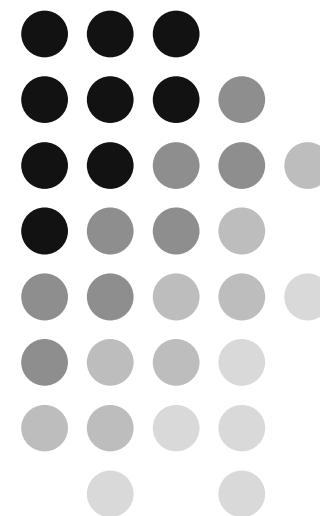


Inżynieria Oprogramowania



Wykład 4 Zarządzanie wymaganiami



Określenie wymagań



- Nie jest łatwe, nawet w przypadku dwóch osób i czegoś tak namacalnego jak kamień
- Systemy informatyczne są ze swej natury nienamacalne, abstrakcyjne i złożone.
- W ich realizacje są zaangażowane więcej niż dwie osoby.
- Klient przedstawiając niejasne wymagania „systemu kamienia” wcale może nie mieć złej woli.

Roadmap

- Typy i charakterystyka wymagań
- Inżynieria wymagań i zarządzanie wymaganiami
- Metody pozyskiwania wymagań
- Metody specyfikacji wymagań
- Model przypadków użycia i jego elementy.



Wymaganie



- Może być definiowane:
 - ... od abstrakcyjnego opisu usług lub ograniczeń systemu
 - ... do szczegółowej matematycznej specyfikacji funkcjonalności
- Z punktu widzenia celu wymaganie:
 - Może być podstawą oferty kontraktowej – czyli musi być otwarte na interpretacje
 - Może być podstawą samego kontraktu – czyli musi być jednoznaczne i szczegółowe

Podstawowe typy wymagań



- Wymagania użytkownika
 - Zdania języka naturalnego powiązane z diagramami ukazującymi usługi systemu wraz z ograniczeniami.
Pisane dla klientów
- Wymagania systemowe
 - Dokument o określonej strukturze ustalający szczegóły funkcjonalności systemu, jego usług oraz ograniczeń przy których ma działać.
 - Definiuje co ma być zaimplementowane – może być podstawą kontraktu pomiędzy zleceniodawcą a wykonawcą.

Definicje i specyfikacje

Definicja wymagań użytkownika – cech systemu

1. Oprogramowanie musi udostępniać mechanizm reprezentacji i dostępu do zewnętrznych plików tworzonych przez inne narzędzia.

Specyfikacja wymagań systemowych

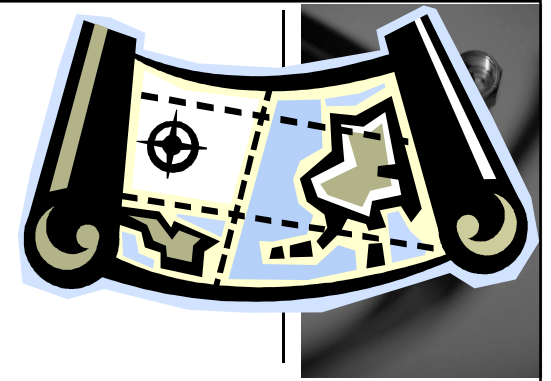
1.1 Użytkownik powinien mieć możliwość określenia typu zewnętrznego pliku

1.2 Każdy zewnętrzny plik może mieć powiązane narzędzie, które może być do niego zastosowane

1.3 Każdy typ zewnętrznego pliku powinien być reprezentowany przez specyficzną ikonę w interfejsie użytkownika

1.4 Użytkownik powinien mieć możliwość definiowania ikony powiązanej z typem zewnętrznego pliku

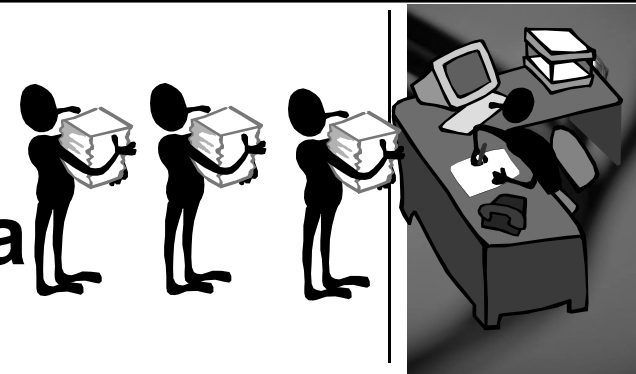
1.5 Efektem zaznaczenia przez użytkownika ikony reprezentującej zewnętrzny plik powinno być zastosowanie narzędzia powiązanego z typem zewnętrznego pliku.



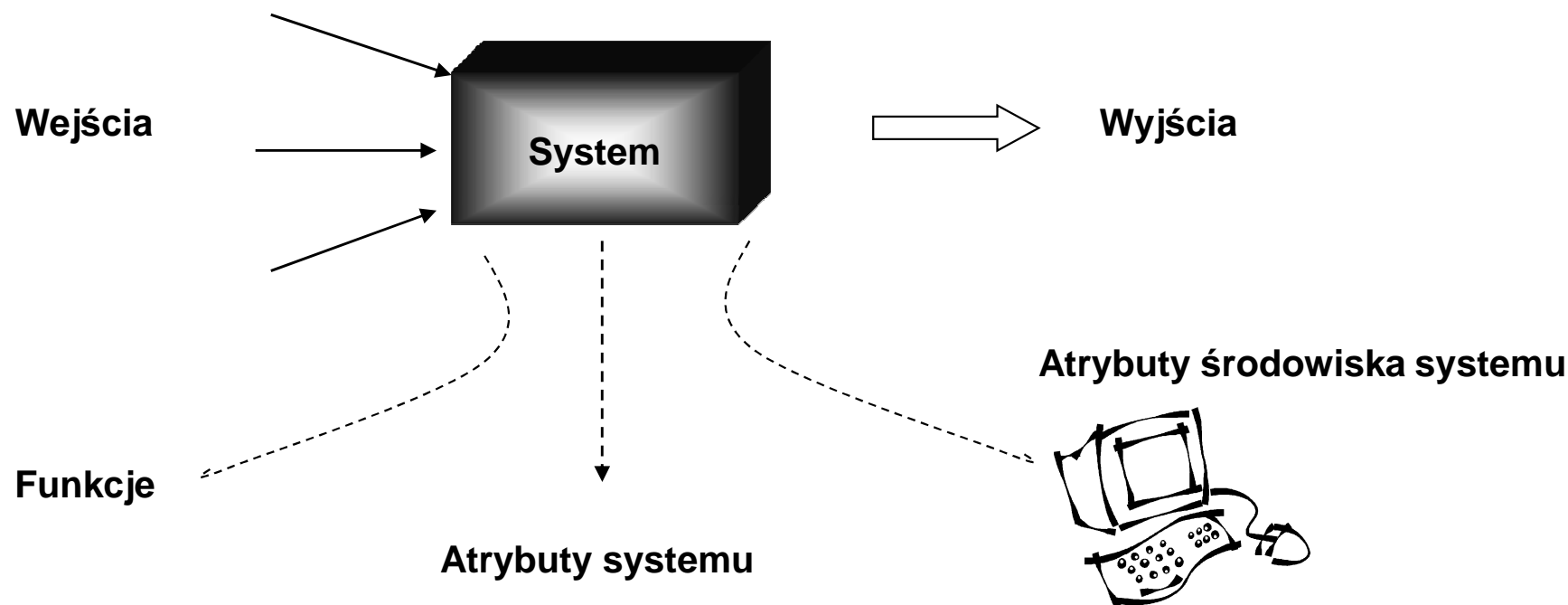
Inny przykład...

Cechy (wymagania użytkownika)	Wymagania systemowe
Cecha 63 – system wykrywania błędów dostarczy informacji trendu, które pomogą użytkownikowi ocenić status przedsięwzięcia	WO63.1 – informacje trendu będą dostarczone w raporcie histogramu, gdzie czas oznaczono na osi x, a liczbę defektów na osi y
	WO63.2 – użytkownik może wprowadzić okres wyznaczania trendu w jednostkach dni, tygodni lub miesięcy.
	WO63.3 – raport trendu powinien być zgodny z przykładem pokazanym na rysunku 3.1 (s. 56)

Wymagania stawiane oprogramowaniu - charakterystyka



Wymagania są tymi „*rzeczami*”, które należy zdefiniować, aby w pełni opisać co robi system traktowany jako czarna skrzynka.





Charakterystyka wymagań

- Rodzaje wymagań:
 - **Wymagania funkcjonalne** – zachowanie systemu (jakie akcje ma wykonywać system bez brania pod uwagę ograniczeń)
 - **Wymagania нефunkcjonalne** – ograniczenia, które mają wpływ na wykonywane zadania systemu.
 - **Ograniczenia projektowe** – ograniczenia dotyczące projektowania systemu, nie mające wpływu na jego zachowanie ale, które muszą być spełnione, aby dotrzymać zobowiązań technicznych, ekonomicznych lub wynikających z umowy



Wymagania funkcjonalne

Określenie wymagania funkcjonalnych obejmuje następujące zadania:

- Określenie wszystkich rodzajów użytkowników, którzy będą korzystać z systemu.
- Określenie wszystkich rodzajów użytkowników, którzy są niezbędni do działania systemu (obsługa, wprowadzanie danych, administracja).
- Dla każdego rodzaju użytkownika określenie funkcji systemu oraz sposobów korzystania z planowanego systemu.
- Określenie systemów zewnętrznych (obcych baz danych, sieci, Internetu), które będą wykorzystywane podczas działania systemu.
- Ustalenie struktur organizacyjnych, przepisów prawnych, statutów, zarządzeń, instrukcji, itd., które pośrednio lub bezpośrednio określają funkcje wykonywane przez planowany system.

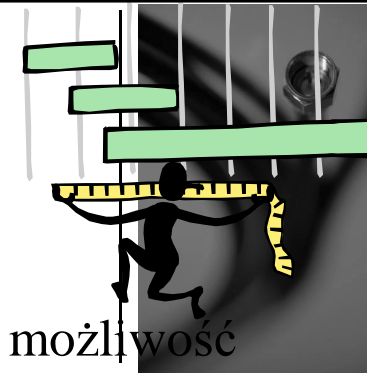


Wymagania niefunkcjonalne

Opisują ograniczenia, przy których system ma realizować swoje funkcje.

- **Użyteczność (ang. Usability)**
 - Wymagany czas szkolenia, czas wykonania poszczególnych zadań, ergonomia interfejsu, pomoc, dokumentacja użytkownika
- **Niezawodność (ang. Reliability)**
 - Dostępność, średni czas międzyawaryjny (MTBF), średni czas naprawy (MTTR), dokładność, maksymalna liczba błędów.
- **Efektywność (ang. Performance)**
 - Czas odpowiedzi, przepustowość, czas odpowiedzi, konsumpcja zasobów, pojemność.
- **Zarządzalność (ang. Supportability)**
 - Łatwość modyfikowania, skalowalność, weryfikowalność, kompatybilność, możliwości konfiguracyjne, serwisowe, przenaszalność.

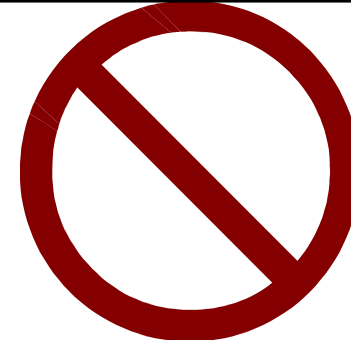
Weryfikacja wymagań niefunkcjonalnych



Wymagania niefunkcjonalne powinny być **weryfikowalne**, tj. powinna istnieć możliwość sprawdzenia czy system je rzeczywiście spełnia. Np. wymaganie “**system ma być łatwy w obsłudze**” nie jest weryfikowalne.

<i>Cecha</i>	<i>Weryfikowalne miary</i>
Wydajność	Liczba transakcji obsłużonych w ciągu sekundy Czas odpowiedzi Szybkość odświeżania ekranu
Zasoby	Wymagana pamięć RAM Wymagana pamięć dyskowa
Łatwość użytkowania	Czas niezbędny dla przeszkolenia użytkowników Liczba stron dokumentacji
Niezawodność	Prawdopodobieństwo błędu podczas realizacji transakcji Średni czas pomiędzy błędnymi wykonaniami Dostępność (procent czasu w którym system jest dostępny) Czas restartu po awarii systemu Prawdopodobieństwo zniszczenia danych w przypadku awarii
Przenaszalność	Procent kodu zależnego od platformy docelowej Liczba platform docelowych Koszt przeniesienia na nową platformę

Ograniczenia projektowe



- Ten rodzaj wymagań nakłada ograniczenia na projekt systemu lub proces, którego używamy do budowy.
 - *Produkt musi spełniać normę ISO 601*
 - *Proces wytwarzania musi być zgodny ze standardem DOD 1200-34*
- Mają często negatywny wpływ na elastyczność projektantów
 - *Użyj systemu Oracle, programuj w Visual Basic, użyj biblioteki klas XYZ.*

Wymagania a inżynieria wymagań



- **Wymagania** – opis usług i ograniczeń systemu generowany w procesie inżynierii wymagań
- **Inżynieria wymagań** – proces pozyskiwania, analizowania, dokumentowania oraz weryfikacji wymagań
 - ... czyli zarządzania wymaganiami





Dlaczego inżynieria wymagań? (1)

- **Niezbędna umiejętność – pozyskiwanie wymagań od użytkowników i klientów.**
 - Zamiana celów klienta na konkretne wymagania zapewniające osiągnięcie tych celów.
 - Klient rzadko wie, jakie wymagania zapewnią osiągnięcie jego celów.
 - Jest to tak naprawdę proces, konstrukcji zbioru wymagań zgodnie z postawionymi celami.

Dlaczego inżynieria wymagań? (2)



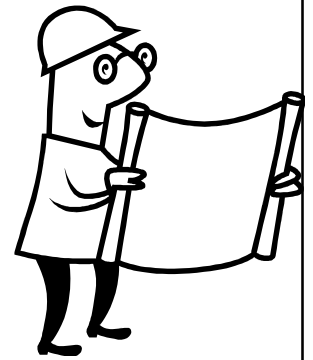
- **Organizacja i dokumentowanie wymagań**
 - Setki, jeżeli nie tysiąca wymagań jest prawdopodobnie związanych z systemem
 - Według klienta prawdopodobnie wszystkie z nich są najważniejsze w 100%.
 - Większość z nas nie może pamiętać więcej niż kilkadziesiąt informacji jednocześnie.



Dlaczego inżynieria wymagań? (3)

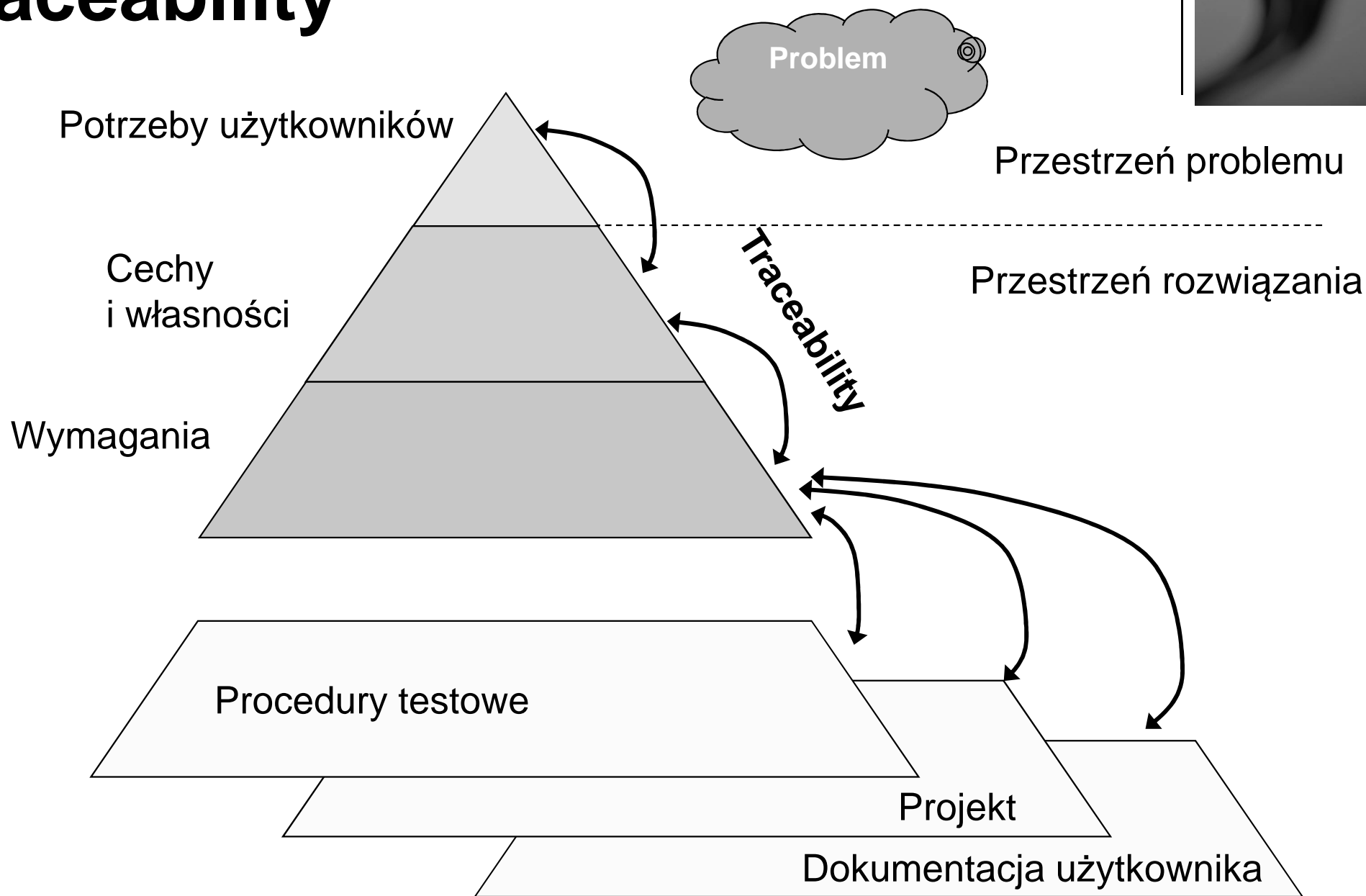
- Śledzenie, kontrola dostępu i weryfikacja wymagań
 - Którzy członkowie zespołu są odpowiedzialni za wymaganie nr 278, a którzy mogą je zmodyfikować lub usunąć?
 - Jeżeli wymaganie nr 278 będzie zmodyfikowane, jaki to będzie miało wpływ na inne wymagania?
 - Kiedy możemy być pewni, że ktoś napisał kod w systemie, spełniający wymaganie nr 278 i które testy z ogólnego zestawu testów są przeznaczone do sprawdzenia, że wymaganie rzeczywiście zostało spełnione?

Zarządzanie wymaganiami

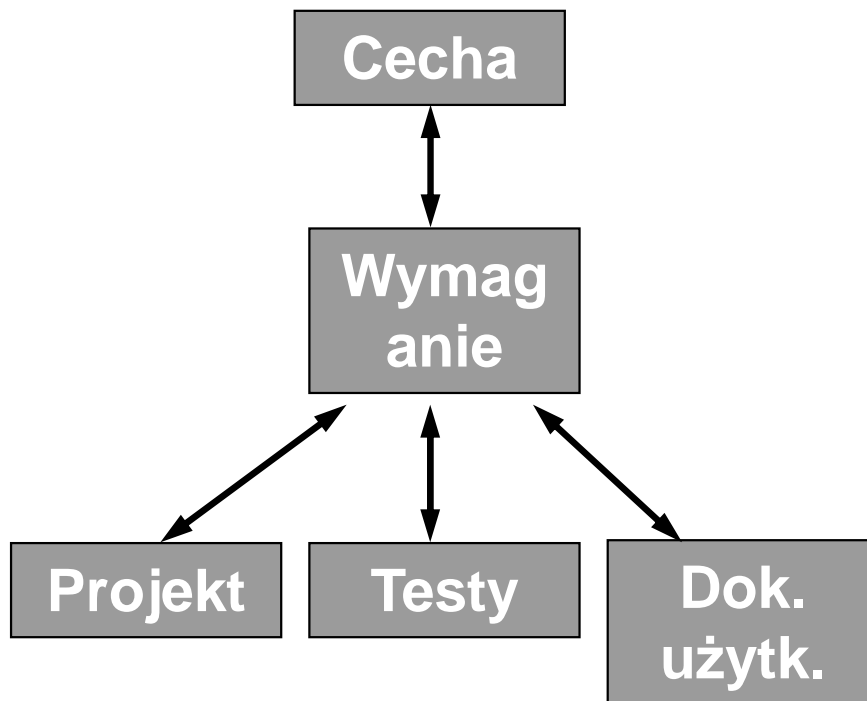


- Zarządzanie wymaganiami dotyczy procesu translacji potrzeb klientów w zbiór kluczowych cech i własności systemu.
- Następnie ten zbiór jest przekształcany w specyfikację funkcjonalnych i niefunkcjonalnych wymagań.
- Specyfikacja jest następnie przekształcana w projekt, procedury testowe i dokumentację użytkownika.

Zarządzanie wymaganiami i traceability

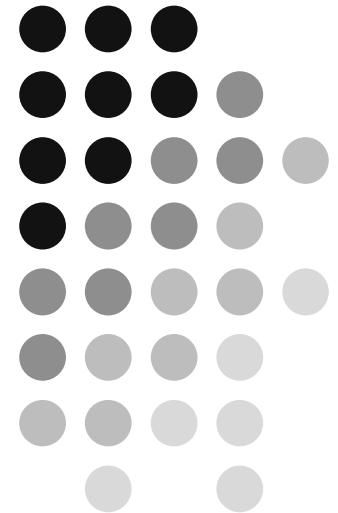
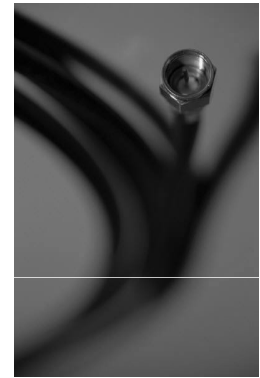


Traceability



- Oszacowanie wpływu zmiany w wymaganiach na projekt.
- Oszacowanie wpływu jaki na wymagania będzie miał „zawalony” test (jeżeli system nie przeszedł testu wymagania mogą nie być spełnione)
- Zarządzanie ramami projektu
- Zweryfikowanie czy wszystkie wymagania zostały spełnione przez implementację systemu
- Zweryfikowanie czy system robi tylko to co miał robić.
- Zarządzanie zmianami.

Pozyskiwanie wymagań



Bariery pozyskiwania wymagań



- **Syndrom „tak, ale”**
 - *„Tak, ale hmmm, teraz kiedy go już widzę, czy będzie można...? Czy nie lepiej byłoby, gdyby...? Przecież jeżeli..., to trudno będzie...”*
- **Syndrom „nieodkrytych ruin”**
- **Syndrom „użytkownik i programista”**
 - **Użytkownicy, nie wiedzą, czego chcą, lub wiedzą, co chcą, ale nie mogą tego wyrazić.**
 - **Użytkownicy uważają, że wiedzą, czego chcą, dopóki programiści nie dadzą im tego, o czym mówili, że chcą.**
 - **Analitycy uważają, że rozumieją problemy użytkownika lepiej niż on sam.**
 - **Wszyscy uważają, że inni mają określone motywacje polityczne.**

Przykładowe techniki pozyskiwania wymagań



- Śledzenie (ang. *Shadowing*)
- Wywiady (ang. *Interviewing*)
- Warsztaty pozyskiwania wymagań (ang. *Focus groups*)
- Przeglądy i ankiety (ang. *Surveys*)
- Instruowanie przez użytkowników (ang. *User instructions*)
- Prototypowanie (ang. *Prototyping*)

Shadowing - śledzenie



- Polega na obserwowaniu użytkownika podczas wykonywania przez niego codziennych zadań w rzeczywistym środowisku.
 - Rodzaje – pasywne i aktywne
 - Zalety
 - Informacja z pierwszej ręki w odpowiednim kontekście
 - Łatwiejsze zrozumienie celu określonego zadania
 - Możliwość zebrania nie tylko informacji ale i innych elementów środowiska (np. dokumenty, zrzuty ekranowe istniejącego rozwiązania)
 - Zebranie informacji na temat istniejącego rozwiązania oraz tego czy i w jaki sposób jest ono frustrujące dla użytkowników
 - Wady
 - Nieodpowiednie dla zadań wykonywanych sporadycznie, zadań związanych z zarządzaniem, zadań długoterminowych, zadań nie wymagających działania użytkowników.



Interviewing - wywiady

- Spotkanie członka zespołu projektowego z użytkownikiem lub klientem
 - Zalety
 - Można uzyskać dużo informacji o problemach i ograniczeniach obecnej sytuacji, która ma być zmieniona przez nowy system.
 - Daje możliwość uzyskania wielu informacji, które niekoniecznie można uzyskać przy wykorzystaniu techniki śledzenia.
 - Wady
 - Zależna od umiejętności i zaangażowania obu stron

Focus groups - warsztaty pozyskiwania wymagań



- Sesja w której wymagania ustala się w większej grupie (np. burza mózgów, odgrywanie ról, itp.)
 - Zalety:
 - Pozwala na uzyskanie szczegółowych informacji o szerszym kontekście aktywności biznesowych. Brak informacji u jednego z uczestników może być uzupełniona przez pozostałych.
 - Wady:
 - Wymaga zebrania większej grupy w jednym miejscu (rozproszenie geograficzne)
 - Wymaga umiejętności prowadzenia dyskusji przez prowadzącego.

Surveys – przeglądy, pomiary (1)



- Zbiór pytań utworzony w celu zebrania określonych informacji
 - Kwestionariusze
 - Wymagane przy rejestracji
 - Informacje zwrotne
 - Arkusze badania poziomu zadowolenia z produktu
 - Zalety
 - Anonimowe wyrażanie swojego zdania
 - Odpowiedzi tabelaryzowane i łatwe w analizie
 - Wady
 - Pracochłonne
 - Wymagana profesjonalna wiedza w celu tworzenia i analizy

Surveys – przeglądy, pomiary (2)



- Może być pomocne w pozyskaniu następujących informacji
 - Struktura organizacyjna, polityka działania, praktyki stosowane w pracy
 - Frustracje związane z wykonywaną pracą
 - Wymagania specjalne związane z oprogramowaniem, sprzętem
 - Efektywność programu szkoleniowego
 - Stopień zadowolenia z produktu

User instructions – instruowanie przez użytkowników (1)



- W technice tej użytkownicy instruują przeprowadzającego badanie w sposobie w jaki wykonują określone zadania.
 - Zalety
 - Widzenie procesu z perspektywy użytkownika
 - Zebranie informacji na temat doświadczenia pojedynczych osób
 - Wady
 - Może być czasochłonne
 - Może być frustrująca dla badacza, jeżeli użytkownik nie jest przyzwyczajony do uczenia kogoś
 - Różne osoby mogą wykonywać to samo zadanie w różny sposób

User instructions – instruowanie przez użytkowników (2)

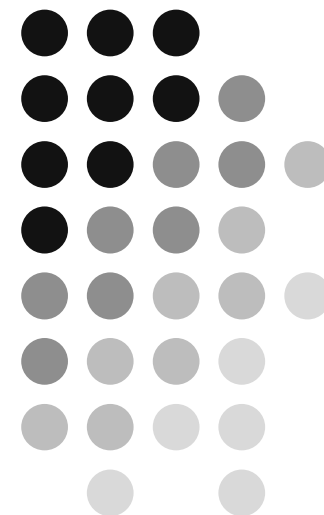


- Może być pomocne w pozyskaniu następujących informacji
 - Projekt interfejsu użytkownika
 - Wymagania związane z procesem szkoleniowym
 - Kryteria wydajnościowe systemu
 - Wpływ środowiska na wykonywane zadania

Specyfikacja wymagań



**Czy wymaganie nr. 31.2 jest
sprzeczne z wymaganiem nr.
34.3, a wymaganie 22.1 jest
powiązane z wymaganiem 14.2?**



Metody specyfikacji wymagań



Język naturalny - najczęściej stosowany. Wady: *niejednoznaczność* powodująca różne rozumienie tego samego tekstu; elastyczność, powodująca wyrazić te same treści na wiele sposobów. Utrudnia to wykrycie powiązanych wymagań i powoduje trudności w wykryciu sprzeczności.

Formalizm matematyczny. Stosuje się rzadko (dla specyficznych celów).

Język naturalny strukturalny. Język naturalny z ograniczonym słownictwem i składnią. Tematy i zagadnienia wyspecyfikowane w punktach i podpunktach.

Metody specyfikacji wymagań



Tablice, formularze. Wyszpecyfikowanie 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.

Model przypadków użycia: poglądowy sposób przedstawienia aktorów i funkcji systemu. Uważa się go za dobry sposób specyfikacji wymagań funkcjonalnych.

Dokument Specyfikacji Wymagań Oprogramowania (SWO)



- Wymagania powinny być zebrane w dokumencie – specyfikacji wymagań oprogramowania.
- Dokument ten powinien być podstawą do szczegółowego kontraktu między klientem a producentem oprogramowania.
- Powinien także pozwalać na weryfikację stwierdzającą, czy wykonany system rzeczywiście spełnia postawione wymagania.
- Powinien to być dokument zrozumiały dla obydwu stron.
- Tekstowy dokument SWO jest najczęściej powiązany z innymi formami specyfikacji wymagań.

Zawartość dokumentu SWO



**Informacje
organizacyjne**

**Przykładowy
spis
treści**

Streszczenie

Spis treści

Status dokumentu (roboczy, 1-sza faza, zatwierdzony, ...)

Zmiany w stosunku do wersji poprzedniej

1. Wstęp

1.1. Cel

1.2. Zakres

1.3. Definicje, akronimy i skróty

1.4. Referencje, odsyłacze do innych dokumentów

1.5. Krótki przegląd

2. Ogólny opis

2.1. Walory użytkowe i przydatność projektowanego systemu

2.2. Ogólne możliwości projektowanego systemu

2.3. Ogólne ograniczenia

2.4. Charakterystyka użytkowników

2.5. Środowisko operacyjne

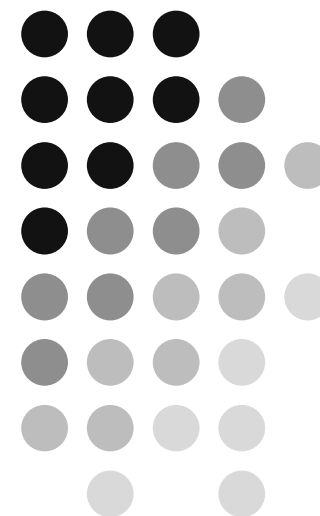
2.6. Założenia i zależności

3. Specyficzne wymagania

3.1. Wymagania co do możliwości systemu

3.2. Przyjęte lub narzucone ograniczenia.

Model Przypadków Użycia



Zachowanie systemu

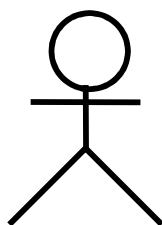


- Zachowanie systemu jest opisem tego jak system działa i reaguje. Jest to widoczny z zewnątrz przejaw aktywności systemu.
- Model przypadków użycia opisuje zachowanie systemu.
- Model przypadków użycia opisuje:
 - System
 - Jego środowisko
 - Związki pomiędzy systemem a jego środowiskiem

Podstawowe elementy modelu przypadków użycia.

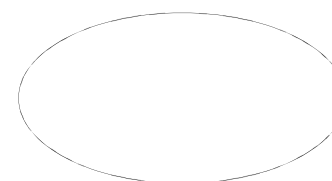


- Przypadki użycia
- Aktorzy
- Specyfikacje przypadków użycia



Aktor

Reprezentuje rolę, którą może grać w systemie jakiś jego użytkownik; (np. kierownik, urzędnik, klient)



Przypadek użycia

Reprezentuje sekwencję operacji inicjowaną przez aktora, niezbędnych do wykonania zadania zleconego (zainicjowanego) przez aktora, np. potwierdzenie pisma, złożenie zamówienia, itp.



Aktor - konkretny byt czy rola?

Aktor modeluje grupę osób pełniących pewną rolę, a nie konkretną osobę.

Użytkownik

Aktor

Przypadek użycia

Może grać rolę

zleca

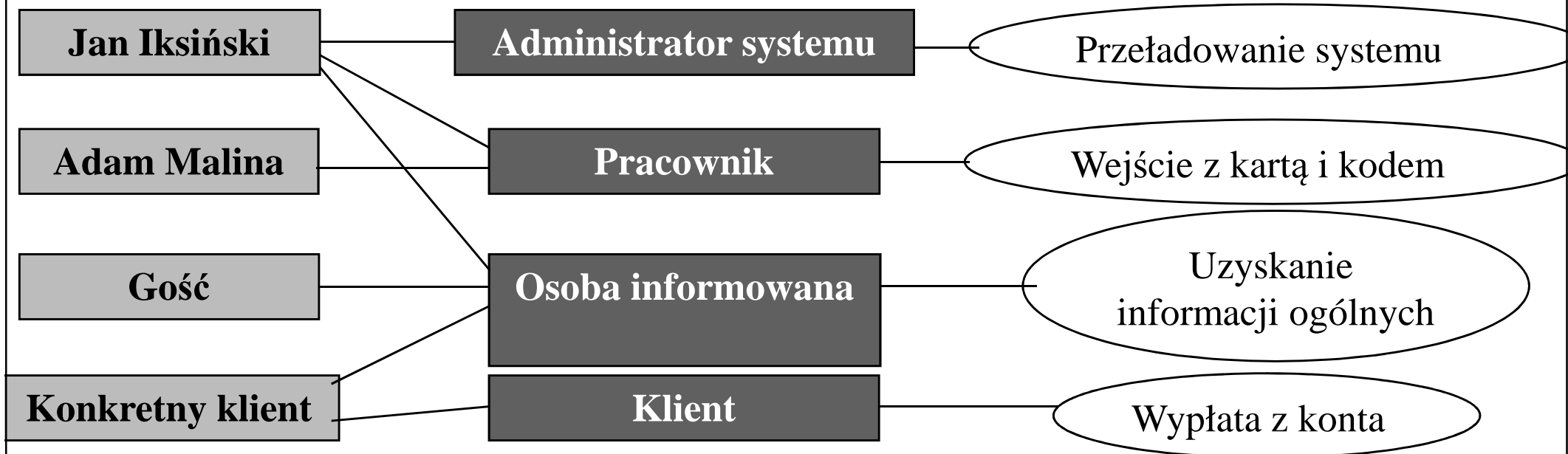
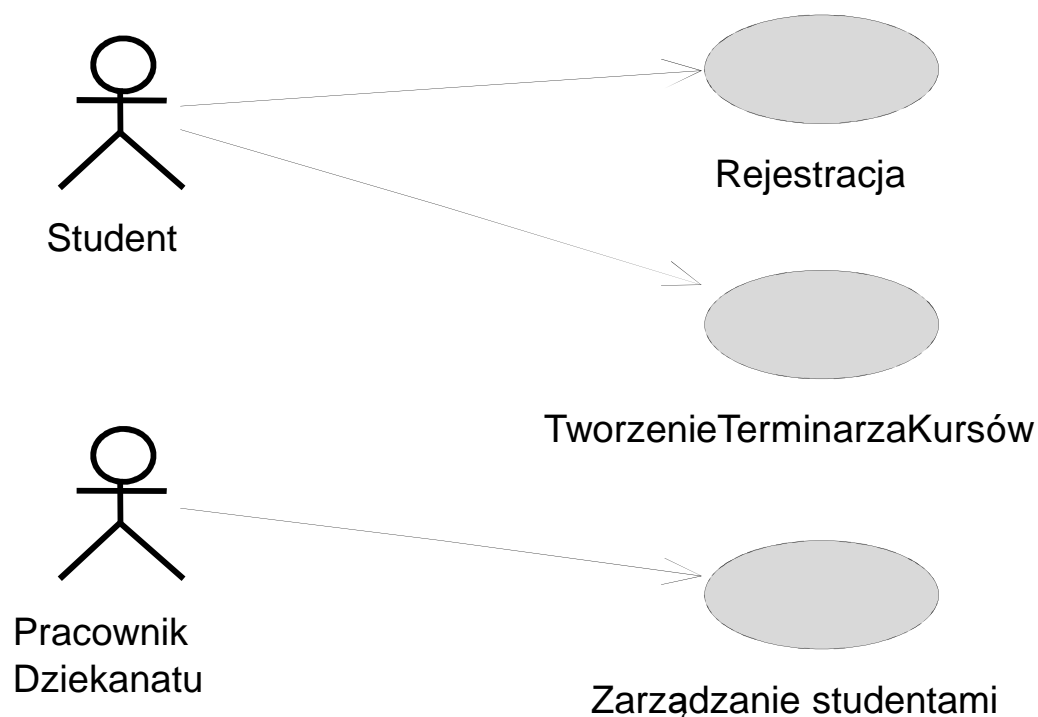


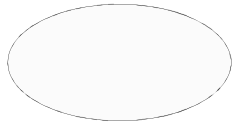
Diagram przypadków użycia – kontekst systemu



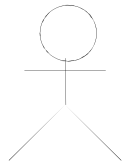
- Opisuje funkcje systemu w terminach przypadków użycia.



Notacja



Rejestracja

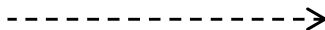


Student



**weryfikacja
klienta**

<<include>>



<<extends>>



Przypadek użycia

Aktor

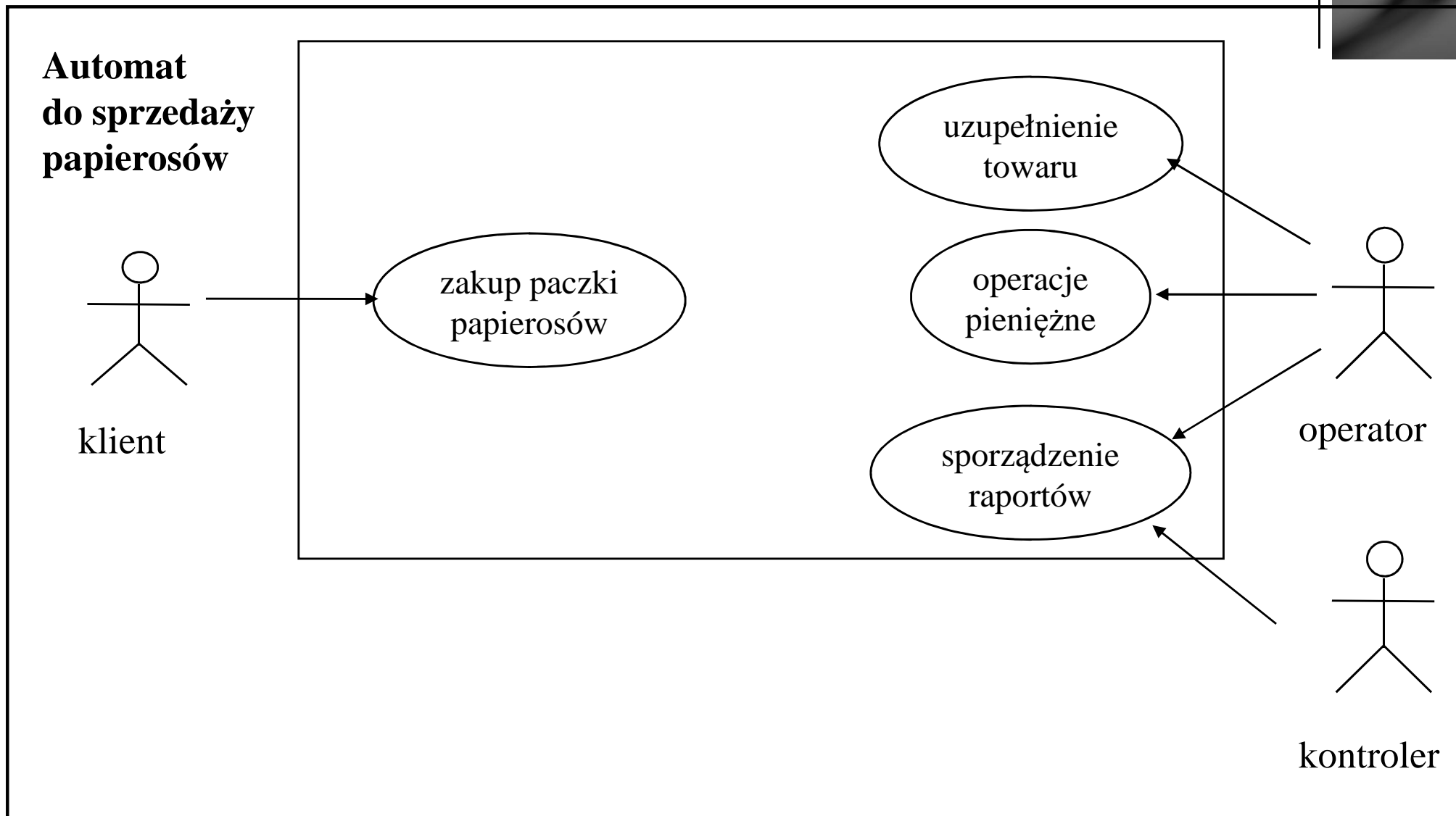
Interakcja.

Blok ponownego użycia

Zależności (pomiędzy przypadkami użycia)



Diagram przypadków użycia - przykład



Podsumowanie



- Zarządzanie wymaganiami - pozyskiwanie, analizowanie, dokumentowanie oraz weryfikacja wymagań
- Wymagania: funkcjonalne i нефunkcjonalne
- Pozyskiwanie wymagań jest procesem trudnym, wymagającym odpowiedniego przygotowania



Do poczytania

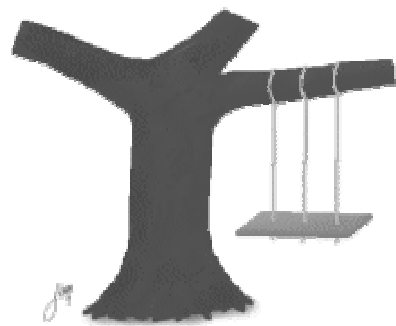
- Dąbrowski, Subieta: *Podstawy Inżynierii Oprogramowania*, rozdział 3.
- Więcej nt. zarządzania wymaganiami:
 - Leffingwell D., Widrig D., *Zarządzanie wymaganiami*, WNT, Wa-wa, 2003.
- Model przypadków użycia:
 - Booch G., Rumbaugh J., Jacobson I.: *UML przewodnik użytkownika*, WNT, Wa-wa, 2001.
- Spis praw użytkownika (A Computer User's Manifesto) - www.businessweek.com/1998/39/b3597037.htm

Dokumenty/Narzędzia

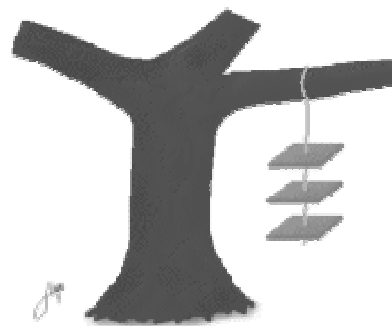


- Rekomendowana przez IEEE struktura dokumentu wymagań (*IEEE recommended practice for software requirements specifications IEEE Std 830-1998*)
- Rational Requisite Pro www-306.ibm.com/software/rational
– narzędzie do zarządzania wymaganiami.

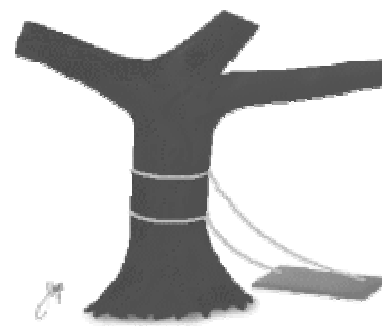
Na koniec metafora ... ku przestrodze



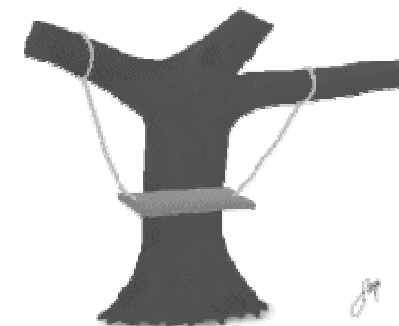
To co klient zamówił



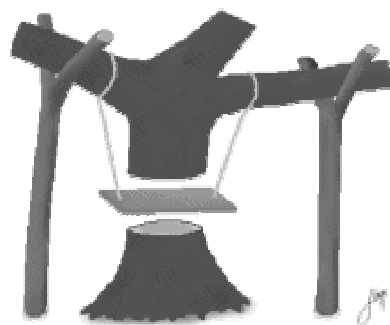
To co analityk zrozumiał



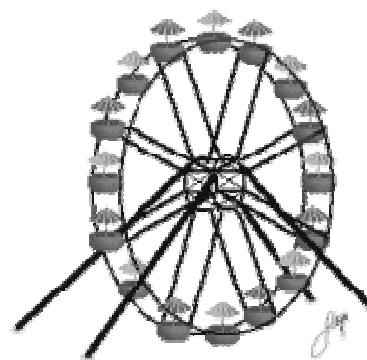
To co projekt opisywał



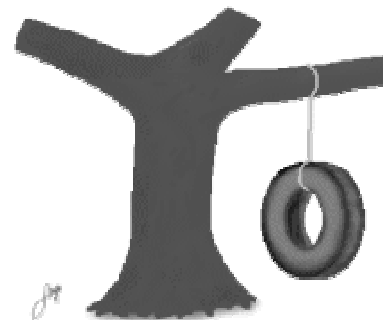
To co zrobili programiści



Projekt po uruchomieniu
i wdrożeniu



To, za co zapłacił
klient



To, czego klient
potrzebował



Praktyczne
zastosowanie
projektu