# Modelowanie i analiza obiektowa

Wykład 3

Zarządzanie wymaganiami

# Problem „kamienia”

# Efekt...

# Modyfikacja...

# Modyfikacja...

...


- W końcu może się okazać, że klient cały czas myślał o małym szarym kawałku granitu
- Może nie był pewien, czego chce, poza tym, że był to mały szary kawałek granitu?
- A może jego wymagania uległy zmianie pomiędzy dostawą pierwszego (dużego) kamienia a dostawą ostatniego (małego szarego)?

# 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 (dlaczego miałby mieć?).

# 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 definiowana 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):**

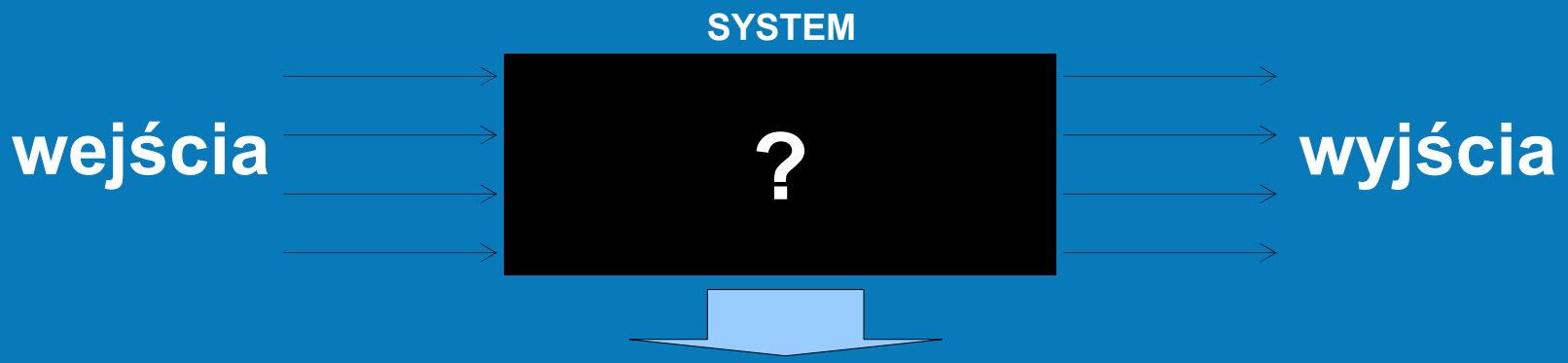
Cecha 63 – system wykrywania błędów dostarczy informacji o trendzie, które pomogą użytkownikowi ocenić status przedsięwzięcia

**Wymagania systemowe:**

WO63.1 – informacje o trendzie będą dostarczone w raporcie histogramu, gdzie czas oznaczono n 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 „*elementami systemu*”, które należy zdefiniować, aby w pełni opisać co robi system traktowany jako czarna skrzynka.




**funkcje systemu**

**atrybuty systemu**

**atrybuty środowiska systemu**

# Charakterystyka wymagań

- Rodzaje wymagań:
- Wymagania funkcjonalne – zachowanie systemu (jakie akcje ma wykonywać system bez brania pod uwagę ograniczeń)
- Wymagania niefunkcjonalne – 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.

| Cecha | Weryfikowalne miary |
|---|---|
| **Wydajność** | • Liczba transakcji obsłużonych w ciągu sekundy<br>• Czas odpowiedzi<br>• Szybkość odświeżania ekranu |
| **Zasoby** | • Wymagana pamięć RAM<br>• Wymagana pamięć dyskowa |
| **Łatwość użytkowania** | • Czas niezbędny dla przeszkolenia użytkowników<br>• Liczba stron dokumentacji |
| **Niezawodność** | • Prawdopodobieństwo błędu podczas realizacji transakcji<br>• Średni czas pomiędzy błędnymi wykonaniami<br>• Dostępność (procent czasu w którym system jest dostępny)<br>• Czas restartu po awarii systemu<br>• Prawdopodobieństwo zniszczenia danych w przypadku awarii |
| **Przenośność** | • Procent kodu zależnego od platformy docelowej<br>• Liczba platform docelowych<br>• 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 jest potrzebna? (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 osiągniecie jego celów.

Jest to tak naprawdę proces, konstrukcji zbioru wymagań zgodnie z postawionymi celami.

## Dlaczego inżynieria wymagań jest potrzebna? (2)

### Organizacja i dokumentowanie wymagań

- Gros wymagań jest prawdopodobnie związanych z systemem
- Według klienta prawdopodobnie wszystkie z nich są najważniejsze.
- Większość z nas nie może pamiętać więcej niż kilkadziesiąt informacji jednocześnie.

## Dlaczego inżynieria wymagań jest potrzebna? (3)

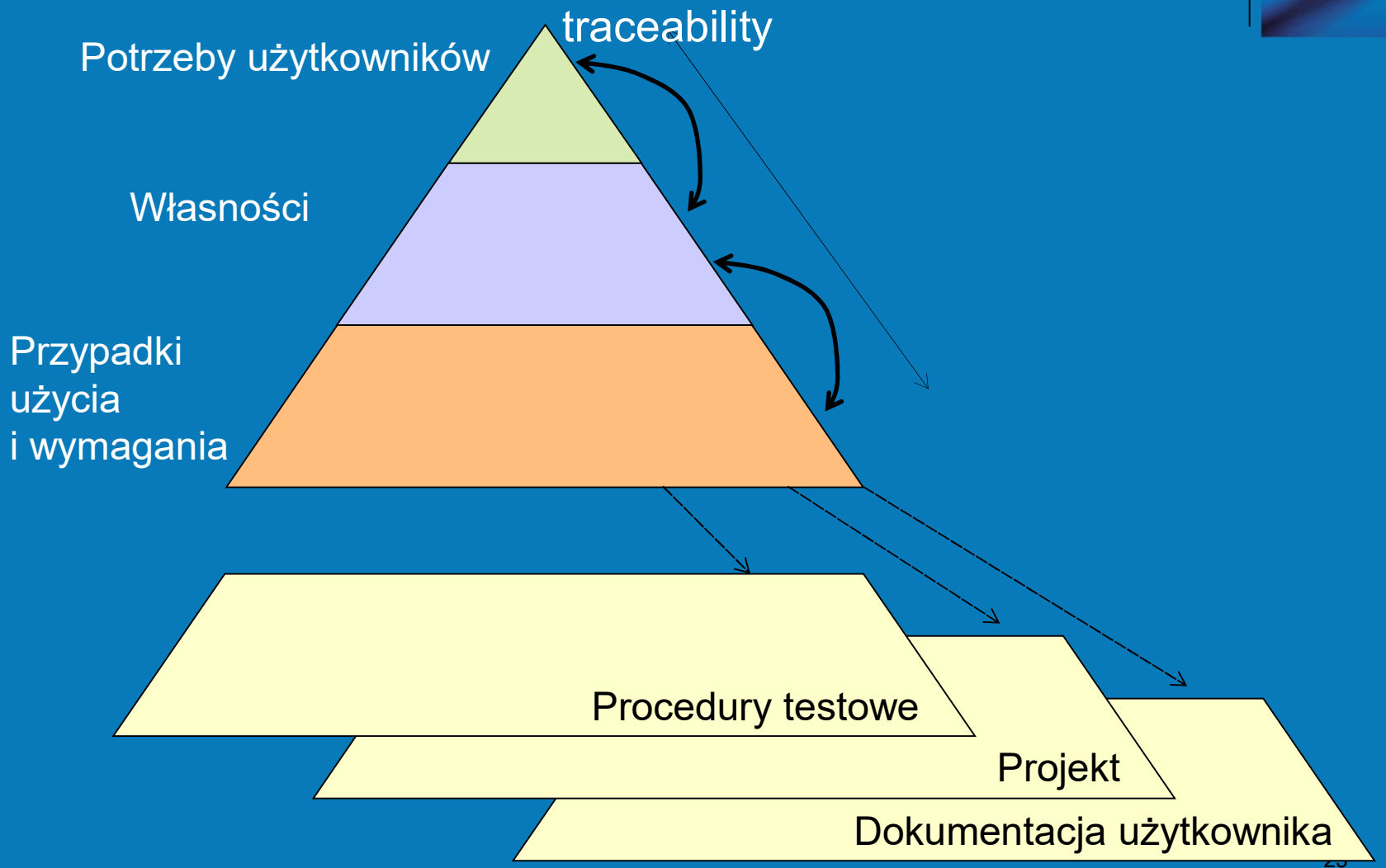
**Śledzenie, kontrola dostępu i weryfikacja wymagań:**

- 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?

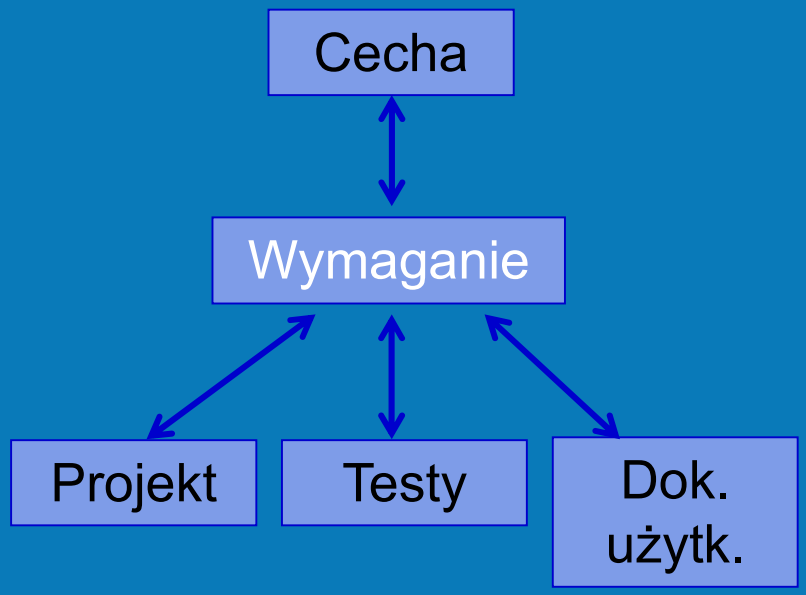
# Zarządzanie wymaganiami

- 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 wymagań 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

W czym tkwi problem i jak sobie z nim poradzić?

# Bariery pozyskiwania wymagań

- Syndrom „tak, ale”
- *„Tak, ale hmmmm, teraz kiedy go już widzę, czy będzie można...? Czy nie lepiej byłoby, gdyby...? Przecież jeżeli..., to trudno będzie...”*
- Syndrom „nieodkrytych ruin” - nigdy nie wiadomo, które z wymagań zostało „nieodkryte”
- 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.

PU1

# 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*)

## Slajd 27

**PU1** egzamin

Piotr Urbanek; 19.01.2017

# *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 informacji o:

- Strukturze organizacyjnej, polityce działania, praktykach stosowanych w pracy
- Frustracjach związanych z wykonywana pracą
- Wymagań specjalnych związanych z oprogramowaniem, sprzętem
- Efektywności programu szkoleniowego
- Stopnia zadowolenia z produktu

# *User instructions* – instruowanie przez użytkowników (1)

W technice tej użytkownicy instruują przeprowadzającego badanie o 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 frustrujące 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)

Mogą być pomocne w pozyskaniu informacji o:

- Wyglądzie interfejsu użytkownika
- Wymaganiach związanych z procesem szkoleniowym
- Kryteriami wydajnościowymi systemu
- Wpływem 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 (1)

- Język naturalny - najczęściej stosowany.

Wady:

- *niejednoznaczność* powodująca różne rozumienie tego samego tekstu;
- *elastyczność*, pozwalająca 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 (2)

- 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.
- 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

Streszczenie
Spis treści
Status dokumentu (roboczy, 1-sza faza, zatwierdzony, ...)
Zmiany w stosunku do wersji poprzedniej

---

Przykładowy spis treści

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

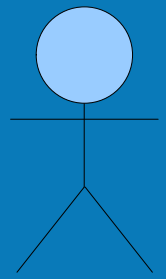
Sposób na sformalizowanie wymagań

# 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

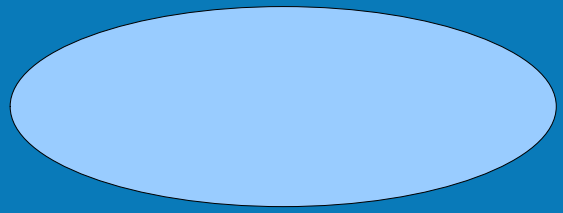
# Podstawowe elementy modelu przypadków użycia

- Przypadki użycia
- Aktorzy
- Specyfikacje przypadków użycia



**aktor**

Reprezentuje rolę, którą może grać w sytemie jakiś jego użytkownik (np. kierownik, urzędnik, klient) lub system zewnętrzny wchodzący w interakcję z modelowanym systemem

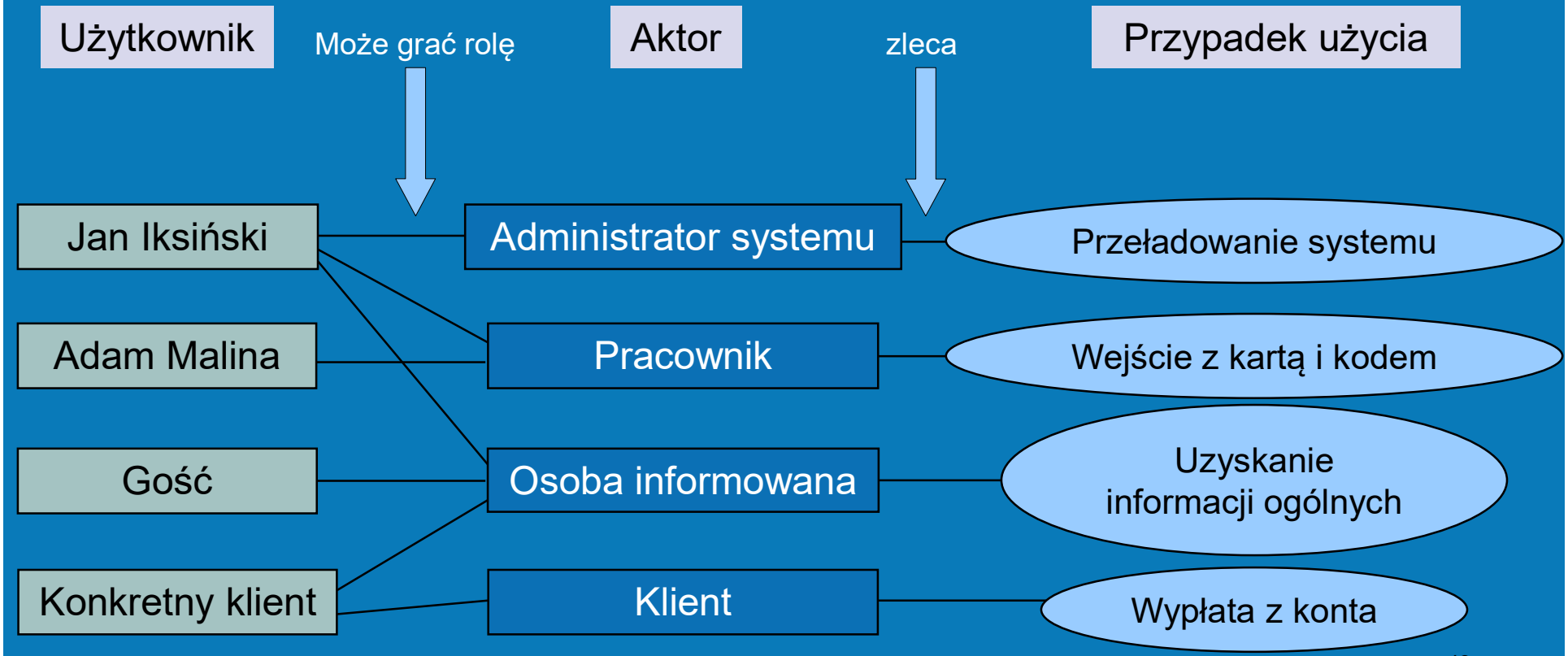


**przypadek użycia**

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

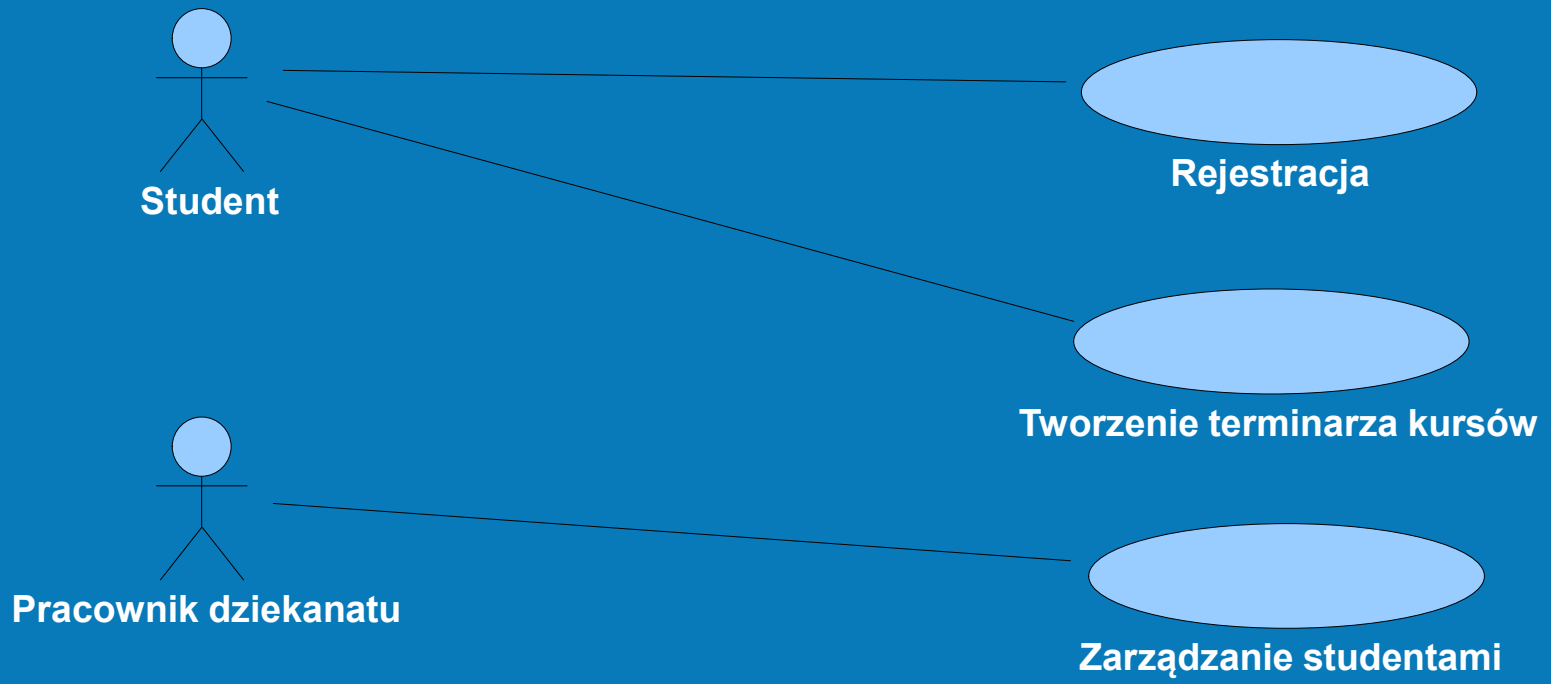
# Aktor - konkretny byt, czy rola?

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



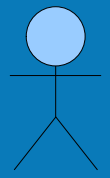
# Diagram przypadków użycia – kontekst systemu

Opisuje funkcje systemu w terminach przypadków użycia.

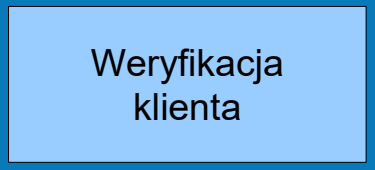


# Notacja

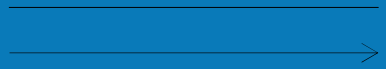

Przypadek użycia




Aktor




Blok ponownego użycia (współdzielony między przypadkami), może być oddzielnym przypadkiem użycia



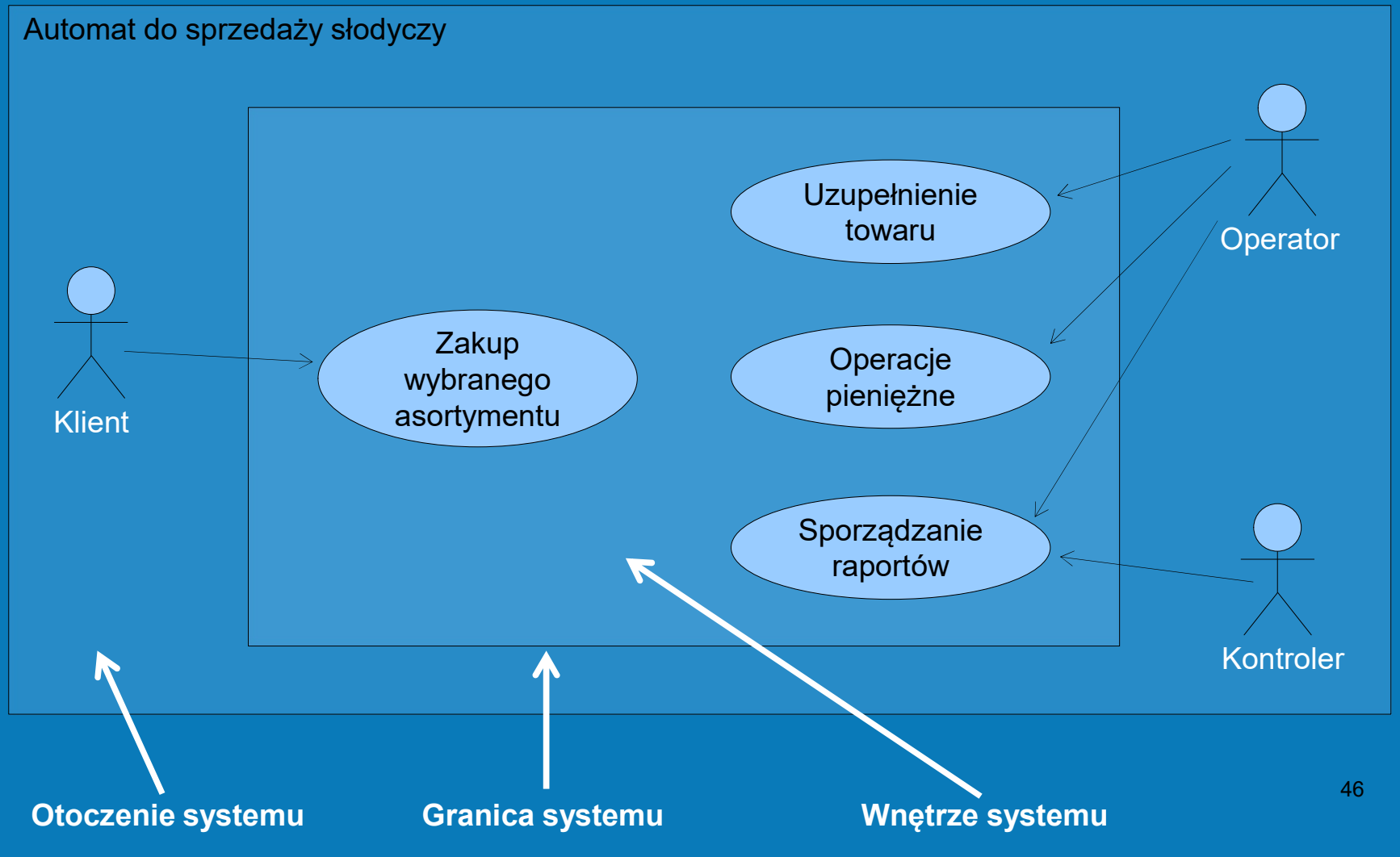
Interakcja

Dziedziczenie (między aktorami lub przypadkami)


Jeden przypadek zawsze włącza drugi


Przypadek użycia czasami rozszerza inny przypadek (przebiegi opcjonalne)

# Diagram przypadków użycia - przykład



# Dokumentacja przypadków użycia

Dokumentacja przypadków użycia powinna zawierać:

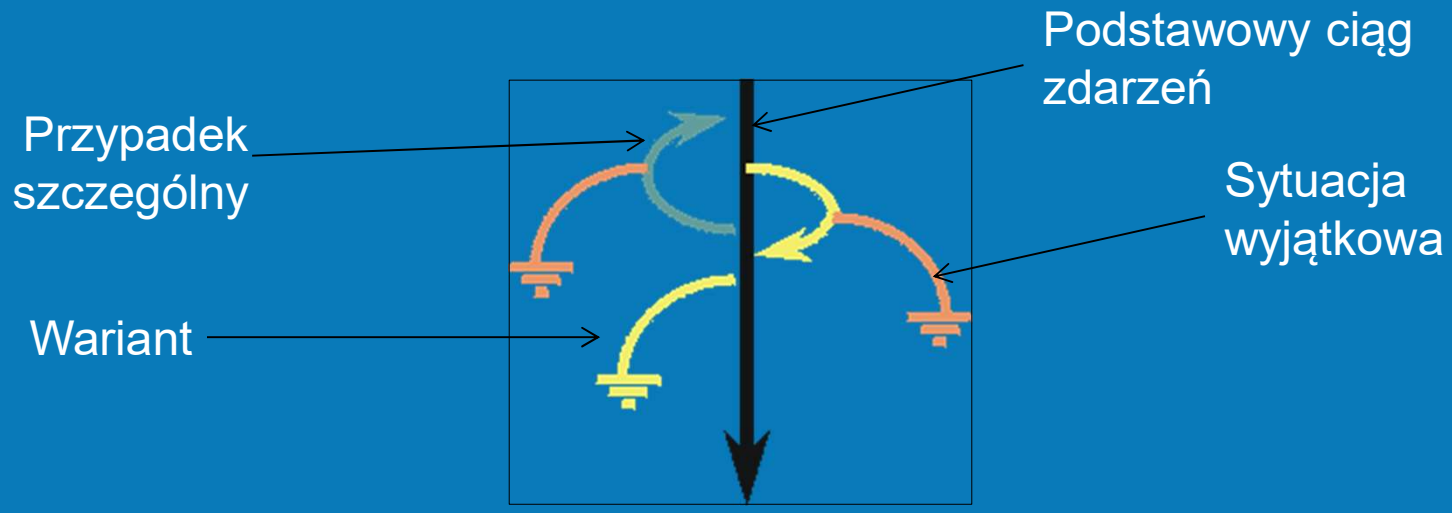
- Diagramy przypadków użycia (aktorzy, przypadki użycia i relacje zachodzące między przypadkami)
- Krótki, ogólny opis każdego przypadku użycia:
  - Jak i kiedy przypadek użycia zaczyna się i kończy
  - Opis interakcji przypadku użycia z aktorami, włączając w to kiedy interakcja ma miejsce i co jest przesyłane
  - Kiedy i do czego przypadek użycia potrzebuje danych zapamiętanych w systemie, lub jak i kiedy zapamiętuje dane w systemie
  - Wyjątki występujące przy obsłudze przypadku
  - Specjalne wymagania, np. czas odpowiedzi, wydajność
  - jak i kiedy używane są pojęcia dziedziny problemowej
- Opis szczegółowy każdego przypadku użycia:
  - Scenariusze
  - Diagram aktywności

# Ciąg zdarzeń (przepływ sterowania) dla przypadku użycia

- Zawiera najważniejsze informacje, będące wynikiem prac nad opracowaniem przypadku użycia
- Powinien opisywać przepływ zdarzeń w przypadku użycia, w taki sposób, aby każdy mógł łatwo zrozumieć istotę działania

# Ciągi zdarzeń (sterowania) dla przypadku użycia - scenariusze

- Podstawowy ciąg zdarzeń
- Alternatywne ciągi zdarzeń
  - Zwykłe warianty
  - Przypadki szczególne
- Sytuacje wyjątkowe



# Ciągi zdarzeń - przykład

**Przypadek użycia:** Wybieranie metody wypłaty

**Podstawowy ciąg zdarzeń:**

Przypadek użycia rozpoczyna się w momencie, gdy Pracownik zgłasza do systemu chęć wyboru metody wypłaty.

1. System prosi Pracownika o wybranie metody wypłaty (w kasie, droga pocztowa lub przelewem).
2. Pracownik wybiera metodę wypłaty.
    1. Jeżeli Pracownik wybrał wypłatę w kasie, żadne dodatkowe informacje nie są potrzebne.
    1. Jeżeli Pracownik wybrał wypłatę drogą pocztową, system prosi użytkownika o podanie adresu, na jaki mają być przesyłane pieniądze.
    2. Jeżeli Pracownik wybrał wypłatę w postaci przelewu bankowego, system prosi o podanie nazwy banku i numeru konta.
3. Po podaniu przez Pracownika wymaganych informacji system uaktualnia informacje o Pracowniku.

# Scenariusze

Liczba scenariuszy jest zawsze znacznie większa niż liczba przypadków użycia. Średnio skomplikowany system ma ok. kilkudziesięciu przypadków użycia, z których każdy rozwija się nawet do kilkudziesięciu scenariuszy.

**Przypadek użycia: *Wybieranie metody wypłaty***

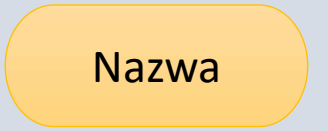
Warianty:

- Wybranie wypłaty w kasie
- Wybranie wypłaty drogą pocztową
- Wybranie wypłaty przelewem
- Pracownik nie znaleziony

# Modelowanie zachowania - Diagramy aktywności

W modelu przypadków użycia graficznie obrazuje działania wykonywane w danym przypadku użycia.

Jest to diagram przepływu, który można wykorzystywać do obrazowania przepływu pracy (*workflow*), sterowania, zdarzeń, itp.




Aktywność

Blok decyzyjny

Przejście

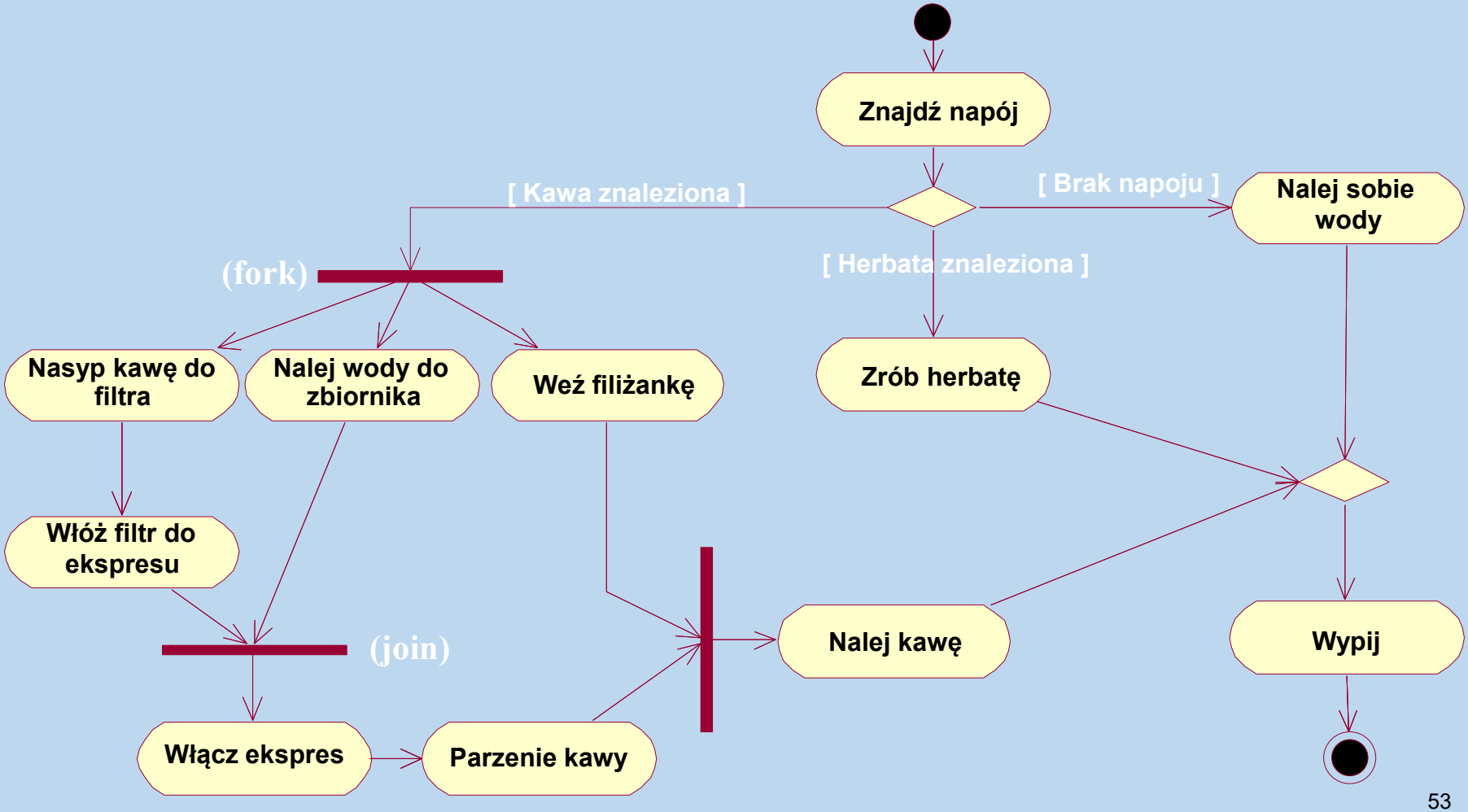
Stan początkowy

Stan końcowy

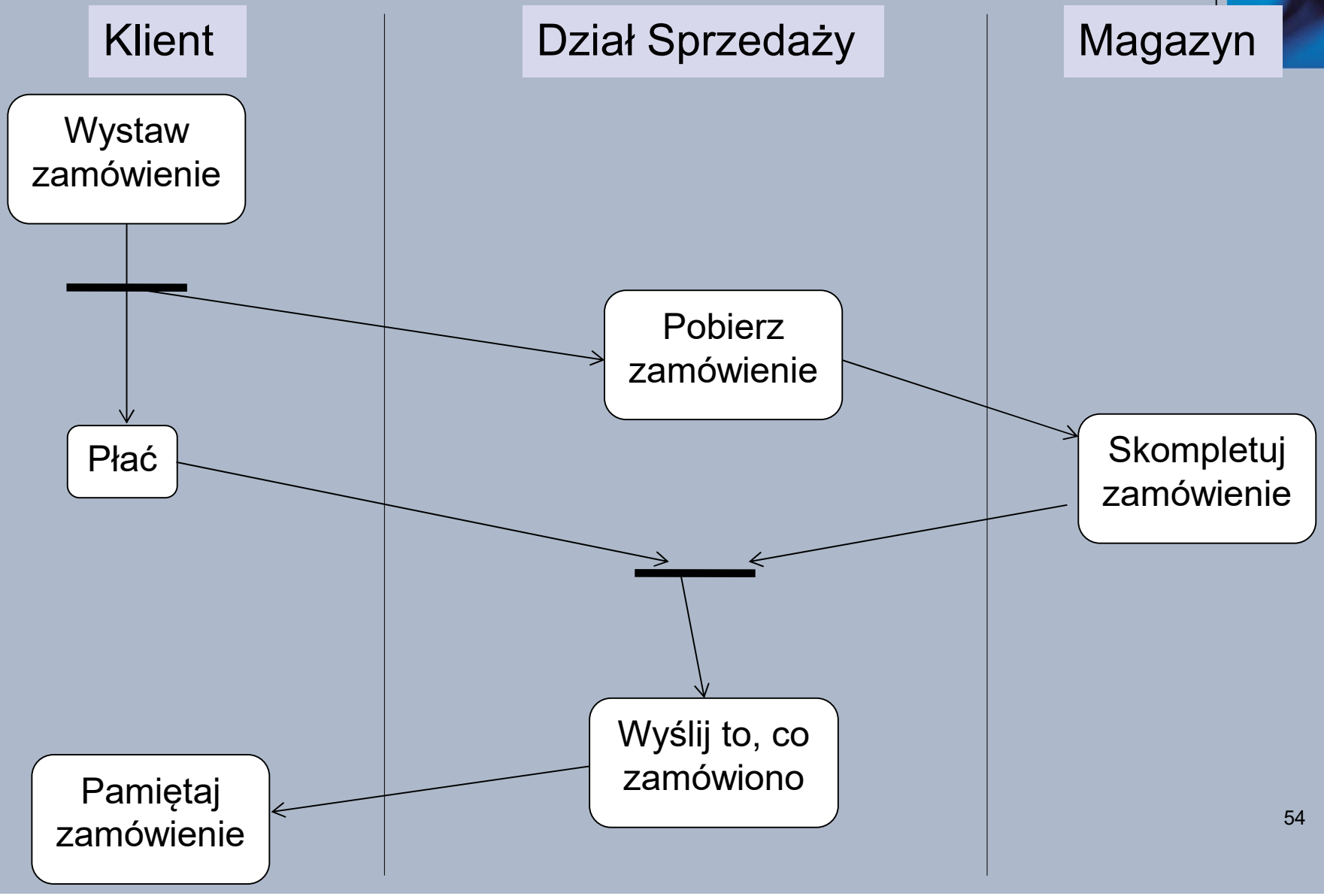


Sztabka synchronizująca

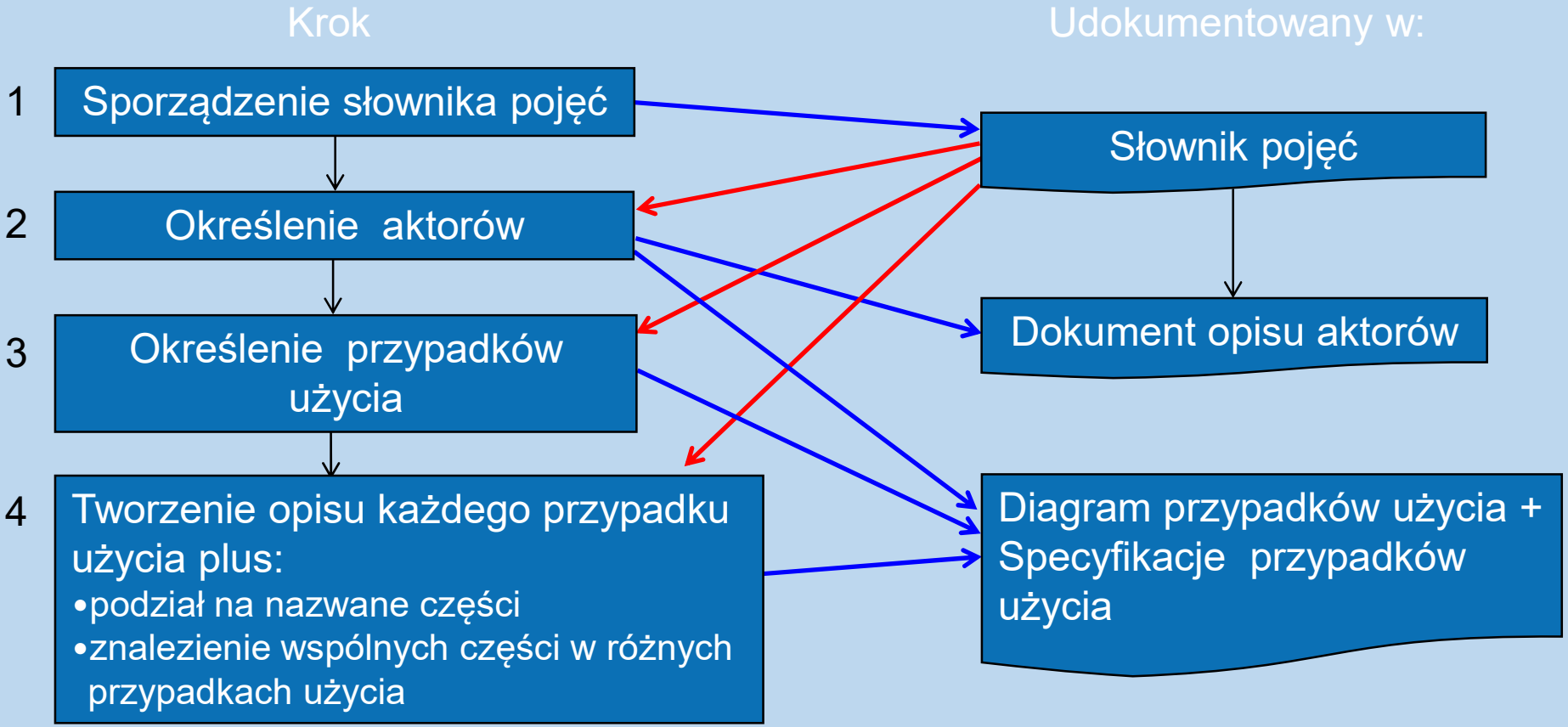
# Diagram aktywności - przykład



# Partycjonowanie - *Swimlanes*



# Kroki konstrukcji modelu wymagań



# Sporządzanie słownika pojęć

- Słownik dotyczy dziedziny problemowej
- Tworzenie go polega na wyłowieniu wszystkich terminów z wymagań użytkownika.
- Terminy mogą odnosić się do aktorów, przypadków użycia, obiektów, operacji,  zdarzeń, itp.
- Terminy w słowniku powinny być zdefiniowane w sposób precyzyjny i jednoznaczny.
- Posługiwanie się terminami ze słownika powinny być regułą  przy opisie każdego kolejnego problemu, sytuacji czy modelu.

# Identyfikacja aktorów

Aby poprawnie określić aktorów należy odpowiedzieć na następujące pytania:

- Jaka grupa użytkowników potrzebuje wspomagania ze strony systemu (np. osoba wysyłająca korespondencję)?
- Jacy użytkownicy są konieczni do tego, aby system działał i wykonywał swoje funkcje (np. administrator systemu)?
- Z jakich elementów zewnętrznych (innych systemów, komputerów, czujników, sieci, itp.) musi korzystać system, aby realizować swoje funkcje.
- Po wyszukaniu aktorów, należy ustalić:
- Nazwę dla każdego aktora/roli,
- Zakresy znaczeniowe dla poszczególnych nazw aktorów oraz relacje pomiędzy zakresami (np. sekretarka, pracownik administracji, pracownik, dowolna osoba).
- Niekiedy warto ustalić hierarchię dziedziczenia dostępu do funkcji systemu dla aktorów.

## Identyfikacja przypadków użycia – porady (1)

- Dla każdego aktora, znajdź funkcje (zadania), które powinien wykonywać w związku z jego działalnością w zakresie zarówno dziedziny przedmiotowej, jak i wspomagania działalności systemu informacyjnego.
- Staraj się powiązać w jeden przypadek użycia zespół funkcji realizujących podobne cele. Unikaj rozbicia jednego przypadku użycia na zbyt wiele podprzypadków.
- Nazwy dla przypadków użycia: powinny być krótkie, ale jednoznacznie określające charakter zadania lub funkcji. Nazwy powinny odzwierciedlać czynności z punktu widzenia aktorów, a nie systemu, np. "*wpłacanie pieniędzy*", a nie "*przyjęcie pieniędzy od klienta*".
- Opisz przypadki użycia przy pomocy zdań w języku naturalnym, używając terminów ze słownika pojęć.

# Identyfikacja przypadków użycia porady (2)

- Uporządkuj aktorów i przypadki użycia w postaci diagramu przypadków użycia.
- Niektóre z powstałych w ten sposób przypadków użycia mogą być mutacjami lub szczególnymi przypadkami innych przypadków użycia. Przeanalizuj powiązania aktorów z przypadkami użycia i ustal, które z nich są zbędne lub mogą być uogólnione.
- Wyodrębnij “przypadki bazowe”, czyli te, które stanowią istotę zadań, są normalnym, standardowym użyciem. Pomiń czynności skrajne, wyjątkowe, uzupełniające lub opcjonalne.
- Nazwij te “przypadki bazowe”. Ustal powiązania “przypadków bazowych” z innymi przypadkami, poprzez ustalenie ich wzajemnej zależności: sekwencji czy alternatywy.

# Identyfikacja przypadków użycia porady (3)

- Dodaj zachowania skrajne, wyjątkowe, uzupełniające lub opcjonalne. Ustal powiązanie "przypadków bazowych" z tego rodzaju zachowaniem. Może ono być także powiązane w pewną strukturę.
- Staraj się, aby bloki specyfikowane wewnątrz każdego przypadku użycia nie były zbyt ogólne lub zbyt szczegółowe. Zbyt szczegółowe bloki utrudniają analizę. Zbyt ogólne bloki zmniejszają możliwość wykrycia bloków ponownego użycia. Struktura nie może być zbyt duża i złożona.
- Staraj się wyizolować bloki ponownego użycia. Przeanalizuj podobieństwo nazw przypadków użycia, podobieństwo nazw i zachowania bloków ponownego użycia występujących w ich specyfikacji. Wydzielenie bloku ponownego użycia może być powiązane z określeniem bardziej ogólnej funkcji lub dodaniu nowej specjalizacji do istniejącej funkcji.

# Specyfikacja przypadku użycia

- Nazwa
- Krótki opis
- Opis ciągów zdarzeń (przepływu sterowania)
- Powiązania z innymi przypadkami użycia
- Diagramy aktywności (ew. stanu)
- Diagramy przypadków użycia
- Wymagania specjalne
- Warunki wejściowe (ang. pre-conditions)
- Warunki wyjściowe (ang. post-conditions)
- Inne diagramy

# Podsumowanie

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

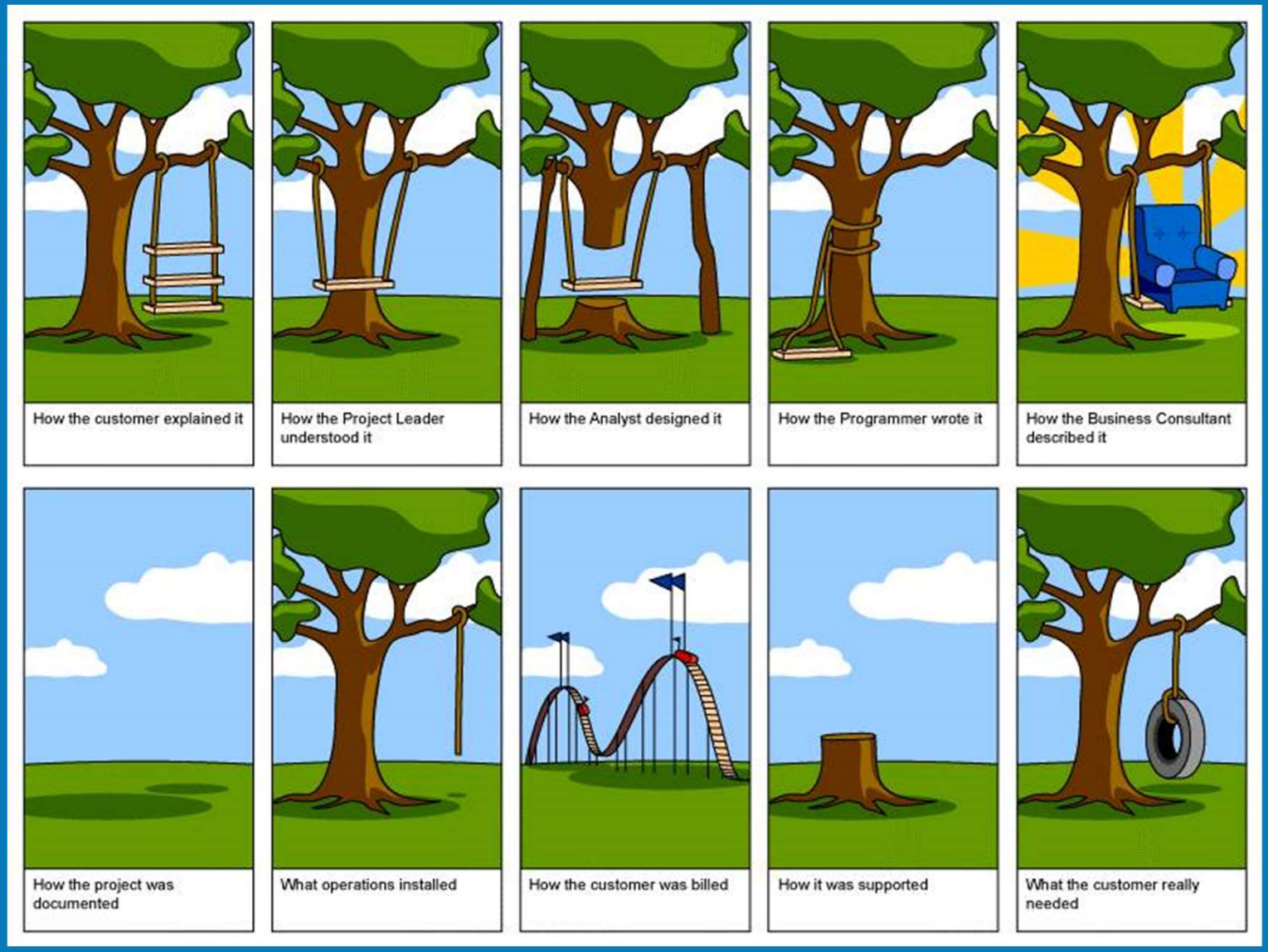
# Do poczytania

- Dąbrowski, Subieta*: Podstawy Inżynierii Oprogramowania,* rozdział 3.
- Wiecej 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

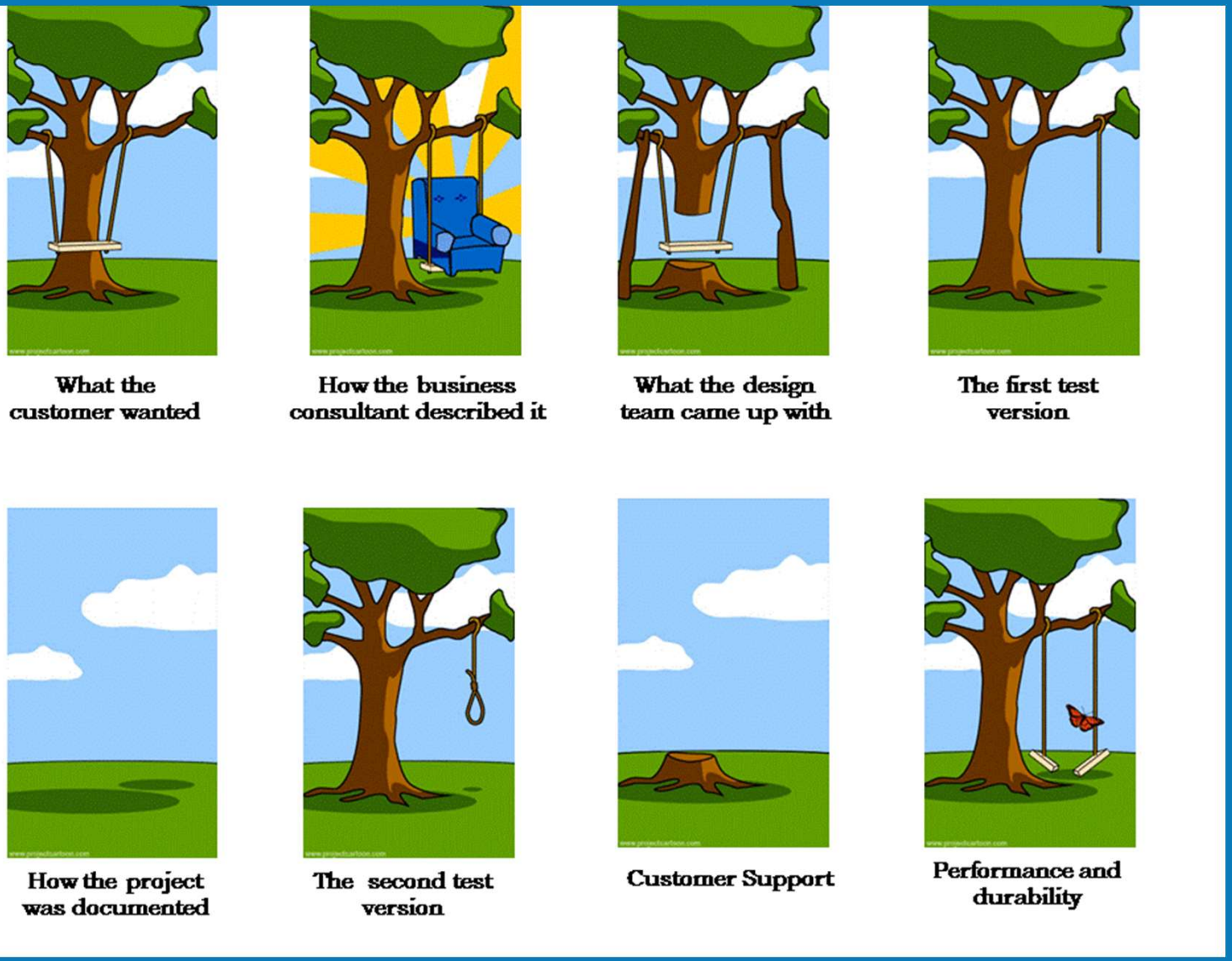
# Dokumenty/Narzędzia

- 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.

# Tree swing


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

# Tree swing

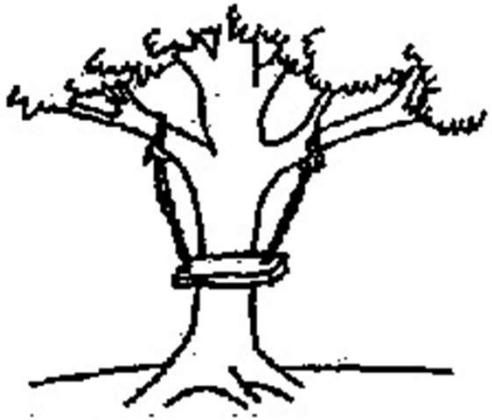

What the customer wanted
How the business consultant described it
What the design team came up with
The first test version
How the project was documented
The second test version
Customer Support
Performance and durability

# Tree swing

As proposed by the project sponsor.

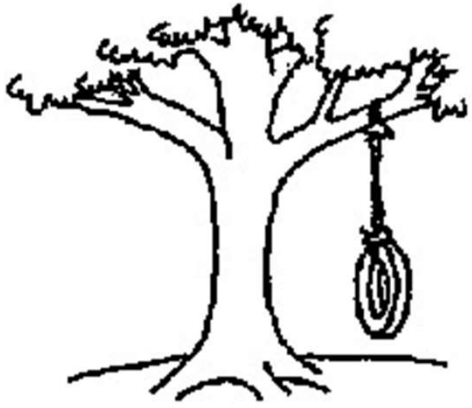
As specified in the project request.

As designed by the senior analyst.



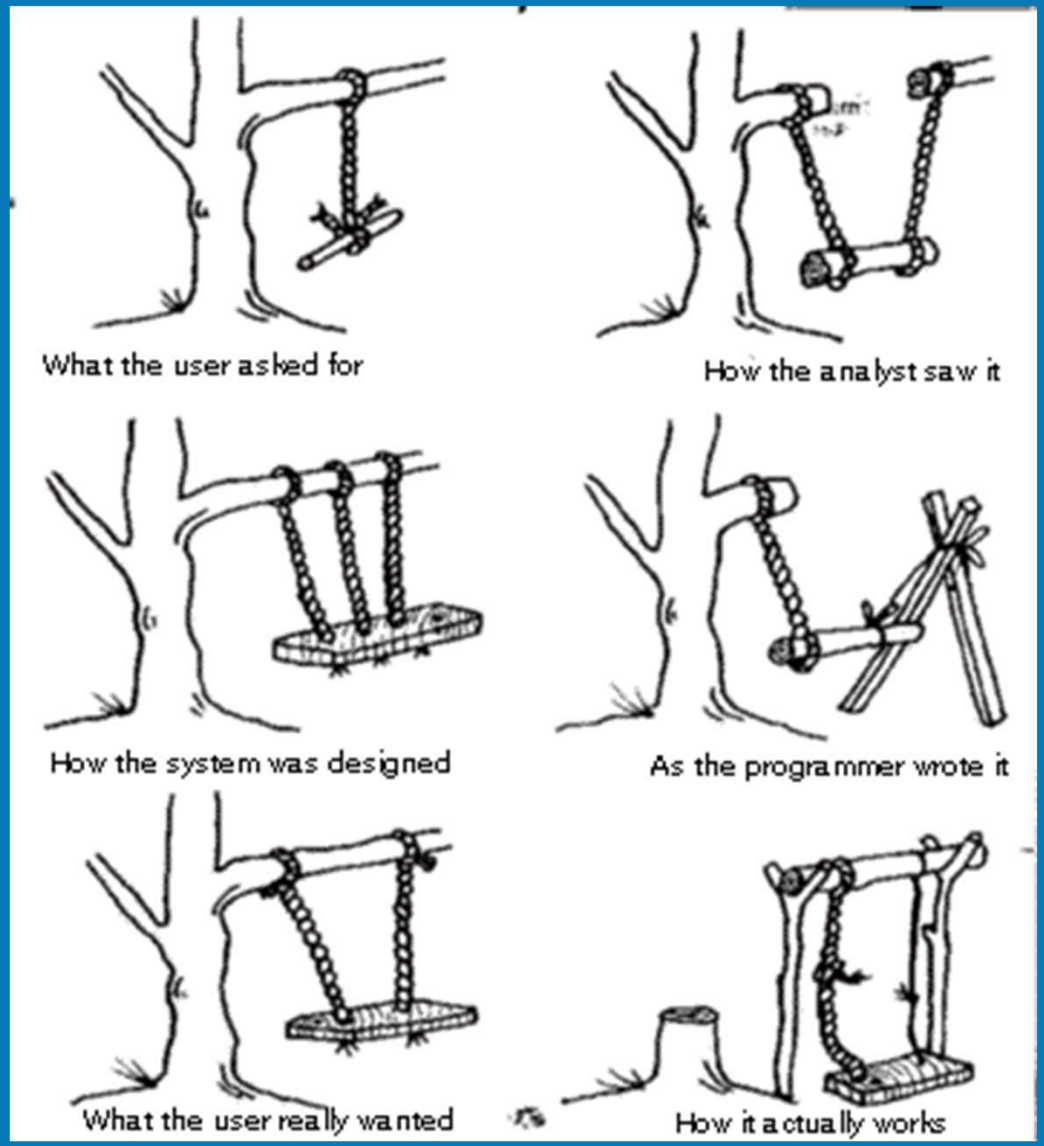
As produced by the programmers.

As installed at the user's site.



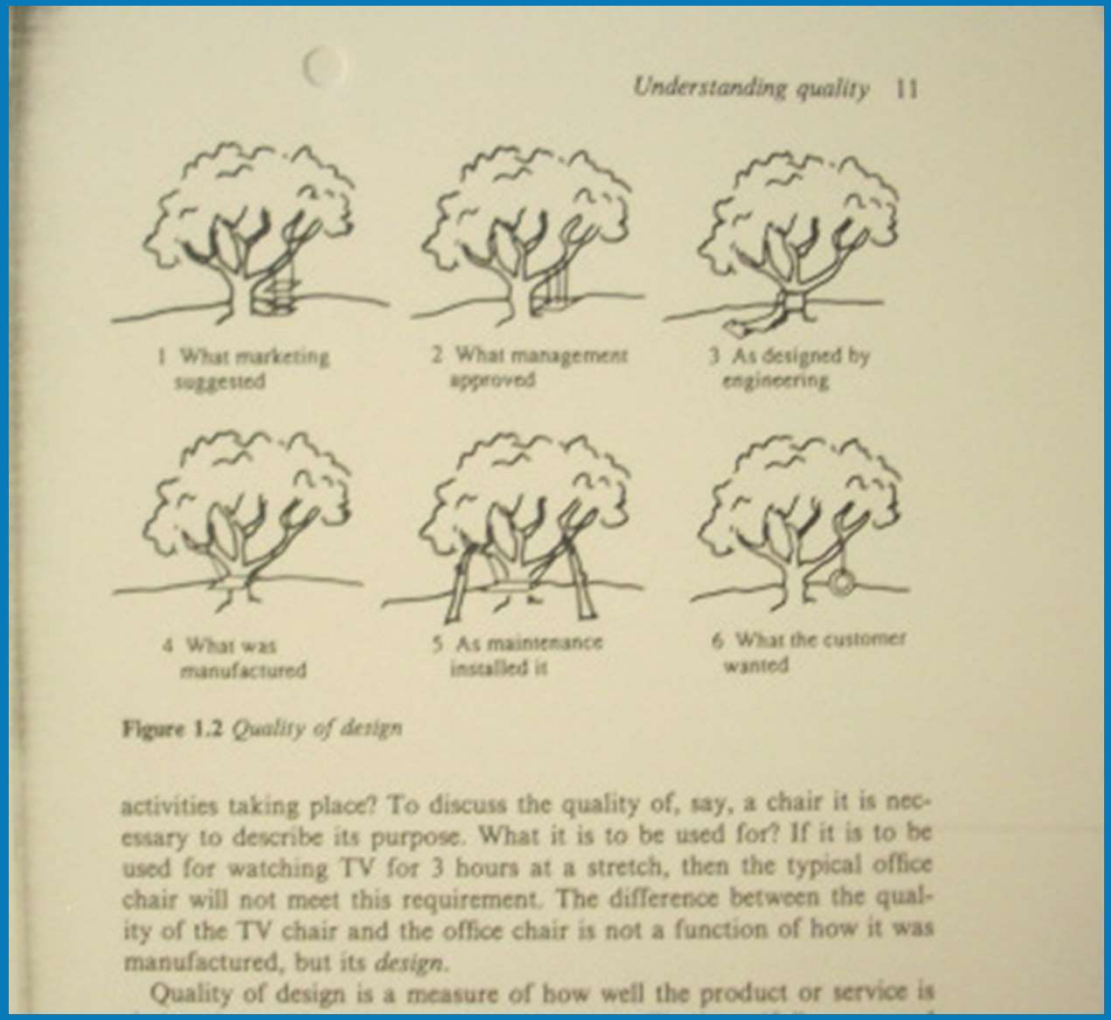
What the user wanted.

# Tree swing


What the user asked for
How the analyst saw it
How the system was designed
As the programmer wrote it
What the user really wanted
How it actually works

# *Tree swing* – historia

***Total Quality Management*, J. Oakland, 1989:**



*Understanding quality* 11

1 What marketing suggested

2 What management approved

3 As designed by engineering

4 What was manufactured

5 As maintenance installed it

6 What the customer wanted

**Figure 1.2** *Quality of design*

activities taking place? To discuss the quality of, say, a chair it is necessary to describe its purpose. What it is to be used for? If it is to be used for watching TV for 3 hours at a stretch, then the typical office chair will not meet this requirement. The difference between the quality of the TV chair and the office chair is not a function of how it was manufactured, but its *design*.

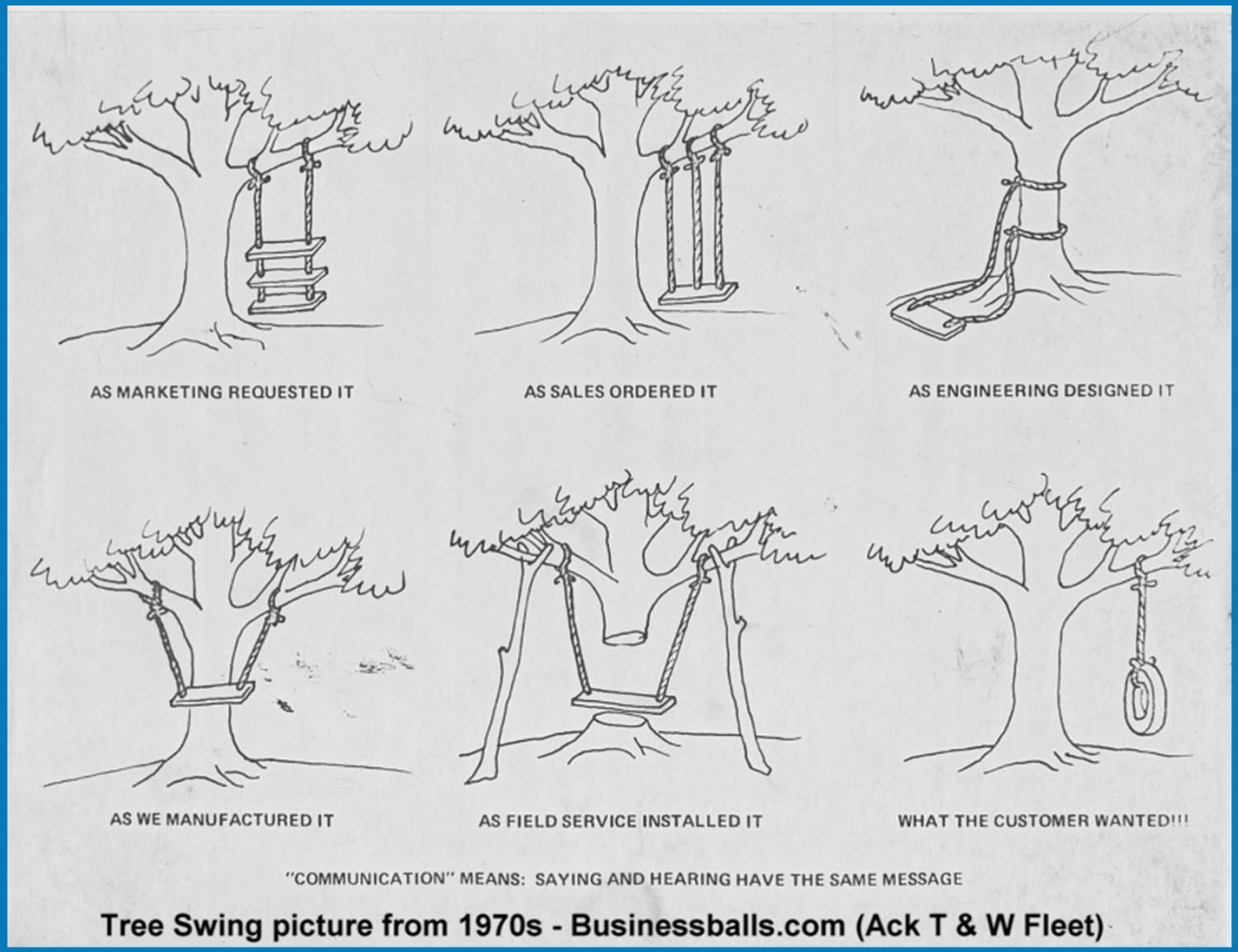
Quality of design is a measure of how well the product or service is



**Źródło: http://www.businessballs.com**

# *Tree swing* – historia

**Lata siedemdziesiąte:**

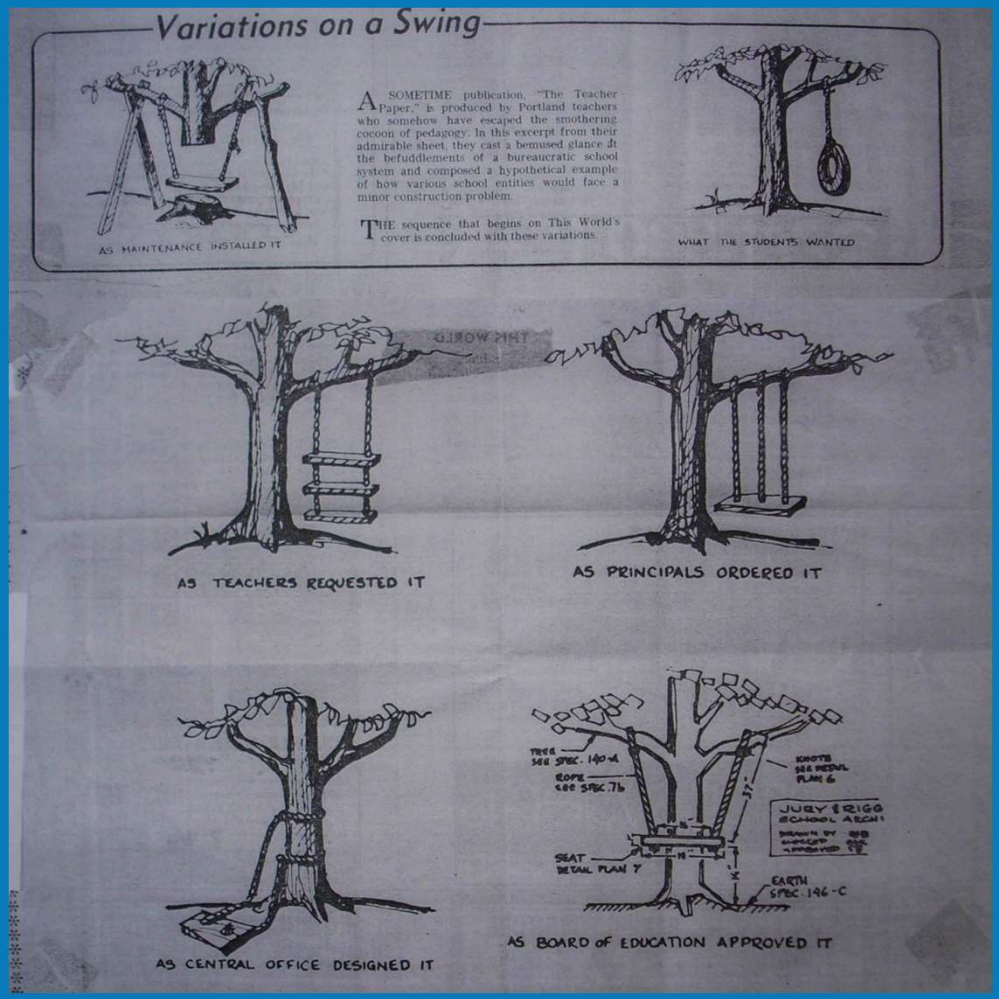





**Źródło: http://www.businessballs.com**

# *Tree swing* – historia

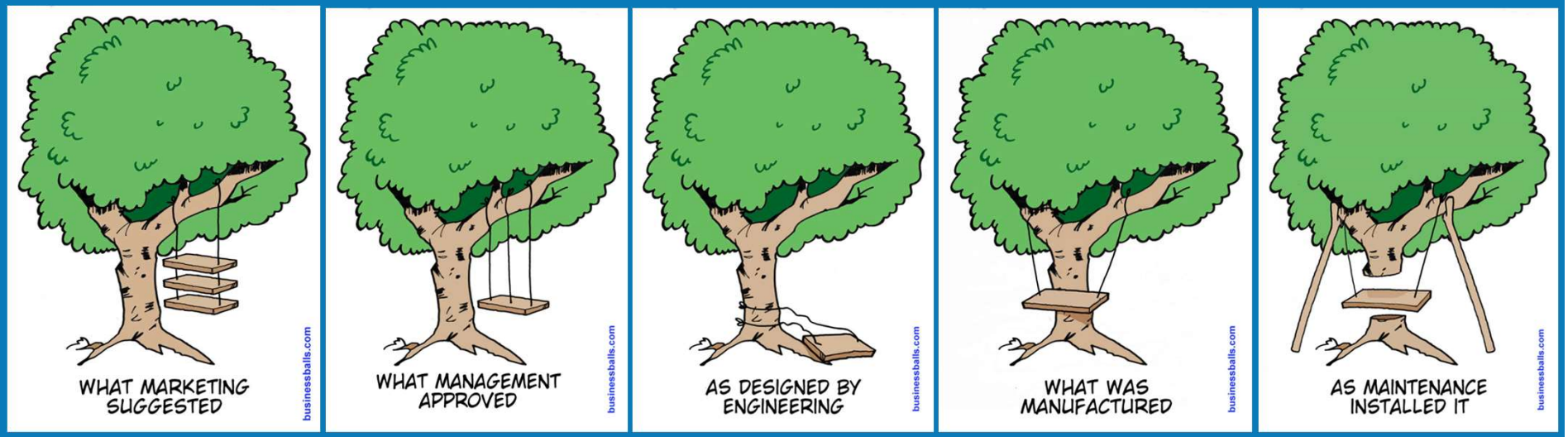
**Lata siedemdziesiąte:**






**Źródło: http://www.businessballs.com**

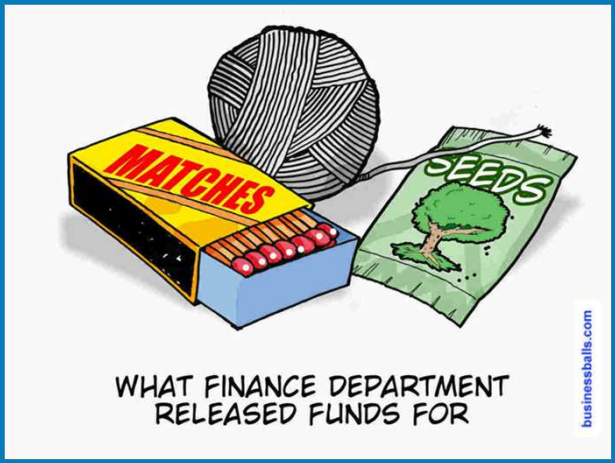
# *Tree swing* – nowe wariacje



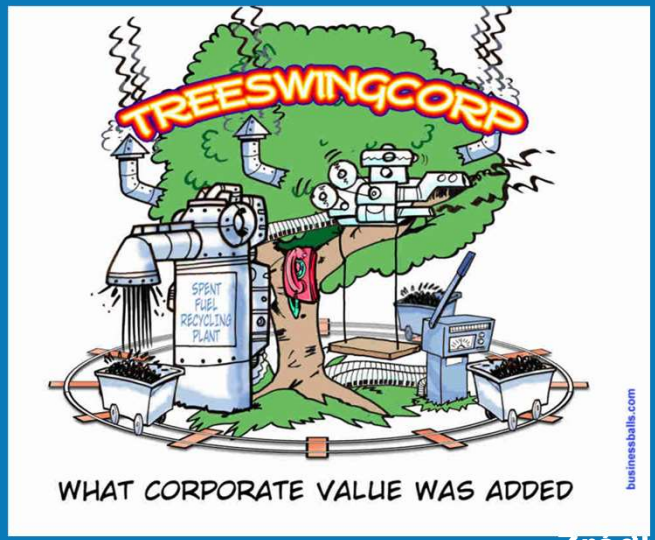



Źródło: http://www.businessballs.com

# *Tree swing* – nowe wariacje



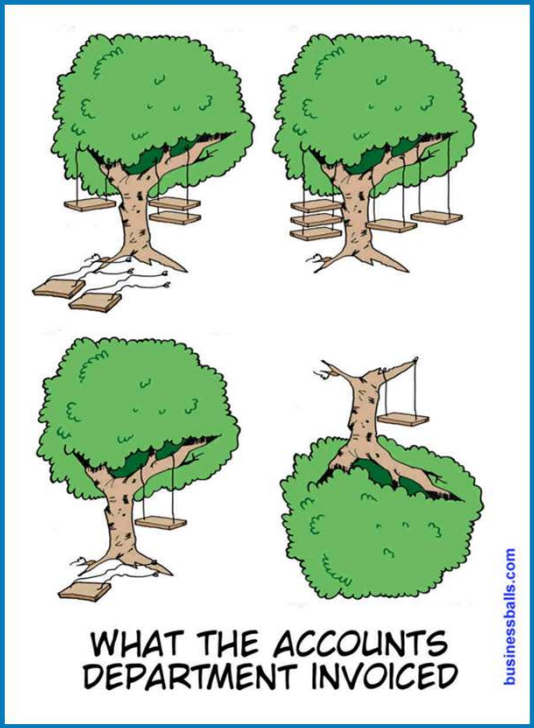


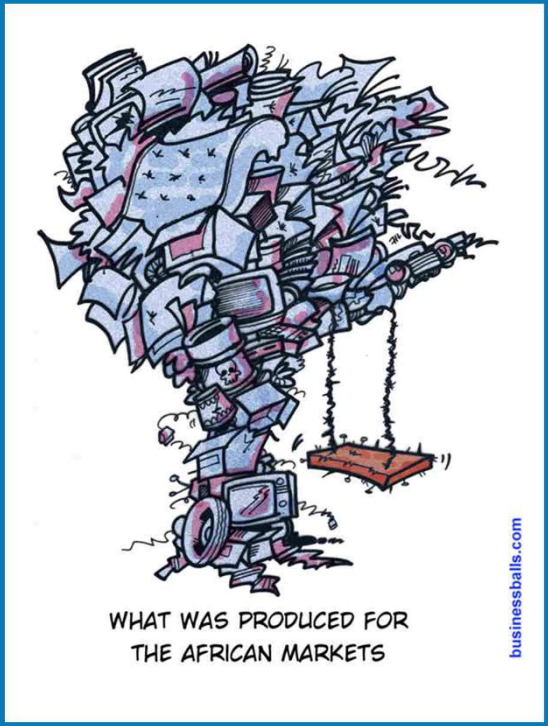



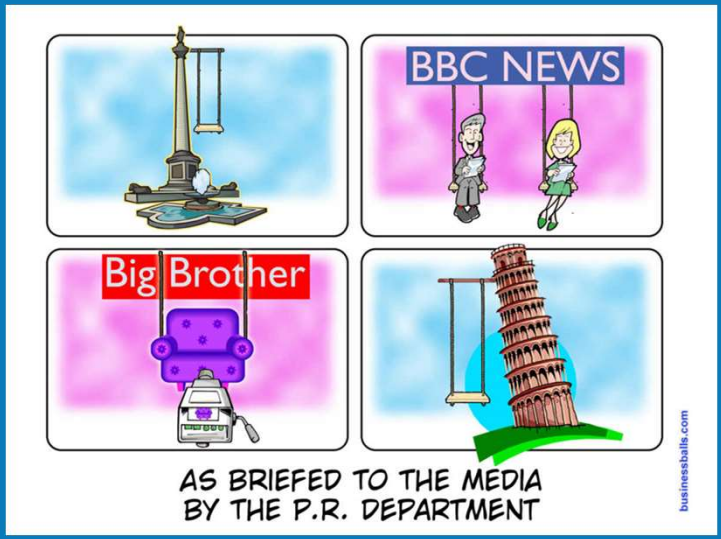



Źródło: http://www.businessballs.com

# *Tree swing* – nowe wariacje



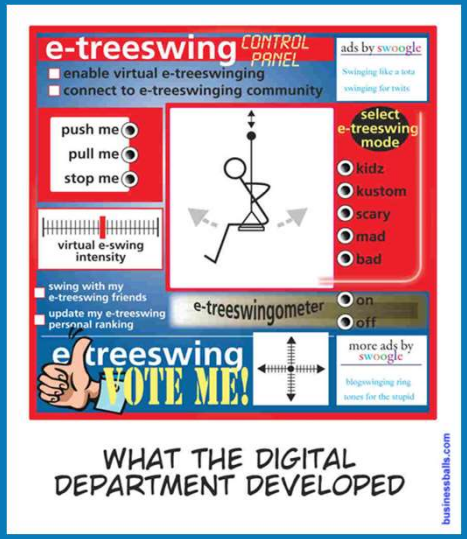






Źródło: http://www.businessballs.com

# Tree swing




WHAT THE DIGITAL DEPARTMENT DEVELOPED

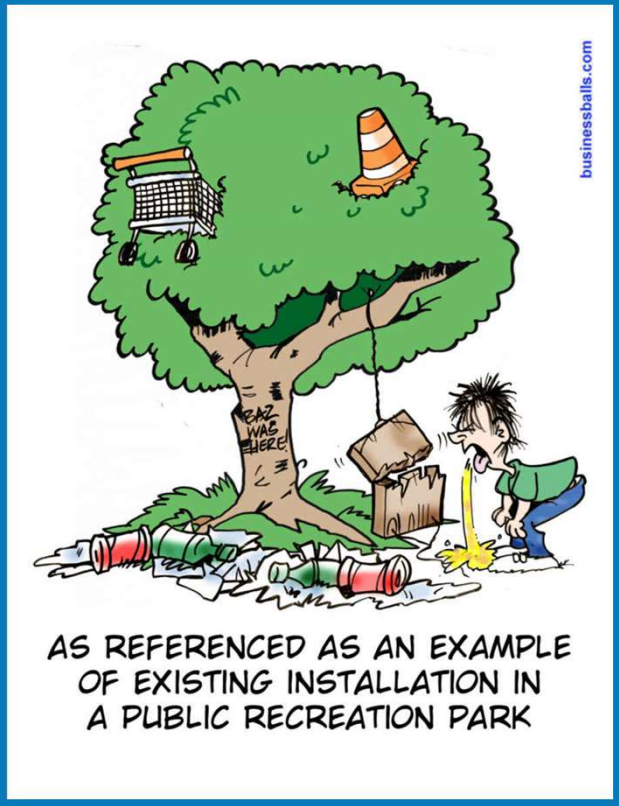

AS PRESENTED BY THE POLITICAL LOBBYIST TO MINISTERS RESPONSIBLE FOR PLANNING APPROVAL

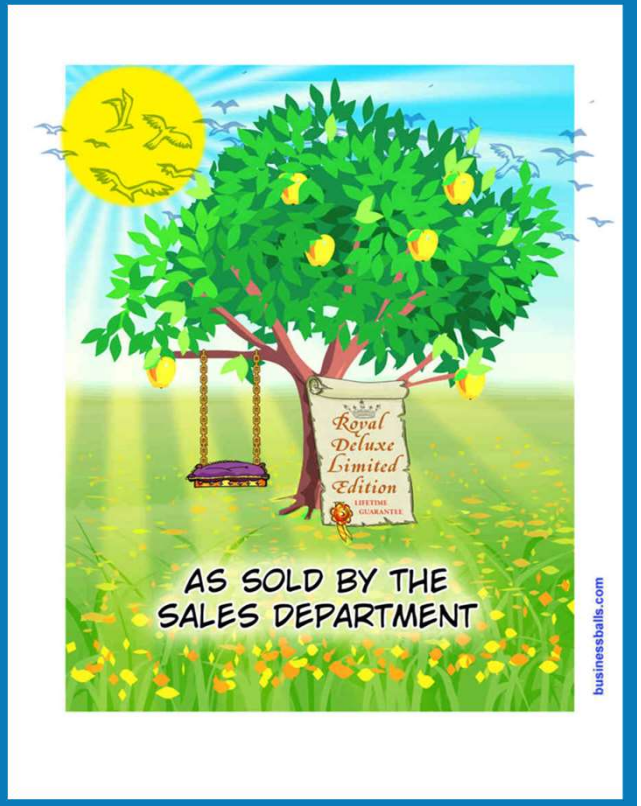

WHAT THE CHAIRMAN ANNOUNCED TO SHAREHOLDERS



Źródło: http://www.businessballs.com

# *Tree swing* – nowe wariacje




Źródło: http://www.businessballs.com

# *Tree swing* – zawsze aktualne




Źródło: http://www.businessballs.com