Model-Driven Engineering

Radosław Klimek

2015-22

http://home.agh.edu.pl/rklimek

Przegląd języka UML

Podstawowe pojęcia

Podstawowe pojęcia w języku UML:

1. Bloki konstrukcyjne
2. Mechanizmy

Bloki konstrukcyjne

Bloki konstrukcyjne w języku UML:

1. Elementy.
2. Związki – powiązania miedzy elementami.
3. Diagramy – grupowanie istotnych elementów.

Elementy – rodzaje elementów

Rodzaje elementów w języku UML:

1. Strukturalne.
2. Opisu dynamiki (ang. *behavioural things*).
3. Grupujące.
4. Komentujące.

Elementy – strukturalne

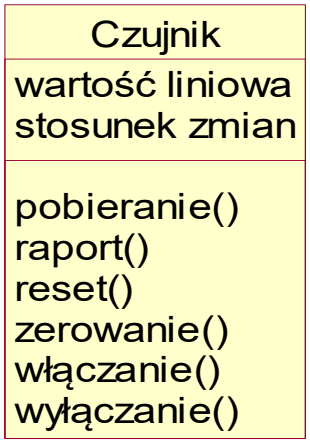
Elementy strukturalne reprezentują składniki fizyczne lub pojęciowe. W modelach są one wyrażone rzeczownikami.

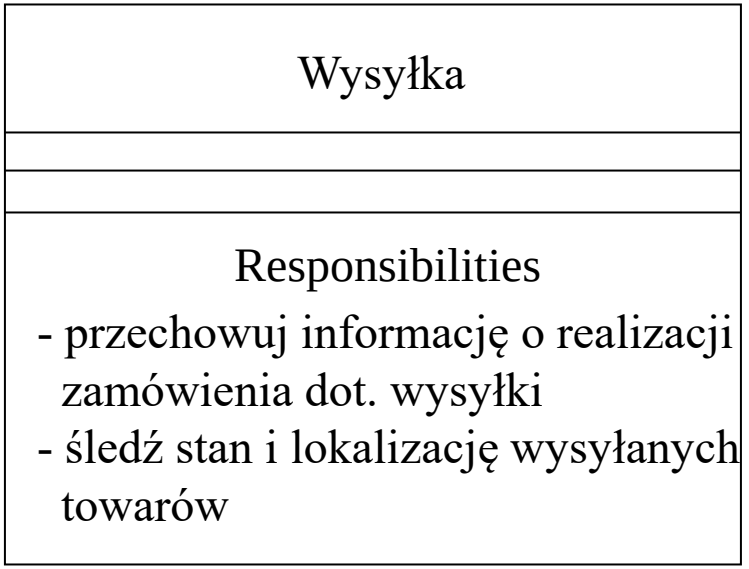
Wyróżnia się 7 rodzajów elementów:

1. Klasy (*Classes*).
2. Interfejsy (*Interfaces*).
3. Kooperacje (*Collaborations*).
4. Przypadki użycia (*Use cases*).
5. Klasy aktywne (*Active classes*).
6. Komponenty (*Components*).
7. Węzły (*Nodes*).

Elementy – strukturalne – klasy/obiekty

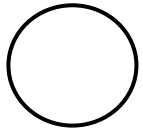
Klasa (ang. class) jest opisem (specyfikacją) zbioru obiektów mających takie same atrybuty, operacje, związki i znaczenie.
Obiekt (ang. object) jest instancją klasy.





Elementy – strukturalne – interfejsy

Interfejs (ang. interface) określa zestaw operacji wyznaczających usługi oferowane przez klasę lub komponent. Interfejs określa zewnętrznie obserwowalne zachowanie elementu. Jest to zbiór deklaracji (sygnatur) tych obserwowalnych (dostępnych) operacji.



IntDostawaTowaru

Elementy – strukturalne – kooperacje

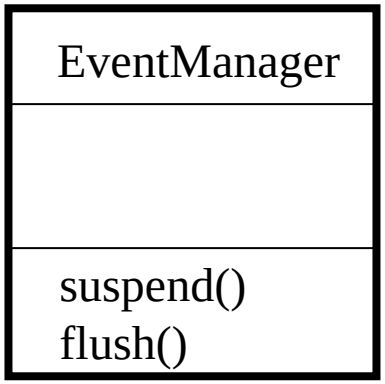
Kooperacja (ang. collaborations) określa interakcję, czyli zestaw ról i innych bytów współdziałających w celu wywołania pewnego zespołowego zachowania, niemożliwego do osiągnięcia w pojedynkę. Pojedyncza klasa może brać udział w wielu kooperacjach. Kooperacje reprezentują implementacje wzorców składających się na system.

Elementy – strukturalne – przypadki użycia

Przypadek użycia (ang. *use case*) jest opisem zbioru ciągów akcji wykonywanych przez system w celu dostarczenia aktorowi obserwowalnego wyniku.

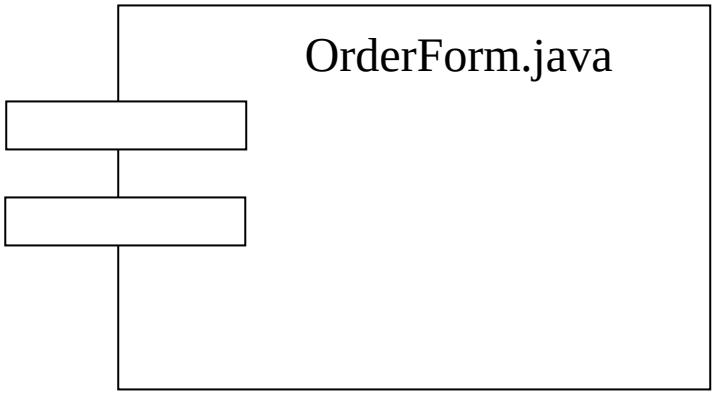
Elementy – strukturalne – klasy aktywne

Klasa aktywna (ang. active classes) zawiera obiekty, przy czym w skład każdego z nichwchodzi przynajmniej jeden proces lub wątek. Oznacza, że dowolny obiekt klasy aktywnej może samodzielnie rozpocząć działanie.



Elementy – strukturalne – komponenty

Komponent (ang. components) jest fizyczną, wymienną częścią systemu, która dostarcza i implementuje pewien zbiór interfejsów. Komponent jest zatem fizycznym „opakowaniem” elementów logicznych, takich jak klasy, interfejsy, kooperacje.



Elementy – strukturalne – węzły

Węzeł (ang. node) jest fizycznym składnikiem działającego systemu.
Reprezentuje zasoby systemowe, ma zazwyczaj pamięć i zdolność przetwarzania.

Zbiór komponentów może znajdować się w węźle, może również przemieszczać się między węzłami.

Elementy – opisu dynamiki

Elementy opisu dynamiki modelują zachowanie w czasie i przestrzeni. Są zazwyczaj wyrażone czasownikami w opisie tekstowym.

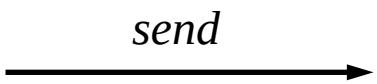
Wyróżnia się dwa podstawowe elementy:

1. Interakcje (ang. *interactions*).
2. Maszyny stanowe (ang. *state machines*).

Elementy – opisu dynamiki – interakcje

Interakcja (ang. *interaction*) jest działaniem (akcją) wymiany komunikatów między obiektami w pewnym otoczeniu i w określonym celu.

Można w ten sposób zdefiniować zachowanie zespołu obiektów, jak również pojedynczą operację. Interakcja składa się z wielu elementów, w tym komunikatów, ciągów akcji (w odpowiedzi na komunikat) i połączeń między obiektami.



komunikat z operacją (*send*)

Elementy – opisu dynamiki – maszyny stanowe

Maszyna stanowa (ang. *state machine*) określa ciąg stanów, jakie obiekt przyjmuje w odpowiedzi na zdarzenia zachodzące w jego otoczeniu. Maszyna stanowa składa się ze **stanów** (*states*), **przejść** (*transitions*) między stanami, **zdarzeń** (*events*), wyzwalających przejścia i **czynności** (*activities*) w odpowiedzi na zdarzenia.

Z wykorzystaniem maszyny stanowej można zdefiniować zachowanie pojedynczej klasy lub kooperacji.

Pojedynczy stan

Elementy – grupujące

Elementy grupujące odgrywają role organizacyjną. Są to bloki, na które możemy rozłożyć (zdekomponować) dany model. Podstawowym rodzajem jest **pakiet**.

Pakiet służy do grupowania elementów i może zawierać elementy strukturalne, dynamiczne i inne grupujące.

W odróżnieniu od komponentu (który istnieje w trakcie wykonywania programu), pakiet jest bytem pojęciowym, tzn. ma znaczenie jedynie w trakcie tworzenia oprogramowania.

Pakiet

Elementy – komentujące

Elementy komentujące (ang. *annotational things*) odgrywają w UML role objaśniającą. Podstawowym elementem jest **notatka** (*note*).

Notatka jest to element umożliwiający specyfikowanie dodatkowych ograniczeń i objaśnień skojarzonych z pojedynczym bytem lub grupą bytów.

Notatka

Związki

Związki służą w UML do łączenia elementów, są zatem podstawowymi elementami konstrukcyjnymi w budowaniu modeli.

Wyróżnia się cztery rodzaje związków:

1. Zależność (*Dependency*).
2. Powiązanie (*Association*).
3. Uogólnienie (*Generalization*).
4. Realizacja (*Realization*).

Wymieniono cztery podstawowe rodzaje związków. W UML są jeszcze inne warianty wspomnianych czterech: uściślenie (*refinement*), ślad (*trace*), zawieranie (*include*), i rozszerzenie (*extend*).

Związki – zależność

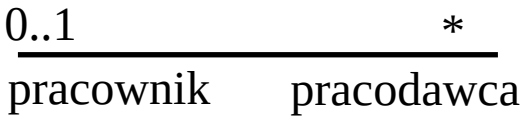
Zależność (ang. *dependency*) jest związkiem między dwoma elementami, gdzie zmiany w definicji jednego z nich (niezależnego) mogą mieć wpływ na znaczenie drugiego (zależnego).

Zależność

Związki – powiązanie

Powiązanie (ang. *association*) jest związkiem strukturalnym, określającym zbiór wiązań między obiektami.

Szczególnym przypadkiem powiązania jest agregacja (składanie), która oznacza relację między całością a częściami.

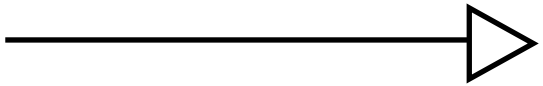


Powiązanie

Związki – uogólnienie

Uogólnienie (*Generalization*) jest związkiem miedzy bytem ogólnym (przodek) a szczegółowym (potomek).

Jest to związek uogólnienie-uszczegółowienie. Obiekt bytu szczegółowego może być używany w zastępstwie ogólnego. W takim przypadku potomek ma strukturę i zachowanie przodka.

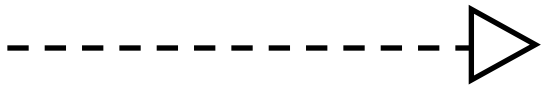


Uogólnienie

Związki – realizacja

Realizacja (ang. *realization*) jest to związek między klasyfikatorami, z których jeden określa kontrakt (specyfikację), natomiast drugi zapewnia wywiązanie się (realizację) kontraktu.

Związki tego typu (realizacje) występują najczęściej między interfejsami a klasami lub komponentami oraz między przypadkami użycia a kooperacjami.



Realizacja

Diagramy

• Diagram w języku UML jest grafem, gdzie najczęściej wierzchołkami są elementy, a krawędziami związki.
• Diagram jest pewnego rodzaju rzutem (projekcją) systemu. Pojedynczy diagram daje zatem niepełną informację dotyczącą elementów (bytów) tworzących system.
• Ten sam byt może występować we wszystkich diagramach. Poszczególne typy diagramów opisują oddzielne perspektywy (punkty widzenia) systemu (podsystemów).

W języku UML wyróżnia się 9 rodzajów diagramów:

1. Diagram klas (*class diagram*).
2. Diagram obiektów (*object diagram*).
3. Diagram przypadków użycia (*use case diagram*).
4. Diagram sekwencji (przebiegu) (*sequence diagram*).
5. Diagram kooperacji (*collaboration diagrams*).
6. Diagram stanów (*statechart diagram*).
7. Diagram czynności (*activity diagram*).
8. Diagram komponentów (*component diagram*).
9. Diagram wdrożenia (implementacji) (*deployment diagram*).

Diagramy – diagram klas

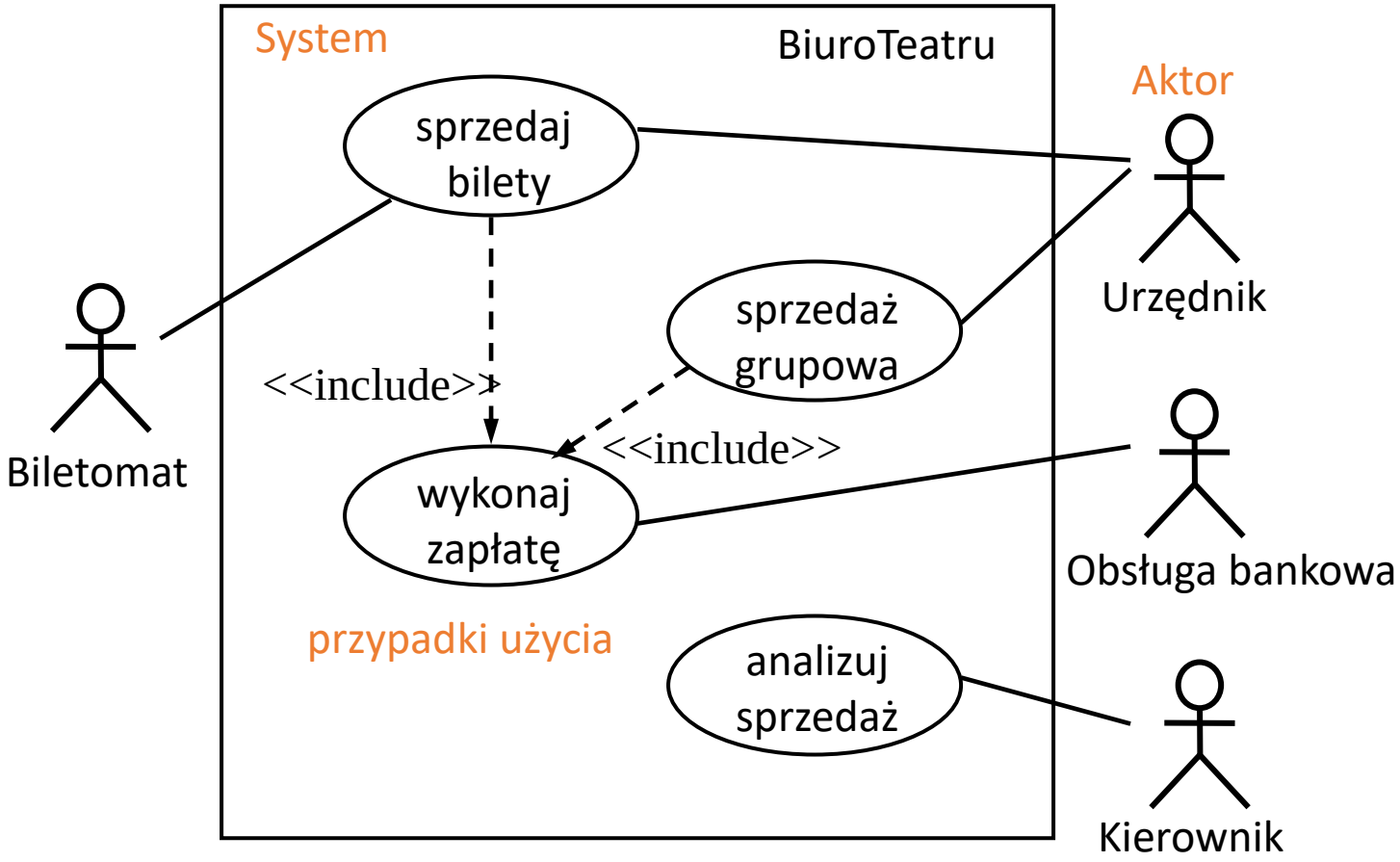
W **diagramie klas** mogą znaleźć się klasy, interfejsy, kooperacje i związki miedzy nimi. Diagram klas odnosi się do statycznych aspektów perspektywy projektowej. Diagram klas uwzględniający klasy aktywne dotyczy statycznych aspektów perspektywy procesowej.

Diagramy – diagram obiektów

Diagram obiektów przedstawia obiekty i związki między nimi. Diagram przedstawia statyczny rzut pewnych elementów występujących na diagramie klas. Bierze się jednak pod uwagę przypadki rzeczywiste lub prototypowe.

Diagramy – diagram przypadków użycia

Diagram przypadków użycia przedstawia przypadki użycia, aktorów i związki między nimi. Diagram odnosi się do statycznych aspektów przypadków użycia.



Diagramy – diagram przebiegu

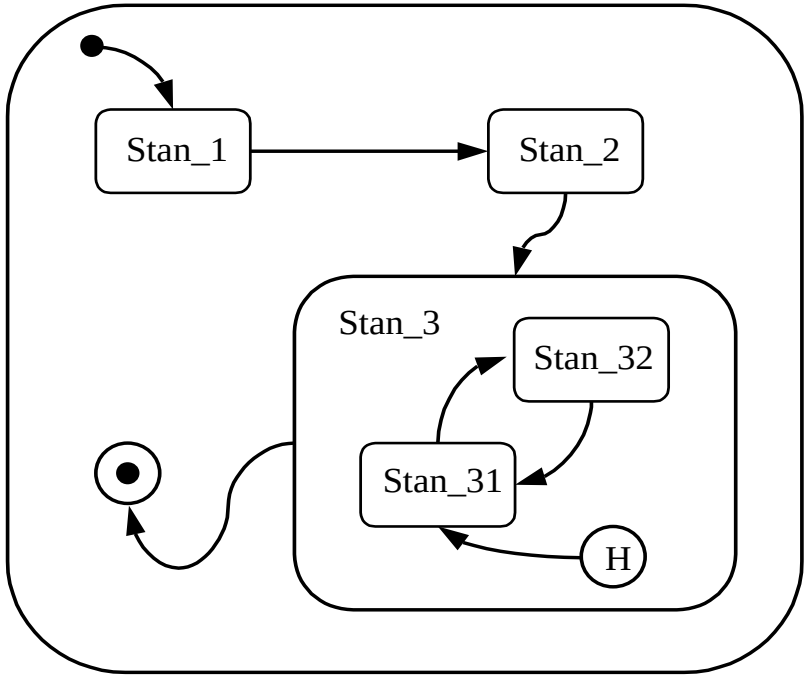
Diagram przebiegu (**sekwencji**) opisuje interakcję. Diagram sekwencji pokazuje kolejność przesyłania komunikatów w czasie.

Diagramy – diagram kooperacji

Diagram kooperacji opisują interakcję. W diagramie kooperacji kładzie się nacisk na organizację strukturalna obiektów wymieniających komunikaty. Diagramy sekwencji oraz przebiegu są „równoważne”, tzn. jeden można przekształcać w drugi.

Diagramy – diagram stanów

Diagram stanów (ang. *statechart diagram*) przedstawia maszynę stanową składającą się ze stanów, przejść, zdarzeń i czynności. Przedstawia reakcje obiektów na ciągi zdarzeń, nadaje się zatem do modelowania systemów interakcyjnych.



Składnia zdarzenia:

Nazwa [dozór] / akcje ^ zdarzenia wysyłane

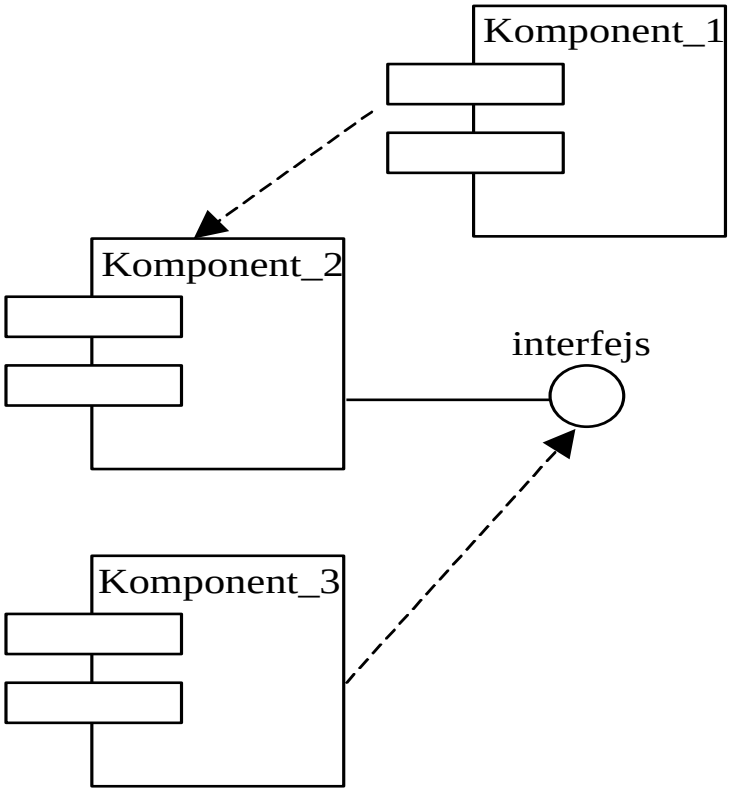
- Nazwa: Zdarzenie komunikat
- [dozór]: Przejście jeśli *true*
- akcje: Wykonanie w trakcie przejścia
- zdarzenia wysyłane: Wysłanie do innych maszyn w trakcie przejścia

Diagramy – diagram czynności

Diagram czynności może być traktowany jako pewna wersja diagramu stanów. Odnosi się do modelowania dynamicznych aspektów systemu. Kładzie się tu też nacisk na przepływ sterowania między obiektami.

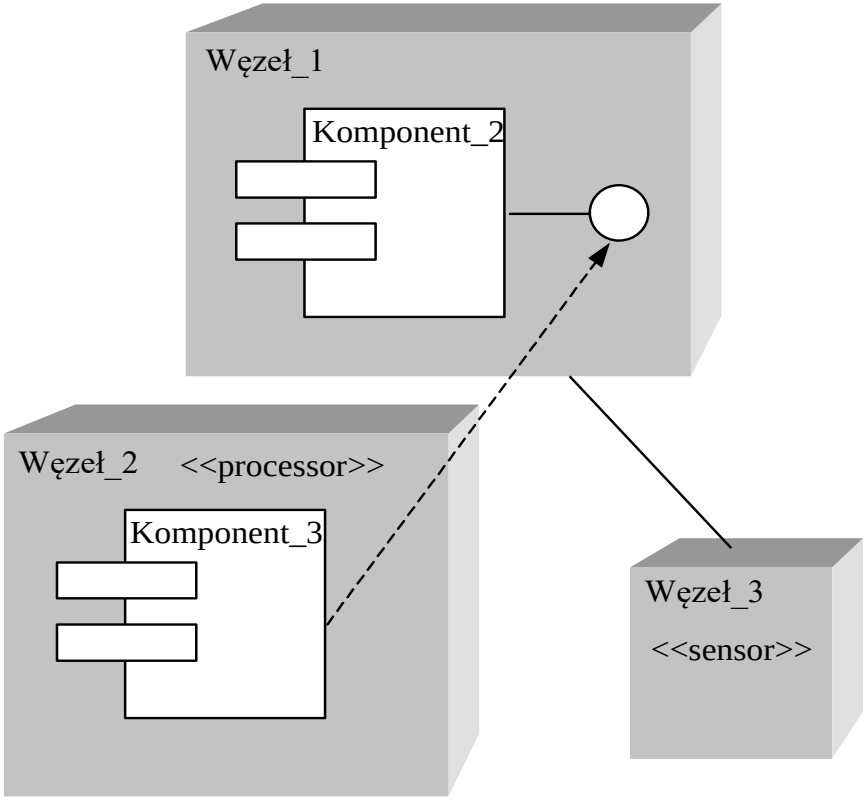
Diagramy – diagram komponentów

Diagram komponentów pokazuje uporządkowanie komponentów i zależności między nimi. Odnosi się do statycznych aspektów perspektywy implementacyjnej. Wiąże się ściśle z diagramem klas, gdyż każdemu komponentowi są przypisane pewne klasy, interfejsy i kooperacje.



Diagramy – diagram wdrożenia/implementacji

Diagram wdrożenia opisuje konfigurację poszczególnych węzłów działających w czasie wykonania i zainstalowania na nich komponentów. Odnosi się do statycznych aspektów perspektywy wdrożeniowej. Wiąże się z diagramem komponentów, ponieważ każdy węzeł zawiera co najmniej jeden komponent.



Mechanizmy

Podstawowe mechanizmy języka UML

1. Specyfikacje (*specifications*).
2. Dodatki (*adornments*).
3. Zasadnicze rozgraniczenia (*common divisions*).
4. Mechanizmy rozszerzania (*extensibility mechanizms*).

Specyfikacje

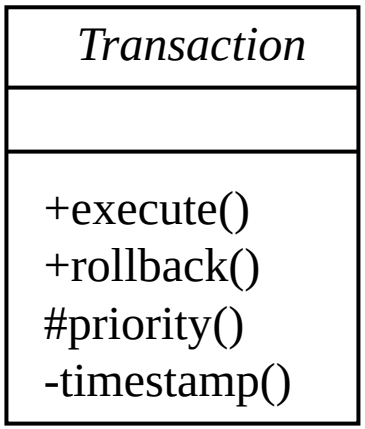
Specyfikacje są znaczeniową platformą UML, która zawiera wszystkie części modeli odpowiednio powiązane ze sobą logicznie.

Przyrostowe tworzenie systemu polega na tworzeniu diagramów i dodawaniu szczegółów (specyfikacji).

Inżynieria wstecz polega na utworzeniu specyfikacji, a następnie stworzeniu odpowiednich diagramów.

Dodatki

Symbol klasy zawiera nazwę, atrybuty i operacje. Dodatkowe szczegóły można podawać jako tekstowe uzupełnienia podstawowej reprezentacji klasy.
Każdy element w notacji UML składa się zatem z symbolu podstawowego i charakterystycznych dla niego dodatków.



+operacja publiczna

-operacja prywatna

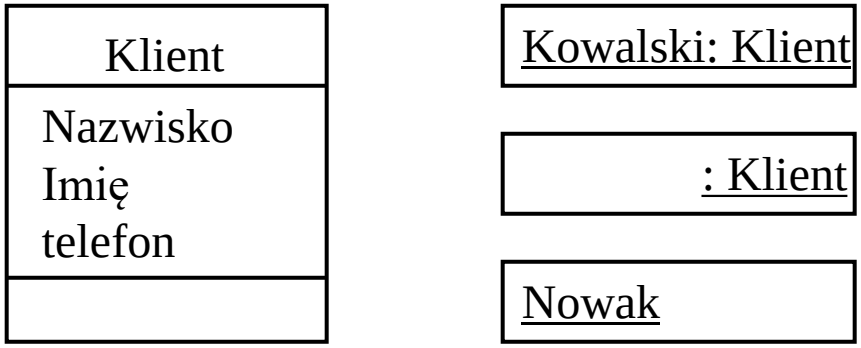
operacja chroniona

Dodatki

Zasadnicze rozgraniczenia

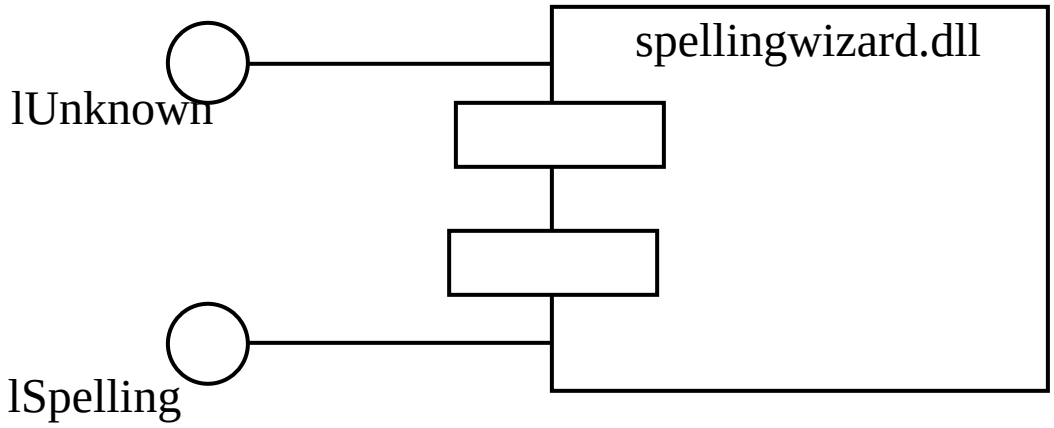
Z każdym blokiem konstrukcyjnym jest związane rozróżnienie: abstrakcja (klasa) - urzeczywistnienie abstrakcji (obiekt).

Podobnie dla przypadków użycia i egzemplarzy przypadków użycia, komponentów, węzłów.



Zasadnicze rozgraniczenia (cd)

Podobnie z każdym blokiem konstrukcyjnym można skojarzyć dychotomię typu interfejs-implementacja, gdzie interfejs wyraża deklarację kontraktu (sygnaturę implementacji), natomiast implementacja jest realizacja tego kontraktu.
Możemy zatem mówić o operacjach i metodach, które je realizują, jak również przypadkach użycia i realizujących je kooperacjach.



Interfejsy i ich implementacja przez komponent *spellingwizard.dll*

Mechanizmy rozszerzenia

Wyróżnia się 3 mechanizmy rozszerzania (ang. *extensibility mechanizm*):

1. Stereotypy (*stereotypes*).
2. Metki (*tagged values*).
3. Ograniczenia *(constraints*).

UML jest językiem otwartym, jednak rozszerzenia są realizowane kontrolowany sposób.

Mechanizmy rozszerzenia – stereotyp

Stereotyp umożliwia rozszerzanie słownictwa UML. Można zatem tworzyć nowe bloki wywodzące się z istniejących, lecz specyficzne dla danego zadania.

Przykładem mogą być wyjątki, które wprost nie istnieją jako wyróżnione bloki konstrukcyjne. Będą one traktowane jako standardowe bloki, jeśli będą oznaczone jako stereotypy.

Mechanizmy rozszerzenia – metka i ograniczenie

Metka umożliwia rozszerzenie listy własności bloku konstrukcyjnego, np. podanie wersji i autora. Informacje te nie elementarnymi konstrukcjami języka.

Ograniczenie umożliwia dodanie reguł lub modyfikację reguł istniejących.