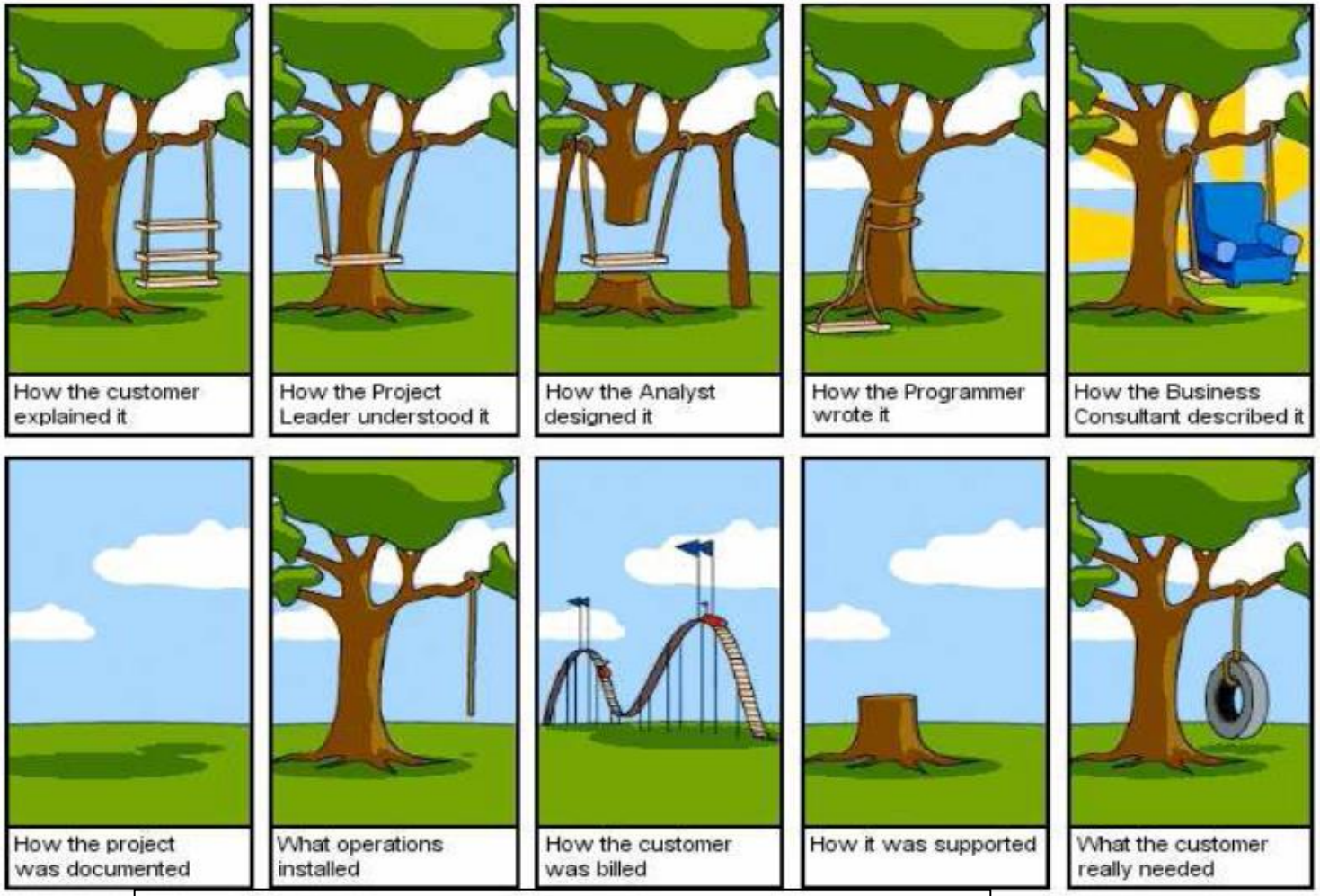
Wykład 2 – Inżynieria wymagań cz.1

1. Wymagania na oprogramowanie – definicja, kategorie, podejścia do specyfikacji;

2. Analiza i specyfikacja wymagań:
   - historyjki użytkownika
3. Przykłady

Problem: **jakie są rzeczywiste wymagania klienta**



Wymaganie - definicja

Wymaganie (IEEE 610):

- **warunek lub własność, której potrzebuje użytkownik** do rozwiązania problemu lub osiągnięcia celu,
- **warunek lub własność, którą musi posiadać system** lub jego komponent, aby spełnić kontrakt, standard, specyfikację lub inny, formalnie wyrażony, dokument,

Pożądana cecha, własność lub zachowanie SYSTEMU oczekiwane przez jego użytkownika

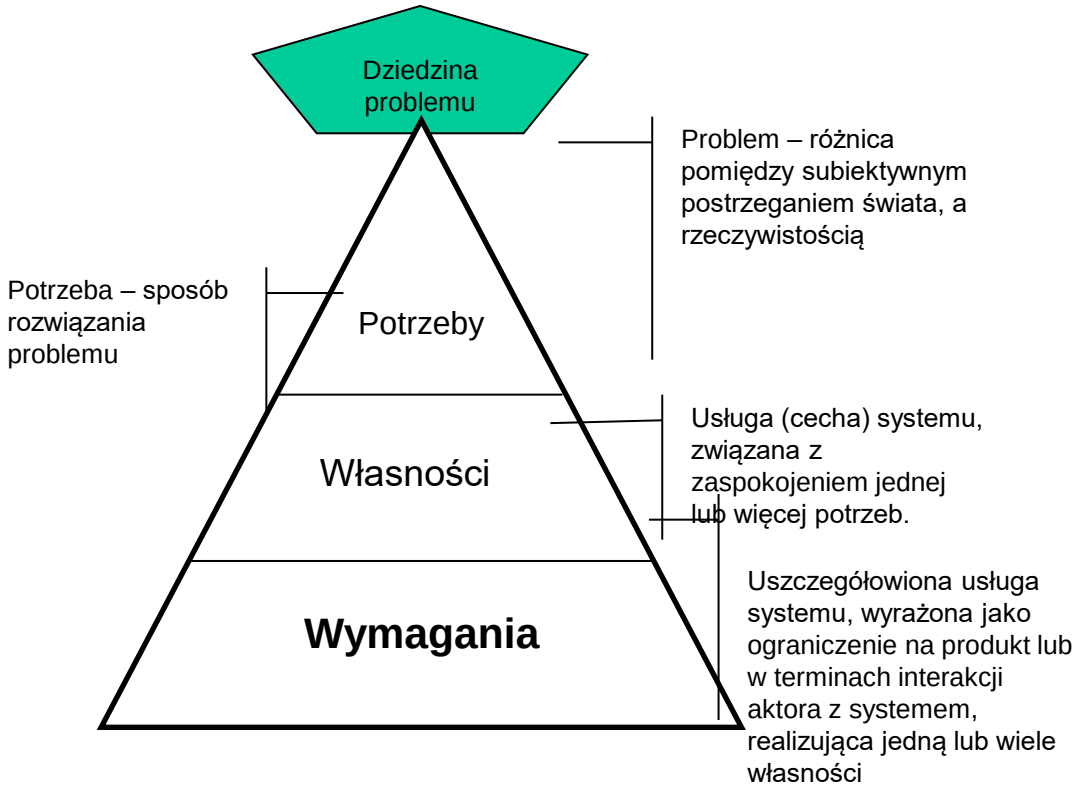
Inżynieria wymagań

DEFINICJA:
systematyczny proces opracowywania wymagań realizowany przez iteracyjną współpracę między procesami:

- analizy problemu,
- dokumentowania końcowych obserwacji w postaci różnorodnych formatów prezentacji oraz
- sprawdzania dokładności uzyskanych informacji

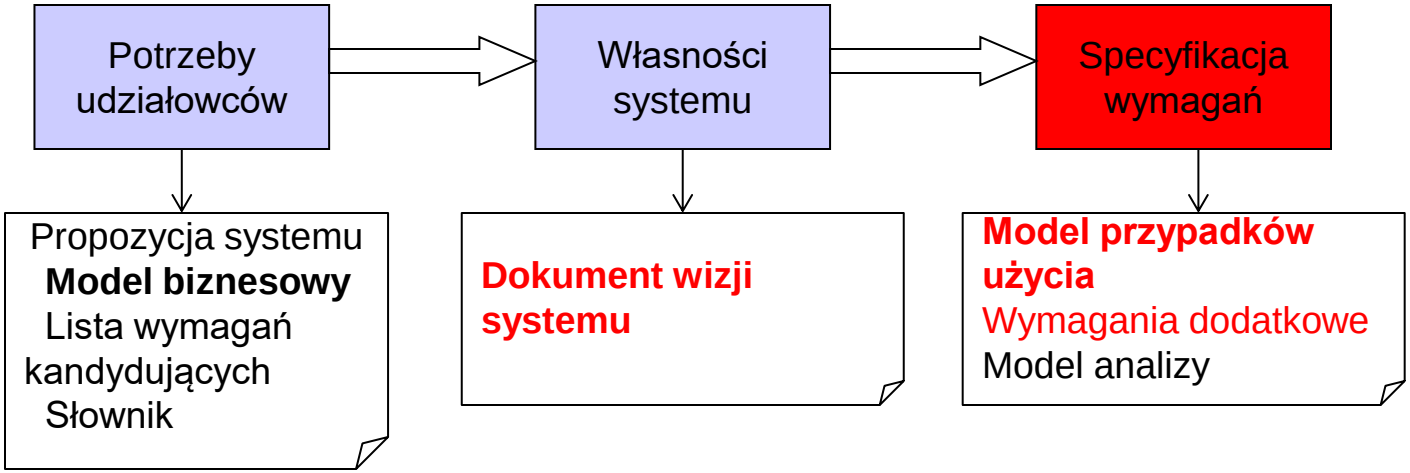
Inżynieria wymagań jest działalnością, która **przekształca potrzeby i życzenia (zwykle niekompletne i wyrażone nieformalnie) klienta i potencjalnego użytkownika systemu komputerowego w kompletną precyzyjną i spójną specyfikację,** i o ile to możliwe, zapisaną w formalnej notacji.

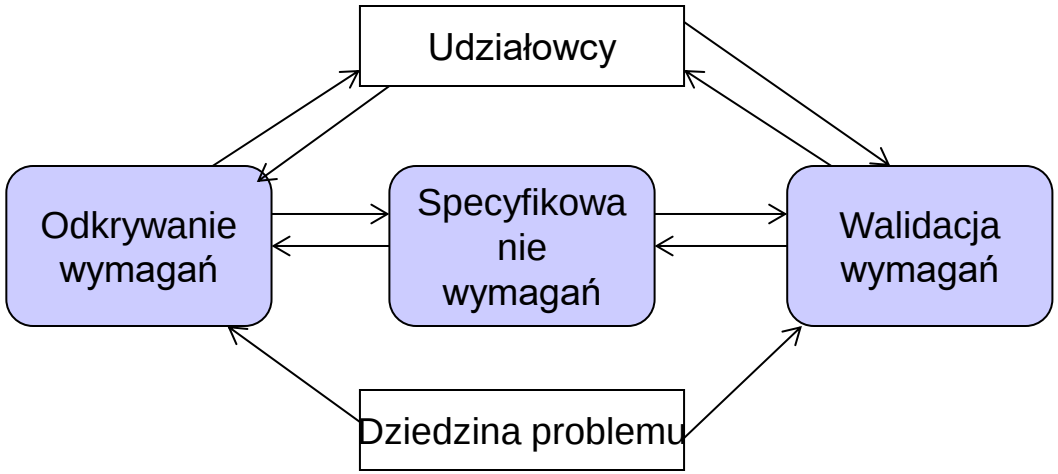
Piramida wymagań



[Cockburn, 2003]

Inżynieria wymagań – procesy i artefakty





Kategorie/perspektywa WYMAGAŃ

Perspektywa wymagań:

- klienta:

  podane/pochodzące od klienta (? kto to)

- wytwórcy:

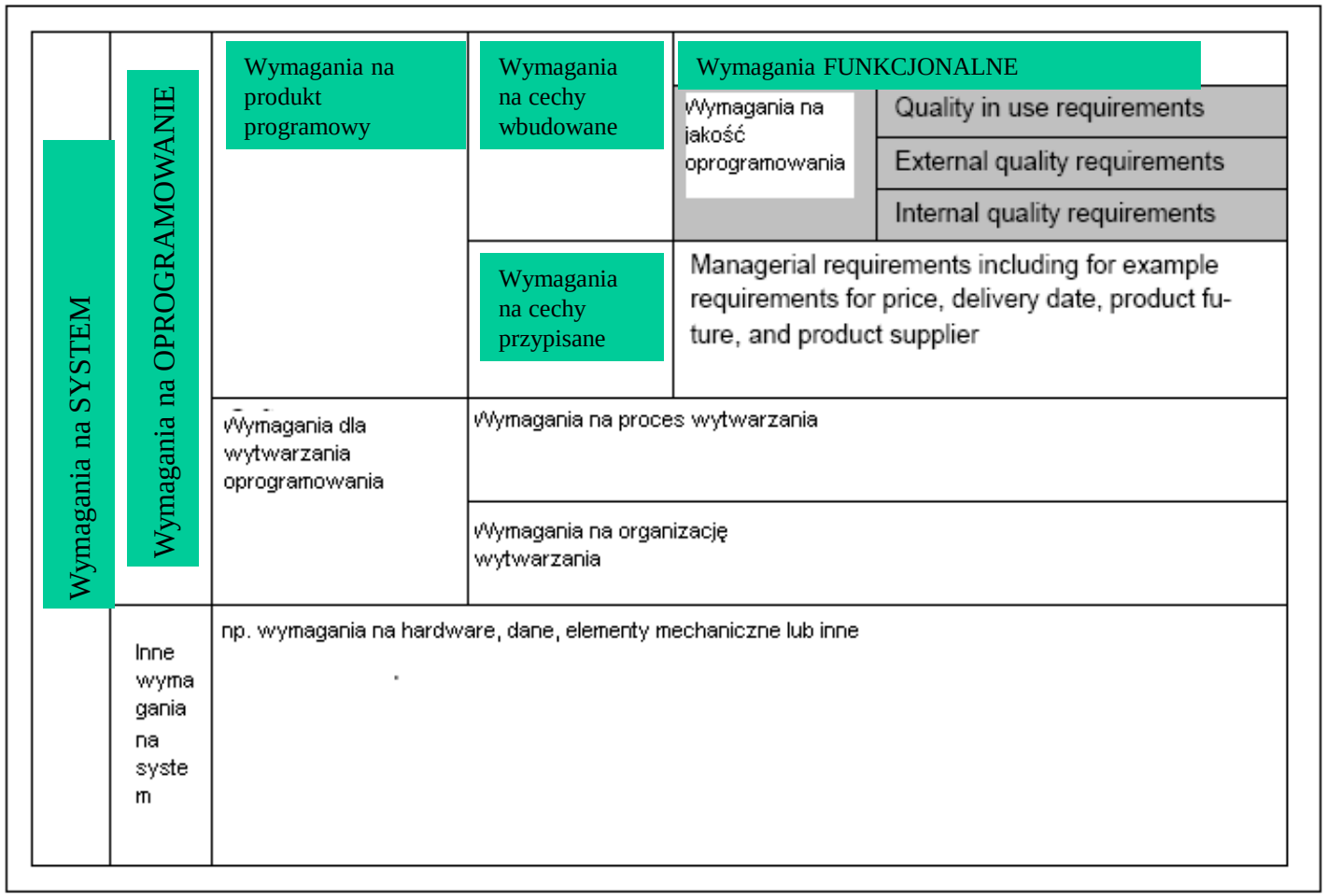
  -kreowane przez zespół tworzący

  -oparte o wymagania Klienta

Klasyfikacja wymagań:

- dotyczące produktu: **funkcjonalne, niefunkcjonalne**
- dotyczące procesu

Klasyfikacja wymagań



[ISO/IEC 25030:06]

Wymagania funkcjonalne dla produktu

Określają **Funkcje (= Usługi),**

- które produkt programowy powinien oferować jego użytkownikom/udziałowcom, i

- które będą realizowane:

  - właściwie (odpowiedni zbiór funkcji do wykonania zadań),
  - z odpowiednią precyzją (dostarczać wyniki),
  - we współpracy (z jednym/wieloma systemami).

Wymagania niefunkcjonalne dla produktu

Rodzaj wymagania	Przykład
Wydajność (efficiency)	System wielodostępny do 20 użytkowników ze średnim czasem odpowiedzi poniżej 0.2 s
Zabezpieczenia (security)	System dokonuje autoryzacji użytkowników
Skalowalność (scalability)	Początkowa wersja systemu ma obsłużyć 50 użytkowników; docelowo 150 użytkowników
Rozszerzalność (extensibility)	Aplikacja jest przystosowana do poszerzania funkcjonalności za pomocą mech. „plug-in”
Kompatybilność (compatibility)	Program ma działać w systemach Win9x, Win2000, WinNT, XP z minimum 16 MB Ram oraz 200 MB dysku
Użyteczność (usability)	Aplikacja zawiera moduł „Pomoc on-line”, który wspomaga 90% opcji
Współpraca (cooperation - z urządzeniami, systemami)	Program ma importować dane z systemu XXX

Kategorie wymagań FURPS

***Functional** requirements*: wymagania funkcjonalne, np. główne cechy systemu, zabezpieczenia, możliwości.

***Usability** requirements*: wymagania dotyczące estetyki, spójności interfejsu użytkownika, dostępności dokumentacji i materiałów szkoleniowych

***Reliability** requirements*: wymagania dotyczące niezawodności aplikacji, odzyskiwania systemu.

***Performance** requirements*: wymagania wydajnościowe nałożone na wymagania funkcjonalne.

***Supportability** requirements*: wymagania dodatkowe dotyczące rozszerzalności produktu, jego testowania, instalacji, konserwacji i pielęgnacji.

Podejścia do formułowania wymagań

1. Nieformalne – opis tekstowy w języku naturalnym
2. **Półformalne –** wykorzystujące języki o formalnie zdefiniowanej składni a semantyce wyrażanej w j. naturalnym **(UML)**
3. Formalne - języki o dobrze zdefiniowanej składni i semantyce np. języki Z, CCS, VDM

Opis potrzeb klienta - przykład

Na podstawie danych demograficznych zaimportowanych z różnych systemów –jest możliwość generacji raportów (co rok lub częściej) dotyczących liczby osób pozostających w związkach (formalnych, nieformalnych) oraz ich potomstwa.
Projekt jest powiązany z projektem zainicjowanym przez pismo Demografia. Powstaje przy współpracy Głównego Urzędu Statystycznego (GUS).

Motywacja:

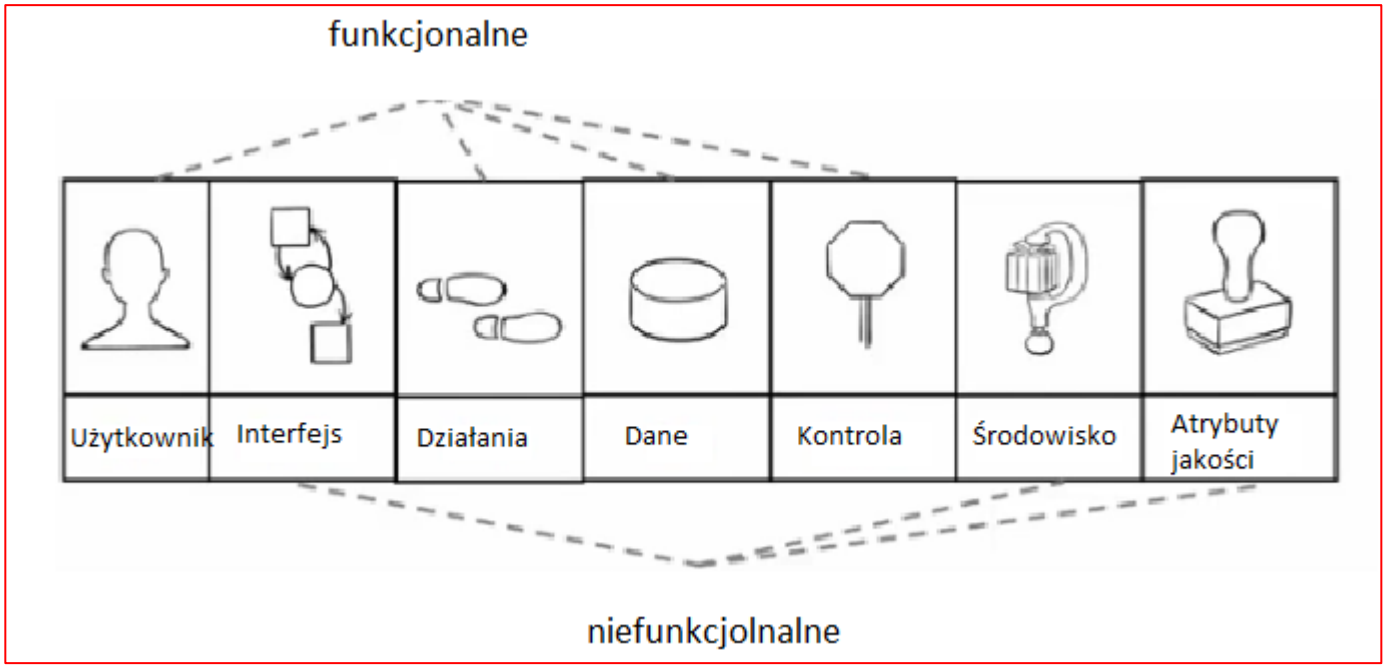
Problemy	a) Trudny dostęp do aktualnej informacji demograficznej o danym regionie. Niezadowalający czas oczekiwania na informacje demograficzne. b) Czasochłonna realizacja wniosków o udostępnienie informacji publicznej przez GUS. Istniejący system **BDR** nie pozwala opracować wszystkich potrzebnych raportów. c)Niska popularność i znajomość pisma *Demografia*.

Nieformalna specyfikacja własności systemu

FEAT 1 *(<<functional>>) – Raportowanie: System udostępnia raporty dotyczące danych demograficznych i ich zmian w czasie.*
FEAT 2 *(<<functional>>) – Import danych: System pobiera i przechowuje kopie wybranych danych z systemu BDR.*
FEAT 3 *(<<non-functional>>) – System jest dostępny w typowych przeglądarkach internetowych (minimum Explorer, FireFox)*
FEAT 4 *(<<functional>>) – System jest dostępny w polskiej i angielskiej wersji językowej*
FEAT 5 *(<<functional>>) – Rejestracja do systemu: Dostęp do systemu mają tylko uprawnieni użytkownicy.*

NREQ1 *– System udostępnia żądane raporty w czasie poniżej 12 s przy transmisji powyżej 56KB przy zalogowanych maksymalnie 10 użytkownikach (źródło: FEAT 8)*
NREQ2a *– Średni czas wykonania prostego raportu dla zalogowanego użytkownika nie przekracza 1 min (źródło: FEAT 9).*

Produkt programowy: 7 perspektyw opisu

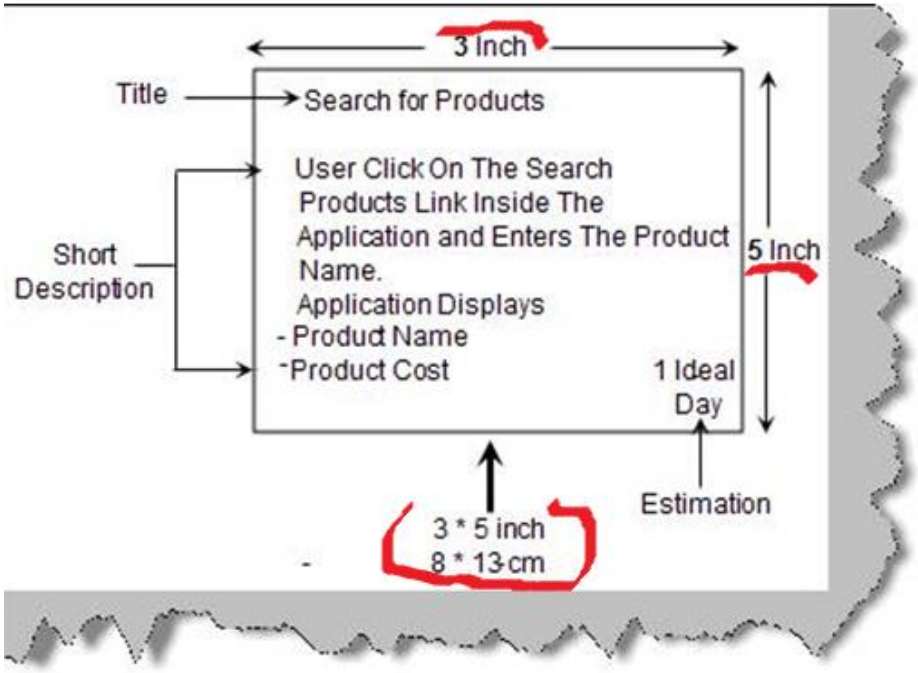


Wstępna specyfikacja wymagań – historyjki

Historyjka użytkownika (User story -US)

- krótko opisana pojedyncza funkcjonalność aplikacji lub
- zestaw funkcjonalności, wzajemnie zależnych (EPIKA)

Historyjka użytkownika - rozmiar, struktura



Wzorzec opisu historii:

„Jako <*rola*>, chcę <*cel/pragnienie*> aby <*korzyść*>„

równoważna - powszechnie stosowana - wersja krótsza:

„Jako <*rola*>, chcę <*cel/pragnienie* >„

Wypłata pieniędzy z bankomatu:

Jako klient

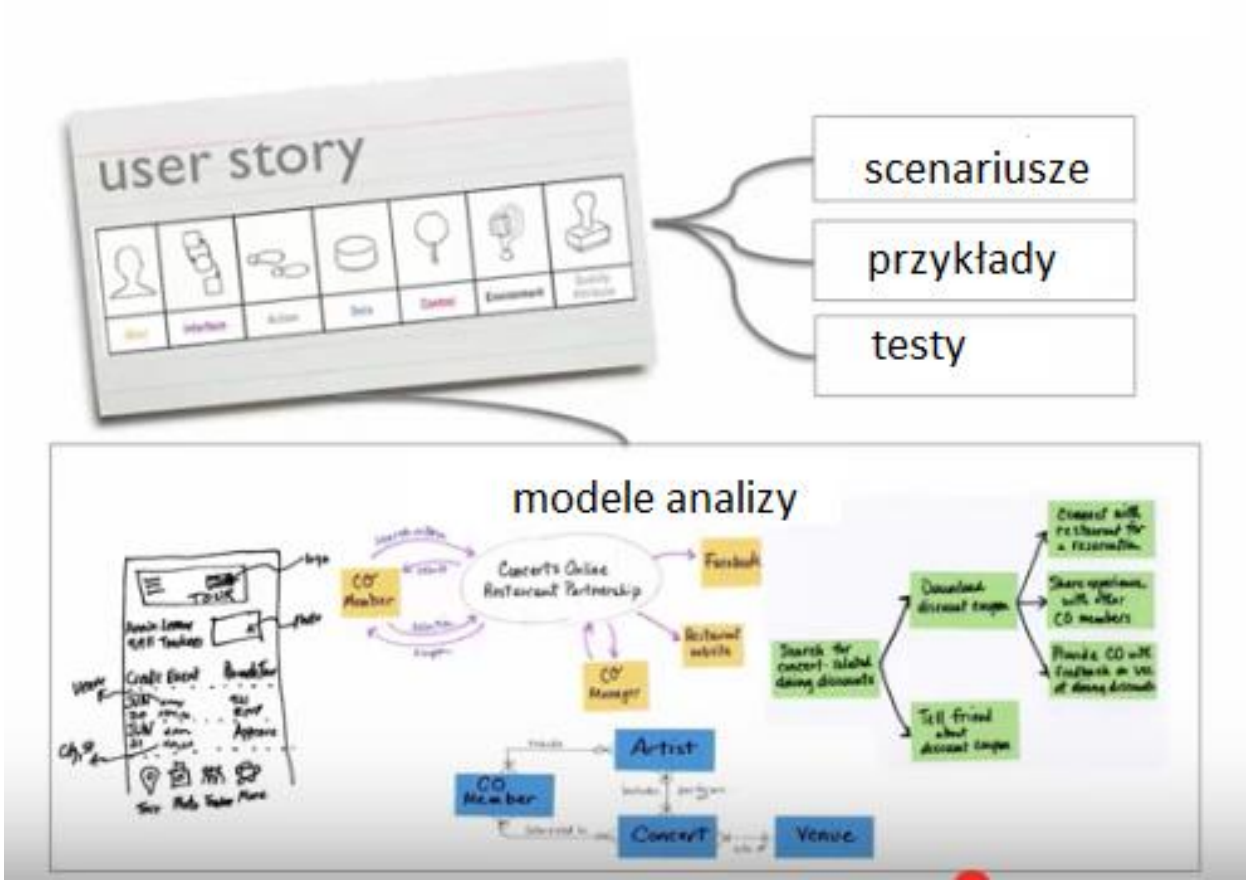
Chcę wypłacić pieniądze z bankomatu

Aby nie musieć stać w kolejce do kasy

Motywacja użycia historyjek

- Zrozumiałe dla każdego
- Dobry zakres do planowania
- Działa przy podejściu iteracyjnym
- Pozwala na odkładanie szczegółów
- Zachęca do udziału w projektowaniu
- Gromadzi/podbudowuje „milczącą” wiedzę

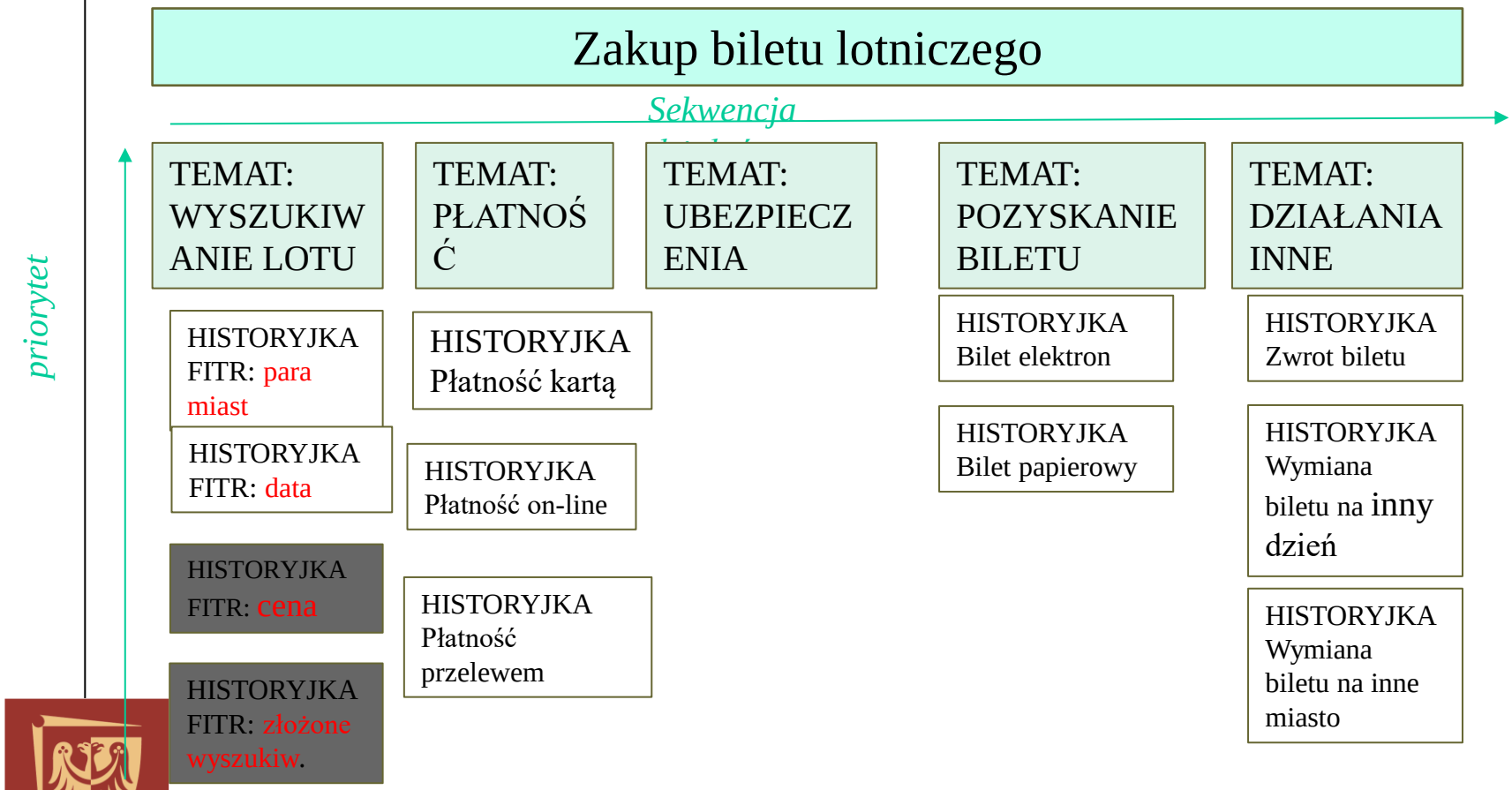
Produkt programowy: elementy specyfikacji



Dekompozycja epik- kryteria

1. Kroki procesu/pracy
2. Reguły biznesowe
3. Duży nakład(w realizacji)
4. Prosta/złożona
5. Wariacje w danych
6. Ważne niefunkcjonalne
7. Operacje CRUD
8. Spikes (architekt)

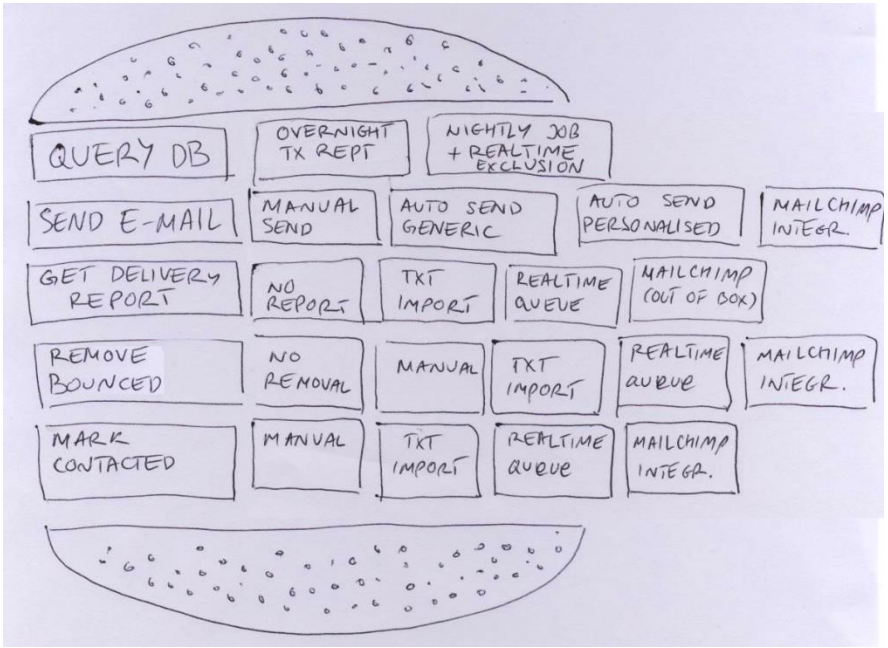
Dekompozycja (wg procesu) epiki – przykład



Dekompozycja epiki - podejście top-down/iteracyjne

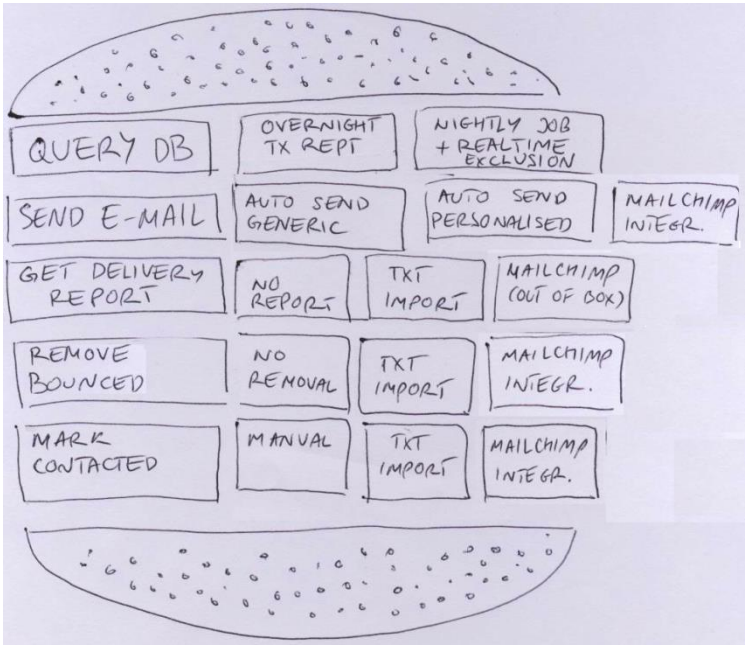
Krok1 –sformułowanie tematów

Krok2 – dekompozycja tematów -> HISTORYJKI

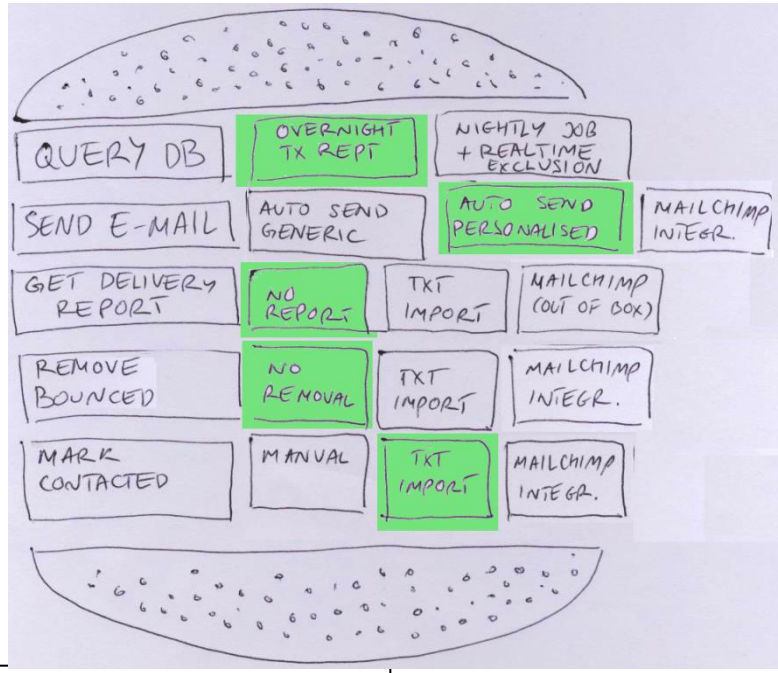


Dekompozycja epiki - podejście top-down/iteracyjne

Krok3 – ‚trymowanie'



Krok4 – wybór do implementacji



Historyjka użytkownika – struktura

Historyjka jest spisaną potrzebą klienta;

składa się z trzech części:

1. **tytułu**
2. **narracji:**
   – opis funkcjonalności (tekst, scenariusz)
   – korzyści, jakie płyną z danej funkcjonalności,
   – osobę (rolę), która będzie czerpać te korzyści)
3. **kryteriów akceptacji**

–**Given**–określający kontekst
–**When**–określający zdarzenie
–**Then**–określający rezultat)

Historyjka użytkownika –przykład (1)

Tytuł historii użytkownika: **Klient podejmuje gotówkę**. Wypłata z konta

Jako *klient*,
Chcę *wypłacić* gotówkę z bankomatu
Tak, by *nie trzeba czekać* w kolejce w banku.

Kryterium akceptacji 1:

- **Zakładając**, że na koncie jest odpowiednia ilość środków
  - **i**, że karta jest ważna
  - **i**, że w kasecie bankomatu są pieniądze
- **Jeżeli** klient zażąda wypłaty gotówki
  - **Wtedy** konto zostanie obciążone wypłatą + prowizja
  - **i** pieniądze (gotówka) zostaną wypłacone
  - **i** karta zostanie zwrócona klientowi

Historyjka użytkownika –przykład (2)

Kryterium Akceptacji 2:

Zakładając, że konto ma debet **(*GIVEN*)**
i karta jest ważna,
Jeśli klient żąda gotówki ***(WHEN)***
Wtedy upewnij się, ***(THEN)***
że wyświetlany jest komunikat odrzucenia
i nie wydaje się środków pieniężnych

Historyjki użytkownika – przykłady

1. Jako kierownik chcę wyszukiwać klienta podając ich imię i nazwisko.

2. Jako użytkownik chcę modyfikować swój harmonogram lecz nie można nic modyfikować innym użytkownikom.

3. *Klient* sklepu internetowego może dodać do swojego zamówienia produkt, którego szczegóły ogląda na stronie.

4. *Pracownik BOK* może zmienić na życzenie klienta zamówienie o podanym numerze. Wymaga to wcześniejszej autentykacji klienta sklepu

5. Użytkownik może wysłać powiadomienie email o nowo zdefiniowanym spotkaniu.

Lista historyjek użytkownika – wydanie 1(przykład)

Id	Tytuł	Definicja historii użytkownika	Obszar
H01	Rejestracja na konferencję	**Jako** osoba zainteresowana konferencją **chcę** mieć możliwość rejestracji na konferencję **po to**, aby zostać pełnoprawnym jej uczestnikiem	Obsługa konferencji
H02	Wykaz osób zarejestrowanych	**Jako** uczestnik konferencji **chcę** mieć wykaz osób zarejestrowanych na konferencję, **by** móc się z nimi skontaktować i sprawdzić i jak długi planują pobyt. **Jako** organizator **chcę** mieć wykaz osób zarejestrowanych **w celu** zarezerwowania miejsc w hotelu oraz posiłków.	Usługi informacyjne
H03	Wymagane przeglądarki internetowe	**Jako** użytkownik portalu **chcę** mieć możliwość pracy z programem poprzez różne przeglądarki internetowe (co najmniej Internet Explorer, FireFox), **by** nie zniechęcić użytkowników, którzy nie chcą korzystać z IE.	Wymaganie niefunkcjonalne
H04	Dostęp całodobowy do systemu	**Jako** użytkownik portalu **chcę** mieć możliwość korzystania z niego przez 24 h 7 dni w tygodniu, **bowiem** konferencja ma charakter międzynarodowy i osoby mieszkające w różnych strefach czasowych muszą mieć zapewniony dostęp do systemu bez ograniczeń czasowych.	Wymaganie niefunkcjonalne

Historyjki użytkownika – własności

- Pisze klient
  - w języku biznesu, pomaga w podaniu priorytetów
- Dobre historyjki (**INVEST**):
  - Niezależne (I)
  - Negocjowalne (N)
  - mają wartość dla klientów (V)
  - można je oszacować (E)
  - Małe (S)
  - Testowalne (T)

Historyjki użytkownika - niezależność

Przykład (poprawny):

- klient może płacić za przesyłkę kartą Visa
- klient może płacić za przesyłkę Mastercard
- klient może płacić za przesyłkę kartą American Express

Historyjki zależne jedna od drugiej są trudne do oszacowania i określenia priorytetu

Historyjki użytkownika - negocjowalne

- Karty z historyjkami – należy traktować jako element przypomnienia a nie kontrakt
- Szczegóły historyjki wyjaśnia się w czasie rozmów
- Karty z historyjkami zawierają frazę lub zdanie dla przypomnienia o konwersacji i notatkach z konwersacji

Historyjki użytkownika - wartość

- Wartość -dla osób korzystających z aplikacji i dla jej zleceniodawców
- Zapobiega wartościowaniu historyjek tylko przez twórcę oprogramowania

Przykład

– „*wszystkie połączenia do bazy danych powinny być realizowane za pomocą ..*”

można zastąpić przez

„do 50 użytkowników może korzystać z aplikacji z licencją na 5 użytkowników bazy danych”

Historyjki użytkownika - szacowanie

Historyjek nie można oszacować (czasowo), bo:

- Twórcy oprogramowania nie mają wiedzy dot. dziedziny problemu
  - wydobywaj szczegóły od klienta
- Twórcom oprogramowania brak odpowiedniej wiedzy technicznej
  - twórz „próbkę” do zbadania technologii
- Historyjka jest za duża
  - podziel na mniejsze

Historyjki użytkownika –rozmiar

- Łatwe do zaplanowania
- Łącz zbyt małe historyjki
- Dziel duże, złożone historyjki
  - W czasie konwersacji odkryto wiele dużych
  - Podział według twórz/usuń/uaktualnij
  - Podział według granic danych
  - Badania
- Sprawdź, że wykonany podział daje dobrą historykę

Historyjki użytkownika – rozmiar (cd.)

- Gdy za duże – trudno testować/kodować
- Za małe – więcej czasu na specyfikacje niż na implementacje
- Implementacja historyjki od 4 godzin do 2 tygodni
- Podział dużych na mniejsze
- Bez drobnych szczegółów – te w czasie konwersacji

Historyjki użytkownika – role

- Może być wiele typów użytkowników systemu
- Różne typy użytkowników mogą mieć różne role i różne historyjki
- Można uwzględnić niefaworyzowanych użytkowników jak i faworyzowanych
- Burza mózgów - początkowy zbiór ról użytkownika
- Utworzyć ten początkowy zbiór
- Skonsolidować role
- Ulepszyć role

Przewodnik – dobre historyjki (2)

- Rozmiar historyjek odpowiedni do czasu przeznaczonego na implementację
- **Nie zajmujemy się UI tak długo jak** to możliwe
- Nie opieramy się tylko na historyjkach, jeśli coś można wyrazić lepiej inaczej
- W historyjkach – rola użytkownika a nie „user” (l. pojedyncza)
- Strona czynna (a nie bierna)
- Nie numeruje się historyjek
- Należy pamiętać o celu historyjek

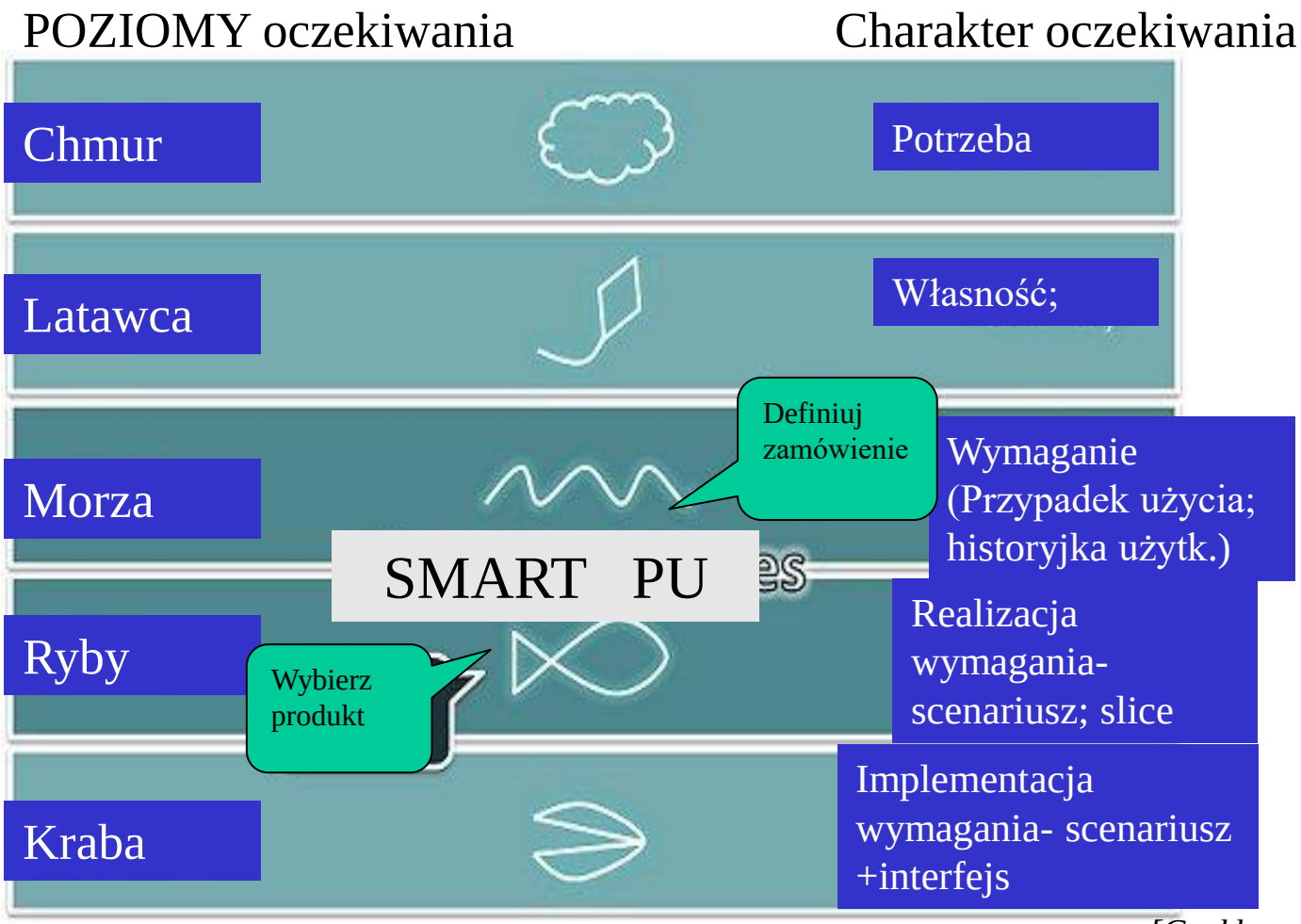
Niepoprawne historyjki

- Historyjki są za małe
- Są zależne między sobą
- Za dużo szczegółów
- Zbyt szybko zawierają szczegóły UI
- Wybiegają za daleko w przyszłość
- Klient ma trudności z określeniem priorytetów
- Klient nie chce pisać ani określać priorytetów historyjek

Wykład 2 – Inżynieria wymagań cz.2

1. Analiza i modelowanie wymagań:
   - przypadki użycia (PU wg UML):
     definicja
     struktura PU
     specyfikacja zachowania PU
2. Przykłady
3. Jakość wymagań

Specyfikacja oczekiwań klienta – poziomy abstrakcji



[Cockburn]

Model UML

zawiera 3 kategorie elementów:

- Klasyfikatory – opisuje zbiór obiektów; obiekt jest bytem /rzeczą posiadającą stan i związki/relacje z innymi obiektami.

- Zdarzenia - opisuje zbiór możliwych wystąpień; wystąpienie jest czymś co się zdarza i ma jakieś konsekwencje w systemie.

- Zachowanie - opisuje zbiór możliwych wykonań; wykonanie jest zachowaniem algorytmu zgodnym ze zbiorem reguł

Modele nie zawierają obiektów, wystąpień czy zachowań –sa one przedmiotem modelu.
Model zawierają diagramy **i specyfikacje**

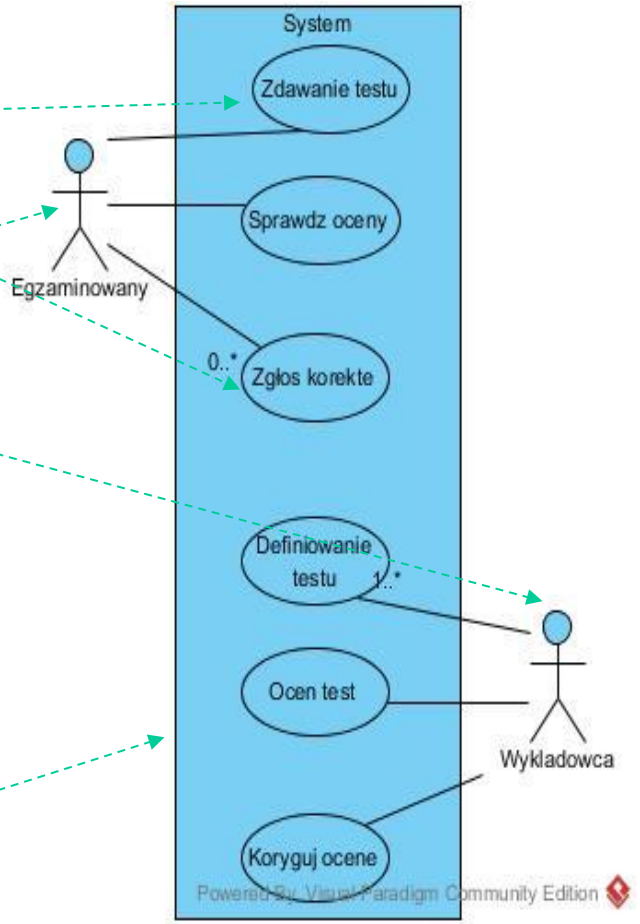
Wymaganie funkcjonalne → ***PRZYPADEK UŻYCIA***

- Kompletny zbiór akcji
- Wykonywany przez SYSTEM
- Inicjowany przez AKTORA
- Który dostarcza widocznych korzyści (Aktorowi)

(AKTOR jest kimś lub czymś, na zewnątrz systemu, co wchodzi w interakcje z SYSTEMEM)

Model *PRZYPADKÓW UŻYCIA*

- **Ilustruje**
  - zbiór przypadków użycia systemu
- **Identyfikuje**
  - **aktorów**, którzy wchodzą w interakcje z systemem
- **Definiuje**
  - jakie działania podejmuje system - przypadki użycia
  - to, co istnieje na zewnątrz systemu - aktorów
- **Wyrażony diagramem przypadków użycia**

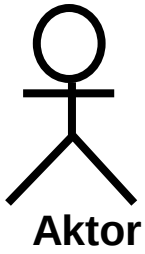


granica systemu

Model *PRZYPADKÓW UŻYCIA – składowe (1/3)*

Aktor

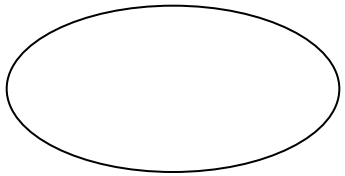
- ktoś (osoba) lub
- coś (inny system lub urządzenie),
- który wchodzi w interakcje (wysyła i odbiera komunikaty - wymienia informacje) z systemem
- oznaczenie w UML



Model *PRZYPADKÓW UŻYCIA – składowe (2/3)*

Przypadek użycia

- definicja
  - specyfikacja zbioru sekwencji akcji, wykonywanych przez system, których celem jest dostarczenie pewnej wartości aktorowi
  - reprezentuje pełną funkcjonalność z punktu widzenia aktora
- Charakterystyka
  - jest zawsze inicjowany przez aktora
- Oznaczenie w UML



Model *PRZYPADKÓW UŻYCIA – składowe (3/3)*

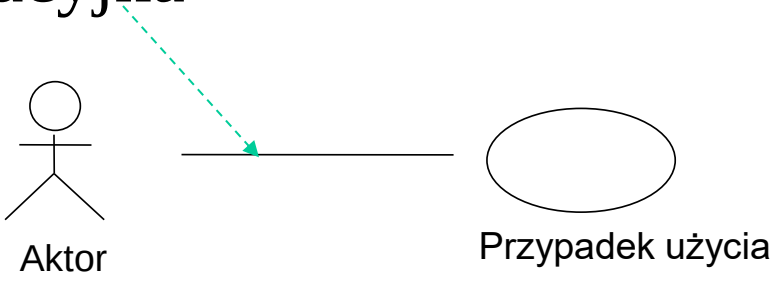
Opis:

- Uzasadnienie przypadku użycia
  - zaangażowane obiekty
  - cele, które mają być osiągnięte
- Jak przypadek użycia jest aktywowany
  - który aktor
  - w jakiej sytuacji
- Przepływ wiadomości pomiędzy aktorami i systemem
- Alternatywne przepływy wiadomości
- Jak przypadek użycia się kończy i jaką wartość zwraca aktorowi

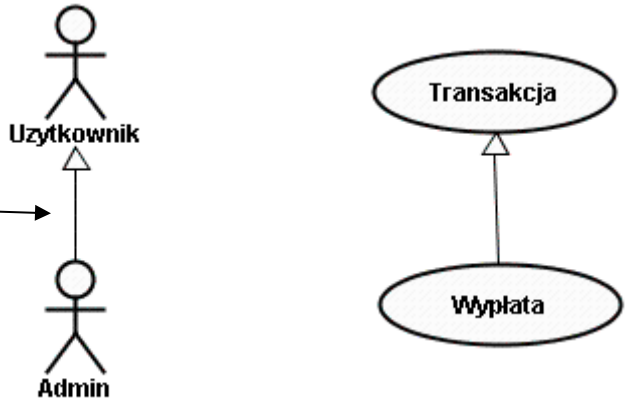
Struktura opisu ***PRZYPADKU UŻYCIA***

Model *PRZYPADKÓW UŻYCIA - relacje*

- Asocjacja komunikacyjna



- Generalizacja
  - Aktorów
  - Przypadków użycia

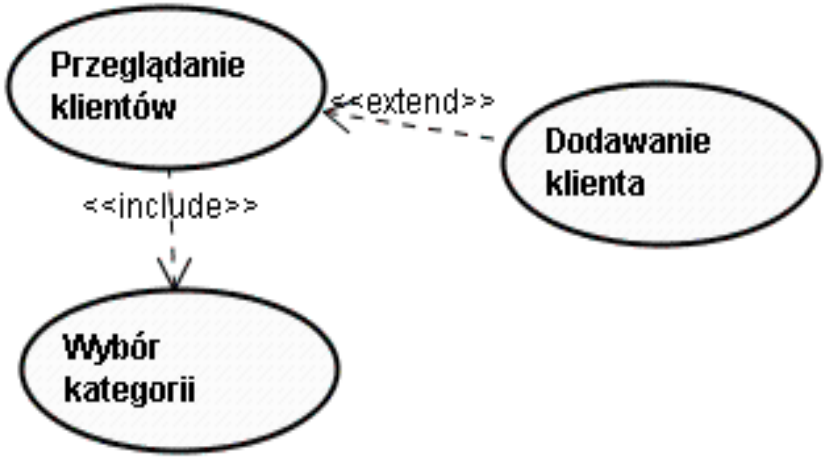


Model *PRZYPADKÓW UŻYCIA - relacje*

- Asocjacje przypadków użycia

Rozszerzenie → <<extend>>

Zawieranie → <<include>>



Styl ***PRZYPADKU UŻYCIA***: dwie kolumny

Akcje Aktora

- Start z inicjatywy Aktora
- Może być wiele akcji aktora zanim System odpowie

Akcje Systemu

- Odpowiedź na akcje aktora
- Nie podaje się wskazówek jak odpowiedź będzie implementowana (BLACK BOX!)

Dialog Aktora z Systemem;

Dobry dla specyfikacji ***Przypadku użycia*** wysokiego poziomu;

Styl ***PU***: dwie kolumny - przykład

	Aktor		**System**
1	Przypadek użycia rozpoczyna się gdy aktor otrzymuje główny dokument wejściowy inicjujący sprawę.	2	Wyświetla wykaz dopuszczalnych akcji
3	Wybiera opcję tworzenia nowej sprawy	4	Wyświetla formularz Nowa Sprawa
5	Wprowadza dane z dokumentu inicjującego sprawę wybierając m.in. typ sprawy.	6	Dla wybranego typu sprawy prezentuje pod-formularz do zarejestrowania specyficznych danych w zależności od typu sprawy
7	Rejestruje dane specyficzne dla wybranego typu sprawy	8	Stwierdza, że wybrany typ sprawy nie wymaga żadnych załączników obowiązkowych. Ustawia status sprawy na „przyjęta”.
		9	Stwierdza, że ten typ sprawy nie przewiduje żadnych załączników opcjonalnych.

Styl ***PRZYPADKU UŻYCIA***: specyfikacja pełna

- **Aktor**
- Udziałowcy i ich interesy
- **Warunki wstępne**
- **Gwarancja sukcesu (warunki końcowe)**
- **Główny scenariusz sukcesu**
- **Rozszerzenia**
- Wymagania specjalne
- Technologia i wykaz zmienności danych
- Częstość występowania
- Sprawy nieustalone

Styl *PU*: specyfikacja pełna - przykład

Identyfikator PU	005
Nazwa	Raportowanie
Aktorzy	Użytkownik
Krótki opis	Przypadek użycia służy do przygotowania wybranego *raportu danych demograficznych* dla zadanego roku lub okresu z określonego regionu (region jest ustalony arbitralnie podczas importu danych).
Warunki wstępne	Użytkownik jest zalogowany
Warunki końcowe	Brak
Scenariusz główny	1.Użytkownik chce poznać raport danych demograficznych. 2.System pyta o *właściwości raportu danych demograficznych*. 3.Użytkownik podaje *właściwości raportu danych demograficznych.* 4.System stwierdza, że *właściwości raportu danych demograficznych* są poprawne. 5.System stwierdza, że dane żądane przez użytkownika są dostępne, przygotowuje i odpowiednio wizualizuje raport.
Scenariusz alternatywny 1 – niepoprawne dane *właściwości raportu*	4a. System stwierdza, że *właściwości raportu* podane przez użytkownika są niepoprawne 5a. System informuje użytkownika o popełnionym błędzie Powrót do kroku 2
Wyjątek 1 – brak dostępu do danych	5a. System stwierdza, że dane żądane przez użytkownika nie są dostępne i informuje o tym użytkownika. Koniec scenariusza
Wymagania specjalne	NREQ1, NREQ2a, NREQ3, NREQ4, NREQ5, NREQ6
Źródło	FEAT 1

Politechnika Wrocławska

Przypadki użycia vs. historyjki użytkownika

Historyjka użytkownika

- zapewnia niewielka rozmiarowo i łatwą w użyciu prezentacje informacji.
- zwykle formułowana w codziennym języku użytkownika ;
- obejmuje mało detali, co pozwala na ich doprecyzowanie, ale powinny pozwolić czytelnikowi na zrozumienie, co ma wykonywać oprogramowanie.
- skojarzona z testami akceptacyjnymi, które doprecyzowują rozumienie wymagania.

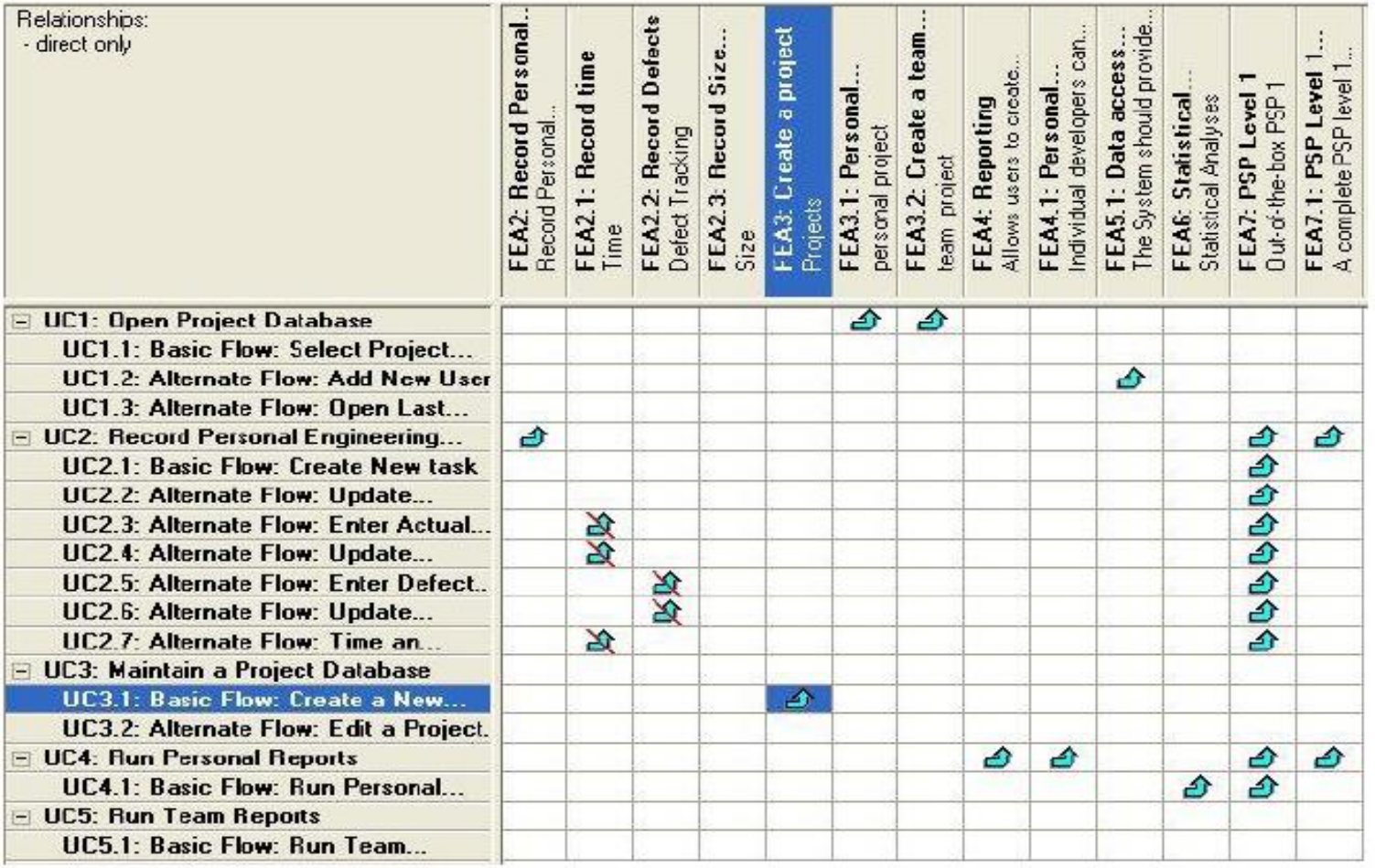
Przypadek użycia

- Opisuje proces i jego kroki w szczegółach i może być opisana w postaci formalnego modelu
- PU w swoim zamierzeniu jest samo-wystarczający do pełnego zrozumienia wymagania.
- Uogólniony zbiór interakcji między aktorem i systemem.

[1]

- Samodzielny dokument.

Weryfikacja kompletności wymagań – macierz ‘śladowania’



PRZYPADki UŻYCIA v 2.0 dla podejściu zwinnym

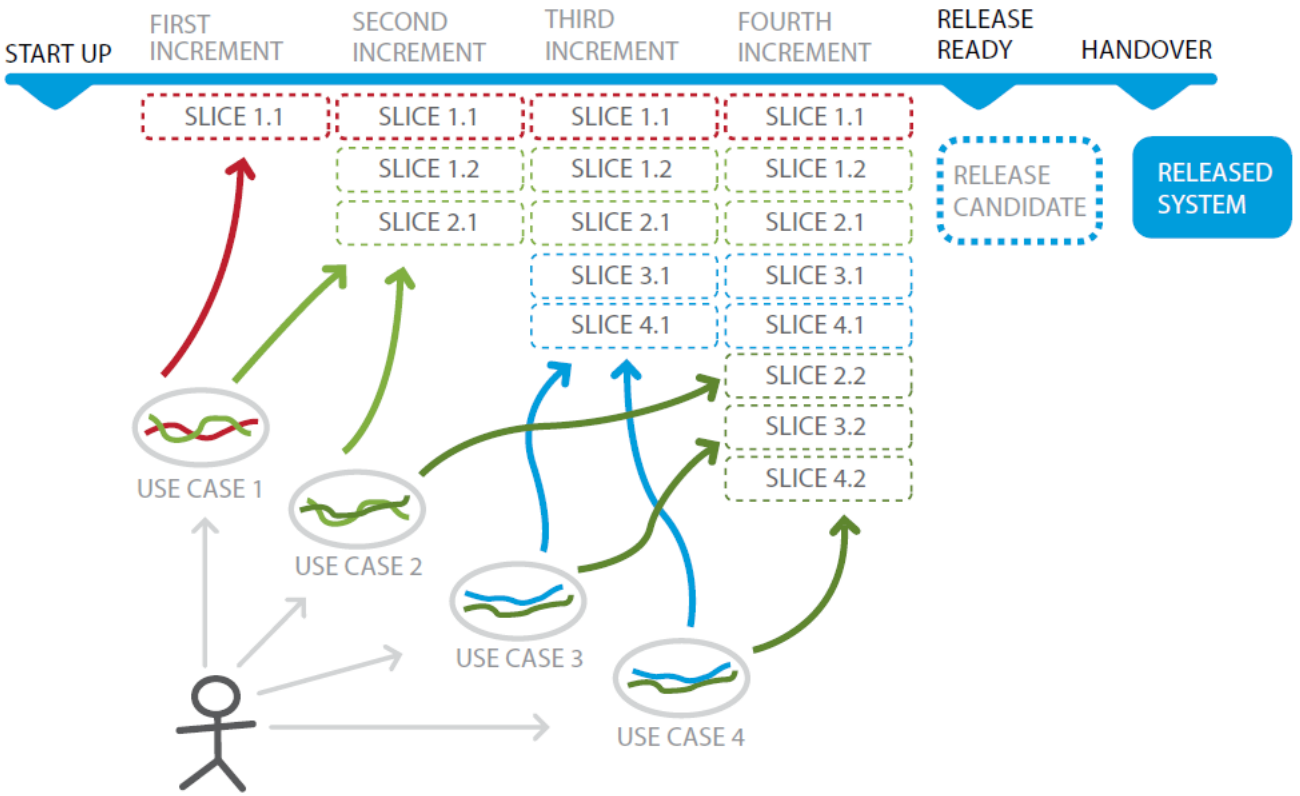


FIGURE 4: USE CASES, USE-CASE SLICES, INCREMENTS, AND RELEASES

PRZYPADEK UŻYCIA a „historyjki”

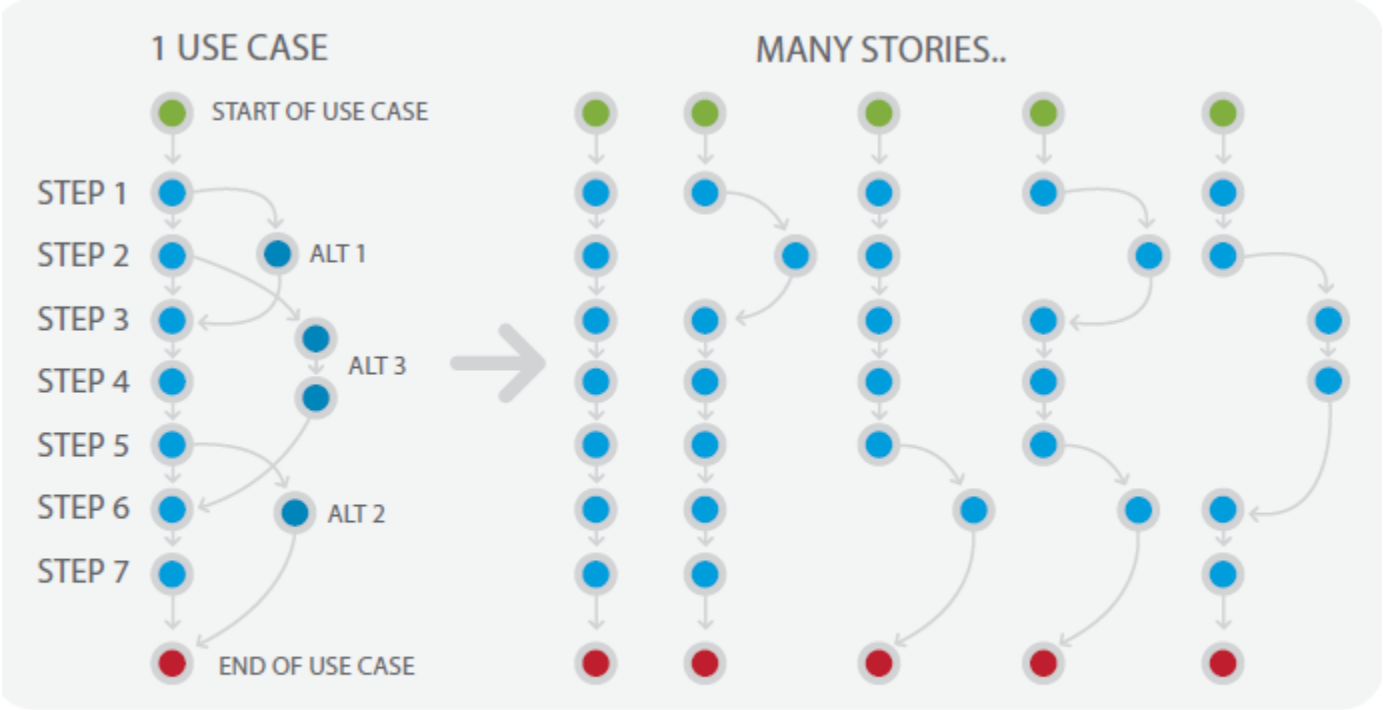


FIGURE 8:
THE RELATIONSHIP BETWEEN THE FLOWS AND THE STORIES

PRZYPADKI UŻYCIA v 2.0 dla podejścia zwinnego

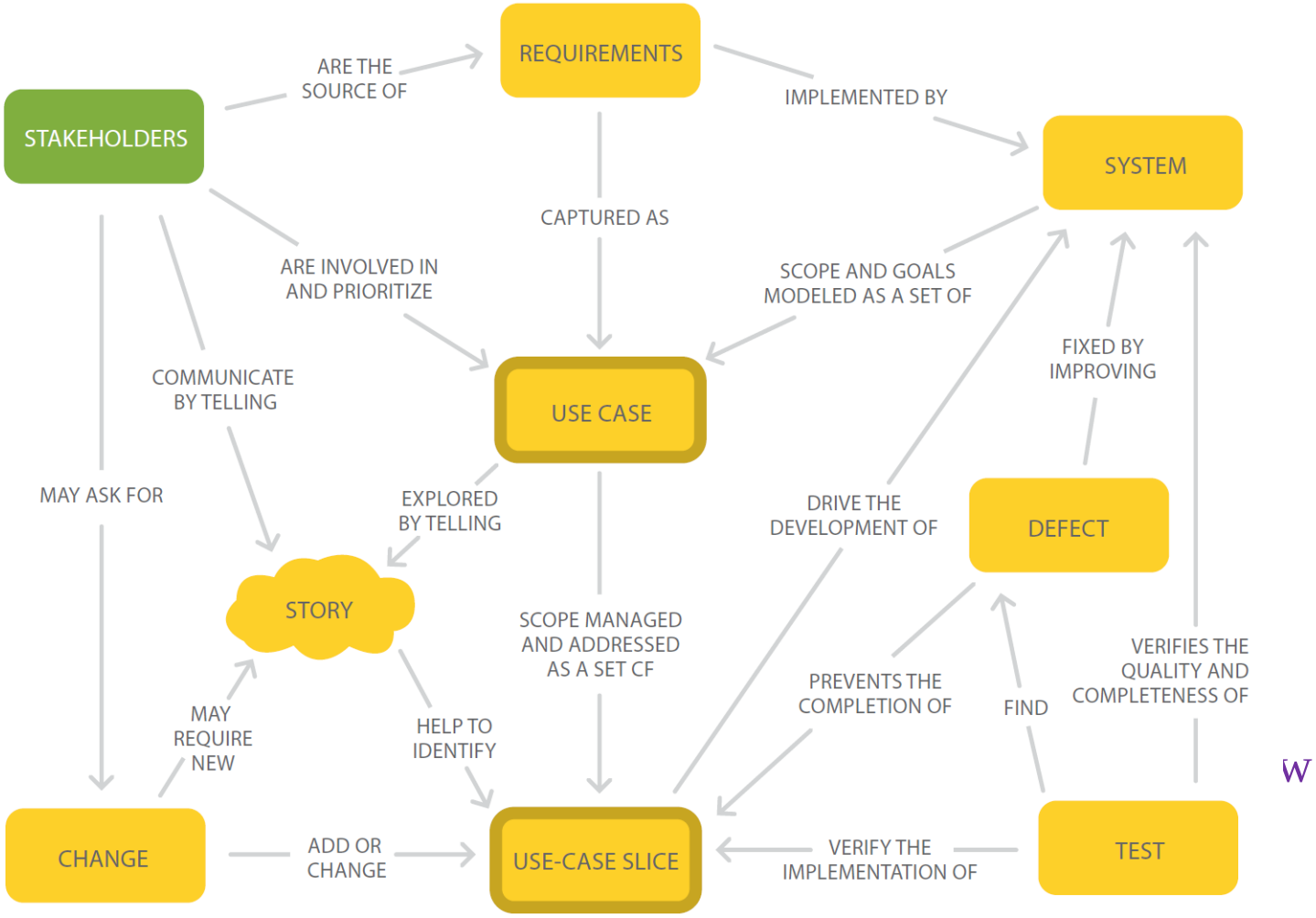


FIGURE 5: USE-CASE 2.0 CONCEPT MAP.

PRZYPADKI UŻYCIA v 2.0 - produkty

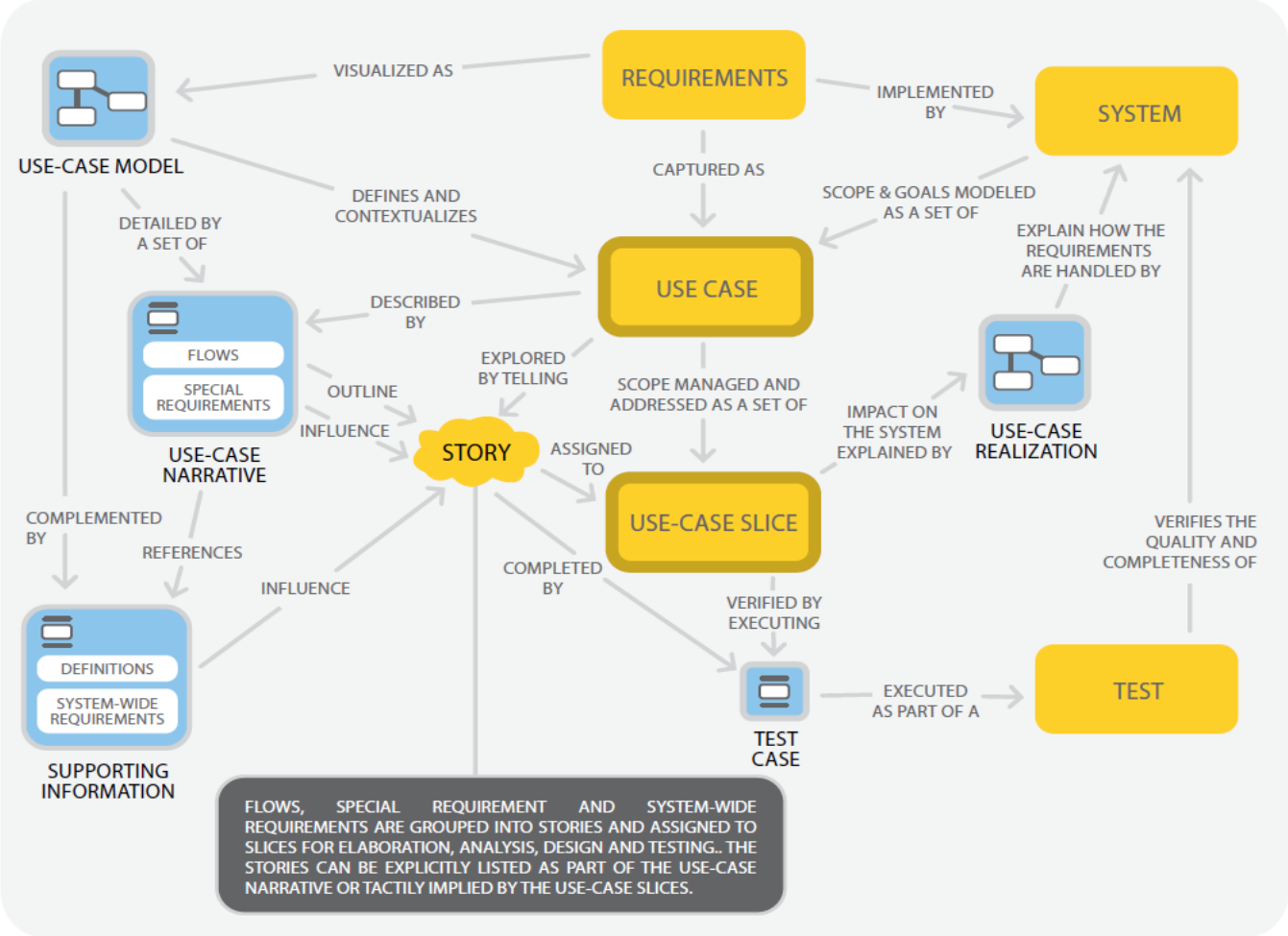


FIGURE 9: THE USE-CASE 2.0 WORK PRODUCTS

Przypadki użycia vs. historyjki użytkownika

Historyjka użytkownika

- zapewnia niewielka rozmiarowo i łatwą w użyciu prezentacje informacji.
- zwykle formułowana w codziennym języku użytkownika ;
- obejmuje mało detali, co pozwala na ich doprecyzowanie, ale powinny pozwolić czytelnikowi na zrozumienie, co ma wykonywać oprogramowanie.
- skojarzona z testami akceptacyjnymi, które doprecyzowują rozumienie wymagania.

Przypadek użycia

- Opisuje proces i jego kroki w szczegółach i może być opisana w postaci formalnego modelu
- PU w swoim zamierzeniu jest samo-wystarczający do pełnego zrozumienia wymagania.
- Uogólniony zbiór interakcji między aktorem i systemem.

[1]

- Samodzielny dokument.