Unified Modeling Language (UML)

Damian Łącz, PK, 2021

Czym jest UML?

Służy do modelowania dziedziny problemu (opisywania-modelowania fragmentu istniejącej rzeczywistości – na przykład modelowanie tego, czym zajmuje się jakiś dział w firmie) – w przypadku stosowania go do analizy oraz do modelowania rzeczywistości, która ma dopiero powstać – tworzy się w nim głównie modele systemów informatycznych. UML jest przeważnie używany wraz ze swoją reprezentacją graficzną – jego elementom przypisane są odpowiednie symbole wiązane ze sobą na diagramach.

Elementy UML można podzielić na następujące typy :

- Structural – Strukturalne (Statyczne elementy systemu)

●Behavioral – Behawioralne (Dynamiczne elementy systemu)

●Grouping – Grupujące (Grupujące elementy systemu)

●Annotational – Adnotacje (Informacje wyjaśniające)

Dla wersji 2.2 języka UML wyróżnia się 14 diagramów głównych oraz 3 abstrakcyjne (struktur, zachowań i interakcji). Istnieją niestety pewne niejednoznaczności co do stosowanego polskiego tłumaczenia diagramów, np. ang. timing diagram jest tłumaczony jako diagram czasowy, zależności czasowych, harmonogramowania, uwarunkowań czasowych czy diagram przebiegów czasowych.

Diagramy struktur:

- Klas (najczęściej spotykane, ang. class diagram)
- Obiektów (ang. object diagram)
- Komponentów (ang. component diagram)
- Wdrożenia (ang. deployment diagram)
- --- UML 2.0 ---
- Struktur złożonych (ang. composite structure diagram)
- Pakietów (ang. package diagram)
- --- UML 2.2 ---
- Profili (ang. profile diagram, nowość wprowadzona w UML 2.2)

Diagram klas

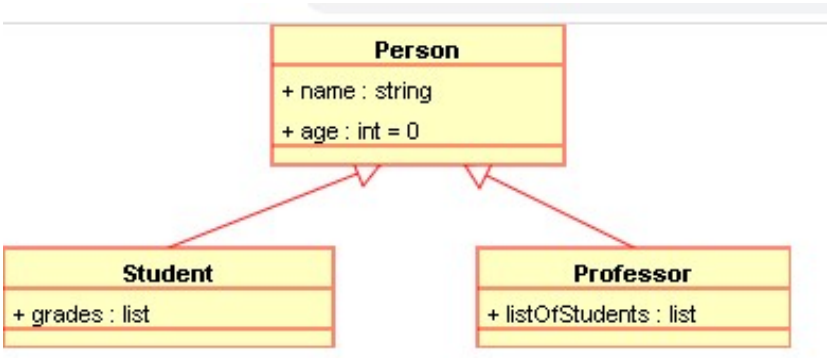
Statyczny diagram strukturalny w UML, przedstawiający strukturę systemu w modelach obiektowych przez ilustrację struktury klas i zależności między nimi.

Diagram klas przedstawia klasy (typy) obiektów w programie, w odróżnieniu od diagramu obiektów, który pokazuje jedynie egzemplarze (instancje) obiektów i ich zależności istniejące w konkretnym momencie.

Diagram klas pokazuje określony fragment struktury systemu. Diagramów klas używa się do modelowania statycznych aspektów perspektywy projektowej. Wiąże się z tym silnie modelowanie słownictwa systemu, kooperacji lub schematów. Diagramy klas pozwalają na sformalizowanie specyfikacji danych i metod. Mogą także pełnić rolę graficznego środka pokazującego szczegóły implementacji klas.

Klasa w modelu UML programu obiektowego jest reprezentowana przez prostokąt z umieszczoną wewnątrz nazwą klasy. Oddzielona część prostokąta pod nazwą klasy może zawierać atrybuty klasy, czyli metody (funkcje), właściwości (properties) lub pola (zmienne). Każdy atrybut pokazywany jest przynajmniej jako nazwa, opcjonalnie także z typem, wartością i innymi cechami.
Metody klasy mogą znajdować się w osobnej części prostokąta. Każda metoda jest pokazywana przynajmniej jako nazwa, a dodatkowo także ze swoimi parametrami i zwracanym typem. Atrybuty (zmienne i właściwości) oraz metody mogą mieć też oznaczoną widoczność (zakres znaczenia ich nazw) jak następuje:

- "+" dla *public* – publiczny, dostęp globalny
- "#" dla *protected* – chroniony, dostęp dla pochodnych klasy (wynikających z generalizacji)
- "−" dla *private* – prywatny, dostępny tylko w obrębie klasy (przy atrybucie statycznym) lub obiektu (przy atrybucie zwykłym)
- "~" dla *package* – pakiet, dostępny w obrębie danego pakietu, projektu.



Związki i powiązania

- **Zależność**

najsłabszy związek znaczeniowy między klasami, gdy jedna z klas używa innej. Na diagramie klas oznaczana przerywaną linią zakończoną strzałką wskazującą kierunek zależności.

- **Asocjacja**

wskazuje na silniejsze powiązanie pomiędzy obiektami danych klas (np. firma zatrudnia pracowników). Na diagramie asocjację oznacza się za pomocą linii, która może być zakończona strzałką z otwartym grotem (oznaczającą kierunek powiązania klas). Nazwy cech (np. zatrudniony, zatrudniający) wraz z krotnością umieszcza się w punkcie docelowym asocjacji. Nazwę asocjacji podaje się pośrodku (np. zatrudnia).

- **Generalizacja**

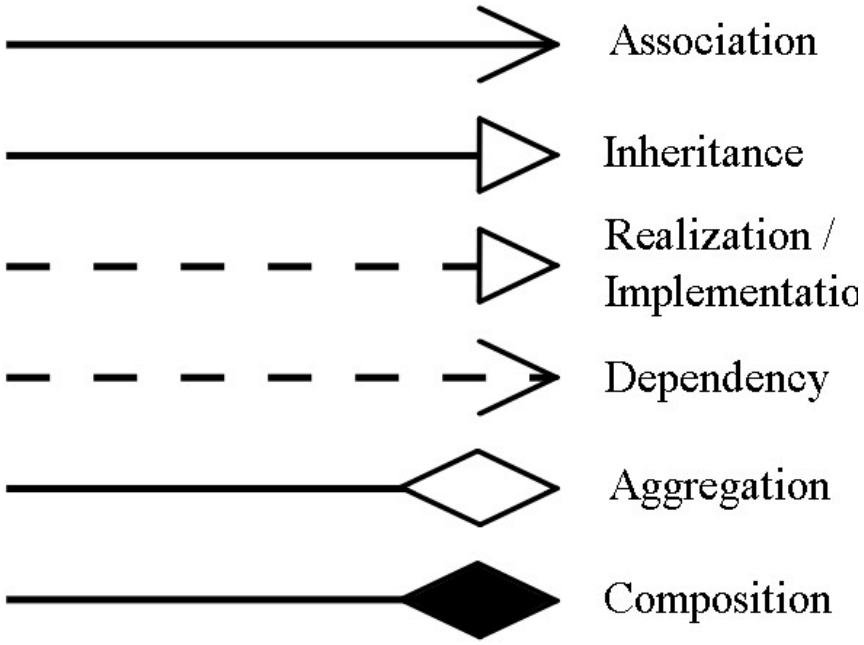
Generalizacja lub dziedziczenie (ang. *generalization* lub *inheritence*) – związek opisujący klasy i podklasy (ogólne klasy i szczegółowe lub inaczej rodziców i dzieci). Na diagramie generalizację oznacza się za pomocą niewypełnionego trójkąta symbolizującego strzałkę (skierowaną od klasy pochodnej do klasy bazowej).

- **Agregacja**

reprezentuje związek typu całość-część, czyli jakaś większa całość jest rozbita na elementy. Oznacza to, że elementy częściowe mogą należeć do większej całości, jednak również mogą istnieć bez niej (np. koła i samochód). Na diagramie agregację oznacza się za pomocą linii zakończonej pustym rombem.

- **Kompozycja**

jest silniejszą formą agregacji. W związku kompozycji, części należą tylko do jednej całości, a ich okres życia jest wspólny — razem z całością niszczone są również części. W dużej mierze jest to kwestia umowna, zależna od danego systemu. Przykładowo silnik samochodowy mógłby istnieć osobno w jednym systemie (zwykła agregacja), a w innym można przyjąć, że jest niszczony razem z samochodem (kompozycja). Na diagramie, kompozycję oznacza się za pomocą linii zakończonej wypełnionym rombem.

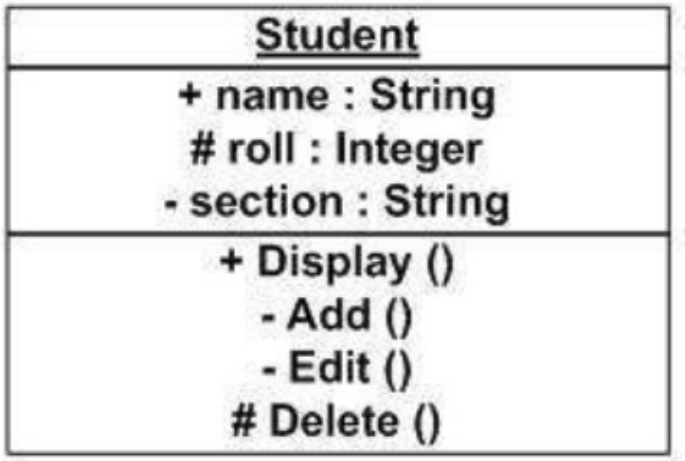


Diagramy obiektów

Diagramy obiektów obrazują obiekty występujące w systemie i ich związki. Są one zazwyczaj wykorzystywane do wyjaśnienia znaczenia diagramów klas. Każdy obiekt może być opisany przy pomocy trzech elementów: tożsamości, stanu i zachowania. Tożsamość jest cechą wyróżniającą obiekt, przedstawiając go jako indywidualną jednostkę. Określa się ją przy pomocny unikatowej i indywidualnej cechy obiektu, która nigdy nie ulega zmianie.

Stan jest zbiorem wszystkich wartości i właściwości obiektu, które każdy z nich posiada i jest przez nie wyróżniany. Obiekt posiada zawsze zestaw właściwości. Ich wartości przez cały okres istnienia obiektu mogą się zmieniać. Zachowanie jest zbiorem usług, które mogą być wykonane przez obiekt na rzecz innych obiektów. Ukazuje dynamikę występującą w modelu, dzięki czemu w łatwy sposób możemy przedstawić relację występujące pomiędzy obiektami lub czynności które dopiero mają zostać wykonane.

Dokładny opis obiektu przy pomocy tych trzech elementów pozwala na dokładną identyfikację oraz na przypisywanie mu dużej ilości informacji. Dzięki temu, jesteśmy w stanie rozróżnić obiekty w dużym stopniu podobne do siebie.



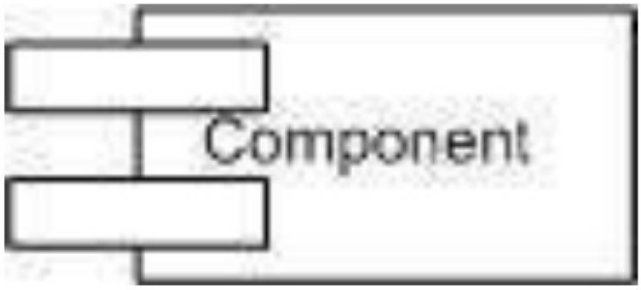
Od klasy różni się nazwą.

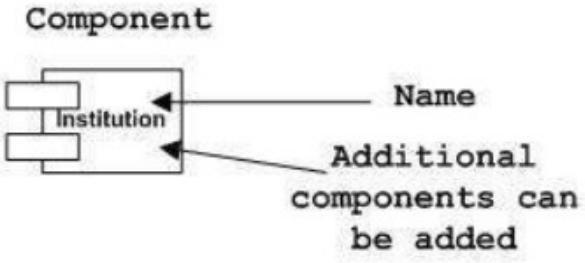
Diagram komponentów

Diagram komponentów (zwany także diagramem implementacji) – diagram przedstawiający jeden z aspektów modelu zgodnego z UML. Przedstawia fizyczne elementy wchodzące w skład systemu i połączenia między nimi.

Elementy

- Komponenty przedstawiane za pomocą dużego prostokąta, z dwoma mniejszymi z jego lewej strony oraz z etykietą w środku.
  Komponenty mogą być przedstawiane zarówno jako klasy jak i instancje. Klasa oznacza elementy systemu istniejące podczas jego działania (np. interfejs użytkownika czy dane). Konkretne instancje precyzują o jaki element chodzi (np. okno programu będące częścią interfejsu).
- Węzły są to zasoby sprzętowe dostępne podczas działania systemu. Obrazowane są za pomocą prostopadłościanów.
- Zależności.





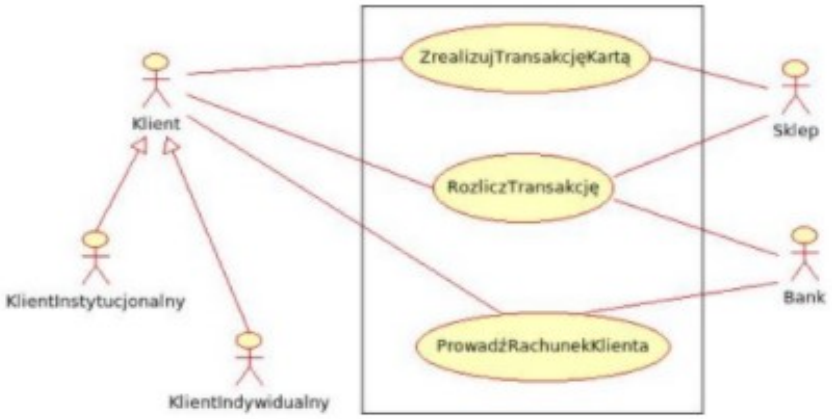
Użycie UML

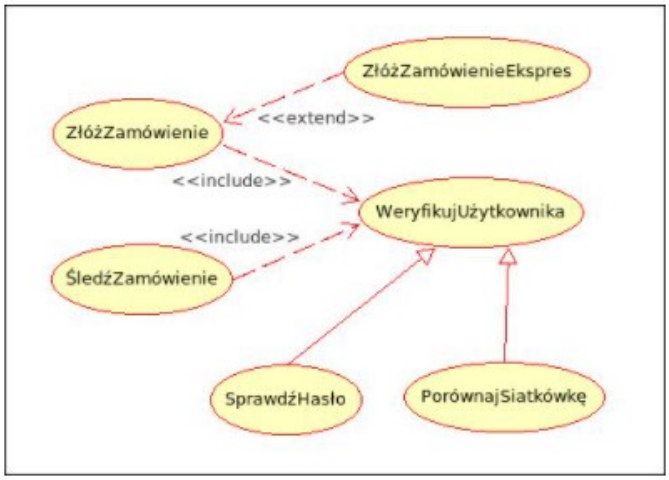
W praktyce rzadko kiedy trzeba opracowywać wszystkie diagramy i w większości przypadków korzysta się z mniej niż połowy wyżej wymienionych. Nie powinno modelować się tylko dla samego modelowania, dlatego nie zawsze wszystkie rodzaje są potrzebne. Projektując system informatyczny, rozpoczyna się przeważnie od tworzenia diagramów w następującej kolejności:

- Przypadków użycia
  - Sekwencji
    - Klas
  - Czynności

Są to najczęściej wykorzystywane diagramy. Pozostałe bywają pomijane, zwłaszcza przy budowaniu niedużych systemów informatycznych.

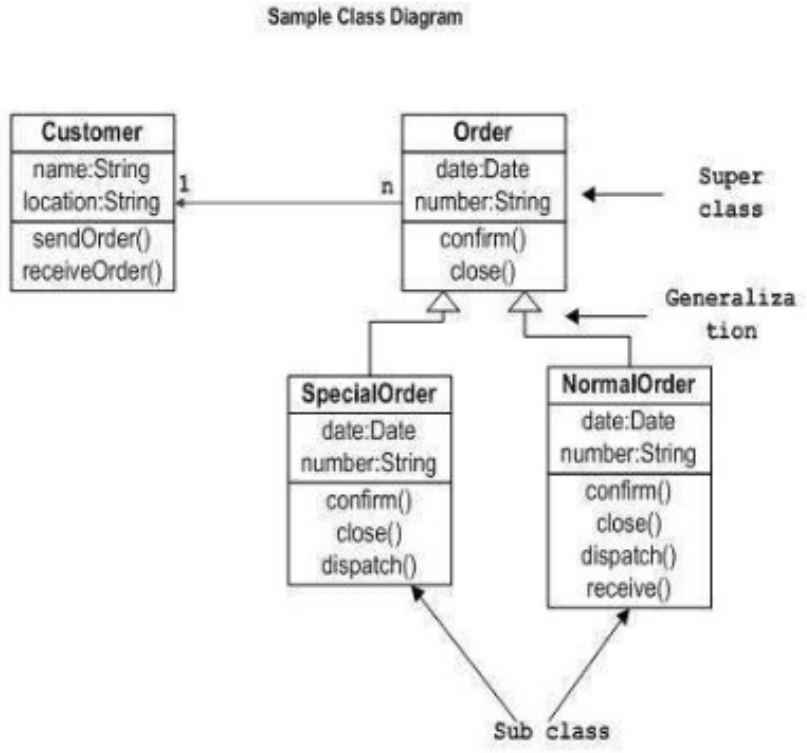
UML jest również stosowany do tworzenia modeli architektury korporacyjnej, jednakże obecnie coraz częściej do tego celu wykorzystywany jest inny standard (rozwijany przez The Open Group), tj. język ArchiMate.





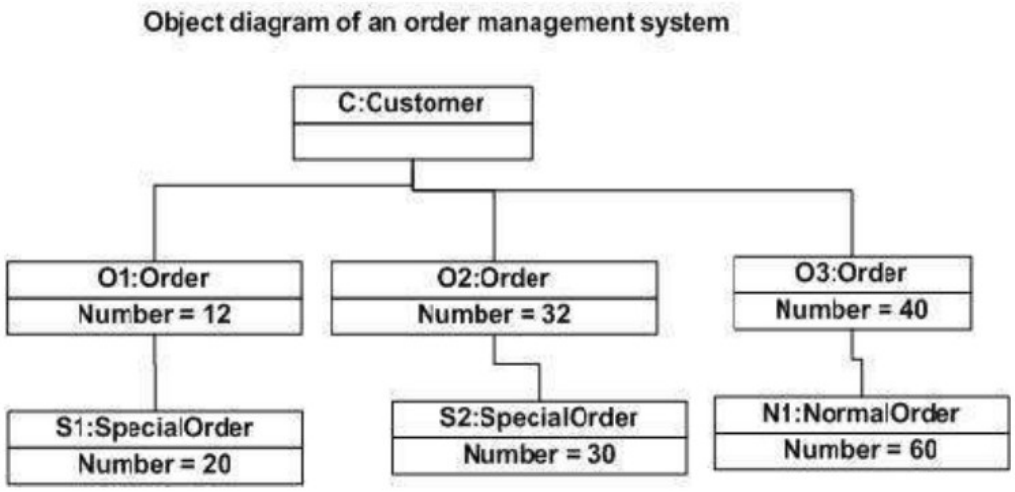
Przykładowe zastosowanie diagramów

- *Diagram klas*



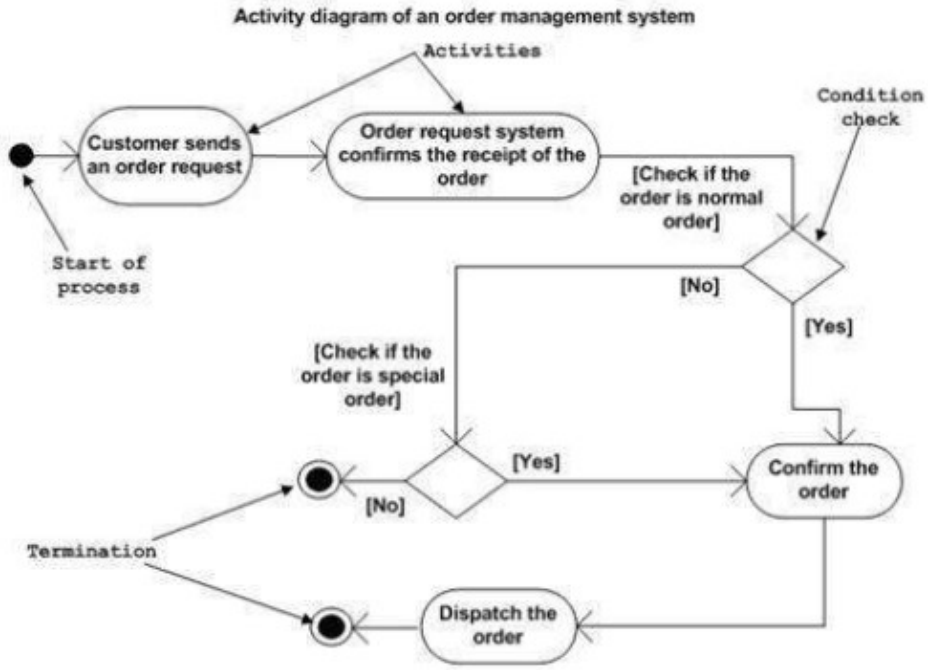
- *- opisanie statycznego widoku systemu*
- *- pokazanie współpracy między elementami widoku statycznego*
- *- opisanie funkcjonalności realizowanych przez system*
- *- budowa aplikacji z wykorzystaniem języków obiektowych*

- *Diagram obiektów*



- konkretny stan – uruchomiony
- wykonanie prototypu systemu
- inżynieria odwrotna
- modelowanie złożonych struktur danych

- *Diagram aktywności*



- modelowanie przeływu pracy za pomocą działań
- modelowanie wymagań biznesowych
-wysoki poziom rozumienia funkcjonalności systemu
-badanie wymagań biznesowych na późniejszym etapie

Bibliografia

- Wikipedia
- http://zasoby.open.agh.edu.pl/~10sdczerner/page/diagramy_przypadkow_uzycia_UML.html