

Planowanie procesu wdrożenia produktu informatycznego za pomoc UML

Krzysztof Torończak, Michał Bunda



Imię i nazwisko	Krzysztof Torończak, Michał Bunda
Nazwa uczelni	Uniwersytet Gdański
Kierunek	Informatyka Praktyczna
Specjalność	-
Zajęcia	Analiza i projektowanie systemów informatycznych
Prowadzący	Dr inż. Stanisław Witkowski
Temat	Planowanie procesu wdrożenia produktu informatycznego za pomoc UML
Data zajęć	08.11.2024
Numer sprawozdania	4
Data oddania	14.11.2024
Ocena	

1. Elementy języka UML	4
Diagramy strukturalne:	5
Diagramy behawioralne:	9
2. Modelowanie wymagań za pomocą przypadków użycia	14
Podstawowe elementy diagramu przypadków użycia	14
Relacje w Przypadkach Użycia	17
3. Diagramy czynności i sekwencji	18
Diagram czynności	18
Zastosowanie diagramu czynności	18
Podstawowe elementy diagramu czynności	18
Diagram sekwencji	21
Zastosowania diagramu sekwencji	21
Podstawowe elementy diagramu sekwencji	21
4. Modelowanie klas i powiązań pomiędzy nimi	24
Diagram klas	24
Zastosowanie diagramu klas	24
Podstawowe elementy diagramu klas	24
5. Diagramy komponentów	26
Diagram komponentów	27
Zastosowanie diagramu komponentów	27
Podstawowe elementy diagramu komponentów	27
6. Podział modelu na pakiety	30
Diagram pakietów	30
Zastosowanie diagramu pakietów	30
Podstawowe elementy diagramu pakietów	31
7. Modelowanie wdrożenia systemu	34
Diagram wdrożenia	34
Zastosowanie diagramu wdrożenia	34
Podstawowe elementy diagramu wdrożenia	34
8. Visual Paradigm online – opis działania oraz alternatywy	36
Visual Paradigm online	36
Alternatywa do Visual Paradigm Online – drawio	45

9. Wnioski.....	45
Potencjalne przypadki użycia UML	45
Potencjalne problemy z używaniem UML:	46
Podsumowanie.....	46

1. Elementy języka UML

Język UML (Unified Modeling Language, czyli “Ujednolicony język modelowania”), jak nazwa wskazuje jest ustandaryzowanym, udokumentowanym i dobrze opisanym standardem/sposobem/językiem modelowania interakcji i zależności zachodzących w systemie oraz samych jego logicznych części z możliwością uwzględnienia ich cech i atrybutów. Język UML ma na celu zapewnienie ustandaryzowanych sposobów modelowania pokrywających wszystkie, a przynajmniej większość potrzeb do przygotowania wysokopoziomowych planów różnorodnych systemów.

Język UML oferuje w tym celu dwa główne rodzaje diagramów dzielące się ze względu na 1. budowę/architekturę systemu

2. sposób działania systemu i interakcji z nim

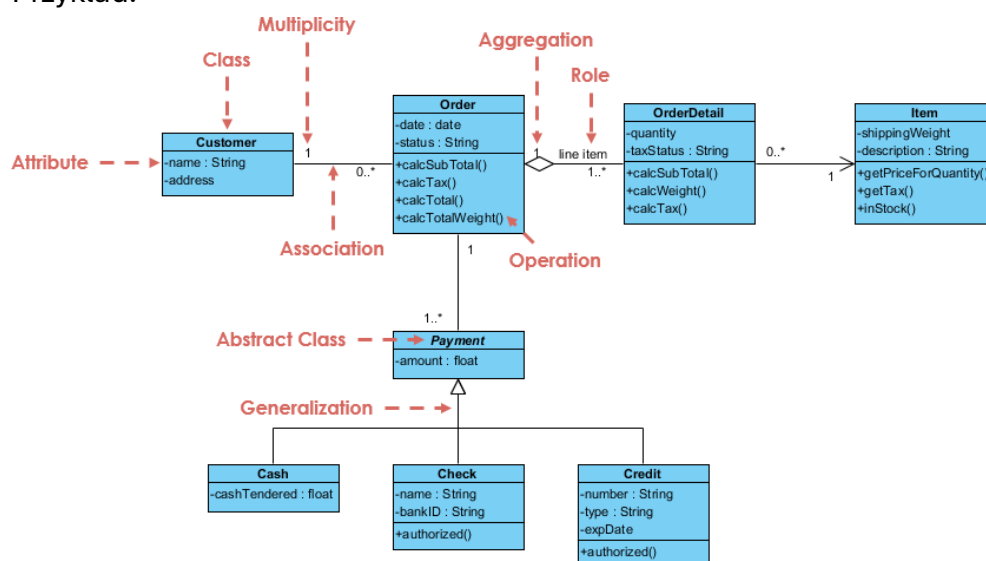
Oba główne rodzaje diagramów dzielą się natomiast znowu na 7 podrodzajów, które mają za zadanie służyć bardziej optymalnemu modelowaniu w dobrze skonkretyzowanych celach.

Podziały diagramów, o których mowa wyglądają następująco:

Diagramy strukturalne:

1. Diagram Klas (Class Diagram) - przedstawia strukturę systemu przy użyciu klas, ich atrybutów, operacji oraz relacji między nimi

Przykład:

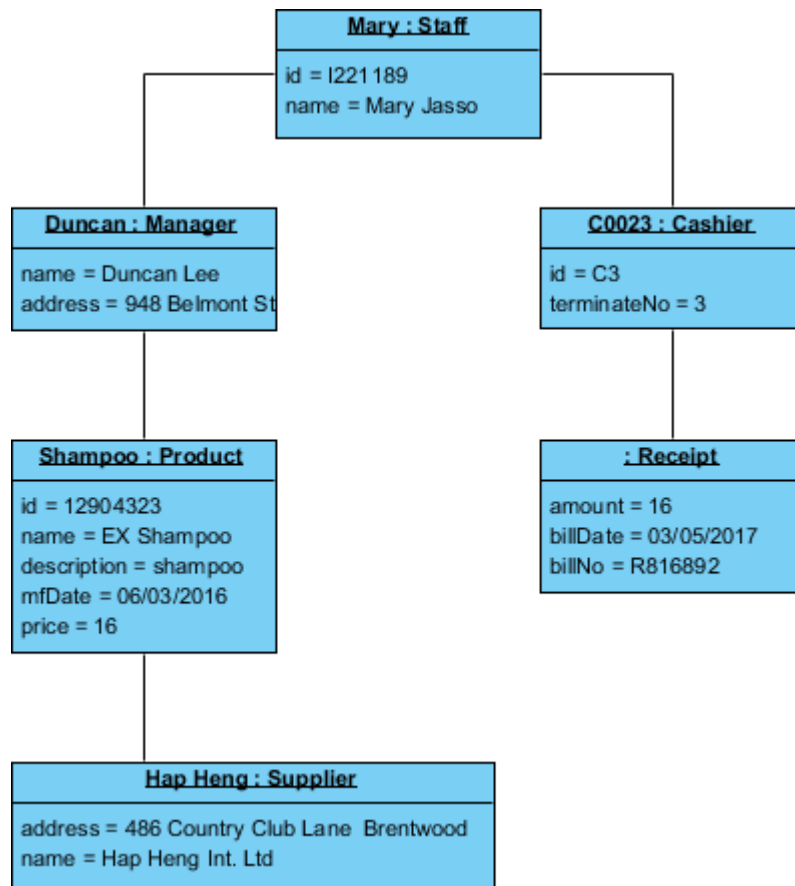


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

Data dostępu: 09.11.2024

2. Diagram Obiektów (Object Diagram) - pokazuje konkretne instancje klas i relacje między nimi w określonym momencie działania systemu, w pewnym sensie stanowiąc migawkę (snapshot) stanu systemu w danym momencie

Przykład:

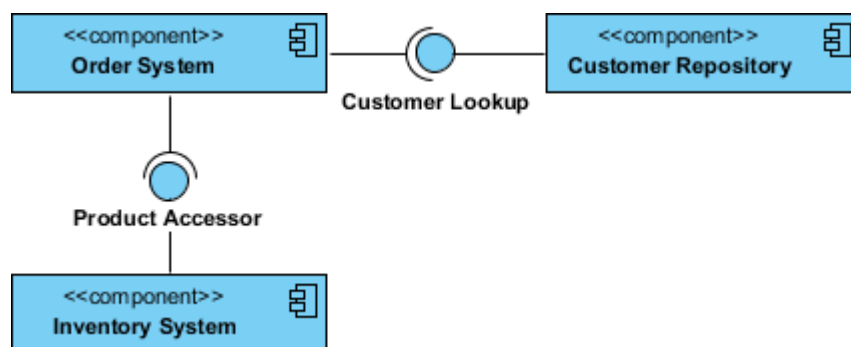


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-object-diagram/>

Data dostępu: 09.11.2024

- Diagram Komponentów (Component Diagram) - prezentuje fizyczną strukturę implementowanego systemu poprzez pokazanie jego komponentów oraz zależności między nimi. Jest szczególnie przydatny przy projektowaniu modularnych systemów

Przykład:



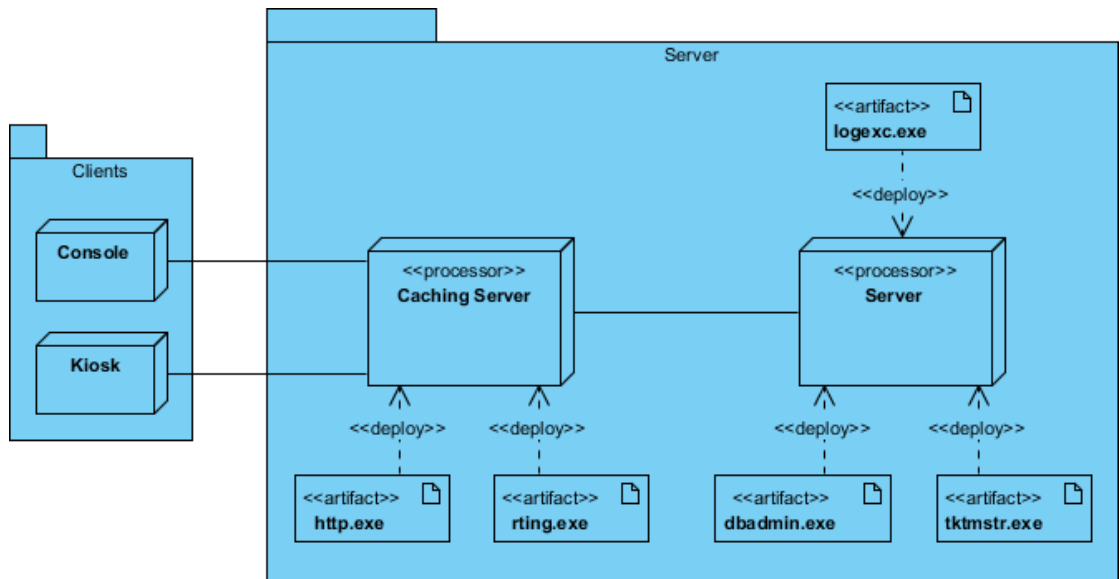
Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling->

[language/what-is-component-diagram/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/)

Data dostępu: 09.11.2024

4. Diagram Wdrożenia (Deployment Diagram) - przedstawia fizyczne rozmieszczenie artefaktów systemu na platformach sprzętowych, pokazując jak komponenty programowe są dystrybuowane w środowisku produkcyjnym

Przykład:

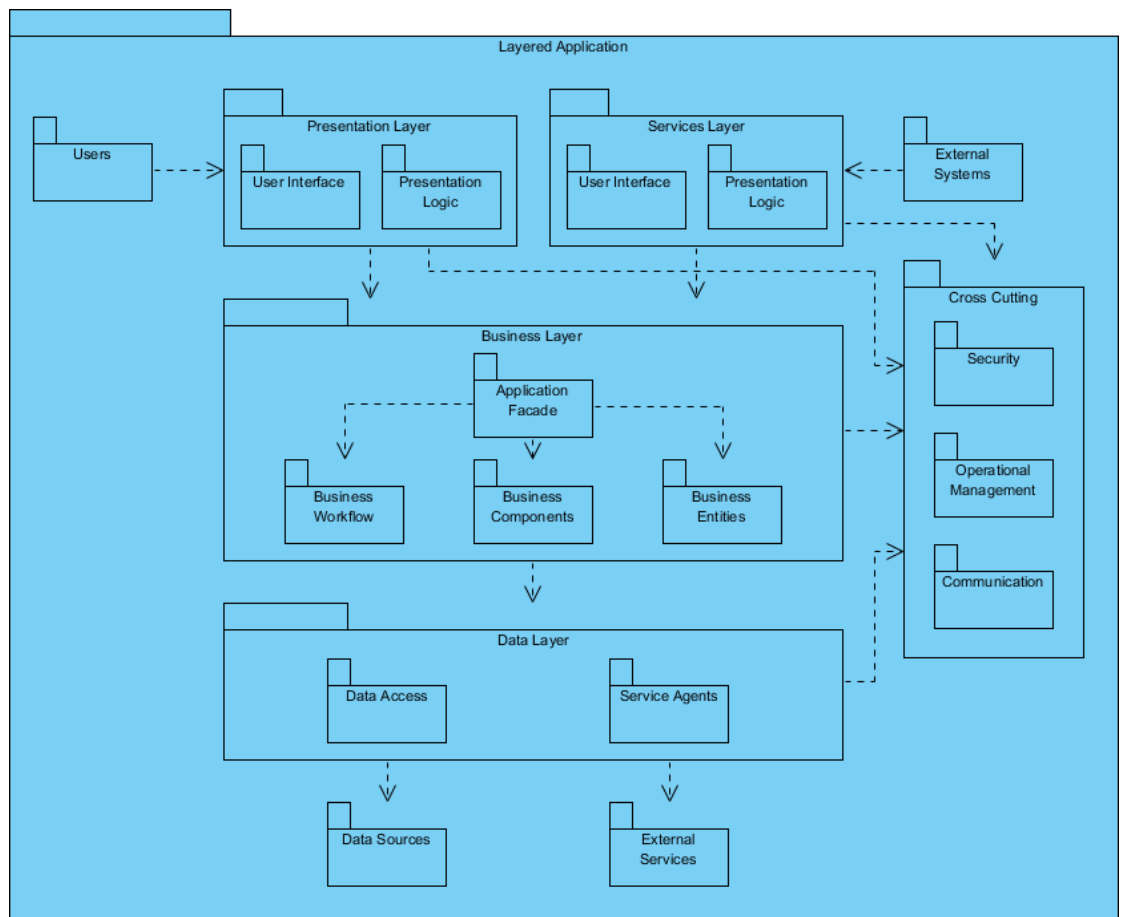


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/>

Data dostępu: 09.11.2024

5. Diagram Pakietów (Package Diagram) - pokazuje organizację elementów modelu w logiczne grupy (pakiety) oraz zależności między tymi grupami, co jest szczególnie przydatne przy zarządzaniu dużymi projektami

Przykład:

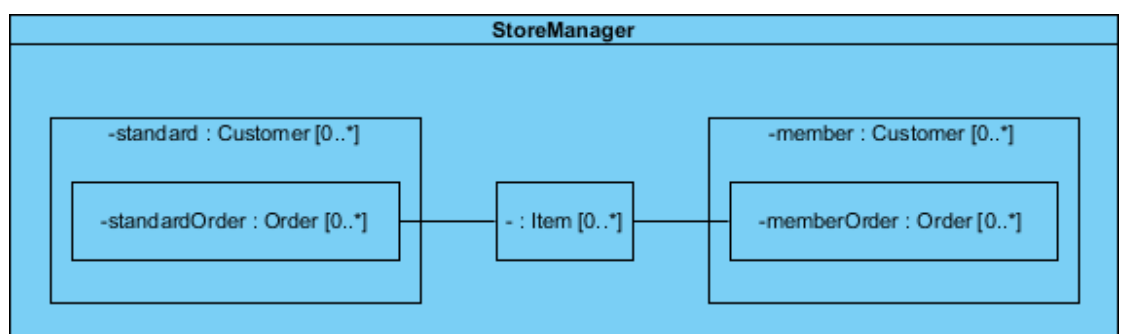


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/>

Data dostępu: 09.11.2024

6. Diagram Struktur Złożonych (Composite Structure Diagram) - umożliwia szczegółowe modelowanie wewnętrznej struktury klas i komponentów, pokazując jak ich części współpracują ze sobą w celu realizacji określonej funkcjonalności. Jest podobny do diagramu klas, ale nie przedstawia całej struktury klasy tylko jej część z większą ilością szczegółów

Przykład:



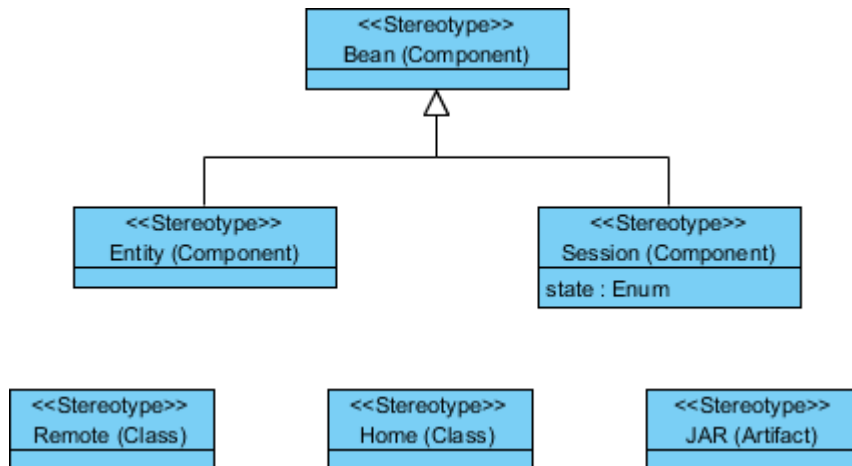
Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling->

[language/what-is-composite-structure-diagram/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-composite-structure-diagram/)

Data dostępu: 09.11.2024

7. Diagram Profilu (Profile Diagram) - pozwala na dostosowanie i rozszerzenie standardowej notacji UML poprzez definiowanie stereotypów, wartości oznaczonych i ograniczeń, umożliwiając tym samym lepsze dopasowanie języka do specyficznych potrzeb dziedziny

Przykład:



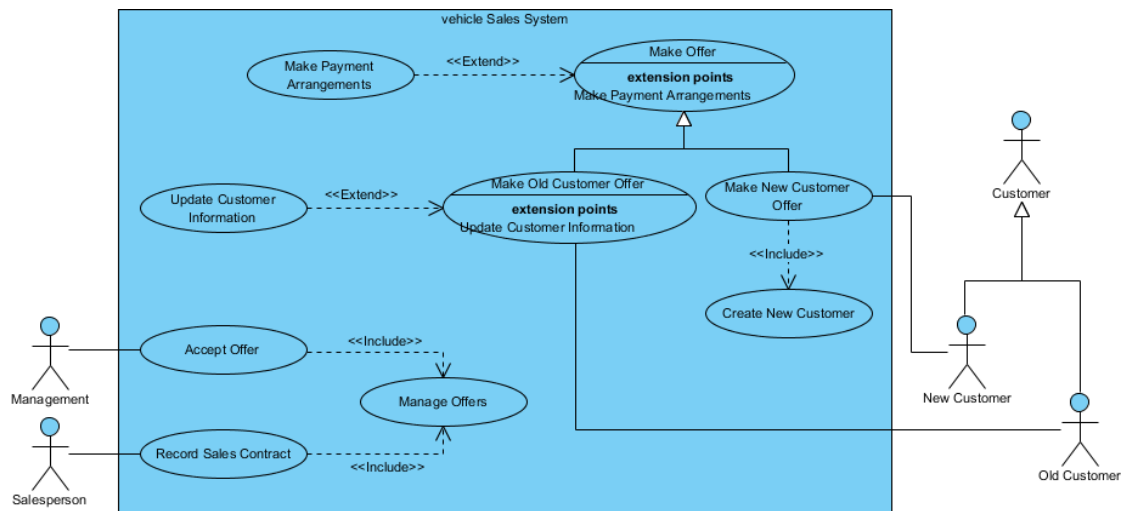
Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-profile-diagram/>

Data dostępu: 09.11.2024

Diagramy behawioralne:

1. Diagram Przypadków Użycia (Use Case Diagram) - diagram prezentujący funkcjonalności systemu widziane z perspektywy użytkownika końcowego. Może on w przejrzysty sposób przedstawiać interakcje zachodzące pomiędzy aktorami (użytkownikami lub zewnętrznymi systemami) a projektowanym systemem informatycznym

Przykład:

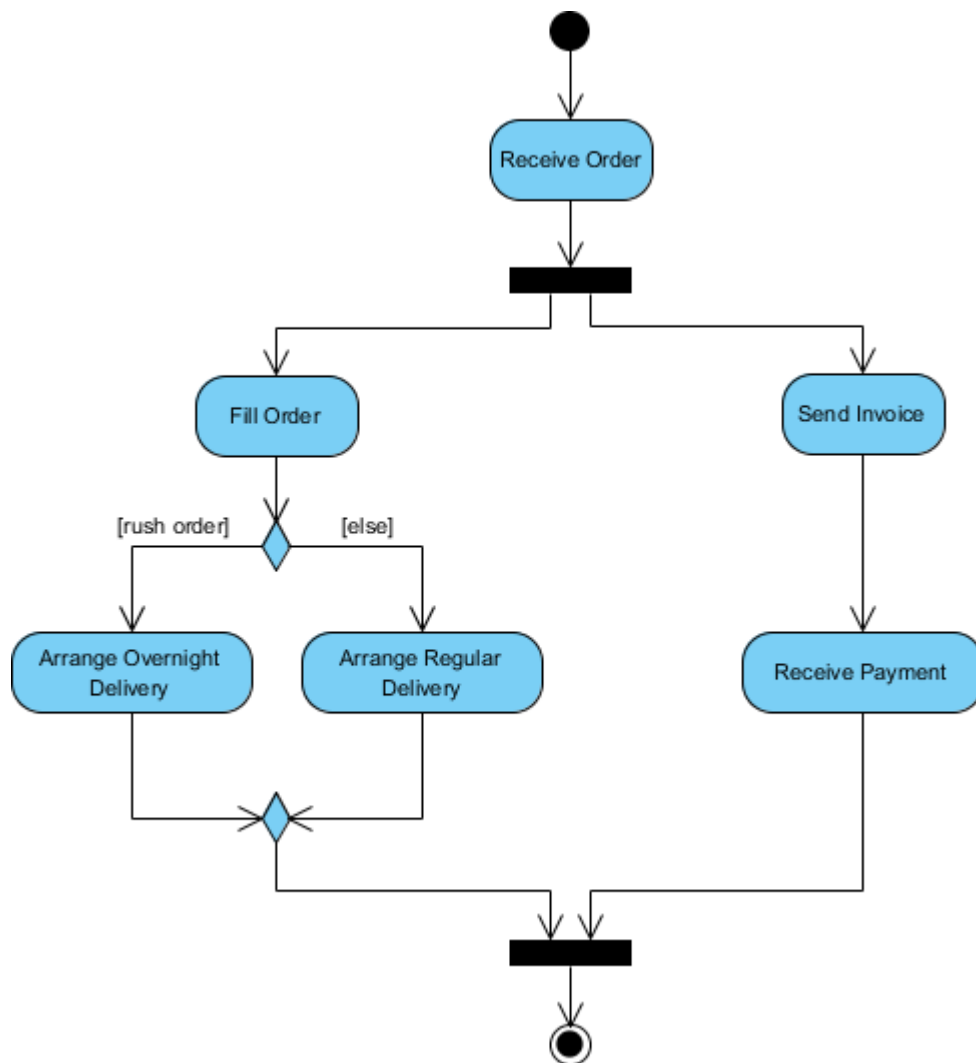


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

Data dostępu: 09.11.2024

2. Diagram Aktywności/Czynności (Activity Diagram) - jest to graficzna reprezentacja przepływu działań w systemie, która pokazuje kolejność wykonywania czynności oraz warunki podejmowania decyzji. Jest on szczególnie przydatny przy modelowaniu procesów biznesowych lub przedstawianiu schematu działania skomplikowanych algorytmów

Przykład:

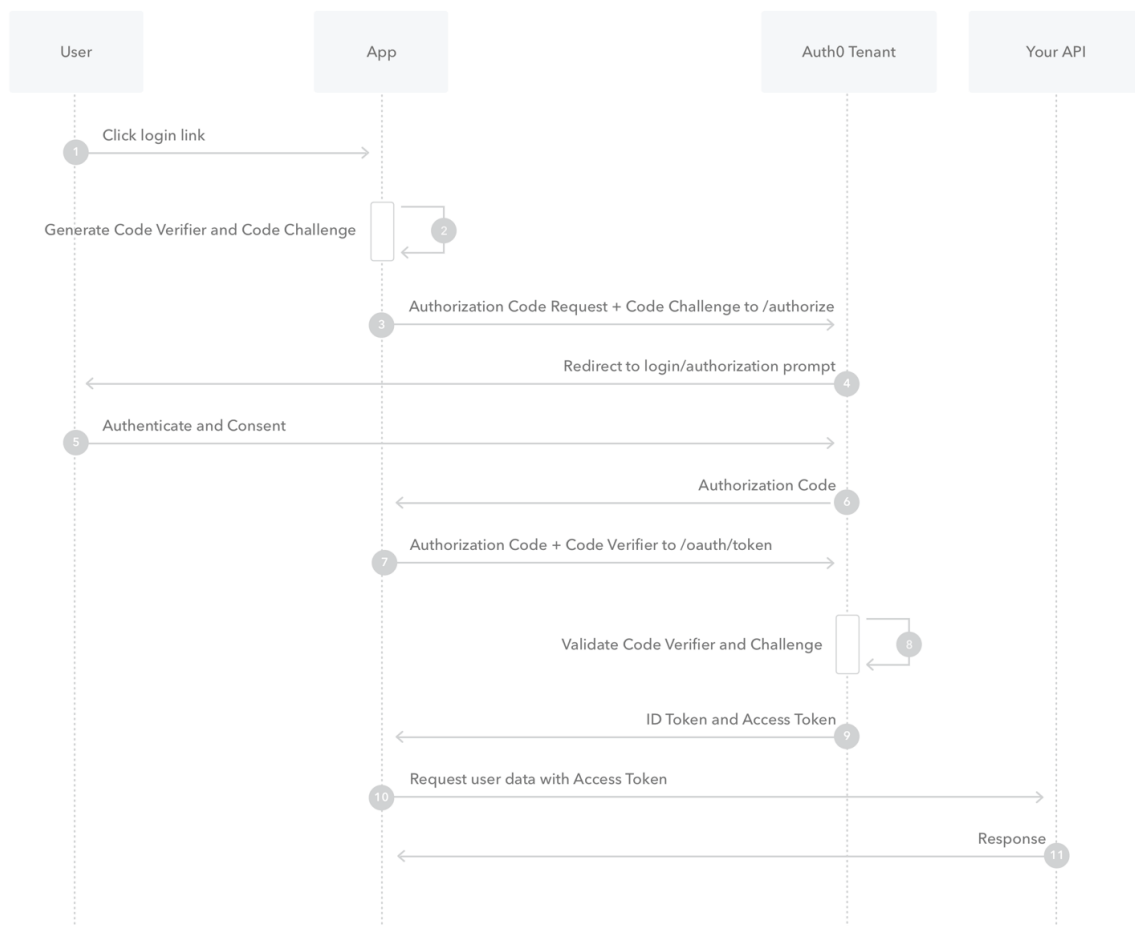


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

Data dostępu: 09.11.2024

3. Diagram Sekwencji (Sequence Diagram) - przedstawia chronologiczną sekwencję interakcji zachodzących między obiektami w systemie, ukazując wymianę komunikatów między nimi w czasie. Jest szczególnie użyteczny przy analizie scenariuszy użycia oraz projektowaniu i analizie protokołów komunikacji np. OAuth 2.0 authorization code grant flow with PKCE. Trudno jest w bardziej przejrzysty sposób pokazać działanie podanego w przykładzie schematu

Przykład (schemat działania OAuth 2.0 authorization code grant flow with PKCE:

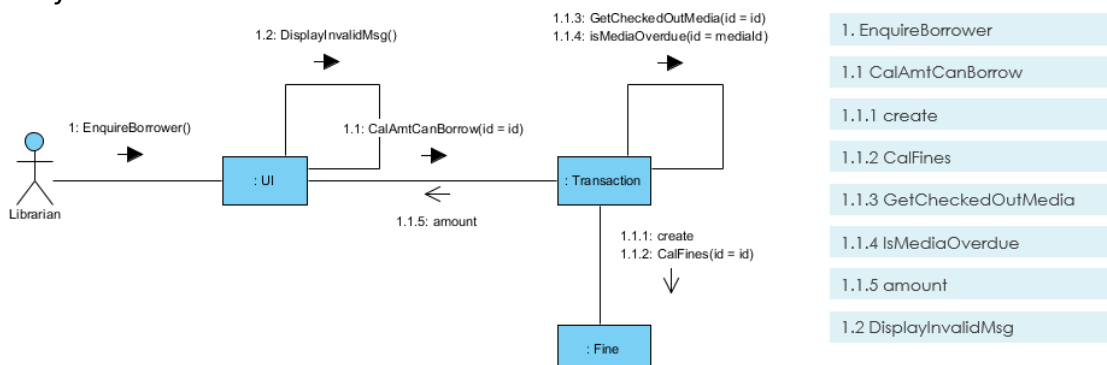


Źródło: <https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow-with-pkce>

Data dostępu: 09.11.2024

4. Diagram Komunikacji (Communication Diagram) - stanowi alternatywne spojrzenie na interakcje między obiektami, kładąc nacisk na strukturalne powiązania między komunikującymi się elementami systemu zamiast na aspekt czasowy tych komunikacji

Przykład:

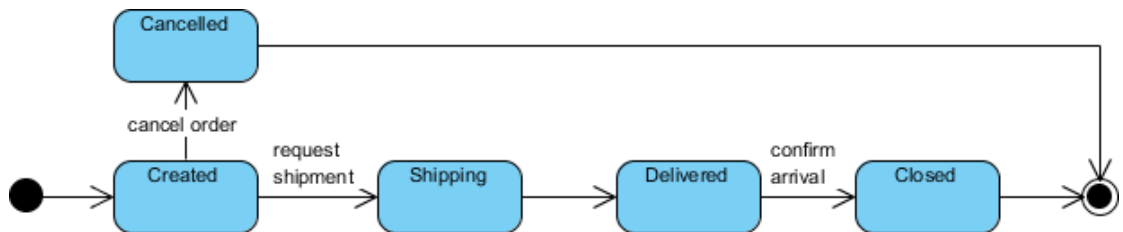


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-communication-diagram/>

Data dostępu: 09.11.2024

- Diagram Stanów (State Machine Diagram) - opisuje różne stany, w których może znajdować się obiekt w czasie swojego istnienia, oraz zdarzenia powodujące przejścia między tymi stanami. Jest szczególnie przydatny przy modelowaniu obiektów, których zachowanie silnie zależy od ich historii

Przykład:

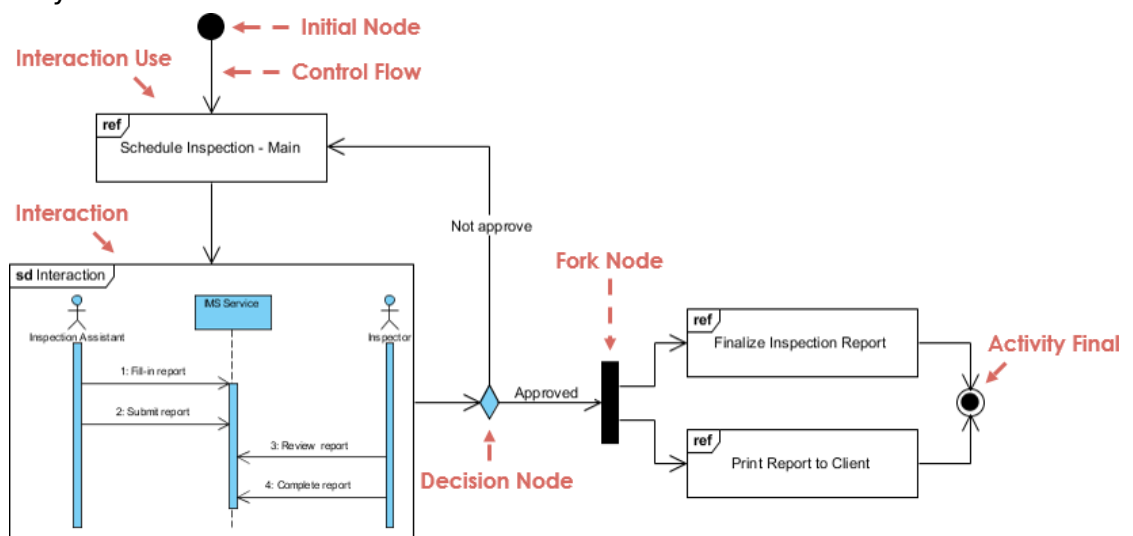


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/>

Data dostępu: 09.11.2024

- Diagram Przeglądu Interakcji (Interaction Overview Diagram) - łączy elementy diagramu aktywności z diagramami interakcji, umożliwiając modelowanie złożonych scenariuszy, w których przepływ sterowania przechodzi przez różne interakcje w systemie

Przykład:

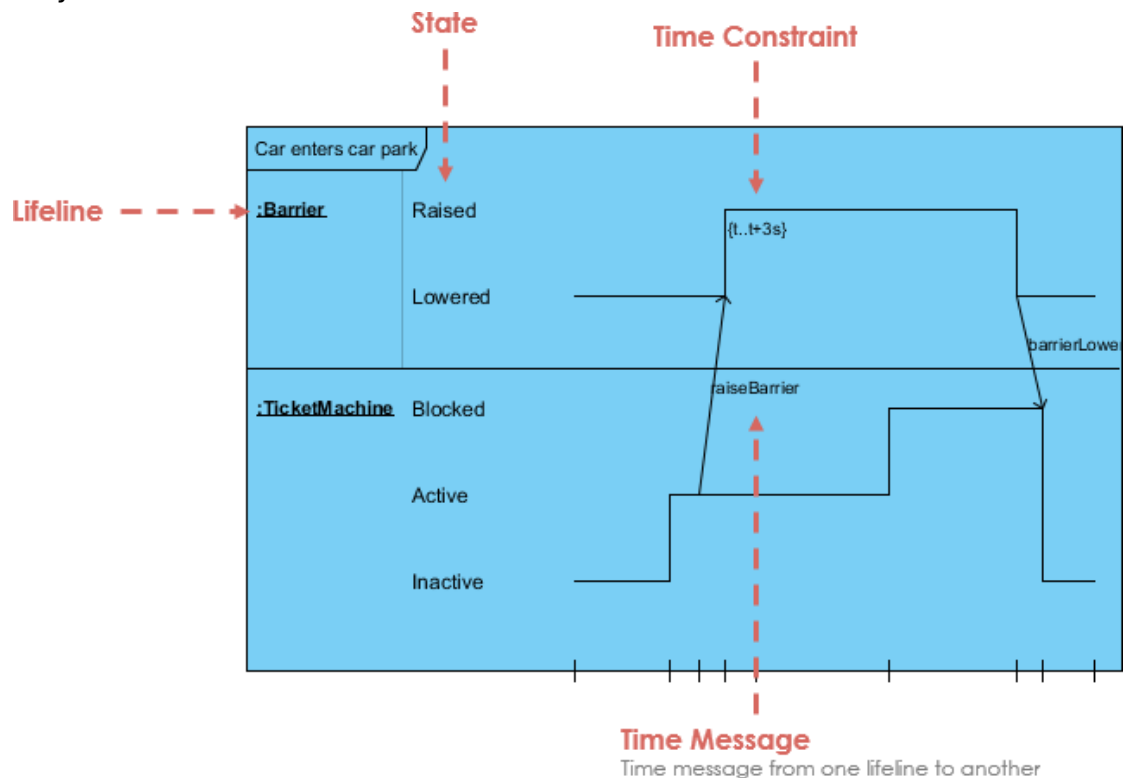


Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-interaction-overview-diagram/>

Data dostępu: 09.11.2024

7. Diagram Czasowy (Timing Diagram) - koncentruje się na zmianach stanu obiektów w czasie, ze szczególnym uwzględnieniem ograniczeń czasowych i synchronizacji. Jest użyteczny w systemach czasu rzeczywistego i przy analizie wydajności

Przykład:



Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-timing-diagram/>

Data dostępu: 09.11.2024

2. Modelowanie wymagań za pomocą przypadków użycia

Podstawowe elementy diagramu przypadków użycia

1. Aktorzy

Są to użytkownicy lub zewnętrzne systemy wchodzące w interakcję z modelowanym systemem. Mogą być ludźmi (np. klient, administrator) lub innymi systemami (np. system płatności). Jeden aktor może uczestniczyć w wielu przypadkach użycia. Aktorzy są reprezentowani przez symbole ludzi lub zdefiniowane ikony.



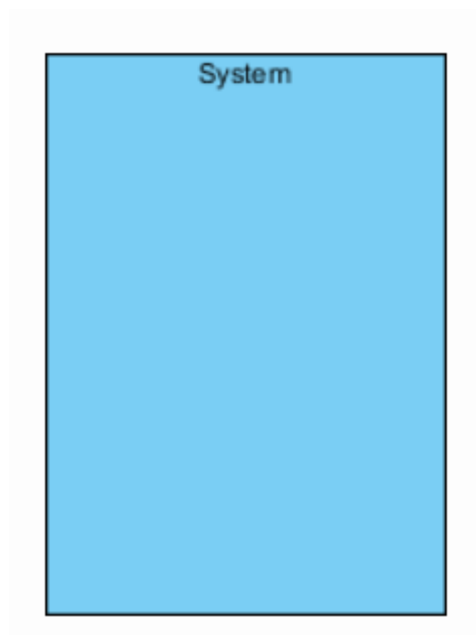
2. Przypadki Użycia

Reprezentują konkretne funkcjonalności systemu i są przedstawiane jako elipsy z nazwą funkcjonalności. Opisują co system robi, bez wchodzenia w szczegóły jak to się dzieje. Powinny zawierać dostarczaną danemu aktorowi funkcjonalność/wartość biznesową.



3. Granica Systemu (System Boundary)

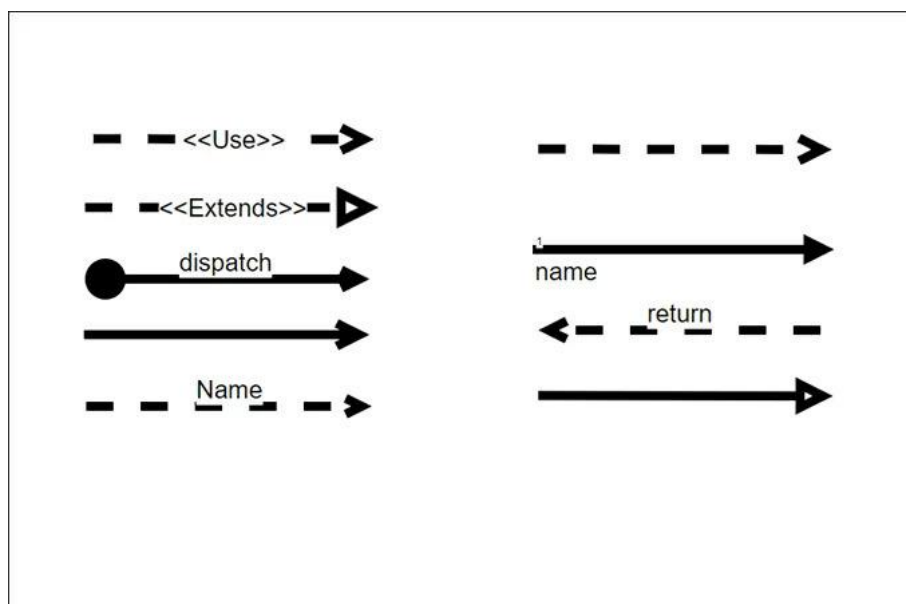
Reprezentowana jest jako prostokąt otaczający wszystkie przypadki użycia i ma za zadanie wyraźnie oddzielić to, co wchodzi w skład systemu od tego, co jest elementem/czynnikiem zewnętrznym. Jest zwykle oznaczona nazwą modelowanego systemu na górze prostokąta. Wszystkie przypadki użycia powinny znajdować się wewnątrz granicy systemu, natomiast wszyscy aktorzy powinni znajdować się na zewnątrz.



4. Relacje i związki

Reprezentują powiązania między elementami na diagramie. Mogą łączyć aktorów z przypadkami użycia lub przypadki użycia między sobą. Służą to ukazaniu różnego rodzaju zależności i związków w systemie. Relacje przedstawiane są jako linie, które mogą być rysowane jako linie ciągłe lub przerywane z różnego rodzaju grotami, w zależności od typu relacji.

Symbole relacji:

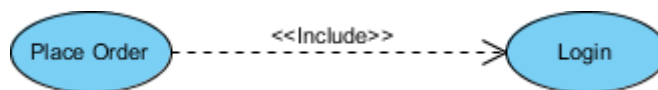


Symbol związku:

Relacje w Przypadkach Użycia

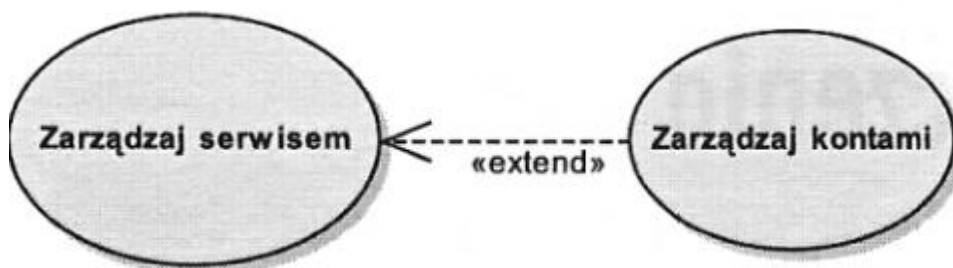
1. Include (zawieranie)

- Używana gdy jeden przypadek użycia zawsze wykorzystuje funkcjonalność innego
- Reprezentowana strzałką przerywaną z etykietą <<include>>
- Przykład: "Zaloguj się" może być zawarte w "Złóż zamówienie"



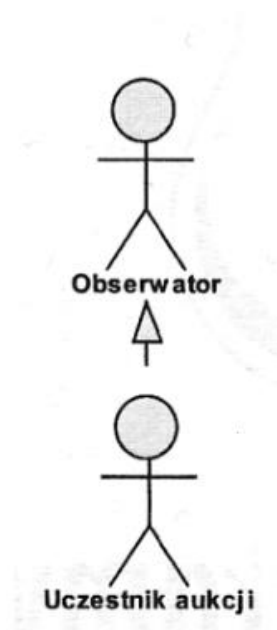
2. Extend (rozszerzenie)

- Używana gdy jeden przypadek użycia opcjonalnie rozszerza inny
- Reprezentowana strzałką przerywaną z etykietą <<extend>>
- Przykład: "Zarządzaj kontami" może rozszerzać "Zarządzaj serwisem"



3. Generalizacja

- Pokazuje hierarchię między przypadkami użycia lub aktorami
- Reprezentowana strzałką z pustym grotem
- Przykład: "Uczestnik akcji" może być specjalizacją aktora "Obserwator"



3. Diagramy czynności i sekwencji

Diagram czynności

Zastosowanie diagramu czynności

- Modelowanie procesów biznesowych
- Przedstawianie algorytmów
- Wizualizacja przepływu sterowania
- Dokumentowanie złożonych procedur
- Modelowanie przepływu pracy (workflow)

Podstawowe elementy diagramu czynności

1. Stan początkowy i końcowy
 - Stan początkowy: czarny wypełniony okrąg



- Stan końcowy: czarny okrąg z obwódką



- Każdy diagram musi mieć stan początkowy
- Może mieć wiele stanów końcowych

2. Czynność (Activity)

- Reprezentowana przez prostokąt z zaokrąglonymi rogami



- Opisuje pojedyncze działanie w procesie
- Zawiera krótki opis wykonywanej akcji
- Może być atomiczna lub złożona

3. Przepływ sterowania (Control Flow)

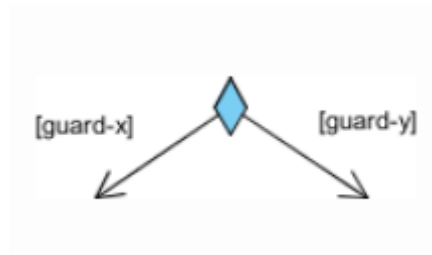
- Strzałki pokazujące kolejność wykonywania czynności



- Wskazują kierunek przejścia między czynnościami
- Mogą zawierać warunki przejścia
- Reprezentują sekwencję działań

4. Decyzja (Decision Node)

- Reprezentowana przez romb
- Pozwala na warunkowe rozgałęzienie przepływu
- Zawiera warunek decyzyjny
- Ma minimum dwa wyjścia



5. Scalanie (Merge Node)

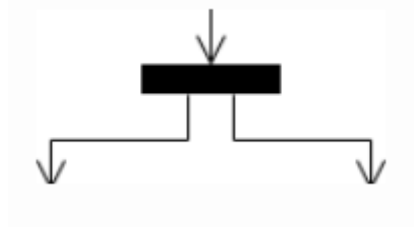
- Reprezentowana przez romb



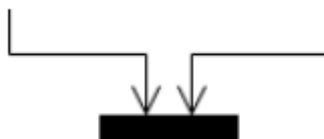
- Ponownie łączy różne ścieżki decyzyjne, które zostały utworzone za pomocą węzła decyzyjnego.

6. Rozdzielenie i złączenie (Fork and Join)

- Reprezentowane przez czarne paski
- Rozdzielenie (fork): rozpoczęcie działań równoległych



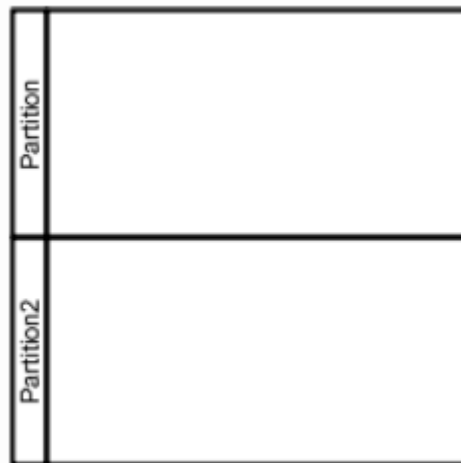
- Złączenie (join): synchronizacja działań równoległych



- Umożliwia modelowanie współbieżności

7. Tory (Swimlanes)

- Pionowe lub poziome sekcje diagramu



- Przypisują czynności do odpowiedzialnych za nie jednostek
- Pomagają w organizacji i czytelności diagramu
- Pokazują podział odpowiedzialności

Diagram sekwencji

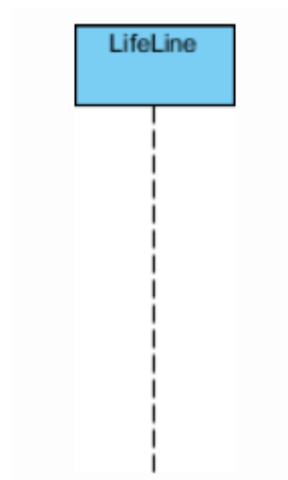
Zastosowania diagramu sekwencji

1. Projektowanie
 - Modelowanie interakcji między komponentami
 - Projektowanie interfejsów
 - Analiza przepływu danych
2. Dokumentacja
 - Dokumentowanie scenariuszy użycia
 - Specyfikacja protokołów komunikacji
 - Opis zachowania systemu
3. Analiza
 - Wykrywanie wąskich gardeł
 - Optymalizacja komunikacji
 - Weryfikacja logiki biznesowej

Podstawowe elementy diagramu sekwencji

1. Linia życia (Lifeline)

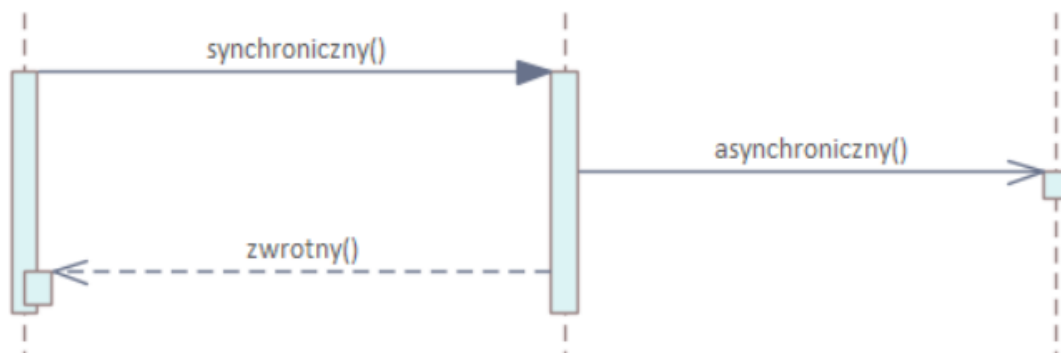
- Reprezentowana przez pionową przerywaną linię



- Przedstawia czas życia obiektu w systemie
- Na górze ma prostokąt z nazwą obiektu
- Pokazuje, jak długo obiekt istnieje w systemie

2. Komunikaty (Messages)

- Reprezentowane przez strzałki między liniami życia



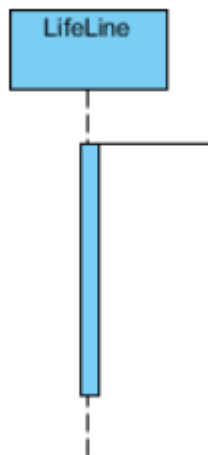
Źródło: <https://wolski.pro/diagramy-uml/diagram-sekwencji/>

Data dostępu: 09.11.2024

- Pokazują interakcje między obiektami
- Mogą być:
 - Synchroniczne (pełna strzałka)
 - Asynchroniczne (strzałka z grotem)
 - Zwrotne (przerywana strzałka)

3. Aktywacje (Activation boxes)

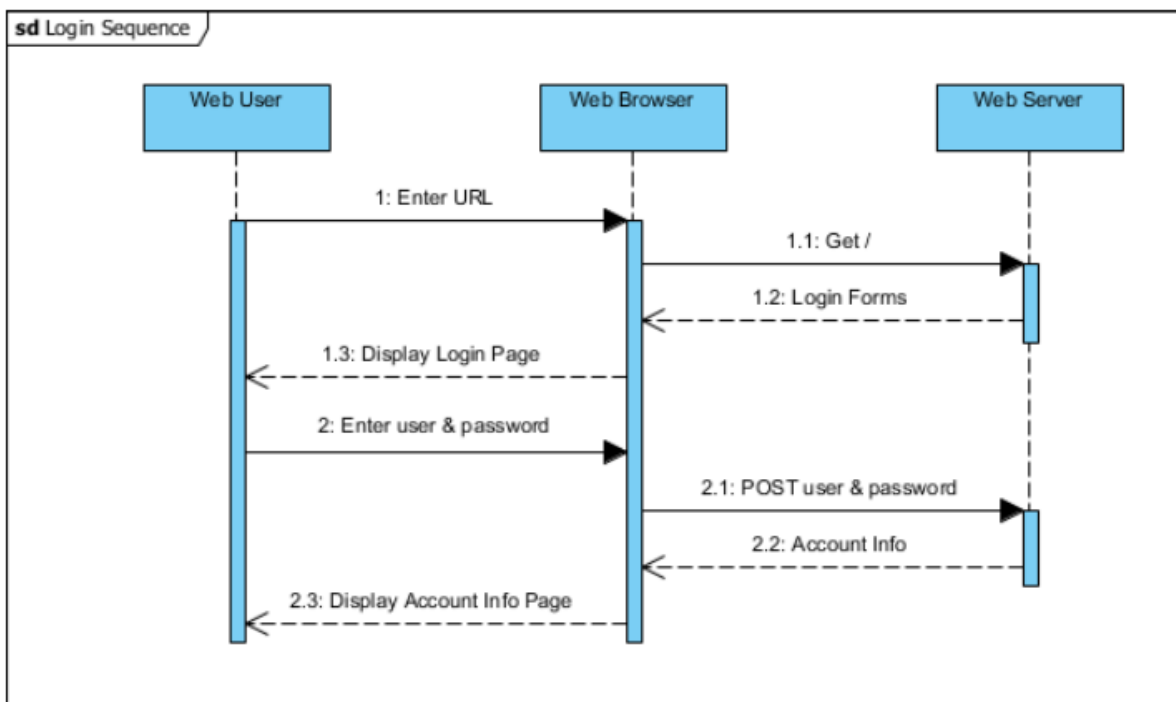
- Prostokąty na linii życia



- Pokazują, kiedy obiekt jest aktywny
- Reprezentują czas wykonywania operacji
- Mogą być zagnieżdżone

4. Ramki (Frames)

- Prostokąty otaczające fragmenty diagramu



Źródło: <https://www.visual-paradigm.com/guide/sysml/modeling-scenarios-with-sequence-diagram/>

Data dostępu: 09.11.2024

- Określają rodzaj interakcji (np. alt, loop, opt)

- Mogą zawierać warunki
- Służą do modelowania złożonych scenariuszy

4. Modelowanie klas i powiązań pomiędzy nimi

Do modelowania klas i powiązań między nimi w języku UML służy diagram klas.

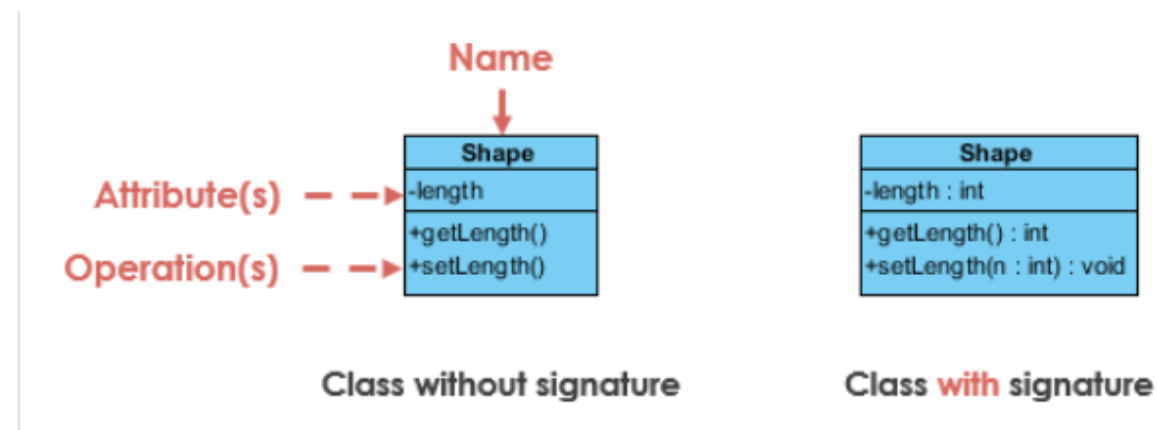
Diagram klas

Zastosowanie diagramu klas

- Modelowanie struktury systemu
- Przedstawianie relacji między obiektami
- Reprezentacja struktury danych oraz logiki biznesowej
- Organizacja klas, ich atrybutów oraz metod
- Dokumentowanie ról i związków między elementami systemu

Podstawowe elementy diagramu klas

1. Klasa
 - Reprezentowana przez prostokąt z trzema sekcjami



Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

Data dostępu 09.11.2024

- Sekcje klasy:
 - Nazwa klasy (na górze)
 - Atrybuty (w środku)
 - Operacje/metody (na dole)

- Klasa opisuje atrybuty i operacje obiektów, które będą jej instancjami

1. Atrybuty

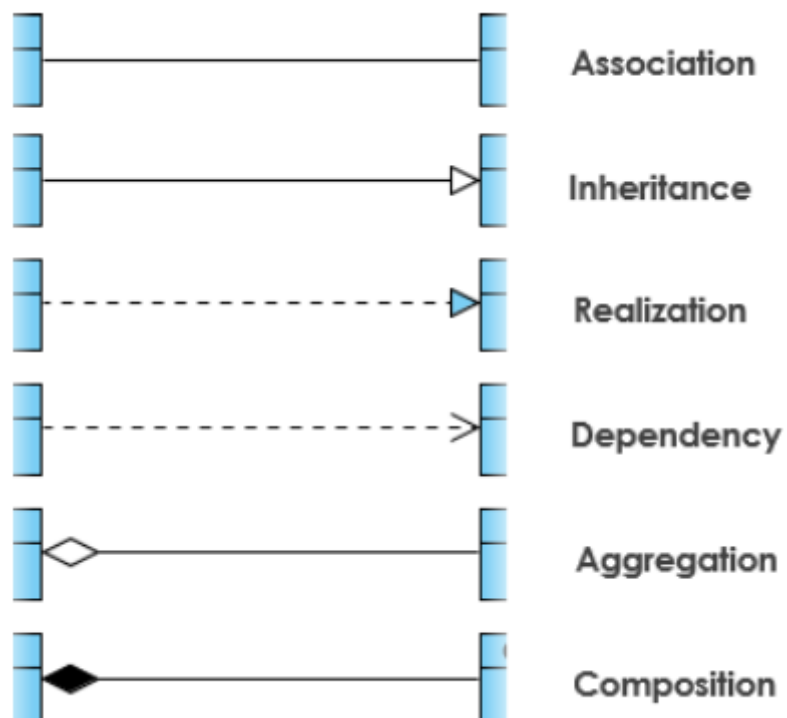
- Wymieniane w środkowej sekcji klasy
- Przechowują dane związane z instancjami klasy
- Atrybuty mogą mieć typ danych i widoczność (np. publiczna, prywatna)
- Notacja widoczności:
 - + publiczna (public)
 - - prywatna (private)
 - # chroniona (protected)

2. Operacje/metody

- Wymieniane w dolnej sekcji klasy
- Definiują funkcjonalność klasy
- Mogą przyjmować parametry i zwracać wartości
- Podobnie jak atrybuty, mają notację widoczności

3. Asocjacja

- Reprezentuje powiązanie między dwoma klasami



- Zazwyczaj przedstawiana jako linia z opcjonalnym opisem kierunku lub nazwą roli
- Może zawierać licznosc (np. 1..*), która określa liczbę instancji danej klasy powiązanych z drugą klasą

4. Agregacja

- Reprezentuje relację „część-całość” o słabym powiązaniu
- Przedstawiona jako linia z pustym rombem po stronie klasy „całości”
- Wskazuje na niezależność istnienia „części” od „całości” (części mogą istnieć niezależnie)

5. Kompozycja

- Silniejszy rodzaj relacji „część-całość”
- Przedstawiona jako linia z wypełnionym rombem po stronie klasy „całości”
- Części są ściśle związane z „całością” i nie mogą istnieć samodzielnie (usunięcie całości usuwa również części)

6. Dziedziczenie (Generalizacja)

- Reprezentuje relację „rodzic-dziecko” (klasa bazowa – klasa pochodna)
- Przedstawiona jako linia z pustym trójkątem wskazującym na klasę nadrzędną
- Klasa pochodna dziedziczy atrybuty i metody klasy bazowej

7. Realizacja

- Reprezentuje relację między klasą a interfejsem, który klasa implementuje
- Przedstawiona jako linia przerywana z pustym trójkątem skierowanym na interfejs
- Klasa implementująca jest zobowiązana do dostarczenia wszystkich metod interfejsu

5. Diagramy komponentów

Diagram komponentów

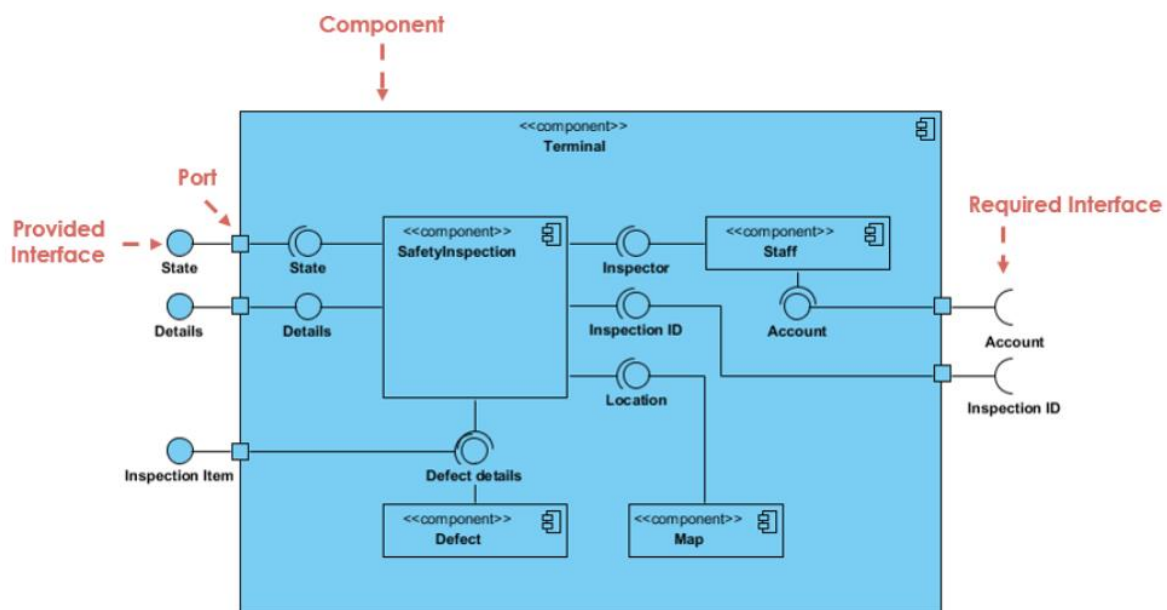
Zastosowanie diagramu komponentów

- Modelowanie fizycznej struktury systemu
- Przedstawianie zależności między komponentami oprogramowania
- Dokumentowanie modułowej architektury systemu
- Ukazanie sposobu organizacji kodu źródłowego, bibliotek, plików oraz usług zewnętrznych
- Wsparcie dla planowania wdrożenia systemu poprzez przedstawienie fizycznych zależności

Podstawowe elementy diagramu komponentów

1. Komponent

- Reprezentowany jako prostokąt z ikoną „pliku” w górnym rogu



Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>

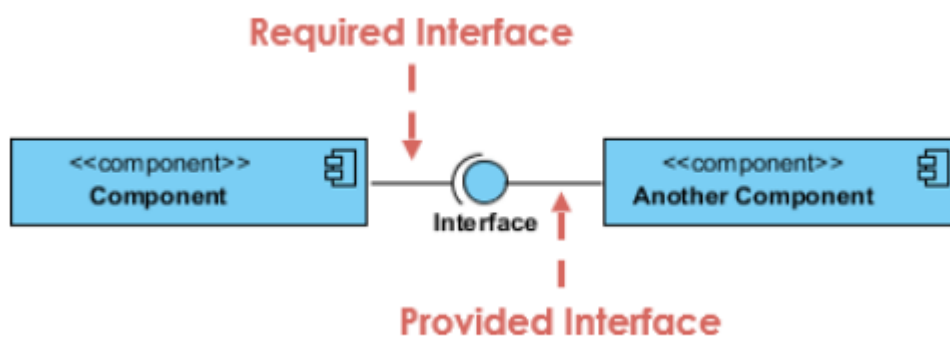
Data dostępu: 09.11.2024

- Przedstawia niezależny moduł oprogramowania, który pełni określoną funkcję
- Może reprezentować klasy, biblioteki, moduły, serwisy lub pakiety kodu

- Zazwyczaj zawiera nazwę, która identyfikuje pakiet lub moduł systemu, np. „komponent logiki biznesowej” lub „komponent bazodanowy”

2. Interfejs

- Przedstawiony jako:
 - Małe kółko połączone linią z komponentem (dla dostarczanych interfejsów)
 - „Gniazdo” (mały półokrąg) przy komponencie w przypadku interfejsu wymaganego



- Interfejs to kontrakt określający operacje, które komponent oferuje lub wymaga
- Pozwala na komunikację między komponentami bez ujawniania szczegółów implementacyjnych

3. Port

- Punkt na obramowaniu komponentu, do którego mogą być podłączone interfejsy

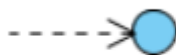


- Reprezentuje miejsce, w którym komponent odbiera lub wysyła komunikaty

- Porty mogą obsługiwać zarówno interfejsy dostarczane, jak i wymagane, ułatwiając organizację połączeń między komponentami

4. Zależność

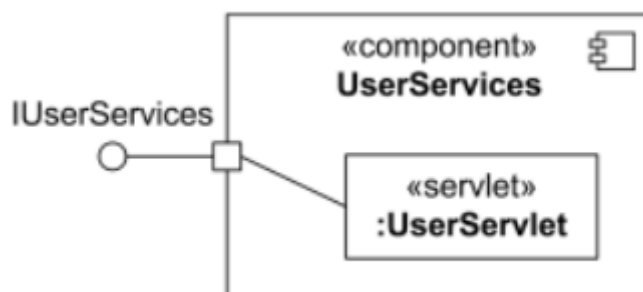
- Przedstawiona jako linia przerywana z otwartą strzałką



- Oznacza, że jeden komponent potrzebuje innego, aby realizować swoje funkcje
- Przykładem może być zależność komponentu interfejsu użytkownika od komponentu logiki biznesowej, aby pozyskać dane

5. Delegacja

- Przedstawiona jako linia ciągła od portu komponentu do interfejsu innego komponentu, który reprezentuje delegowane zadania



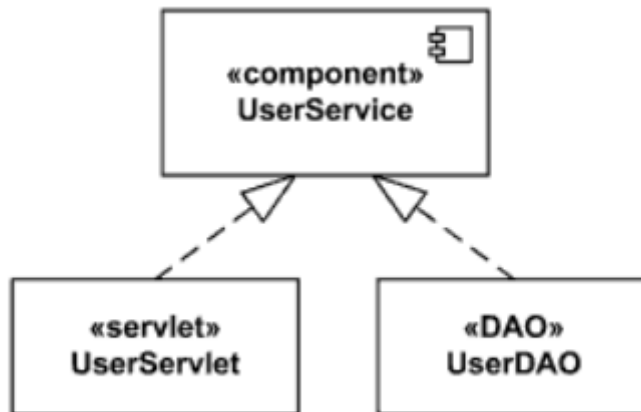
Źródło: <https://www.uml-diagrams.org/component-diagrams-reference.html>

Data dostępu: 09.11.2024

- Wskazuje na przekazanie operacji lub zadań między komponentami, np. delegacja przetwarzania danych do wyspecjalizowanego komponentu

6. Realizacja

- Przedstawiona jako linia ciągła z pustym trójkątem wskazującym na interfejs, który komponent implementuje lub dostarcza



Źródło: <https://www.uml-diagrams.org/component-diagrams-reference.html>

Data dostępu: 09.11.2024

- Wskazuje, że komponent implementuje kontrakt określony przez interfejs, umożliwiając interakcję z innymi komponentami na poziomie abstrakcji

7. Artefakt

- Reprezentowany jako prostokąt z ikoną „dokumentu”



- Fizyczny zasób, który realizuje komponent, np. plik JAR, DLL, lub wykonywalny plik
- Artefakty mogą być wdrażane na serwerze, urządzeniu lub systemie i odzwierciedlają faktyczne zasoby wykorzystywane w systemie

6. Podział modelu na pakiety

Diagram pakietów

Zastosowanie diagramu pakietów

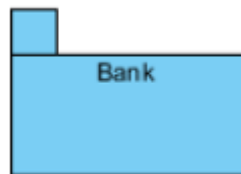
- Modelowanie struktury logicznej systemu

- Organizacja klas i komponentów w grupy (pakiety) dla lepszej czytelności i zarządzania
- Przedstawienie zależności między pakietami
- Ułatwienie modularności i zarządzania kodem źródłowym w dużych projektach
- Identyfikacja relacji między pakietami, co wspomaga proces refaktoryzacji i organizacji projektu

Podstawowe elementy diagramu pakietów

1. Pakiet

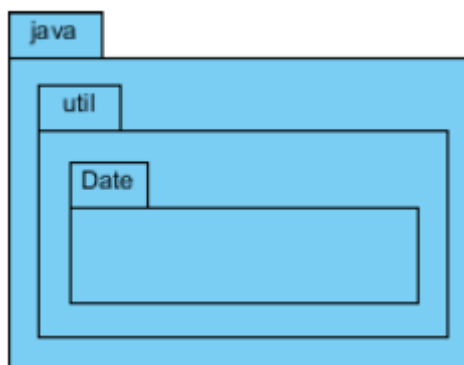
- Reprezentowany jako prostokąt z zakładką w lewym górnym rogu



- Oznacza logiczną grupę powiązanych klas, interfejsów lub komponentów, która reprezentuje określoną funkcjonalność
- Pakiety często odpowiadają strukturze katalogów w projekcie lub modułom biznesowym, np. „pakiet logiki biznesowej” lub „pakiet interfejsu użytkownika”
- Może zawierać inne pakiety, co pozwala na tworzenie hierarchii pakietów

2. Zawartość pakietu

- Elementy wewnątrz pakietu, takie jak klasy, interfejsy lub inne pakiety

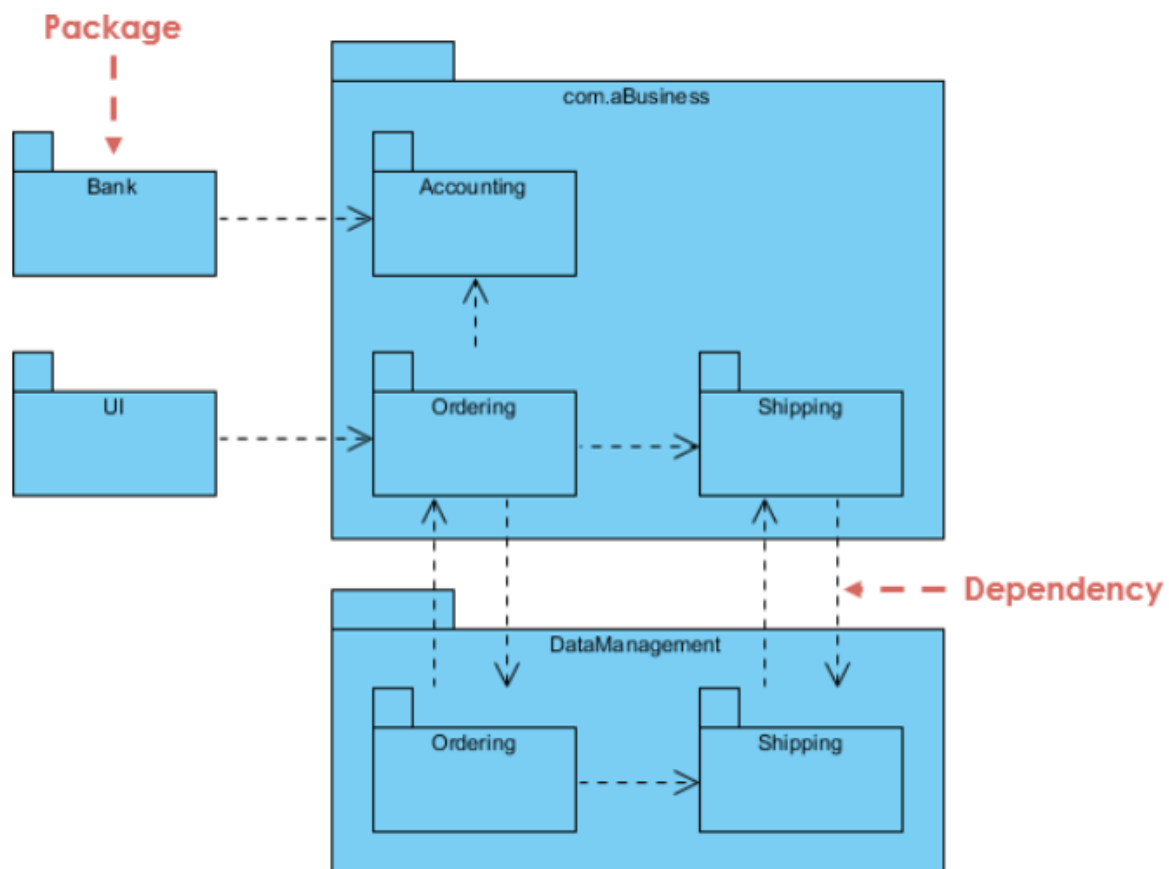


- Mogą być przedstawione w ramach prostokąta pakietu lub pozostawione jako abstrakcyjna reprezentacja pakietu bez szczegółowej zawartości

- Przedstawianie zawartości jest opcjonalne, szczególnie w przypadku dużych diagramów, aby zachować czytelność

3. Zależność między pakietami

- Przedstawiona jako linia przerywana z otwartą strzałką skierowaną do pakietu zależnego



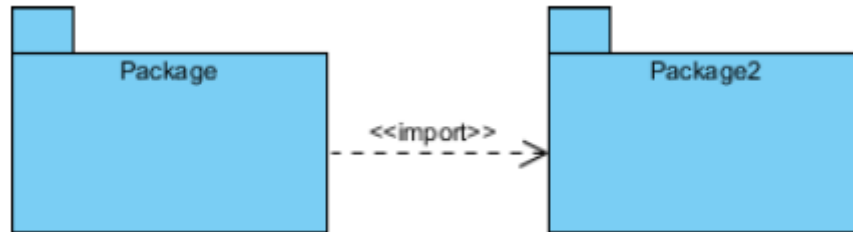
Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/>

Data dostępu: 09.11.2024

- Pokazuje, że jeden pakiet wymaga elementów z innego pakietu, aby funkcjonować
- Zależności pomagają w analizie zależności między modułami systemu i identyfikacji potencjalnych problemów z cyklicznymi zależnościami

4. Importowanie pakietu

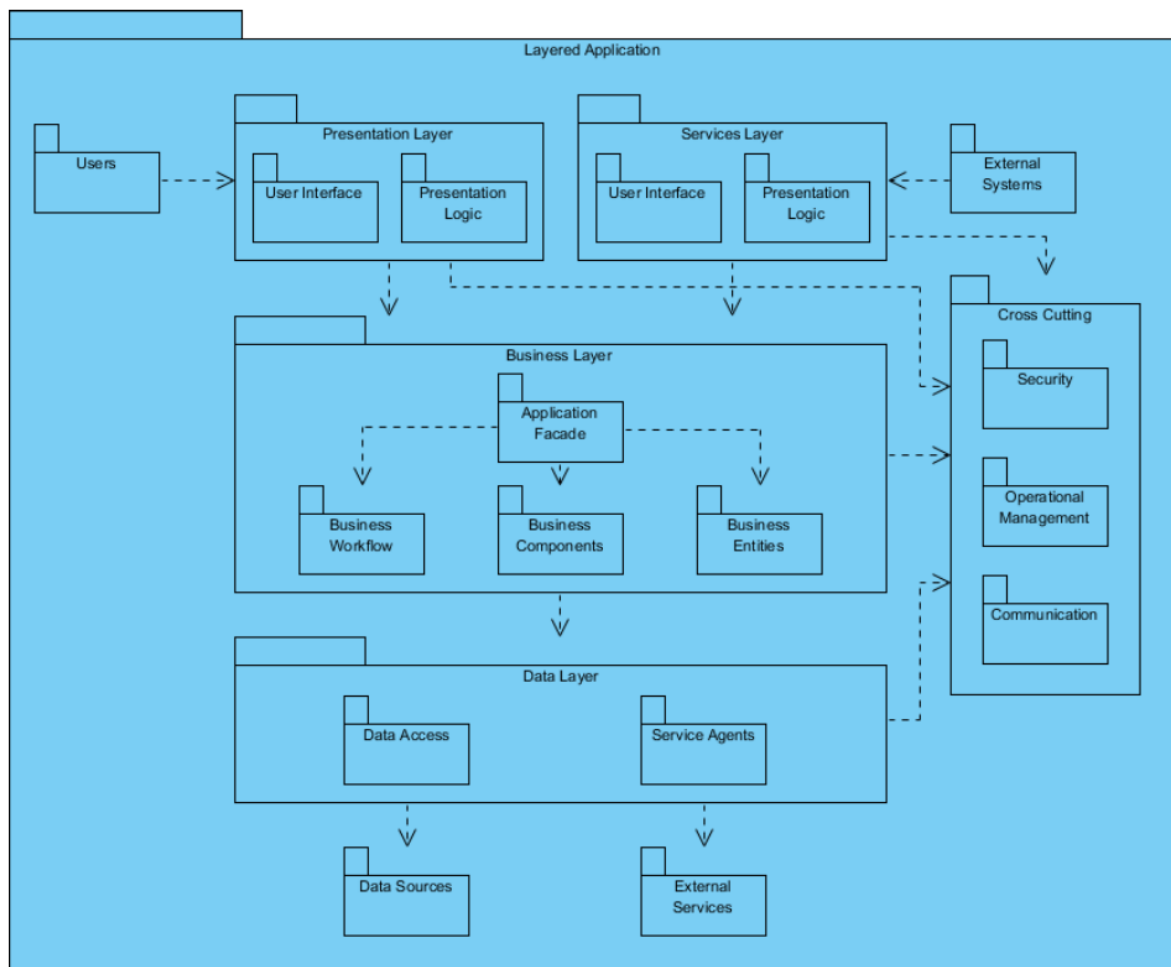
- Przedstawione jako zależność, która oznacza, że pakiet importuje klasy lub interfejsy z innego pakietu



- Często stosowane, aby podkreślić wykorzystanie zasobów z innych części systemu w ramach konkretnego modułu

5. Hierarchia pakietów

- Przedstawiona przez umieszczanie jednego pakietu wewnątrz drugiego lub przez zależności między pakietami



Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/>

Data dostępu: 09.11.2024

- Hierarchia pakietów pomaga w uporządkowaniu struktury systemu i tworzeniu logicznych grup o określonej odpowiedzialności
- Umożliwia organizację projektu według warstw, np. warstwa prezentacji, warstwa logiki biznesowej, warstwa dostępu do danych

7. Modelowanie wdrożenia systemu

Diagram wdrożenia

Zastosowanie diagramu wdrożenia

- Modelowanie fizycznego wdrożenia systemu
- Przedstawienie konfiguracji sprzętu i oprogramowania oraz zależności między nimi
- Wizualizacja rozlokowania komponentów na różnych urządzeniach i serwerach
- Wsparcie dla planowania i zarządzania infrastrukturą systemu
- Dokumentowanie topologii sieci oraz połączeń między węzłami

Podstawowe elementy diagramu wdrożenia

1. Węzeł (Node)

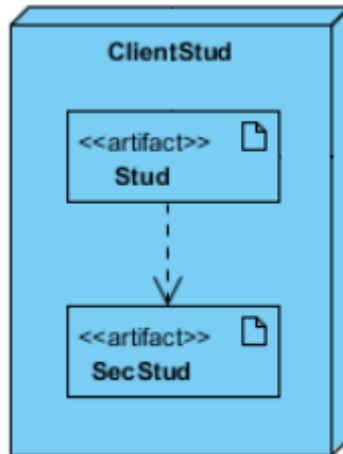
- Reprezentowany jako trójwymiarowy prostokąt (sześcian)



- Przedstawia fizyczny lub wirtualny zasób sprzętowy (np. serwer, komputer, urządzenie mobilne) lub środowisko wykonawcze (np. maszyna wirtualna)
- Węzły mogą mieć etykiety opisujące ich funkcję lub typ (np. „Serwer aplikacji”, „Baza danych”)
- Węzły mogą zawierać informacje o systemie operacyjnym, specyfikacji sprzętowej oraz oprogramowaniu uruchomionym na danym urządzeniu

2. Artefakt

- Reprezentowany jako prostokąt z ikoną dokumentu lub pliku



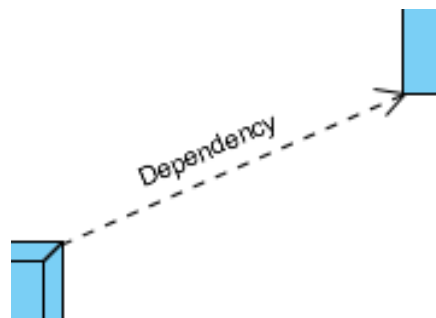
Źródło: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/>

Data dostępu: 09.11.2024

- Przedstawia fizyczny plik lub zasób, np. plik wykonywalny, bibliotekę, skrypt lub plik konfiguracyjny, który jest wdrażany na węzłach
- Artefakt jest realizacją komponentu i jest umieszczany na węźle, gdzie ma być uruchomiony lub wykorzystywany
- Przykłady artefaktów to plik JAR, plik DLL, plik EXE

3. Zależność między węzłami

- Przedstawiona jako linia przerywana z otwartą strzałką skierowaną do zależnego węzła



Źródło: <https://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html>

Data dostępu: 09.11.2024

- Wskazuje, że jeden węzeł wymaga zasobów lub usług dostępnych na innym węźle, np. serwer aplikacji może zależeć od węzła bazy danych
- Pomaga w analizie zależności i połączeń między elementami systemu

4. Komunikacja między węzłami

- Przedstawiona jako linia ciągła z etykietami oznaczającymi protokół lub typ połączenia, np. HTTP, TCP/IP, gRPC



- Reprezentuje połączenie sieciowe lub interakcję między węzłami systemu
- Komunikacja pomaga zrozumieć, jak dane przepływają przez system oraz jakie zasoby są potrzebne do wymiany informacji

5. Grupowanie węzłów (Node Clustering)

- Grupowanie kilku węzłów w ramach jednego większego węzła, co reprezentuje np. klastry serwerów lub infrastruktury chmurowej
- Może być przedstawione jako jeden większy węzeł zawierający inne węzły lub jako węzły o nazwach wskazujących na klaster
- Stosowane w przypadku systemów rozproszonych, aby ułatwić modelowanie infrastruktury opartej na wielu zasobach

6. Procesy i aplikacje uruchomione na węzłach

- Przedstawione jako etykiety lub dodatkowe elementy wewnątrz węzła, które symbolizują uruchomione procesy, usługi lub aplikacje
- Pomagają zrozumieć, jakie konkretne operacje są realizowane przez węzeł oraz jakie komponenty są uruchomione w środowisku
- Przykładowo, węzeł „Serwer aplikacji” może mieć procesy „Serwer Tomcat”, „Serwer HTTP” itp.

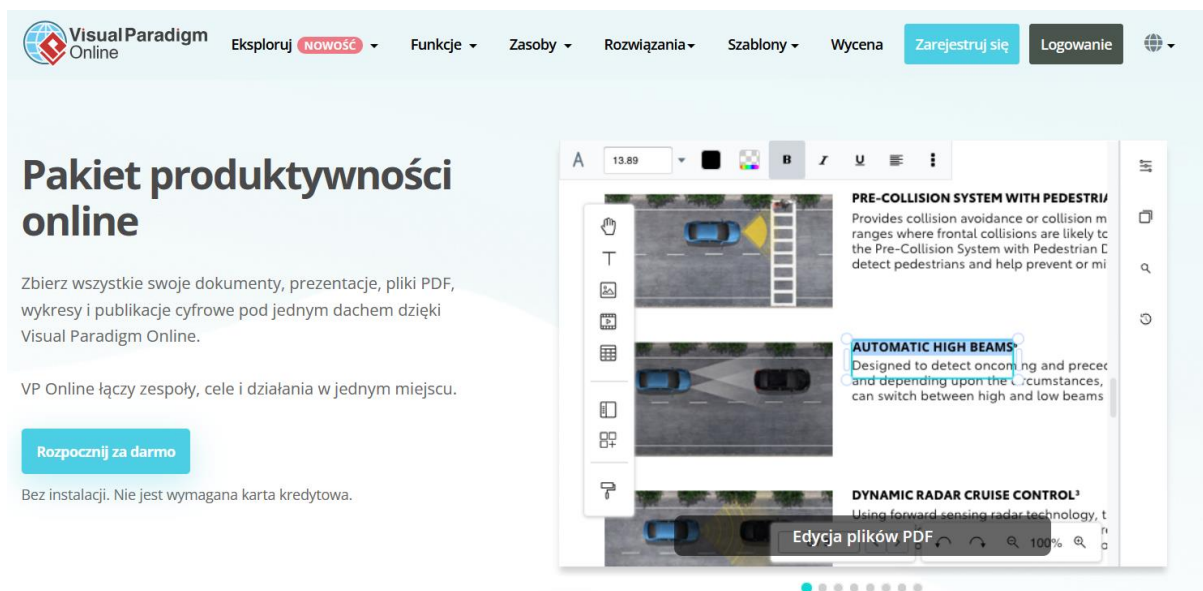
8. Visual Paradigm online – opis działania oraz alternatywy

Visual Paradigm online

Żeby stworzyć diagramy UML w aplikacji webowej Visual Paradigm należy w pierwszej kolejności zaożyć konto w serwisie.

Czynności:

1. Kliknięcie “Zarejestruj się” na landing page-u



2. Wypełnienie formularza rejestracji i wysłanie formularza



Jedno rozwiązanie. Nieskończone możliwości.

Zarejestruj się, aby zacząć tworzyć niesamowite treści. **Fliplify**, **AniFuzion** and more.

☒ **Korzystanie z wersji bezpłatnej**

* Możesz zdecydować się na 30-dniowy okres próbny w dowolnym momencie.



Skorzystaj z 30-dniowej wersji próbnej Combo Edition

[Porównanie funkcji](#)

Adres e-mail

Wprowadź swój służbowy adres e-mail

Zarejestruj się

lub zarejestrować się za pomocą:



Zarejestruj się w Google



Zarejestruj się w Microsoft

Masz już konto? [Zaloguj się](#)

3. Aktywowanie konta

Witamy w Visual Paradigm!

Twój obszar roboczy online jest gotowy do pracy. Kliknij poniżej, aby rozpocząć.

[Odwiedź swój obszar roboczy Visual Paradigm](#)

Aby **aktywować swoje konto**, po prostu kliknij link w wiadomości e-mail, którą wysłaliśmy i gotowe! Jeśli nie możesz znaleźć naszej wiadomości e-mail, sprawdź folder spamu.

Zapraszamy!

Przy okazji, jeśli masz znajomych lub współpracowników, którzy chcieliby korzystać z Visual Paradigm razem z Tobą, po prostu zaloguj się i zaproś ich! Nie ma limitu liczby osób, które możesz zaprosić. Zacznijmy współpracować razem!

Activate your Visual Paradigm account



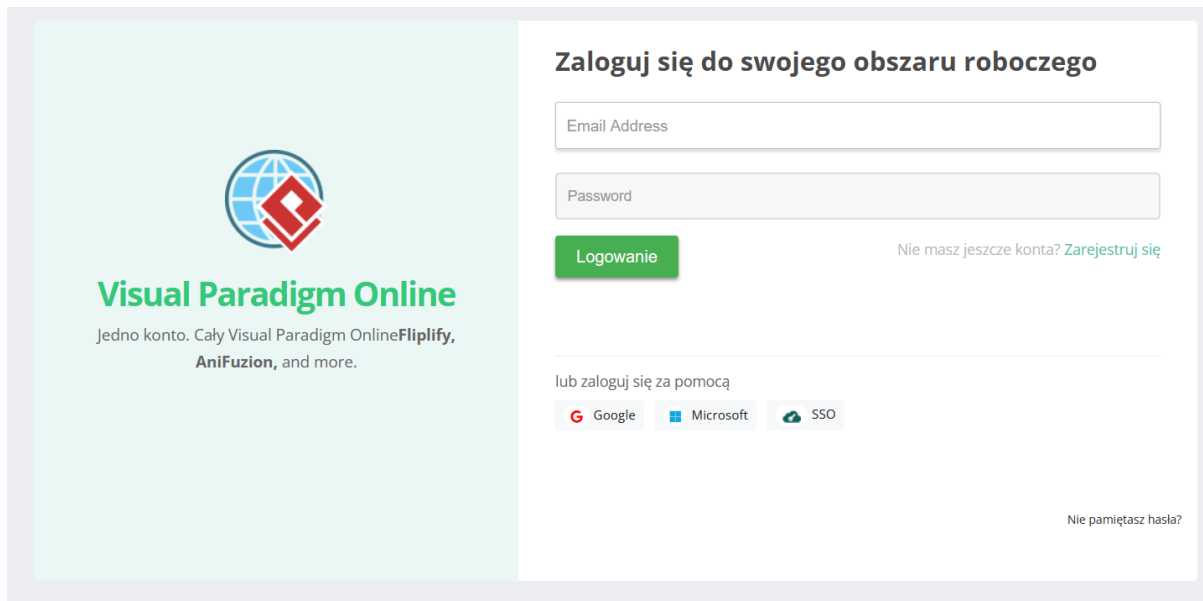
••



••

Confirm

4. Zalogowanie się



The image shows the login page for Visual Paradigm Online. On the left, there is a light blue vertical banner with the Visual Paradigm Online logo (a globe with a red diamond) and the text "Visual Paradigm Online" in green. Below the logo, it says "Jedno konto. Cały Visual Paradigm Online, Fliplify, AniFuzion, and more." On the right, the main login area has a title "Zaloguj się do swojego obszaru roboczego". Below the title are two input fields: "Email Address" and "Password". A green "Logowanie" button is positioned below the password field. To the right of the button, there is a link "Nie masz jeszcze konta? Zarejestruj się". Below the login fields, there is a section titled "lub zaloguj się za pomocą" with three buttons: "Google", "Microsoft", and "SSO". At the bottom right of the login area, there is a link "Nie pamiętasz hasła?".

Zaloguj się do swojego obszaru roboczego

Email Address

Password

Logowanie Nie masz jeszcze konta? [Zarejestruj się](#)


lub zaloguj się za pomocą


[Google](#) [Microsoft](#) [SSO](#)

[Nie pamiętasz hasła?](#)

Po stworzeniu konta należy kliknąć w bocznym menu opcję “Diagram” i po odbyciu nawigacji na stronie kliknąć przycisk “CREATE” i wybrać opcję “Use case diagram”.

New



 **Visual Paradigm**
Online

Draft

Home

My Drive

Animations

Flipbooks

Slideshows

Shelf

PDF


Document

Presentation

Spreadsheet

Diagram

Get Started



Create an Animation

Elevate your content with creative animations and 3D characters.

You might want to try...

Suggested

Visual

Diagram

Recent Documents

You don't have any designs yet.

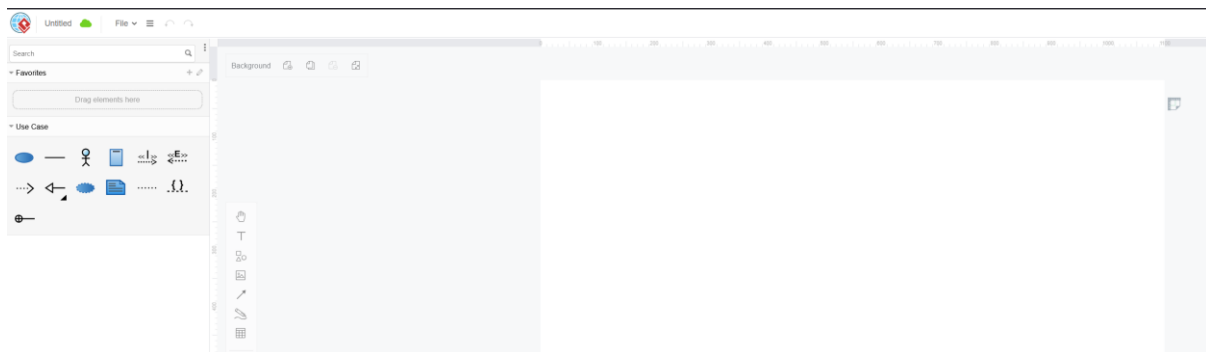
My Diagrams

+ CREATE

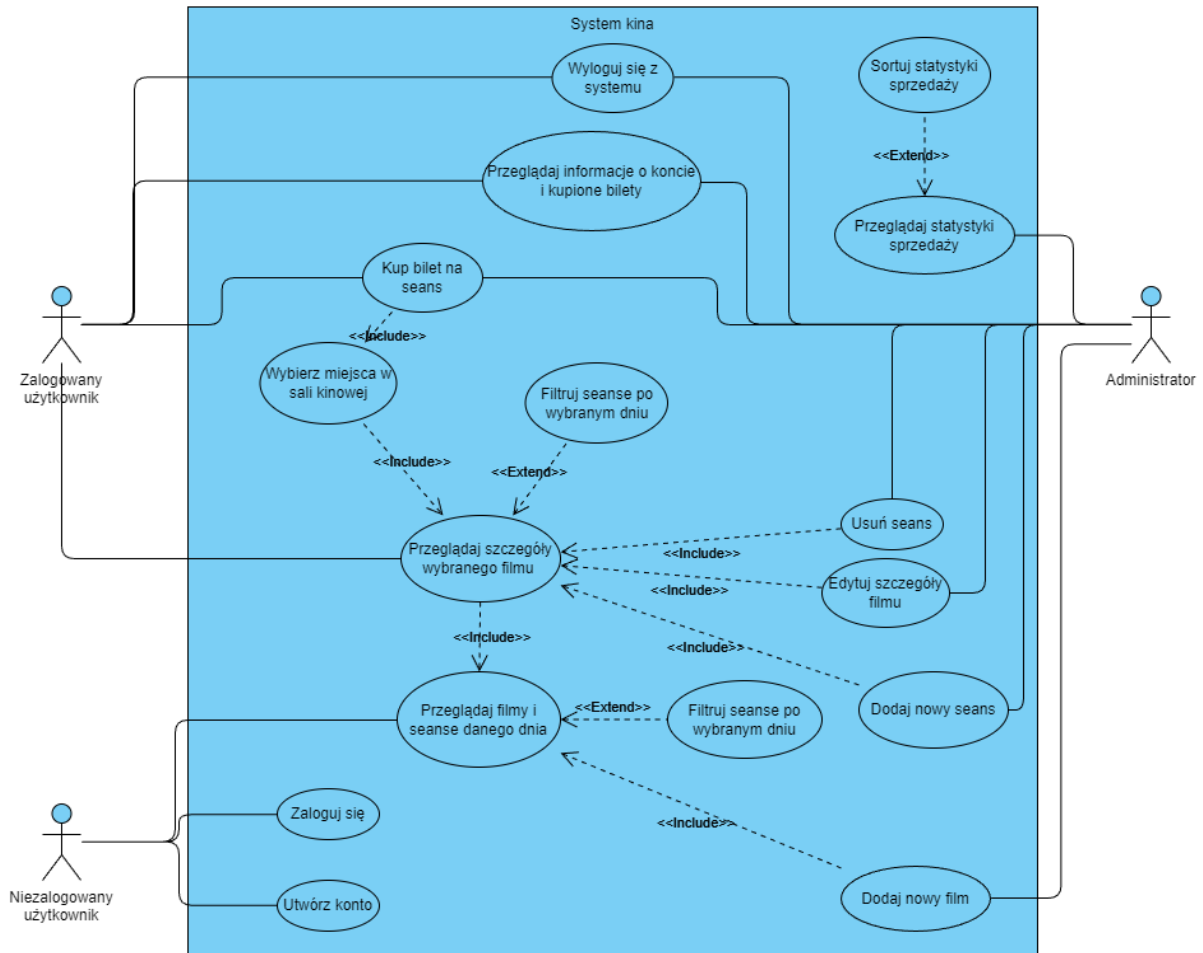
🔍 |Try "infographic" or "flowchart"

- ☆ 📦 Class Diagram
- ☆ 📦 Use Case Diagram
- ☆ 📦 Sequence Diagram
- ☆ 📦 Activity Diagram
- ☆ 📦 Deployment Diagram
- ☆ 📦 Component Diagram
- ☆ 📦 State Machine Diagram
- ☆ 📦 Package Diagram
- ☆ 📦 Data Flow Diagram
- ☆ 📦 Entity Relationship Diagram
- ☆ 📦 Requirement Diagram
- ☆ 📦 Block Definition Diagram
- ☆ 📦 Internal Block Diagram
- ☆ 📦 Parametric Diagram

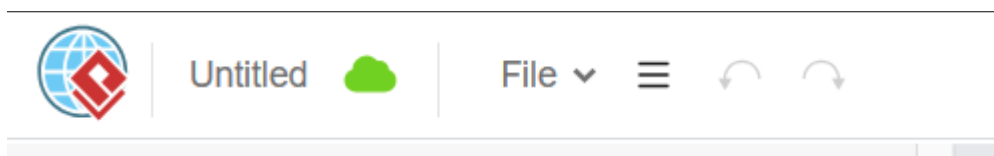
Po utworzeniu diagramu, następuje przekierowanie na stronę z edytorem diagramu, gdzie po lewej stronie znajdują się dostępne narzędzia i główne opcje edytora, a na środku znajduje się edytowany dokument z diagramem.



Przykład diagramu stworzonego w edytorze w aplikacji Visual Paradigm Online:



Aby nadać/zmienić tytuł dokumentu, należy kliknąć w jego nazwę (domyślnie “untitled”) w lewym górnym rogu, wypełnić pole pożądaną nazwą i kliknąć przycisk “rename”



×

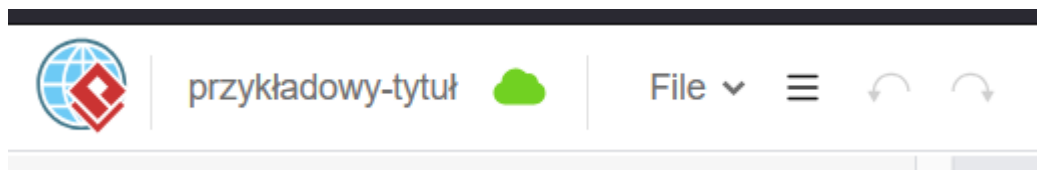
Filename:

przykładowy-tytuł

Cancel

Rename

Po zmianie:




Aby zapisać dokument, należy w lewym górnym logu kliknąć “File” -> “Save as” -> uzupełnić pożądaną nazwę pliku -> “SAVE”

Save as

File name
przykładowa-nazwa-pliku

Storage Media:



Visual Paradigm Workspace
Store your work in the Visual Paradigm cloud workspace for seamless access across all devices.

Current Storage: Not yet saved

MORE OPTIONS

CANCEL SAVE

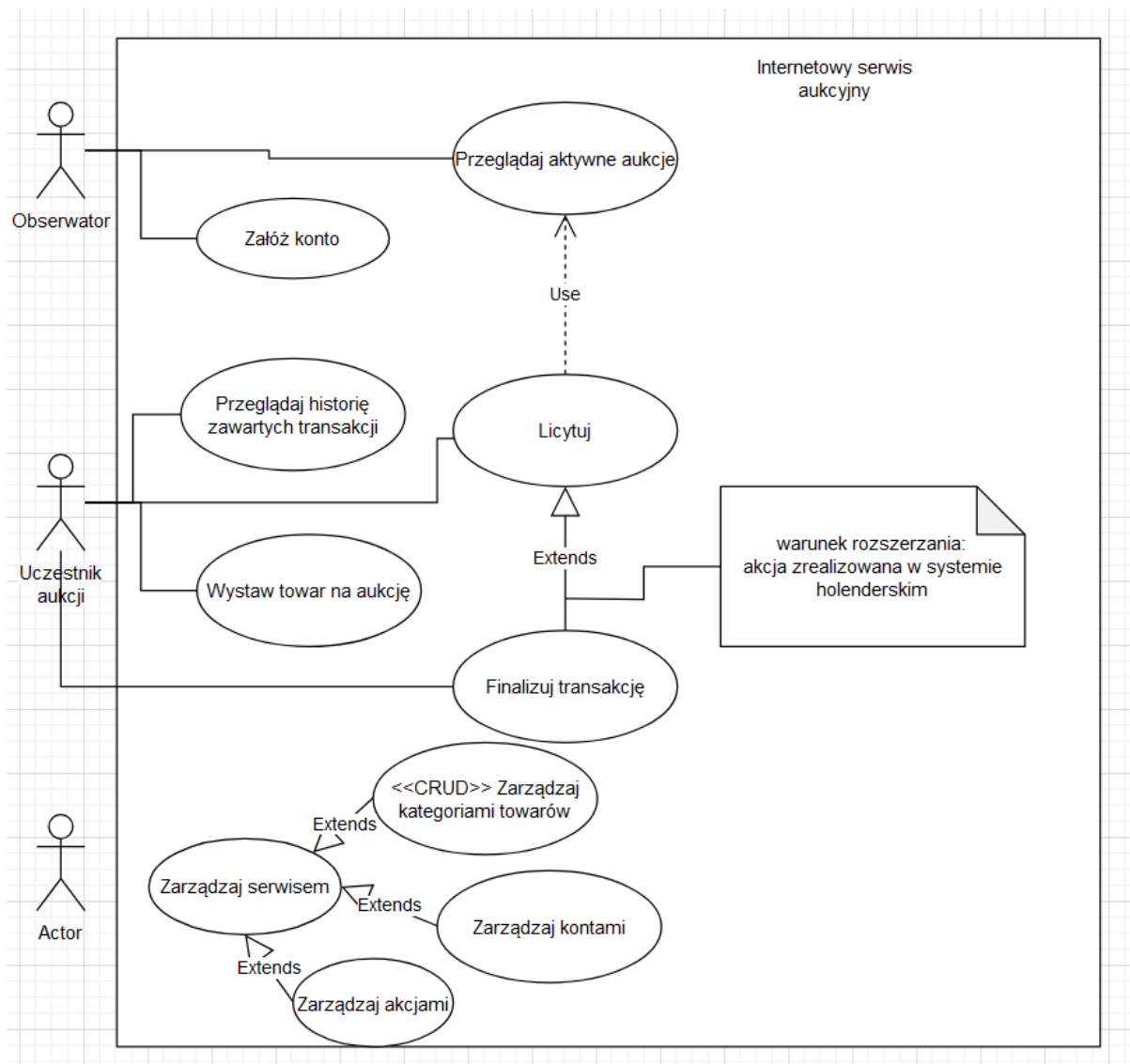
Strona z posiadanymi diagramami po zapisaniu dokumentu:

My Diagrams + CREATE		<input type="text" value="search"/>	
<input type="checkbox"/>	Name	Last Modified	Actions
<input type="checkbox"/>	przykładowa-nazwa-pliku	09/11/2024	

Alternatywa do Visual Paradigm Online – drawio

Innym serwisem, gdzie można m.in. tworzyć diagramy UML, w tym diagram przypadków użycia, jest serwis “drawio”.

Poniżej znajduje się przykładowy diagram przypadków użycia stworzony w tym serwisie:



9. Wnioski

Potencjalne przypadki użycia UML

Wykorzystanie UML w planowaniu procesu wdrożenia produktu informatycznego:

- Zapewnia standardową notację zrozumiałą dla wszystkich interesariuszy produktu
- Umożliwia precyzyjne modelowanie różnych aspektów systemu
- Wspiera komunikację w zespole projektowym
- Ułatwia dokumentację systemu
- Pomaga w identyfikacji potencjalnych problemów na wczesnym etapie rozwoju

Potencjalne problemy z używaniem UML:

Powyższe punkty mogą być prawdziwe zakładając dobrą znajomość i doświadczenie w posługiwaniu się językiem UML projektantów oraz osób odczytujących zaprojektowane modele. W przeciwnym wypadku korzystanie z UML i odczytywanie UML może być nieefektywne i kontrproduktywne.

Należy również zauważyć, że tworzenie diagramów UML jest czasochłonnym procesem i bez odpowiedniego powodu i usprawiedliwienia takiej formy dokumentacji systemu może nie być ona pożądana.

Oprócz tego, wraz z rozrostem systemu diagramy UML mogą stawać się nieczytelne, co zabija cel ich tworzenia.

Podsumowanie

Język UML może stanowić skuteczne narzędzie w procesie planowania wdrożenia produktu informatycznego. Jego wszechstronność i standaryzacja mogą przyczynić się do lepszego zrozumienia systemu przez wszystkich uczestników projektu oraz efektywniejszego zarządzania procesem rozwoju oprogramowania. UML nie jest natomiast panaceum na problemy z dokumentowaniem wszystkich aplikacji i systemów i przed jego zastosowaniem należy się zastanowić, czy w danym scenariuszu cele jego wykorzystania są realistyczne do zrealizowania. Może się okazać, że nie są np. z powodu:

- Słabej lub żadnej znajomości języka UML przez zespół pracujący nad systemem
- Słabej lub żadnej znajomości języka UML przez zewnętrznych interesariuszy
- Zbyt dużego poziomu skomplikowania systemu (w tym przypadku warto się zastanowić, czy zamiast stosować UML do opisu całego systemu, nie lepiej wypracować indywidualny sposób dokumentacji i zastosować UML do jego uzupełniania/komplementowania w określonych przypadkach)

- Zbyt dużego na dany budżet nakładu czasu związanego z tworzeniem diagramów i dokumentacji UML (tutaj również warto zastanowić się nad indywidualnym sposobem dokumentacji do zastosowania w konkretnym projekcie)