laboratoria 1—3

Modelowanie wymagań i przypadków użycia

Cel zajęć: Organizacja zespołu projektowego.
Ustalenie dziedziny tworzonego oprogramowania.
Określenie wymagań stawianych tworzonemu oprogramowaniu.
Budowa modelu przypadków użycia tworzonego oprogramowania.

Źródła wiedzy: Wykład IO: *Języki graficznego modelowania* (dostępny też w *ePortalu PWr*).
Wykład IO: *Modelowanie wymagań i przypadków użycia* (dostępny też w ePortalu PWr).
Przykład: *Miniprojekt Biblioteka* (dostępny też w *ePortalu PWr*).
Tutorial *Visual Paradigm*: *How to Draw Use Case Diagram?*

Zawartość instrukcji: 6 zadań – ich wykonanie jest oceniane.
4 ćwiczenia pomocnicze.
Spis rzeczy, które należy umieścić w sprawozdaniu z wykonania zadań.
Spis błędów i braków mogących obniżyć ocenę.

Zadania

Zadanie 1
Ustalenie dziedziny tworzonego oprogramowania

Należy wybrać dziedzinę dla tworzonego oprogramowania.

Przykładowe dziedziny:

1) Sprzedawanie biletów na seanse filmowe w multikinie.
2) Wykonywanie operacji bankowych przy pomocy infokiosku lub bankomatu.
3) Rejestrowanie statusu obywateli w urzędzie przez dział ewidencji ludności.
4) Zarządzanie rozkładem jazdy i kontrolowanie przejazdów komunikacji miejskiej.
5) Magazynowanie i wypożyczanie sprzętu budowlanego.
6) Zarządzanie siecią paczkomatów i przewozem paczek przez firmę kurierską.
7) Zapisywanie się studenta na zajęcia.
8) Księgowanie operacji finansowych przez zewnętrzną księgowość.
9) Rezerwowanie i opłacanie wynajmu miejsc w hotelu przez klienta.
10) Zarządzanie pracą lotniska pasażerskiego.
11) Zarządzanie projektem informatycznym przy pomocy systemu Git.

Zadanie 2
Przygotowanie projektu informatycznego

Należy utworzyć nowy projekt w programie *Visual Paradigm*. W tym i kolejnych etapach laboratoriów będą w nim umieszczane dokumenty tekstowe i diagramy modelujące tworzone oprogramowanie.

Nazwa projektu powinna krótko i zrozumiale opisywać dziedzinę tworzonego oprogramowania.

Należy umieścić ten projekt w repozytorium *Visual Paradigm* (robi to jedna osoba z zespołu projektowego) i udostępnić go pozostałym członkom zespołu oraz prowadzącemu laboratoria.

Praca z repozytorium *Visual Paradigm*:

– Connect Visual Paradigm to VP Online for team collaboration
– Tworzenie współdzielonego projektu Visual Paradigm

Zadanie 3
Opisanie miejsca wdrożenia tworzonego oprogramowania

Należy utworzyć dokument tekstowy w *Visual Paradigm* i umieścić w nim krótki, szkicowy opis miejsca wdrożenia tworzonego oprogramowania (tzw. „świat rzeczywisty”), który byłby przekazany przez zamawiającego oprogramowanie.

Główne elementy opisu:

– zasoby ludzkie,
– przepisy i strategie,
– dane techniczne.

Zasoby ludzkie:

Kim są i co robią przyszli użytkownicy oprogramowania?

W kolejnych zadaniach na podstawie tego opisu określi się aktorów, czyli role użytkowników, i wymagania funkcjonalne, czyli zadania, które oprogramowanie musi wykonać lub wykonywać, aby wspomagać pracę jego użytkowników.

W przypadku systemów wbudowanych ten opis dotyczy procesów technologicznych, które należy zautomatyzować.

Przepisy i strategie:

Co ogranicza działalność klienta zamawiającego oprogramowanie?

Mogą to być np.: ustawy, zarządzenia, strategie klienta lub użytkownika.

W kolejnych zadaniach na podstawie tego opisu określi się wymagania niefunkcjonalne, czyli ograniczenia dla wymagań funkcjonalnych i dla działania oprogramowania.

Dane techniczne:

Jakie dane (artefakty) są związane z pracą tworzonego oprogramowania?

Mogą to być np.:
– liczba pracowników klienta, rodzaj i liczba przetwarzanych danych, częstość wykonywania operacji na danych;
– umiejscowienie i charakterystyka użytkownika;
– używany sprzęt i oprogramowanie, w tym to przeznaczone do wdrożenia tworzonego oprogramowania.

Zadanie 4
Określenie wymagań stawianych tworzonemu oprogramowaniu

Należy utworzyć dokument tekstowy w *Visual Paradigm* i umieścić w nim numerowaną listę wymagań funkcjonalnych i numerowaną listę wymagań niefunkcjonalnych tworzonego oprogramowania.

Wymagania powinny współgrać z dokumentem wykonanym w poprzednim zadaniu, być zwięźle nazwane i szczegółowiej opisane, jeśli nazwa nie wystarcza.

Wymaganie funkcjonalne:

Co tworzone oprogramowanie powinno robić lub jakie cele osiągać, aby zautomatyzować procesy wykonywane przez klienta.

Wymaganie niefunkcjonalne:

Jak lub w jakich warunkach tworzone oprogramowanie ma to osiągnąć – przy jakich ograniczeniach sprzętowych, organizacyjnych, prawnych itp.

Jakie zastosować rozwiązania technologiczne m. in. w zakresie: bezpieczeństwa, niezawodności, skalowalności i wydajności, aby spełnić oczekiwania użytkownika lub klienta.

Zadanie 5
Budowa modelu przypadków użycia tworzonego oprogramowania – diagram

Należy utworzyć diagram przypadków użycia w *Visual Paradigm* i umieścić w nim: system, aktorów i przypadki użycia oraz relacje między nimi.

Najpierw należy określić system (granice tworzonego oprogramowania) i jego aktorów (role pasywnych lub aktywnych użytkowników tego oprogramowania).

Następnie należy określić przypadki użycia tworzonego oprogramowania, w tym 2 bardziej lub 3 mniej złożone. W tym celu należy wykonać analizę wspólności i zmienności przypadków użycia, która znajdzie relacje:

– uogólnienia, zawierania (musi być jej przykład) i rozszerzania (musi być jej przykład) między przypadkami użycia;

– uogólnienia między aktorami.

Kolejności realizacji przypadków użycia ten diagram nie modeluje.

Następnie należy połączyć aktorów relacjami asocjacji z przypadkami użycia.

System:

Wyznacza granice tworzonego oprogramowania i zapewnia realizację umieszczonych w nim przypadków użycia.

Jeśli tworzone oprogramowanie ma składać się z odrębnych, współpracujących ze sobą części, to każdą część modeluje osobny system.

Aktor:

Osoba, urządzenie, inne oprogramowanie itp. może być aktorem tworzonego oprogramowania, jeśli ma mieć bezpośredni udział w jego działaniu, czyli bez pośrednictwa innego aktora. Tylko taki aktor może być połączony relacją asocjacji z przypadkiem użycia.

Współpracę aktorów, będącą poza tworzonym oprogramowaniem, ale ważną dla jego modelu, można zamodelować przez połączenie ich ze sobą relacją asocjacji.

Przypadek użycia:

Proces biznesowy składający się z operacji wykonywanych przez tworzone oprogramowanie (jego funkcja), zmierzający do spełnienia jednego lub więcej wymagań funkcjonalnych z uwzględnieniem ograniczeń zawartych w wymaganiach niefunkcjonalnych.

Przypadek użycia powinien być nazwany ogólnie jak funkcja tworzonego oprogramowania, a nie jak obiekt. Przykład dobrej nazwy: *Włączenie sygnalizacji alarmowej*. Przykład złej nazwy: *Sygnalizacja alarmowa*.

Relacja uogólnienia:

Łączy przypadki użycia lub aktorów w związek wersja podstawowa (ogólna) – wersja pochodna (szczególna).

Gdy dwaj aktorzy mają identyczny lub prawie identyczny udział w realizacji przypadku użycia, to łączy ich uogólnienie i tylko ogólny aktor jest połączony relacją asocjacji z tym przypadkiem użycia.

Relacja zawierania («include»):

Łączy przypadki użycia w związek zależna całość – obowiązkowa część.

Gdy część realizacji przypadku użycia może być wykonywana osobno (nie tylko jako jego część) lub gdy może być też wykonywana jako część innego przypadku użycia, to należy ją zamodelować osobnym przypadkiem użycia i przyłączyć go relacją zawierania.

Relacja rozszerzania («extend»):

Łączy przypadki użycia w związek niezależna całość – opcjonalna lub alternatywna część.

Gdy część realizacji przypadku użycia może być wykonywana opcjonalnie (losowo lub zależnie od spełnienia jakiegoś warunku) lub alternatywnie z inną, lub innymi częściami jego realizacji, to należy ją zamodelować osobnym przypadkiem użycia i przyłączyć go relacją rozszerzania.

Złożony przypadek użycia:

Zawiera w sobie lub jest rozszerzany przez inne przypadki użycia.

Za bardziej złożony przypadek użycia należy przyjąć taki, który sam składa się ze złożonych przypadków użycia.

Relacja asocjacji:

Głównie łączy aktora z tym przypadkiem użycia, którego realizację ten aktor inicjuje (aktywny udział) lub który inicjuje z tym aktorem przepływ danych albo sterowania (pasywny udział). Relacja ta może być skierowana i mieć określoną liczność (w ilu naraz realizacjach przypadku użycia może brać udział aktor).

Zadanie 6
Budowa modelu przypadków użycia tworzonego oprogramowania – słowny opis

Należy wykonać słowny opis każdego przypadku użycia, który znajduje się na diagramie wykonanym w poprzednim zadaniu. Ten opis należy umieścić w odpowiednich zakładkach okna *Use Case Details* programu *Visual Paradigm* dla danego przypadku użycia.

Elementy opisu przypadku użycia:

– numer: identyfikator;
– nazwa: krótka, mająca formę wykonywanej operacji, a nie obiektu, zaczynająca się od numeru przypadku użycia;
– cel: dłuższe wyjaśnienie, jeśli nazwa nie wyjaśnia przeznaczenia przypadku użycia;
– warunki wstępne: możliwość realizacji przypadku użycia, w tym: wymagania, których wcześniejsze spełnienie lub przypadki użycia, których wcześniejsze zrealizowanie pozwala rozpocząć wykonywanie opisywanego przypadku użycia;
– warunki końcowe: poprawność zrealizowania przypadku użycia, w tym wymagania spełnione lub przypadki użycia zrealizowane przez opisywany przypadek użycia;
– scenariusz: przepływ zdarzeń, które składają się na realizację przypadku użycia; algorytm w postaci szczegółowego, numerowanego pseudokodu w języku naturalnym;
– alternatywne scenariusze: alternatywne przepływy zdarzeń (jeśli istnieją) z podaniem ich warunków wstępnych.

Scenariusz musi pokazywać miejsce (punkt w pseudokodzie) realizacji innego przypadku użycia, będącego w relacji zawierania z opisywanym przypadkiem użycia oraz miejsce (punkt w pseudokodzie) i warunek (opcjonalność, losowość, alternatywność) realizacji innego przypadku użycia, będącego w relacji rozszerzania z opisywanym przypadkiem użycia.

Scenariusz nie może zawierać scenariusza innego przypadku użycia, w szczególności takiego, który jest z nim połączony relacją zawierania lub rozszerzania.

Zadania 3, 5 i 6 opracowano na podstawie instrukcji 2 i instrukcji 3 „Laboratorium z przedmiotu: Inżynieria Oprogramowania W04ITE-SI0011G", autorstwa dr inż. Zofii Kruczkiewicz.

Ćwiczenia

Przykład wstępnego opisu wymagań:

Poniższa lista zawiera razem wymagania funkcjonalne i niefunkcjonalne. Wiele z nich jest złożonych i mogą wymagać dekompozycji na prostsze wymagania, w szczególności przez rozdzielenie danego wymagania na wymaganie funkcjonalne i ograniczające je wymaganie niefunkcjonalne. Jest to więc podstawa do dalszej analizy wymagań.

1) Projektowany system informatyczny (SI) automatyzuje kontrolę bezpiecznego poziomu promieniowania i reagowanie na jego przekroczenie w sterowni elektrowni atomowej.
2) SI jest obsługiwany przez kontrolera bezpieczeństwa i nadzorowany przez kierownika sterowni.
3) Kierownik sterowni ma te same uprawnienia do obsługi SI, co kontroler bezpieczeństwa, a dodatkowo na koniec każdego miesiąca otrzymuje z SI raport o pracy SI.
4) SI działa na podstawie poleceń wydawanych przez kontrolera bezpieczeństwa lub kierownika sterowni oraz odczytów stanu urządzeń pomiarowych i urządzeń alarmowych.
5) Kontroler bezpieczeństwa sprawdza sprawność systemu alarmowego przez przeprowadzenie testu sygnalizacji świetlnej i dźwiękowej przekroczenia bezpiecznego poziomu promieniowania, opcjonalnie razem z testem sygnalizacji świetlnej nakazującej ewakuację pracowników sterowni.
6) Test sygnalizacji świetlnej i dźwiękowej systemu alarmowego nie może być wykonywany częściej niż raz na tydzień.
7) SI stale monitoruje poziom promieniowania w sterowni przy pomocy odczytów z urządzeń pomiarowych.
8) SI na bieżąco informuje kontrolera bezpieczeństwa o stanie i zmianach poziomu promieniowania w sterowni.
9) SI włącza w urządzeniach alarmowych alarm przekroczenia bezpiecznego poziomu promieniowania i nakaz ewakuacji pracowników sterowni w przypadku przekroczenia bezpiecznego poziomu promieniowania w sterowni oraz informuje o tym kontrolera bezpieczeństwa.
10) Kontroler bezpieczeństwa odwołuje alarm przekroczenia bezpiecznego poziomu promieniowania i nakaz ewakuacji pracowników sterowni, jeśli uzna taką potrzebę. Wówczas SI wyłącza w urządzeniach alarmowych ten alarm i nakaz.

Ćwiczenie 1
Dekompozycja i segregacja wymagań

Proszę przekształcić wybrane wymagania tak, aby:
- były niepodzielne (nie zawierały w sobie innego wymagania),
- jednoznacznie wskazywały, czy są funkcjonalne czy niefunkcjonalne.

Przykład dekompozycji wymagania 1) na wymagania funkcjonalne 1a) i 1b):

1) *Projektowany system informatyczny (SI) automatyzuje kontrolę bezpiecznego poziomu promieniowania i reagowanie na jego przekroczenie w sterowni elektrowni atomowej.*

1a) *SI automatyzuje kontrolę bezpiecznego poziomu promieniowania w sterowni.*

1b) *SI automatyzuje reagowanie na przekroczenie bezpiecznego poziomu promieniowania w sterowni.*

Przykład dekompozycji wymagania 7) na wymaganie funkcjonalne 7a) i niefunkcjonalne wymagania 7b) i 7c):

7) *SI stale monitoruje poziom promieniowania w sterowni przy pomocy odczytów z urządzeń pomiarowych*.

7a) *SI monitoruje poziom promieniowania w sterowni*.

7b) *Monitorowanie poziomu promieniowania w sterowni przez SI jest stałe*.

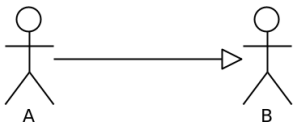
7c) *Monitorowanie poziomu promieniowania w sterowni przez SI bazuje na odczytach z urządzeń pomiarowych*.

Ćwiczenie 2
Określenie aktorów i relacji między nimi

Z wymagania 2) wynika, że aktorami są *kontroler bezpieczeństwa* i *kierownik sterowni*.

Proszę na podstawie wymagań:
- narysować relację uogólnienia między tymi aktorami,
- określić innych aktorów (podpowiedź: nie są to ludzie).

Przykład relacji uogólnienia aktorów:
Z diagramu obok wynika, że aktor A może wejść w rolę aktora B.



Ćwiczenie 3
Określenie przypadków użycia

Proszę na podstawie wybranych wymagań funkcjonalnych:
- narysować spełniające je przypadki użycia,
- nazwać te przypadki użycia w sposób jednoznacznie wskazujący na ich cel,
- narysować relacje uogólnienia łączące te przypadki z odpowiednimi aktorami.

Przykład określenia nazwy przypadku użycia PU1 na podstawie wymagania 7a):

7a) *SI monitoruje poziom promieniowania w sterowni*.

PU1: *Monitorowanie poziomu promieniowania*.

Ten przypadek użycia może być połączony relacją asocjacji z aktorem *urządzenie pomiarowe*.

Ćwiczenie 4
Dekompozycja przypadków użycia

Proszę przekształcić diagram przypadków użycia tak, aby z wybranego przypadku użycia wydzielić inny, nowy przypadek użycia, będący z nim w relacji całość – część:
- w relacji zawierania (gdy dwa przypadki użycia współdzielą część swego przebiegu),
- w relacji rozszerzania (gdy część przebiegu przypadku użycia jest opcjonalna).

Na podstawie wymagania 5) można określić przypadek użycia PU2. Przykład jego dekompozycji przy pomocy przypadków użycia PU3 i PU4 pokazuje diagram obok:

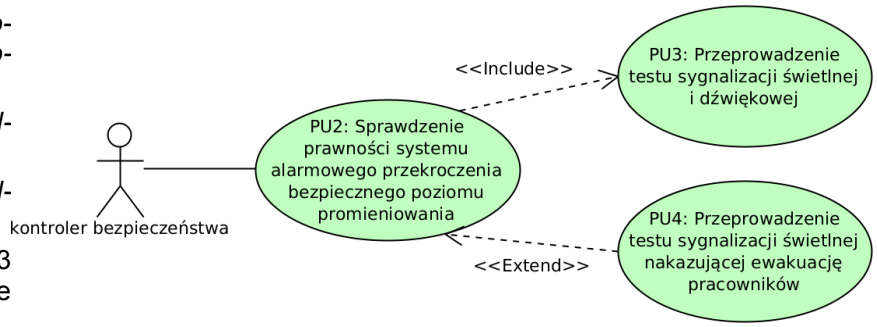
5) *Kontroler bezpieczeństwa sprawdza sprawność systemu alarmowego przez przeprowadzenie testu sygnalizacji świetlnej i dźwiękowej przekroczenia bezpiecznego poziomu promieniowania, opcjonalnie razem z testem sygnalizacji świetlnej nakazującej ewakuację pracowników sterowni.*

PU2: *Sprawdzenie prawności systemu alarmowego przekroczenia bezpiecznego poziomu promieniowania*.

PU3: *Przeprowadzenie testu sygnalizacji świetlnej i dźwiękowej*.

PU4: *Przeprowadzenie testu sygnalizacji świetlnej nakazującej ewakuację pracowników.*

Takie wydzielenie przypadku użycia PU3 oznacza, że może być też użyty samodzielnie lub jako część innego przypadku użycia.



Na podstawie wymagania 5) tylko przypadek użycia PU2 może być połączony relacją asocjacji z aktorem *kontroler bezpieczeństwa*, ponieważ pozostałe przypadki użycia są tylko częściami PU2.

Sprawozdanie

Co powinno być na początku sprawozdania:

- Autorzy (imię, nazwisko, nr albumu).
- Nazwa tworzonego oprogramowania (zwięzły opis jego dziedziny).
- Temat tego etapu laboratoriów (*Modelowanie wymagań i przypadków użycia*).

Co powinno być w treści sprawozdania:

- Dokument wykonany w zadaniu 3 (opis miejsca wdrożenia).
- Dokument wykonany w zadaniu 4 (lista wymagań funkcjonalnych i lista wymagań niefunkcjonalnych).
- Diagram przypadków użycia wykonany w zadaniu 5.
- Słowny opis każdego przypadku użycia wykonany w zadaniu 6 (jak w przykładowym miniprojekcie, który można przeszukiwać (Ctrl+F), a nie zrzut ekranu programu *Visual Paradigm*).

Błędy i braki

Jeśli opisane niżej rzeczy mają miejsce, ocena za ten etap laboratoriów może być niższa:

- Wymagania funkcjonalne i niefunkcjonalne są na wspólnej liście.
- Wymagania na liście są nieponumerowane.
- Nazwa wymagania jest nieprecyzyjna, zamiast np. składać się z podmiotu, orzeczenia i dopełnienia.
- Wymaganie, które ogranicza działanie systemu, jest na liście wymagań funkcjonalnych.
- Wymaganie, które opisuje zadanie stawiane tworzonemu oprogramowaniu, jest na liście wymagań niefunkcjonalnych.
- Przypadek użycia, znajdujący się na diagramie przypadków użycia, nie jest opisany lub nie ma pełnego opisu.
- Opisany przypadek użycia nie znajduje się na diagramie przypadków użycia.
- Nazwa przypadku użycia nie zaczyna się od jego numeru.
- Nazwa przypadku użycia jest nieprecyzyjna lub nie brzmi jak operacja (proces biznesowy).
- Nazwa przypadku użycia w jego opisie jest inna niż na diagramie przypadków użycia.
- Samodzielna, wspólna, alternatywna lub opcjonalna część realizacji przypadku użycia nie jest osobnym przypadkiem użycia.
- Brak przykładu relacji zawierania między przypadkami użycia.
- Brak przykładu relacji rozszerzania między przypadkami użycia.
- Przypadki użycia, między którymi nie ma relacji całość – część, są powiązane relacją zawierania lub rozszerzania.
- Powiązanie między przypadkami użycia pokazuje kolejność ich realizacji, zamiast relację całość – część.
- Przypadki użycia są powiązane ze sobą inną relacją niż zawierania, rozszerzania lub uogólnienia.
- Grot relacji zawierania wskazuje na zawierający przypadek użycia.
- Grot relacji rozszerzania wskazuje na rozszerzany przypadek użycia.
- Aktor jest powiązany z aktorem lub przypadkiem użycia relacją zawierania, lub rozszerzania.
- Brak relacji uogólnienia między aktorami, którzy mają identyczny udział w realizacji danego przypadku użycia.

Ta lista zawiera typowo pojawiające się błędy i braki i nie wyczerpuje powodów, dla których ocena za laboratoria może być niższa.