowaniem.
  - nowe oprogramowanie zastępuje stare
  - Ustalenie wad i zalet starego oprogramowania.
- Prototypowanie.
  - budowana prototypu systemu działającego w zmniejszonej skali, z uproszczonymi interfejsami
  - oszacowanie przydatności modelu, sprecyzowanie wymagań.

Wymagania funkcjonalne

Specyfikują funkcje (czynności, operacje) realizowane przez system (lub przy użyciu systemów zewnętrznych).

- Specyfikacja wymagań :
  - Jednoznaczne wskazanie wszystkich użytkowników korzystających z systemu,
  - Jednoznaczne wskazanie wszystkich użytkowników, niezbędnych do działania systemu (obsługa, wprowadzanie danych, administracja).
  - Określenie funkcji systemu udostępnianych użytkownikowi oraz sposobów korzystania z planowanego systemu.
  - Określenie systemów zewnętrznych (obcych baz danych, sieci, Internetu), wykorzystywanych w czasie pracy systemu.
  - Ustalenie struktur organizacyjnych, przepisów prawnych, statutów, zarządzeń, instrukcji, itd., które pośrednio lub bezpośrednio określają funkcje wykonywane przez planowany system.

Opis wymagania - przykład

- Wymaganie bazy danych systemu APSE*

4.A.5 The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name.

** Ada Programming Support Environment*

Wymaganie - przykład

- Wymaganie dotyczące funkcjonalności siatki edytora

> **2.6 Grid facilities** To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimetres at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.

Definicja wymagania

- Wymaganie dotyczące edytora łączy w sobie wymagania funkcjonalne i niefunkcjonalne oraz jest niepełne
- Wymaganie to miesza ze sobą koncepcje oraz detale dotyczące funkcjonowania mechanizmu
- Rozwiązaniem może być korzystanie z standardowego formatu z predefiniowanymi polami do wypełnienia
  - Mniejsze prawdopodobieństwo pominięcia istotnych informacji

Definicja siatki edytora

2.6 Grid facilities

2.6.1 The editor shall provide a grid facility where a matrix of horizontal and vertical lines provide a background to the editor window. This grid shall be a passive grid where the alignment of entities is the user's responsibility.
Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities 'snap-to' grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

2.6.2 When used in ‘reduce-to-fit’ mode (see 2.1), the number of units separating grid lines must be increased.
Rationale: If line spacing is not increased, the background will be very cluttered with grid lines.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

Wymaganie a powód

- Jest istotną rzeczą aby przy definiowaniu wymagań podawać powody
- Jest to pomoc dla programisty w zrozumieniu dziedziny problemu oraz dlaczego wymaganie zostało właśnie tak zdefiniowane
- Szczególnie istotne w momencie zajścia potrzeby zmiany wymagań. Dostępność powodów zmniejsza ryzyko pojawienia się niepożądanych efektów ubocznych

Przykład - definicja tworzenia węzła

3.5.1 Adding nodes to a design

3.5.1.1 The editor shall provide a facility where users can add nodes of a specified type to a design. Nodes are selected (see 3.4) when they are added to the design.

3.5.1.2 The sequence of actions to add a node should be as follows:
1. The user should select the type of node to be added.
2. The user moves the cursor to the approximate node position in the diagram and indicates that the node symbol should be added at that point.
3. The symbol may then be dragged to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

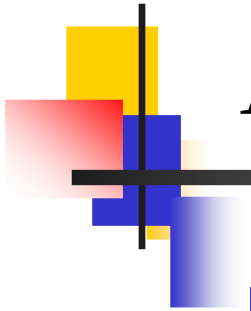
Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

Specyfikacja wymagań

- Specyfikacja rozwija bardziej szczegółowo definicję wymagań. Stąd oba te dokumenty powinny być **spójne**
- Zwykle specyfikacja zawiera model systemu stworzony w fazie analizy wymagań. Takie modele zwykle opisują części tworzonego systemu
- Często specyfikacja jest tworzona w języku naturalnym ale generuje to pewne problemy

Język naturalny - problemy

- Język naturalny jest uzależniony od tego czy osoby tworzące oraz czytające dane stwierdzenia używają tych samych określeń do opisu danej koncepcji
- Specyfikacja tworzona w języku naturalnym jest nadmiernie elastyczna i podlega różnym interpretacjom
- Struktura języka naturalnego nie wprowadza logicznego podziału wymagań



Alternatywy języka naturalnego

- Strukturyzowany język naturalny
  - Formularze oraz wzorce
- Języki opisu projektu
  - Rodzaj pseudokodu
  - Bardziej abstrakcyjne od języków programistycznych
- Języki specyfikacji wymagań
  - PSL/PSA
  - RSL

Alternatywy - c.d.

- Notacje graficzne
- Specyfikacje matematyczne
  - Oparte o formalizmy
    - Maszyna stanów
    - Sieci Petriego
    - Zbiory
  - Podstawowa zaleta – jednoznaczność
  - Wada: trudne do przyjęcia przez klienta

Faza specyfikacji wymagań

Wymagania funkcjonalne - metody

Język naturalny - najczęściej stosowany. Wady:
- niejednoznaczność powodująca różne rozumienie tego samego tekstu;
- elastyczność, powodująca wyrazić te same treści na wiele sposobów.

Konsekwencje :
- trudne wykrycie powiązanych wymagań i sprzeczności.

- **Formalizm matematyczny**. Stosuje się rzadko (dla specyficznych celów) np.: język **Z**
- **Język naturalny strukturalny**. Język naturalny z ograniczonym słownictwem i składnią. Tematy i zagadnienia wyspecyfikowane w punktach i podpunktach.
- **Tablice, formularze**. Wyspecyfikowanie wymagań w postaci (zwykle dwuwymiarowych) tablic, kojarzących różne aspekty (np. tablica ustalająca zależność pomiędzy typem użytkownika i rodzajem usługi).
- **Diagramy blokowe**: forma graficzna pokazująca cykl przetwarzania.
- **Diagramy kontekstowe**: ukazują system w postaci jednego bloku oraz jego powiązania z otoczeniem, wejściem i wyjściem.
- **Diagramy przypadków użycia**: poglądowy sposób przedstawienia aktorów i funkcji systemu.

Śledzenie wymagań

- Wymagania dają się śledzić (*ang. are traceable*) jeśli istnieją powiązania pomiędzy podobnymi wymaganiami oraz opcjonalnie pomiędzy wymaganiami a ich źródłami
  - Np. wymaganie szczegółowe (podwymaganie) odnosi się do wymagania macierzystego
- *Traceability* – własność specyfikacj wymagań ułatwiająca odszukiwanie powiązanych ze sobą
- Niektóre z narzędzi CASE wspierają śledzenie wymagań, np. umożliwiają wyszukiwanie wymagań zawierających te same określenia

Śledzenie wymagań- kroki

- Przypisanie unikalnych identyfikatorów do każdego wymagania
- Powiązanie ze sobą spokrewnionych wymagań za pomocą utworzonych identyfikatorów
- Utworzenie macierzy powiązań – dla każdego wymagania definiuje się listę powiązanych z nim wymagań. W razie potrzeby tworzy się kilka macierzy dla różnych typów powiązań

Specyfikacje w strukturyzowanym języku naturalnym

- Forma języka naturalnego z narzuconymi pewnymi ograniczeniami jako środek specyfikacji wymagań
- Takie podejście eliminuje część problemów wynikających z niejednoznaczności i elastyczności języka oraz wymusza pewien stopień jednolitości specyfikacji
- Daje dobre rezultaty w połączeniu z podejściem opartym o predefiniowane formularze

Specyfikacje oparte o formularze

- Definicja funkcji bądź komponentu
- Opis danych wejściowych oraz źródeł ich pochodzenia
- Opis danych wyjściowych oraz ich miejsca przeznaczenia
- Opis wymaganych innych komponentów/funkcji
- Warunki początkowe oraz końcowe (w razie potrzeby)
- Efekty uboczne (jeśli takowe występują)

Przykład: dodawanie węzła – specyfikacja formularzowa

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function Add node

Description Addsa node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.

Inputs Node type, Node position, Design identifier.

Source Node type and Node position are input by the user, Design identifier from the database.

Outputs Design identifier.

Destination The design database. The design is committed to the database on completion of the operation.

Requires Design graph rooted at input design identifier.

Pre-condition The design is open and displayed on the user's screen.

Post-condition The design is unchanged apart from the addition of a node of the specified type at the given position.

Side-effects None

Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1

Faza specyfikacji wymagań

Wymagania funkcjonalne - formularz

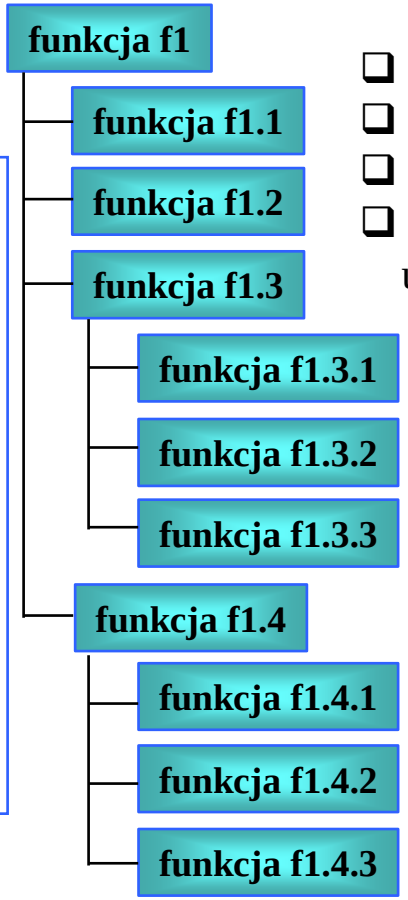
Nazwa funkcji	*Edycja dochodów pracownika*
Opis	Funkcja pozwala edytować łączne dochody podatnika uzyskane w danym roku.
Dane wejściowe	Informacje o dochodach pracowników uzyskane uzyskanych z różnych źródeł: kwoty przychodów, koszty uzyskania przychodów oraz zapłaconych zaliczek na poczet podatku dochodowego. Informacje o dokumentach opisujących dochody z poszczególnych źródeł.
Źródło danych wejściowych	Dokumenty oraz informacje dostarczone przez podatnika.
Wynik	Dane wpisywane przez pracownika firmy podatkowej.
Warunek wstępny	Kwota dochodu = kwota przychodu - kwota kosztów (zarówno dla konkretnych dochodów, jak i dla łącznych dochodów podatnika). Łączne kwoty przychodów, kosztów uzyskania dochodów oraz zapłaconych zaliczek są sumami tych kwot dla dochodów z poszczególnych źródeł.
Warunek końcowy	Jak wyżej.
Efekty uboczne	Uaktualnienie podstawy opodatkowania.
Powód	Funkcja pomaga przyśpieszyć obsługę klientów oraz zmniejszyć ryzyko popełnienia błędów.

Faza specyfikacji wymagań
Porządkowanie zadań

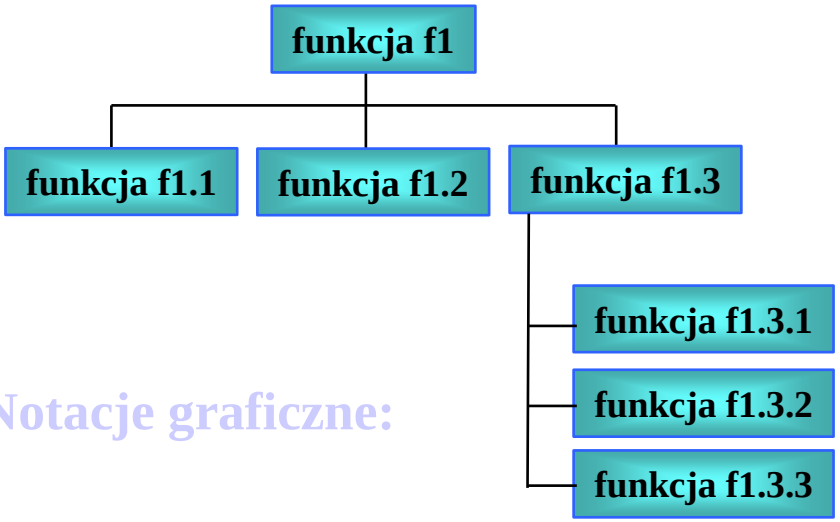
- Z reguły funkcje można rozbić na podfunkcje – **hierarchia**
- Na każdym poziomie ten sam poziom szczegółowości.
- Kolejność funkcji nie ma znaczenia.
- Niektóre funkcje mogą być wykonywane wielokrotnie.
- Specyfikujemy **tylko** funkcje widoczne dla użytkowników.

Format tekstowy:

Funkcja nadrzędna f1
- funkcja f1.1
- funkcja f1.2
- funkcja f1.3
  - funkcja f1.3.1
  - funkcja f1.3.2
  - funkcja f1.3.3
- funkcja f1.4
  - funkcja f1.4.1
  - funkcja f1.4.2
  - funkcja f1.4.3



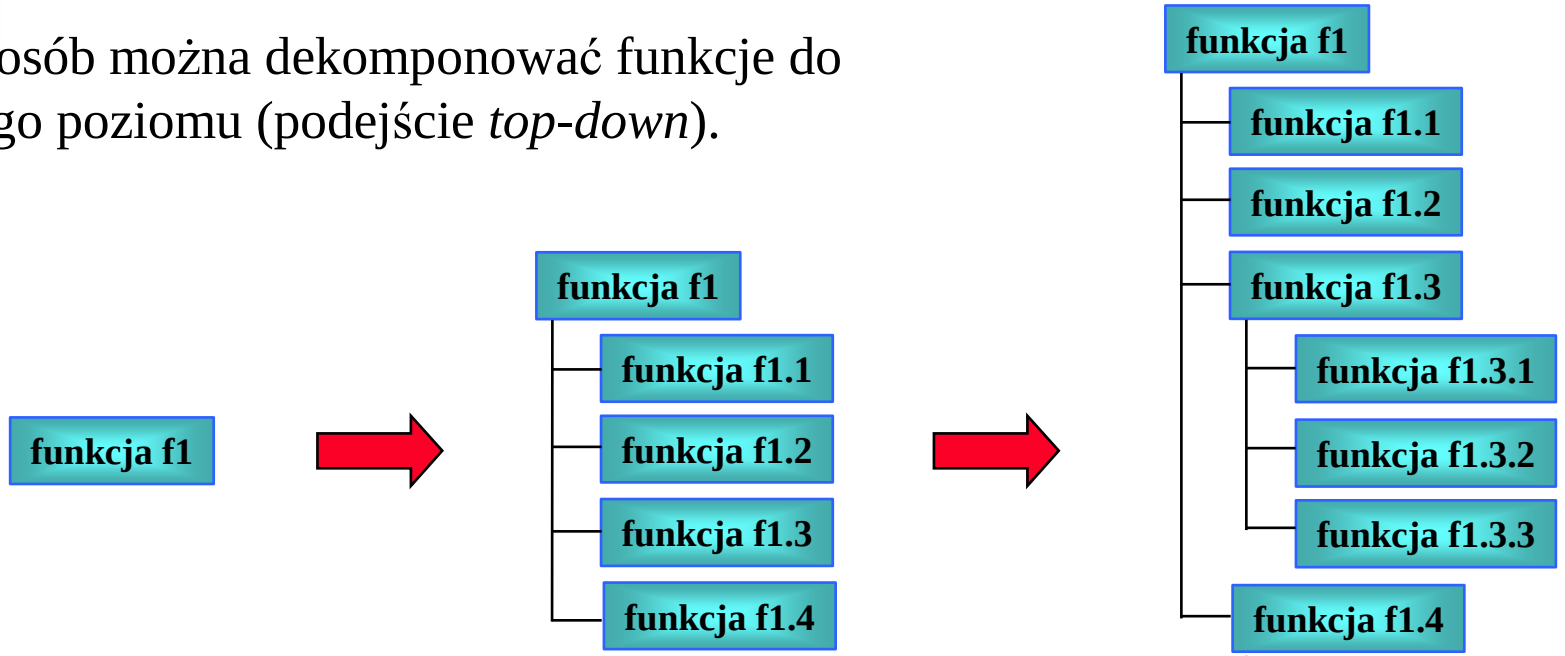
Notacje graficzne:



Faza specyfikacji wymagań

Zstępujące konstruowanie hierarchii funkcji

W ten sposób można dekomponować funkcje do dowolnego poziomu (podejście *top-down*).



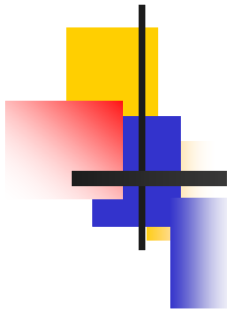
Możliwe jest również podejście odwrotne (*bottom-up*), kiedy składamy kilka funkcji bardziej elementarnych w jedną funkcje bardziej ogólną. Możliwa jest również technika mieszana.

Faza specyfikacji wymagań - wymagania funkcjonalne
Przykład: program podatkowy. Wynik wywiadu z klientem

Program ułatwia przygotowanie formularzy zeznań podatkowych (PIT-ów) oraz przechowanie informacji o źródłach przychodów i ulg. Zeznanie może być tworzone przez pojedynczego podatnika lub małżeństwa. Zeznania mogą obejmować informacje o rocznych przychodach (w przypadku małżeństwa z podziałem na przychody męża i żony) oraz o ulgach podatkowych. Przychody podzielone są na klasy ze względu na źródło uzyskania, np. poza-rolnicza działalność gospodarcza, wolny zawód,... W ramach danej klasy przychodów podatnik mógł osiągnąć szereg przychodów z różnych źródeł.

Wszystkie przychody opisane są przez kwotę przychodu, kwotę kosztów, kwotę zapłaconych zaliczek oraz kwotę dochodu. Powyższe informacje pozwalają obliczyć należny podatek oraz kwotę do zapłaty lub zwrotu. Zeznanie zawiera także informację o podatniku oraz adres Urzędu Skarbowego.

System pozwala wydrukować wzorzec zeznania zawierający wszystkie informacje, jakie podatnik musi umieścić w formularzu. Zeznanie można zabezpieczyć przed dalszymi zmianami (po złożeniu w Urzędzie Skarbowym). System pozwala na tworzenie listy podatników oraz urzędów skarbowych, które mogą być pomocne przy tworzeniu nowego zeznania. Przechowuje także informację o wszystkich złożonych zeznaniach.



Program podatkowy: hierarchia funkcji

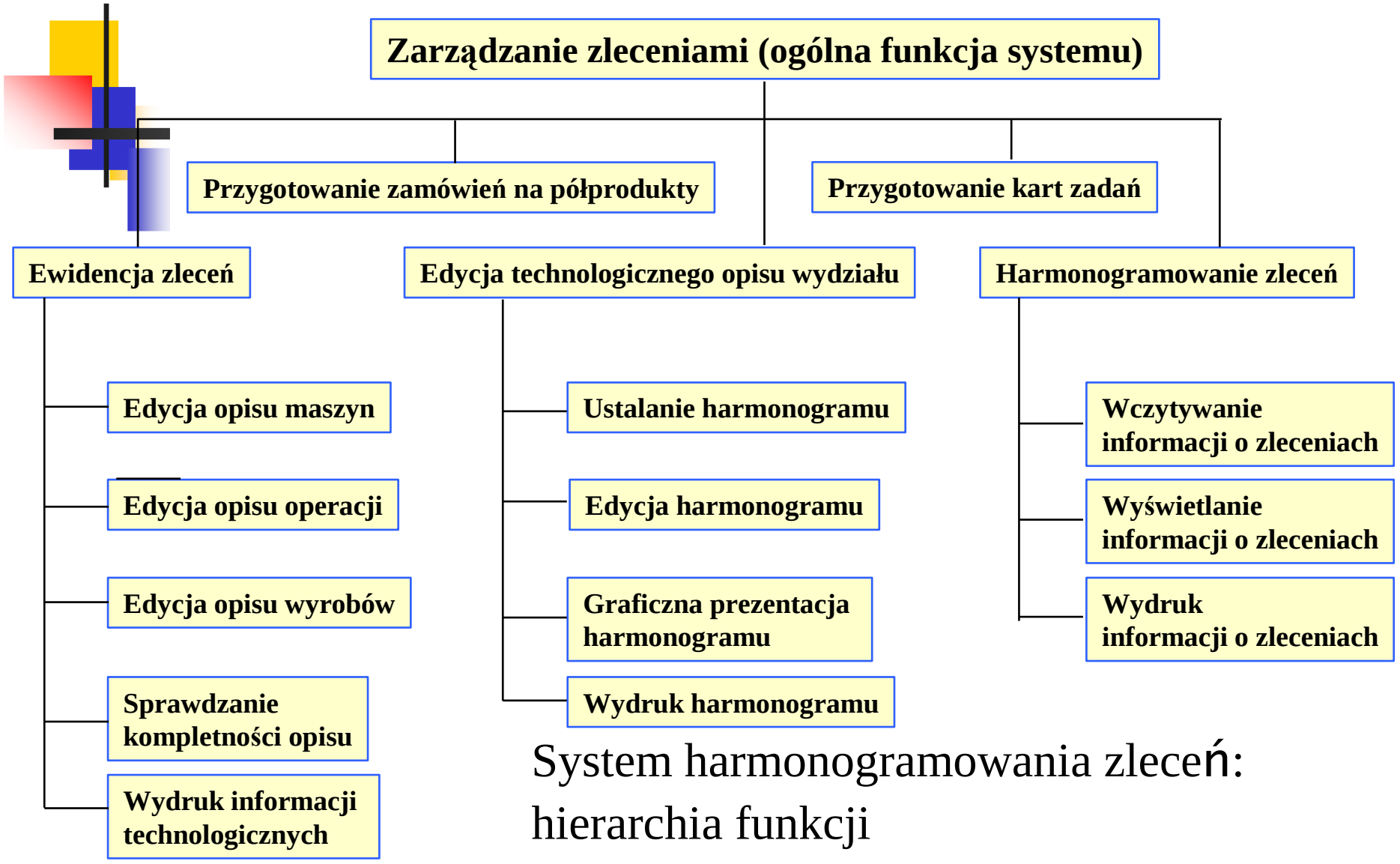
- Ewidencja podatników
- Ewidencja Urzędów Skarbowych
- Ewidencja zeznań podatkowych
  - Dodawanie zeznania
  - Usuwanie zeznania
  - Zabezpieczanie zeznania
  - Edycja zeznania
    - Edycja informacji o przychodach
    - Edycja informacji o ulgach
    - Wyświetlanie rozliczenia
    - Wydruk zeznania
  - Wyświetlenie rozliczenia (kopia)
  - Wydruk zeznania (kopia)
  - Przeglądanie zeznania (bez możliwości zmiany dla zeznań zabezpieczonych)

Przykład: system harmonogramowania zleceń
Wynik wywiadu z klientem

Zlecenia dla wydziału przygotowywane są przez dział marketingu. Zlecenie oznacza konieczność wyprodukowania konkretnej ilości pewnego wyrobu przed upływem konkretnego terminu. Czasami może być określony najwcześniejszy pożądany termin realizacji. Dział marketingu wykorzystuje własny system informatyczny, w którym między innymi umieszczane są informacje o zleceniach, pożądane jest więc, aby system harmonogramowania zleceń automatycznie odczytywał te informacje.

Wyprodukowanie danego wyrobu (realizacja zlecenia) wymaga wykonania ciągu operacji. Pewne operacje mogą być wykonywane tylko na jednym urządzeniu; w innych przypadkach możliwe jest wykorzystanie kilku maszyn, które mogą różnić się efektywnością pracy. Po wykonaniu pewnych operacji może być konieczna przerwa, zanim rozpocznie się realizacja następnych; z drugiej strony taka przerwa może trwać przez ograniczony czas. Przestawienie maszyn na inne operacje może wymagać czasu. Co pewien czas (np. raz na miesiąc) ustalany jest harmonogram niezrealizowanych zleceń.

System powinien opracować harmonogramy w formie łatwej do wykorzystania przez kadrę kierowniczą wydziału oraz przygotowywać zamówienia do magazynu na półprodukty. Zlecenia wykonane są usuwane ze zbioru niezrealizowanych zleceń.



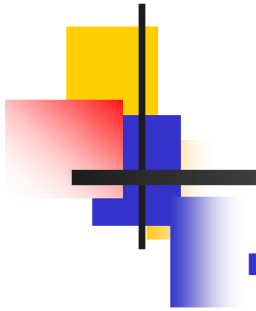
System harmonogramowania zleceń: hierarchia funkcji

Opis wymagań z wykorzystaniem PDL

- Wymagania mogą być definiowane za pomocą języka podobnego do języków programistycznych ale z większą elastycznością wyrażeń
- Najbardziej użyteczne w dwóch sytuacjach
  - Gdy działanie systemu jest zdefiniowane jako sekwencja akcji gdzie istotna jest kolejność akcji
  - Gdy zachodzi konieczność wyspecyfikowania interfejsów sprzętowych i programistycznych
- Wady
  - PDL może okazać się nie dość ekspresywny aby określić koncepcje dotyczące danej domeny problemu
  - Specyfikacja tego typu będzie postrzegana raczej jako projekt systemu a nie specyfikacja wymagań

Przykład PDL - opis działania elektronicznego zamka do drzwi

```
procedure CARDLOCK is
   code:    CODE_TYPE;
   card_id: CARD_NO_TYPE;
   card_locked, code_correct, is_inside: Boolean;
   in_room, out_room: ROOM_TYPE;
begin
  loop
    GetCardId(card_id, code, card_locked);
    if not card_locked then
       VerifyCode(code, code_correct);
       if code_correct then
           GetRooms(in_room, out_room);
           CheckIfInside(in_room, card_id, is_inside);
           if is_inside then
              OpenDoor;
              SetOutside(in_room, card_id);
              SetInside(out_room, card_id);
           end if;
       end if;
    else
       DisplayError;
    end if;
  end loop;
end CARDLOCK;
```



Specyfikacja interfejsów

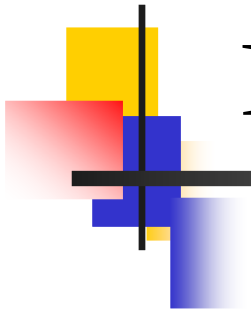
- Niemal wszystkie systemy programistyczne działają w środowisku w którym znajdują się inne systemy; współpraca odbywa się poprzez interfejsy
- W specyfikacji wymagań mogą się pojawić trzy typy interfejsów
  - Proceduralne – opis funkcji dostarczanych przez istniejące systemy
  - Interfejsy danych. Są to struktury danych wymieniane pomiędzy systemami
  - Interfejsy reprezentacji danych. Wzorce poszczególnych reprezentacji wykorzystywanych danych

Przykład interfejsu proceduralnego

```
package Print_server  is
        procedure Initialise (P: PRINTER) ;
        procedure Print (P: PRINTER ; F: PRINT_FILE ) ;
        procedure Display_print_queue (P: PRINTER ) ;
        procedure Cancel_print_job  (P: PRINTER; N: PRINT_ID) ;
        procedure Switch_printer (P1, P2: PRINTER; N: PRINT_ID) ;
end Print_server ;
```

Przykład interfejsu danych

```
type MESSAGE is record
        Sender : SYSTEM_ID;
        Receiver : SYSTEM_ID;
        Dispatch_time : DATE;
        Length: MESSAGE_LENGTH ;
        Terminator: CHARACTER ;
        Message : TEXT;
end record;
type SYSTEM_ID is range 20_000..30_000 ;
type YEAR_TYPE is range 1980..2080 ;
type DATE is record
        Seconds: NATURAL ;
        Year: YEAR_TYPE ;
end record ;
type MESSAGE_LENGTH is range 0..10_000 ;
type TEXT is array (MESSAGE_LENGTH) of CHARACTER ;
```



Przykład interfejsu reprezentacji

```
type STATE is (Halted, Waiting, Ready, Running);
for STATE use (Halted => 1, Waiting => 4, Ready => 16,
                                          Running => 256);
```

Wymagania niefunkcjonalne

- Określają cechy i ograniczenia systemu
  - Cechy to np. niezawodność, czas odpowiedzi czy wymagania dotyczące pojemności pamięci masowych.
  - Ograniczenia to np. parametry urządzeń I/O czy reprezentacje danych innych systemów
- Wymagania procesowe
  - Używanie określonego narzędzia CASE
  - Specyfikacja standardów jakości dla systemu
- Wymagania niefunkcjonalne mogą być nawet ważniejsze od funkcjonalnych
  - Nie spełnienie takich wymagań – może oznaczać brak akceptacji systemu

Klasyfikacja wymagań niefunkcjonalnych

- Wymagania dot. produktu
  - Określają w jaki sposób powinien zachowywać się system, np. szybkość wykonywania, niezawodność, itp.
- Wymagania organizacyjne
  - Są konsekwencją standardów danej organizacji. Np. standardy procesu tworzenia systemu, wykorzystanie określonych narzędzi, języków programowania, itp.
- Wymagania zewnętrzne
  - Powstają jako efekt czynników zewnętrznych w stosunku do systemu i procesu jego tworzenia, np. wymagania dot. współpracy z systemami innych organizacji; wymagania legislacyjne

Przykłady wymagań niefunkcjonalnych

- Wymaganie dot. produktu
  - 4.C.8 Komunikacja użytkownika z systemem powinna być wyrażona za pomocą standardowego zestawu znaków ASCII z standardem kodowania polskich liter ISO 8859-2.
- Wymaganie organizacyjne
  - 9.3.2 Proces tworzenia systemu oraz dostarczane dokumenty powinny być zgodne ze standardami dot. procesu oraz dokumentacji określonymi w XYZCo-SP-STAN-95.
- Wymaganie zewnętrzne
  - 7.6.5 System ma umożliwić dowolnemu użytkownikowi weryfikację danych osobowych przechowywanych w systemie. Oprogramowanie powinno dostarczyć odpowiednie procedury pozwalające na przegląd tych danych i weryfikację błędów.

Weryfikowalność wymagań

- Wymagania powinny być wyrażane w sposób umożliwiający ich **obiektywną** weryfikację
- Przykład: problem z poniższym wymaganiem polega na użyciu nieweryfikowalnego stwierdzenia „liczba błędów jest jak namniejsza”
  - System powinien być prosty w obsłudze przez niedoświadczonych użytkowników. Powinien być zorganizowany tak aby liczba błędów popełnianych przez użytkowników była jak najmniejsza
- Liczba błędów powinna była być skwantyfikowana
  - Niedoświadczeni użytkownicy powinni być w stanie obsługiwać system po dwóch godzinach treningu. Po odbyciu takiego treningu średnia liczba błędów popełnianych przez użytkownika nie powinna przekraczać dwóch na dzień.

Niektóre miary wymagań

Własność	Miara
Szybkość	Liczba transakcji na sekundę Czas odpowiedzi na zdarzenie/reakcję użytkownika Czas odświeżania ekranu
Rozmiar	KB Liczba kości pamięci RAM
Łatwość używania	Czas niezbędnego szkolenia Liczba formatek z pomocą
Niezawodność	Średni czas bezawaryjnej pracy Prawdopodobieństwo niedostępnośći Dostępność
Wydolność	(Średni) czas restartu po awarii Odsetek zdarzeń powodujących awarię Prawdopodobieństwo uszkodzenia danych przez awarię
Przenośność	Odsetek wyrażeń zależnych od platformy Liczba obsługiwanych platform

Separacja wymagań

- W zasadzie wymagania funkcjonalne oraz niefunkcjonalne powinny być rozdzielone w specyfikacji
- Ale jest to trudne ze względu na fakt że poszczególne wymagania są wyrażane jako stwierdzenia dotyczące całości systemu a nie jako ograniczenia poszczególnych funkcji
- Czasami trudno zadecydować jakiego rodzaju jest dane wymaganie
  - Przykładowo wymagania dotyczące bezpieczeństwa dotyczą własności niefunkcjonalnych ale mogą wymuszać wprowadzenie dodatkowych funkcji do systemu

Wymagania systemowe

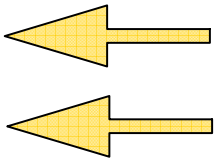
- Pewna część wymagań nakłada ograniczenia na system jako całość a nie na poszczególne funkcjonalności systemu
- Np.
  - Czas potrzebny do wyszkolenia operatora w stopniu dającym mu umiejętność biegłego posługowania się systemem nie może przekroczyć dwóch dni roboczych

IEEE Std 830-1998

- Defines „Recommended Practice” – a template
- An excerpt from **IEEE Std 830-1998**
  - A good SRS [***Software Requirements Specification***] should provide [...] specific benefits [...]:
    - Basis for agreement what the software product is to do
    - Reduce the development effort
    - A basis for estimating cost and scheduling (!)
    - Baseline for validation and verification
    - Facilitate transfer
    - Serve as a basis for enhancement

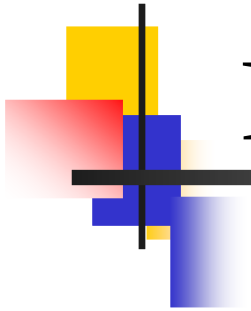
How to Use the Template?

- IEEE Std describes the practice
  - Chapter 4.3 „Characteristics of a good SRS”
    - Correct
    - Unambiguous
    - Complete
    - Consistent
    - Ranked for importance
    - **Verifiable** — *test cases, RTMX*
    - **Modifiable** — *applied doc. template*



Overview of the Template

- Introduction
- Overall description
  - Product perspective ← Environment of the product
  - Product functions ← Overview of the functionality
  - Constraints
  - Assumptions &dependencies
- Specific requirements



Podsumowanie (1/2)

- Definicja wymagań jest wykorzystywana przez klientów i użytkowników. Musi być ona napisana w języku dla nich zrozumiałym
- Zawsze przy w definicji danego wymagania powinien być zawarty powód dla którego to wymaganie zostało określone w taki a nie inny sposób
- Wymagania powinny być napisane w sposób umożliwiający ich weryfikację

Podsumowanie (2/2)

- Specyfikacja wymagań w swoim założeniu ma precyzyjnie opisywać funkcjonalność systemu oraz ograniczenia w jego działaniu. Może być ona tworzona z pomocą strukturyzowanego języka naturalnego
- Wyróżniamy trzy klasy wymagań niefunkcjonalnych: wymagania dot. produktu, wymagania procesowe oraz wymagania zewnętrzne
- Do zapisu wymagań niefunkcjonalnych używa się języka naturalnego ze względu na różnorodność i złożoność tego typu wymagań
- Istnieją standardy ułatwiające tworzenie specyfikacji wymagań – np. Std IEEE 830-1998