Unified Modeling language (UML), popis jazyka, typy diagramů a způsob jejich použití při návrhu různých aspektů systému, syntax a sémantika jazyka a jeho symbolů.

Co je UML (Unified modeling language)

- grafický modelovací jazyk pro vytváření visuálních (grafických) **modelů**, **specifikaci**, **vizualizaci**, **konstrukci** a **dokumentaci** při OO **analýze** a **návrhu** (OOAD)
- pro **modelování organizace** (business modelling)
- není modelovací metodika – neříká nic o navrhování SW
- UML je otevřený, standardizovaný, průmyslově užívaný jazyk
- podpora v CASE nástrojích

části UML 2.0 specifikace

- Superstructure – definuje zápis a sématiku diagramů a jejich modelovacích elementů
- Infrastructure – definuje metamodel na němž je postavena část 1.
- Object constrain language (**OCL**) - jazyk pro specifikaci vstupních a výstupních podmínek, invariantů v jednotlivých diagramech.
- Diagram interchange **- popis XML struktur** pro výměnu konkrétních modelů mezi jednotlivými modelovacími nástroji.

Základní mechanizmy UML

Specifikace

Každý element by měl být **specifikován textem**, který popisuje sémantiku tohoto elementu. Tato specifikace upřesňuje, blíže popisuje, udává smysl modelovaného elementu. Popisuje business pravidla elementů (tudíž má největší význam u elementů popisujících problémovou doménu).

Ozdoby (adornments)

Další informace známé o elementu modelu. Každý element může být vyjádřen jednoduchým tvarem, ale je možno k němu přidávat i další informace - ozdoby (např. **seznam metod, seznam atributů, název objektu, stereotyp**,..).
Proč je těchto ozdob u elementu zobrazeno někdy více a někdy méně:

- model vytváříme postupně : zpočátku máme málo informací, které postupně doplňujeme
- tvorbou určitého diagramu sledujeme určité cíle - nechceme v něm zobrazit ty podrobnosti, které jsou v tomto nepodstatné z hlediska toho, co právě chceme zdůraznit (jedno z prvořadých hledisek při tvorbě diagramu je totiž **snadná čitelnost**)

Podskupiny (common divisions)

Udávají, jak je možno rozdělovat jednotlivé elementy; první způsob dělení :

- **klasifikátor a instance**: objekt je instance, třída je klasifikátor
- **rozhraní a implementace**: protokol zpráv je rozhraní objektu

- abychom použili (již hotový) objekt, nemusíme znát jeho implementaci - stačí nám znát jeho rozhraní, vnitřek objektu může být libovolně změněn

Mechanismy rozšiřitelnosti

Jazyk UML obsahuje připravené mechanismy umožňující rozšířit jazyk tak, aby vyhovoval momentálním potřebám. Máme k dispozici tři mechanismy rozšiřitelnosti:

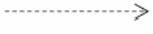
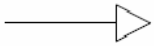
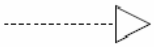
- **omezení** (constraints) : text ve složených závorkách {}. Podmínka či pravidlo v tomto textu musí být vždy splněna
- **stereotypy** (stereotypes) : s jejich pomocí lze z existujícího elementu vytvořit nový
- **označené hodnoty** (taggedd values) : umožňuje přidávat nové vlastnosti k elementům modelu. Zavedeme ji přidáním názvu s připojenou vlastní hodnotou ve složených závorkách, např. {autor=pavus, verze=0.1}
- **profily**: Profil UML definuje množinu stereotypů, značek a omezení, díky nimž lze jazyk UML přizpůsobit konkrétnímu účelu. (např. můžeme mít nějaký profil modelování aplikací pro platformu J2EE, jiný profil pro .NET, apod.)

Stavební bloky UML

Modelovací elementy

- **Strukturální** – class, interface, collaboration, active class, component, node, (lze si je představit jako podst. jména UML modelů)
- **Behaviorální** – interakce, stavové stroje, use case
- **Skupiny** – balíček (package)

Vztahy – Relationships

vztah	UML	význam
závislost (dependency)		zdrojový prvek závisí na cílovém a může být ovlivněn změnou jeho stavu
asociace (association)		Popis množiny spojení mezi objekty
agregace (aggregation)		cílový element je součástí zdrojového
kompozice (composition)		silnější forma agregace
generalizace (generalization)		zdrojový prvek je specializace obecnějšího cílového prvku a může jím být zastoupen-nahrazen
realizace (realization)		cílový element realizuje chování, které zdrojový element specifikuje

Diagramy struktury systému

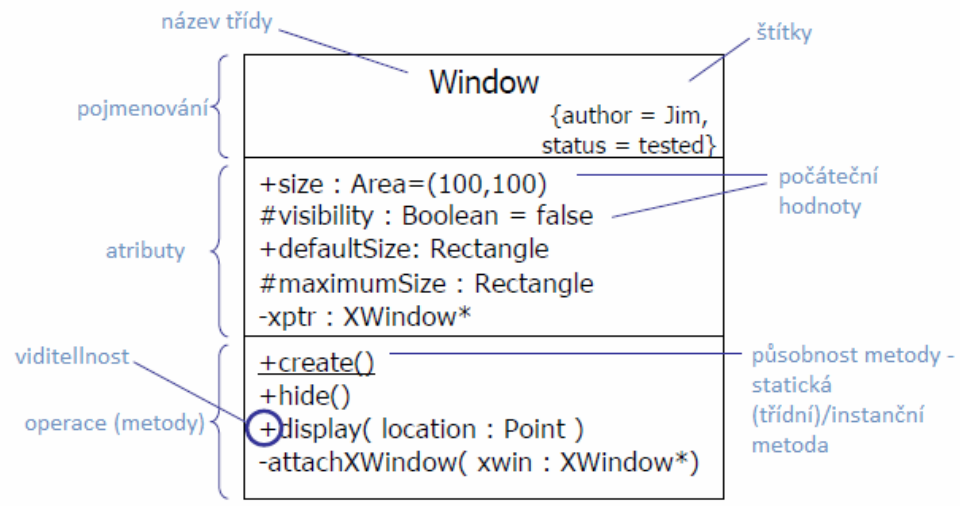
Diagram tříd

-statická struktura systému, popis typů objektů a statických vztahů mezi nimi
Tři pohledy na systém při použití diagramu tříd:

- **Konceptuální** – koncepty aplikační domény, bez vztahu k implementaci, jazykově nezávislá.
- **Specifikační** – pohled na program, specifikace rozhraní bez specifikace implementace.
- **Implementační** – je vidět implementace - hranice nejsou ostré, UML podporuje všechny tři pohledy

Tipy:

- nesnažit se použít veškerou dostupnou notaci
- přizpůsobit pohled etapě projektu:
  - analýza: konceptuální pohled
  - návrh: specifikační + implementační
- koncentrovat se na klíčové oblasti
- nezabřednout brzy do implementačních detailů



Návrhové relace

- **Agregace**
  - Volná vazba mezi objekty
  - Typ celek/součást (celek řídí chod relace, součást poskytuje služby)
  - Agregované objekty
    - mohou být sdílené
    - nezanikají se zánikem celku
  - Celek někdy existuje nezávisle na součástech, někdy je na nich závislý
  - Součást může být sdílena více celky

- **Kompozice**
  - Silnější forma agregace

    - Součásti nemohou existovat mimo celek
    - Součást patří právě jednomu celku
    - Celek odpovídá za použití svých součástí
    - Za předpokladu, že odpovědnost za součásti přejde na jiný objekt, může celek součásti uvolnit.
    - Je-li zničen celek, musí zničit své součásti (nebo přenést odpovědnost na jiný objekt)

Dědičnost tříd

- Potomci dědí
    - Atributy
    - Operace
    - Relace
    - Omezení
- Potomci mohou ke zděděnému fondu přidat novou charakteristiku

Realizace rozhraní

Diagram komponent

- propojení komponent
- komponenta je část systému, která realizuje specifikovaná **rozhraní**
- komponenta je **modulární nahraditelná** část systému, která zapouzdřuje svůj obsah (stav a chování)

Diagram nasazení

Mapuje SW artefakty na fyzické uzly.

Diagramy chování (behaviorální)

Modelování případů užití

Use case diagram (UC diagram) zobrazuje chování systému (nebo jeho části) z hlediska uživatele. Vychází z funkčních požadavků na systém.
postup při modelování:

- najít hranice systému (za kterou funkčnost je systém zodpovědný a za kterou již není)
- najít aktéry
- najít případy užití – **diagramy** + **scénáře**

Aktéři

Aktér specifikuje roli, kterou si může vnější entita (osoba, HW zařízení, jiný informační systém) přisvojit. Akteři definují hranice systému. Při identifikaci aktérů zjišťujeme:

- Kdo nebo co používá systém?
- V jakých rolích se přitom nachází?
- Kdo instaluje/nasazuje systém?

- Kdo systém udržuje?
- Které jiné systémy tento systém využívají?
- Spouští se nějaká událost v závislosti na čase? (např. zálohování apod.)

Případy užití

- Případ užití je nějaká činnost, kterou aktéři se systémem provádějí. Případy užití jsou vždy iniciovány aktérem.

Scénáře případů užití

POZOR! Scénáře případů užití jsou nedílnou součástní modelu. Samotné diagramy (bez scénářů) nemají potřebnou vypovídací hodnotu.
Struktura scénáře:

- název
- identifikátor
- stručný popis
- seznam primárních aktérů
- seznam sekundárních aktérů
- Vstupní podmínky
- Hlavní scénář, popisuje jednotlivé interakce mezi systémem a aktérem
- Alternativní scénáře
- Výstupní podmínky (např. Kniha byla zapůjčena, byl odeslán příkaz k výdeji)

Relace include vs. relace extend

include

- vyčleňuje společné kroky několika případů užití do samostatného případu užití
- klientský případ užití je bez dodavatelského neúplný

extend

- umožňuje rozšířit případ užití o nové chování
- bázový případ užití neví o rozšiřujícím; má pro něj jen tzv. místa rozšíření; je úplný i bez rozšiřujícícho
- rozšiřující bývá zpravidla bez bázového neúplný

Diagram aktivit

- popis dynamických aspektů systému
- tok řízení z aktivity do aktivity
- modelování obchodních (business) procesů a workflow
- soustřeďuje se spíše na proces výpočtu než na objekty účastnící se výpočtu
- akce, partitions, objektové uzly, hrana, tok, počáteční a koncový uzel

Stavový diagram

- založeno na state machine
- vyjadřuje stavy určitého objektu a přechody mezi těmito stavy

- **přechod** je relace mezi dvěma stavy, kde objekt v prvním stavu přejde do druhého stavu na popud nějaké **události**, pokud jsou splněny specifikované **podmínky** (guards), přičemž se provede specifikovaný **efekt**

sekvenční diagram

- vhodný pro zdůraznění **časové posloupnosti interakcí**
- objekty si posílají **zprávy** (synchronní a asynchronní)
- lze řídit tok pomocí **bloků** loop (while, for), alt (if, else), opt(if), par (paralelní zpracování)

diagram komunikace

- **podobný jako sekvenční diagram**, ale zdůrazňuje **strukturální aspekty spolupráce** (kdo s kým komunikuje)
- **nezdůrazňují časovou posloupnost interakcí**
- uzly, komunikační cesty, zprávy