Testowanie oprogramowania

#3 Wymagania i przypadki testowe

Autor: Paulina Kucharczyk

AGENDA

1

Punkt widzenia

Inżynieria wymagań

2

Przypadki testowe

Przyczyny i źródła testów

01.

Punkt widzenia

Inżynieria wymagań

Wymagania

Biznesowe

Cele biznesowe, regulacje prawne, zgodności

Dziedzinowe

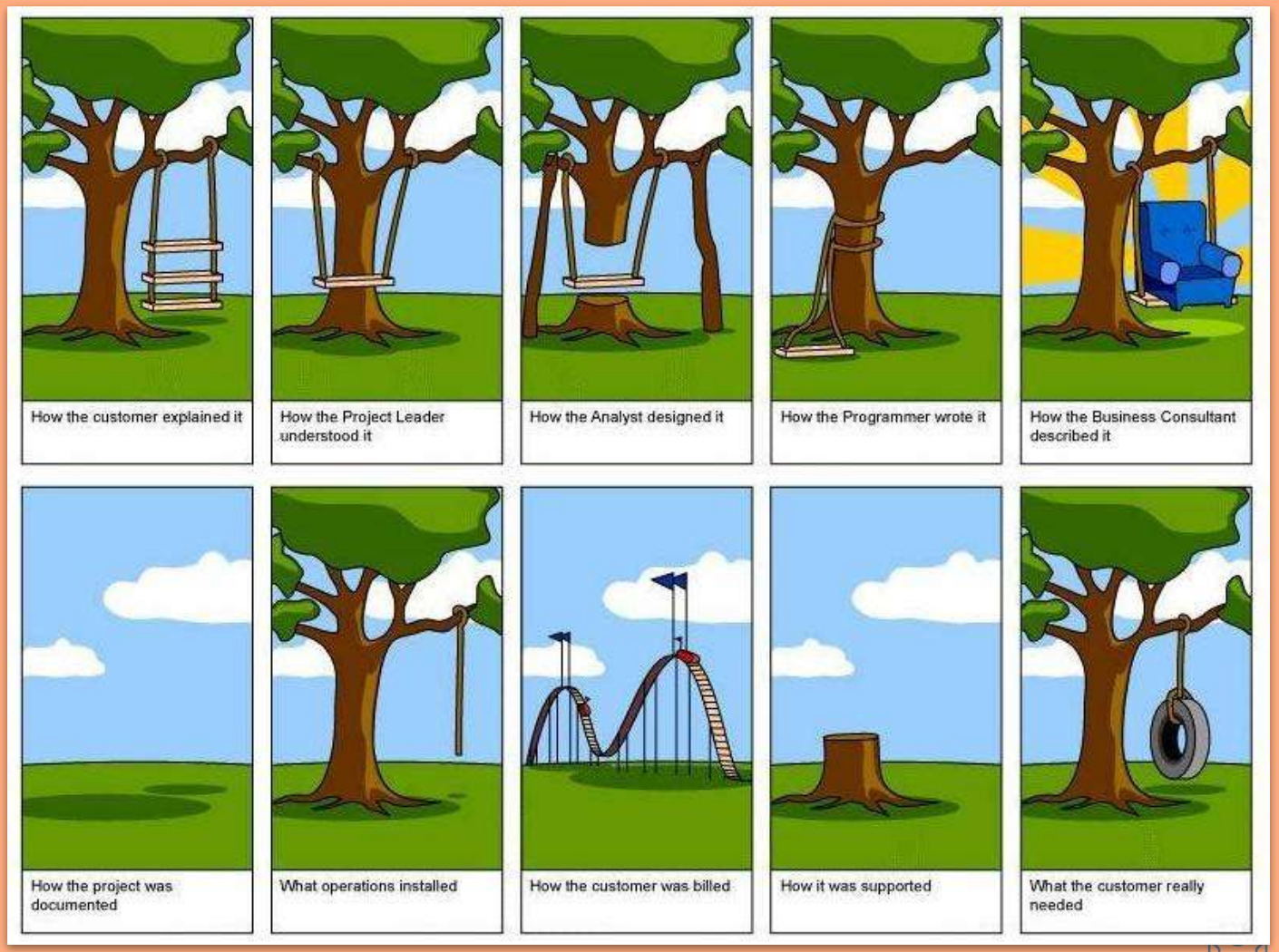
Zastosowanie systemu i sposób wykonywania

Niefunkcjonalne

Możliwości ograniczenia systemu.
Jak system ma działać?

Funkcjonalne

Funkcjonalności i usługi systemu.
Co system ma robić?


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

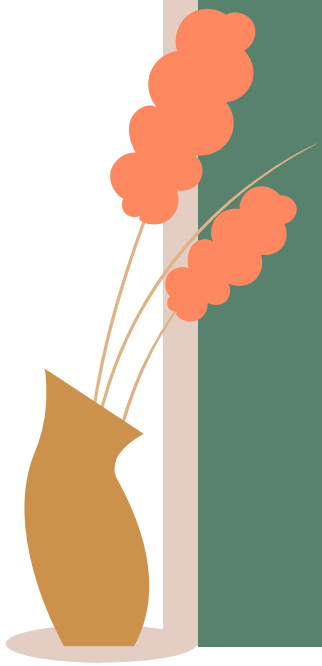
Wymagania Biznesowe

WO1. Wspomaganie nauczania podstaw komputera przy wykorzystaniu idei i technologii Learning Objects do tworzenia zestawów aktywnych ćwiczeń praktycznych.

WO2. Wypełnienie luki na rynku dla podobnego oprogramowania.

WO3. System przeznaczony jest na rynek.

WO4. Użytkownikami systemu będą adepci informatyki oraz osoby tworzące zestawy



Wymagania Dziedzinowe

D1. Wszystkie bazy danych powinny być dostępne przez interfejs użytkownika, którego podstawą jest standard Z39.50.

D2. Ze względu na ochronę praw autorskich system powinien udostępniać normy jedynie w formie do odczytu na ekranie monitora, bez możliwości pobrania pliku ani jego wydruku.

D3. D1. Tempo hamowania pociągu będzie wyznaczane ze wzoru: ...

Wymagania funkcjonalne

Definiują funkcję systemu lub jego składnik z dokładnie opisanym zachowaniem oraz wskazaniem wejścia i wyjścia.

Zachowania do wymagań mogą wynikać z reguł organizacyjnych, biznesowych lub zostać wypracowane z użytkownikami, klientami czy ekspertami w danej dziedzinie. Wiele wymagań może zostać odkrytych podczas opracowywania przypadku użycia (use case).

Przypadek użycia: UC01: Wyświetlenie ocen
Aktorzy: Student
Scenariusz Główny
1. Student wybiera opcję wyświetlenia semestrów. 2. System wyświetla listę dotychczasowych semestrów. 3. Student wybiera jeden z dostępnych semestrów. 4. System wyświetla listę przedmiotów wraz z ocenami.
Rozszerzenia
3.A Student chce wrócić do menu głównego. 3.A.1 Student wybiera opcję powrotu. 3.A.2 System wyświetla menu główne. 3.A.3 Koniec przypadku użycia

Przypadek użycia: UC02: Edytowanie protokołu
Aktorzy: Wykładowca
Scenariusz Główny
1. Wykładowca wybiera opcję dostępnych protokołów. 2. System wyświetla listę dostępnych protokołów. 3. Wykładowca wybiera opcję edycji danego protokołu. 4. System wyświetla dany protokół z listą studentów oraz wprowadzonymi ocenami. 5. Wykładowca zmienia żądane dane. 6. Wykładowca potwierdza wprowadzenie zmian. 7. System zapisuje zmiany. 8. System wyświetla informację o wprowadzonych zmianach.
Rozszerzenia
3.A Wykładowca chce wrócić do menu głównego. 3.A.1 Wykładowca wybiera opcję powrotu. 3.A.2 System wyświetla menu główne. 3.A.3 Koniec przypadku użycia. 6.A Wykładowca chce anulować wprowadzone zmiany. 6.A.1 Wykładowca wybiera opcję anulowania. 6.A.2 Powrót do kroku 2. 7.A System nie może zapisać wprowadzonych zmian. 7.A.1 System wyświetla informację o błędzie. 7.A.2 Powrót do kroku 4.

Wymaganie funkcjonalne

Przykład wymagań

- ❑ **Utworzenie protokołu (Wykładowca)**
- ❑ **Wypełnienie protokołu (Wykładowca, dodatkowo generuje powiadomienie dla Studenta)**
- ❑ **Edytowanie protokołu (Wykładowca, dodatkowo generuje powiadomienie dla Studenta)**
- ❑ **Wysłanie protokołu (Wykładowca)**
- ❑ **Zgłoszenie błędu (Student)**

Wymagania Niefunkcjonalne

Definiują potrzeby, aplikacji informatycznej dotyczące oczekiwań, jakie ona musi spełniać w zakresie jakości jej działania.

Wymagania niefunkcjonalne określają jak ma być zbudowany produkt oraz jakie kryteria jakościowe i ilościowe ma spełniać np. wydajność, dostępność, sposób wdrożenia, skalowalności, stabilności, bezpieczeństwo.

Wymaganie Niefunkcjonalne

Zawartość wymagań

- **Dopasowanie** – dokładność aplikacji.
- **Wydajność** – czas reakcji na zapytania; zużycie procesora, RAM, dysku.
- **Kompatybilność** – współpraca z innymi aplikacjami i systemami.
- **Użyteczność** – łatwość korzystania i szybkość uczenia się z aplikacji, estetyka
- **Niezawodność** – brak wad, czas naprawy wad i błędów, dostępność korzystania z wszystkich cech aplikacji wyrażona w procentach.
- **Bezpieczeństwo** – poufność i stopień zabezpieczenia danych w aplikacji.
- **Utrzymanie** – określenie wymagań minimalnych
- **Przenoszalność** – łatwość zmiany lokalizacji aplikacji, środowiska, eksportu danych.

Wymaganie Niefunkcjonalne

Problemy wymagań

- ❏ Dlaczego ładowanie strony z produktami trwa 3 sekundy? Skoro inne aplikacje otwierają się w ciągu 1,5 sekundy.
- ❏ Dlaczego z aplikacji może korzystać tylko 15 użytkowników jednocześnie? Skoro mamy 67 pracowników.
- ❏ Dlaczego aplikacja internetowa działa poprawnie tylko na Chrome? Przecież mamy w firmie również komputery Apple.
- ❏ Dlaczego aplikacja nie ma widgetów? Skoro wszystkie aplikacje w firmie korzystają z widgetów?

Wymagania Funkcjonalne

F1. Odtwarzanie kursu.

F2. Drukowanie.

F3. Nawigacja po slajdach.

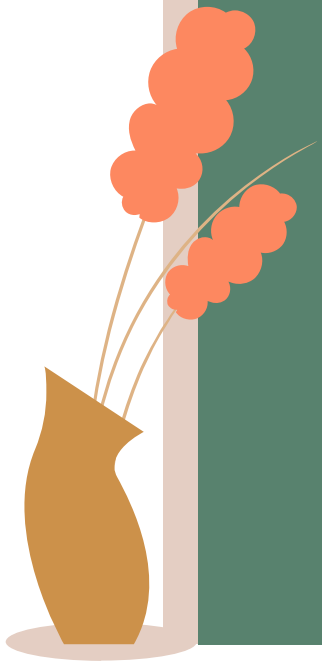
F4. Pauzowanie lekcji.

F5. Zamieszczanie filmów, dźwięków, animacji, pokazywania grafiki.

F6. Przeprowadzanie testów, quizów.

F7. Administrowanie użytkownikami.

F8. Dodawanie nowych kont.



Wymagania Niefunkcjonalne

N1. System powinien posiadać zabezpieczenia przed niepowołanym dostępem do funkcji administrowania użytkownikami

N2. System powinien płynnie odtwarzać lekcje

N3. System należy wdrożyć do końca semestru dyplomowego.

N4. System powinien działać pod systemami operacyjnymi Windows, Mac OS i Linux. N5. Należy wykorzystać język programowania Java

02.

Przypadki testowe

Przyczyny i źródła testów

Przypadek testowy

Zbiór danych pozwalających na przetestowanie danej funkcjonalności..

Przypadek służy do opisywania danych wejściowych, wstępnych warunków wykonania, oczekiwanych rezultatów i końcowych warunków wykonania i opracowuje się go w określonym celu lub dla warunku testowego. Pozwala na zweryfikowanie zgodności z konkretnym wymaganiem.

Po co to komu?

Przypadki testowe gwarantują nam:

Przejrzystość możliwości

Obsłużenie każdej dostępnej funkcjonalności

Pewność kompletności

Nie pominięcie niczego na etapie testowania

Bazę do raportów

Potwierdzenie działania systemu np. w testach akceptacyjnych

Elementy przypadku testowego

- ID
- Tytuł
- Warunki wstępne
- Kroki do wykonania
- Oczekiwany rezultat
- Warunek końcowy

Dodatkowo:

- Cel
- Środowisko
- Dane testowe
- Status

PRZYPADEK NISKOPOZIOMOWY 1

Tytuł: Logowanie do aplikacji za pomocą konta użytkownika.

Cel: Weryfikacja możliwości zalogowania do aplikacji za pomocą poprawnych danych.

Warunki początkowe:

– W aplikacji istnieje aktywne konto użytkownika.

– Użytkownik znajduje się na ekranie logowania do aplikacji.

KROK	REZULTAT
1. Wprowadź poprawne dane w pole Login i Hasło	1. Pola zostały uzupełnione.
2. Naciśnij przycisk Zaloguj	2. Użytkownik został zalogowany do aplikacji.

Priorytet: Wysoki

Wykonanie: Manualne

Estymowany czas: Minuta

PK-019: Patcher file fixing

Requirements: Patcher should re-download a broken file before game can be played.

Details

If any installed game file accidentally gets deleted or changed, patcher should detect that fact and re-download that file without the need of re-downloading the whole game.

To test it out:

1. Install the game
2. Close the patcher
3. Remove or replace any random file inside the game's directory
4. Start the launcher
5. See that patcher is re-downloading some data to fix that file

Objective: Fix a broken game.

- Wymagania bazowe
  - PK-001: User Registration
    - Details
  - PK-028: User should be able to register an account from the landing page
    - Details
  - PK-029: Password resetting
  - PK-023: Data Transfer Usage indicator
    - Details
  - PK-024: User is allowed to upgrade account
    - Details
- Obsługa interfejsu webowego
  - PK-002: Tworzenie nowej aplikacji

Jakie są cele przypadków?

1. Jakość

Powinny stanowić miarę jakości testowanego oprogramowania;

2. Bugi

Powinny być wykonywalne i nastawione na znalezienie defektów;

3. Założenia

Powinny oceniać zgodność z dokumentacją i weryfikować poprawność systemu;

4. Wsparcie

Powinny minimalizować koszty wsparcia technicznego

5. Ocena

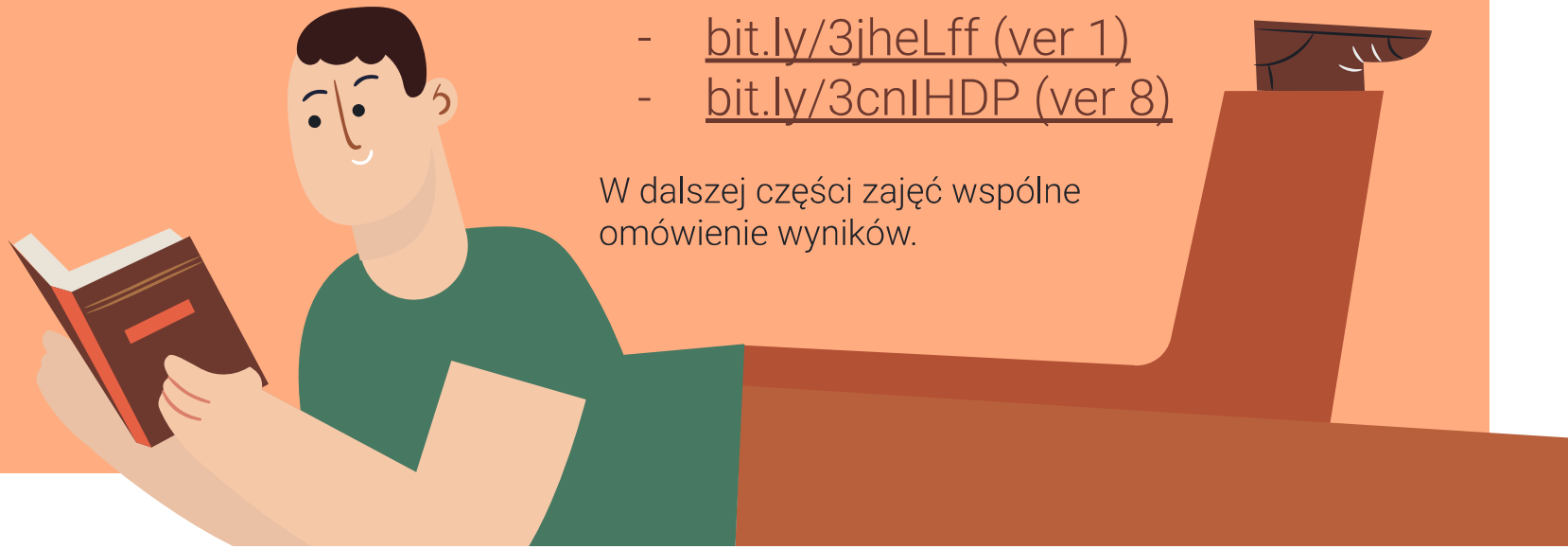
Powinny dostarczać informacje wspomagające decyzję o wprowadzeniu systemu na produkcję

Zadanie praktyczne

Myszka w dłoń!
i zapraszam do testowania gry oraz sporządzania listy przypadków:

- bit.ly/3jheLff (ver 1)
- bit.ly/3cnIHDP (ver 8)

W dalszej części zajęć wspólne omówienie wyników.



Dzięki za uwagę!

Więcej materiałów znajdziesz na
kucharczyk.dev

Kontakt
pkucharczyk@technikumkreatywne.pl

Materiały i inspiracje

Prezentacja powstała w oparciu o serwisy:

TestNaTestera.pl
Testuj.pl
Testerzy.pl
Kodilla.pl
ToNieBug.pl

...oraz własną wiedzę i doświadczenie ;)