UNIWERSYTET OPOLSKI

INSTYTUT INFORMATYKI
ul. Oleska 48, 45-052 Opole
tel.: +48 (77) 452 72 05
fax: +48 (77) 452 72 07
ii@uni.opole.pl
informatyka.wmfi.uni.opole.pl

Wykłady z inżynierii oprogramowania

Analiza wymagań funkcjonalnych. Model use case'ów.

Zawartość

1. Wymagania funkcjonalne, niefunkcjonalne i wymagania dotyczące jakości modelowanego systemu

Wymagania niefunkcjonalne i wymagania dotyczące jakości modelowanego systemu dotyczą

- wyboru platformy systemowej,
- parametrów środowiska pracy systemu (operational requirements),
- wymagań związanych z wielkością zużywanych przez system zasobów takich, jak pamięć, czas pracy procesora (resource requirements),
- bezpiecznego użycia (safety requirements),
- struktury interfejsów (interface requirements),
- parametrów jakości (quality requirements):
  - niezawodności (reliability),
  - obsługiwalności (maintainability),
  - dostępność (availability),
  - łatwości przenoszenia budowanej aplikacji między systemami (portability requirements),
  - wydajności (performance),
  - bezpieczeństwa (security requirements),
- procesów implementacji systemu, stosowanych standardów wytwarzania oprogramowania,
- zasad dokumentowania systemu (documentation requirements),
- metod weryfikacji wymagań (verification requirements),
- zasad akceptacji i odbioru systemu (acceptance requirements),
- procesów dostarczania produktów projektu klientowi,
- ograniczeń prawnych dotyczących praw autorskich, licencji i innych ograniczeń prawnych,
- ograniczeń na metody zarządzania procesami wytwarzania oprogramowania,
- ograniczeń finansowych, ograniczeń dotyczących np. harmonogramu,
- inne więzy i ograniczenia nie związane funkcjonalnością tworzonego systemu.

Standard **IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications,** określa następujące grupy wymagań niefunkcjonalnych

- **interfejsy zewnętrzne (external interfaces)**.
  Jak oprogramowanie oddziałuje na użytkowników, system operacyjny, sterowniki urządzeń, inne oprogramowanie?
  How does the software interact with people, the system Os hardware, other hardware, and other software?
- **wydajność (performance)**.
  Jaka jest szybkość, dostępność, czasy odpowiedzi, czas odtwarzania funkcji systemu?
  What is the speed, availability, response time, recovery time of various software functions?
- **atrybuty jakość oprogramowania (attributes)**.
  Jakie są własności systemu dotyczące przenoszalności, odpowiedniości, obsługiwalności, bezpieczeństwa?
  What are the portability, correctness, maintainability, security, etc. considerations?
- **więzy nałożone na metody projektowania i implementacji (design constraints).**
  Czy są określone wymagania dotyczące standardów inżynieryjnych, języków implementacji, procedur dla integracji baz danych, ograniczenia na zasoby, środowisko pracy modelowanego system?
  Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s)?

Inne wymagania według IEEE Std 830-1998 które powinny znaleźć się w SRS:

- koszty (cost),
- harmonogram dostaw produktów (delivery schedules),
- procedury raportowania (reporting procedures),
- metody i metodologie wytwarzania oprogramowania (software development methods),
- zapewnienie jakości (quality assurance),
- kryteria weryfikacji i walidacji systemu (validation and verifcation criteria),
- procedury akceptacyjne (acceptance procedures).

Poziomy ważności wymagań według IEEE Std 830-1998:

- **istotna (essential)**.
  Brak wymagania (funkcji systemu) oznacza brak akceptacji systemu.
  Implies that the software will not be acceptable unless these requirements are provided in an agreed manner.
- **warunkowa (conditional)**.
  Wymaganie (funkcja systemu) które można traktować jako ulepszenie systemu, bez którego system może być zaakceptowany.
  Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent.
- **opcjonalna (optional)**.
  Wymaganie (funkcje systemu) które mogą być dodatkowo zaimplementowane przez dostawcę sytemu , które wykraczają poza specyfikację wymagań.
  Implies a class of functions that may or may not be worthwhile. This gives the supplier the opportunity to propose something that exceeds the SRS.

Standard **ISO/IEC 25010:2011, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.**

Model jakości oprogramowania według standardu ISO/IEC 25010:2011 klasyfikuje wymagania według następujących kryteriów:

- **Funkcjonalność (functionality**).

  A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs. ***Suitability, Accuracy, Interoperability, Security, Functionality Compliance***.

  Zbiór atrybutów jakości które dotyczą funkcji systemu:
  - odpowiedniość (suitability), miara zgodności budowanego system z oczekiwaniami klienta.
  - dopasowanie (accuracy), miara zgodności systemu z uzgodnioną specyfikacją.
  - interoperacyjność (interoperability), miara możliwości systemu do integracji z innymi systemami.
  - bezpieczeństwo danych (security), miara możliwości systemu do zapewnienia integralności, poufności i uwierzytelnienia danych w systemie.

- **Niezawodność (reliability)**.

  A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. ***Maturity, Fault Tolerance, Recoverability, Reliability Compliance***.

  Zbiór atrybutów jakości które określają zdolności systemu do utrzymania wymaganego poziomu wydajności w określonych warunkach przez określony czas:
  - dojrzałość (maturity), zdolność systemu obsługi awarii spowodowanych błędami w oprogramowaniu,
  - odporność na błędy (fault tolerance), zdolność systemu do funkcjonowania w sytuacji istnienia błędów w oprogramowaniu.

  fault, bug - błąd w programie, defekt, przyczyna, która powoduje niewłaściwe działanie programu.
  - łatwość odzyskiwania/przywracania systemu do stanu działania (recoverability), zdolność systemu do przywrócenia go do działania w sytuacji zajścia awarii.
  - dostępność (availability), zdolność systemu realizacji swoich funkcji w określonej chwili czasu.

- **Użyteczność (usability)**.

  A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. ***Understandability, Learnability, Operability, Attractiveness, Usability Compliance***.

  Zbiór atrybutów jakości które określają nakłady na obsługę, naukę, używanie systemu:
  - łatwość zrozumienia (understandability), cecha systemu umożliwiająca użytkownikowi łatwe/szybkie zrozumienie czy system spełnia jego oczekiwania.
  - łatwość nauczenia się (learnability), cecha systemu umożliwiająca użytkownikowi łatwe/szybkie nauczenie się obsługi systemu.
  - łatwość posługiwania się (operability), cecha systemu umożliwiająca użytkownikowi kontrolowanie systemu.
  - atrakcyjność (likeability, attractiveness), miara przyjemność posługiwania się systemem.

- **Wydajność (efficiency)**.

  A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions. ***Time Behaviour, Resource Utilization, Efficiency Compliance***.

  Zbiór atrybutów jakości które określają relację między wydajnością i zużywanymi zasobami:
  - wydajność czasowa (time behavior), zdolność systemu do przetwarzania danych i odpowiedzi w określonym czasie.
  - wydajność pamięci (resource utilisation), zdolność systemu do użytkowania określonych zasobów (pamięć) przy realizowaniu swoich funkcji w określonym czasie.

- **Obsługiwalność (maintainability)**.

  A set of attributes that bear on the effort needed to make specified modifications.
  Analyzability, Changeability, Stability, Testability, Maintainability Compliance.
  Zbiór atrybutów jakości które określają nakłady na modyfikację, obsługę, naprawę sytemu:
  - łatwość analizy (analyzability),
  - modyfikowalność (changeability), łatwość implementacji modyfikacji system.
  - stabilność (stability), odporność systemu na niespodziewane/niewłaściwe zachowanie po modyfikacji.
  - łatwość testowania (testability), zdolność do łatwego testowania dokonanych zmian w systemie.

- **Przenoszalność (portability)**.

  A set of attributes that bear on the ability of software to be transferred from one environment to another. ***Adaptability, Installability, Co-Existence, Replaceability, Portability Compliance***.

  Zbiór atrybutów jakości które określają zdolność systemu do pracy w różnych środowiskach systemowych i aplikacyjnych:
  - łatwość dostosowania (adaptability), łatwość dostosowania system do pracy w różnych środowiskach.
  - łatwość instalowania (installability), łatwość instalacji systemu w danym środowisku.
  - co-existence, zdolność systemu funkcjonowania z innymi programami, wykorzystującymi wspólne zasoby.
  - zastępowalność (replaceability), możliwość zastąpienia innego system w jego środowisku pracy.

2. Zbieranie wymagań i tworzenie specyfikacji wymagań funkcjonalnych

Celem **specyfikacji wymagań** jest określenie wymagań klienta wobec tworzonego systemu komputerowego i określenie priorytetów ich realizacji.

Typy wymagań

- wymagania funkcjonalne,
- wymagania niefunkcjonalne,
- wymagania dotyczące jakości modelowanego systemu,
- więzy nałożone na budowany system, metody prowadzenia projektu.

Celem specyfikacji wymagań jest określenie

- W jakim celu klient zamawia dany system komputerowy?
- W jakim środowisku będzie pracował dany system komputerowy?
- Kto będzie użytkownikiem systemu?
- Jakie procesy funkcjonalne system powinien obsługiwać?
- Jakie są procesy i funkcje systemu?
- Kiedy system ma być dostarczony?
- Jaki jest dopuszczalny/maksymalny koszt systemu?
- Inne ograniczenia na system.

Dokumenty jakie mogą być podstawą do tworzenia specyfikacji wymagań

- kontrakt,
- opis wymagań dostarczony przez klienta (np. dokumentacja przetargowa),
- wywiad z klientem,
- korespondencja z klientem,
- inna dokumentacja, np. notatki ze spotkań, przedstawiony prototyp systemu.

Dokumenty jakie tworzą specyfikacje

- specyfikacja wymagań funkcjonalnych,
- specyfikacja wymagań niefunkcjonalnych
- specyfikacja wymagań dotyczących jakości modelowanego systemu,
- ograniczenia nałożone na projektowany system,
- wstępny plan projektu (plan dla fazy analizy i projektowania systemu),
- historia wszystkich podjętych decyzji.

Metody zbierania wymagań tworzonego systemu komputerowego

- specyfikacja funkcjonalna dostarczona przez klienta,
- wywiad z klientem,
- wykorzystanie dodatkowych źródeł informacji: eksperci, literatura, przegląd istniejących systemów o podobnej funkcjonalności i przeznaczeniu.

Zasady weryfikacji kompletności specyfikacji

- Czy specyfikacja określa wymagania klienta?
- Czy wymagania są sformułowane w sposób jednoznaczny?
- Czy specyfikacja jest kompletna?
- Czy specyfikacja określa zasady obsługi wyjątków i błędów?
- Czy specyfikacja jest spójna, tzn. nie ma sprzeczności między wymaganiami funkcjonalnymi, dotyczącymi jakości modelowanego systemu, więzami?
- Czy wymagania są realizowalne?

Struktura specyfikacji wymagań.

1. Wstęp
2. Cele specyfikacji
3. Zakres specyfikacji
4. Wymagania funkcjonalne
5. Wymagania niefunkcjonalne
   - 5.1 Interfejsy
     - 5.1.1 Interfejsy użytkownika
     - 5.1.2 Interfejsy sprzętowe
     - 5.1.3 Interfejsy komunikacyjne
     - 5.1.4 Interfejsy programowe
   - 5.2 Wymagania dotyczące jakości modelowanego systemu
   - 5.3 Warunki serwisowania (support)
   - 5.4 Ograniczenia architektury systemu
   - 5.4 Dokumentacja użytkownika
   - 5.5 Bazy danych
   - 5.6 Wymagania licencyjne
   - 5.7 Prawa autorskie i inne zagadnienia prawne
   - 5.8 Stosowane standardy
6. Opis więzów

Struktura specyfikacji wymagań (szablon).

1. Wstęp
[*Rozdział powinien zawierać ogólny opis celów, zakres, definicje skrótów i terminów użytych w specyfikacji, referencje do dokumentów do których odwołuje się specyfikacja.*]
2. Cele specyfikacji
[*Rozdział powinien zawierać szczegółowy opis celów specyfikacji, ogólny opis funkcjonalny systemu którego dotyczy.*]
3. Zakres specyfikacji
[*Rozdział powinien zawierać szczegółowy opis zakresu specyfikacji. Należy podać funkcje systemu, ogólną charakterystykę użytkowników systemu, wymagań niefunkcjonalnych, więzów.*]
4. Wymagania funkcjonalne
[*Rozdział powinien zawierać szczegółowy opis wymagań funkcjonalnych budowanego systemu.*]
5. Wymagania niefunkcjonalne
[*Rozdział powinien zawierać szczegółowy opis wymagań niefunkcjonalnych budowanego systemu.*]
5.1 Interfejsy
[*Rozdział powinien zawierać opis interfejsów jakie będą implementowane w tworzonym systemie.*]
5.1.1 Interfejsy użytkownika
[*Rozdział powinien zawierać szczegółowy opis interfejsów użytkownika modelowanego systemu.]*
5.1.2 Interfejsy sprzętowe
[*Rozdział powinien zawierać szczegółowy opis interfejsów sprzętowych systemu, w tym opis logicznej struktury interfejsów.*]
5.1.3 Interfejsy komunikacyjne
[*Rozdział powinien zawierać szczegółowy opis interfejsów komunikacyjnych systemu; karty sieciowe, modemy i opis stosowanych standardów.*]
5.1.4 Interfejsy programowe
[*Rozdział powinien zawierać szczegółowy opis interfejsów programowych systemu służących do komunikacji miedzy wewnętrznymi i zewnętrznymi komponentami systemu lub innymi zewnętrznymi systemami. Rozdział powinien zawierać szczegółowy opis standardów sieciowych, użytych protokołów, portów.*]
5.2 Wymagania dotyczące jakości modelowanego systemu
[*Rozdział powinien zawierać szczegółowy opis wymagań dotyczących jakości modelowanego systemu.*]
5.3 Warunki serwisowania (support)
[*Rozdział powinien zawierać szczegółowy opis wymagań dotyczących serwisowania systemu w tym narzędzia do zarządzania i utrzymania systemu.*]
5.4 Ograniczenia architektury systemu
[*Rozdział powinien zawierać szczegółowy opis wymagań i ograniczeń architektury systemu, w tym stosowane standardy programowania, języki programowania, stosowane konwencje nazewnictwa, stosowane klasy, komponenty.*]
5.4 Dokumentacja użytkownika
[*Rozdział powinien zawierać szczegółowy opis wymagań i ograniczeń dotyczących dokumentacji użytkownika w tym specyfikacje 'Pomocy' Aplikacji.*]
5.5 Bazy danych
[*Rozdział powinien zawierać wymagania dotyczące baz danych implementowanych w systemie.*]
5.6 Wymagania licencyjne
[*Rozdział powinien zawierać szczegółowy opis wymagań licencyjnych i ograniczeń nałożonych na tworzony system.*]
5.7 Prawa autorskie i inne zagadnienia prawne
[*Rozdział powinien zawierać szczegółowy opis wymagań dotyczących prawnych aspektów tworzonego systemu: praw autorskich, praw do patentów, praw dystrybucji, gwarancji.*]
5.8 Stosowane standardy
[*Rozdział powinien zawierać szczegółowy opis wymagań dotyczących stosowanych standardów przemysłowych i technologii w tworzonym systemie.*]
6. Opis więzów
[*Rozdział powinien zawierać szczegółowy opis więzów nałożonych tworzony system.*]

3. Analiza wymagań funkcjonalnych

Analiza wymagań tworzonego systemu komputerowego składa się z

- analizy wymagań funkcjonalnych, polega na tworzeniu modelu use case'ów,
- analizy wymagań niefunkcjonalnych,
- analizy wymagań dotyczących jakości modelowanego systemu,
- analizy więzów.

Analiza projektu uwzględnia wyniki analizy wymagań dotyczących jakości modelowanego systemu i analizy więzów.

We wstępnej fazie analizy wymagań można stworzyć **model 'obszarów problemowych'**,
Problem Domain Object Model (PDOM).
PDOM jest zbiorem wyobrażeń klienta o tworzonym systemie przedstawionym w formie obszarów problemów i związkach między problemami.

'Problem Domain Object Model' składa się z

- diagramu PDO (Problem Domain-Object Diagram),
- opisu PDO (Problem Domain-Object Description).

Definicja PDO:

- odzwierciedla wyobrażenia klienta o systemie,
- jest obiektem, nie jest klasą w sensie języka programowania,
- może wyobrażać podsystemy.

PDO jest elementem zbioru podstawowych pojęć ('key abstraction') tworzonego modelu programu komputerowego.

Proces tworzenia PDOM

- PDOM tworzony jest razem z klientem,
- na podstawie informacji od klienta, struktury już istniejących systemów, dobrego zrozumienia PDO budowany jest model systemu,
- model PDOM po ukończeniu nie jest zmieniany.

Etapy tworzenia modelu obszarów problemowych

- zrobić listę potencjalnych obszarów problemowych,
- zrobić krótki opis obszarów problemowych,
- wykonać wstępny diagram PDO wskazując na związki między obszarami problemowymi,
- dokładniej opisać obszary problemowe koncentrując się na roli i odpowiedzialności danego obszaru.

Dostarczone przez klienta lub przygotowana wspólnie z klientem specyfikacja wymagań podlega procesowi analizy.

W wyniku procesu analizy systemowej powstaje

- model use case'ów (model przypadków użycia),
- prototypy elementów systemu, np. interfejsy graficzne.

4. Model use case'ów

Model use case'ów jest obrazem funkcjonalnym systemu, nie zawiera wymagań dotyczących jakości modelowanego systemu, specyfikacji więzów, specyfikacji bazy danych.

Model use case'ów opisuje funkcjonalność systemu z punktu widzenia zewnętrznych użytkowników systemu, tzn. aktorów.

Model use case'u definiuje warunki zewnętrzne (względem systemu), tzn. aktorów i zadania jakie ma do wykonania system (lista use case'ów).

Kolejność tworzenia modelu use case'ów

- sporządzić listę aktorów (ról) w modelowanym systemie,
- sporządzić listę use case'ów (listę procesów funkcjonalnych dla każdego aktora),
- sporządzić diagram use case'ów,
- napisać use case'y.

Diagram use case'ów obrazuje zależności między use case'ami i aktorami.

Zasady tworzenia diagramu use case'ów

- na diagramie musi być kompletna lista use case'ów i aktorów.
- aktorzy i use case'y muszą mieć nazwy.
- aktorzy muszą być umieszczeni poza granicą systemu (aktor jest zewnętrznym elementem modelu w stosunku do systemu).
- use case może być połączony tylko z jednym aktorem, use case jest napisany dla konkretnego aktora, może zachodzić tylko relacja 1..*. Uwaga - aktor to rola w systemie.
- jeden aktor może być połączony z wieloma use case'ami (relacja 1..*).

Rola use case'ów modelowaniu systemu.

Use case opisuje oddziaływanie aktora z systemem, określając czynności jakie aktor wykonuje w celu realizacji specyfikowanej funkcjonalności.

Use case jest opisem sposobu użytkowania systemu polegającym na opisie jednego procesu w systemie (jednej funkcjonalności systemu).

Use case jest opisem procesu funkcjonalnego w formie interakcji aktora z systemem.

Use case'y są podstawą do tworzenia

- dokumentów w fazie analizy i projektowania systemu,
- dokumentacji użytkownika,
- test case'ów.

Use case'y tworzy się aby

- zrozumieć funkcjonalność systemu,
- zweryfikować rozumienie funkcjonalności z klientem,
- określić strukturę systemu: klasy, operacje i atrybuty,
- weryfikować i akceptować fazę analizy i projektowania,
- testować tworzony system (przygotowanie test case'ów).

Use case'y są zorientowane na procesy.
Maksymalna liczba use case'ów w modelu 20 - 30.

Standardowo istnieją trzy typy use case'ów

- opisujące standardowe procesy funkcjonalne,
- opisujące instalacje i zarządzanie systemem,
- opisujące procesy tła (monitorowanie urządzeń, systemu).

Proces tworzenia modelu use case'ów

1. Sporządzić listę aktorów z krótkim opisem każdego aktora.
2. Stworzyć listę use case'ów dla każdego aktora.
3. Dodać do use case'a cel i opis.
4. Dodać do use case'a warunki wstępne i stan końcowy.
5. Dodać do use case'a opis kroków, wyjątków, czynności alternatywnych.
6. Uwzględnić rozszerzenia (extension) i zawieranie (inclusion) use case'ów.
7. Narysować diagram use case'ów.

Proces funkcjonalny opisany w use case'e można opisać **diagramem aktywności lub diagramem stanu** systemu.

Struktura use case'u (szablon).

1. Nazwa use case'u
[*Opisowa nazwa use case'u, np. UzytkownikSystemu Loguje się do Systemu.*]
2. Wstęp
[*Rozdział określa role use case'a w modelu.*]
3. Cel use case'u
[*Rozdział zawiera charakterystyk use case'u, cele jakie use case realizuje. Należy podać krótki opis funkcjonalności która use case realizuje.*]
4. Lista aktorów
[*Należy podać listę aktorów biorących udział w realizacji use case'u. Listę należy sporządzić z podziałem na aktora inicjującego i aktorów uczestniczących (aktorów statycznych) w realizacji use case'u. Należy podać definicje i charakterystykę (role) aktora inicjującego i aktorów uczestniczących w use case. Opisać relacje między aktorami.*]
4.1 Aktor inicjujący
[*Nazwa aktora inicjującego use case.*]
4.2 Uczestnicy
[*Lista aktorów uczestniczących w realizacji use case'u.*]
5. Przebieg use case'u
[*Przebieg use case'u zapisany jest w formie numerowanych zdań (kroków). Każde zdanie musi zaczynać się od nazwy aktora lub słowa 'System'. Każde zdanie powinno opisywać pojedyncza czynność jaką wykonuje aktor lub zawierać opis reakcji systemu na działanie aktora. Przebieg use case'u jest forma dialogu miedzy aktorem a systemem. Należy unikać nic nie znaczących zwrotów, np. zamiast pisać 'Klient wprowadza dane do systemu' należy napisach 'Klient wprowadza Imię, Nazwisko, adres do systemu '. Klient jest aktorem - musi znajdować się na liście w rozdziale 'Aktorzy'. Na końcu use case'a można sporządzić słownik użytych terminów. Przy opisie kroków można dodatkowo posługiwać się rysunkami i diagramami.*]
Krok 1
[*Opis pierwszej czynności która aktor wykona.*]
Krok 2
[*Opis czynności która wykona system w odpowiedzi na działanie aktora.*]
…
Krok N
[*Opis czynności aktora w kroku N.*]
6 Wyjątki
<Wyjątek 1>
[*W tej części use case'u należy podać listę i przebieg sytuacji wyjątkowych (wyjątków). Wyjątkiem jest odstępstwo od głównego przebiegu use case'u opisanego w rozdziale 'Przebieg Use case'u' wynikające z nieoczekiwanego zachowania się aktora lub systemu. Szczególnym przypadkiem sytuacji wyjątkowej jest wystąpienie błędu, wówczas rozdział zawiera informacje o obsłudze błędu. Każdy wyjątek powinien być opisany w osobnym paragrafie.*]
7. Przebiegi alternatywne
<Przebieg Alternatywny 1>
[*W tej części use case'u należy podać listę i przebieg alternatywny dla use case'u. Opis przebiegu alternatywnego powinien być zapisany w formie numerowanych zdań. Pierwsze zdanie w opisie alternatywnym powinno mieć numer zdania z głównego przebiegu use case'u od którego zaczyna się przebieg alternatywny. Jeżeli przebieg alternatywny jest dostatecznie długo można zawrzeć go w osobnym use case'ie i podać do niego referencje.*]
8. Zagadanienia implementacyjne
[*Ten rozdział zawiera wymagania dotyczących jakości modelowanego systemu jakie należy spełnić przy implementacji danej funkcjonalności w systemie. Np. wymagania określające wydajność, liczbę użytkowników realizujących jednocześnie daną funkcjonalność (min. liczba 'concurrent users'), kwestie bezpieczeństwa (dostępu do konkretnych zasobów, szybkości reakcji systemu na działania aktora (użytkownika).*]
9. Warunki rozpoczęcia use case'u
[*Należy opisać warunki wstępne jakie należy spełnić aby można było wykonać use case w rzeczywistym systemie. Np. przygotować dane które podczas wykonywania use case'u będą wprowadzane do systemu, odpowiednio skonfigurować system (utworzyć użytkownika, poznać hasło dostępu, ...).*]
10. Stan końcowy
[*Należy opisać wynik końcowy, stan systemu po realizacji use case'u.*]
11. Nierozwiązane problemy
[*W tym rozdziale należy opisać nierozwiązane problemy, pytania jakie powstały podczas pisania use case'u, np. problemy implementacji use case'u*]

Przykład. Use case: wypłata gotówki z bankomatu.

1. Nazwa use case'u: Klient banku wypłaca gotówkę z bankomatu.
2. Wstęp
3. Cel use case'u
4. Lista Aktorów
4.1 Aktor inicjujący: Klient banku
4.2 Uczestnicy: system bankowy, system autoryzacyjny.
5. Przebieg use case'a
Krok 1
Klient banku wkłada kartę do bankomatu.
Bankomat sprawdza dane na karcie bankomatowej.
Bankomat żąda wprowadzenia PIN'u.
Krok 2
Klient banku wprowadza PIN.
Bankomat weryfikuje PIN w systemie autoryzacyjnym.
Bankomat żąda wyboru operacji.
Krok 3
Klient banku wybiera operację 'Wypłata gotówki'.
Bankomat żąda wyboru kwoty operacji.
Krok 4
Klient banku wybiera kwotę operacji.
Bankomat sprawdza saldo na rachunku w systemie bankowym.
Bankomat żąda wyboru opcji: wydruk / rezygnacja z wydruku ‘Potwierdzenia Wypłaty’.
Krok 5
Klient banku wybiera opcje: rezygnacja z wydruku 'Potwierdzenia Wypłaty'.
Bankomat zwraca kartę bankomatową.
Krok 6
Klient banku odbiera kartę bankomatową.
Bankomat wypłaca żądaną kwotę gotówki.
Krok 7
Klient banku odbiera gotówkę z bankomatu.
6.Wyjatki
Wyjątek 1. Wprowadzenie nieprawidłowego PIN'u.
Krok 3.
Bankomat nie może zweryfikować PIN'u w systemie autoryzacyjnym (PIN niezgodny).
Bankomat informuje o niezgodności PIN'u i ponownie żąda wprowadzenia 'poprawnego PIN'u'.
Wyjątek 2. Brak środków na rachunku
Krok 5
Bankomat informuje o braku środków na rachunku.
7. Przebieg Alternatywny.
Przebieg Alternatywny 1. Wybór opcji: wydruk 'Potwierdzenia Wypłaty'.
Krok 5
Bankomat żąda wyboru opcji: wydruk / rezygnacja wydruku 'Potwierdzenia Wypłaty'.
Klient banku wybiera opcję: wydruk 'Potwierdzenia Wypłaty'.
Krok 8
Bankomat drukuje 'Potwierdzenie Wypłaty'.
Klient banku odbiera 'Potwierdzenie Wypłaty'.
8. Problemy Implementacji
Brak.
9. Warunki Rozpoczęcia
Posiadanie karty bankomatowej, znajomość PIN'u, niezerowe saldo na koncie.
10. Stan Końcowy
Klient banku wypłacił żądana ilość gotówki, odebrał kartę.
Przebieg Alternatywny 1: wydrukowany paragon 'Potwierdzenie Wypłaty'.
11. Nierozwiązane Problemy
Brak.