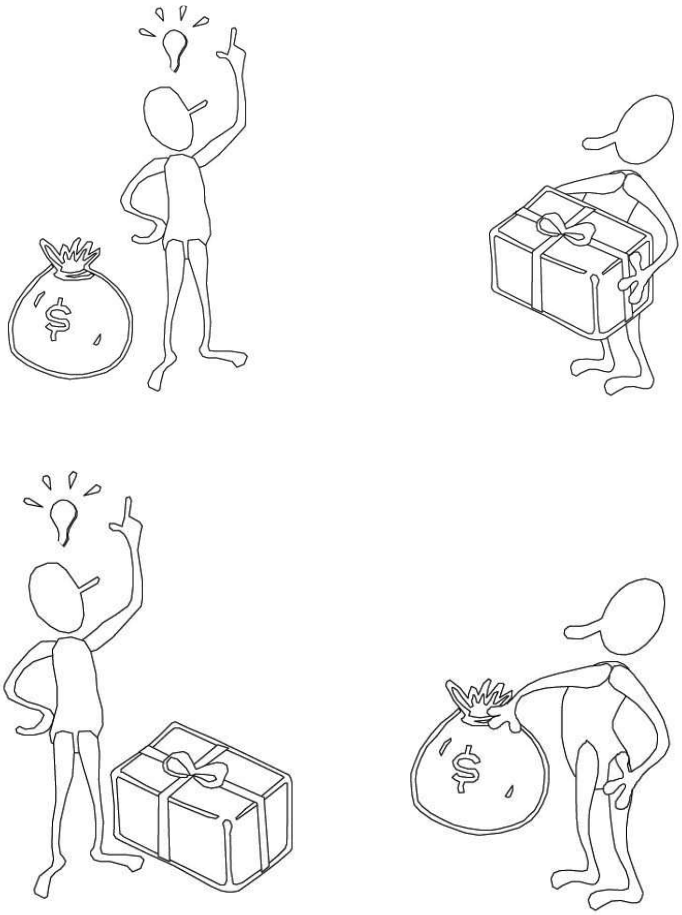
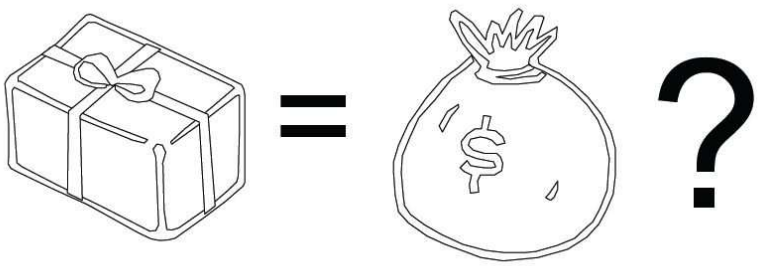
Wprowadzenie do metodologii tworzenia specyfikacji wymagań

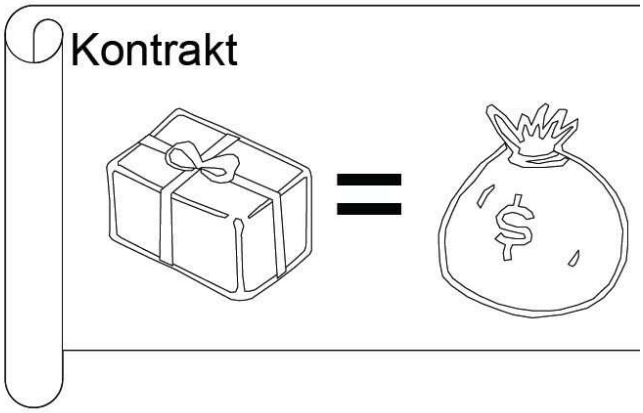
W celu lepszego zrozumienia, czym są wymagania w projektach informatycznych załóżmy, że właściciel restauracji dostrzega potrzebę wdrożenia systemu informatycznego, w celu usprawnienia obsługi kelnerskiej klientów. Właściciel ten zgłasza się do firmy informatycznej i pragnie zamówić taki system.



Dochodzi do transakcji pomiędzy klientem, posiadającym pieniądze, oraz firmą informatyczną, będącą w stanie spełnić oczekiwania klienta.



Z punktu widzenia takiej transakcji ważne jest, aby była równoważność pomiędzy tym ile klient zapłaci, tym co otrzyma w zamian.



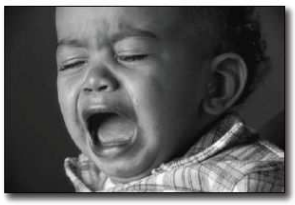
Główną częścią kontraktu jest dokument zwany specyfikacją wymagań. Co oznacza ten termin?

Wymagania

- Wymaganie = opis *co* system powinien robić
  źródło: www.wikipedia.org
- Specyfikacja = zbiór wymagań
- **W momencie kiedy spiszemy wymagania, klient dostanie dokładnie to, czego potrzebuje**

Rozbijmy go na poszczególne wyrazy…
Jedno wymaganie, to opis pojedynczej funkcji, którą system powinien udostępniać swoim użytkownikom. Specyfikacja natomiast jest zbiorem wymagań, czyli zakresem funkcjonalności zamawianego systemu. Mogłoby się więc wydawać, że jeżeli spiszemy wymagania na początku projektu informatycznego (co nie jest powszechną praktyką), mamy zapewnienie, że klient dostanie dokładnie to, czego potrzebuje (i że nie będzie chciał od nas więcej!).

~~W momencie kiedy spiszemy wymagania, klient dostanie dokładnie to, czego potrzebuje~~
Słowa nie mają znaczenia!



Niestety nie jest to takie proste, przynajmniej z kilku powodów. Po pierwsze, słowa nie mają znaczenia!

Lingwistyka filozoficzna mówi, że to ludzie umówili się, iż krowa to krowa, dziecko to dziecko, a wiatrak to wiatrak. Równie dobrze można by się umówić, że krowę będziemy nazywać wiatrakiem, a wiatrak krową.

Może to rodzić problemy związane z nazewnictwem, terminologią. W momencie kiedy przystępujemy do informatyzacji pewnego przedsiębiorstwa, z pewnością spotkamy się z szeregiem terminów, których nie zrozumiemy. Np. w firmach spedycyjnych powszechne są określenia: załadunek kompletny, raport miesięczny, SAD.
Co one znaczą?
Czy załadunek kompletny, to samochód, który zapakowano do określonej wagi? Czy też może samochód, którego ładunek zajmuje pewną określoną objętość minimalną? A może też samochód załadowany wszystkimi towarami uwzględnionymi w zleceniu?
Co oznacza termin raport miesięczny? Czy to podsumowanie faktur z całego miesiąca? A może liczba transportów wraz z informacją o sumarycznym załadunku? A może jeszcze coś innego? SAD to kawałek ziemi obsadzony drzewami owocowymi, czy też może dokument celny?

2. Wiedza:
– świadoma
– nieświadoma

Stwierdzenie nasze nie jest poprawne również z drugiego powodu: wiedza każdej osoby (a więc również klienta) składa się z części świadomej oraz nieświadomej. Może się więc zdarzyć (i jest tak dosyć często), że klient nie jest w pełni świadomy każdego wymagania, więc nie jest w stanie wszystkiego przekazać analitykowi. System zbudowany na podstawie takich niekompletnych wymagań na pewno nie spełni do końca jego potrzeb.

Problemy te najlepiej pokazać na przykładzie. Spójrzmy jak mogłyby wyglądać wymagania telefonu komórkowego.

Wymagania telefonu Nokia N80:
– ***duży*** wyświetlacz
– ***duże***, wygodne klawisze
– aparat o ***wysokiej*** rozdzielczości
– WiFi

Co oznacza Określenie „duże”?
Co oznacza określenie „wysoka”
Czy klawiatura nie może się znaleźć po drugiej stronie?

Co oznacza określenie: duży wyświetlacz? Czy wyświetlacz wystarczający do odczytywania wiadomości tekstowych, a może przeglądania zdjęć w dużej rozdzielczości? Podobnie z pozostałymi określeniami: duże klawisze, wysoka rozdzielczość aparatu. Ciekawe spostrzeżenie możemy wysnuć w związku z wiedzą nieświadomą. Możemy być pewni, że nigdzie w wymaganiach żadnego telefonu komórkowego nie znajdziemy określenia, że klawiatura powinna być po tej samej stronie telefonu co wyświetlacz. Dlaczego więc nikt nie zrobi na odwrót: przecież gdybyśmy ułożyli klawiaturę po jednej stronie, a wyświetlacz po drugiej - bylibyśmy w stanie jeszcze bardziej zmniejszyć rozmiary telefonu. Jest to wiedza nieświadoma - każdy z nas korzystał z telefonu i wie że podczas naciskania klawiszy musi

widzieć, co się dzieje na ekranie. Gdyby jednak te produkty były projektowane przez osoby, które nigdy nie miały do czynienia z czymkolwiek podobnym (tak jest w dużej części projektów informatycznych), wtedy z pewnością znalazłoby się wiele rzeczy zrobionych wbrew intuicji przyszłych użytkowników.

Spisywanie wymagań to sztuka

Nie ma uniwersalnego sposobu,
Korzystaj z porad innych:

– dobre praktyki
– metody, które się sprawdziły

No dobrze. Widzimy, że na inżynierów wymagań czyha wiele niebezpieczeństw. Czy jest jakiś uniwersalny sposób poradzenia sobie z nimi? Niestety nie. Pozyskiwanie wymagań jest pewnego rodzaju sztuką. Można jednak patrzeć, jak to robią inni oraz spróbować pewnych „dobrych praktyk" zaproponowanych przez ekspertów w tej dziedzinie.

W celu lepszego ogarnięcia wymagań systemów informatycznych, specjaliści podzielili je na dwie kategorie: na wymagania funkcjonalne i poza-funkcjonalne. Warto również skorzystać do doświadczenia innych przez wykorzystanie dobrych praktyk (Sommerville i Sawyer'a).

Wymagania funkcjonalne:
– Wprowadzanie nowej faktury
– Generowanie raportu miesięcznego

Wymagania poza-funkcjonalne:
– minimum 20 faktur na godzinę
– 200000h MTBF
– maksimum 2 godziny potrzebne na przeszkolenie 1 pracownika

Wymagania funkcjonalne to funkcje systemu widziane od strony użytkownika, np. wprowadzenie nowej faktury, lub wygenerowanie raportu miesięcznego. Natomiast wymagania poza-funkcjonalne to wszystkie inne wymagania, zwłaszcza takie związane z bezpieczeństwem, wydajnością, awaryjnością, np: system ma umożliwiać wprowadzenie co najmniej 20 faktur w ciągu godziny, lub średni czas pomiędzy awariami (ang. MTBF) powinien wynosić 200000h.

Podział FURPS

• **F**unctionality - funkcjonalność

• **U**sability - użyteczność
• **R**eliability - niezawodność
• **P**erformance - wydajność
• **S**ecurity – bezpieczeństwo

Bardzo powszechny jest również podział FURPS. Pierwsza litera oznacza wymagania funkcjonalne, natomiast pozostałe - wymagania poza-funkcjonalne.

U - użyteczność, oznacza łatwość użytkowania systemu. Takie wymagania można precyzować np. poprzez maksymalny czas szkolenia pracownika, liczba kontaktów ze wsparciem klienta, liczbą sytuacji, w których konieczne jest skorzystanie z systemu pomocy.
R - niezawodność, może być mierzona poprzez: średnią liczbę godzin pracy bez awarii (ang. MTBF - Mean Time Between Failure), maksymalną liczbą godzin w miesiącu w ciągu których system może być wyłączony w celach pielęgnacyjnych (ang. maintainance) - ma to znaczenie szczególnie w przypadku systemów, które muszą pracować na okrągło - np. systemy bankowości elektronicznej, P - wydajność - mierzona np. liczbą transakcji, którą system jest w stanie obsłużyć w ciągu godziny, liczbą użytkowników, którzy mogą być zalogowani jednocześnie do portalu, S - bezpieczeństwo, to wymagania związane z szyfrowaniem, polityką praw, itp.

Wymagania funkcjonalne:

Z punktu widzenia klienta dużo ciekawsze wydają się wymagania funkcjonalne od poza-funkcjonalnych.

Wymagania funkcjonalne opisują funkcje poszczególnych użytkowników. Istnieją trzy podejścia. Pierwsze dwa są podejściami historycznymi, natomiast ostatni - przypadki użycia staje się obecnie coraz bardziej popularny.

- „System powinien…”
- Funkcje systemu
- Przypadki użycia

„System powinien…”

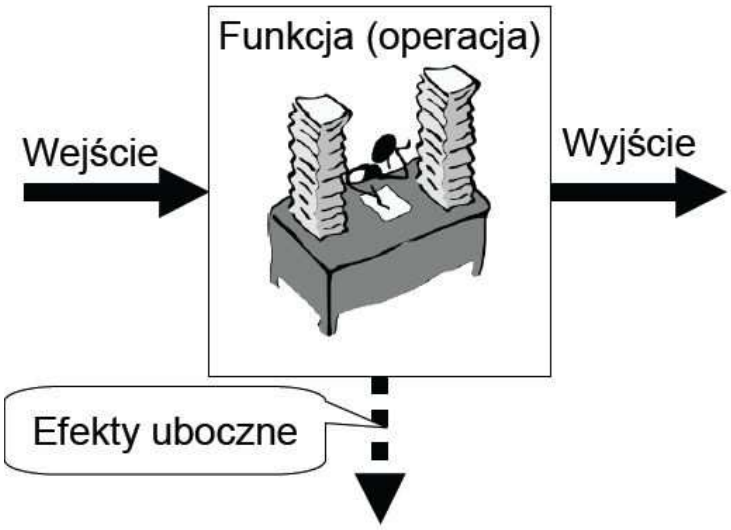
• Zalety:
– łatwość spisywania
• Wady:
– słaba czytelność
– trudne sprawdzanie kompletności, spójności

Przykład:
– System powinien umożliwić wystawianie faktur
– System powinien generować zestawienie miesięczne faktur
– Faktura powinna zawierać co najmniej jedną pozycję
…
(tak przez kolejnych 200 stron)

Wymagania w stylu „System powinien…” były intensywnie wykorzystywane w latach 80-tych, 90-tych. W roku 1998 nawet zostały zaproponowane jako standard (IEEE 830-1998). Ich dużą zaletą jest łatwość spisywania.

Niestety obecne systemy stają się coraz bardziej złożone, a wymagania średniej wielkości systemu w tej postaci zajęłyby kilkaset stron. Tak wielka liczba luźnych zdań, niepowiązanych ze sobą sprawia trudności podczas sprawdzania jakości takiej specyfikacji.

Funkcje systemu



• Wady:
– słaba czytelność
– trudne do zrozumienia

Innym podejściem jest opisywanie poszczególnych funkcji systemu. Analogicznie do funkcji matematycznych, każda funkcja systemu informatycznego ma swoje wejście, wyjście, efekty uboczne.
Przykładowo, rozpatrując funkcję wystawiania faktury, wejściem mogą być pozycje faktury, wyjściem wydruk faktury (lub wysłanie jej faksem), natomiast efektem ubocznym zapisanie tej faktury w rejestrze faktur.
Podobnie jak w przypadku wymagań typu „System powinien”, taka specyfikacja wymagań zawiera bardzo dużą liczbę małych funkcji, więc nadal są problemy ze zrozumieniem idei systemu i czytelnością.

Przypadki użycia

UC1: Wystawianie faktury
Główny scenariusz:

1. Sprzedawca pragnie wystawić fakturę.
2. Sprzedawca wpisuje pozycje faktury.
3. System podlicza fakturę, nadaje jej nowy numer i zapisuje w rejestrze
4. Sprzedawca drukuje fakturę.

- Zalety:

–łatwość spisywania
–czytelność
–łatwość zrozumienia i wyobrażenia sobie przyszłego systemu

Trzecie podejście to przypadki użycia. Poprzednie podejścia opisywały wymagania z perspektywy systemu - jak system powinien się zachować w określonej sytuacji. Przypadki użycia natomiast skupiają się na interakcji pomiędzy użytkownikiem, a systemem. Dzięki takiemu podejściu zyskujemy na czytelności - przypadki użycia nie pokazują zbędnych szczegółów. Dużo łatwiej też jest klientowi wyobrazić sobie jak system będzie funkcjonował. Nie czyta opisu matematycznego, lecz pewną opowieść, która mówi o tym, w jaki sposób np. wystawić fakturę.

Przypadek użycia

Forma strukturalizowana

Nazwa, np. **Wystawianie faktury**

Jak już było powiedziane wcześniej - przypadek użycia jest opisem interakcji pomiędzy użytkownikiem, a systemem informatycznym. Mogą one występować w różnych formach. Najprostsza to akapit tekstu pisany językiem naturalnym. Bardziej powszechna jest jednak postać strukturalna. Każdy przypadek użycia w tej formie składa się z następujących elementów:
Nazwa - wyraża cel przypadku użycia (np. wystawianie faktury, zamówienie książki, dodanie wpisu na forum)

Identyfikator

UC1: Wystawianie faktury

Identyfikator - unikalny w obrębie danej specyfikacji wymagań, przydaje się podczas dyskusji na temat wymagań - do odwoływania się do poszczególnych elementów. Dzięki nim można prosto powiedzieć, że np. w przypadku użycia UC1, w kroku 2 jest błąd.

Główny scenariusz

UC1: Wystawianie faktury
Główny scenariusz:

1. Sprzedawca pragnie wystawić fakturę.
2. Sprzedawca wpisuje pozycje faktury.
3. System podlicza fakturę, nadaje jej nowy numer i zapisuje w rejestrze
4. Sprzedawca drukuje fakturę.

Główny scenariusz - sekwencja kroków, która musi zostać wykonana do osiągnięcia celu przypadku użycia (wyrażonego w nazwie). Opisuje najczęstszy przebieg interakcji pomiędzy głównym aktorem, a systemem.

Rozszerzenia

UC1: Wystawianie faktury
Główny scenariusz:

1. Sprzedawca pragnie wystawić fakturę.
2. Sprzedawca wpisuje pozycje faktury.
3. System podlicza fakturę, nadaje jej nowy numer i zapisuje w rejestrze.
4. Sprzedawca drukuje fakturę.

Rozszerzenia:

3.A. Sprzedawca nie dodał żadnej pozycji
3.A.1. System prosi o ponowne wprowadzenie pozycji (powrót do 2.)

Nie zawsze jest możliwość wykonania głównego scenariusza. Czasem użytkownik popełni jakiś błąd, innym razem nie są spełnione pewne warunki, co uniemożliwia przejście dalej. Reakcje na takie wydarzenia możemy umieścić w rozszerzeniach. Przykładowo w kroku 3. może się okazać, że sprzedawca nie dodał żadnej pozycji. Dzięki rozszerzeniom możemy podać sekwencję kroków, która będzie wykonana w takiej sytuacji.

Atrybuty

UC1: Wystawianie faktury
Atrybuty:
Główny aktor: Użytkownik
Priorytet: Wysoki
Źródło: Jarosław Szkoła
Główny scenariusz:

1. Sprzedawca pragnie wystawić fakturę.
2. Sprzedawca wpisuje pozycje faktury.
3. System podlicza fakturę, nadaje jej nowy numer i zapisuje w rejestrze.
4. Sprzedawca drukuje fakturę.

Rozszerzenia: …

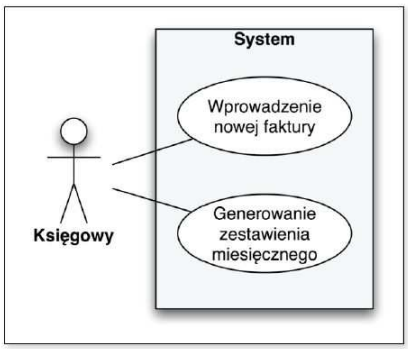
Dodatkowo, do każdego wymagania możemy dołączyć listę atrybutów, np.

- nazwę głównego aktora - użytkownika, który gra główną rolę w danym przypadku użycia,
- priorytet tego wymagania z punktu widzenia klienta,
- źródło wymagania - osoba, od której pochodzi konkretne wymaganie - taka informacja przydaje się później, kiedy programiści będą mieli jakieś wątpliwości i dodatkowe pytania - będą w stanie trafić bezpośrednio do osoby najlepiej poinformowanej.

Przypadki użycia, a UML

Diagram przypadków użycia:

– Nazwy przypadków użycia
– Powiązania z aktorami
– Dobre jako „**mapa**”



Przypadki użycia są często mylone z diagramami przypadków użycia z UML’a. Diagram przypadków użycia prezentuje tylko połączenie aktorów z przypadkami użycia. Każdy przypadek użycia jest tutaj reprezentowany jako nazwa. Jest to dobre w początkowej fazie analizy wymagań, kiedy odkrywamy funkcje systemu, lecz wymaga późniejszego doprecyzowania, czyli napisania specyfikacji wymagań, która będzie zawierać kompletne przypadki użycia. W takiej specyfikacji wymagań dobrze jest umieścić na początku diagram przypadków użycia, który będzie służył jako „spis treści” dokumentu.

Jak pisać przypadki użycia?

Jedną z lepszych książek mówiących o tym są: „Wzorce Efektywnych Przypadków Użycia" [Steve Adolph, Paul Bramble: „Patterns for Effective Use Cases"], napisana przez ludzi stosujących to podejście od lat w projektach informatycznych i mających duże doświadczenie w tym zakresie.

- Przypadek użycia
- Zbiór przypadków użycia
- Proces
- Zespół

Stworzyli oni listę „wzorców", czyli „dobrych praktyk" dotyczących pisania i podzielili ją na 4 grupy:

-pierwsza dotyczy pojedynczego przypadku użycia,
-druga - całego ich zbioru, czyli sposobu komponowania specyfikacji z pojedynczych wymagań
-trzecia - dotyczy procesu ich tworzenia, oraz
-czwarta - zespołu osób, które przygotowują specyfikację wymagań

Pojedynczy przypadek użycia – założenia

• „**Fraza czasownikowa w nazwie**":
– Wystawianie faktury
– Generowanie raportu miesięcznego
– ~~Główny przypadek użycia~~
– ~~Przypadek użycia 2~~
– ~~Zarządzanie~~

Nazwa przypadku użycia jest bardzo ważna - występuje w wielu miejscach dokumentu (spisie treści, diagramach przypadków użycia, itp), tak więc w sformułowanie prawidłowej nazwy powinna być włożona odrobina wysiłku, tak aby dobrze odzwierciedlała ona zawartość przypadku użycia.
Mówi o tym pierwszy wzorzec z książki: „Fraza czasownikowa w nazwie". Frazy czasownikowe dobrze opisują cele przypadków użycia, np jak spotkamy nazwę: wystawianie faktury, albo generowanie raportu miesięcznego, to od razu wiemy o jakie wymaganie chodzi.

Kilka przykładów złych nazw:

-Nazwy nic nie znaczące: Główny przypadek użycia, Przypadek użycia 2
-Zbyt ogólna, przez co nie stanowi żadnej wartości dla czytelnika: Zarządzanie

Scenariusz i rozszerzenia

Nadrzędny cel: czytelność!

- Scenariusz główny – najbardziej prawdopodobna ścieżka 3-9 kroków,
- Rozszerzenia – alternatywne scenariusze
    - kiedy coś pójdzie nie tak
    - numer rozszerzenia: numer kroku + kolejna litera

Drugi omawiany wzorzec to „Scenariusz i rozszerzenia" Przypadek użycia powinien być podzielony na takie 2 części - dzięki temu czytelnik może się zapoznać najpierw z głównym scenariuszem i zrozumieć na czym polega wymaganie, a następnie przejść do niuansów zawartych w rozszerzeniach. Generalną zasadą jest zwięzłość i przejrzystość, aby czytelnik się nie pogubił. Dlatego, każdy scenariusz powinien zawierać od 3-9 kroków (gdy jest ich mniej, specyfikacja wymagań jest zbyt fragmentaryczna, natomiast gdy jest ich więcej - czytelnik nie jest w stanie ogarnąć pamięcią całości). Ze względu na czytelność, również poszczególne kroki scenariusza nie powinny być zbyt skomplikowane. Najlepiej gdy są wyrażone prostym zdaniem zawierającym podmiot (czyli aktora). Rozszerzenia natomiast, to sekwencje kroków wykonywane podczas sytuacji wyjątkowych. Numer rozszerzenia składa się zawsze z numeru kroku, którego rozszerzenie dotyczy, oraz litery – stanowiącej numer rozszerzenia (w ramach tego kroku).

Czyli przykładowo możemy się spotkać z takimi rozszerzeniami:
1.A.
2.A.
3.A.
1.B.

Rozszerzenia mogą być wielokrotnie zagnieżdżane, czyli przykładowo, jeżeli mamy krok 1. A.2 i chcemy do niego dodać rozszerzenie, to będzie ono miało numer 1.A.2.A.

„Obojętność technologiczna":

- technologia jest zmienna
- niepotrzebne ograniczenia
- szczegóły GUI zaciemniają obraz
- klient nie rozumie terminy technicznych

Przykłady:
~~– Pracownik **klika** na **link** kalendarium~~
~~– System zapisuje dane użytkownika w **bazie danych**~~
~~– System za pomocą **protokołu SOAP** pobiera aktualną temperaturę~~

Błędem jest, gdy przypadki użycia zawierają szczegóły technologiczne:

- technologia jest zmienna - może sie okazać, że następny podobny projekt będzie realizowany w nowej technologii, mimo że część wymagań będzie identyczna. Jeżeli

przypadki użycia będą zawierać szczegóły technologiczne, to ponowne zastosowanie raz napisanych wymagań będzie dużo trudniejsze

- szczegóły Graficznego Interfejsu Użytkownika (ang. GUI - Graphical User Interface) zaciemniają jedynie obraz czytelnika - gubi się on w gąszczu szczegółów. Czasem jednak jest potrzeba zobrazowania klientowi takich szczegółów - wtedy warto naszkicować poszczególne ekrany aplikacji i dodać je jako załącznik do specyfikacji wymagań,
- klient nie rozumie terminów technicznych w stylu SOAP, baza danych, HTTP, itp.

Zbiór przypadków użycia

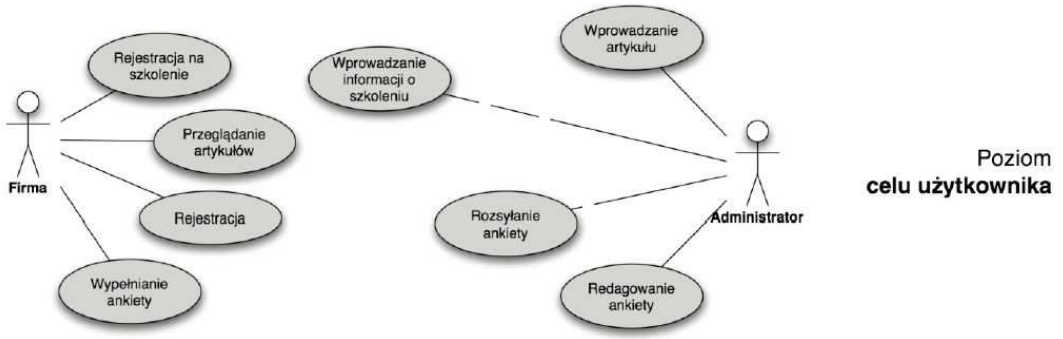
Kiedy potrafimy napisać pojedynczy przypadek użycia, czas dowiedzieć się, w jaki sposób składać te elementy w całą specyfikację wymagań.

• „**Rozwijalna historia**”:
–Hierarchiczne scenariusze
–Można rozwijać lub zwijać w celu pokazania lub ukrycia szczegółów

Pierwszy wzorzec z tej grupy to „Rozwijalna historia”

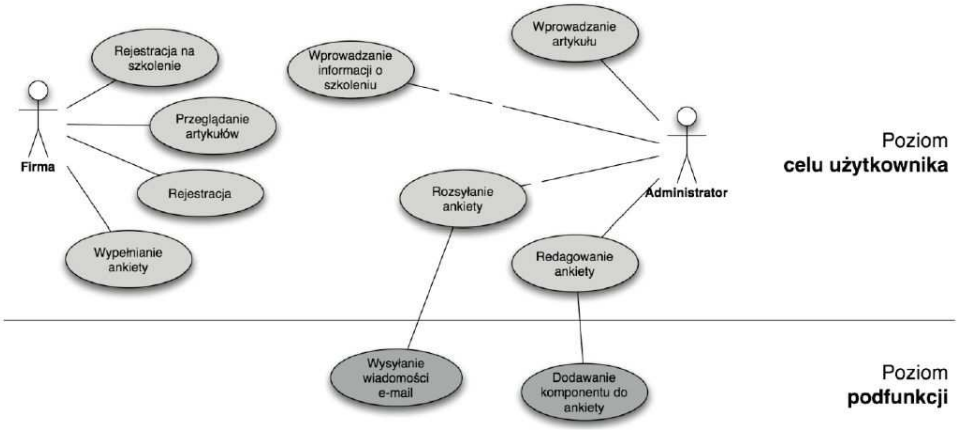
Zbiór przypadków użycia powinien stanowić rozwijalną historię, czyli być zorganizowany w hierarchiczne drzewo scenariuszy. Im głębiej, tym przypadki użycia są bardziej szczegółowe. Dzięki temu będzie można przeczytać tylko przypadki użycia najwyższego poziomu, aby mieć ogólną wiedzę o systemie, lub zagłębiać się coraz bardziej jeżeli potrzebujemy większych szczegółów.

Poziom celu użytkownika:
– główny aktor osiąga zamierzony cel w ciągu jednej sesji przy komputerze



W celu łatwiejszego zorientowania się w tej hierarchii scenariuszy, wprowadzono podział przypadków użycia na trzy poziomy. Środkowy poziom - to poziom celu użytkownika. Przypadki użycia na tym poziomie opisują poszczególne funkcje systemu, z punktu widzenia użytkowników, np. Rejestracja na szkolenie, Wypełnienie ankiety, Redagowanie ankiety…

Poziom podfunkcji:
– precyzuje wykonanie funkcji

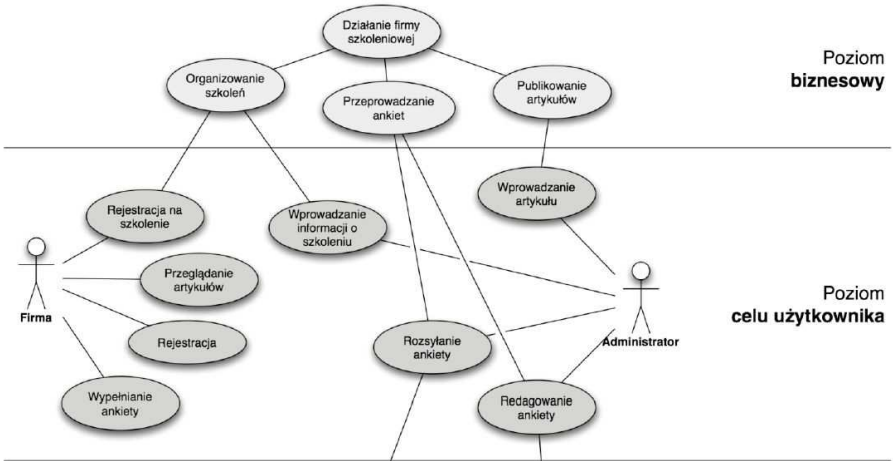


Niekiedy do opisania pewnych funkcji potrzebujemy większych szczegółów. Wtedy korzystamy z przypadków użycia poziomu podfunkcji. Jeżeli np. uważamy, że „Dodawanie komponentu do ankiety", lub „Wysyłanie wiadomości email" wymaga doprecyzowania, możemy stworzyć taki przypadek użycia i wywołać go z przypadku użycia wyższego poziomu. Przykładowo, mamy dane 2 przypadki użycia:

UC1. Rozsyłanie ankiety, oraz
SUB1. Wysyłanie wiadomości email.
Wtedy z UC1, możemy wywołać drugi przypadek użycia, poprzez wymienienie tego przypadku użycia w treści jednego z kroków:
UC1. Rozsyłanie ankiety
….
3. System wysyła prośbę o wypełnienie

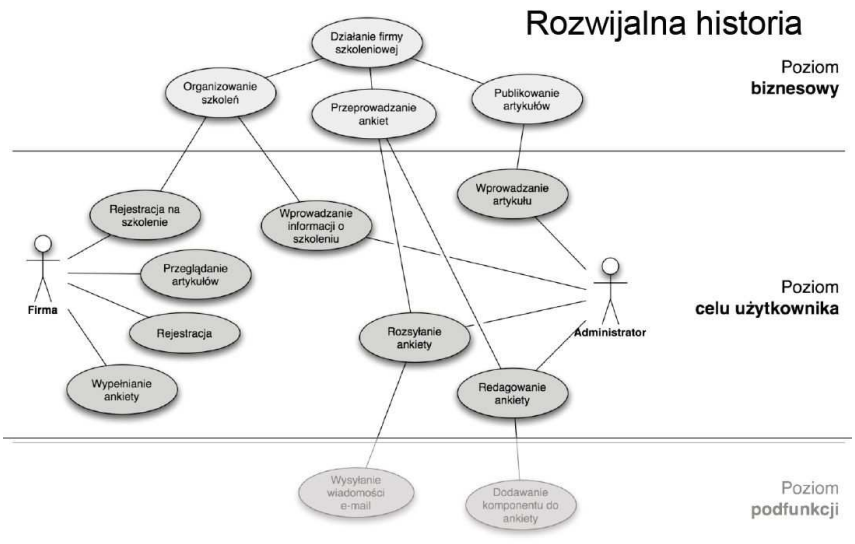
To jest znak, że jeżeli interesują nas szczegóły danego kroku, może zajrzeć do przypadku użycia SUB1

Poziom streszczenia (biznesowy):
– kontekst dla wymagań użytkownika



Trzeci poziom, to poziom biznesowy. Służy on do naszkicowania kontekstu dla tworzonego systemu. W momencie kiedy analityk pozna specyfikę konkretnej firmy, jej procesy biznesowe, dużo prościej zrozumieć wymagania użytkownika. W celu opisania tych procesów można skorzystać z przypadków użycia poziomu streszczenia (biznesowego).

Rozwijalna historia



Na rysunku widać, w jakiej kolejności czytać przypadki użycia, aby maksymalnie szybko zrozumieć wymagania systemu. W dowolnym momencie można przerwać czytanie, jeżeli nie potrzebujemy większej liczby szczegółów.

Proces

Trzecia grupa - wzorce dotyczące procesu powstawania przypadków użycia. Mówią one, w jaki sposób spisywać przypadki użycia, aby zrobić to najefektywniej

„Szerokość przed głębokością”:

pozyskiwanie wymagań - odkrywczy proces

– zmiany na tym etapie - bardzo prawdopodobne
– pisanie kompletnych przypadków użycia - strata energii
– rozwijaj w kolejności:

1. lista aktorów
2. nazwy przypadków użycia
3. główny scenariusz
4. Rozszerzenia

Pozyskiwanie wymagań jest procesem odkrywczym. Nie ma osób, które są w stanie od razu zaproponować w pełni poprawne wymagania. Często pytamy klienta, jak wyobraża sobie poszczególne funkcje systemu, następnie zapisujemy to w formie przypadków użycia, po czym okazuje się, że chodziło o coś zupełnie innego.
Dlatego warto robić to stopniowo, na początku bardzo zgrubnie, a w momencie kiedy upewnimy się, że nasz tor myślenia jest prawidłowy, można dodawać szczegóły.
Takie podejście właśnie oznacza określenie „Szerokość przed głębokością”

Dla przypadków użycia proponowane są następujące etapy:

1. odkrycie wszystkich aktorów
2. zaprezentowanie funkcji - nazwy przypadków użycia
3. dopisanie głównego scenariusza do każdej nazwy przypadku użycia
4. uzupełnienie rozszerzeń

Lista Aktor-Cel:

Firma	Rejestracja na szkolenie
	Pobieranie artykułu
	Zgłoszenie do udziału w projekcie
Konsultant	Akceptuje zgłoszenia nowych firm
	Zarządza artykułami w portalu
	Przeprowadza ankiety

Pierwsze dwa kroki (aktorzy i nazwy przypadków użycia) można przedstawić na diagramie przypadków użycia, lub prościej - w formie tabeli Aktor-Cel, której przykład jest zaprezentowany powyżej.

Zespół

Ostatnia grupa wzorców, to wzorce dotyczące zespołu autorów przypadków użycia.

„Mały zespół autorów”

- wielkość zespołu to **najważniejszy czynnik wpływający na jakość,**
- **2-3** osoby w zupełności **wystarczają,**
- zaangażuj **więcej osób w proces recenzji,**
- duże systemy – **kilka małych zespołów** z **jednym architektem** odpowiedzialnym za spójną wizję systemu,

Kluczowym czynnikiem stanowiącym o jakości specyfikacji wymagań jest wielkość zespołu analityków. Z praktyki wynika, iż 2-3 osoby są w zupełności wystarczające - o tym mówi ten wzorzec. Jeżeli będzie więcej piszących, to narzut związany z komunikacją między nimi będzie zbyt duży, a kompromisy trudne do osiągnięcia. Więcej osób można zaangażować na etapie recenzji - wtedy inne osoby (testerzy, użytkownicy) będą w stanie wyrazić swoje zdanie. W trakcie pracy nad ogromnymi systemami może się okazać, że 2-3 osoby nie są w stanie ogarnąć całości. Wtedy warto taki system podzielić na moduły i powołać po jednym małym zespole do analizowania wymagań tylko w ramach modułu.

„Zrównoważony zespół”

- grupa podobnych specjalistów skupi się jedynie na **ograniczonych problemach,**
- synergia: kompensuj **słabe** strony jednych, **dobrymi** stronami innych
- połącz ludzi **różnej specjalności,**
- analitycy i **użytkownicy**

Drugim kluczowym czynnikiem, jeżeli chodzi o zespół analityków, jest różnorodność specjalistów.
W dzisiejszych czasach rzadko kto ma tak obszerne doświadczenie, że zna się jednocześnie na tworzeniu oprogramowania i dziedzinie, którą informatyzujemy. W związku z tym warto na etapie powstawania wymagań połączyć siły specjalistów z różnych dziedzin. Autorzy wzorców nazwali to „Zrównoważonym zespołem”. Tak skomponowany zespół będzie w stanie dostrzec dużo wcześniej wiele problemów i szybciej stworzyć poprawne wymagania.

Często popełniane błędy

UC1: Faktura
Główny scenariusz:
1. Sprzedawca wpisuje kod dostępu.
2. System weryfikuje użytkownika.
3. Kliknięcie na przycisk wystawiania faktury.
4. System prezentuje formularz.
5. Wpisanie pozycji w dolnym okienku.
6. Wpisanie wartości pozycji , stawki VAT, liczby pozycji i nr. porządkowego.
7. System podlicza fakturę i prezentuje sumę.
8. System nadaje nowy numer i zapisuje w rejestrze faktur.
9. Wydruk faktury.
10. Jeżeli wystawianie faktur zakończyło się, to użytkownik się wyloguje.

Rozszerzenia:
3.A. Sprzedawca nie dodał żadnej pozycji
3.A.1. System prosi o ponowne wprowadzenie pozycji (powrót do 2.)

Spróbujmy znaleźć naruszenia podanych wcześniej zasad na przykładzie UC1: Faktura.

Najważniejsze błędy to:

- tytuł przypadku użycia nie mówi, czym przypadek użycia będzie się zajmował (nie wiemy czy to jest wystawienie faktury, wysłanie, przefaksowanie, odbiór, czy cokolwiek innego)
- W krokach 3, 5, 6, 7 znajdują się zbędne szczegóły (część to szczegóły techniczne, np. informacje o GUI, wyliczenia pól na ekranie, itp.)
- W krokach 3, 5, 6, 9 - nie ma sprecyzowanego podmiotu, czyli aktora, który wykonuje krok.
- Krok 10 zawiera konstrukcję warunkową - warunki powinny się znaleźć w sekcji rozszerzeń, a nie w głównym scenariuszu.
- Scenariusz posiada 10 kroków, czyli jest za długi dla przeciętnego czytelnika. Liczba kroków powinna oscylować pomiędzy 3-9.

Dobre praktyki Sommerville i Sawyer'a

To, co zostało przedstawione to jedynie kilka podstawowych pomysłów (dobrych praktyk) związanych z dokumentowaniem wymagań. Inżynieria wymagań to dziedzina dużo bardziej skomplikowana. Skąd więc czerpać nowe pomysły? W momencie kiedy będziemy już pracować jako analitycy w firmach informatycznych przydałby się sposób na ocenienie, na ile dobrze wykonujemy fazę analizy wymagań. Skąd mogę wiedzieć, czy to co robię do tej pory, wystarcza już do efektywnego pozyskiwania wymagań, czy też może mogę robić to znacznie lepiej?

Z odpowiedzią na te dylematy przychodzi książka opublikowana w 1997 roku przez panów Yana Sommerville i Pete Sawyera poświęcona dobrym praktykom inżynierii wymagań. Zdefiniowali oni również pewien model, wg którego można oszacować poziom dojrzałości procesów inżynierii wymagań w firmie.

W celu sprawdzenia stopnia dojrzałości inżynierii wymagań w firmie, należy wziąć listę wszystkich praktyk zaproponowanych przez autorów, a następnie każdej z nich przydzielić od 0-3 punktów, w zależności od tego, w jaki sposób praktyka jest wykorzystywana w organizacji:

-3 punkty jeżeli praktyka jest standardem w organizacji
-2 punkty jeżeli nie jest standardem, lecz jednak jest powszechnie wykorzystywana
-1 punkt jeżeli jest wykorzystywana jedynie sporadycznie przez nieliczne jednostki
-0 jeżeli nigdy nie jest używana

Autorzy zdefiniowali 3 poziomy dojrzałości procesów inżynierii wymagań w przedsiębiorstwach (kryteria są lekko rozmyte):

- na pierwszym poziomie - początkowym znajduje się firma, która ma mniej niż 55 punktów z praktyk podstawowych
- na drugim poziomie - powtarzalnym - są te firmy, które mają powyżej 55 punktów za praktyki podstawowe oraz poniżej 40 z praktyk pośrednich i zaawansowanych

- na najwyższym poziomie - zdefiniowanym - znajdują się takie firmy, które mają powyżej 85 punktów za praktyki podstawowe, oraz powyżej 40 punktów za pośrednie i zaawansowane.

Poziomy te można krótko scharakteryzować:

- Poziom początkowy - to podejście ad-hoc do zbierania wymagań, pojawiają się częste problemy z wymaganiami
- Poziom powtarzalny - na tym poziomie są zdefiniowane standardy dla dokumentu wymagań oraz czynności pozyskiwania wymagań - problemów w fazie analizy wymagań jest dużo mniej
- Poziom zdefiniowany - posiada z góry zdefiniowany proces inżynierii wymagań, bazujący na najlepszych praktykach. Jest obecny program poprawy tego procesu.

Dobre praktyki – zalecenia

Zbieranie wymagań

- Oceń możliwość zbudowania systemu
- Zapisuj źródła wymagań
- Zdefiniuj środowisko operacyjne

Analiza i negocjacja wymagań

- Określ granice systemu
- Korzystaj z list kontrolnych podczas analizy wymagań
- Określ priorytety wymagań

Opisywanie wymagań

- Zdefiniuj standardowy szablon do opisu wymagań
- Korzystaj z prostego i spójnego języka
- Dobrze wykorzystuj diagramy

Walidacja wymagań

- Sprawdź, czy wymagania spełniają standardy
- Zorganizuj formalne inspekcje wymagań
- Zdefiniuj listy kontrolne walidacji
- Wykorzystaj zespoły wielodyscyplinarne do przeglądów wymagań