Wymagania funkcjonalne, przypadki użycia

Postawienie problemu i ustalenie zakresu

Dla każdego projektu informatycznego (i nie tylko), aby miał sens i warto było go realizować, powinien zostać najpierw zdefiniowany **problem**, który powinien zostać rozwiązany lub po prostu motywacja do podjęcia działania. Jest to często bardzo trudna część cyklu tworzenia projektu - czasami problem jest powszechnie znany lub napotykany bardzo często, jednak należy umieć go dostrzec i odpowiednio zdefiniować. Często zaczyna się realizować konkretny projekt myśląc o tym, "co" chcemy stworzyć, natomiast zapomina się "dlaczego". Jest o tyle ważne, że mając na względzie problem, który chcemy rozwiązać, ma się większą motywację do ukończenia projektu oraz tego, co właściwie chcemy zrobić - jest mniejsza szansa na to, że projekt zacznie ewoluować w zupełnie nieznanym kierunku i nie będzie widać do czego dążymy. Warto sobie również uświadomić znaczenie tego, co chcemy osiągnąć - czy ma to wymiar zaledwie lokalny (jak w przypadku np. aplikacji usprawniającej sesje gier karcianych) czy znacznie większy (jak ułatwienie zarządzania kadrą pracowników na uczelni).

Zdefiniowanie problemu pozwala postawić sobie cel, zwany również **celem biznesowym**. Powodem rozpoczęcia realizacji projektu jest chęć osiągnięcia celu i z tego powodu powinien on być stały (lub względnie stały), mimo często zmieniających się wymagań lub wizji aplikacji. Należy mieć na uwadze, że często na problem napotykają osoby niezwiązane z informatyką, a ich dziedziny mogą być najróżniejsze. Na dodatek, osoby potencjalnie korzystające na powstałej aplikacji mogą mieć różne cele, które mogą być ze sobą sprzeczne.

Postawienie celu determinuje ustalenie **zakresu** funkcjonalności projektu (w bardzo ogólnej postaci). Z uwagi na często bardzo szeroką problematykę, nie zawsze należy zakładać, iż uda się dotknąć każdego aspektu poruszanej tematyki. Dlatego jasne zdefiniowane zakresu pozwala na skupieniu się na konkretnych rzeczach, ułatwia specyfikację wymagań i stanowi też czynnik psychologiczny ("tego nie zrobimy, ale tak zakładaliśmy, czyli nie czujemy, że coś pomijamy").

Przykład problemu: Pod koniec każdego semestru na Politechnice Poznańskiej studenci otrzymują oceny za swoją pracę. Dotyczy to zarówno wykładów, ćwiczeń jak i laboratoriów, a same noty pochodzą z różnych „źródeł” (egzaminów, projektów itd.). Oceny powinny znaleźć się w indeksie oraz na karcie ocen, opatrzone m.in. personaliami wystawiającego oraz jego podpisem, a po uzbieraniu wszystkich wpisów student powinien udać się do dziekanatu, w którym następuje weryfikacja i dopuszczenie do następnego semestru (w pozytywnym przypadku). Wiąże się to również z wypełnianiem protokołów przez prowadzących oraz mozolnym zliczaniu punktów ECTS.

Studenci chcieliby uzyskać oceny możliwie jak najszybciej, aby po uzbieraniu ich kompletu oddać indeks i kartę ocen do dziekanatu i cieszyć się końcem semestru. Istnieje jednak kilka problemów, które trzeba mieć na uwadze:

- kolejki po wpisy u różnych prowadzących mogą być bardzo długie,

- prowadzący mogą być niedostępni i zwykle niechętni do umawiania się na udzielenie pojedynczego wpisu (a zebranie wystarczającej grupy studentów nie zawsze jest możliwe),
- zarówno indeks jak i (przede wszystkim) karta ocen mogą się zgubić,
- oddawanie indeksów do dziekanatu również wiąże się z długimi kolejkami.

Prowadzący zajęcia również chcieliby jak najszybciej wypełnić formalności. Podobnie jak dla studenta, uzupełnienie indeksów jest bardzo czasochłonne dla pracownika dydaktycznego (szczególnie prowadzącego wykład), podobnie jak przygotowanie protokołu, który następnie trzeba zanieść do dziekanatu. Dodatkowo są określone terminy czasowe na protokoły, o których trzeba pamiętać.

Także pracownicy dziekanatu potrzebowaliby usprawnień. Pod koniec semestru muszą przyjąć wielu studentów oraz wszystkie protokoły od prowadzących. Oceny muszą zostać wpisane do systemu komputerowego oraz zweryfikowane pod kątem punktów ECTS. Nie trzeba mówić, iż jest to bardzo czasochłonne i męczące, a musi zostać zrealizowane w stosunkowo krótkim okresie czasu.

Propozycja rozwiązania: Rozwiązaniem zaistniałego problemu jest przygotowanie systemu informatycznego (o nazwie eProto), który wspomógłby obsługę ocen studentów i znacznie zredukował czas ich wprowadzania i późniejszego przetwarzania. Z uwagi na rozporządzenie Ministerstwa Nauki i Szkolnictwa Wyższego z 2011 roku, możliwe stało się, aby prowadzący wypełniał protokoły elektronicznie, oceny automatycznie trafiały do dziekanatu i były od razu weryfikowane i przygotowywane do zatwierdzenia przez dziekana. W dodatku student mógłby śledzić oceny, które otrzymuje i zgłaszać sytuacje, kiedy wystąpiła jakaś pomyłka.

Zakres projektu:

- zarządzanie protokołami ocen
- śledzenie wpisywanych ocen i możliwość reagowania
- automatyczna weryfikacja ocen

Kontekst

Często samo zdefiniowanie problemu, celu i zakresu, do którego dążymy nie wystarczy - potrzebny jest również **kontekst**, w jakim aplikacja zostanie umieszczona. Można go opisać jako całe środowisko, które istnieje wokół naszej aplikacji i które trzeba mieć na uwadze, aby nie tworzyć projektu oderwanego od rzeczywistości. Na kontekst składają się przede wszystkim użytkownicy, którzy będą korzystać z systemu informatycznego, a konkretnie role, które będą pełnić. Aby dobrze zrozumieć ideę przyświecającą powstawaniu aplikacji, jej znaczenie, a także później wymagania, musimy wiedzieć, dla kogo ją tworzymy (czyli zdefiniować **grupę docelową**) i kto będzie ją wykorzystywał.

Takie role (w pewnym uogólnieniu) nazywamy **aktorami**. Najczęściej program musi spełniać oczekiwania każdej grupy użytkowników i żadnej nie pomijać. Być może istnieją również jakieś zainteresowane strony, którym zależy na tym, aby cel został zrealizowany, ale które nie

będą "typowymi" użytkownikami uczestniczącymi w procesie biznesowym (o czym na następnych zajęciach).

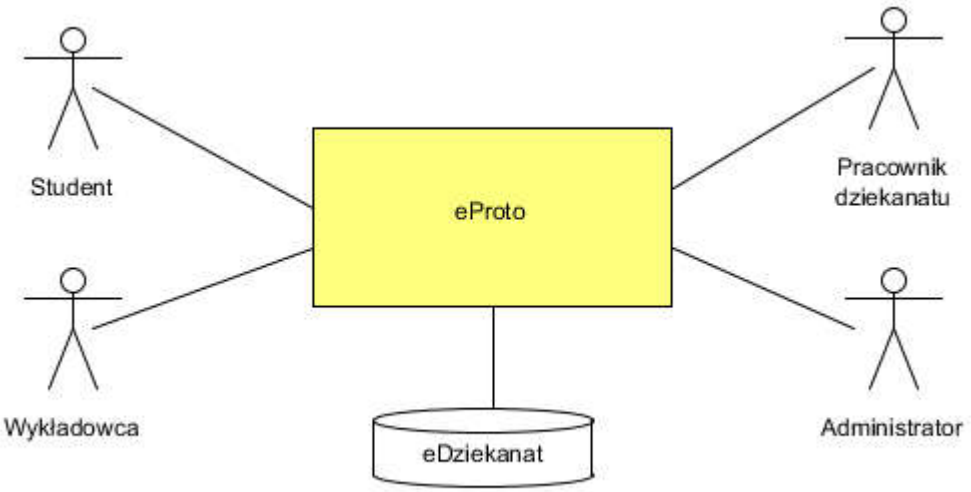
W eProto można wyróżnić następujących aktorów:

- Student
- Prowadzący zajęcia (zwany dla uproszczenia Wykładowcą)
- Pracownik dziekanatu
- Administrator – jest to właśnie przykład „nietypowej" roli, której zadanie jest utrzymywanie systemu i reagowanie na zaistniałe problemy i prośby

Dodatkowo, dla każdego członka grupy docelowej warto opisać korzyści, jakie pojawią się dla niego po wprowadzeniu systemu. W omawianym przykładzie, student będzie mógł bardziej kontrolować swoje oceny i szybciej zgłaszać wątpliwości. Wykładowcy nie będą musieli trudzić się wypełnianiem papierowych formularzy, natomiast pracownicy dziekanatu - nie będą musieli przetwarzać wszystkich danych "ręcznie".

W kontekst aplikacji wchodzić mogą również inne aplikacje lub systemy, które będą wykorzystywane lub same będą pobierać dane. Dobrym przykładem mogą być jakieś portale lub bazy danych udostępniające informacje, które są parsowane i pobierane przez naszą aplikację. Dodatkowo, kontekst obejmuje również wszelkie założenia, które możemy przyjąć (np. istnieją przygotowane serwery po stronie klienta, które możemy wykorzystać) lub zależności, jakie trzeba uwzględnić (np. musimy korzystać z SZBD Oracle).

W przypadku eProto takim systemem, który należy uwzględnić, jest eDziekanat – system dziekanatowy przechowujący i zarządzający wszystkimi informacjami dotyczącymi pracowników i studentów wydziału, który obsługuje listy grup, oceny oraz (nie)zaliczone semestry. Wynikiem opracowania kontekstu jest zwykle **diagram kontekstu** ukazujący w sposób graficzny aktorów oraz ich zależności między sobą a systemem(ami). Przykład dla eProto został przedstawiony na rysunku 1.



Rysunek 1: Diagram kontekstu dla systemu eProto.

Opowieści użytkowników

Wspomniano wcześniej o tym, iż użytkownicy mają różne cele, który chcieliby zrealizować i pewne oczekiwania. Często posiadają również pewną wizję korzystania z systemu, co jest istotne o tyle, że pozwala nam łatwiej zacząć definiować konkretne procesy, wymagania, a nawet prototypować interfejs użytkownika. Takie zdania opisujące wizję są nazywane opowieściami użytkowników (ang. *user stories*).

Opowieści mogą przybierać różną formę. Mogą to być pojedyncze zdania "zapisane" przez różnych użytkowników, czasami na małych karteczkach, które ułatwiają zarządzanie tym, co rzeczywiście powinno zostać zrealizowane.

Jako student chcę mieć możliwość przeglądania ocen, które zostały mi wystawione.

Jako wykładowca chcę mieć dostępną listę studentów, którym mogę wystawić oceny.

Jako pracownik dziekanatu chcę móc w łatwy sposób wydrukować wszystkie protokoły.

Zamiast pojedynczych zdań, opowieści często mogą być bardziej rozbudowane i rzeczywiście przypominać pewną historię oraz opis tego, jak użytkownik chce używać systemu.

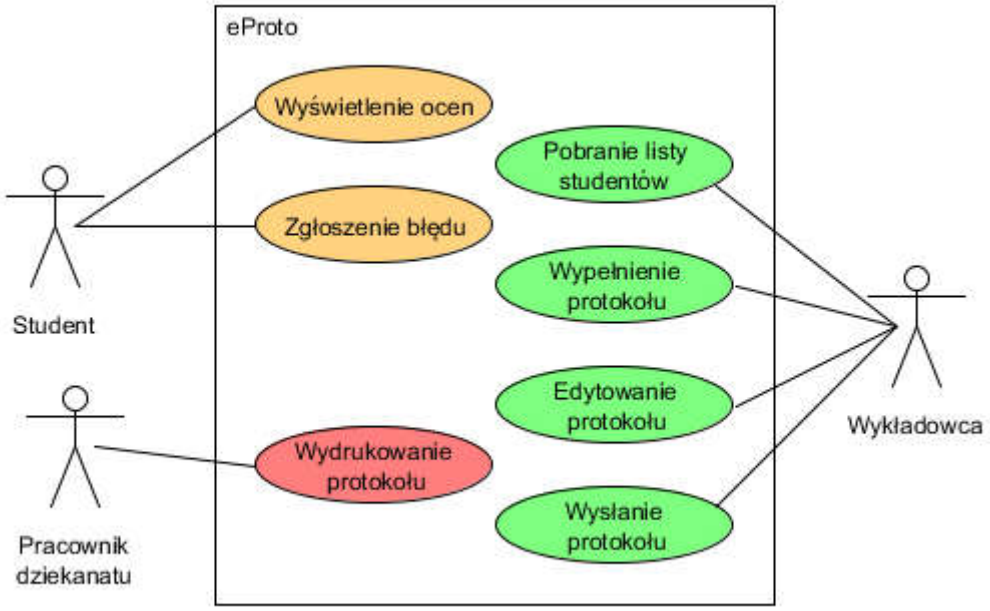
Jestem wykładowcą i po zakończonym egzaminie chcę wystawić oceny swoim studentom. Po określeniu przedmiotu pobieram sobie zatem najpierw listę osób, które uczestniczyły w egzaminie i po jej wyświetleniu wpisuję noty w odpowiednie rubryki. Gdy skończę, mogę łatwo wysłać wypełnione dane do dziekanatu. Może się oczywiście zdarzyć, że gdzieś się pomylę – w takim wypadku mogę jeszcze raz wyświetlić dany protokół i dokonać poprawki. Ponieważ mam dużo spraw na głowie i mogę łatwo przeoczyć termin wpisywania ocen, system wyśle mi maila przypominającego z pewnym wyprzedzeniem.

Należy pamiętać, że same opowieści nie są traktowane jako wymagania, gdyż czasami są zbyt ogólne i chaotyczne. Są jednak dobrym wstępem do specyfikacji funkcjonalności systemu, a także cech pozafunkcjonalnych.

Diagram przypadków użycia

Jedną z form diagramów UML (które będą omawiane w późniejszej fazie semestru) są diagramy przypadków użycia, które w prosty graficzny sposób przedstawiają funkcjonalność tworzonej aplikacji z uwzględnieniem każdego aktora. De facto, do przygotowania tego diagramu nie jest wymagane stworzenie pełnych przypadków użycia – wystarczą nazwy wymagań.

Przykład dla systemu eProto (przy okrojonej liście wymagań) znajduje się na rysunku 2.



Rysunek 2: Diagram przypadków użycia dla eProto.

Zasada dziewięciu kroków

Aby dobrze przeprowadzić i usystematyzować proces definiowania wymagań funkcjonalnych, w praktyce poleca się stosowanie tzw. zasady dziewięciu kroków. Znajdują się one poniżej.

1. Określenie problemu
2. Określenie procesu biznesowego
3. Identyfikacja aktorów (diagram kontekstu)
4. Identyfikacja przypadków użycia (diagram przypadków użycia)
5. Identyfikacja obiektów biznesowych

6. Zbadanie kompletności przypadków użycia (tablica CRUD)
7. Opisanie scenariuszy głównych
8. Identyfikacja zdarzeń
9. Opisanie scenariuszy rozszerzeń

Od czego zacząć?

W praktyce są dwa podejścia do rozpoczęcia specyfikowania wymagań funkcjonalnych. Pierwsze polega na zdefiniowaniu procesu (lub procesów) biznesowego na podstawie wiedzy domenowej, a drugie wiąże się z analizą opowieści użytkownika pod kątem funkcjonalności, jaką będzie musiał zapewniać system. Oczywiście, w praktyce te dwie drogi się przeplatają i mogą być łączone.

Procesy biznesowe

Proces biznesowy jest listą kolejnych kroków, działań, które umożliwiają osiągnięcie wcześniej wypracowanego celu biznesowego. W dość ogólny sposób pokazuje on przepływ informacji w programie i współdziałanie aktorów między sobą. Jest to szersze spojrzenie na funkcjonalność, niedotyczące konkretnego wymagania, ale ich zbioru i wpływu jednego na drugie. Definiowanie procesu biznesowego wiąże się z poznaniem wiedzy dziedzinowej, a jej źródłem jest najczęściej klient, który umie opisać obecnie obowiązujące procedury lub swoją wizję działania aplikacji.

Dla omawianego wcześniej systemu eProto można zdefiniować następujący proces biznesowy:

1. Na początku semestru Pracownik dziekanatu tworzy protokoły dla przedmiotów w aktualnym semestrze.
2. Wykładowca pobiera listę studentów dla wybranego przedmiotu.
3. Pracownik dziekanatu ustala terminy wypełnienia protokołu.
4. Po zakończeniu semestru Wykładowca wystawia oceny studentom.
5. Wykładowca uzupełnia protokół o wystawione oceny.
6. Pracownik dziekanatu sprawdza status protokołów.
7. Wykładowca drukuje i podpisuje wypełnione protokoły.
8. Wykładowca dostarcza protokoły do Pracownika dziekanatu.
9. Jeśli protokoły nie są dostarczone na czas, Wykładowcom wysyłane jest przypomnienie o obowiązku terminowego wypełnienia protokołów.
10. Studenci otrzymują informacje o wystawionych ocenach. Jeśli jakaś ocena jest niepoprawna, Student wysyła informacje o błędzie, która przekazywana jest do odpowiedniego Wykładowcy w celu korekty.

Procesy biznesowe często przedstawia się w sposób graficzny za pomocą własnych szkiców lub - w bardziej sformalizowany sposób - w postaci diagramów BPMN.

Obiekt biznesowy

Aby ułatwić tworzenie wymagań, należy wprowadzić pojęcie **obiektu biznesowego**. Najprościej rzecz ujmując, jest to obiekt, który reprezentuje pewną informację zarządzaną w systemie i na którym wykonywane są operacje, a dodatkowo działania na nim stanowią esencję systemu i bezpośrednio łączą się z celem biznesowym. Aby lepiej je opisać, można zdefiniować pola, z których się składają. Zarządzanie obiektami często sprowadza się do definicji operacji CRUDL (*Create, Retrieve, Update, Delete, List*) oraz innych specyficznych metod.

Warto zauważyć, iż często mylone jest pojęcie aktora oraz obiektu biznesowego, zwłaszcza, kiedy dane użytkownika są głównym obiektem zainteresowania aplikacji. Przykładowo - Student jako aktor jest rolą w procesie biznesowym (użytkownik, który może obserwować swoje oceny i zgłaszać nieprawidłowości), ale może też być obiektem biznesowym, jako zbiór ocen.

Obiekty biznesowe często pojawiają się na diagramie kontekstu jako sygnalizacja, iż dany aktor najczęściej operuje w systemie na danym typie obiektów. Często też podaje się obiekty, które nie mają jawnego znaczenia biznesowego, ale znacznie ułatwiają przeprowadzenie działań i zrozumienie całego procesu - należy wówczas to wyraźnie zaznaczyć.

Wymagania funkcjonalne i przypadki użycia

Wymagania funkcjonalne to konkretne aspekty funkcjonalności aplikacji, które definiują działania, które mogą być udostępniane dla użytkownika czy samego oprogramowania. Ich specyfikacja jest jednym z kluczowych punktów inicjowania projektu, gdyż dobrze zdefiniowane i opisane wymagania mogą znacznie uprościć implementację oraz analizę wprowadzanych modyfikacji.

Ich zbieranie można zacząć od procesów biznesowych (zwykle kolejne punkty opisują co najmniej jedno wymaganie) oraz opowieści użytkownika, gdzie „zaszyta" jest konkretna funkcjonalność. Sam proces identyfikacji wymagań może następować od najbardziej ogólnych do szczegółowych (tzw. podejście *top-down*), od szczegółowych do najbardziej ogólnych (*bottom-up*) lub w sposób mieszany.

Najpopularniejszym sposobem opisu wymagań funkcjonalnych są **przypadki użycia** (ang. *use cases*). Można je określić jako listy kroków (składających się na scenariusz) kolejnych interakcji użytkownika z systemem w celu realizacji danego wymagania. Jest to forma przyjazna zarówno dla klienta (gdyż nie wymaga technicznej wiedzy), analityka (jest prosta w użyciu) oraz programisty (wiadomo dokładnie jak należy zaimplementować wymaganie), ale także testera (który na tej podstawie może łatwiej ułożyć scenariusze testowe).

Przypadki użycia składają się zwykle z paru elementów:

- Identyfikator – z uwagi na często dużą liczbę wymagań w dokumentacji, łatwiej jest je znajdować poprzez alfanumeryczne identyfikatory aniżeli nazwy.

- Nazwa – powinna dobrze opisywać wymaganie, konkretnie je podsumowywać (np. "dodawanie oceny", a nie "zarządzanie oceną"). Poleca się używać formy bezosobowej.
- Aktorzy – wymaganie opisuje interakcję użytkownika (o konkretnej roli lub rolach) z systemem. System nie jest wymieniany osobno – jest "domyślnym" aktorem.
- Warunki wstępne i końcowe - określają zdarzenia lub stany, które muszą zaistnieć, aby był sens rozpatrywać całe wymaganie. W warunkach wstępnych często umieszcza się informację o konieczności zalogowania się użytkownika.
- Główny scenariusz – lista kroków opisujących kolejne kroki realizacji wymagania (a więc osiągnięcia jego "celu") przez użytkownika. Obejmuje ona tzw. *happy day scenario* – pozytywny przebieg wydarzeń, najbardziej typowy, nieprzerwany o początku do końca.
- Rozszerzenia – w przypadku, kiedy mogą wystąpić pewne sytuacje wyjątkowe, należy je umieścić w rozszerzeniach. Z uwagi na przejrzystość, poleca się umieszczać je po głównym scenariuszu, z numeracją adekwatną do konkretnych kroków z typowego przebiegu.

Oto przykłady przypadków użycia dla wymagań systemu eProto:

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.

Przypadki użycia mogą być jednego z trzech poziomów:

- Poziom biznesowy – opisuje procesy biznesowe i w ogólny sposób precyzuje przepływ informacji w systemie. Pokazuje interakcję pomiędzy różnymi rolami, oddziałami firmy i abstrahuje od poszczególnych funkcji systemu. Scenariusz często obejmuje wiele osób i niekoniecznie musi zamykać się w jednej sesji (może to być kilka sesji, łącznie trwających przez dłuższy okres czasu).
- Poziom użytkownika – najbardziej typowy, opisuje konkretne wymaganie funkcjonalne. Zwykle dotyczy interakcji jednego użytkownika (o konkretnej roli) z systemem informatycznym, jak np. edytowanie protokołu. Ważne jest, aby w tym poziomie nie wnikać w techniczne szczegóły dotyczące np. interfejsu użytkownika, pomijamy także opis szczegółowych danych przy wypełnianiu wszelkiego rodzaju formularzy (chyba że są one krytyczne dla wymagania – przykładowo możemy wspomnieć o konieczności nadania przez użytkownika nazwy przedmiotu, ale nie opisujemy jego konkretnych cech jak liczba punktów ECTS, liczba godzin itp.). Ten poziom obejmuje jedną sesję, mieszczącą się w kilku lub kilkunastu minutach.
- Poziom podfunkcji – dotyczy szczegółów technicznych wymagania, architektury i graficznego interfejsu użytkownika. Ten poziom może być wykorzystywany przy niektórych wymaganiach, aby można było je lepiej zrozumieć.

Warto zaznaczyć, iż przypadki użycia mogą tworzyć hierarchie. W szczególności na poziomie biznesowym, przy konkretnych krokach, można dodać adnotację dotyczącą konkretnego przypadku na poziomie użytkownika. Można to wykonać w następujący sposób:

5. Wykładowca uzupełnia protokół o wystawione o ceny [UC03: Uzupełnianie protokołu].

Warto wymienić dobre praktyki związane z pisaniem przypadków użycia:

- System uczestniczy we wszystkich przypadkach użycia na poziomie użytkownika, ale nie ma potrzeby wypisywania go w aktorach.
- Zawsze należy wypisać aktora, który wykonuje daną akcję – dla każdego kroku musi być on zdefiniowany. Wyjątkiem są "tytuły" rozszerzeń i standardowe kroki w rodzaju "Powrót do kroku" oraz "Koniec przypadku użycia".
- Przypadek użycia powinien być odpowiedniej długości. Scenariusze poniżej 3 kroków wydają się bezużyteczne i mogą być "wchłonięte" przez inne, natomiast powyżej 9 kroków są ciężkie w odczytaniu, zrozumieniu i najprawdopodobniej dotyczą zbyt wielu rzeczy.

- Należy unikać "technikaliów", w tym określeń typowych dla interfejsu użytkownika (jak "klika", "wpisuje" – zamiast tego piszemy "wybiera", "podaje"). Ważna jest funkcjonalność, a nie sposób jej realizacji – ten aspekt pokrywają prototypy i trochę wymagania pozafunkcjonalne. Jednakże, na poziomie podfunkcji szczegóły techniczne są dopuszczalne.
- Główny scenariusz to lista typowych, nieprzerwanych kroków, które kończą się optymistycznie. Na wszelkie konstrukcje warunkowe i możliwe anomalie miejsce jest w rozszerzeniach.
- Należy stosować proste zdania i nie bać się powtórzeń – przypadek użycia nie jest wierszem z okresu romantyzmu do analizy na języku polskim, lecz ma precyzyjnie "opowiadać" o danym wymaganiu i być bezproblemowo zrozumiany przez czytającego (a przede wszystkim realizującego).
- Nazwa przypadku użycia powinna odzwierciedlać cel, który chce osiągnąć aktor. Przykładowo, prawidłowa nazwa wymagania to "usunięcie studenta", a nie "dezaktywacja rekordu studenta", nawet jeżeli technicznie tak to wygląda.
- Przypadek użycia powinien kończyć się na ostatnim kroku w głównym scenariuszu lub punkcie "koniec przypadku użycia" w jednym z rozszerzeń. Każde rozszerzenie powinno kończyć się zakończeniem przypadku użycia (jak wyżej) lub powrotem do konkretnego punktu w głównym scenariuszu.
- Zwykle zakłada się, że aktor jest zalogowany do systemu oraz że znajduje się na pierwszej (głównej) stronie czy widoku. Aby uzyskać klarowność, poleca się dodanie do przypadku użycia opisu warunków wstępnych, gdzie znajduje się informacja o założeniach przyjętych przy realizacji danego wymagania.

Specyfikowanie wymagań funkcjonalnych

Specyfikowanie wymagań funkcjonalnych jest niezbędnym procesem, jeżeli twórcom zależy na odpowiedniej realizacji oprogramowania i założonych celów. Nie jest to jednak zadanie łatwe i wymaga pewnego doświadczenia oraz ćwiczeń.

Wymagania funkcjonalne są specyfikowane najczęściej dzięki trzem źródłom:

- procesy biznesowe i wiedza dziedzinowa
- opowieści użytkownika
- spotkania z klientem, zauważenie potrzeby rozszerzenia funkcjonalności

Nie jest to proces kończący się przed implementacją, ale ciągłe zadanie, którym obarczony jest analityk projektu - wymagania mogą się zmieniać, zarówno pod względem swojej treści jak i priorytetu. Oczywiste jest, iż nie wszystkie obszary funkcjonalności systemu są równie ważne i skupienie się na wszystkim w równym stopniu mogłoby negatywnie wpłynąć na sam proces powstawania oprogramowania. Istnieją wymagania obowiązkowe dla systemu, bez realizacji których traci on swój pierwotny sens i nie będzie mógł być osiągnięty cel biznesowy, ale także wymagania ważne, które warto zrealizować jak i opcjonalne, które dodatkowo podnoszą wartość rozwiązania, ale brak ich obecności nie będzie mocno rzutował na negatywny odbiór systemu.

Podsumowując, najczęściej przyjmuje się następujące wartości priorytetów:

- H - *High*, wysoki priorytet (obowiązkowe wymaganie, musi być bezwzględnie zrealizowane ze względu na osiągnięcie celu biznesowego)
- M - *Medium*, średni priorytet (ważne wymaganie, powinno być zrealizowane)
- L - *Low*, niski priorytet (wymaganie opcjonalne, zostanie zrealizowane przy wolnych zasobach czasowych)

Czasami proponuje się rozszerzenie tej skali o VH (*Very High*) oraz VL (*Very Low*), jeżeli dane wymaganie zasługuje na wyjątkowe wyróżnienie i jest krytyczne lub bardzo opcjonalne.

W omawianym cyklu zajęć proponuje się specyfikację wymagań w oparciu o opowieści użytkownika, które były tematem jednych z poprzednich zajęć. Poniżej znajduje się przykład dłuższej opowieści użytkownika dla systemu eProto.

Jestem wykładowcą i zbliża się koniec semestru. To oznacza, iż studenci powinni otrzymać oceny za zajęcia, które prowadzę. Aby wystawić im noty, muszę wejść do systemu eProto, wybrać przedmioty, które prowadzę i utworzyć dla nich protokoły z listą studentów. Następnie, każdemu studentowi wpisuję oceny wraz z datą. Po każdym wprowadzeniu oceny i zapisaniu, student dostaje powiadomienie, aby był świadomy wystawionego stopnia i ewentualnie mógł zgłosić błąd. Dany protokół mogę edytować w każdej chwili, ponieważ wiadomo, że nie zawsze od razu dysponuję wszystkimi pracami studentów. Gdy w którymś momencie uznam, że wprowadziłem wszystkie noty, zatwierdzam swój protokół, co powoduje udostępnienie go pracownikom dziekanatu w celu dalszej analizy i dopełnienia formalności.

Jakie wymagania funkcjonalne można wyróżnić z tego opisu?

1. Utworzenie protokołu (Wykładowca)
2. Wypełnienie protokołu (Wykładowca, dodatkowo generuje powiadomienie dla Studenta)
3. Edytowanie protokołu (Wykładowca, dodatkowo generuje powiadomienie dla Studenta)
4. Wysłanie protokołu (Wykładowca)
5. Zgłoszenie błędu (Student)

Oczywiście, zakres wyróżnionych wymagań może być nieco inny w zależności od przyjętego podejścia - na przykład, można się skupić na wprowadzaniu pojedynczych ocen, a nie od razu całego protokołu. Należy również podkreślić fakt, iż w danej opowieści może być "zaszyte" wiele wymagań funkcjonalnych - nawet w krótszych można czasem doszukać się kilka funkcji wartych wypisania.

Poniżej znajduje się inny przykład:

Chciałbym kupić dobry tablet za nie więcej niż 800zł. Myślę, że da się za to kupić coś dobrego, ale mało się na tym znam i nie chcę się naciąć. Wchodzę więc do systemu recenzowania produktów, wyszukuję "tablet", wpisuję przedział cenowy i dostaję listę tabletów posortowanych wg ocen z recenzji. Wybieram pierwszy z nich i widzę kilka recenzji.

Klikam na pierwszą – ma formę filmu opis słowny. Z zainteresowaniem oglądam film. Widzę, że recenzent ma w systemie status "Ekspert". Klikam na recenzenta i sprawdzam jakie inne produkty recenzował. Okazuje się, że recenzował tylko trochę droższy tablet z jeszcze lepszą opinią. Co tam, wydam trochę więcej, ale będę zadowolony. Zapisuję się na obserwowanie kolejnych recenzji tego recenzenta

Funkcje wynikające z opowieści:

1. Wyszukiwanie produktu
2. Sortowanie produktów
3. Przejście do produktu
4. Przeglądanie recenzji
5. Przeglądanie recenzji wybranego recenzenta
6. Subskrypcja recenzenta