Bohatý Martin, 760227/0170

UML
Unified Modeling Language

Úvod

Cílem této práce je seznámit čtenáře se základy a principy jazyka UML. Podat vyčerpávající popis všeho co UML nabízí není v práci tohoto rozsahu možné, proto se soustředí hlavně na oblasti Class diagramů a Use Case diagramů, které jsou dle názoru autora nejdůležitější z hlediska návrhu a vývoje informačních systémů. Závěr je pak věnován implementaci UML v modelovacích nástrojích.

Co je UML?

UML (Unified Modeling Language) vychází z potřeb softwarových návrhářů, analytiků a vývojářů na jednotnou metodiku pro vytváření a užívání modelů informačních systémů. Nepopiratelnou úlohu při nárůstu těchto potřeb zapříčinil i velký rozmach objektových přístupů a objektově orientovaného programování v první polovině 90. let. První komplexní verze UML vznikla roku 1997 po předchozích pokusech o vytvoření jednotného modelovacího jazyka (zhruba od roku 1995) a vychází z řady starších standardů (Booch, OMT a další).

UML používá pro znázornění chování, struktury a složení informačních systémů (případně jejich částí – aplikací, komponent) grafické vyjádření prostřednictvím diagramů, které vycházejí z modelů několika základních typů. Hlavní předností UML je pak skutečnost, že nejen zahrnuje moderní trendy, ale především poskytuje komplexní pohled na celý životní cyklus objektově orientovaného vývoje.

Diagramy obecně

Různé druhy diagramů charakterizují určité oblasti. Diagramy jsou navíc mezi sebou provázány (lépe řečeno pracují se stejným modelem respektive jeho částí), tak že některé jejich prvky navzájem sdílejí určité informace.

Ve standardu UML je definováno celkem osm (potažmo devět) diagramů. Jsou to:

- **Activity Diagram** – znázorňuje chování operace jako souslednost určitých akcí
- **Class Diagram** – reprezentuje statickou strukturu prostřednictvím tříd a jejich vztahů
- **Component Diagram** – znázorňuje komponenty aplikace (soubory, knihovny atd.); komponenta je zde chápána jako tělo třídy respektive její implementace
- **Deployment Diagram** – znázorňuje určité hardwarové prvky a jejich vazby; jsou definovány dva základní typy zařízení - processors, tedy takové hardwarové zařízení, které je schopné spouštět programy a procesy, a device, které tuto schopnost nemají
- **Interaction Diagram** – zahrnuje Collaboration Diagram (prostorové znázornění objektů, vazeb a jejich vzájemných interakcí, v podstatě je přímým rozšířením Object diagramů) a Statechart Diagram (znázorňuje chování třídy definováním určitých stavů)
- **Object Diagram** – reprezentuje dynamickou strukturu prostřednictvím objektů (instancí) a jejich vztahů
- **Sequence Diagram** – znázorňuje objekty a jejich interakce; na rozdíl od Object diagramů se ale detailněji zaměřuje na popis interakcí
- **Use Case Diagram** – znázorňuje funkce systému z pohledu uživatele

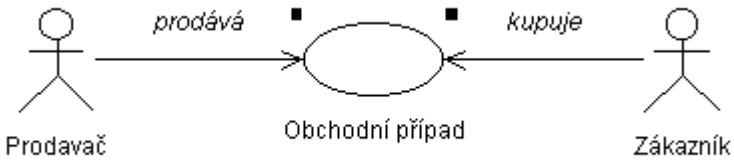
Use Case Diagram

Use Case diagramy znázorňují chování systému z pohledu uživatele. Umožňují definovat ohraničený systém a jeho vztahy z vnějším okolím. Podávají v podstatě obraz funkčnosti systému, která je vyvolávána podněty z vnějšku.

Use Case diagramy zahrnují aktéry (actors), systém a případy použití (use case). Aktér reprezentuje činnost osob nebo jiný prvek ovlivňující chování systému. V žádném případě neplatí, že každá fyzická osoba je reprezentována jedním právě aktérem. Ve skutečnosti totiž může každá osoba vystupovat v různých rolích a tudíž může být reprezentována více aktéry a naopak.

Existují čtyři základní kategorie aktérů:

Principal actors	- lidé využívající hlavní funkce systému.
Secondary actors	- lidé, kteří se starají o administraci a zprávu systému
External hardware	- další periferní zařízení, se kterými systém pracuje (např. čtečka elektronických platebních karet)
Other systems	- ostatní systémy, se kterými musí daný systém spolupracovat



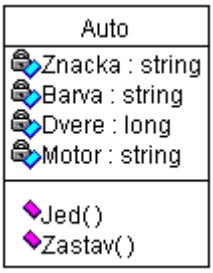
Obr. 1 – Use Case diagram
Obrázek znázorňuje jednoduchý Use Case diagram ve kterém vystupují dva aktéři – prodávající a kupující. Oba mají vazbu na obchodní případ.

Class Diagram

Class diagramy reprezentují statický pohled na třídy a vztahy mezi nimi.

Class diagramy a Object diagramy jsou v podstatě alternativními pohledy na objektový model jako takový. V praxi se ve většině případů dává přednost Class diagramům. Objektové diagramy, které jsou na rozdíl od Class diagramů dynamické (znázorňují jednotlivé instance daných tříd), se používají jen výjimečně pro ilustraci objektů s komplikovanou datovou strukturou.

Hlavním prvkem Class diagrmů jsou třídy reprezentované obdélníkovou ikonou rozdělenou do tří částí. V horní částí je název třídy, v prostřední jsou její atributy a v dolní pak její metody.



Obr. 2 - Třída
Obrázek znázorňuje třídu pojmenovanou Auto, která má atributy Znacka, Barva, Dvere, Motor a metody Jed() a Zastav ().

Atributy jsou definovány jako:

AtrributeName : AttributeType = Initial Value

Metody jsou definovány jako:

MethodName (ArgumentName : ArgumentType = DefaultValue, ...) : ReturnValue

V praxi se většinou vytváří více Class diagramů tak, aby reprezentovaly hierarchickou strukturu tříd od shora dolů, nebo aby znázorňovali obsah jednotlivých balíků (package).

UML navíc umožňuje popisovat viditelnost atributů a metod. Standardně jsou definovány tyto možnosti:

- **public** – atribut nebo metoda je přístupná pro všechny
- **protected** – atribut nebo metoda je přístupný pouze v této třídě a jejích podtřídách
- **private** – atribut nebo metoda je přístupný pouze pro tuto třídu

Další možností jak detailněji specifikovat třídu je použití stereotypů – Signal, Interface, MetaClass a Utility. Asi nejpužívanější je Interface, který popisuje pouze viditelné metody a atributy třídy jiné.

Asociace

Asociace reprezentují vztahy mezi dvěma (potažmo více) třídami. V diagramech se znázorňují rovnou, případně lomenou čarou, která spojuje ikony tříd. UML umožňuje pojmenovat jednotlivé asociace tak, aby vystihovali jejich funkci.

Počet objektů jedné třídy, které mohou být propojeny s objektem třídy druhé je určen násobností vazeb.

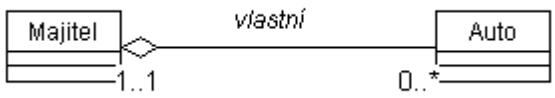
Zakončení čar symbolizujících asociace se nazývá role. Role může být rovněž pojmenována, ale hlavně se využívá k určení násobnosti vazeb. Ta může být následující:

1	Právě jeden
0..1	Žádný nebo jeden
M..N	Od M do N (kladné celé číslo)
*	Žádný nebo jeden a více
1..*	Jeden a více

Agregace

Agregace reprezentuje asociaci, kde objekt na jedné straně hraje daleko významnější roly než objekt na straně druhé. Typickými příklady je třída, která obsahuje třídu druhou nebo třída, jejíž metoda vyvolává metody druhé třídy. Stejně jako asociace může být i agregace násobná.

Agregace je graficky znázorněna čarou, která navíc obsahuje malý kosočtverec na straně "významnější" třídy.

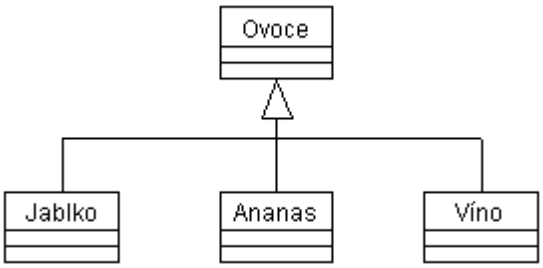


Obr. 3 - Agregace
Objekt Majitel může vlastnit libovolný počet objektů Auto nebo žádný.

Generalizace

Generalizace odráží skutečnost, že atributy a metody jedné třídy jsou zároveň atributy a metodami třídy jiné. Atributy, metody a vztahy nadřízené třídy jsou plně zděděny ve třídě podřízené. V programování je generalizace nejčastěji interpretována jako dědičnost.

Graficky je generalizace znázorněna jako čára zakončená šipkou na straně nadřízené třídy.



Obr. 4 – Generalizace
Třída Ovoce je nadřízená třídám Jablko, Ananas a Víno. Ty mají stejné metody a atributy jako rodičovská třída. Navíc mohou obsahovat metody a atributy vlastní.

Implementace UML

UML je více či méně implementováno v celé řadě modelovacích nástrojů. Nejznámějším je bezesporu Rational Rose Modeler, potažmo jeho varianta Visual Modeler distribuovaná spolu s produktem Microsoft Developer Studio. Nástroje tohoto typu umožňují kromě tvorby diagramů založených na standardech UML i celou řadu dalších efektivních funkcí. Je to zejména možnost generovat zdrojové kódy (definice tříd, metod, atributů atd.) přímo z modelovacích nástrojů na základě příslušných diagramů a to do celé řady programovacích jazyků (C++, Visual Basic, Java).

Neméně zajímavou možností je i využití technologie Reverse Engeneering, s jejíž pomocí je možné naopak získat diagramy ze zdrojových kódů.

Kritika UML

Přestože je UML velmi cenným pomocníkem při návrhu informačních systémů, jednotlivých aplikací a komponent, nelze se ubránit i troše kritiky. Nejnepříjemnější je asi skutečnost, že UML zatím neobsahuje standardy pro modelování datové základny, což v praxi bohužel znamená použití minimálně dvou modelovacích nástrojů. I když i zde se pravděpodobně blýská na lepší časy, neboť podle nejnovějších zpráv bude tento prvek začleněn do nové verze UML, a tak je jen otázkou času, než se objeví jeho implementace i v některém z modelovacích nástrojů.