Modelowanie systemu

Spis

Czym jest Model ?

- Modele są budowane w celu lepszego zobrazowania istniejących lub przyszłych systemów
- Model nigdy nie będzie w pełni odpowiadał rzeczywistości
- Modelowanie jest nierozerwalnie związane z uwypuklaniem pewnych cech i pomijaniem innych
- Modelowanie można określić jako próbę uchwycenia w kategoriach informatycznych i wyrażenia za pomocą specjalnej notacji graficznej
- Nie ma uniwersalnej odpowiedzi na pytanie o to, co wyróżnić a co pominąć.

Jakie zadania realizuje model systemu ?

- Opisanie modelu komunikacji i powiązań między elementami systemu
- Zobrazowanie przebiegu wszystkich procesów z punktu widzenia klientów, specjalistów i użytkowników
- Weryfikacja faktów pod kątem kompletności spójności i poprawności

Typy Modelu

Modele systemu można podzielić na dwie podstawowe grupy:

- Modele zachowania systemu (aspekt dynamiczny)
- Modele struktury systemu (aspekt statyczny)

W modelach zachowania istotny jest zestaw działań realizowanych przez system. Działania należą zazwyczaj do dwóch grup:

- Zmiana stanu systemu
- Przetwarzanie danych wejściowych w dane wyjściowe

Modele struktury systemu pokazują statyczne elementy składowe systemu i ich wzajemne zależności. Istnieją rodzaje modeli ukazujące oba aspekty systemu.

Modele Maszyn Stanowych

Modelami zachowania ukierunkowanymi na zmiany stanów systemu są modele maszyn stanowych (state machine models).

- Modele maszyn stanowych odpowiadają systemom, w których występuje pewna liczba stanów systemu, a przejścia między stanami odbywają się z małym udziałem przetwarzania danych. Przykładami takich systemów są rozmaite systemy czasu rzeczywistego oraz systemy sterowania.

Modele Maszyn Stanowych

- W przypadku stosowania modelu maszyny stanowej do oprogramowania obiektowego, maszyna stanowa może dotyczyć obiektu (egzemplarza klasy). Modele stanu dla obiektów mają praktyczne znaczenie w przypadku obiektów, które mogą znajdować się w kilku stanach i których zachowanie jest różne w zależności od stanu w którym się znajdują. Diagram stanu dla obiektu staje się ilustracją jego cyklu życia (object lifecycle).
- Oprogramowanie odpowiadające maszynie stanowej bywa złożone i istnieją odpowiednie wzorce projektowe służące do rozwiązania tego problemu. Problemem modeli maszyn stanowych jest gwałtownie rosnąca liczba możliwych stanów w bardziej złożonych systemach.

Modele przepływu

- **Modelami zachowania ukierunkowanymi na aspekty przetwarzania danych (nazywanymi także modelami przepływu danych – data flow models) są:**
- **modele przepływu sterowania (control flow models)**
- **modele przepływu obiektów (object flow models)**
- W obu wypadkach elementami modelu są działania przekształcające dane, a powiązania między elementami oznaczają przekazanie danych. W modelach przepływu obiektów istotną rolę w diagramach odgrywają elementy przedstawiające dane (mogą to być obiekty, ale mogą też pliki lub struktury).

Modele struktury

Modele struktury systemu koncentrują się na statycznych aspektach jego budowy.

Elementami systemu są w takim ujęciu:

- podsystemy
- komponenty
- interfejsy
- klasy
- obiekty

Należy zwracać uwagę na rozróżnienie abstrakcyjnej struktury systemu (potencjalna zawartość systemu) od struktury systemu w trakcie wykonania (konkretna jednostkowa realizacja możliwości systemu). Zapisany program opisuje potencjalną zawartość systemu, uruchomiony dla konkretnych danych staje się systemem konkretnym jednostkowym.

Modele danych

- Osobnym zagadnieniem w modelowaniu systemu jest modelowanie sposobu przechowywania i dostępu do podstawowych danych, na których operuje system.
- W większości dużych systemów dane te przechowywane są w bazie danych. Zagadnienie przenosi się wtedy na problem modelowania bazy danych.Bazy danych dostarczane są zazwyczaj z odpowiednimi systemami zarządzania. W ramach systemu konieczne staje się więc modelowanie interakcji z systemem zarządzania bazą danych.

Narzędzia CASE

Narzędzia CASE wykorzystywane do modelowania są często związane z konkretną metodologią rozwijania oprogramowania.

Elementami środowiska wspomagania modelowania są np.:

- edytory diagramów wybranej notacji
- narzędzia sprawdzania poprawności modeli
- narzędzia korzystania z repozytorium danych
- słownik danych
- narzędzia tworzenia raportów i innej dokumentacji systemowej
- narzędzia projektowania i wypełniania formularzy
- narzędzia wymiany danych z zewnętrznymi zasobami
- generatory kodu

Unified Modeling Language – geneza i ewolucja

UML został opracowany przez Grady Booch , Ivar Jacobson i

James Rumbaugh w Rational Software w latach 1994–1995, a dalszy rozwój prowadzili przez nich do 1996 r.

W styczniu roku 1997 powstaje wersja UML 1.0 przekazana jako propozycja standardu organizacji OMG (Object Managment Group) .

Sierpień 2003 została zaprezentowana wersja 2.0

W grudniu 2017 została wypuszczona wersja 2.5.1

2.5.1	December 2017
2.4.1	July 2011
2.3	May 2010
2.2	January 2009
2.1.2	October 2007
2.0	July 2005
1.5	March 2003
1.4	September 2001
1.3	February 2000
1.2	July 1999
1.1	December 1997

Unified Modeling Language – geneza i ewolucja

UML - ujednolicony język modelowania systemów informatycznych.

Modelowanie systemów informatycznych obejmuje aspekty funkcjonalne oraz niefunkcjonalne. Możliwość ich modelowania była i nadal jest motywem ewolucji języka UML, którego kolejne wersje są wzbogacane o nowe kategorie, będące odpowiedzią na wymagania twórców systemów informatycznych stawiane metodom i technikom modelowania. Założżenia autorów języka UML wykraczały poza dokonanie prostej kompilacji.

Czym UML nie jest

- językiem programowania, chociaż istnieje możliwość generowania kodu z niektórych diagramów
- narzędziem, choć zawiera specyfikacje dla narzędzi
- metodyką, nie definiuje procesu tworzenia oprogramowania, chociaż istnieją metodyki bazujące na UML
- nie jest sposobem na analizę i projektowanie systemów komputerowych

Diagramy UML

Język UML przyjmuje w praktyce postać graficznej reprezentacji tworzonego systemu, składającej się z logicznie powiązanych z sobą diagramów. Wyróżniamy następujące ich kategorie:

- diagramy abstrakcyjne
- diagramy konkretne.

W standardzie UML występuje trzynaście rodzajów diagramów, które charakteryzują statystykę i dynamikę tworzonego systemu.

Opis Klas

- Klasy: nazwa, atrybuty (pola), operacje (metody)

[widoczność] nazwa [:typ] [wielokrotność][= wartość domyślna]

[widoczność] nazwa [(lista parametrów)] [:typ zwracany]

Poziom widoczności

  - prywatne, tylko w obrębie klasy
  - publiczne, dostęp globalny
  - chronione, dostęp przez dziedziczenie
  - zakres pakietu

- Zakres
  - klasowy (classifier), wspólny dla klas, pola i metody statyczne
  - instancji, tylko w zakresie obiektu
- Klasy abstrakcyjne, metody abstrakcyjne

Dodawanie diagramów klas do projektu

- Narzędzia do kodowania
- Musimy posiadać Projektant klas (Class Designer)
- Jak już to posiadamy w projekcie ctrl + shift + a która otwiera mam dodawanie nowych elementów
- Nastepnie ogólne(GENERAL) i diagram klas

Przykładowy Diagram

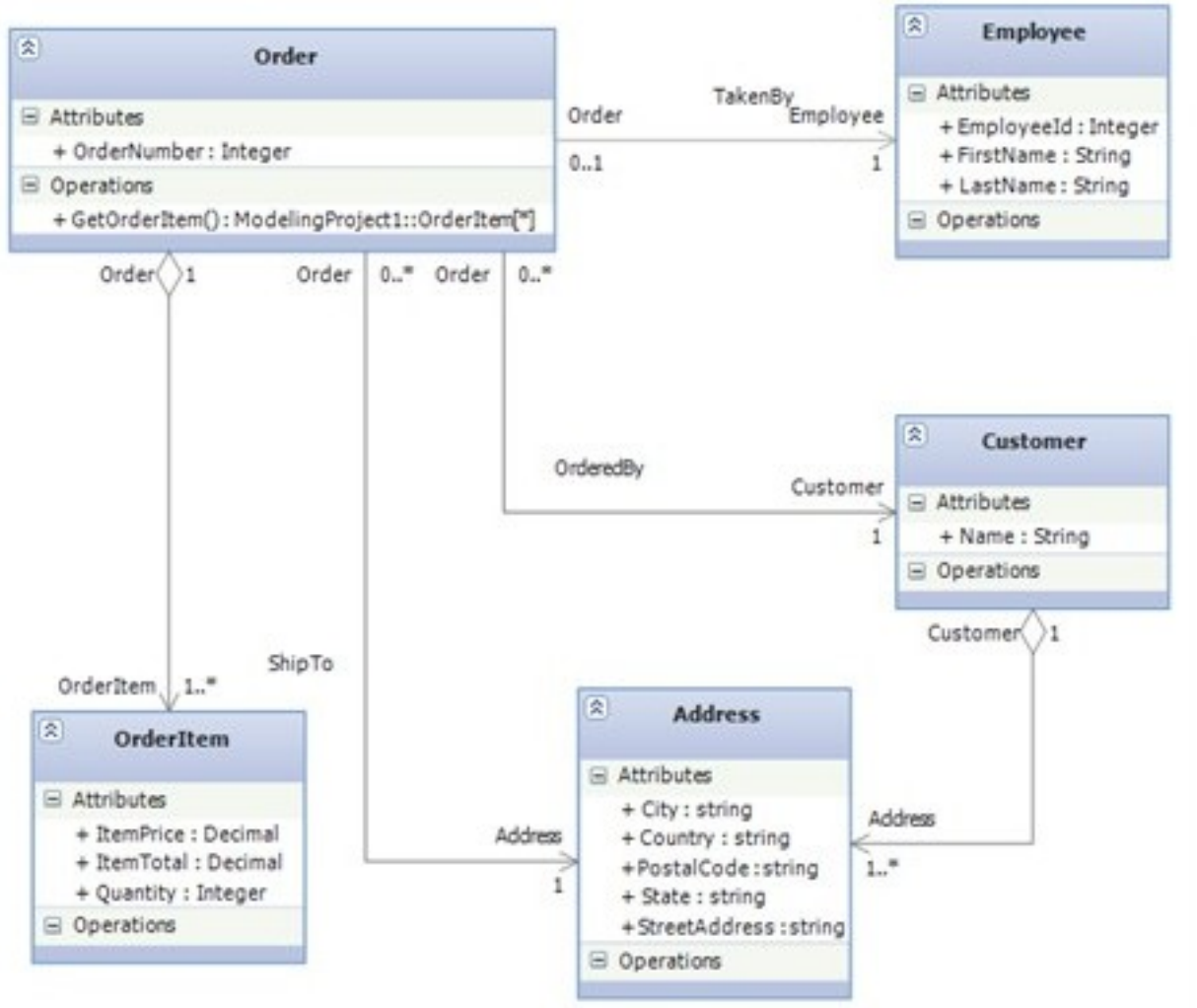


Diagram Pakietów

Pakiet (ang. package) jest uniwersalnym mechanizmem służącym do organizowania elementów w grupy. Pakiet grupuje elementy modelu i diagramy.

Pakiet stanowi obszar nazw, co pozwala na oznaczenie przynależności (właściciela) bytu, a w konsekwencji umożliwia lepsze i skuteczniejsze zarządzanie elementami systemu. W związku z tym bytu należące do tego samego pakietu muszą mieć unikatowe nazwy.

Pakiety to jeden z najczęściej stosowanych elementów dzięki któremu staram się utrzymać porządek w repozytorium projektu.

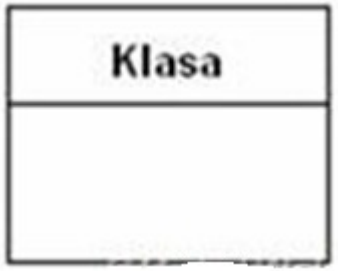


Diagram klas

Diagram klas obrazuje pewien zbiór klas, interfejsów i kooperacji oraz związki między nimi. Jest on grafem złożonym z wierzchołków (klas, interfejsów, kooperacji) i łuków (reprezentowanych przez relacje). Diagram klas stanowi opis statyki systemu, który uwypukla związki między klasami, pomijając pozostałe charakterystyki.

Klasa stanowi opis wybranego podzbioru obiektów, w którym każdy z obiektów posiada takie same atrybuty, operacje, metody, związki i znaczenie. Z określenia tego wynika założenie dotyczące inwariantności (niezmienności) cech i zachowań obiektów, którym przypisano wspólny klasyfikator w postaci nazwy klasy.

Związek asocjacji (powiązania)

Związek asocjacji to semantyczna relacja (związek) pomiędzy dwoma bądź większą liczbą klas, która ustanawia powiązania (wiązania) pomiędzy instancjami klasyfikatorów. W szczególności asocjacja może zachodzić pomiędzy dwoma lub większą liczbą klas. Często do powiązania dodawane są nazwy ról i liczebności.

Związek agregacji

Związek agregacji to rodzaj asocjacji pomiędzy klasyfikatorami, która określa relację „całość-część” pomiędzy agregatem (całością) a jego częściami; egzemplarz należący do klasyfikatora reprezentującego „całość” zawiera – jako swoje komponenty – elementy należące do klasyfikatora reprezentującego „część”. W przypadku klas, wartościami atrybutów obiektu zagregowanego mogą być obiekty należące do klas reprezentujących „części”.

Silna agregacja (kompozycja)

Związek silnej agregacji (kompozycja, agregacja całkowita) to rodzaj agregacji z przynależnością elementów składowych do elementu macierzystego oraz z powiązanym okresem życia elementów składowych z ich elementami macierzystymi. Elementy składowe mogą być kreowane po utworzeniu elementu macierzystego. Raz utworzone istnieją i są kasowane wraz ze swoim elementem macierzystym. Mogą też być kasowane przed momentem kasowania elementu macierzystego. Kompozycja może być rekursywna.

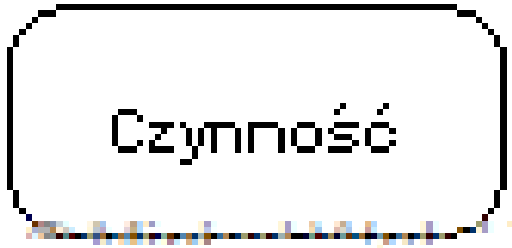
Diagram aktywności i czynności

Modelowanie przepływu czynności jest jedną z najważniejszych technik oferowanych przez język UML. Rozdział ten zawiera opis notacji i semantyki diagramu czynności zwanym także diagramem aktywności.

Diagram czynności (ang. activity diagram) jest diagramem interakcji, który służy do modelowania dynamicznych aspektów systemu. Jego zasadniczą funkcją jest przedstawienie sekwencji kroków, które są wykonywane przez modelowany fragment systemu.

Aktywność

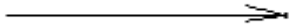
Aktywność (ang. activity) jest sparametryzowanym zachowaniem systemu, przedstawionym w postaci uporządkowanych, podrzędnych elementów, z których najważniejszym jest akcja. Aktywność służy do zaprezentowania tych behawioralnych aspektów systemu, które mogą zostać zdekomponowane za pomocą czynności. Aktywność może stanowić oddzielny diagram czynności.



Czynność

- Czynność (ang. action) jest wykonywalnym węzłem aktywności, który reprezentuje transformację lub proces modelowanego systemu. Wynikiem działania czynności może być zmiana stanu systemu lub zwrócenie wyniku (ang. reference). Czynność nie podlega dekompozycji.

Przepływ sterujący i obiektu



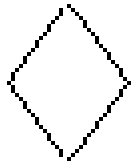
Przepływ sterujący (ang. control flow) to element, który prezentuje przejście pomiędzy węzłami diagramu czynności.

Przepływ obiektu (ang. data flow), zwany także przepływem danych to element, który prezentuje przejście pomiędzy węzłami diagramu widoku interakcji.

Węzeł początkowy i końcowy

Węzeł początkowy (ang. activity initial node) to element rozpoczynający aktywność. Aktywność może mieć tylko jeden węzeł początkowy.

Węzeł końcowy (ang. activity finale node) to element kończący aktywność. Aktywność może mieć więcej niż jedno zakończenie.



Węzeł decyzyjny

Węzeł decyzyjny (ang. decision node) to element, który umożliwia dokonanie wyboru pomiędzy kilkoma możliwościami. Umieszczenie węzła decyzyjnego na diagramie oznacza, że nie ma jednej ścieżki wykonywania poszczególnych aktywności (istnieją ścieżki alternatywne). Węzeł decyzyjny może mieć jedno wejście dla przepływu sterującego i minimalnie dwa wyjścia przepływów. Na wyjściach z węzła decyzyjnego znajdują się wykluczające się warunki dozoru. Istotne jest, by warunki dozoru uwzględniały wszystkie możliwości, tak by nie dopuścić do zatrzymania przepływu na węźle.

Język UML dopuszcza, by jedno wyjście z węzła było opisane warunkiem [else], który oznacza ścieżkę, jaka powinna być wybrana w przypadku niespełnienia warunku dozoru wszystkich pozostałych wyjść.

Często warunek dozoru jest bardzo długi, dlatego też, by zwiększyć czytelność diagramu, można zastosować wyrażenie decyzji, które wraz ze słowem kluczowym <<decisionInput>> zapisane jest w notce

Węzeł połączenia

- Węzeł połączenia (ang. merge node) to węzeł, w którym następuje scalenie kilku alternatywnych przepływów. Węzeł ten nie jest elementem synchronizującym.

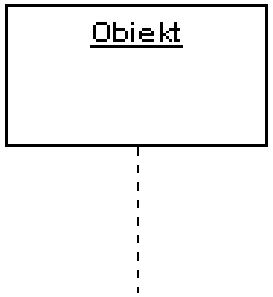
Partycja aktywności

Partycja aktywności (ang. activity partition) jest to część diagramu czynności, która grupuje czynności charakteryzujące się podobnymi cechami. Partycję aktywności można stosować zarówno pionowo, jak i poziomo, co umożliwia grupowanie czynności w wydajniejszy sposób.

W UML 1.X nosiła inna nazwę torów

Diagram sekwencji

Diagram sekwencji (ang. sequence diagram) służy do prezentowania interakcji pomiędzy obiektami wraz z uwzględnieniem w czasie komunikatów, jakie są przesyłane pomiędzy nimi. Na diagramie sekwencji obiekty ułożone są wzdłuż osi X, a komunikaty przesyłane są wzdłuż osi Y. Zasadniczym zastosowaniem diagramów sekwencji jest modelowanie zachowania systemu w kontekście scenariuszy przypadków użycia. Diagramy sekwencji pozwalają uzyskać odpowiedź na pytanie, jak w czasie przebiega komunikacja pomiędzy obiektami. Dodatkowo diagramy sekwencji stanowią podstawową technikę modelowania zachowania systemu, które składa się na realizację przypadku użycia.



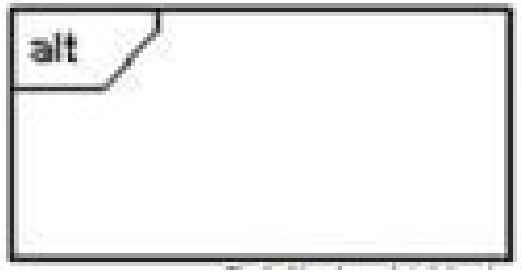
Linia życia

Linia życia to rola uczestnika interakcji, jaką pełni w czasie jej trwania. Linia życia reprezentuje współuczestnika interakcji i czas jego istnienia podczas realizacji scenariusza. Linie życia reprezentują konkretne byty – obiekty, systemy i mogą przyjmować stereotypy, które świadczą o roli, jaką pełni dany obiekt
w systemie. Takimi stereotypami mogą być aktor (ang. actor), obiekt klasy granicznej (ang. boundary class), obiekt klasy sterującej (ang. control class), obiekt klasy danych (ang. entity class).

Komunikat

Komunikat (ang. message) jest to informacja przesyłana pomiędzy obiektami. W języku UML można korzystać z różnych typów komunikatów . Komunikat synchroniczny oznacza, że obiekt musi czekać na odpowiedź. Komunikat asynchroniczny nie wymaga oczekiwania na odpowiedź.

Stosując na diagramie komunikaty synchroniczne, przyjmuje się, że natychmiast po jego wywołaniu przychodzi odpowiedź. Taka konwencja pozwala na zmniejszenie liczby elementów na diagramie. W sytuacji, gdy modelowany fragment rzeczywistości jest inny niż założenia konwencji, należy umieścić komunikat zwrotny.



Fragment

Fragment (ang. combined fragment) to konceptualnie zamknięta część diagramu sekwencji, która rozszerza możliwości obejmowanego przez siebie obszaru diagramu sekwencji. Fragment może zawierać w sobie pętle, powtórzenia, scenariusze alternatywne lub wskazywać poziom abstrakcji modelowanego fragmentu systemu.

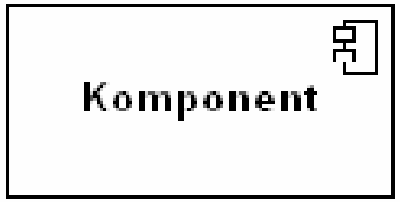
Rodzaj fragmentu jest określany poprzez umieszczenie odpowiedniego słowa kluczowego w lewym górnym rogu. Poniżej opis wszystkich słów kluczowych, które mogą wystąpić we fragmentach. Najpopularniejsze typy to:

- alt – dzieli fragment interakcji zgodnie z warunkami logiki Boole'a na dwa alternatywne scenariusze; każda z alternatyw musi być opatrzona warunkiem dozoru, którego spełnienie gwarantuje wykonanie danej alternatywy.
- loop – powtórzenie fragmentu interakcji określoną warunkiem liczbę razy.
- par – prezentuje równoległe wykonywanie przepływu wiadomości.

Możliwość stosowania fragmentów w diagramach sekwencji pomaga przy modelowaniu scenariuszy.

Diagram komponentów

Diagram komponentów (ang. component diagram) służy do ilustracji organizacji i zależności pomiędzy komponentami. Diagram komponentów prezentuje system na wyższym poziomie abstrakcji niż diagram klas, gdyż każdy z komponentów może być implementacją jednej lub większej liczby klas.Diagramy komponentów służą do określania szczegółów niezbędnych do budowy systemu.



omponent

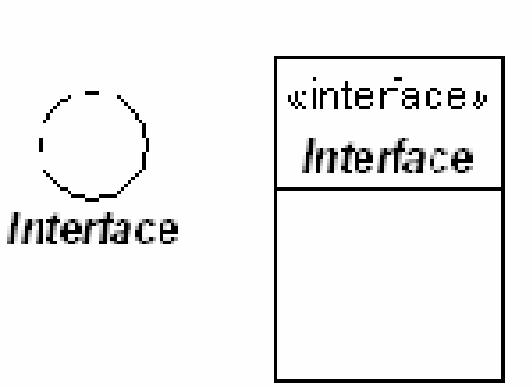
Komponent (ang. component) opisuje modularny – logiczny bądź fizyczny – fragment systemu, który stanowi bardziej zwięzły opis obrazu zachowania systemu niż jego implementacja. Z tego też powodu komponenty mogą być stosowane w dwóch aspektach. Definiują one zewnętrzne oblicze systemu oraz stanowią implementację funkcjonalności systemu.

Komponent

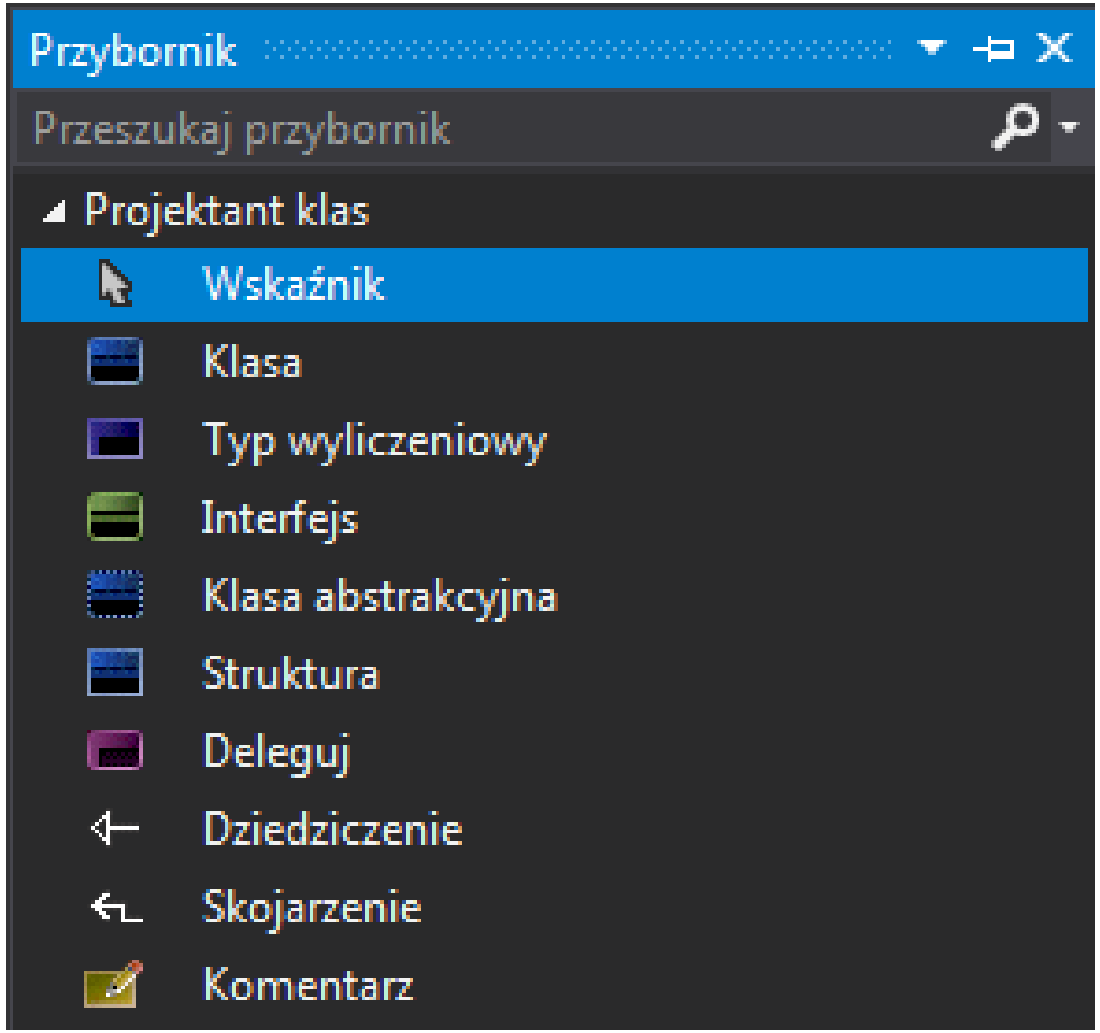
Komponent jest agregatem dla podsystemów we wszystkich częściach/ poziomach systemu. Stosowany ze stereotypem <<subsystem>>, oznacza, że jest agregatem dla komponentów dużej skali.

Interfejs

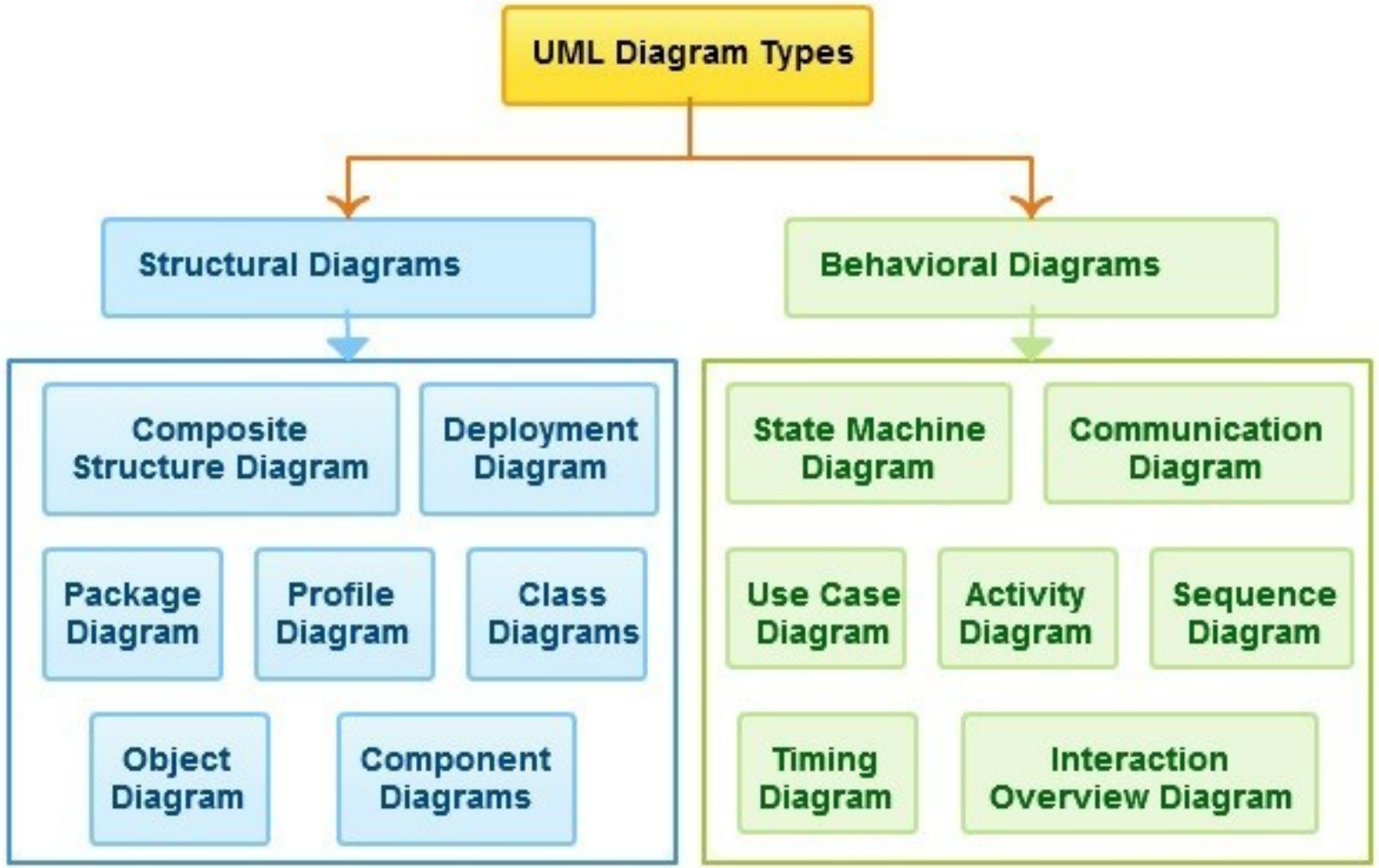
Interfejs (ang. interface) jest to zestaw operacji, które wyznaczają usługi oferowane przez komponent (lub klasę). Interfejsy służą do prezentowania komunikacji pomiędzy komponentami.



Elementy Diagramu Klas



Rodzaje diagramów



Opis rodzaju Diagramu

- diagram klas (ang. Class Diagram) - kls (cld) - diagram klas to graficzne przedstawienie statycznych, deklaratywnych elementów dziedziny przedmiotowej oraz związków między nimi
- diagram obiektów (Object Diagram) - obk (od) - diagram obiektów to wystąpienie diagramu klas, odwzorowujące strukturę systemu w wybranym momencie jego działania
- diagram pakietów (ang. Package Diagram) - pkt (pd) - diagram pakietów to graficzne przedstawienie logicznej struktury systemu w postaci zestawu pakietów połączonych zależnościami i zagnieżdżeniami;

Opis rodzaju Diagramu

- diagram struktur połączonych (ang. Composite Structure Diagram) - spl (csd) - diagram struktur połączonych to graficzne przedstawienie wzajemnie współdziałających części dla osiągnięcia pożądanej funkcjonalności współdziałania;
- diagram komponentów (Component Diagram) - kmp (cod) - diagram komponentów to rodzaj diagramu wdrożeniowego, który wskazuje organizację i zależność między komponentami;
- diagram rozlokowania (ang. Deployment Diagram) - rzk (dd) - diagram rozlokowania to rodzaj diagramu wdrożeniowego, który przedstawia sieć połączonych ścieżkami komunikowania węzłów z ulokowanymi na nich artefaktami;

Opis rodzaju Diagramu

- diagram przypadków użycia (ang. Use Case Diagram) - uzc (ud) - diagram przypadków użycia to graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, występujących w danej dziedzinie przedmiotowej
- diagram czynności (Activity Diagram) - czn (ad) - diagram czynności to graficzne przedstawienie sekwencyjnych i (lub) współbieżnych przepływów sterowania oraz danych pomiędzy uporządkowanymi ciągami czynności, akcji i obiektów;
- diagram maszyny stanowej (ang. State Machine Diagram) - stn (sm) - diagram maszyny stanowej to graficzne odzwierciedlenie dyskretnego, skokowego zachowania skończonych systemów stan-przejście;

Opis rodzaju Diagramu

- diagram sekwencji (ang. Sequence Diagram) - skw (sd) - diagram sekwencji jest rodzajem diagramu interakcji, opisującym interakcje pomiędzy instancjami klasyfikatorów systemu w postaci sekwencji komunikatów wymienianych między nimi;
- diagram komunikacji (Communication Diagram) - kmn (cd) - diagram komunikacji jest rodzajem diagramu interakcji, specyfikującym strukturalne związki pomiędzy instancjami klasyfikatorów biorącymi udział w interakcji oraz wymianę komunikatów pomiędzy instancjami;
- diagram harmonogramowania (ang. Timing Diagram) - hrm (tm) - diagram harmonogramowania jest rodzajem diagramu interakcji, reprezentującym na osi czasu zmiany dopuszczalnych stanów, jakie może przyjmować instancja klasyfikatora uczestnicząca w interakcji;

Opis rodzaju Diagramu

- diagram sterowania interakcją (ang. Interaction Overview Diagram) - sin (iod) - diagram sterowania interakcją jest rodzajem diagramu interakcji, dokumentującym przepływ sterowania pomiędzy logicznie powiązanymi diagramami i fragmentami interakcji z wykorzystaniem kategorii modelowania diagramów czynności.

https://www.math.uni.lodz.pl/~robpleb/wyklad1_3.pdf

http://www.is.umk.pl/~grochu/wiki/doku.php?id=zajecia:npr_2019_2:wyklad:architektura_uml

http://zasoby.open.agh.edu.pl/~10sdczerner/page/diagramy_pakietow_uml.html

https://en.wikipedia.org/wiki/Unified_Modeling_Language

https://www.omg.org/spec/UML/

https://www.michalwolski.pl/diagramy-uml/