

Dr Katarzyna Grzesiak-Kopeć

Inżynieria oprogramowania



3. Inżynieria wymagań



Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- Mierzenie oprogramowania
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

Wymaganie



Wymaganie w inżynierii jest pojedynczą, udokumentowaną potrzebą określonego produktu czy usługi, albo sposobu ich działania.

Jest to stwierdzenie identyfikujące potrzebne cechy, możliwości, charakterystyki lub jakość systemu, aby był on wartościowy i pożyteczny dla użytkownika.

Jakie wymagania?

- Funkcjonalne
 - Złożenie zamówienia
 - Generowanie korespondencji seryjnej
- Pozafunkcjonalne
 - Minimum 10 zamówień na godzinę
 - 100 000h MTBF
 - Maximum 8h na przeszkolenie użytkownika
- Ograniczenia

Spisanie wymagań, to sztuka!

- Nie ma uniwersalnego sposobu
- Ucz się na cudzych doświadczeniach
 - Korzystaj ze sprawdzonych praktyk

- Software Requirements Specification

- Specyfikacja konkretnego produktu oprogramowania realizującego zadaną funkcjonalność w określonym środowisku
- SRS może być napisana przez jedną lub wiele osób reprezentujących dostawcę albo/i klienta



SRS zyski

- Uzgodnienie co ma produkt robić pomiędzy klientem a dostawcą
- Zmniejsza wysiłek związany z budową systemu
- Podstawa szacowania kosztów i harmonogramu
- Podstawa walidacji i weryfikacji
- Ułatwia przenoszenie
 - Nowi użytkownicy
 - Nowe maszyny
 - Nowy klient
- Podstawa dla rozbudowy systemu

SRS zakres

- Funkcjonalność
 - Co oprogramowanie powinno robić?
- Zewnętrzne interfejsy
 - Jak komunikuje się z ludźmi, sprzętem, innym oprogramowaniem?
- Wydajność
 - Jaka jest prędkość, czas reakcji, dostępność etc.?
- Atrybuty
 - Przenośność, poprawność, pielęgnowalność, bezpieczeństwo etc.
- Ograniczenia projektowe
 - Wymagane standardy, języki programowania, ograniczenia zasobów, środowisko operacyjne etc.

Dobra specyfikacja

- Poprawna (Correct)
 - Każde wymaganie jest takim wymaganiem, które powinien realizować system
- Jednoznaczna (Unambiguous)
 - Każde wymaganie ma dokładnie jedną interpretację

Dobra specyfikacja

- Kompletna (Complete) zawiera
 - Wszystkie znaczące wymagania funkcjonalne, wydajnościowe, ograniczenia projektowe, atrybuty, czy zewnętrzne interfejsy
 - Definicje odpowiedzi na wszystkie możliwe rodzaje wejść we wszystkich możliwych typach sytuacji; na dane wejściowe poprawne, jak i na te niepoprawne
 - Etykiety rysunków, tabel i diagramów
 - Definicje wszystkich pojęć i miar

Dobra specyfikacja

- Spójna (Consistent)
 - Z pozostałą dokumentacją
- Ustalająca priorytety ważności i stabilności
 - Kluczowe, pożądane, opcjonalne, warunkowe
- Weryfikowalna (Verifiable)
 - Każde wymaganie jest weryfikowalne
- Umożliwiająca śledzenie powiązań (Traceable)
 - Jasno sprecyzowane skąd się wzięło wymaganie
 - Odwołania do wymagań w pozostałej dokumentacji projektu

Dobra specyfikacja

- Modyfikowalna (Modifiable)
 - Struktura i styl nie ulegają zmianie przy wprowadzaniu zmian w wymaganiach
 - Spójna i jasna organizacja ze spisem treści, indeksem, wewnętrznymi odnośnikami
 - Bez redundancji (każde wymaganie występuje tylko raz!)
 - Każde wymaganie jest oddzielnie opisane (nie łączymy różnych wymagań!)

Zawartość SRS

IEEE Recommended Practice for Software Requirements Specifications

IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

Table of Contents

1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, acronyms, and abbreviations
 - 1.4 References
 - 1.5 Overview
 2. Overall description
 - 2.1 Product perspective
 - 2.2 Product functions
 - 2.3 User characteristics
 - 2.4 Constraints
 - 2.5 Assumptions and dependencies
 3. Specific requirements (See 5.3.1 through 5.3.8 for explanations of possible specific requirements. See also Annex A for several different ways of organizing this section of the SRS.)
- Appendices
- Index

Model Sommerville'a-Sawyera

- Ian Sommerville & Pete Sawyer
 - Requirements Engineering: A Good Practice Guide, Wiley, 1997



Podział dobrych praktyk

- Podstawowe
 - Zdefiniuj standardową strukturę dokumentu specyfikacji wymagań
 - Zdefiniuj szablony do opisywania wymagań
- Pośrednie
 - Stosuj scenariusze użycia
- Zaawansowane
 - Przypisz ryzyko wymaganiom

Klasyfikacja dobrych praktyk

J. Nawrocki, Inżynieria wymagań

	Podst. 36	Pośr. 21	Zaaw. 9
Dokument SRS	8	–	–
Zbieranie wymagań	6	6	1
Analiza i negocjacja wymagań	5	2	1
Opisywanie wymagań	4	1	–
Modelowanie systemu	3	3	–
Walidacja wymagań	4	3	1
Zarządzanie wymaganiami	4	3	2
IW dla systemów krytycznych	2	3	4

1. Zdefiniuj standardową strukturę dokumentu
2. Wyjaśnij, jak korzystać z dokumentu
3. Dołącz streszczenie wymagań
4. Opracuj uzasadnienie biznesowe dla systemu
5. Zdefiniuj terminy specjalistyczne
6. Wybierz czytelny szablon dokumentu
7. Pomóż czytelnikom znaleźć informację
8. Uczyń dokument łatwym do zmiany

Przypadki użycia

It shouldn't take longer than 15 minutes to teach someone how to write a use case!

Robert Martin

Przypadki użycia

UC 1 Loguj

Aktor:	Użytkownik
Stan systemu przed:	System uruchomiony
Stan systemu po:	Użytkownik otrzymał dostęp do odpowiednich zasobów

SCENARIUSZ PODSTAWOWY

- 1 Użytkownik wyraża chęć zalogowania się.
- 2 System prosi o wprowadzenie identyfikatora oraz hasła.
- 3 Użytkownik wprowadza żądane informacje i zatwierdza je.
- 4 System stwierdza, że wprowadzono odpowiednie dane, weryfikuje tożsamość użytkownika oraz dokonuje autoryzacji.
- 5 System daje użytkownikowi dostęp do odpowiadających mu zasobów.

Przypadki użycia

UC 1 Loguj

Aktor:	Użytkownik
Stan systemu przed:	System uruchomiony
Stan systemu po:	Użytkownik otrzymał dostęp do odpowiednich zasobów

SCENARIUSZ PODSTAWOWY

- 1 Użytkownik wyraża chęć zalogowania się.
- 2 System prosi o wprowadzenie identyfikatora oraz hasła.
- 3 Użytkownik wprowadza żądane informacje i zatwierdza je.
- 4 System stwierdza, że wprowadzono odpowiednie dane, weryfikuje tożsamość użytkownika oraz dokonuje autoryzacji.
- 5 System daje użytkownikowi dostęp do odpowiadających mu zasobów.

Tutaj może być scenariusz alternatywny.

Przypadki użycia

UC 1 Loguj

Aktor:	Użytkownik
Stan systemu przed:	System uruchomiony
Stan systemu po:	Użytkownik otrzymał dostęp do odpowiednich zasobów

SCENARIUSZ PODSTAWOWY

- 1 Użytkownik wyraża chęć zalogowania się.
- 2 System prosi o wprowadzenie identyfikatora oraz hasła.
- 3 Użytkownik wprowadza żądane informacje i zatwierdza je.

- 4 System stwierdza, użytkownika oraz

- 5 System daje użytk

SCENARIUSZE ALTERNATYWNE

3a Rezygnacja

3a1 Użytkownik rezygnuje z akcji logowania, co kończy PU.

3b Anulowanie

3b1 Użytkownik anuluje wprowadzone informacje.

3b2 Powrót do pkt. 2.



Przypadki użycia

SCENARIUSZE ALTERNatywne

4a	Brak lub błędne dane
4a1	System stwierdza, że brakuje pewnych informacji lub, że są one niepoprawne.
4a2	System informuje użytkownika o wykrytych problemach i powrót do pkt. 2.
4b	Błąd uwierzytelnienia
4b1	System stwierdza, że w systemie nie figuruje albo użytkownik o podanym identyfikatorze, albo para identyfikator-hasło .
4b2	System informuje użytkownika o wykrytych problemach i powrót do pkt. 2.
4c	Brak autoryzacji
4c1	System stwierdza, że podany użytkownik nie posiada uprawnień do żadnych zasobów o czym go informuje, co kończy PU.
5	System daje użytkownikowi dostęp do odpowiadających mu zasobów.

Dobre przypadki użycia

- Używają interdyscyplinarnego języka
 - Techniczny, nietechniczny
- Opisują co system będzie robił
 - Jego zachowanie realizujące żądane usługi
- Zapisują podejmowane decyzje
- Pozwalają weryfikować kompletność
- Dostarczają kontekstu dla wymagań
 - Wprowadzają strukturę do opowieści

SĄ	NIE SĄ
Tekst	Diagramami PU w UMLu
Brak GUI	Opisem GUI
Brak formatu danych	Opisem formatu danych
3 - 9 kroków w głównym scenariuszu	Wielostanicowym scenariuszem
Łatwo je zrozumieć	Nieczytelne
Przedstawiają potrzeby użytkownika	Opisują cechy programu
Zapisują podjęte decyzje	Opisem dziedziny biznesowej

Zalety i wady PU

ZALETY	WADY
Zapisują wymagania funkcjonalne w łatwej do zrozumienia formie	Zapisują wyłącznie wymagania funkcjonalne
Stanowią szkielet dla zapisu wymagań pozafunkcjonalnych oraz dla harmonogramu	Projektowanie nie ogranicza się do samych PU

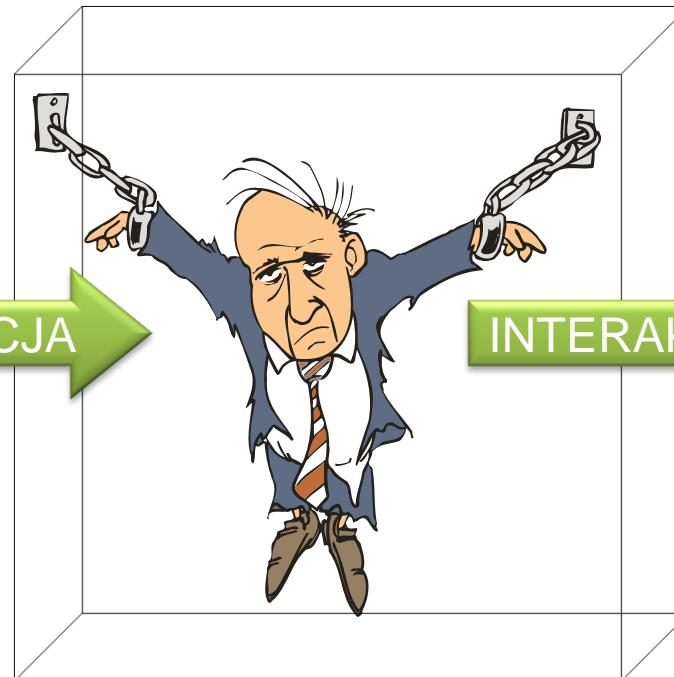
Przypadek użycia

- Aktorzy współpracują by zrealizować określony cel

Główny Aktor
osoba lub system
chce wykorzystać PS
w określonym celu



Projektowany System

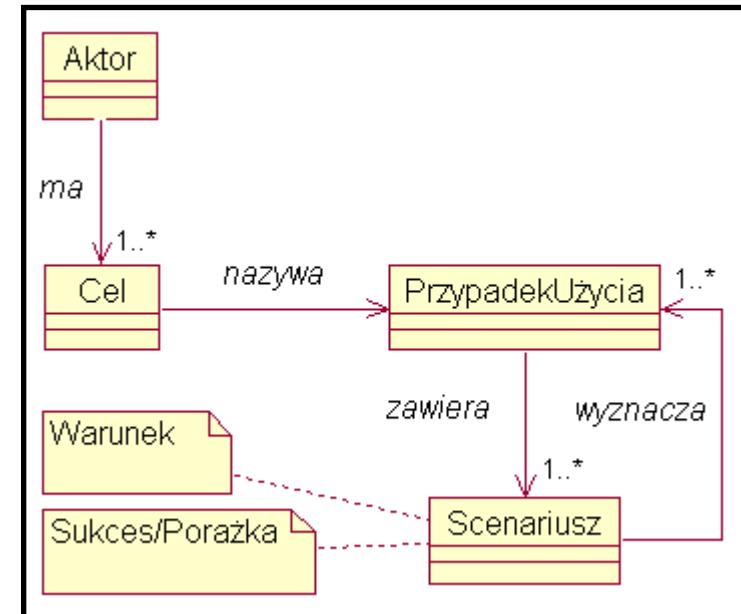


Aktor drugoplanowy
inni system realizujący
cel dla PS



Przypadek użycia

- Aktor ma cele
- Cele nazywają przypadki użycia (UC)
- Przypadki mają scenariusze nazywające „podprzypadki użycia”



Jak pisać?

- Wskaż aktorów i ich cele
 - Aktor to coś/ktoś, co/kto wchodzi w interakcję z naszym systemem
 - Czego potrzebuje od systemu aktor?
- Napisz krótko
 - Pomyślny scenariusz
 - Wychwyć zamierzenia oraz odpowiedzialność wszystkich aktorów
 - Jakie wymieniają między sobą informacje
 - Ponumeruj linie

Jak pisać?

- Wypisz warunki zakończone niepowodzeniem (porażki) jako rozszerzenia
 - Zazwyczaj każdy krok może zakończyć się niepowodzeniem
- Opisz porażki do momentu zakończenia lub powrotu do głównego scenariusza
- Zwróć uwagę na zmieniające się dane
 - np. Zwrot pieniędzy – gotówką, czekiem, ...
 - Wskazują sytuacje, które muszą być obsłużone przez przypadki użycia niższego poziomu



Typowe wpadki!

Register for Courses

1. Display a blank schedule.
2. Display a list of all classes in the following way:
The left window lists all the courses in the system in alphabetical order.
The lower window displays the times the highlighted course is available.
The third window shows all the courses currently in the schedule.
3. Do
4. Student clicks on a course.
5. Update the lower window to show the times the course is available.
6. Student clicks on a course time and then on the “Add Course” button.
7. Check if the Student has the necessary prerequisites and that the course offering is open.
8. If the course is open and the Student has the necessary prerequisites, add the Student to the course. Display the updated schedule showing the new course. If no, put up a message, “You are missing the prerequisites. Choose another course.”
9. Mark the course offering as “enrolled” in the schedule.
10. End do when the Student clicks on “Save Schedule.”
11. Save the schedule and return to the main selection screen.

Typowe wpadki!



Register for Courses

1. ?Display a blank schedule.
2. ?Display a list of all classes **in the following way**:
The left window lists all the courses in the system in alphabetical order.
The lower window displays the times the highlighted course is available.
The third window shows all the courses currently in the schedule.
3. ?Do
4. Student **clicks** on a course.
5. ?Update **the lower window** to show the times the course is available.
6. Student **clicks** on a course time and then on **the “Add Course” button**.
7. ?Check if the Student has the necessary prerequisites and that the course offering is open.
8. ?If the course is open and the Student has the necessary prerequisites, add the Student to the course. **Display** the updated schedule showing the new course. If no, **put up a message**, “You are missing the prerequisites. Choose another course.”
9. ?Mark the course offering as “enrolled” in the schedule.
10. ?End do when the Student **clicks** on **“Save Schedule”**.
11. ?Save the schedule and **return** to the main selection screen.

Powinno być tak



Register for Courses

System: Course Enrollment System

Goal level: User Goal

1. Student requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets available courses from the Course Catalog System.
4. Student selects up to 4 primary and 2 alternate course offerings.
5. For each course, the system verifies that the Student has the necessary prerequisites, adds the Student to the course, marking Student as "enrolled" for that course in the schedule.
6. When the Student indicates the schedule is complete, the system saves it.

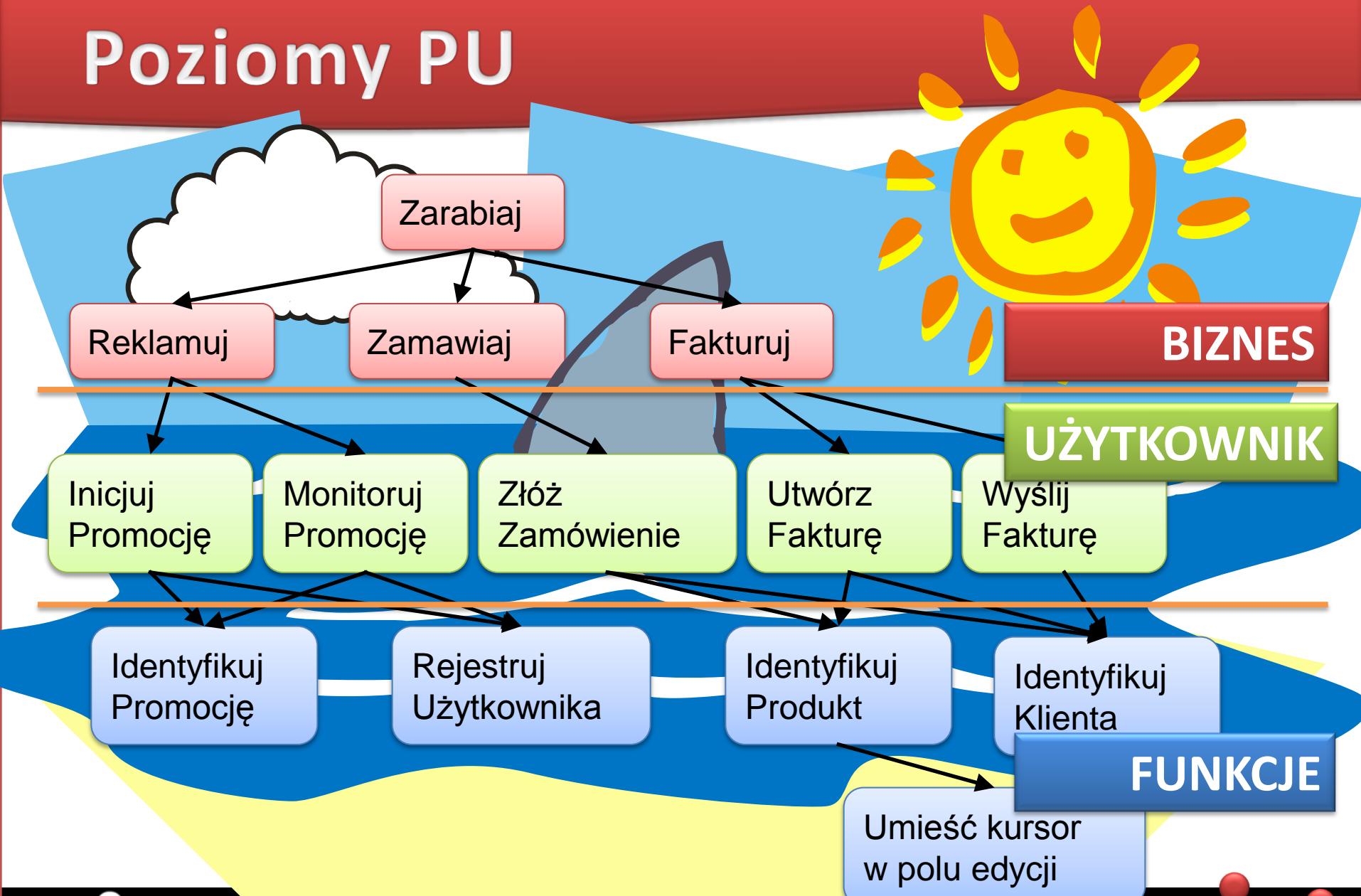
Extensions:

- 1a. Student already has a schedule: System brings up the current version of the Student's schedule for editing instead of creating a new one.
- 1b. Current semester is closed and next semester is not yet open: System lets Student look at existing schedules, but not create new ones.
- 3a. Course Catalog System does not respond: The system notifies the Student and the use case ends.
- 5a. Course full or Student has not fulfilled all prerequisites: System disables selection of that course and notifies the Student.

Poziomy przypadków

- Poziom celu biznesowy (Summary Goals)
- Poziom celu użytkownika (User Goals)
- Poziom podfunkcji (Subfunctions)

Poziomy PU



Jak pisać?

- <http://alistair.cockburn.us/>
- A. Cockburn, Jak pisać efektywne przypadki użycia
- S. Adolph, P. Bramble, A. Cockburn, A. Pols:
Patterns for Effective Use Cases



- Pojedynczy przypadek użycia
- Zbiór przypadków użycia
- Proces tworzenia
- Zespół autorów

Przypadek użycia

- Fraza czasownikowa w nazwie
 - Akcja
 - Generuj raport
 - Źle
 - Raport
- Scenariusz i rozszerzenia
 - 3-9 kroków w scenariuszu podstawowym
 - Scenariusze alternatywne

Przypadek użycia

- Obojętność technologiczna
 - Dlaczego pomijamy
 - Technologia jest zmienna
 - Niepotrzebne ograniczenia
 - Szczegóły GUI zaciemniają obraz
 - Klient nie rozumie terminów technicznych
 - Przykład
 - Student zaznacza checkbox

Zbiór przypadków

- Rozwijalna struktura
 - Hierarchia
 - Można rozwijać lub zwijać w celu pokazania lub ukrycia szczegółów

Proces tworzenia

- Pisz najpierw w szerz, potem w głęb
 - Lista aktorów
 - Nazwy przypadków użycia
 - Główne scenariusze
 - Rozszerzenia

Zespół

- Mały
 - 2-3 osoby (duży system – kilka zespołów)
 - Więcej recenzentów
- Zrównoważony
 - Zróżnicowany (analityk + klient)
 - Synergia (uzupełnianie się)

Zwinna specyfikacja wymagań

- Karty wymagań (User Stories)
 - Opowiadania/Scenariusze użytkownika
 - Historyjki użytkownika ☺
- Szkice ekranów
- Testy akceptacyjne

Madeyski L., Kubasiak M.: Zwinna specyfikacja wymagań

Karta wymagań

Jako [osoba odgrywająca daną rolę] chciałbym móc [wykonać jakąś czynność] aby [osiągnąć jakiś cel].

Karta wymagań

Jako [osoba odgrywająca daną rolę] chciałbym móc [wykonać jakąś czynność] aby [osiągnąć jakiś cel].

- Jako klient chciałbym móc złożyć zamówienie.
- Jako użytkownik chciałbym móc się zalogować.

Karty wymagań

UTWÓRZ KONTO

OPIS

Jako administrator chciałbym móc utworzyć konto pracownikowi dydaktycznemu, aby umożliwić mu dostęp do systemu.

Po wypełnieniu wymaganych pól, system automatycznie generuje hasło oraz wysyła nowemu użytkownikowi wiadomość e-mail z danymi dostępowymi.

PRIORYTET

Krytyczne

OSZACOWANIE (godz.)

6

Szkic ekranu

Testy akceptacyjne

UTWÓRZ KONTO testy akceptacyjne

- 1 Administrator wypełnił prawidłowo wszystkie wymagane pola, wprowadził również unikalną wartość identyfikatora użytkownika – w nowo otwartym oknie system informuje o pomyślnym utworzeniu konta i wysłaniu wiadomości e-mail.
- 2 Administrator wypełnił wszystkie wymagane pola, jednak w systemie figuruje już użytkownik o takim samym identyfikatorze – system wyświetla na formularzu komunikat ostrzegawczy o tym, że pole „login” musi zawierać unikalną wartość.
- 3 Administrator nie wypełnił wymaganych pól (oznaczonych *) – system wyświetla na formularzu informację o niekompletnych danych wraz z prośbą o ich uzupełnienie.

PU vs karty wymagań

PRZYPADKI UŻYCIA	KARTY WYMAGAŃ
Opisuję proces jako sekwencję kroków	Informują jedynie o pewnej funkcjonalności
Poziom szczegółowości pozwala na rozważanie ich pojedynczo	By je jednoznacznie zinterpretować należy dołączyć do nich testy akceptacyjne

PU vs karty wymagań

... DON'T SUBDIVIDE ...

- A **user story** cut in half makes two smaller user stories, as a flatworm cut in half makes two smaller flatworms, but
- a **use case** cut in half doesn't make two small UCs, as a horse cut in half doesn't make two small horses.

Alistair Cockburn



KONIEC

