

**MINISTERSTWO SPRAWIEDLIWOŚCI**  
**DEPARTAMENT INFORMATYZACJI I REJESTRÓW SĄDOWYCH**

**METODYKA WYTWARZANIA OPROGRAMOWANIA**  
**WERSJA .....**

<b>MINISTERSTWO SPRAWIEDLIWOŚCI</b> <b>DEPARTAMENT INFORMATYZACJI I REJESTRÓW SĄDOWYCH</b>						
<b>Dokument</b>	<b>METODYKA WYTWARZANIA OPROGRAMOWANIA W PROJEKTACH IT W  RESORCIE SPRAWIEDLIWOŚCI</b>					
<b>Krótki opis dokumentu</b>	<p>Niniejszy dokument szczegółowo definiuje i normalizuje zasady modelowania i dokumentowania systemów informatycznych w zakresie :</p> <p style="padding-left: 40px;"> Analizy Biznesowej,  Analizy Wymagań,  Analizy Systemowej  Projektowania i Dokumentowania </p>					
<b>Opracowano na podstawie</b>	Zasad doświadczeń w przy budowie rozwiązań teleinformatycznych z DIRS w ciągu ostatnich 2 lat..					
<b>Właściciel dokumentu</b>	Architekt					
<b>Weryfikacja merytoryczna</b>	Imię i nazwisko, stanowisko	, Ekspert – Koordynator WPU,	Data		Podpis	
<b>Weryfikacja merytoryczna</b>	Imię i nazwisko, stanowisko	Architekt	Data		Podpis	
<b>Zatwierdził</b>	Imię i nazwisko, stanowisko	, Dyrektor Departamentu	Data		Podpis	
<b>Data druku</b>					Liczba stron	
<b>Nazwa pliku</b>	.docx				Status	z (zatwierdzony)

# Spis treści

1. METRYKA DOKUMENTU .....	4
2. CEL DOKUMENTU .....	5
2.1. Przeznaczenie dokumentu .....	5
2.2. Wykorzystanie .....	5
2.3. Słownik pojęć i skrótów .....	5
3. ZAŁOŻENIA .....	7
3.1. Podejście – modele.....	7
3.2. Standardy i notacje.....	7
3.3. Narzędzia .....	7
4. DYSCYPLINY W PROCESIE WYTWÓRCZYM .....	8
5. OGÓLNE ZAŁOŻENIA DOTYCZĄCE MODELOWANIA SYSTEMU .....	9
5.1. Przyjęte konwencje opisu.....	9
6. ANALIZA BIZNESOWA.....	11
7. ANALIZA WYMAGAŃ .....	12
7.1. Modelowanie wymagań .....	12
8. ANALIZA SYSTEMOWA .....	14
8.1. Model przypadków użycia .....	14
8.2. Model zachowania przypadku użycia .....	14
8.2.1. Kontrakt przypadku użycia .....	15
8.2.2. Przebieg przypadku użycia .....	15
8.3. Model dziedziny systemu oraz model słowników i parametrów .....	16
8.4. Wzory dokumentów .....	17
8.5. Model interfejsu użytkownika .....	18
9. PROJEKTOWANIE.....	19
9.1. Model komponentów.....	19
9.2. Model interakcji.....	21
9.3. Fizyczny model danych.....	21
9.4. Model wdrożenia (rozmieszczenia).....	23
10. DOKUMENTACJA PROJEKTOWA.....	24
10.1. Mapowanie modeli na dokumenty projektowe.....	24

# 1. Metryka dokumentu

---

Tytuł dokumentu	
Wersja	
Data wydania	
Dotyczy aplikacji	
Sygnatura dokumentu	

Historia zmian			
Wersja	Data	Autor	Opis zmian
1.0			

## 2. Cel dokumentu

---

### 2.1. Przeznaczenie dokumentu

---

Niniejszy dokument został przygotowany przez zespół analityków i projektantów pracujących w projekcie przy projektach informatycznych i jego celem jest:

- zdefiniowanie metodyki wytwarzania oprogramowania w zakresie analizy i projektowania,
- jednoznaczne wskazanie elementów notacji UML stosowanych w poszczególnych fragmentach dokumentacji analitycznej i projektowej przekazywanej do klienta.

Dokument będzie uzupełniany, zmieniany i rozwijany w każdym przypadku, kiedy zajdzie taka potrzeba.

### 2.2. Wykorzystanie

---

Dokument nie mówi które modeli należy wykorzystać w konkretnym projekcie IT wskazuje tylko te diagramy które zostały ustalone jako standard modelowania oprogramowania w projektach realizowanych Ministerstwie Sprawiedliwości

### 2.3. Słownik pojęć i skrótów

---

Lista pojęć i skrótów używanych w dokumencie.

BPMN	- Business Process Model and Notation (Notacja i Model Procesu biznesowego) – graficzna notacja służąca do opisywania procesów biznesowych
CIM	- Computation Independent Model – poziom biznesowy modelu systemu w podejściu MDA, niezależny od informatyki.
Diagram	- uproszczona reprezentacja graficzna, służy do opisanie modelu; dany model może być opisany przy pomocy wielu diagramów, zaś dany element modelu może pojawiać się na wielu diagramach opisujących ten model.
EA	- Enterprise Architect - narzędzie do modelowania głównie za pomocą UML tworzone przez firmę Sparx Systems
Komponent funkcjonalny	- Element architektury funkcjonalnej w domenach aplikacji i danych obejmujący funkcjonalność o jawnie określonych granicach oraz podlegający niezależnemu zarządzaniu.
MDA	- Model Driven Architecture – pojęcie z dziedziny inżynierii oprogramowania, określające zbiór metod porządkujących proces tworzenia systemów komputerowych opartych na budowie modeli i ich transformacji. Koncepcja MDA opracowana została przez grupę OMG
Model systemu	- Semantycznie spójna abstrakcja projektowanego systemu, która stanowi kompletny opis systemu utworzony z określonej perspektywy na pewnym poziomie szczegółowości, co oznacza, że niektóre elementy systemu zostały ukryte, a inne wyeksponowane. Fraza "kompletny opis" zaświadcza, że żadna dodatkowa informacja nie jest potrzebna dla zrozumienia systemu z danej perspektywy. Pojedynczy

model zazwyczaj nie wystarcza ani do zrozumienia wszystkich aspektów złożonego systemu jednocześnie ani do znalezienia odpowiedniego rozwiązania, zwykle potrzebujemy ich wiele. Razem stanowią kompletny opis systemu. Model może być utworzony za pomocą określonych narzędzi graficznych, formalnych notacji, ale także z użyciem języka naturalnego.

- |     |   |  |
|-----|---|--|
| MS  | - | Ministerstwie Sprawiedliwości  |
| OMG | - | Object Management Group – międzynarodowe otwarte konsorcjum powstałe w 1989 r., którego celem jest tworzenie i pielęgnowanie standardów technologicznych w obszarze IT.  |
| PIM | - | Platform Independent Model – logiczny poziom (perspektywa) modelu systemu (niezależny od platformy) w podejściu MDA  |
| PSM | - | Platform Specific Model – techniczny poziom (perspektywa) modelu systemu w podejściu MDA   |
| UML | - | Unified Modeling Language - język formalny wykorzystywany do modelowania różnego rodzaju systemów, stworzony przez Grady Boocha, Jamesa Rumbaugh oraz Ivara Jacobsona, obecnie rozwijany przez Object Management Group (OMG) |

## 3. Założenia

### 3.1. Podejście – modele

---

Proces wytwarzania oprogramowania w MS opiera na podejściu MDA (Model Driven Architecture). Wg koncepcji MDA w procesie tworzenia oprogramowania wymagane są następujące elementy:

- jednoznaczne zrozumienie stosowanych pojęć i określeń dziedzinowych;
- zrozumienie tematu na poziomie niezależnym od środków informatyki (CIM – Computation Independent Model);
- modelowanie niezależne od platformy (PIM – Platform Independent Model);
- możliwie automatyczne odwzorowanie (transformacje) z poziomu PIM do PSM (Platform Specific Model);
- posługiwanie się jednoznacznymi, powtarzalnymi elementami projektowymi; np. wzorce projektowe.

Model CIM – opisuje system z perspektywy biznesowej, w oderwaniu od informatyki.

Model PIM – opisuje logikę systemu (co system będzie realizował i w jaki sposób), ale w oderwaniu od konkretnej platformy informatycznej (język oprogramowania, technologia),

Model PSM – opisuje sposób implementacji systemu z zastosowaniem konkretnych technologii.

Te trzy modele są odpowiednio wynikiem aktywności w trzech dyscyplinach procesu wytwarzania oprogramowania:

- analizy biznesowej (model CIM),
- analizy systemowej (model PIM),
- projektowania (model PSM).

### 3.2. Standardy i notacje

---

W wymienionych dyscyplinach procesu wytwarzania oprogramowania i wynikających z nich modelach MS stosuje poniższe standardy (notacje, wzorce):

- Unified Modeling Language (UML), w wersji 2.4
- Business Process Model and Notation (BPMN), w wersji 2.0
- Wzorce i zasady projektowania obiektowego: SOLID oraz PoPCC

Ponadto w niektórych modelach (w szczególności: Wzory dokumentów, Model interfejsu użytkownika) nie jest wykorzystywana żadna z wymienionych notacji, natomiast są używane inne środki wyrazu udostępniane przez używane narzędzie. Są one opisane w odpowiednich rozdziałach dotyczących poszczególnych modeli.

### 3.3. Narzędzia

---

W projekcie IT jako narzędzie do modelowania wykorzystywane jest narzędzie Sparx Systems Enterprise Architect (EA) w wersji 13.5 lub wyższej.

## 4. Dyscypliny w procesie wytwórczym

---

Proces wytwarzania oprogramowania obejmuje dyscypliny inżynierskie:

- Analiza biznesowa
- Analiza wymagań
- Analiza systemowa
- Projektowanie,
- Implementacja
- Testowanie
- Dokumentowanie
- Wdrożenie

i pomocnicze:

- Zarządzanie konfiguracją i zmianami,
- Zarządzanie projektem.

Niniejszy dokument w następnych rozdziałach definiuje, jakie rodzaje modeli i diagramów oraz jakie środki wyrazu mogą być użyte w ramach aktywności w poszczególnych dyscyplinach procesu wytwórczego oprogramowania w projekcie IT, a także, jakie powiązania muszą istnieć pomiędzy różnymi elementami modelu, aby było możliwe śledzenie realizacji potrzeb i wymagań biznesowych w całym procesie wytwórczym.



## 5. Ogólne założenia dotyczące modelowania systemu

---

Niniejszy rozdział zbiera informacje wspólne, dotyczące wszystkich bądź większości elementów i modeli.

1. Elementy modelu muszą być identyfikowalne i unikalne w ramach modelu. Unikalność może być zapewniona poprzez nazwę lub poprzez nadanie dodatkowego identyfikatora.
2. Dla przejrzystości dokumentacji analitycznej należy stosować do modelowanych elementów jednolite konwencje nazewnictwa.
3. Akcje odpowiadające czynnościom wykonywanym w ramach przebiegu powinny być nazwane zdaniem pojedynczym, którego podmiot wskazuje na odpowiedzialnego za wykonanie tej akcji lub równoważnikiem zdania.
4. Nazwy elementów strukturalnych powinny być rzeczownikami z liczbie pojedynczej, nawet jeśli wiadomo, że chodzi o kolekcję obiektów.
5. Każdy element modelu (typ) powinien posiadać ogólny opis umieszczony we właściwościach elementu w EA (własność *Notes*). Nie dotyczy to konektorów używanych na diagramach.
6. Dla elementów typowanych (np. atrybutów klas) nie powinno się określać typu elementu przez nazwę, ale poprzez powiązanie w EA z właściwym typem.
7. Stereotypy używane w modelach (poza stereotypami standardowymi zdefiniowanymi w specyfikacji UML) powinny być zdefiniowane i opisane w odpowiednich profilach UML.
8. Na każdym diagramie w celu ułatwienia zrozumienia treści diagramu mogą być umieszczane notatki (element *Notes* w EA).

Przy czym, jeśli elementy są już zamodelowane w dotychczasowych modelach, to nie są wymagane ich zmiany w celu dostosowania do powyższych zasad.

Natomiast nowe elementy modelu tworzone lub w znaczący sposób zmieniane przez MS i podlegające modyfikacji w ramach projektu będą tworzone zgodnie z powyższymi ogólnymi zasadami oraz zasadami opisanymi dla poszczególnych modeli.

### 5.1. Przyjęte konwencje opisu

---

W tabelach specyfikujących elementy wykorzystywane w poszczególnych modelach oraz relacje pomiędzy nimi przyjęto następujące konwencje:

- W pierwszej linii pojawia się nazwa elementu w języku polskim
- W drugiej linii (na szarym tle) pojawia się nazwa elementu w języku angielskim, jest to nazwa elementu UML lub nazwa elementu w EA (dla elementów spoza notacji UML),
- W kolejnej linii – pojawia się nazwa stereotypu dla elementu, jeśli ma zastosowanie.
- Nazwy relacji w tabelach podane są w języku polskim. Poniżej wyspecyfikowano używane nazwy relacji w języku polskim i odpowiadające im nazwy z notacji UML lub nazwy relacji w EA (dla relacji spoza UML).

<b>Generalizacja</b>	Generalization
<b>Asocjacja</b>	Association
	Realization
<b>Realizacja</b>	InterfaceRealization
	ComponentRealization
<b>Użycie</b>	Usage
<b>Śladowanie</b>	«Trace» Abstraction
<b>Wnioskowanie</b>	«Derive» Abstraction
<b>Zależność</b>	Dependency

<b>Wiązanie</b>	TemplateBinding
<b>Włączanie</b>	Include
<b>Rozszerzanie</b>	Extend
<b>Komunikacja</b>	CommunicationPath
<b>Rozmieszczenie</b>	Deployment
<b>Manifestacja</b>	Manifestation
<b>Przynależność</b>	Element::owner
	ActivityNode::inPartition
	Umieszczenie w drzewie projektu wewnątrz elementu posiadacza
<b>Posiadanie</b>	Element::ownedElement
	ActivityPartition::node
	Umieszczenie w drzewie projektu wewnątrz elementu wszystkich elementów posiadanych
<b>Typ</b>	TypedElement::type
	Określenie Type, Behavior, Instance Classifier
<b>Klasyfikator</b>	InstanceSpecification::classifier
	Określenie Instance Classifier
<b>Reprezentacja</b>	ActivityPartition::represents
	Lifeline::represents
	Określenie Instance Classifier
<b>Wartości wejściowe</b>	OpaqueAction::inputValue
	Posiadany Element <i>Action pin</i> z kierunkiem <i>in</i>
<b>Wartości wyjściowe</b>	OpaqueAction::outputValue
	Posiadany Element <i>Action pin</i> z kierunkiem <i>out</i>
<b>Cel</b>	CallOperationAction::target
	Posiadany Element <i>Action pin</i> z kierunkiem <i>in</i> o odpowiednim typie
<b>Atrybut</b>	Class::ownedAttribute
	Interface::ownedAttribute
	DataType::ownedAttribute
<b>Wysłanie</b>	Message::sendEvent
<b>Odebranie</b>	Message::receiveEvent
<b>Sygnatura</b>	Message::signature

## 6. Analiza biznesowa

---

Analiza biznesowa może być wykonywana przez dostawcę oprogramowania w różnym zakresie, w zależności od rozmiaru przewidywanej modyfikacji oprogramowania, zmiany w istniejących procesach biznesowych, specyfiki umowy itp.

W szczególnym przypadku, dla konkretnej modyfikacji może nie powstać model biznesowy.

W zakresie analizy biznesowej tworzone mogą być dwa rodzaje modeli

- Model informacyjny (na poziomie biznesowym)
- Model procesów biznesowych
- Dodatkowo Model Stanu (**na poziomie biznesowym -obiekty**).

Model informacyjny jest tworzony, jeśli na poziomie biznesowym istnieje potrzeba zdefiniowania pojęć występujących w danej dziedzinie oraz powiązań w między nimi. Stosowana jest w tym celu notacja UML, a konkretnie diagramy klas.

Model stanu jest tworzony, jeśli na poziomie biznesowym istnieje potrzeba zdefiniowania stanów i podstawów obiektu/ systemu pojęć występujących w danej dziedzinie oraz powiązań w między nimi. Stosowana jest w tym celu notacja UML, a konkretnie diagramy maszyny stanowej.

Model procesów biznesowych, jeśli wymagany, jest tworzony w notacji BPMN z wykorzystaniem:

- Diagramu procesu prywatnego – dla zamodelowania procesu wewnętrznego,
- Diagramu procesu publicznego lub diagramu współpracy (kolaboracji) - dla zamodelowania procesu, w którym uczestniczy podmiot zewnętrzny.

Przy czym w przypadku diagramów współpracy uczestnik spoza organizacji może być zamodelowany jako basen zwinięty – bez pokazywania szczegółów przebiegu procesu u tego uczestnika.

## 7. Analiza wymagań

---

MS stosuje klasyfikację wymagań zaproponowaną w BABOK (Business Analysis Body of Knowledge), czyli trzy poziomy wymagań:

- Wymagania biznesowe – czyli potrzeby biznesowe, wynikające z celu biznesowego. Wymagania te realizują podstawowy cel stawiany przed systemem. Są produktem analizy strategicznej,
- Wymagania interesariuszy/ użytkowników – czyli konkretne potrzeby udziałowców/użytkowników systemu lub opis funkcji. Wymagania te realizują wymagania biznesowe. Są produktem analizy wymagań.
- Wymagania rozwiązania – czyli opis produktu będącego rozwiązaniem zapisany w języku zrozumiałym dla programistów. Wymagania te realizują wymagania użytkowników. Stanowią rozwiązanie problemu lub realizują cel.

Z kolei wymagania rozwiązania podzielone są na dwie kategorie:

- Wymagania funkcjonalne,
- Wymagania нефunkcjonalne.

Dodatkowo mogą pojawić się tzw. wymagania przejścia, które wynikają z konieczności dodatkowych działań związanych z uruchomieniem nowego systemu/ nowej wersji systemu. Może to być np. migracja danych, zmiany w słownikach systemowych itp.

### 7.1. Modelowanie wymagań

---

Notacja UML nie obejmuje modelowania elementów takich jak wymaganie. Jednak narzędzie EA umożliwia modelowanie wymagań za pomocą elementów *Requirement*.

W modelu powinny się znaleźć co najmniej wymagania rozwiązania, aby móc pokazać, w jaki sposób elementy rozwiązania realizują te wymagania.

Jeśli to możliwe – są zdefiniowane wymagania biznesowe oraz wymagania interesariusza, to także te wymagania powinny zostać umieszczone w modelu.

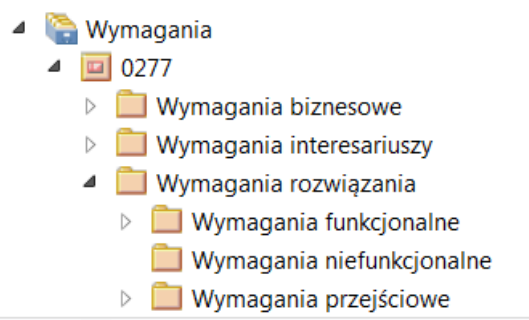
Każde wymaganie w modelu powinno mieć zdefiniowane następujące atrybuty:

- Nazwa,
- Opis (treść),
- Unikalny identyfikator
- Typ (funkcjonalne, нефunkcjonalne, przejściowe, interesariusza, biznesowe)
- Sposób odbioru testu wymagania
- Wersja

Dodatkowo wymagania niższego poziomu powinny być powiązane relacją realizacji (Realization) z odpowiednim wymaganiem wyższego poziomu (jeśli zostało zdefiniowane).

Ponadto mogą zostać zdefiniowane zależności pomiędzy wymaganiami tego samego poziomu. Jeśli takie zależności istnieją, powinny być zobrazowane w modelu z użyciem relacji zależności (Dependency).

Wymagania w projekcie powinny być także ustrukturyzowane, co najmniej w zakresie zaprezentowanym na poniższym rysunku.



Rysunek 1. Przykład struktury wymagań w modelu

Tabela 1 Powiązania elementów Modelu wymagań

Wymaganie Requirement					Wymaganie powiązane
Biznesowe Business	Interesariusza Stakeholder	Funkcjonalne Functional	Niefunkcjonalne Non-functional	Przejściowe Transition	
zależność →	realizacja →	x	x	x	Biznesowe
x	zależność →	realizacja →	realizacja →	realizacja →	Interesariusza
x	x	zależność →	zależność →	zależność →	Funkcjonalne
x	x	zależność →	zależność →	zależność →	Niefunkcjonalne
x	x	x	x	zależność →	Przejściowe

## 8. Analiza systemowa

W ramach analizy systemowej w projekcie mogą być tworzone (w zależności od potrzeb) następujące modele:

- Model przypadków użycia
- Model zachowania przypadków użycia
- Model dziedziny systemu (logiczny model danych)
- Model słowników i parametrów systemowych
- Wzory dokumentów
- Model interfejsu użytkownika

### 8.1. Model przypadków użycia

Model przypadków użycia wykorzystuje diagramy przypadków użycia zdefiniowane w UML i obejmuje następujące rodzaje elementów:

- Przypadek użycia,
- Aktor,
- Relacje (include - włączanie, extend - rozszerzanie, association - asocjacja, generalization - generalizacja)

Każdy przypadek użycia musi posiadać unikalną nazwę/ identyfikator oraz opis specyfikujący jego cel oraz dodatkowe informacje pozwalające zrozumieć kontekst jego użycia.

Zarówno dla aktorzy, jak i przypadki użycia mogą podlegać klasyfikacji (podziałowi na rodzaje) poprzez zdefiniowanie odpowiednich stereotypów.

Poniższa tabela definiuje dopuszczalne relacje (powiązania) w modelu przypadków użycia.

W szczególności należy zwrócić uwagę na powiązania z elementami innych modeli:

- Przypadek użycia realizuje wymaganie funkcjonalne,
- Przypadek użycia realizuje (wspiera realizację) zadanie procesu biznesowego.

Tabela 2 Powiązania elementów Modelu przypadków użycia

Element modelu		Element powiązany
Przypadek użycia UseCase	Aktor Actor	
włączanie→ rozszerzanie→ generalizacja→	asocjacja→	Przypadek użycia
asocjacja→	generalizacja→	Aktor
realizacja→	realizacja→	Wymaganie rozwiązania: Funkcjonalne
zależność→	x	Klasa modelu dziedziny
realizacja→	x	Zadanie procesu biznesowego
x	typ→	Partycypant (z diagramu sekwencji)

### 8.2. Model zachowania przypadku użycia

Model zachowania przypadku użycia obejmuje dwa aspekty:

- Kontrakt przypadku użycia,

- Przebieg przypadku użycia.

Dla zamodelowania obu aspektów wykorzystywane są diagramy aktywności (zachowania) zdefiniowane w UML oraz następujące główne rodzaje elementów:

- Aktywność,
- Parametr aktywności,
- Partycja
- Akcja
- Pin
- ControlFlow – jako krawędź aktywności

Przy czym stosowanie partycji nie jest obligatoryjne, jeśli wykonawca akcji jest wyspecyfikowany w polu Alias elementu akcji.

Spośród dostępnych rodzajów akcji (w UML oraz w EA) stosowane są akcje: CallBehaviorAction, CallOperationAction, OpaqueAction (w EA akcja rodzaju Atomic).

Dodatkowo dla przypadku użycia specyfikowane są warunki początkowe i końcowe stanowiące element kontraktu przypadku użycia oraz reguły systemowe stanowiące uzupełnienie przebiegu przypadku użycia. Warunki brzegowe i reguły systemowe są definiowane jako własności Constraint w przypadku użycia.

### 8.2.1. Kontrakt przypadku użycia

Kontrakt przypadku użycia definiuje jego parametry wejściowe i wyjściowe oraz warunki brzegowe (początkowe i końcowe). Jest on a zamodelowany jako diagram aktywności zawierający element Activity odpowiadający zachowaniu przypadku użycia, którego dotyczy. Zarówno diagram, jak i element Activity umieszczone są w przestrzeni przypadku użycia (wewnątrz).

Poniższa tabela definiuje powiązania elementów kontraktu przypadku użycia z elementami innych modeli.

Tabela 3 Powiązania elementów kontraktu przypadku użycia

Element modelu		Element powiązany
Aktywność Activity	Parametr aktywności ActivityParameter	
x	przynależność [1]	Aktywność
posiadanie	x	Parametr aktywności
x	typ	Typ danych (z modelu dziedziny)
x	typ	Klasa, Słownik / Parametr systemowy z modelu dziedziny)
przynależność [1]	x	Przypadek użycia (posiadacz kontraktu)

### 8.2.2. Przebieg przypadku użycia

Przebieg przypadku użycia modelowany jest na diagramie aktywności umieszczonym wewnątrz elementu Activity reprezentującego zachowanie przypadku użycia z użyciem elementów opisanych wyżej.

Poniższa tabela prezentuje powiązania elementów modelu przebiegu przypadku użycia z elementami innych modeli.

Tabela 4 Powiązania elementów Modelu zachowania przypadku użycia

Element modelu	Element powiązany
----------------	-------------------

Akcja (opisowa) Action	Akcja CallBehavior CallBehaviorAction	Akcja CallOperation CallOperationAction	Pin Pin	Partycja Partition	
x	x	x	x	posiadanie	Akcja (opisowa)
x	x	x	x	posiadanie	Wywołanie zachowania
x	x	x	x	posiadanie	Wywołanie operacji
wartości wejściowe i wyjściowe	argumenty wyniki	cel [1] argumenty wyniki	x	x	Pin
przynależność [1]	przynależność [1]	przynależność [1]	x	posiadanie przynależność	Partycja
x	x	operacja [1]	x	x	Operacja interfejsu logicznego (z Modelu komponentów)
x	x	x	x	reprezentacja	Aktor osobowy (Z Modelu przypadków użycia)
x	x	typ celu	typ	reprezentacja	Aktor systemowy (Z Modelu przypadków użycia)
x	zachowanie [1]	x	x	x	Aktywność
x	x	x	typ	x	Klasyfikator (z Modelu dziedziny systemu)
zależność→	x	x	x	x	Formularz (z Modelu interfejsu użytkownika)
zależność→	x	x	x	x	Szablon i zakres informacyjny dokumentu (z Modelu Wzorów dokumentów)

## 8.3. Model dziedziny systemu oraz model słowników i parametrów

Model dziedziny systemu obejmuje modelowanie:

- klas i ich cech strukturalnych,
- typów danych,
- związków pomiędzy klasami,
- maszyny stanów dla obiektów klasy,
- architektury modelu dziedziny.

W modelu dziedziny systemu wykorzystywane są diagramy klas zdefiniowane w UML i następujące główne elementy tych diagramów:

- Typ danych,
- Typ prymitywny (typy prymitywne zdefiniowane w specyfikacji UML)
- Enumeracja,
- Klasa,
- Relacje (asocjacja – z uwzględnieniem kompozycji i agregacji, generalizacja)

Te same środki wyrazu wykorzystywane są w modelu słowników i parametrów. Klasy będące reprezentacją słowników lub parametrów systemów są dociążane odpowiednim stereotypem



(«Dictionary», «SystemParameter»). Dodatkowo klasy reprezentujące słowniki lub parametry systemowe mogą być uzupełnione o specyfikację wartości słownikowych i parametrów systemowych (z wykorzystaniem LinkedDocument lub elementów Artifact Internal w EA).

Klasy definiowane są z dokładnością do atrybutów, ich liczności i typów oraz ewentualnych dodatkowych ograniczeń).

Poniższa tabela prezentuje wzajemne powiązania elementów modelu dziedziny systemu oraz modelu słowników i parametrów.

Tabela 5 Powiązania elementów Modelu dziedziny systemu

Element modelu				Element powiązany
Klasa Class	Typ danych DataType	Słownik / Parametr systemowy Class «Dictionary» «SystemParameter»	Enumeracja Enumeration	
generalizacja→ asocjacja→ typ parametru operacji typ atrybutu	x	x	x	Klasa
typ atrybutu typ parametru operacji	generalizacja→ asocjacja→ typ atrybutu typ parametru operacji	typ atrybutu	x	Typ danych
asocjacja→ typ parametru operacji	x	generalizacja→ asocjacja→	x	Słownik / Parametr systemowy
typ atrybutu typ parametru operacji	typ atrybutu typ parametru operacji	typ atrybutu	generalizacja→	Enumeracja

Dodatkowym aspektem modelowania dotyczącym modelu dziedziny jest cykl życia obiektu klasy modelu dziedziny. Do zamodelowania tego aspektu wykorzystywany jest diagram maszyny stanów zdefiniowany w UML oraz jego następujące główne elementy:

- Maszyna stanów (StateMachine),
- Stan (State),
- Przejście (Transition).

## 8.4. Wzory dokumentów

Każdy generowany przez system dokument, niezależnie od rodzaju (pismo, raport itp.), powinien być tworzony zgodnie z określonym wzorem.

W modelu Wzory dokumentów pismo lub raport jest definiowane w postaci pary powiązanych ze sobą artefaktów (element Document Artifact w EA):

- szablonu pisma/ raportu,
- zakresu informacyjnego pisma/ raportu.

Szablon pisma definiuje jego stałą treść i wygląd oraz zawiera znaczniki określające elementy zmienne. Natomiast Zakres informacyjny pisma definiuje sposób ustalania/ wyliczania treści elementów zmiennych wskazanych w szablonie przez znaczniki.

W EA dla zamodelowania tego zakresu systemu wykorzystywany jest diagram dokumentacji (spoza specyfikacji UML) oraz wspomniane elementy Document Artifact. Odpowiednie definicje szablonu i zakresu informacyjnego umieszczane są w Linked Document tych elementów.

## 8.5. Model interfejsu użytkownika

---

Model interfejsu użytkownika obejmuje dwa aspekty:

- Mapę nawigacji, czy sposób nawigowania pomiędzy poszczególnymi formularzami (oknami, ekranami) aplikacji,
- Definicję poszczególnych formularzy interfejsów użytkownika.

W EA do tego modelu wykorzystywany jest diagram interfejsu użytkownika (User Interface Diagram) spoza specyfikacji UML oraz elementy Screen dostępne dla tego diagramu.

Przejścia pomiędzy poszczególnymi formularzami modelowane są z użyciem relacji trace nazwanych tak jak akcje użytkownika powodujące te przejścia.

Natomiast projekt formularza (jego wygląd i zakres informacyjnych) wykonywany jest w innym dowolnym narzędziu. Następnie jest on uzupełniany o szczegółowy opis zawartości informacyjnej oraz dostępnych akcji i umieszczany w Linked Document elementu Screen, tak aby pełny opis funkcjonalności systemu był w jednym repozytorium modeli.

## 9. Projektowanie

---

W ramach aktywności związanych z projektowaniem w projekcie IT mogą być (w zależności od potrzeb) tworzone następujące modele:

- Model komponentów,
- Model interakcji (diagramy sekwencji),
- Fizyczny model danych,
- Model wdrożenia (rozmieszczenia) bazujący na modelu platformy.

### 9.1. Model komponentów

---

System podzielony jest na komponenty funkcjonalne grupujące funkcjonalność, którą da się logicznie wyodrębnić i może być ona niezależnie zarządzana.

Model komponentów dzieli się na dwie części:

- Model specyfikacji komponentów (zwany również Logiczną perspektywą Modelu komponentów) – zawiera analityczną specyfikację interfejsów publicznych niezależnych od platformy (MDA PIM);
- Model realizacji komponentów (zwany również Fizyczną perspektywą Modelu komponentów) – zawiera realizację produktu z podziałem na komponenty z uwzględnieniem warstw technologicznych platformy, model specyficzny dla platformy (MDA PSM).

Celem modelu specyfikacji komponentów jest zobrazowanie współpracy komponentu funkcjonalnego (logicznego) z innymi komponentami poprzez pokazanie na diagramie komponentów (UML):

- Elementów komponent o standardowym stereotypie UML «specification»,
- Osadzonych na portach interfejsów (na poziomie logicznym):
  - Dostarczanych przez komponent,
  - Wymaganych przez komponent,
- Współpracy pomiędzy komponentami zamodelowanej w postaci relacji zależności (Dependency) łączącej interfejs wymagany przez jeden komponent z interfejsem dostarczającym przez inny komponent,
- Elementów interfejs (UML: Interface) wraz operacjami, które realizuje.

Komponenty na poziomie fizycznym (ze stereotypem «realization») są pochodną komponentów na poziomie logicznym («specification»), wynikającą m.in. z uwzględnienia warstw technologicznych platformy oraz technologii implementacji. Interfejsy (i ich operacje) na poziomie fizycznym pokazują rzeczywiste (fizyczne) nazwy operacji, typy danych itp.

Komponenty fizyczne mogą tworzyć hierarchię poprzez zagnieżdżanie.

Dodatkowo na poziomie fizycznym w modelu komponentów mogą pojawić się klasy implementacyjne.

Poniższa tabela specyfikuje podstawowe elementy i relacje wykorzystywane w modelu komponentów.

W tabeli zastosowano pojęcia: Interfejs logiczny oraz Interfejs fizyczny. Należy to rozumieć jako przynależność interfejsu do odpowiednio: logicznego (specification) i fizycznego (realization) poziomu opisu Komponentów.

Tabela 6 Powiązania elementów modelu komponentów

Element modelu						Element powiązany	
Komponent Logiczny Component «Specification»	Komponent Fizyczny Component «Realization»	Port Port	Interfejs logiczny Interface	Interfejs fizyczny Interface	Klasa implementacyjna Class		
zależność→	realizacja [1]→	x	x	x	x	Komponent Logiczny	Model komponentów
x	zależność→ posiadanie przynależność	x	x	x	przynależność	Komponent Fizyczny	
x	x	x	x	x	x	Port	
x	x	typ	generalizacja→	śladowanie [1] →	x	Interfejs logiczny	
x	realizacja→ użycie→	x	x	generalizacja→	x	Interfejs fizyczny	
x	śladowanie→ typ atrybutu asocjacja→ posiadanie	x	x	x	x	Klasa implementacyjna	Model wymagań
x	realizacja→	x	x	x	x	Wymaganie rozwiązania: niefunkcjonalne	
x	x	x	śladowanie [1]→	x	śladowanie→	Przypadek użycia	
x	x	x	x	x	śladowanie→	Formularz	Model interfejsu użytkownika



## 9.2. Model interakcji

Model interakcji stanowią diagramy sekwencji (UML: Sequence Diagrams) opisujące realizację poszczególnych przypadków użycia.

Uczestnikami interakcji są zazwyczaj:

- aktor (instancja aktora),
- obiekt boundary – reprezentujący element systemu zapewniający komunikację ze środowiskiem systemu, np. okno,
- obiekt control – reprezentujący element sterujący,
- obiekt entity – reprezentujący dane (informacje), które są przechowywane przez system.

W praktyce obiekty: boundary, control i entity mogą reprezentować: komponenty fizyczne, klasy implementacyjne, elementy fizycznego modelu danych, środowisko logiczne (np. węzeł bazy danych).

Poniższa tabela specyfikuje podstawowe elementy i relacje wykorzystywane w modelu interakcji.

Tabela 7 Powiązania elementów Modelu interakcji

Element modelu			Element powiązany	
Interakcja Interaction	Linia życia Lifeline	Komunikat Message		
x	przynależność	przynależność	Interakcja	Model interakcji
posiadanie	x	wysłanie odebranie	Linia życia	
posiadanie	x	x	Komunikat	
x	reprezentacja	x	Komponent Fizyczny	Model komponentów
x	x	sygnatura	Operacja Interfejsu fizycznego	
x	reprezentacja	x	Klasa implementacyjna	
x	reprezentacja	x	Klasa (fizyczna) Tabela, Widok	Fizyczny model danych
x	reprezentacja	x	Środowisko logiczne	Infrastruktura oprogramowania
x	reprezentacja	x	Aktor	Model przypadków użycia
śladowanie→	x	x	Aktywność	

## 9.3. Fizyczny model danych

Na fizyczny model danych składają się definicje:

- Standardowe typy danych wykorzystywane w fizycznym modelu danych,
- Schemat bazy danych
- Specyfikacja protokołów komunikacji,
- Fizyczny model usług.

Fizyczny model danych stanowi konkretną realizację logicznego modelu danych zdefiniowanego na poziomie logicznym. Stąd istotne jest śledzenie elementów fizycznego modelu danych na elementy modelu logicznego.

Podstawowym modelem jest model odpowiadający schematowi bazy danych.

Dla tworzenia tego modelu w EA wykorzystywany jest udostępniony przez EA pakiet narzędzi do modelowania danych – Data Modeling.

Dla każdej tablicy definiowane są:

- nazwa,
- opis,
- lista kolumn:
  - nazwa,
  - typ danych,
  - dopuszczalna długość, dla typów które tego wymagają,
  - wymagalność,
  - wartość domyślna,
- klucz główny tablicy,
- lista indeksów:
  - nazwa,
  - lista kolumn z określonym porządkiem sortowania,
- lista kluczy obcych:
  - nazwa,
  - definicja klucza,
- lista ograniczeń (check-ów):
  - nazwa,
  - treść ograniczenia.

Do zamodelowania innych niż schemat bazy danych elementów fizycznego modelu danych rekomendowanym środkiem wyrazu jest klasa rozszerzona odpowiednim stereotypem.

Dopuszczalne jest w tym celu także użycie innych narzędzi niż Enterprise Architect – np. plików ze schematami xml (xsd), WSDL itp.

Poniższa tabela definiuje podstawowe elementy i powiązania stosowane w fizycznym modelu danych.

Tabela 8 Powiązania elementów Fizycznego Modelu Danych

Element modelu					Element powiązany	
Typ danych (fizyczny) DataType	Klasa <sup>1</sup> Class	Tabela Class «Table»	Synonim Alias Class «Synonym» «Alias»	Widok Class «View»		
generalizacja →	typ atrybutu	typ kolumny	zależność→	typ kolumny	Typ danych	Fizyczny model danych
x	generalizacja → typ atrybutu asocjacja→	x	zależność→	x	Klasa	
x	x	generalizacja → klucz obcy (jako asocjacja→)	zależność→	wnioskowanie → klucz obcy (jako asocjacja→)	Tabela	
x	x	x	x	x	Synonim Alias	

<sup>1</sup> Klasa została tu przyjęta jako domyślny środek wyrazu do modelowania fizycznych elementów strukturalnych. W zależności od potrzeb danego Modelu, klasy mogą być rozszerzane właściwymi stereotypami.

x	x	x	zależność→	x	<b>Widok</b>	<b>Model dziedziny systemu</b>
śladowanie→	x	x	x	x	<b>Typ danych (logiczny)</b>	
x	śladowanie→	śladowanie→	x	śladowanie→	<b>Klasa</b>	

## 9.4. Model wdrożenia (rozmieszczenia)

Celem modelu wdrożenia (rozmieszczenia) jest zaprezentowanie rozmieszczenia artefaktów wdrożeniowych oprogramowania dedykowanego w infrastrukturze sprzętowej i infrastrukturze oprogramowania. Tak więc niezbędnym elementem dla zbudowania modelu rozmieszczenia jest model platformy obejmujący infrastrukturę sprzętową i oprogramowania.

W modelu rozmieszczenia wykorzystywane są diagramy rozmieszczenia (Deployment Diagrams) zdefiniowane w UML i następujące główne elementy tych diagramów:

- Urządzenie (Device),
- Środowisko wykonania (Execution Environment),
- Artefakt wdrożeniowy (Artifact)
- Relacja rozmieszczenia (Deployment),
- Relacja manifestacji (Manifestation)

Diagram rozmieszczenia może być wykonany na poziomie referencyjnym – klasyfikatorów lub też na poziomie konkretnych wystąpień (instancji klasyfikatorów).

Poniższa tabela definiuje podstawowe elementy i powiązania stosowane w modelu rozmieszczenia.

Tabela 9 Powiązania elementów modelu rozmieszczenia

Element modelu			Element powiązany	
Urządzenie Device	Środowisko Execution Environment	Artefakt Artifact		
posiadanie przynależność komunikacja→ generalizacja→	przynależność	x	Urządzenie	Modele rozmieszczenia
posiadanie	posiadanie przynależność komunikacja→ generalizacja→	rozmieszczenie→	Środowisko	
x	x	posiadanie przynależność zależność→	Artefakt	
x	x	manifestacja→[1]	Komponent Fizyczny	Model komponentów
realizacja→	realizacja→	x	Wymaganie rozwiązania: niefunkcjonalne	Model wymagań

# 10. Dokumentacja projektowa

W dokumentacji projektowej mogą być wykorzystane elementy modeli poprzez umieszczenie w dokumentacji wybranych rodzajów diagramów.

Ponadto część dokumentacji projektowej może być tworzona automatycznie jako odpowiednio zdefiniowany raport z modeli.

## 10.1. Mapowanie modeli na dokumenty projektowe

Zakłada się, że dla może być tworzona dokumentacja na dwóch poziomach:

- Analitycznym,
- Technicznym

Poniższa tabela prezentuje wykorzystanie modeli opisanych w poprzednich rozdziałach w dokumentacji dla obu poziomów.

Tabela 10 Mapowanie modeli na dokumenty projektowe

Poziom dokumentacji	Wykorzystane modele
Dokumentacja analityczna	Model procesów biznesowych
	Model wymagań
	Model przypadków użycia
	Model zachowania przypadków użycia
	Model dziedziny systemu
	Wzory dokumentów
	Model interfejsu użytkownika
Dokumentacja techniczna	Model komponentów – poziom publicznych interfejsów logicznych
	Model komponentów
	Model interakcji
	Fizyczny model danych
	Model wdrożenia