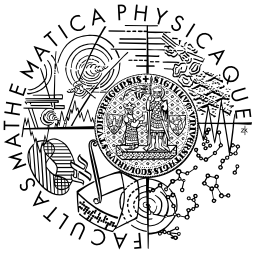


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁRSKA PRÁCA

Richard Hvizdoš

Nástroj pro konceptuální modelování multi-modelových dat

Katedra softwarového inženýrství

Vedúci bakalárskej práce: Ing. Pavel Koupil, Ph.D.

Študijný program: Informatika

Študijný obor: Informatika se specializací
Databáze a web

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dňa ....

Podpis autora

Chcel by som poďakovať svojmu vedúcemu Ing. Pavlovi Koupilovi, Ph.D., za odbornú pomoc, cenné rady a trpezlivosť pri vypracovávaní ročníkového projektu, a tejto bakalárskej práce. Ďalej chcem poďakovať rodine a svojmu blízkemu okoliu za podporu, a pochopenie.

Názov práce: Nástroj pro konceptuální modelování multi-modelových dat

Autor: Richard Hvizdoš

Katedra: Katedra softwarového inženýrství

Vedúci bakalárskej práce: Ing. Pavel Koupil, Ph.D., Katedra softwarového inženýrství

Abstrakt: Cieľom tejto BP je vytvoriť aparát podporujúci modelovanie v jazyku UML a novo vytvorenom kategorickom modeli (CAT), taktiež preklad UML modelu na CAT model. Práca obsahuje základný popis UML, CAT, rešerš dostupných editorov na modelovanie a v neposlednom rade programátorskú a používateľskú dokumentáciu. V prílohách sú mimo iné zobrazené aj testy, návrhový vzor a sekvenčné diagramy vybraných metód. Výsledok práce je funkčný, používateľský prívetivý nástroj dostupný na operačnom systéme Windows. Prednosťami nástroja sú napríklad: možnosti zmeny grafických vlastností objektov, uloženie a načítanie súboru vo formáte JSON, možnosť otvoriť viacero canvasov v jednom okne nástroja.

Kľúčové slová: multi-modelové dáta konceptuální modelování CAT UML

Title: A tool for conceptual modeling of multi-model data

Author: Richard Hvizdoš

Department: Department of Software Engineering

Supervisor: Ing. Pavel Koupil, Ph.D., Department of Software Engineering

Abstract: The aim of the bachelor's thesis is to create a tool that supports modeling in the UML and the newly created categorical model (CAT), as well as translation from UML model to CAT model. Thesis contains basic description of UML and CAT, research of popular tools for modeling, programming and user documentation. The appendices contains tests, design pattern and sequence diagrams of selected methods. The outcome of the thesis is functional, user friendly tool available on the Windows operating system. The advantages of the tool are for example possibility to change grafic properties of objects, save and load file in JSON format or open multiple canvases in one tool window.

Keywords: multi-model data conceptual modeling CAT UML

Obsah

Úvod

Modelovanie a následné spravovanie údajov s viacerými modelmi je veľkou výzvou, pretože používatelia pracujú s rôznymi dátovými modelmi (t. j. nielen s relačnými údajmi, ale aj s hierarchickými a grafovými údajmi) [3]. Medzi najobľúbenejšie používané dátové modely patria dokumenty eXtensible Markup Language (XML) [12], dokumenty JavaScript Object Notation (JSON) [4], stĺpcové údaje, grafové údaje, Resource Description Framework (RDF) [11], model polí a model kľúč/hodnota [9]. Tieto modely sú si v mnohom zdanlivo podobné, stavebným prvkom je vždy dvojica kľúč-hodnota [8]. Okrem toho môžu identifikovať jednoduché (atribúty) a zložité (tabuľky, dokumenty, vrcholy, hrany, polia, dvojice, trojice) prvky. Líšia sa v detailoch a ďalších vlastnostiach, ktoré sú často protichodné [7], napríklad:

- Ukladanie informácií o poradí atribútov/elementov, pričom niektoré modely (relačné) poradie úplne ignorujú, niektoré ho striktne dodržiavajú (XML) a niektoré používajú zmiešaný prístup (graf, JSON).
- Prístup založený na schéme, kde rozlišujeme medzi prístupom založeným na schéme (relačný model, XML, stĺpcový model, model poľa), prístupom bez schémy (graf, kľúč/hodnota) a zmiešaným prístupom (JSON).
- Reprezentácie chýbajúcich dát (napríklad: null, neznáma vlastnosť, chýbajúci objekt).
- Štruktúrovanie informácií, niektoré sú plne štruktúrované (relačné modely), niektoré čiastočne štruktúrované (JSON, XML), niektoré formáty nie sú štruktúrované vôbec (audio, email).
- Podpora agregácie, kde niektoré sú agregátne orientované (stĺpcový model, model kľúč/hodnota) a niektoré sú naopak agregátne neznalé (relačný model, graf).

Okrem toho, kombinácia viacerých modelov vytvára nové vlastnosti, ktoré sme na úrovni jednotlivých modelov nepozorovali.

Všetky tieto (protichodné) vlastnosti je veľmi ťažké pochopiť. Dokonca ani existujúce prístupy, ako napríklad Entity-Relationship (ER) [2] alebo Unified Modeling Language (UML) [5], ich často nedokážu modelovať jednotným spôsobom. Dokonca, niektoré vlastnosti moderných a populárnych dátových modelov vôbec nemožno vyjadriť v typických prístupoch modelovania (napr. identifikátory typov vzťahov v ER na rozlíšenie duplicitných prvkov polí v modeloch dokumentov). Preto bol navrhnutý nový prístup, ktorý uvažuje o rozmanitých a protichodných vlastnostiach údajov viacerých modelov, tzv. kategorický model (CAT) [10].

Hoci je CAT robustný model, pre používateľov je stále ťažko uchopiteľný. Preto bol navrhnutý algoritmus, ktorý umožňuje preložiť schému ER do CAT [10], a bezstratový algoritmus, ktorý umožňuje vyjadriť niektoré schémy v CAT do ER (vyjadrovacia sila CAT je vyššia ako vyjadrovacia sila ER, a preto nie všetky schémy v CAT možno bezstratovo preložiť do ER).

Ďalším prirodzeným krokom je podpora UML, t. j. navrhnúť algoritmus na preklad medzi schémami CAT a UML (a naopak) a vytvoriť editor, ktorý tento preklad umožní.

Cieľom tejto bakalárskej práce je navrhnúť a implementovať aparát, ktorý (1) umožňuje modelovanie viacmodelových dátových schém pomocou jazykov UML a CAT a ktorý (2) umožňuje preklad medzi týmito jazykmi (t. j. umožňuje preklad schém UML do schém CAT).

0.1 Požiadavky na editor

Od začiatku boli na editor kladené požiadavky, popísané nižšie, ktorých splnenie je pre túto bakalársku prácu žiadúce.

Požiadavky:

- undo-redo operácie,
- vymazanie požadovaného objektu,
- vytvorenie UML modelu pre multi-modelové dáta,
- vytvorenie CAT modelu pre multi-modelové dáta,
- algoritmus, ktorý preloží jeden model do druhého (UML model na CAT modelu),
- uloženie schémy do súboru (t. j. serializácia),
- načítanie existujúcej schémy zo súboru (t. j. deserializácia),
- podporovať karty (t. j. možnosť otvoriť viac canvasov naraz v jednom okne editoru),
- presun a zmena veľkosti požadovaného objektu,
- pestrý grafický výstup napríklad prostredníctvom zmeny farby, podfarbenia alebo štýlu textu.

Štruktúra tejto bakalárskej práce je nasledovná: V kapitole 1 opisujeme vlastnosti diagramov tried jazyka UML, v kapitole 2 opisujeme vlastnosti prístupu CAT, v kapitole 3 robíme dôkladnú rešerš existujúcich nástrojov na modelovanie diagramov tried a ich porovnanie, v kapitole 4 uvádzame programátorskú dokumentáciu navrhovaného a implementovaného nástroja, v kapitole 5 nasleduje používateľská dokumentácia a nakoniec záver.

Kapitola 1

UML

Unified Modelling Language (UML) [5] je grafický jazyk používaný na vizuálne modelovanie systémov. Zjednodušuje návrh a vývoj, má široké uplatnenie, ale najčastejšie je spojovaný s návrhom objektovo orientovaných softvérových systémov.

Celkovo existuje trinásť rôznych typov diagramov UML. Diagramy rozdeľujeme na statický model, ktorý zachytáva predmety a štruktúrne asociácie medzi predmetmi, a dynamický model zachytávajúci spôsob, ako na seba jednotlivé predmety navzájom pôsobia, aby bolo dosiahnuté požadované správanie softvérového systému. Medzi statické modely patrí diagram tried, zložených štruktúr, komponent, nasadenia, balíčku a objektový diagram. Do druhej časti, t. j. do dynamického modelu patrí diagram aktivít, interakcie, postupnosti, komunikácie, časový diagram, stručný diagram interakcie, diagram stavového automatu a diagram prípadu použitia. [5]

V tejto bakalárskej práci sa zameriame iba na UML diagram tried, ktorého prvky sú popísané nižšie. Zvyšné diagramy jazyka UML nebudeme používať, pretože modelujeme štruktúru dát. Pri tomto modelovaní nemáme pre ostatné podporované diagramy využitie (resp. je minimálne).[1]

Jazyk poskytuje syntax, pomocou ktorej sa modeluje. Stavebné bloky UML diagramov tried sa skladajú z :

- predmetov predstavujúcich samotné modelovacie prvky,
- vzťahov vyjadrujúcich sémantickú väzbu medzi viacerými objektami,
- diagramov zachytávajúcich konkrétnu skupinu objektov a vzťahov uložených v modeli.

1.1 Diagram tried

Diagram tried obsahuje návrhové triedy, ktorých špecifikácia je na takom stupni, že ich je možné implementovať. To znamená, že návrhové triedy modelujú spôsob, akým má byť správanie systému implementované. [5]

[1] Poznámka: V prílohe A.5 je použitý UML sekvenčný diagram a na obrázku 4.2, v kapitole 4, je použitý UML diagram prípadov použitia. Tieto diagramy pomáhajú zjednodušiť vizualizáciu možností editoru alebo postupu pri vykonávaní príkazov. Samotné prvky daných diagramov sú v daných častiach popísané a vysvetlené.

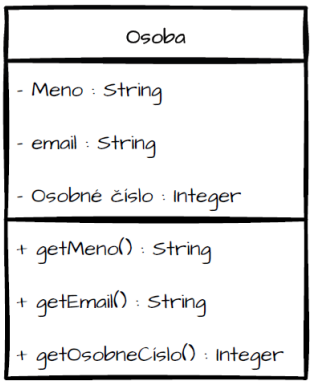
1.1.1 Trieda

Trieda, v literatúre označovaná aj ako koncept [5], reprezentuje časť systému alebo softvéru. Samotná trieda sa skladá z troch častí v uvedenom poradí:

- názov,
- atribúty,
- metódy.

Celá trieda sa obvykle zobrazuje ako obdĺžnik s graficky oddelenými časťami. Pri návrhu taktiež špecifikujeme viditeľnosť použitím niektorého zo znakov: `+` (verejný), `-` (súkromný), `#` (chránený) alebo `~` (súkromný balíček) na začiatku atribútu a metódy.

Napríklad na obrázku 1.1 má trieda, s názvom `Osoba`, iba súkromné atribúty a verejné metódy, ktorých návratová hodnota je hodnota daného súkromného atribútu.

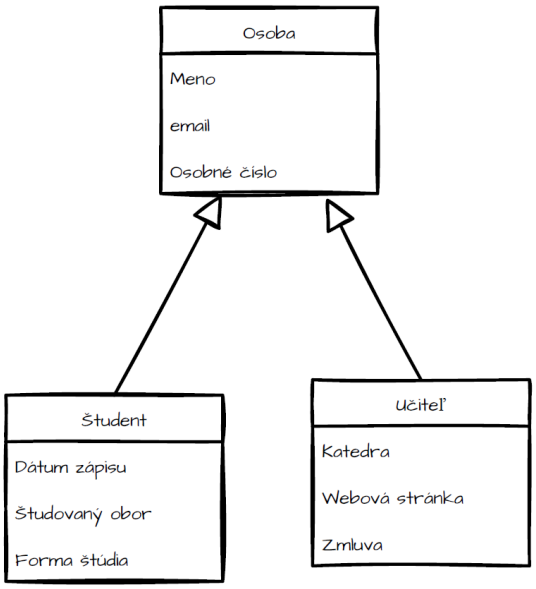


Obr. 1.1: UML trieda

1.1.2 Dedičnosť

Dedičnosť popisuje hierarchiu medzi triedami, keď podradená trieda presnejšie špecifikuje nadradenú. Podradená trieda dedí od nadradenej jej atribúty a metódy (okrem privátnych), relácie a obmedzenia. Graficky sa označuje prázdnym trojuholníkom na konci prepojenia pri nadradenej triede.

Napríklad na obrázku 1.2 sú potomkovia `Študent` a `Učiteľ`, každá trieda má svoje vlastné atribúty a môže pristupovať aj k atribútom jej rodiča, v tomto prípade to je trieda `Osoba`.



Obr. 1.2: UML dedičnosť

1.1.3 Asociácia

V UML existujú asociácie:

- binárne – medzi dvoma triedami,
- ternárne – medzi troma triedami,
- n-árne – medzi troma a viacerými triedami.

Asociácie môžu napríklad obsahovať názov, multiplicity (násobnosti), môže byť vyjadrená asociačnou triedou (sekcia 1.1.6) alebo môže byť medzi viacerými triedami (sekcia 1.1.7).

Multiplicita je obmedzenie špecifikujúce počet objektov danej triedy, ktoré sa danej relácie účastnia v ľubovolnom okamihu. [5] Píše sa na koncoch asociácie vo formáte:

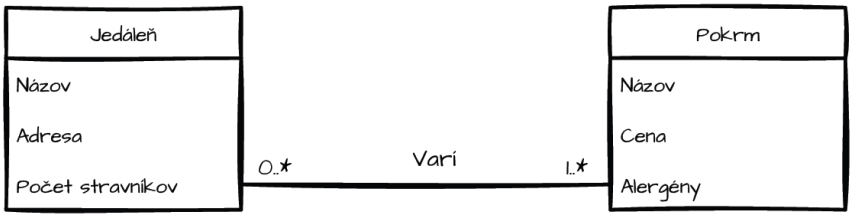
- `x..y` – počet z rozsahu (od..do),
- `x,y` – počet z vybraných možností (počet x alebo y),
- `z,x..y` – požadovaný počet z alebo počet z rozsahu x až y,
- `x` – nutný počet x,

kde `x`, `y` a `z` sú celé čísla alebo výraz, ktorého výsledok je celé číslo, a navyše `y` môže byť aj znak „`*`“ určujúci neohraničený počet. Rozsah všetkých multiplicít je od 0 do *, to znamená, že nie je možné mať zápornú multiplicitu. Napríklad multiplicita `5..*` vyjadruje počet *5 a viac*, `17,20..22` vyjadruje počet *17 alebo 20 až 22*.

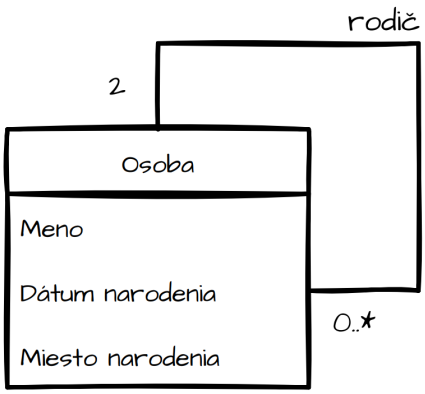
Čítanie multiplicity v UML prebieha mierne odlišným spôsobom ako v ER. [2] Čítanie začíname vo vybranom objekte v smere požadovaného vzťahu s multiplicitou, ktorá sa nachádza na opačnej strane vzťahu (ďalej od objektu, v ktorom sme začínali a bližšie k objektu, v ktorom čítanie ukončíme). Pri vhodnom modelovaní môže mať čítanie nasledovný postup: *názov objektu na začiatku, názov vzťahu, multiplicita pri koncovom objekte, názov koncového objektu.*

Napríklad na obrázku 1.3 môžeme prečítať: *Jedáleň varí aspoň jeden pokrm (resp. jeden a viac pokrmov).*

Binárn Asociácia môže byť taktiež vytvorená iba pomocou jedného objektu, ako napríklad na obrázku 1.4. Nazýva sa reflexívna alebo rekurzívna asociácia. Z obrázku môžeme vyčítať: *Osoba má práve dvoch rodičov., Osoba môže byť rodičom pre 0 až * osôb (potomkov).*



Obr. 1.3: UML asociácia



Obr. 1.4: UML rekurzívna (reflexívna) asociácia

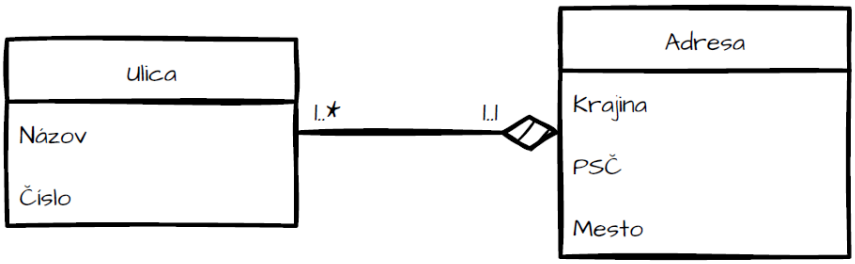
1.1.4 Agregácia

Agregácia vyjadruje slabší vzťah medzi objektom a jeho časťami, vychádza z asociácie. Objekt reprezentujúci celok sa nazýva agregácia. Graficky sa označuje prázdnym diamantom na konci vzťahu pri celku.

Vlastnosti:

- celok môže existovať bez svojich súčastí,
- súčasti môžu existovať nezávisle na celku,
- súčasti môžu byť súčasťou viacerých celkov.

V nasledujúcom príklade, na obrázku 1.5, je `Ulica` súčasťou `Adresy`, ktorá reprezentuje celok.



Obr. 1.5: UML agregácia

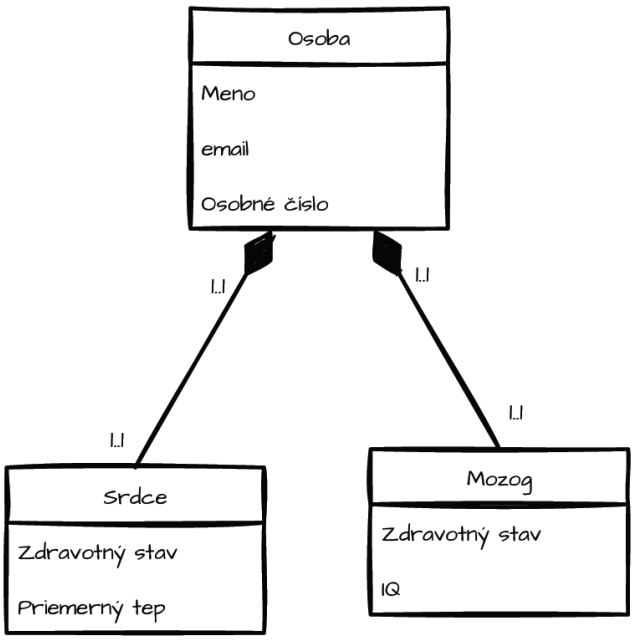
1.1.5 Kompozícia

Kompozícia vyjadruje silnejší vzťah medzi objektom a jeho časťami, vychádza z asociácie. Objekt reprezentujúci celok sa nazýva kompozitný (zložený). Graficky sa označuje plným diamantom na konci vzťahu pri celku.

Vlastnosti:

- kompozitný objekt nemôže existovať bez svojich súčastí,
- súčasti môžu byť súčasťou práve jednej kompozície.

V príklade na obrázku 1.6 je kompozitný objekt `Osoba`, jeho súčasti sú `Srdce` a `Mozog`, bez ktorých celok existovať nemôže.

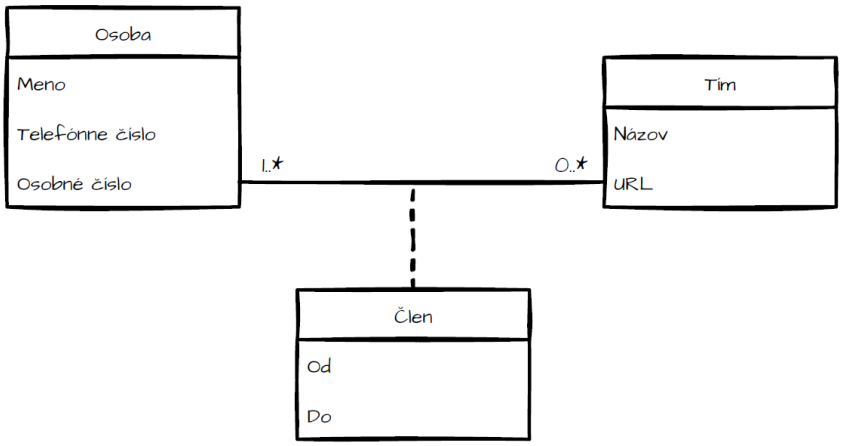


Obr. 1.6: UML kompozícia

1.1.6 Asociačná trieda

Je to alternatíva k asociácii, kedy asociácia môže navyše obsahovať ďalšie atribúty vo forme asociačnej triedy. Graficky sa označuje prerušovanou čiarou.

V príklade na obrázku 1.7 asociácia medzi `Osoba` a `Tím` obsahuje triedu `Člen`, ktorá špecifikuje vlastnosti danej asociácie.

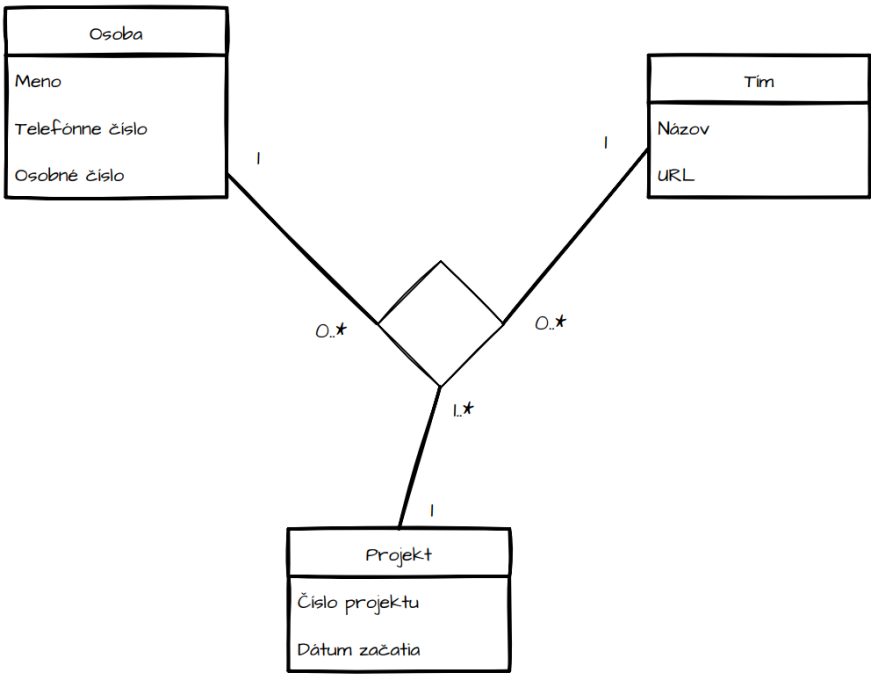


Obr. 1.7: UML asociačná trieda

1.1.7 N-árna asociácia

N-árna asociácia je podobná binárnej, ale obsahuje viac účastníkov. Graficky sa označuje kosoštvorcom.

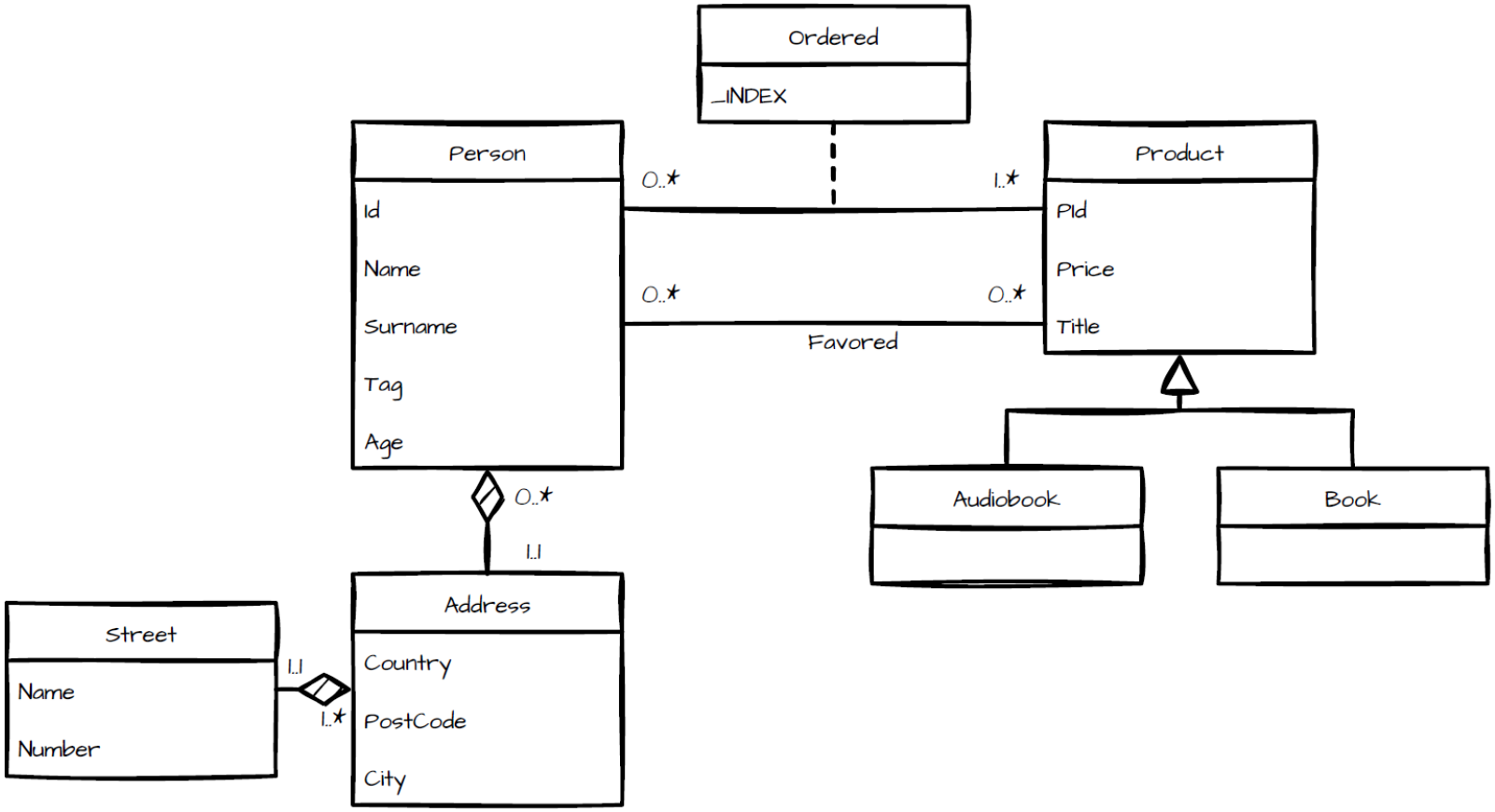
Napríklad obrázok 1.8 popisuje: *osoba pracuje na projekte iba ako člen tímu.*



Obr. 1.8: UML n-árna asociácia

1.2 Príklad

Nasledujúci príklad modelu, na obrázku 1.9, používa vyššie spomínané prvky jazyka UML. V časti 2.7 sa nachádza rovnaký model v jazyku CAT. Môžeme ich navzájom porovnať.



Obr. 1.9: UML príklad modelu

Kapitola 2

CAT

Kategorický model (CAT) [10] je jazyk, ktorý používa teóriu kategórií [1] na modelovanie objektov reálneho sveta a vzťahov medzi nimi. Bol vytvorený ako alternatíva k ER [2] a diagramu tried v UML [5], ktoré nereflektujú všetky štruktúrne vlastnosti moderných dátových modelov a multimodelových údajov [3].

Hlavnou ideou CAT [10] je reálny svet, v ktorom je všetko jednoduché alebo komplexné, všetko má svoju doménu a každý prvok domény môže byť identifikovaný.

Z toho vyplýva nasledujúce delenie prvkov jazyka na:

- jednoduchú vlastnosť – môžeme na ňu pozerať ako na entitný typ s jedným atribútom,
- komplexnú vlastnosť – môžeme na ňu pozerať ako na podmnožinu Kartézskeho súčinu jednoduchých a komplexných vlastností.

CAT reprezentuje schéma ako orientovaný graf, ktorý je zložený z objektov a šípok (funkcií) popísaných nižšie.

2.1 Objekt

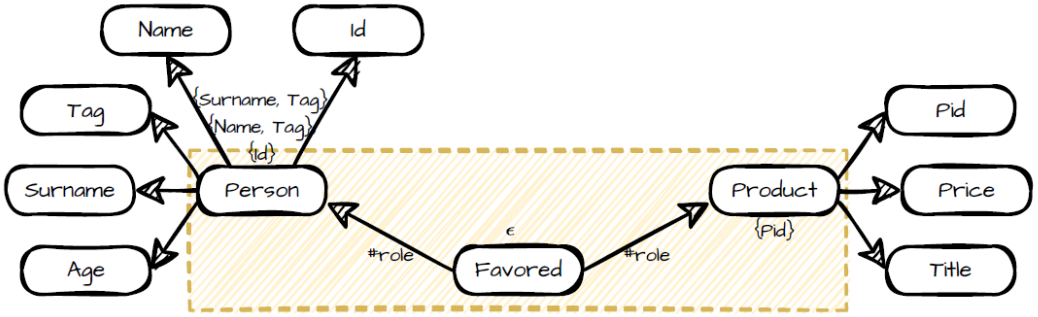
Objekt reprezentuje aktívnu doménu vlastností a zároveň každému objektu je pridelená identifikačná funkcia.

2.2 Prepojenie – hrana

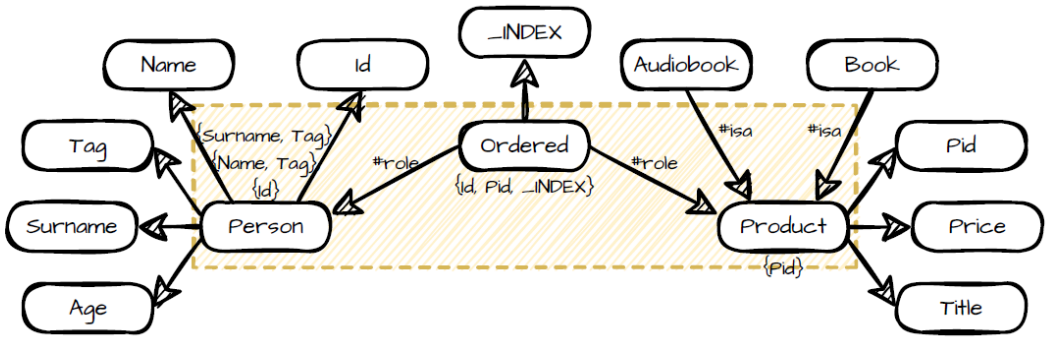
Hrana reprezentuje spojenie medzi objektami (napríklad: „má vlastnosť“, „má rolu“ alebo „je“). Vykresľuje sa ako orientovaná šípka. Hranu môžeme označiť tagmi:

- **#role** Týmto tagom označujeme vzťahy, ktoré po preklade z UML zastávajú funkciu asociácie. Od asociácie preberajú jej pomenovanie, ktoré sa nachádza v objekte uprostred, z ktorého vedú hrany s tagom #role do pôvodne prepojených prvkov v UML schéme. Príklad tejto hrany je na obrázku 2.1.

- **#ISA** Týmto tagom označujeme hierarchiu medzi vlastnosťami, t. j. dedičnosť v UML schéme. Na obrázku 2.2 sú `Audiobook` a `Book` potomkovia `Product`, t. j. obaja môžu pristupovať k vlastnostiam rodiča.
- **bez tagu** Prepojenie bez použitia tagu používame pri priradení vlastnosti danému objektu. Predstavuje to zvyšné UML vzťahy spomínané v kapitole 1.

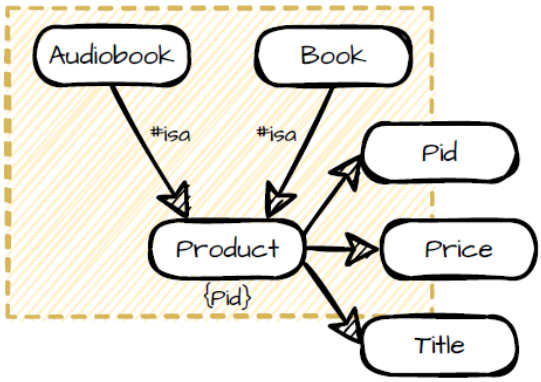


(a) CAT #role, identifikujúci sám seba, prevzaté z [6]



(b) CAT #role, identifikovaný pomocou jeho účastníkov, prevzaté z [6]

Obr. 2.1: CAT #role



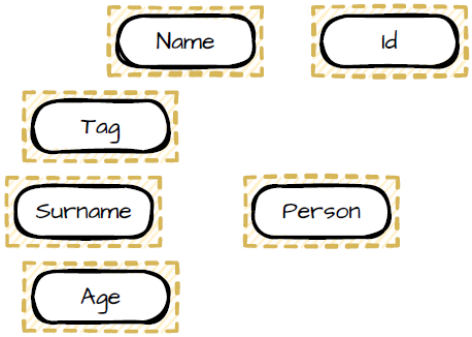
Obr. 2.2: CAT #ISA, prevzaté z [6]

2.3 Jednoduchá vlastnosť

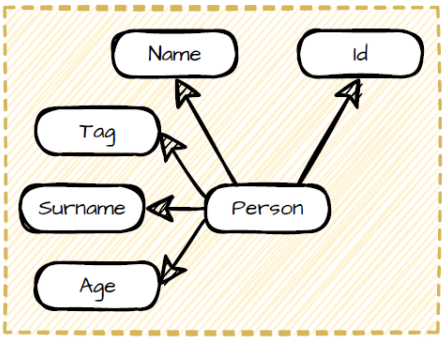
Jednoduchý objekt ako napríklad `Person`, `Id` alebo `Age`, zobrazené na obrázku 2.3a. To znamená, že sa neskladá z ďalších vlastností, ale existuje sám o sebe.

2.4 Komplexná vlastnosť

Komplexná vlastnosť vznikne prepojením jednoduchých, prípadne aj komplexných vlastností. Na nižšie uvedenom príklade, obrázku 2.3b, je komplexná vlastnosť `Person`.



(a) CAT jednoduché vlastnosti, prevzaté z [6]



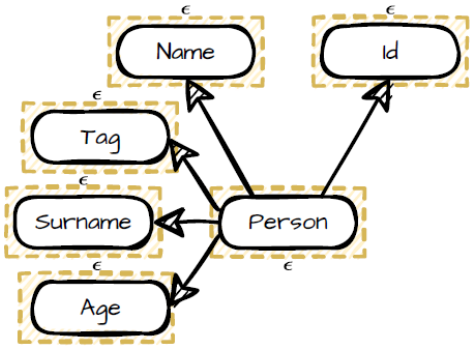
(b) CAT komplexná vlastnosť, prevzaté z [6]

Obr. 2.3: CAT objekty

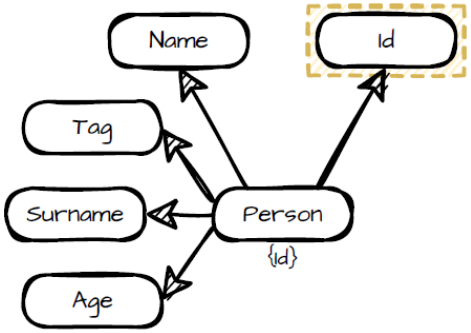
2.5 Identifikačná funkcia – identifikátor

Každej vlastnosti môžeme priradiť identifikátor. Východzí identifikátor je identifikátor seba samého (t. j. ε), ktorý je zobrazený na obrázku 2.4a. Komplexné vlastnosti môžu využívať na identifikovanie vlastnosti, s ktorými sú prepojené iba po smere hrany (obrázok 2.4b). Zároveň, komplexná vlastnosť môže mať viacero, navzájom sa prekrývajúcich, identifikátorov (obrázok 2.4d), ktoré môžu byť taktiež komplexné (t. j. pozostávať z viacerých vlastností).

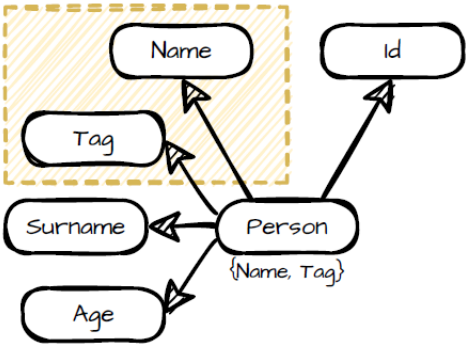
Napríklad na obrázku 2.4b je `Person` identifikovaný vlastnosťou `Id` (existuje isomorfizmus medzi `Person` a `Id`) a na obrázku 2.4c je `Person` identifikovaný komplexným identifikátorom (existuje isomorfizmus medzi `Person` a `Name` $\times$ `Tag`).



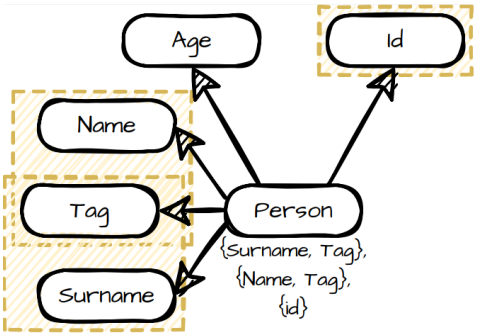
(a) CAT identifikátor seba samého, prevzaté z [6]



(b) CAT identifikovanie prostredníctvom vlastnosti, prevzaté z [6]



(c) CAT komplexný identifikátor, prevzaté z [6]



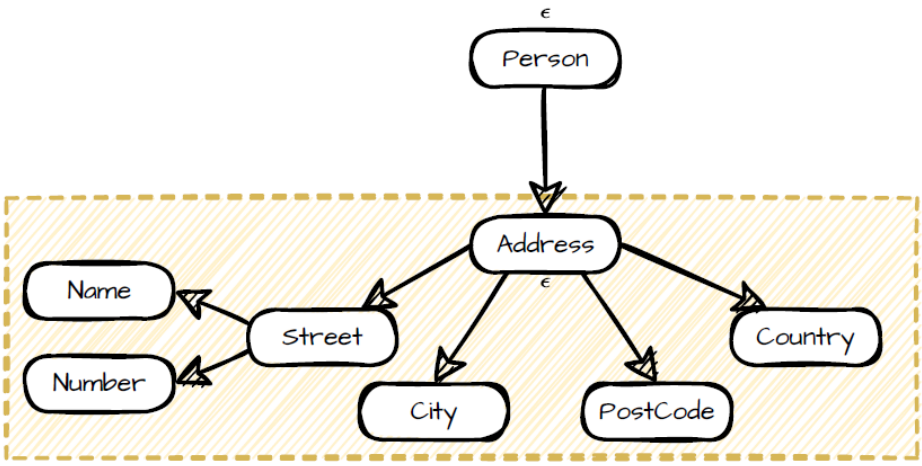
(d) CAT viacero prekrývajúcich sa identifikátorov, prevzaté z [6]

Obr. 2.4: CAT identifikátory

2.6 Štruktúrovaný atribút

Je to špeciálny typ atribútu, ktorý je ďalej štruktúrovaný. Podobá sa entitnému typu z konceptuálneho modelovania (ER), ale narozdiel od entitného typu nemôže existovať sám o sebe, t. j. musí byť súčasťou nejakého entitného alebo vzťahového typu. Z toho vyplýva zásadný rozdiel medzi typmi a atribútmi.

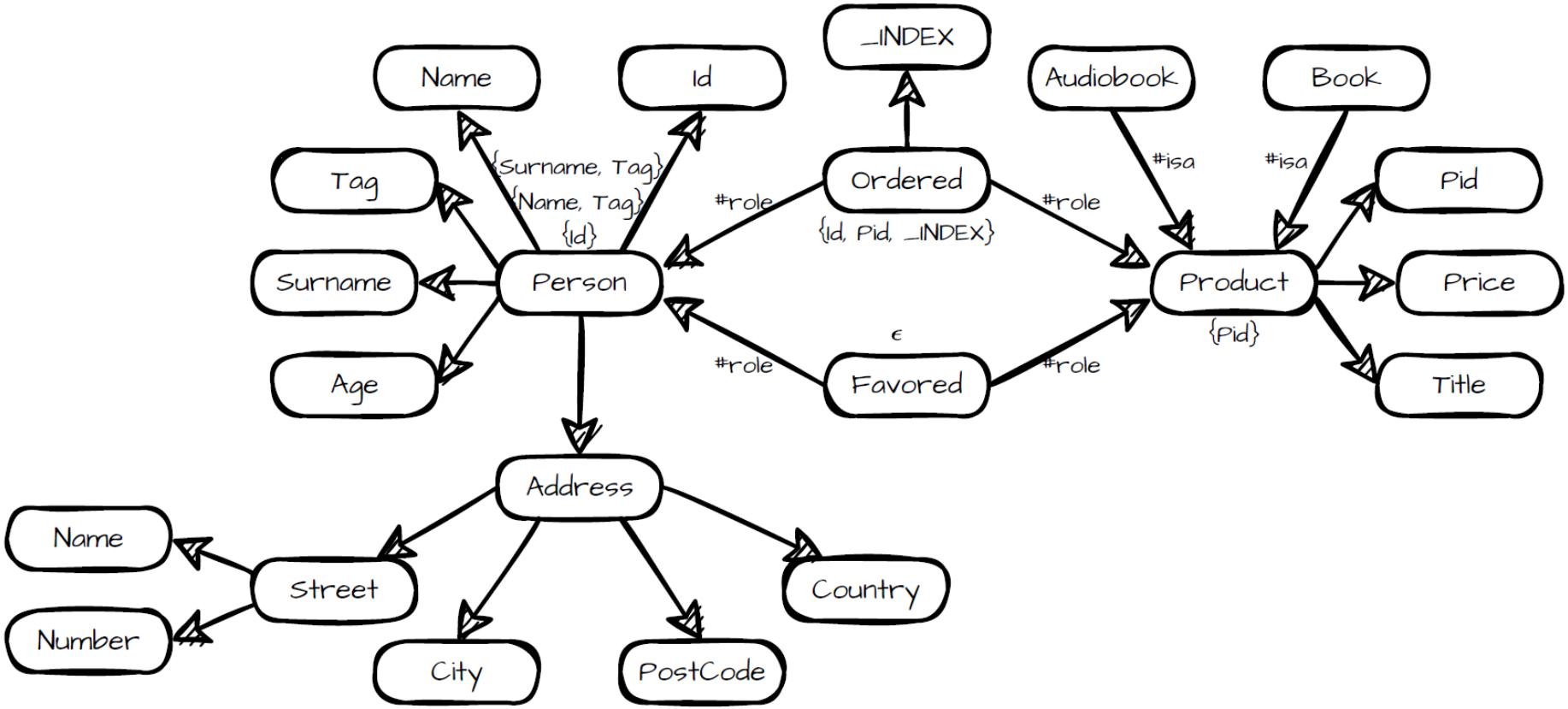
Na obrázku 2.5 môžeme `Address` chápať ako seba samého identifikujúci štruktúrovaný atribút, ktorý pozostáva zo `Street`, `City`, `PostCode` a `Country`. Všimnime si, že aj `Street` môžeme chápať ako štruktúrovaný atribút.



Obr. 2.5: CAT štruktúrovaný atribút, prevzaté z [6]

2.7 Príklad

Nasledujúci príklad modelu, na obrázku 2.6, obsahuje všetky, vyššie spomínané, možnosti modelovania CAT grafu. Pre porovnanie sa môžeme pozrieť na rovnaký model v jazyku UML, nachádzajúci sa v časti 1.2.



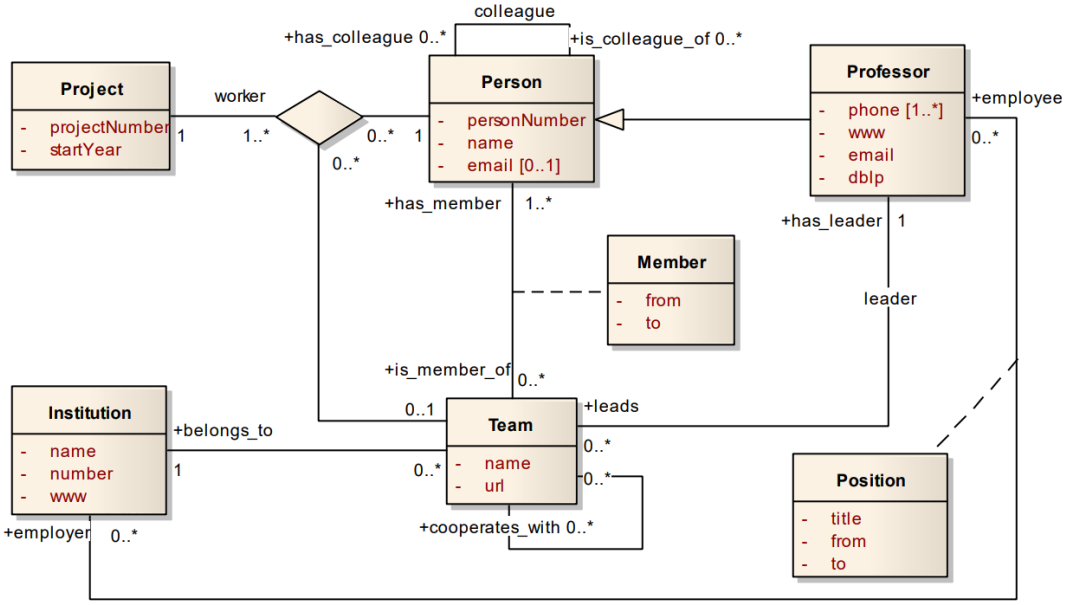
Obr. 2.6: CAT príklad modelu, prevzaté z [6]

Kapitola 3

Rešerš dostupných editorov

Nasledujúca rešerš obsahuje popis a porovnanie šiestich populárnych nástrojov podporujúcich UML modelovanie tried, na základe vybraných vlastností a dostupnosti online.

Na testovanie UML nástrojov používame príklad na obrázku 3.1 z prednášky Databázové systémy[1] na Matematicko-fyzikální fakultě Univerzity Karlovy.



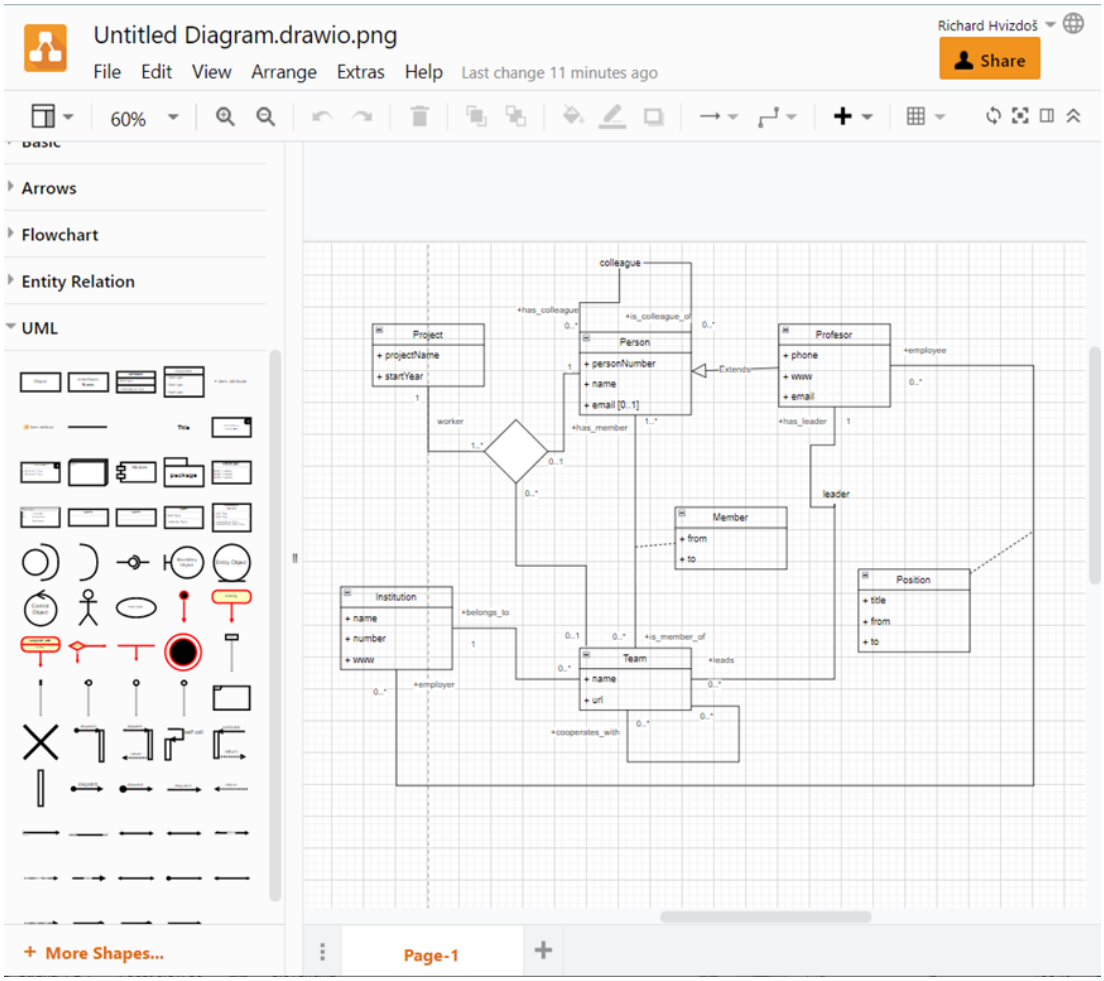
Obr. 3.1: Príklad UML modelu, používaného na testovanie nástrojov

3.1 Diagrams.net (Drawio)

Diagrams.net[2] je multifunkčný nástroj podporujúci rôzne druhy modelovania. Je k dispozícii pre všetky platformy v online forme prostredníctvom webovej aplikácie alebo off-line prostredníctvom desktopovej aplikácie.

Môžeme si vybrať z množstva kolekcií, ktoré obsahujú potrebné tvary, ikony či objekty.

[1] https://www.ms.mff.cuni.cz/~kopecky/vyuka/dbs/
[2] https://app.diagrams.net/



Obr. 3.2: Používateľské prostredie Diagrams.net

Z toho plynie aj jeho možná nevýhoda, v potrebe venovať viac času na zoznámenie sa s nástrojom. Po prvotnom zoznámení sa pôsobí prostredie, zobrazené na obrázku 3.2, príjemne a je možné jeho prispôsobenie.

U objektov je možné zmeniť ich vlastnosti ako napríklad podfarbenie, farbu čiary, typ čiary. Výsledný model si môžeme personalizovať a sprehľadniť. Zmena alebo pridanie textu je možné po dvojkliku na zvolené miesto. Takýmto spôsobom môžeme pridať multiplicity a pomenovania asociácií.

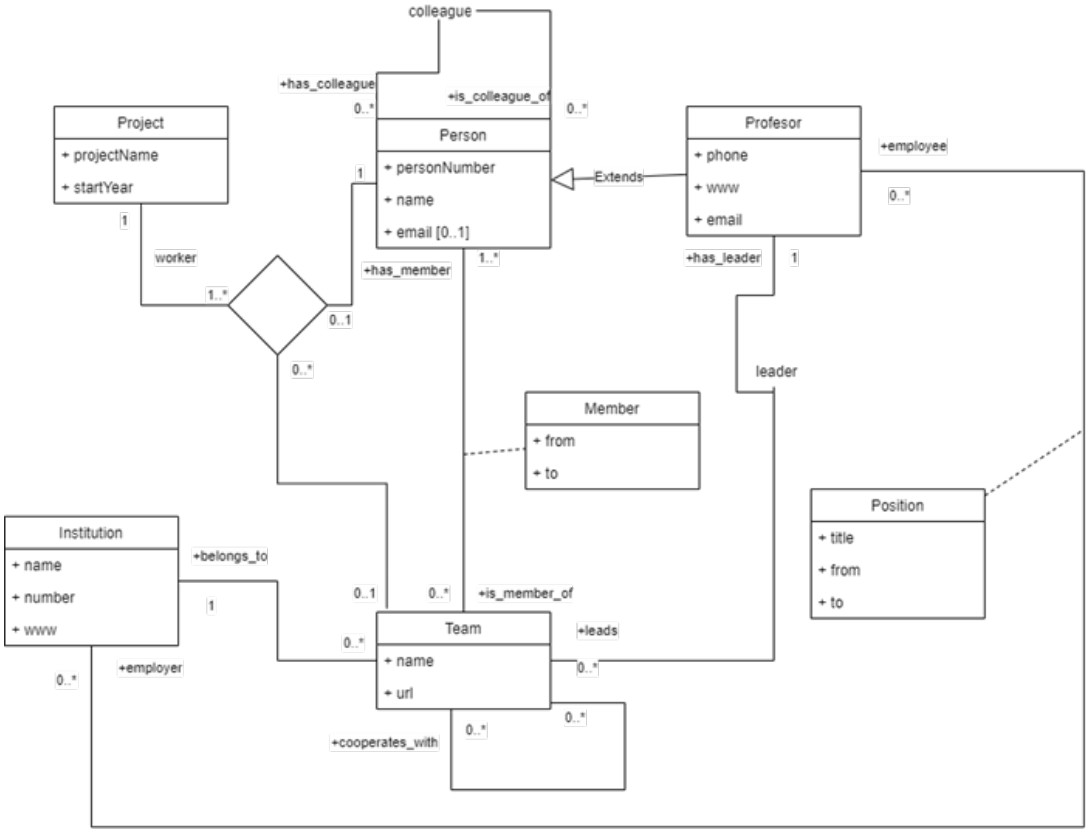
UML kolekcia obsahuje pomerne veľa možností na výber z asociácií, relácií, agregácií, závislostí a množstva iných objektov, preto je potrebné vedieť, čo je pre náš model vhodné. Samotné kreslenie asociácií pôsobí jednoducho a ich polohu môžeme prispôsobiť už vytvorenému modelu.

Používateľské prostredie má čistý, jednoduchý dizajn. Po načítaní stránky sa zobrazí plocha na modelovanie, ktorú je možné zväčšiť, zmenšiť, prípadne aj pridávať nové hárky. Na ľavej strane sa nachádza panel s objektmi, hore sa nachádza lišta s tlačidlami na prispôsobenie. Samozrejmé sú možnosti undo-redo operácií a kopírovanie namodelovaných objektov. Na druhej strane z dôvodu jeho univerzálnosti obsahuje množstvo objektov a ich ikony sú malé – pri voľbe správneho objektu je potrebné namieriť kurzor na menšiu plochu.

Nástroj nám ponúka veľa možností na uloženie modelu ako napríklad vo formáte Portable Network Graphics (PNG), Property Domain Footprint (PDF),

XML a zároveň možnosť jeho uloženia na rôznych online platformách (napríklad: Google Disk, GitLab a GitHub). Uložený model následne môžeme v editore opäť otvoriť a pokračovať v práci. Výsledný model je zobrazený na obrázku 3.3.

Diagrams.net ponúka veľa možností a bohatý výber, preto by som tento nástroj primárne odporúčal pokročilejším, alebo tým, ktorí vedia, čo potrebujú.



Obr. 3.3: Výsledný UML model použitím editora Diagrams.net

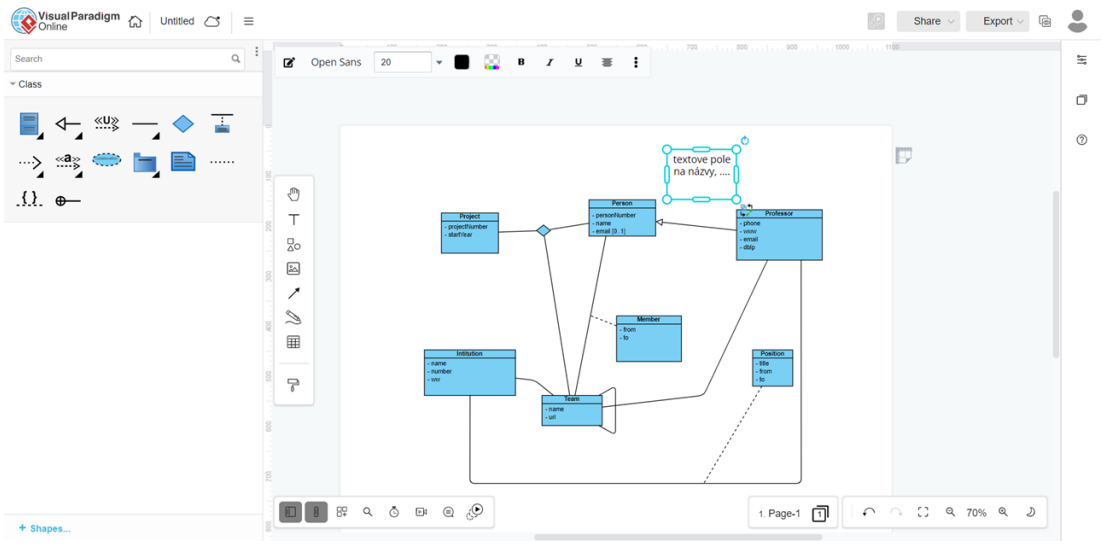
3.2 Visual Paradigm

Visual Paradigm[3] je mimo iné aj online nástroj na modelovanie. Po výbere class diagramov, z rôznych možností, sa nám načíta pracovné prostredie. Dizajn je jednoduchý a čistý. Na prvý pohľad nás zaujme malé množstvo ikon a tlačidiel. Ukážka pracovného prostredia je zobrazená na obrázku 3.4.

Na ľavej strane sa nachádza jediná kolekcia s objektami na modelovanie class diagramov. Je možné pridať aj iné kolekcie z výberu. Malé množstvo objektov na výber podporuje jednoduchšie a rýchlejšie orientovanie sa v samotnom editore. Možnosti na výber z objektov pre class diagram sú napríklad trieda, asociácie, asociačná trieda, ale nechýba ani n-árna asociácia.

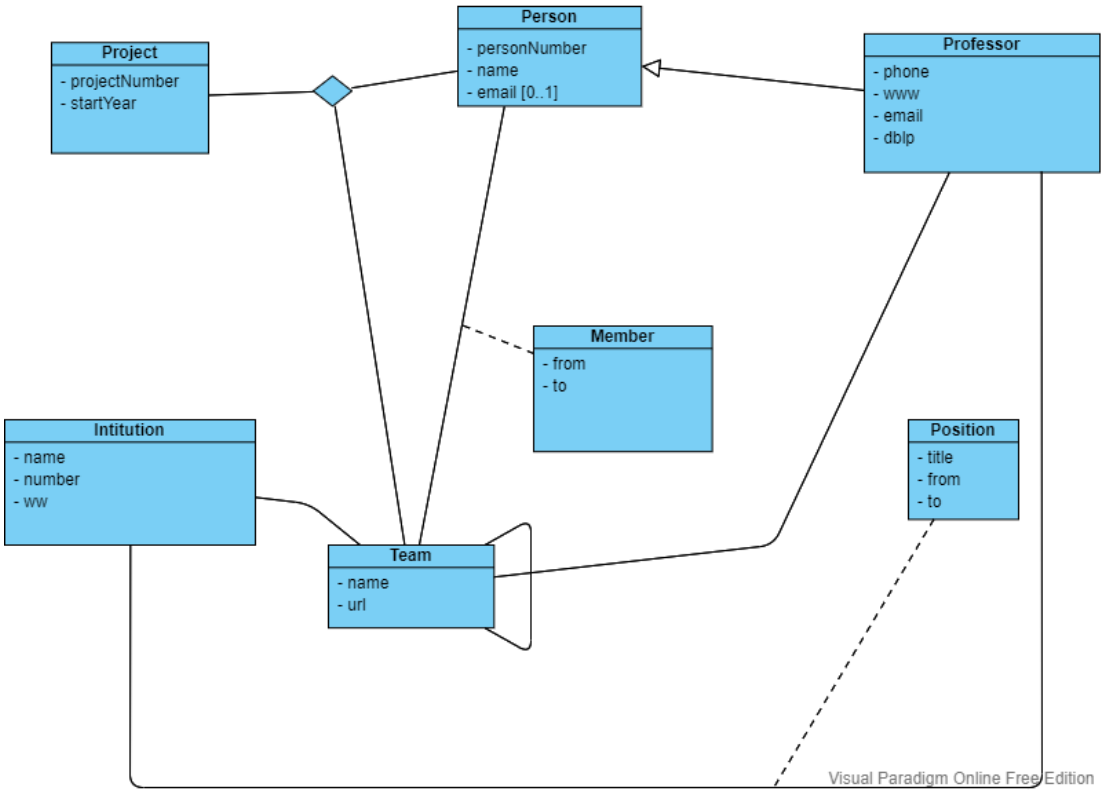
Nespornou výhodou pre vizuálne náročnejších používateľov je možnosť výberu grafického dizajnu jednotlivých objektov. Nechýba napríklad zmena podfarbenia, vloženie textu do modelu, zväčšenie či zmenšenie plochy na modelovanie. Nástroj taktiež ponúka nočný režim.

[3] https://online.visual-paradigm.com/app/diagrams/#diagram:proj=0&type=ClassDiagram&width=11&height=8.5&unit=inch



Obr. 3.4: Používateľské prostredie Visual Paradigm

Vytváranie asociácií je jednoduché, stačí ak vyberieme asociáciu z kolekcie a spojíme dva požadované objekty. Spojenie bude priame, takzvane vzdušnou čiarou. Z toho dôvodu následne musíme prepojenia prispôsobiť tak, aby boli dobre viditeľné a ničomu neprekážali. Prípadné multiplicity alebo pomenovanie je potrebné vytvoriť pomocou textového poľa a vhodne umiestniť.



Obr. 3.5: Výsledný UML model použitím editora Visual Paradigm

Objekty tried môžeme presúvať, kopírovať či zväčšovať. Menšie ťažkosti nastávajú pri pridávaní atribútov alebo metód triedy, keď musíme jedným pravým

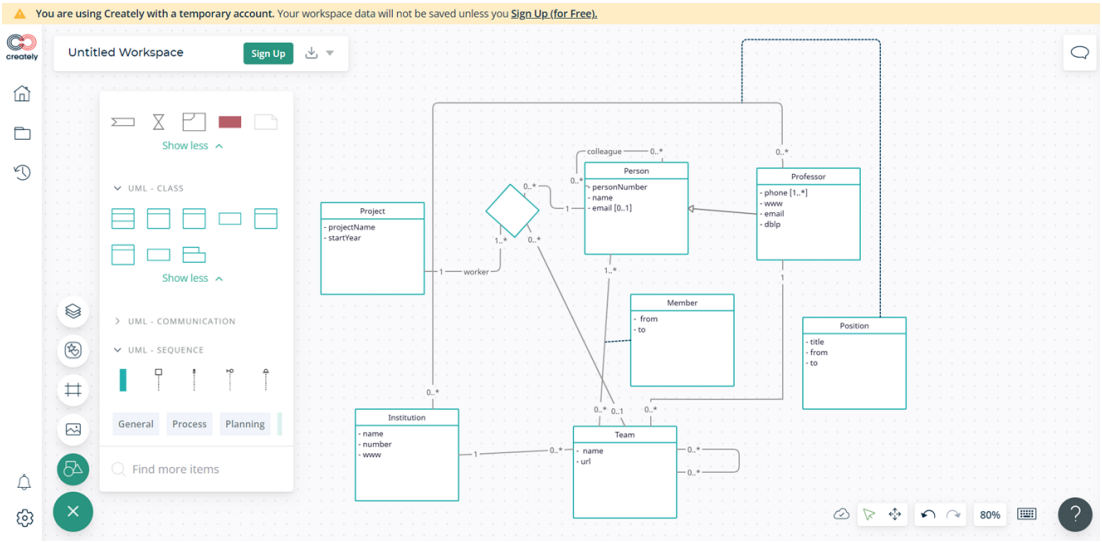
klikom vybrať danú možnosť z ponuky. Následne sa nám automaticky vytvorí textové pole v objekte triedy, kde môžeme napísať požadovaný text.

Nástroj ponúka undo-redo operácie, pridávanie nových strán a obsahuje množstvo ďalších kolekcií, ktoré je možné pridať po výbere z ponuky, zobrazenej po kliku na tlačidlo `Shapes`. Množstvo a obsah kolekcií je podobné ako u Diagrams.net.

Výsledný model môžeme uložiť ako obrázok v rôznych formátoch, napr. Joint Photographic Experts Group (JPEG), PNG alebo aj vo forme PDF súboru. Výsledný model je zobrazený na obrázku 3.5.

Prostredie je prehľadné, pomerne rýchlo sa v ňom dá naučiť pracovať, preto tento nástroj považujem za užitočný aj pre menej pokročilých používateľov.

3.3 Creately



Obr. 3.6: Používateľské prostredie Creately

Creately[4] je online nástroj podporujúci napríklad class modelovanie, diagramy a whiteboard, ktorý sa využíva písanie poznámok, brainstorming alebo aj edukáciu.

Pre prístup k online Creately nástroju je nutné sa prihlásiť alebo zaregistrovať. Ponúkaná je aj bezplatná verzia.

Po výbere UML z bohatej ponuky, sa nám zobrazí prostredie na modelovanie, zobrazené na obrázku 3.6. Zaujme nás moderný a čistý dizajn. Na výber z ponuky objektov slúži rozbaľovacie okno, ktoré sa zobrazí použitím tlačidla v ľavej dolnej časti prostredia.

V tomto nástroji, ako aj pri predošlých, je možný výber z rôznych kolekcií. Pri jednoduchom modelovaní postačí z UML iba ikona class. Class nakreslíme jednoduchým potiahnutím z okna a po dvoch klikoch na už vytvorenú class vieme zmeniť text a upraviť jeho vlastnosti. Asociácie a iné spojenia objektov môžeme kresliť po kliku na objekt, vybraní možnosti spojenia a následnom kliku na druhý objekt, s ktorým chceme spojenie vytvoriť. Nakreslená čiara má automaticky

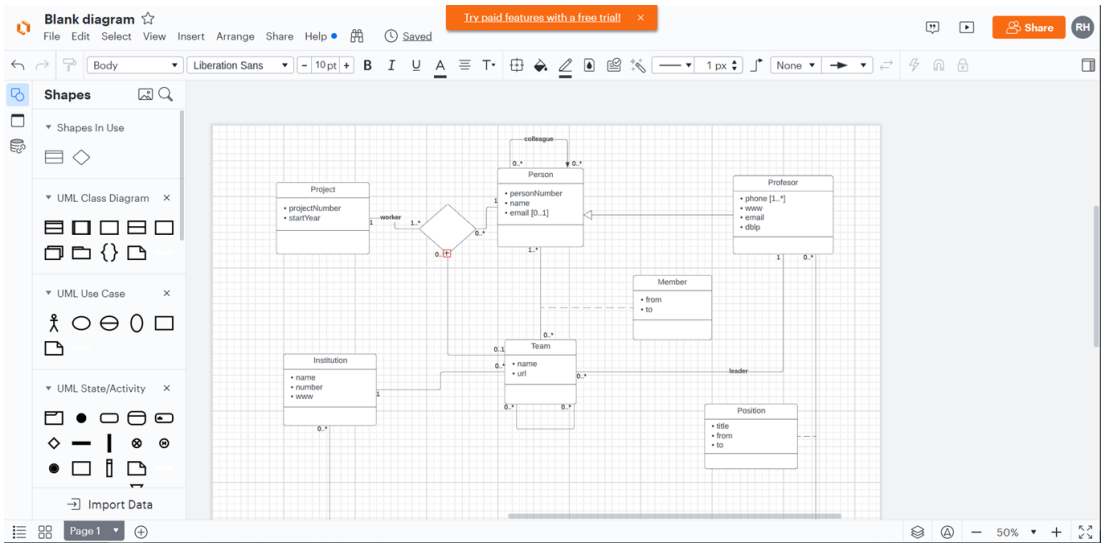
[4] https://creately.com/lp/uml-diagram-tool/

zakomponované multiplicity – nemusíme hľadať možnosť ako ich vytvoriť. Ich zmena je taktiež jednoduchá, stačí dvakrát kliknúť na požadovanú multiplicitu. Vlastnosti sa dajú zmeniť po kliku na daný objekt. U spojení je to napríklad tvar, farba, a následne aj jej kategória. Na výber z kategórií vzťahov je napríklad asociácia, agregácia alebo aj obyčajná čiara. Pri class sa dá zmeniť veľkosť a podfarbenie.

Samotný posun čiar, aby nepretínali iné objekty, je náročnejší. Pri rovných čiarach to nie je možné, pri zalomených sa dajú posúvať iba rovné celky zalomenej čiary. Väčší problém nastáva pri kreslení cyklu (asociácia triedy samej so sebou). Nástroj má problémy nakresliť cyklus spôsobom, aby bol viditeľný a dobre čitateľný. Nechýba tu kopírovanie, výber objektov z modelu a ich posúvanie. Samozrejmosťou sú undo-redo operácie, približovanie a aj klávesové skratky. Na druhej strane tu chýba napríklad n-árna asociácia.

Výsledný model môžeme uložiť v rôznych formátoch ako napríklad PDF, PNG a JPEG po zakúpení premium verzie. Prostredie je príjemné a užitočné aj pre menej skúsených používateľov.

3.4 LucidChart



Obr. 3.7: Používateľské prostredie LucidChart

LucidChart[5] je nástroj dostupný online, vytvorený spoločnosťou Lucid[6], prinášajúci nemalé množstvo funkcií a šablón ako napríklad taskboard, edukáciu, brainstorming a diagramy.

Do editoru sa potrebujeme prihlásiť, napr. pomocou Google účtu, vyplniť pár otázok a potom môžeme začať pracovať. V možnostiach musíme nastaviť, aby sa zobrazovali nástroje pre modelovanie UML.

Následne sa zobrazí používateľky prívetivé prostredie, zobrazené na obrázku 3.7, v ktorom sa dobre pracuje. Na ľavej strane sa nachádza panel s výberom z rôznych objektov, na vrchu sa nachádza lišta s vlastnosťami, ktoré je možné meniť.

[5] https://www.lucidchart.com/pages/examples/uml_diagram_tool
[6] https://lucid.co/about

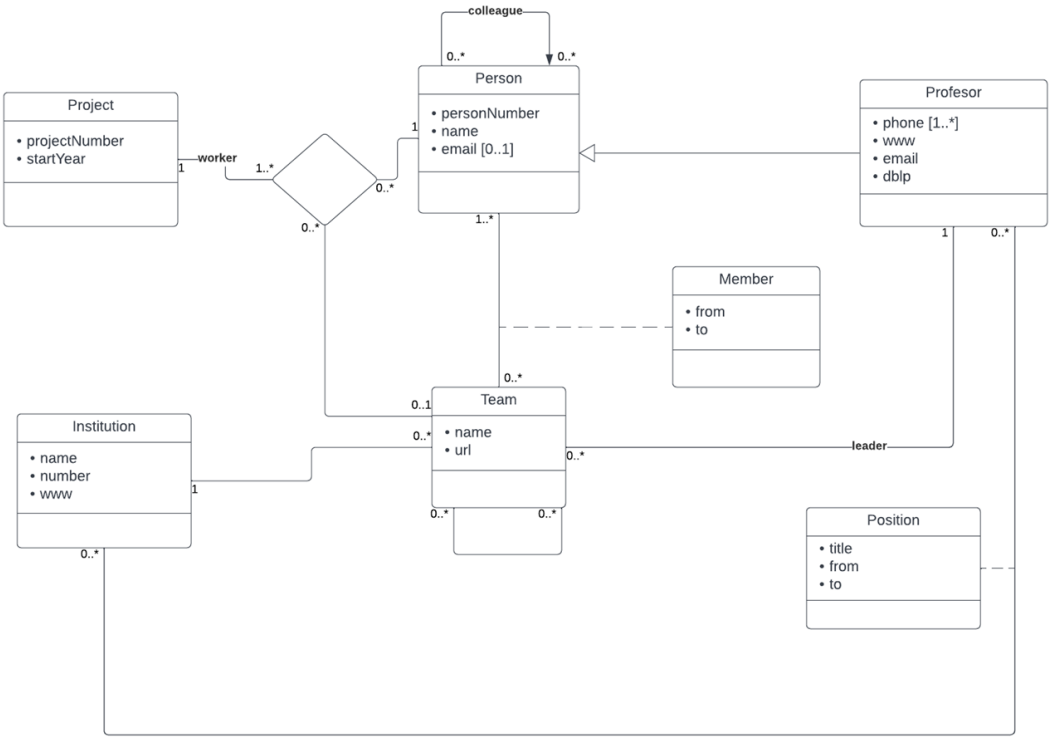
Pretiahnutím požadovaného objektu z bočného panelu sa vykreslí na canvas. Jednotlivé asociácie kreslíme po kliknutí na objekt a následnom spojení s druhým objektom. Na hornej lište si môžeme upraviť takmer všetky vlastnosti. Napríklad font písma, veľkosť, podfarbenie, štýl čiary, hrúbku a jej zakončenia. Samotné čiary môžeme ľubovoľne posúvať a pri vybranej čiare je možné po kliku na ozubené koliesko zvoliť možnosť aby sa zobrazili multiplicity.

Samozrejmosťou je možnosť výberu objektov, ich presun alebo kopírovanie. Pri presune sú asociácie naďalej prepojené s objektami, predlžujú sa a zalamujú podľa potreby. Dvojklikom na triedu môžeme zmeniť text a taktiež jeho vlastnosti. Podobne môžeme pridať názov pre asociácie.

Jednoducho môžeme zostrojiť aj n-árnu asociáciu, prepojenie objektov alebo aj cyklus triedy. Nechýba priblíženie plochy, undo-redo operácie, či mnohé iné pomôcky.

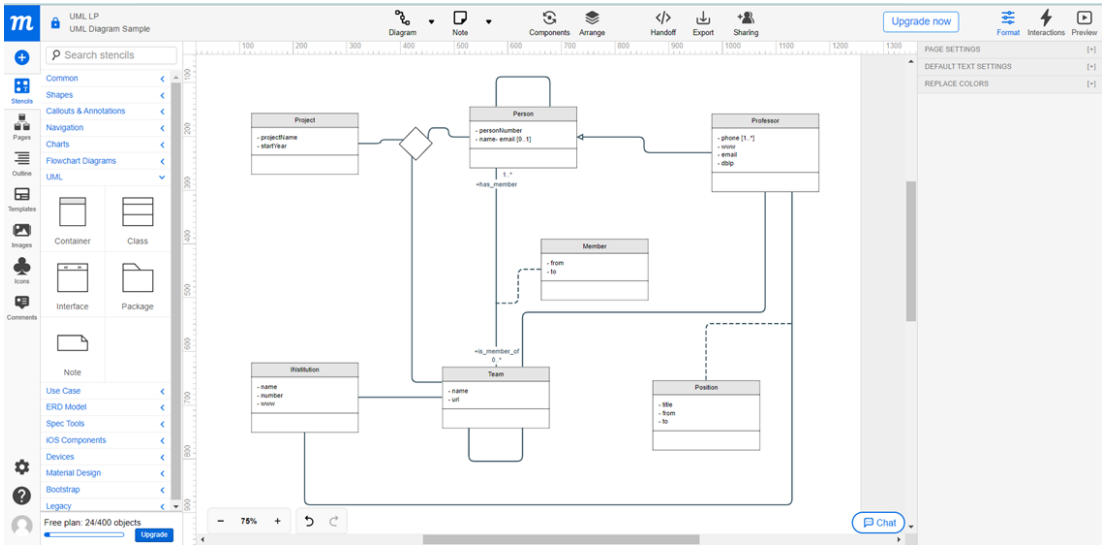
Výsledný model môžeme uložiť vo formátoch ako napríklad PDF, PNG a JPEG. Nástroj ponúka možnosť importovať model zo súboru v jednom z podporovaných formátov. Výsledný model je zobrazený na obrázku 3.8.

Prostredie je aj napriek veľkému množstvu možných nastavení prehľadné a naučiť sa s ním pracovať nezaberie veľa času. Odporúčal by som tento nástroj skôr pre pokročilých alebo pre profesionálnejších používateľov, pretože je možné nastaviť alebo upraviť takmer všetky vlastnosti objektov. Pre používateľov, ktorý sa ešte len zoznamujú s UML alebo sú v začiatkoch, by som skôr odporúčal jednoduchší nástroj.



Obr. 3.8: Výsledný UML model použitím editora LucidChart

3.5 Moqups



Obr. 3.9: Používateľské prostredie Moqups

Moqups[7] je online univerzálny nástroj s funkciami a šablónami ako napríklad modelovanie, vytváranie grafov alebo vizualizáciu biznis stratégie. Do nástroja sa musíme prihlásiť.

Pracovné prostredie, zobrazené na obrázku 3.9, pôsobí jednoducho, čisto, ale na oboch stranách sa nachádzajú panely. Panel na ľavej strane zobrazuje možnosti rôznych druhov modelovania. Pre nás je zaujímavá iba UML časť, ktorá obsahuje niekoľko základných objektov. Vykresľujú sa klikom na ikonu alebo presunutím požadovaného objektu na canvas. Následne sa na paneli na pravej strane objaví mnoho nástrojov, pomocou ktorých môžeme meniť vlastnosti ako napríklad podfarbenie, farbu textu, typ písma, hrúbku čiary či veľkosť alebo pozíciu objektu.

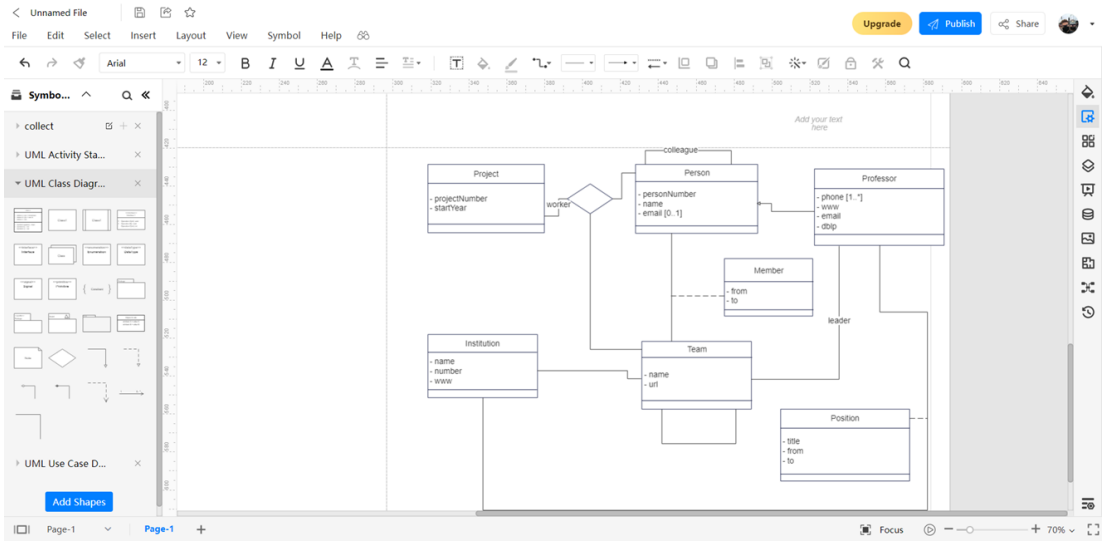
Jednotlivé asociácie môžeme kresliť pomocou tlačidla na hornej lište s názvom `Diagram`. Následne sa na objektoch objavia malé body, na ktorých je možné pripojiť asociáciu. Multiplicity a názov vieme jednoducho vložiť pomocou dvojkliku na asociáciu. Zmenu vlastností čiar môžeme uskutočniť pomocou panelu na prvej strane. Pre n-árne asociácie sa tu nenachádza priamo vytvorený objekt, preto si ju musíme vytvoriť pomocou po-otočeného štvorca. Do tried sa text vkladá a mení opäť dvojklikom na požadované miesto.

Uloženie modelov je prístupné po zakúpení premiun verzie. Celková funkčnosť je dobrá, ale priestor na prácu uberá množstvo panelov a vo výsledku nástroj pôsobí stiesnene a neprehľadne. Ak používateľ nepotrebuje vyberať nové objekty alebo meniť vlastnosti vytvorených objektov, je možné bočné panely skryť a tým sa zväčší priestor na prácu.

[7] https://moqups.com/templates/diagrams-flowcharts/uml-diagrams/

3.6 EdrawMax

EdrawMax[8] je online modelovací nástroj, do ktorého je nutné sa prihlásiť. Ponúka rôzne šablóny a funkcie ako napríklad modelovanie diagramov, brainstorming alebo zobrazovač office produktov.



Obr. 3.10: Používateľské prostredie EdrawMax

Po kliknutí na požadované modelovanie, pre nás to je UML modelovanie, sa otvorí nástroj a môžeme začať pracovať.

Jednofarebné čisté prostredie, zobrazené na obrázku 3.10, obsahuje hárky s canvasom, ktoré je možné pridávať a samozrejme aj zväčšovať či zmenšovať. Na ľavej strane sa nachádza panel s výberom objektov. V prípade záujmu používateľa je možné si do ľavého panelu pridať aj iné objekty z ponuky. Nechýbajú undo-redo operácie, možnosť zamknúť jednotlivé objekty na posun, či napríklad aj zmenu veľkosti.

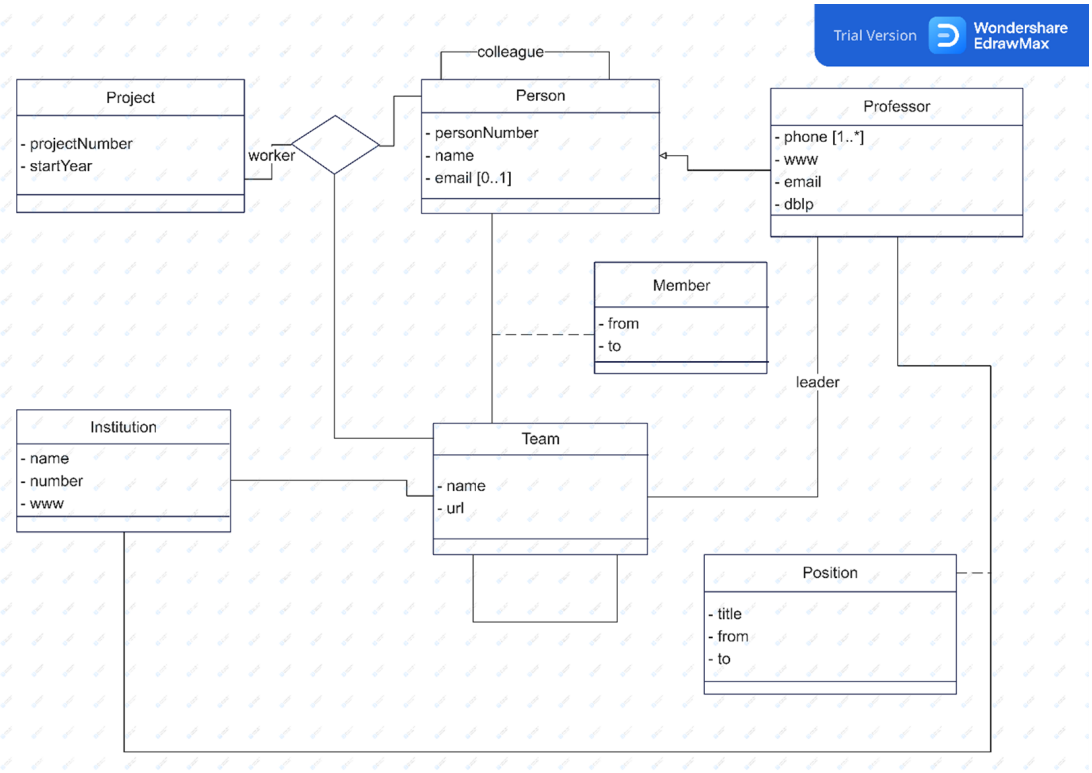
Objekty kreslíme preťahovaním z bočného panelu, samozrejmý je ich posun, kopírovanie a výber viacerých objektov naraz. Asociácie sa kreslia umiestnením kurzoru nad okraj objektu. Následne sa objaví malý červený kruh a je možné následne nakresliť čiaru stlačením a podržaním tlačidla. Môžeme nastavovať vlastností objektov ako napríklad výplň, farba čiary, font a veľkosť textu, štýl asociácie či jej zmena na agregáciu alebo dedičnosť. Multiplicity alebo názov asociácie je možné zadávať iba formou textového poľa na kresliacu plochu. Výhodou tohto editoru je aj objekt n-árnej asociácie. Pri nevhodnou nakreslení asociácie niekedy nastane situácia, v ktorej je obtiažne upraviť polohu tak, aby vyhovovala a zároveň vyzerala esteticky.

Text v textovom poli alebo objekte je možné meniť po dvojkliku na daný priestor. Pri posúvaní objektov sa vykonáva automatická úprava, t. j. predĺženie prepojení podľa novej polohy objektov.

Celkové hodnotenie práce v tomto editore je dobré, menšie problémy nastali pri vytváraní asociácií. Výslednú prácu je možné stiahnuť napríklad vo formáte Document (DOC), PowerPoint (PPTX) a PDF, ale je nutné zakúpenie prémiovej

[8] https://www.edrawmax.com/online/en/

verzie. Výsledok môžeme stiahnuť aj zadarmo napríklad vo forme obrázku s logom nástroja (obrázok 3.11).



Obr. 3.11: Výsledný UML model použitím editora EdrawMax

3.7 Porovnanie testovaných editorov

Nasledujúca tabuľka 3.1 zobrazuje v stĺpcoch jednotlivé editory a v riadkoch vlastnosti, na základe ktorých sme ich medzi sebou porovnávali.

Výsledky sú zaznačené „áno“, slovným popisom alebo znakom „-“, ktorý znamená, že danú vlastnosť alebo informáciu editor nepodporuje, neobsahuje alebo ju nebolo možné zistiť.

Tabuľka 3.1: Tabuľka porovnania jednotlivých editorov na základe vybraných vlastností

	Diagrams.net	**Visual Paradigm**	**Creately**	**LucidChart**	**Moqups**	**EdrawMax**
Distribúcia	Online/desktopová aplikácia	Online/desktopová aplikácia	Online/desktopová aplikácia	online	online	Online/desktopová aplikácia
Export	XML, PNG, HTML, SVG	PNG, JPEG, SVG, PDF	Spoplatnené	PNG, PDF, JPEG, SVG, CSV, (spoplatnené - VSDX, VDX)	Spoplatnené	PNG, JPEG, PDF, ostatné sú spoplatnené
Import	Je možné importovať už vytvorený diagram	Po prihlásení	Import obrázkov na canvas	Možný XML súborov, výsledok nemusí byť prijateľný	-	EDDX, VSDX, VSD
Formát	Formát súboru je možné meniť a znovu uložiť	Súbory nie je možné medzi sebou prevádzať	-	-	-	-
Licencia	Bezplatná	Spoplatnená	Spoplatnená	Spoplatnená	Spoplatnená	Spoplatnená
Open source	áno	-	-	-	-	-
Vyvinutý v	JavaScript, HTML	-	-	-	-	-
Plugin a rozšíriteľnosť	áno	-	-	Možnosť pridať vlastné tvary na modelovanie	-	-
Čo je možné modelovať	napr.: UML, siete, pôdorysy, logické hradlá	napr.: UML, siete, mapa mysle, logické hradlá	napr.: UML, plán ovanie, chemické laboratórium (niektoré sú spoplatnené)	napr.: UML, siete, pôdorysy, Vennove diagramy	napr.: UML, časový diagram, Vennove diagramy	napr.: UML, pôdorys, elektrické okruhy, mapa mysle
Prepojenie s úložiskom	áno	áno (Google Drive)	áno	áno	áno	-
Spolupráca užívateľov	Áno ak pracujú na zdieľanom dokumente	Áno, zdieľanie dokumentov je možné	áno	áno	áno	Pomocou zdieľaného odkazu
Verzovanie modelov	áno	-	-	Pomocou revison history (spoplatnené)	-	-
Editov v českom jazyku	áno	áno	-	-	-	-
Zablokovanie objektov	celý objekt	celý objekt	celý objekt	celý objekt alebo výber z možností	celý objekt	výber z možností
Pridávanie komentárov	-	-	áno	áno	áno	áno

Kapitola 4

Programátorská dokumentácia

V tejto kapitole sa nachádza samotná programátorská dokumentácia popisujúca zdrojové súbory programu, ich verejné triedy a metódy. Na zjednodušenie sú prípady použitia používateľom zobrazenú v diagrame (sekcia 4.2), z ktorých niektoré sú popísané slovne v sekcii 4.10 a niektoré taktiež vizualizované prostredníctvom UML sekvenčných diagramov v prílohe A.5. Požiadavky kladené na program sú dostupné v úvode bakalárskej práce (sekcia 0.1).

4.1 Vývojové prostredie, programovací jazyk

Program **UMLtoCATeditor** bol vytvorený vo vývojovom prostredí Visual Studio 2022[1] na platforme .Net 6[2]. Platforma podporuje formulárové aplikácie a prácu s grafickým prostredím. Ako programovací jazyk som zvolil C#[3]. Program taktiež používa externú knižnicu Json.NET[4].

4.2 Diagram prípadov použitia

Nasledujúci diagram na obrázku 4.1 zobrazuje prípady použitia editoru používateľom. V bublinách, prepojených s používateľom jednoduchou čiarou, sú prípady použitia, ktoré môže v editore sám vykonať. Prerušovanou čiarou s textom `<<include>>` sú prepojené akcie, ktoré sú vykonané v rámci prípadu použitia, resp. inej akcie. Naviac v prílohe A.5 sú zobrazené sekvenčné diagramy pre vybrané prípady použitia používateľom.

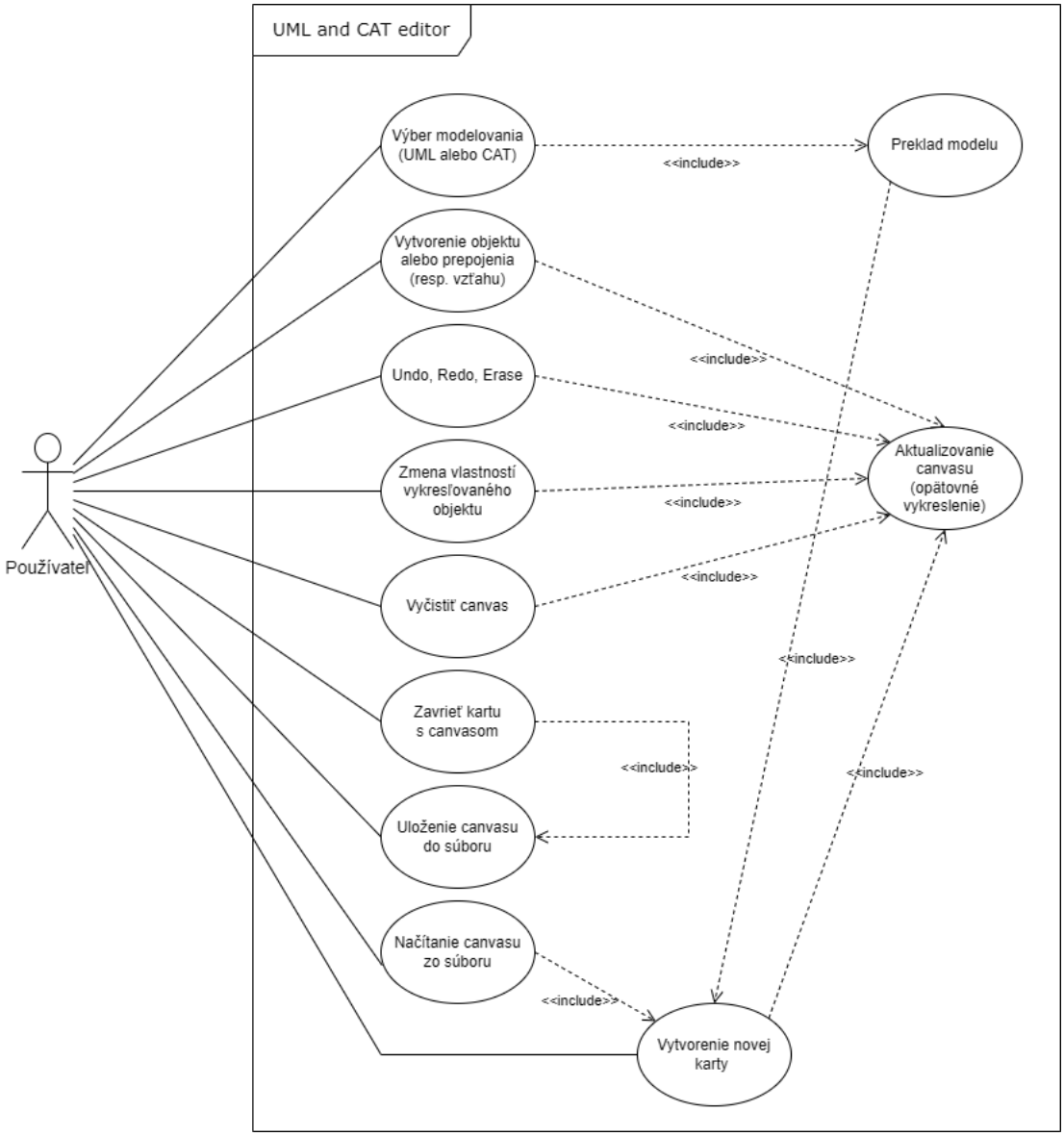
4.3 Zdrojové súbory

Aplikácia je rozdelená do šiestich súborov, ktoré sú popísané nižšie, s názvami:

- Program.cs (sekcia 4.4),
- uml_to_cat_widget.cs (sekcia 4.5),

[1] https://visualstudio.microsoft.com/
[2] https://dotnet.microsoft.com/en-us/
[3] https://learn.microsoft.com/en-us/dotnet/csharp/
[4] https://www.newtonsoft.com/json

- Form1.cs (sekcia 4.6),
- UMLandCATtranslator.cs (sekcia 4.7),
- UndoRedoInterface.cs (sekcia 4.8),
- UndoRedoCommands.cs (sekcia 4.9).



Obr. 4.1: Diagram prípadov použitia editoru

4.4 Program.cs

Obsahuje `Main()`, triedu na uloženie otvorených kariet jednotlivých canvasov, vytvorenie bodov na prepojenie vykreslených objektov, pomocné metódy.

4.4.1 PositionOfPoint

Enum, ktorý poskytuje informáciu o polohe bodu na pripájanie vzťahov, z dôvodu zlepšenia grafického výstupu aplikácie.

4.4.2 UML, CAT, BASIC

Interfacy slúžiace na overenie, do ktorého modelovania patrí daný objekt, na základe jeho typu.

`UML` a `CAT` objekty patria iba svojmu modelovaniu, objekt typu `BASIC` môže byť v oboch. Pod interface `BASIC` patrí napríklad textové pole.

4.4.3 OpenedCanvas

Trieda na uloženie otvorenej karty canvasu a k tomu potrebných objektov.

Obsahuje parametre `undoStack` a `redoStack`, v ktoré obsahujú jednotlivé undo a redo zásobníky daného canvasu, list `ObjectsOnCanvas` všetkých objektov vykresľovaných na canvase, reťazec `filePath` s uloženou adresou otvoreného súboru, hodnotu `modelingUML`, ktorá je nastavená na `true`, ak sa na canvase modeluje v UML a `myTabPage` odkazujúcu na kartu daného canvasu vytvorenú C#.

Obsahuje metódy `getFilePath` na získanie a `setFilePAth` na nastavenie cesty k súboru, `isModelingUML` na overenie modelovania v UML, nastavenie modelovania na UML alebo CAT prostredníctvom metód `modelingModeUML` a `modelingModeCAT`.

4.4.4 PointOfObjectForConnection

Trieda na vytvorenie bodu na objekte, ktorý je možné prostredníctvom vzťahov prepájať s inými objektami.

Trieda neobsahuje žiadnu metódu.

Obsahuje parametre `x` a `y`, ktoré určujú počiatočný bod (jeho x-ové a y-ové súradnice), kde sa bude bod vykresľovať. Keďže daný bod sa vykresľuje ako štvorec, postačuje nám iba parameter `width`. Ďalej obsahuje referenciu na objekt `onObject`, na ktorom bode bod vykreslený, farbu výplne `fillColor` a parametre `correctionX` a `correctionY`, ktoré sa využijú pri konštrukcii spojenia tak, aby sa oba koncové body čiary, s výslednými súradnicami

$$(lineX,lineY) = (correctionX + x, correctionY + y),$$

dotýkali daných objektov.

Veľkosť bodu je prednastavená na 7x7 a farba výplne na `OrangeRed`.

4.4.5 MyHelpers

Trieda obsahuje iba pomocné metódy na výpočet uhlu priamky, otočenie objektu podľa vypočítaného uhlu a overenie, či list obsahuje iba povolené prvky pre dané modelovanie.

Metóda `isListOfObjectsToDrawOnCanvasModelingUML` overí správnosť prvkov v liste na základe interface zo sekcie 4.4.2, v prípade chyby nastane vyhodenie

výnimky `ArgumentException`. Pre získanie otočeného objektu je potrebné využiť metódu `GetRotatedObjectDependentOnStartAndEndOfConnection`,

4.5 uml_to_cat_widget.cs

Obsahuje jednotlivé triedy a metódy, prostredníctvom ktorých sú zadefinované objekty, ktoré sa vykresľujú na canvas. Všetky triedy sú priamym alebo nepriamym potomkom triedy `EveryObjectOnCanvas`.

Návrhový vzor tried v tomto súbore sa nachádza v prílohe A.3.

4.5.1 EveryObjectOnCanvas

Z tejto triedy vychádzajú zvyšné triedy v tomto súbore a teda všetky objekty na plátne.

Obsahuje parameter `width` určujúci šírku objektu, `lineColor`, ktorý je prednastavený na čiernu farbu, `boldText` a `italicText`, ktoré upravujú font textu ak sú nastavené na hodnotu `true`. Z konštruktoru sú prednastavené na hodnotu `false`.

Trieda má metódy `NegateBoldText` a `NegateItalicText`, ktoré menia hodnoty `boldText` a `italicText` na opačné. Využitie metód je pri stláčaní tlačidiel na úpravu fontu textu. Ďalej obsahuje virtuálne metódy `ChangeWidth` upravujúcu šírku objektu, `AddConnection` a `RemoveConnection` na pridanie alebo odstránenie prepojenia daného objektu.

4.5.2 BoxObjectsOnCanvas

Je priamym potomkom triedy `EveryObjectOnCanvas` a zároveň predkom pre všetky objekty na plátne, ktoré majú tvar boxu (napríklad štvorec alebo obdĺžnik).

Na správne vykresľovanie a určenie polohy obsahuje parametre `x` a `y` určujúce absolútnu polohu ľavého horného bodu daného objektu, výška objektu je uložená v parametri `height`.

Keďže tvar objektu je box, trieda obsahuje aj parameter `backgroundColor` na nastavenie podfarbenia, ktorá je prednastavená na bielu farbu.

Trieda obsahuje iba virtuálne metódy na zmenu pozície objektu podľa polohy kurzora na plátne (`ChangePosition`), zmenu šírky a výšky, taktiež podľa polohy kurzora alebo podľa zadaného čísla (`ChangeWidth`, `ChangeHeight`).

4.5.3 BoxObjectsWithConnections

Potomok triedy `BoxObjectsOnCanvas`, ktorý je rovnaký až na rozdiel, že tento objekt je možné prostredníctvom bodov spájať s inými objektami na canvase, ktoré sú taktiež potomkovia tejto triedy.

Obsahuje list `pointsForConnection` objektov triedy `PointOfObjectForConnection` určujúcich body na prepojenie objektov a list `connectionsOfBox` objektov triedy `ConnectionsOnCanvas`, ktorý bude obsahovať všetky spojenia daného boxu.

Na výpočet polôh jednotlivých bodov na spojenia sa používa virtuálna metóda `CalculatePositionsOfPointsForConnection`, ktorá vytvorí 4 body na spojenia. Uprostred každej strany boxu je jeden bod. Pri vytváraní spojení sa využívajú metódy `AddConnection`, ktorá pridá spojenie do listu `connectionsOfBox` a `RemoveConnection`, ktorá z daného listu spojenie naopak vymaže.

Metóda `CalculatePositionOfPointForAttributeOnAssociation` vypočíta polohu bodu na asociáciách boxu na pridanie asociačnej triedy
a `ModifierOrMovedBox` ktorá zavolá funkcie na opätovné určenie polohy bodov na asociáciách a na danom boxu z dôvodu jeho posunu alebo zmeny veľkosti.

Na odstránenie všetkých spojení daného boxu slúži virtuálna metóda `RemoveAllConnectionsFromConnectedBoxes`.

4.5.4 UMLClassComponents

Táto trieda, ktorá je potomkom triedy `BoxObjectsOnCanvas`, vytvára jednotlivé časti UML tried a to: pole s názvom, atribútmi a metódami. Trieda `UMLclass` obsahuje 3 inštancie tejto triedy.

Na určenie polohy boxu používa parametre `x` a `y`, ktoré sú **relatívne** voči súradniciam `x` a `y` triedy `UMLClass`, v ktorej táto inštancia existuje. Referencia na danú triedu, kde existuje má názov `childOfClass`. Ďalej obsahuje parameter typu `textInside`, ktorý obsahuje text zobrazovaný vo vnútri tohto boxu.

Trieda obsahuje metódu `ChangeHeight`, ktorá zmení výšku a následne zavolá svoju nadriadenú UML triedu, aby zmenila svoju výšku a `ChangeWidth`, ktorá zavolá nadriadenú UML triedu na zmenu šírky. Šírka boxu sa nastavuje podľa šírky `childOfClass`, výška je prednastavená manuálne a text vo vnútri podľa určenia daného boxu v `UMLClass`.

4.5.5 UMLClass

Potomok triedy `BoxObjectsWithConnections` a inteface `UML` obsahujúci referencie na objekty `UMLClassComponents` reprezentujúce meno, atribúty a metódy triedy s príslušnými názvami `nameOfClass`, `attributesOfClass` a `methodsOfClass`.

Trieda obsahuje metódy `ChangeWidth` a `ChangeHeight` na zmenu šírky a výšky, podľa čísla alebo polohy kurzora, s upravením rozmerov jednotlivých častí triedy, `AddToHeight` na rovnomerné zväčšenie všetkých častí triedy o danú veľkosť a taktiež `ChangePosition` na zmenu polohy podľa pozície kurzora.

4.5.6 UMLNaryAssociation

Trieda je potomkom triedy `BoxObjectsWithConnections`, inteface `UML` a neobsahuje žiadne ďalšie parametre.

Obsahuje metódy `ChangeHeight` a `ChangeWidth` na zmenu šírky resp. výšky objektu podľa zadaného čísla alebo polohy kurzora. Keďže n-árna asociácia má tvar kosoštvorca (strany sú rovnako dlhé), tak nezáleží na tom, ktorá metóda bude zavolaná, pretože vždy sa zmenia obe hodnoty naraz. Na získanie bodov, podľa ktorých sa vykresľuje objekt na canvas slúži metóda `GetPointsForDrawingRhombus`.

4.5.7 CATArrowOptions

Enum, ktorý slúži na výber tagu pre CAT prepojenie. Podrobnejší popis o tagoch je v sekcii 2.2.

4.5.8 CATBoxObject

Trieda je potomok triedy `BoxObjectsWithConnections` a inteface `CAT`, obsahuje reťazec textu `textInside`, ktorý sa vykresľuje v jej vnútri, `TextArea identifier`, ktorým je identifikovaná a `HashSet possibleIdentifiers` všetkých možných identifikátorov pre daný objekt.

Obsahuje metódy `ChangeWidth` a `ChangeHeight` na zmenu šírky a výšky podľa zadaného čísla, zmenu polohy objektu na základe polohy kurzora prostredníctvom `ChangePosition`, `ChangeIdentifierPosition` na zmenu polohy identifikátorov, aby boli aj naďalej pod objektom, podľa súradníc objektu.

Na nastavenie všetkých možných identifikátorov objektu slúži `SetPossibleIdentifiers` a na nastavenie identifikátoru objektu, zadaného používateľom, po overení jeho správnosti, slúži metóda `setIdentifier`.

4.5.9 ConnectionsOnCanvas

Trieda je potomkom triedy `EveryObjectOnCanvas`. Obsahuje parametre `startPoint` a `endPoint` triedy `PointOfObjectForConnection` na uloženie bodov, kde dané prepojenie začína a končí. Má funkciu rodiča pre všetky ostatné druhy prepojení.

Obsahuje metódu `ClickedOnConnection` na overenie, či bolo na dané prepojenie kliknuté, `RemoveConnectionFromBoxes`, ktorá odstráni dané prepojenie z oboch boxov, ktoré spája. Metódy `RemoveConnectionFromConnectedBox` a `AddConnectionToDisconnectedBox`, ktoré odstránenia resp. pridajú prepojenie do boxu na opačnej strane prepojenia, než je zadaný box v argumente. Ďalej obsahuje virtuálnu metódu `getPointsForObjectOnSpecificConnection`, ktorá by mala v potomkoch triedy vrátiť body, podľa ktorých sa bude na koncovom bode prepojenia vykresľovať geometrický útvar.

4.5.10 UMLAssociation

Trieda je potomkom triedy `ConnectionsOnCanvas` a interface `UML`. Obsahuje `startMultiplicity` a `endMultiplicity`, ktoré obsahujú multiplicity špecifikované používateľom na koncoch vzťahu, `name` obsahujúce meno vzťahu. Ďalší parameter je `pointForAttributes` obsahujúci bod uprostred vzťahu na pripojenie UML asociačnej triedy a `attributeObjectConnection`, v ktorom je uložená prípadná asociačná trieda daného vzťahu.

Obsahuje metódy `SetStartMultiplicity`, `SetEndMultiplicity` a `SetName`, ktoré nastavia požadovaný parameter, `CalculatePositionOfPointForAttribute` prepočítavajúcu polohu bodu uprostred vzťahu na pripojenie asociačnej triedy. Taktiež trieda obsahuje prepísané triedy `AddConnection` a `RemoveConnection` na pridanie resp. odstránenie asociačnej triedy, ktorá môže byť iba jedna.

4.5.11 UMLAtribute

Trieda je potomkom triedy `ConnectionsOnCanvas` a interface `UML`.

Neobsahuje žiadne ďalšie pridané parametre alebo metódy.

4.5.12 UMLInheritance

Trieda je potomkom triedy `ConnectionsOnCanvas` a interface `UML`.

Neobsahuje žiadne ďalšie parametre, obsahuje prepísanú metódu `getPointsForObjectOnSpecificConnection`, vracajúcu pole bodov, podľa ktorých sa bude vykresľovať objekt (prázdny trojuholník) na konci vzťahu.

4.5.13 UMLAgregation

Trieda je potomkom triedy `UMLAssociation`.

Neobsahuje žiadne ďalšie parametre, obsahuje prepísanú metódu `getPointsForObjectOnSpecificConnection`, vracajúcu pole bodov, podľa ktorých sa bude vykresľovať objekt (prázdny diamant) na konci vzťahu.

4.5.14 UMLComposition

Trieda je potomkom triedy `UMLAssociation`.

Neobsahuje žiadne ďalšie parametre, obsahuje prepísanú metódu `getPointsForObjectOnSpecificConnection`, vracajúcu pole bodov, podľa ktorých sa bude vykresľovať objekt (plný diamant) na konci vzťahu.

4.5.15 CATConnectionArrow

Trieda je potomkom triedy `ConnectionsOnCanvas` a interface `CAT`.

Obsahuje parameter `arrowSpecification` obsahujúci tag daného vzťahu, prepísanú metódu `getPointsForObjectOnSpecificConnection`, vracajúcu pole bodov, podľa ktorých sa bude vykresľovať objekt (plný trojuholník) na konci vzťahu a `getTextAreaForArrowSpecification`, ktorá vráti vhodné miesto na vykreslenie tagu vzťahu na canvas.

4.5.16 TextArea

Potomok triedy `BoxObjectsOnCanvas` a interface `BASIC`. Obsahuje reťazec `textInside`, ktorý obsahuje text vo vnútri textového poľa, neobsahuje žiadne metódy.

4.6 Forms1.cs

Funkčná časť aplikácie, ktorá obsahuje jednotlivé metódy na overovanie, spracovávanie a vykresľovanie jednotlivých informácií.

4.7 UMLandCATtranslator.cs

Súbor obsahujúci triedu `UMLandCATtranslator` podporujúcu algoritmus prekladu UML modelu na CAT model.

Preklad sa uskutoční zavolaním metódy `TrasnlateFromUMLToCAT` s atribútom typu `List<EveryObjectOnCanvas>` obsahujúcim objekty na canvase, ktoré sú potomkovia interface `UML` alebo `BASIC`.

4.7.1 Popis algoritmu prekladu

Algoritmus prekladu UML schéma do CAT schéma funguje na základe prehľadávania modelu do hĺbky, t. j. Depth-First Search (DFS). Na rozdiel od prehľadávania do šírky, t. j. Breadth-First Search (BFS), má menšie nároky na pamäť, ale na druhej strane nemusí nájsť optimálne riešenie. V našom prípade, kde máme konečný graf a každý vrchol musíme navštíviť, nájdenie optimálneho riešenia nie je podstatné, a z toho dôvodu je algoritmus používajúci DFS vhodnou voľbou.

Pod vrcholom si v danom kontexte predstavujeme box vykresľovaný na canvas s možnosť jeho prepojenia s inými boxami. V tomto editore to sú: UML trieda (trieda `UMLClass`) a UML n-árna asociácia (`UMLNaryAssociation`), ktoré sú potomkami triedy `BoxObjectsWithConnections`.

Zo všetkých objektov v liste z atribútu metódy `TrasnlateFromUMLToCAT` sa vyberú objekty typu `BoxObjectsWithConnections` a na tomto výbere sa spustí algoritmus prekladu `UMLtoCATrecursionDFS`. Samozrejmosťou je uloženie si už navštívených vrcholov a ich prekladu, pre prípadnú potrebu vytvorenia prepojenia medzi vrcholmi, z ktorých jeden už bol navštívený a preložený. Po preložení objektu pokračujeme v prehľadávaní grafu a v preklade jednotlivých prepojení. V prípade UML triedy ešte pred samotným pokračovaním v prehľadávaní grafu preložíme atribúty danej triedy a pripojíme ich k prekladu triedy. Pri preklade sa zachováva grafická úprava ako napríklad podfarbenie a štýl textu. Jednotlivé UML vzťahy sa prekladajú na CAT prepojenia na základe popisu v časti 2.2.

4.8 UndoRedoInterface.cs

Obsahuje interface `UndoRedoInterface`, ktorý musia jednotlivé objekty v undo a redo zásobníkoch implementovať s požadovanými metódami `Undo` a `Redo`.

4.9 UndoRedoCommands.cs

Obsahuje vytvorené triedy implementujúce `UndoRedoInterface`, ktoré reprezentujú jednotlivé zmeny na canvase. Pridávajú sa do undo a redo zásobníkov.

Zoznam vytvorených tried s popisom príkazov, ktoré implementujú:

- `CreateBoxObjectOnCanvasCommand` – vytvorenie nového box objektu na canvase,
- `CreateConnectionOnCanvasCommand` – vytvorenie nového prepojenia na canvase,

- `LineColorChangeCommand` – zmena farby čiary akéhokoľvek objektu na canvase,
- `BackgroundColorChangeCommand` – zmena farby výplne box objektu na canvase,
- `PositionChangeCommand` – zmena polohy box objektu na canvase,
- `ItalicBoldTextCommand` – zmena štýlu fontu akéhokoľvek objektu na canvase,
- `WidthHeightChangeCommand` – zmena šírky alebo výšky akéhokoľvek objektu na canvase,
- `TextChangeCommand` – zmena textu box objektu na canvase,
- `DeleteBoxObjectOnCanvasCommand` – vymazanie objektu z canvasu,
- `DeleteConnectionOnCanvasCommand` – vymyzanie prepojenia z canvasu.

4.10 Postup pri vykonávaní príkazov

Táto sekcia obsahuje niektoré z najdôležitejších príkazov, ktoré sú popísané podrobnejšie. V prílohe A.5 sú sekvenčné diagramy zobrazujúce priebeh vykonávania určitých príkazov.

4.10.1 Nový canvas

Nový canvas sa vytvorí po kliknutí na položku `New Canvas` v ponuke menu alebo po stlačení kláves `ctrl+T`.

Oba spôsoby zavolajú metódu `AddNewCanvas()`, ktorá vytvorí kartu typu`TabPage` s unikátnym názvom, vytvorí nový canvas prepojený s danou `TabPage`, pridá nový canvas do listu otvorených canvasov, kartu pridá do lišty, nastaví vybranú kartu na novo vytvorenú a zmení canvas zavolaním metódy `ChangeActualWorkingCanvas(OpenedCanvas canvas)`

4.10.2 Zatvorenie canvasu

Po kliknutí na tlačilo `Close Canvas` z ponuky v menu alebo po stlačení kláves `ctrl+W` sa zavolá metóda `closeCanvasToolStripMenuItem_Click(object sender, EventArgs e)`, ktorá otvorí dialógové okno s otázkou, či si želá používateľ uložiť canvas. Ak áno, zavolá sa metóda `saveCanvasIntoJSONFIle(object sender, EventArgs e)` ktorá canvas uloží, ak aktuálny canvas už bol uložený, t. j. obsahuje cestu k súboru. V opačnom prípade sa zavolá rovnaká funkcia ako pri príkaze 4.10.6.

4.10.3 Zmena canvasu

Zmena canvasu sa vykoná zavolaním metódy `ChangeActualWorkingCanvas(OpenedCanvas canvas)`, ktorá nastaví premenné podľa požadovaného canvasu, t. j. uloženie požadovaného canvasu, načítanie objektov vykresľovaných na canvas, načítanie druhu modelovania, uloženie undo a redo zásobníkov. Ďalej resetuje všetky tlačidlá a premenné používané pri modelovaní prostredníctvom metódy `ResetAllButtonsVariables()`, nastaví modelovací mód zavolaním metódy `SetModelingMode()` a nakoniec prekreslí canvas.

4.10.4 Vyčistenie canvasu

Nastane po kliknutí na tlačidlo `Clear Canvas` v ponuke menu. Resetujú sa všetky tlačidlá a premenné používané pri modelovaní prostredníctvom metódy `ResetAllButtonsVariables()`, vymažú sa undo a redo zásobníky, prekreslí sa canvas.

4.10.5 Otvorenie — načítanie súboru

Načítanie súboru nastane po stlačení kláves `ctrl+O` alebo po kliknutí na `Open file` v ponuke menu. Zavolá sa metóda `openFileToolStripMenuItem_Click(object sender, EventArgs e)`, ktorá otvorí systémové dialógové okno na výber požadovaného súboru na otvorenie, následne sa pokúsi načítať model zo súboru, overí jeho korektnosť a v prípade správnosti otvorí vytvorí nový canvas, a nakreslí načítaný model.

V prípade chyby sa zobrazí varovná hláška.

4.10.6 Uloženie súboru

Stlačením kláves `ctrl+S` sa ukladá súbor rovnakým spôsobom ako v časti 4.10.2 po odsúhlasení, že si používateľ želá uložiť súbor.

Inak sa súbor ukladá prostredníctvom metódy `saveAsToolStripMenuItem_Click(object sender, EventArgs e)`, ktorá sa taktiež zavolá po stlačení tlačidla `Save as...` v ponuke menu. Používateľovi sa zobrazí systémové dialógové okno na výber umiestnenia a názvu súboru. Následne sa uloží cesta k súboru a zavolá sa metóda `saveCanvasIntoJSONFIle(OpenedCanvas canvas)`, ktorá súbor uloží na požadovanej ceste.

V prípade chyby sa zobrazí varovná hláška. Ukážka vygenerovaného súboru vo formáte JSON sa nachádza v prílohe A.6.

4.10.7 Zatvorenie editoru

Zatvorenie editoru nastane po stlačení tlačidla `Exit` v ponuke menu. Zavolá sa funkcia `exitToolStripMenuItem_Click(object sender, EventArgs e)`, ktorá sa najprv používateľa opýta, či si je istý, že chce zatvoriť editor. Ak áno, následne sa ho opýta, či chce uložiť otvorené canvasy. Ak používateľ odpovie kladne, pre každý otvorený canvas zavolá metódu

`saveCanvasIntoJSONFIle(OpenedCanvas canvas)`, ktorá canvas uloží na požadovanú cestu. V prípade, že cesta nie je špecifikovaná, otvorí sa systémové dialógové okno, v ktorom používateľ vyberie umiestnenie a názov súboru.

4.10.8 Vykresľovanie objektov na canvas

Vykresľovanie prebieha prostredníctvom metódy `panelForDrawing_Paint(object sender, PaintEventArgs e)`, ktorá vykreslí všetky objekty v liste `objectsToDrawOnCanvas` na canvas. Podľa typu objektu sa zvolí požadovaná vetva a objekt sa vykreslí s jeho uloženými vlastnosťami ako napríklad vnútorný text, štýl textu alebo farba výplne. Ak je daný objekt používateľom vybraný, jeho farba obrysov bude tyrkysová. Následne sa podľa zakliknutého tlačidla na vytváranie prepojení vykreslia na objektoch triedy `BoxObjectsWithConnections` body, na ktoré je možné dané prepojenie pripojiť.

Pri vykresľovaní bodov na pripojenie prepojenia sú aj obmedzenia ako napríklad: asociačnú triedu môžeme pripojiť iba na asociáciu, a preto sa body vykreslia iba na asociácii a UML triedach, ktoré môžeme prepojiť.

4.10.9 Vytvorenie nového objektu

Vytvorenie nového objektu prebieha po stlačení príslušného tlačidla. Vytvorí sa nový objekt, pridá sa do list vykresľovaných objektov a prekreslí sa canvas.

Pri vytváraní prepojení zostáva zvolené tlačidlo stlačené, zobrazia sa určené body na vytváranie prepojení. Nutné je vybrať dva body alebo znovu kliknúť na požadované tlačidlo prepojenia, aby sa výber bodov zrušil. Po kliku na požadovaný bod sa bod uloží prostredníctvom metódy `AddClickedPointForConnection(PointOfObjectForConnection point)`, po kliku na druhý bod nastane vytváranie prepojenia podľa stlačeného tlačidla. Prepojenie sa pridá do listu vykresľovaných objektov a zavolá sa metóda `StopProcessOfConnectingObjects()`, ktorá vráti pomocné premenné do pôvodného stavu a pripraví ich na vytváranie nového prepojenia.

4.10.10 Undo, Redo, Erase

Po kliku na tlačidlo `Undo` alebo stlačení kláves `ctrl+Z` sa zavolá metóda `undoButton_Click(object sender, EventArgs e)`, ktorá z existujúceho zásobníku vyberie posledný prvok typu `UndoRedoInterface` a zavolá jeho `Undo()` metódu, ktorá vráti vykonané úpravy, pridá objekt do redo zásobníku a prekreslí canvas i pravý panel.

Po kliku na tlačidlo `Redo` alebo stlačení kláves `ctrl+Y` sa zavolá metóda `redoButton_Click(object sender, EventArgs e)`, ktorá z existujúceho redo zásobníku vyberie posledný objekt, zavolá jeho `Redo()` metódu, ktorá vykoná úpravy, pridá objekt do undo zásobníku a prekreslí canvas i pravý panel. Redo zásobník sa vyčistí v metóde `addToUndoStack(UndoRedoInterface command)` pri pridaní nového objektu do undo zásobníku (t. j. nie pri vykonávaní undo a redo operácií).

Po kliku na tlačidlo `Erase` alebo stlačení kláves `ctrl+D` sa zavolá funkcia `eraseButton_Click(object sender, EventArgs e)`, ktorá podľa typu objektu

vykoná potrebné úkony na jeho bezpečné odstránenie ako napríklad: odstráni všetky jeho prepojenia alebo odstráni jeho textové polia. Nakoniec prekreslí canvas a pravý panel.

Kapitola 5

Používateľská dokumentácia

Používateľská dokumentácia obsahuje všetky potrebné informácie pre používateľa ako napríklad spôsob inštalácie programu, jeho spustenie, základné postupy a možnosti pri práci v editore.

5.1 Systémové požiadavky

Program **UMLtoCATeditor** je aplikácia bežiaca pod Windows 10[1] alebo novším. Bola vytvorená v programe Visual Studio 2022[2] a vyžaduje, aby bol nainštalovaný .Net 6. Program .Net je možné stiahnuť zo stránky spoločnosti Microsoft[3].

5.2 Inštalácia

Inštalácia programu **UMLtoCATeditor** začne po spustení programu s názvom UMLtoCATeditorSetup. Otvorí sa dialógové okno, ktoré vás prevedie inštaláciou s voľbou umiestnenia zdrojových súborov. Po inštalácii sa na pracovnú plochu pridá odkaz na samotnú aplikáciu.

5.3 Spustenie

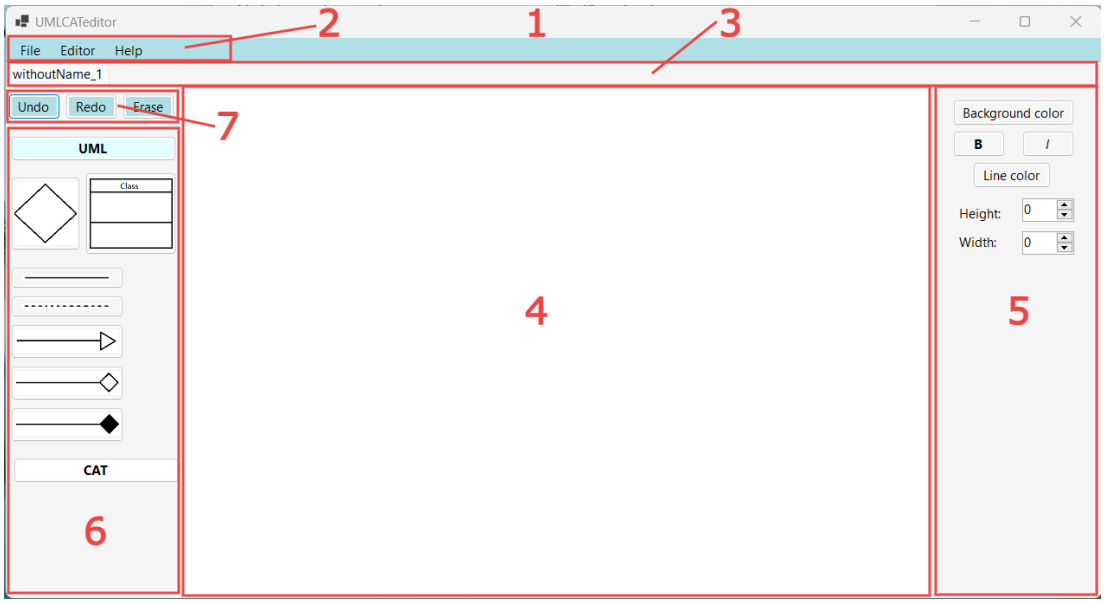
Editor je možné spustiť napríklad po dvojkliku na odkaz aplikácie s názvom *UMLtoCATeditor*, ktorý sa nachádza na pracovnej ploche alebo v zozname nainštalovaných aplikácií. Po spustení sa vám otvorí okno editoru s jednou kartou obsahujúcou prázdny canvas. Následne môžete v editore začať pracovať. Okno editoru je zobrazené na obrázku 5.1.

5.4 Editor

Editor sa otvára vo forme okna s rozmermi 1285 x 693 pixelov, ktoré je možné zväčšovať a obsahuje prázdnu plochu na modelovanie. Na obrázku 5.1 sú číslami znázornené nasledujúce prvky editoru: Okno editoru (1), Horná lišta, ponuka (2),

[1] https://support.microsoft.com/sk-sk/windows
[2] https://visualstudio.microsoft.com/
[3] https://dotnet.microsoft.com/en-us/download

Lišta kariet (3), Plocha na modelovanie (4), Panel na úpravu vlastností jednotlivých objektov (na pravej strane; 5), Panel s objektami, ktoré je možné modelovať (na ľavej strane; 6), a Tlačidlá - Undo, Redo, Erase (7).



Obr. 5.1: Ukážka prostredia editoru

5.4.1 Horná lišta

Horná lišta obsahuje ponuku (resp. menu) s možnosťou výberu. Ponuka je rozdelené do kategórií `File` (súbor), `Editor` a `Help` (pomoc pre používateľa). Na obrázku 5.1 je vyznačená číslom 2.

`File` ponúka prácu so súbormi. Obsahuje:

- `Save as...` – uloženie súboru ako, otvorí sa dialógové okno,
- `Open file` – otvorenie súboru, otvorí sa dialógové okno. Pamätajte, že otvárať je možné iba súbory vo formáte JSON, ktoré boli vygenerované a uložené týmto editorom.

`Editor` ponúka prácu s editorom. Obsahuje:

- `New Canvas` – vytvorí novú kartu s prázdnym canvasom,
- `Clear Canvas` – vymaže aktuálny canvas. Pamätajte, že vymazaný canvas nie je možné vrátiť naspäť,
- `Close Canvas` – zatvorí aktuálnu kartu canvasu a ponúkne možnosť uložiť aktuálny canvas,
- `Exit` – zatvorenie okna editoru, ponúkne uloženie canvasov.

`Help` ponúka pomoc pre používateľa.

5.4.2 Plocha na modelovanie

Biela plocha, tzv. canvas, vyznačená na obrázku 5.1 číslom 4, na ktorú sa vykresľujú napríklad: jednotlivé objekty, prepojenia alebo textové pole. Je možné na nej objekty presúvať, zmeniť ich vlastnosti a taktiež ich aj odstrániť.

5.4.3 Panel na úpravu vlastností jednotlivých objektov

Panel na úpravu vlastností (ďalej len „pravý panel“), vyznačený na obrázku 5.1 číslom 5, má v základnom zobrazení prístupné tlačidlá na zmenu podfarbenia, tučného písma, šikmého písma, farbu čiary, výšky a šírky.

Po výbere objektu sa zobrazia aj doplňujúce, jemu príslušné možnosti na zmenu vlastností, ale základné vlastnosti nie je možné meniť u každého objektu.

Na obrázku A.18 môžete vidieť ukážku pravého panelu, na základe vybraného objektu. Nižšie je uvedený zoznam vlastností, ktoré je možné meniť u jednotlivých objektov.

U **UML class** je možné meniť:

- výšku, šírku,
- podfarbenie a štýl textu u jednotlivých častí samostatne,
- farbu čiary vykresľovaného objektu,
- text v časti názov, atribúty a metódy.

U **UML n-árnej asociácie** je možné meniť:

- výšku a šírku (hodnoty sú rovnaké),
- farbu čiary vykresľovaného objektu.

U **UML asociácie, agregácie a kompozície** je možné meniť:

- šírku, ktorá slúži na zmenu hrúbky vykreslenej čiary,
- farbu čiary vykresľovaného objektu,
- názov vzťahu a jednotlivé multiplicity.

U **UML asociačnej triedy a dedičnosti** je možné meniť:

- šírku, ktorá slúži na zmenu hrúbky vykreslenej čiary,
- farbu čiary vykresľovaného objektu.

U **CAT objektu** je možné meniť:

- výšku, šírku,
- podfarbenie, štýl textu,
- farbu čiary vykresľovaného objektu,

- text objektu a jeho identifikátor.

U **CAT prepojenia** je možné meniť:

- šírku, ktorá slúži na zmenu hrúbky vykreslenej čiary,
- farbu čiary vykresľovaného objektu,
- špecifikáciu prepojenia, formou tagu.

5.4.4 Panel s objektami, ktoré je možné modelovať

Panel sa nachádza na ľavej strane editoru (ďalej len „ľavý panel“), na obrázku 5.1 je vyznačený číslom 6. Obsahuje UML a CAT objekty s ich názvom a obrázkom.

Podrobnosti o vytváraní objektov sú v sekcii 5.5.3, informácie o UML objektoch sú v kapitole 1, informácie o CAT objektoch sú v kapitole 2.

5.4.5 Tlačidlá Undo, Redo, Erase

Nachádzajú sa na ľavej strane editoru, pod hornou lištou. Na obrázku 5.1 sú vyznačené číslom 7. Stláčajú sa jednotlivo, ich použitie je popísané v sekcii 5.5.9.

5.5 Práca s editorom

V tejto sekcii sa podrobnejšie zoznámime s prácou v editore, ako napríklad vytvorenie, presun alebo zmena vlastností objektu.

5.5.1 Výber modelovania

Výber formy modelovania je možný iba pri prázdnom canvase. Ak je na canvase vykreslený nejaký objekt, tlačidlo následne funguje ako tlačidlo prekladu modelu. (Preklad je možný iba smerom z UML do CAT. pozri: 5.5.12, 5.5.13)

Východzia forma modelovania je UML. Kliknutím na tlačidlo `CAT`, na ľavom paneli, sa skryjú UML objekty a zobrazia sa CAT modely, t. j. modelovanie sa zmení na CAT.

5.5.2 Práca s kartami editoru

Karty sa zobrazujú na lište, vyznačenou na obrázku 5.1 číslom 3. Kliknutím na kartu sa požadovaná karta vyberie a canvas sa prekreslí.

Karty sa môžu vytvárať kliknutím na `New Canvas`, odstrániť všetky objekty vykreslené na cavase na danej karte pomocou `Clear Canvas` a zavrieť danú kartu s canvasom pomocou `Close Canvas`.

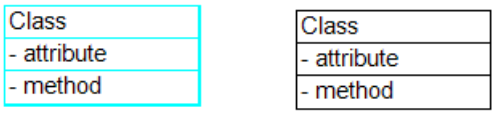
5.5.3 Vytvorenie objektu

Pre vytvorenie objektu je potrebné stlačiť požadované tlačidlo na ľavom paneli.

Asociácie alebo iné prepojenia sa vytvárajú medzi objektami, preto je potrebné najprv vytvoriť objekty a potom vytvárať prepojenia.

5.5.4 Výber objektu

Na výber objektu stačí kliknúť ľavým tlačidlom myši na požadovaný objekt. Jeho obrysy následne zmenia farbu na tyrkysovú, ako je možné vidieť na ľavej časti obrázku 5.2. Pri výbere časti objektu stačí opäť kliknúť na požadovanú časť objektu, ktorú chceme vybrať (využitie napríklad pri UML triede).



Obr. 5.2: Príklad Vybraného a nevybraného objektu

5.5.5 Presun objektu

Na presúvanie objektov, je potrebné vybrať požadovaný objekt. Presúvať je možné iba celý objekt, t. j. napríklad pri UML triede, nie je možné vybrať a presunúť iba časť objektu.

Po výbere objektu stačí kliknúť pravým tlačidlom myši na vybraný objekt a následne tlačidlo držať stlačené. Kurzor sa zmení na ruku a pohybom myši je možné objekt presúvať. Po uvoľnení tlačidla je presun dokončený.

5.5.6 Zmena vlastností objektu

Na zmenu vlastnosti objektu je potrebné vybrať daný objekt. Následne sa vám, podľa daného typu objektu, ukážu prídavné vlastnosti na pravom paneli (pozri: 5.4.3). Z ponúkaných vlastností stačí vybrať požadovanú a zmeniť jej hodnotu.

Poznámky:

- pri zmene farby `Background color` a `Line color` sa zobrazí dialógové okno na výber farby,
- pri zmene textu stačí kliknúť na požadované textové pole, zmeniť text a stlačiť enter na potvrdenie,
- pri zmene veľkosti `Height` a `Width` je možné zadať požadovanú veľkosť priamo alebo ju zmeniť pomocou dvojice tlačidiel označených šípkami,
- CAT identifikátory majú definovanú formu zadávania uvedenú v časti 5.5.7,
- CAT prepojeniu je možné špecifikovať jeho tag pomocou výberu zo zobrazených možností.

5.5.7 Definícia zadávania CAT identifikátoru v editore

Identifikátory sa píšu do textového pola označeného `Identifier:`, ktorý sa nachádza na pravom paneli. Na potvrdenie a uloženie identifikátoru je nutné stlačiť enter.

Jednotlivé identifikátory oddeľujeme znakom bodkočiarky „;“, jednotlivé vlastnosti medzi sebou oddeľujeme použitím dvojice znakov „&&“.

Napr. identifikátory objektu `Person`, na obrázku 2.4d, by sme zapísali ako: *Surname && Tag; Name && Tag; Id;* .

5.5.8 Vytvorenie prepojenia, vzťahu

Prepojenie resp. vzťah sa vytvára medzi dvomi objektami. Z toho vyplýva, že objekty je potrebné mať vytvorené pred samotným vytvorením prepojenia, resp. vzťahu. Na jeho vytvorenie stačí stlačiť tlačidlo prepojenia, resp. vzťahu, podľa potreby. Následne sa na objektoch zobrazia oranžové body, na ktoré je možné pripevniť prepojenie, resp. vzťah. Po výbere oboch bodov sa vytvorí požadované prepojenie, resp. vzťah.

Pre viac informácií o zmenách vlastností prepojení, resp. vzťahov, pozri: 5.4.3, 5.5.6.

5.5.9 Odstránenie a pridanie objektu naspäť

Odstránenie objektu je možné pomocou tlačidla `Erase` alebo `Undo`, ktoré vás vráti o krok späť. Pridanie objektu naspäť je možné iba pomocou jedného alebo viacnásobného stlačenia tlačidla `Redo`.

Tlačidlo `Erase` po kliknutí vymaže objekt, ktorý bol vybraný pred stlačením tlačidla.

Tlačidlá `Undo` a `Redo` fungujú ako „krok dozadu“ a „krok dopredu“. Vždy menia model iba o jeden krok.

5.5.10 Uloženie vytvoreného modelu

Uloženie vytvoreného modelu prebieha po stlačení kláves `ctrl+S` alebo po stlačení `Save as...` z ponuky v hornej lište. Zobrazí sa dialógové okno, po ktorého vyplnení sa všetko z canvasu uloží do súboru vo formáte JSON, s názvom a umiestnením podľa výberu v dialógovom okne.

5.5.11 Načítanie modelu zo súboru

Načítanie súboru je možné po stlačení kláves `ctrl+O` alebo po stlačení `Open file` z ponuky v hornej lište. Je možné načítať iba súbory vo formáte JSON, ktoré boli vytvorené a uložené pomocou tohto editoru.

5.5.12 Preklad modelu z UML do CAT

Preklad prebieha po kliknutí na tlačidlo `CAT` v paneli na ľavej strane editoru. Následne sa otvorí nová karta, kde sa vykreslí preložený model. Pamätajte, že model na novej karte nie je uložený, preto je odporúčané jeho uloženie.

Preklad z UML do CAT sa neuskutoční, ak je UML model nesprávne vytvorený alebo keď na canvase nie je nič vykreslené. (V druhom prípade sa iba zmení forma modelovania z UML na CAT. pozri: 5.5.1)

5.5.13 Preklad modelu z CAT do UML

Tento preklad nie je v editore podporovaný, z dôvodu nemožnosti jeho vykonania. CAT model je viac komplexný a ponúka možnosti, ktoré nie je možné preložiť do UML modelu.

5.5.14 Klávesové skratky

Zoznam podporovaných klávesových skratiek, ktoré je možné využiť. V zátvorke je uvedený názov tlačidla, ktoré nahrádza.

- `ctrl+S` uloženie aktuálneho canvasu do súboru (`Save as...`),
- `ctrl+O` otvorenie a načítanie modelu zo súboru (`Open file`),
- `ctrl+W` zatvorenie aktuálneho canvasu (`Close Canvas`),
- `ctrl+T` otvorenie novej karty s prázdnym canvasom (`New Canvas`),
- `ctrl+D` vymazanie vybraného objektu na canvase (`Erase`),
- `ctrl+Z` krok dozadu (`Undo`),
- `ctrl+Y` krok dopredu (`Redo`).

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať aparát umožňujúci modelovanie viacmodelových dátových schém pomocou jazykov UML a CAT, ktorý taktiež umožňuje preklad UML schémy na CAT schému. Výsledný funkčný editor kladené požiadavky splňuje a pokladám ho za vhodný nástroj na podporu pochopenia novo vytvoreného kategorického modelu.

Nástroj má jednoduchý dizajn, je prívetivý k používateľovi a obsahuje všetky základné prvky jazykov UML a CAT. Používateľovi dovoľuje modelovať v oboch schémach, prekladať UML model na CAT model a mimo iné aj ukladanie modelu do súboru vo formáte JSON.

Na druhej strane si uvedomujem nemožnosť konkurovať viacročným vývojom editorov, napríklad zobrazených v rešerši, s bohatou ponukou možností a spôsobov na prácu v samotnom editore alebo komfortom pre používateľa. Zároveň, ako už bolo spomínané, hlavným zámerom editoru je primárne podporiť jednoduchšie pochopenie nového prístupu, preto naň neboli kladené tak prísne požiadavky, ako napríklad na editory slúžiace iba na modelovanie. Tieto skutočnosti avšak nebránia ďalšiemu vývoju a zdokonaľovaniu editora.

Možný priestor na zdokonalenie vidím napríklad v týchto oblastiach:

- možnosť priblíženia canvasu,
- ukladanie modelu vo forme obrázku,
- viac možností na úpravu modelu (napríklad zmena veľkosti písma, možnosť výber fontu textu)
- možnosť zalamovať a presúvať prepojenia, resp. vzťahy, podľa preferencie používateľa.

Zoznam použitej literatúry

[1] Barr, M. ; Wells, C. : *Category Theory for Computing Science*. Bd. 1. New York : Prentice Hall, 1990

[2] Chen, P. : The Entity-Relationship Model – Toward a Unified View of Data. In: *ACM Transactions on Database Systems* 1 (1976), Nr. 1, S. 9–36. – ISSN 0362–5915

[3] Holubová, I. ; Contos, P. ; Svoboda, M. : Multi-Model Data Modeling and Representation: State of the Art and Research Challenges. In: *Proc. of IDEAS '21*, ACM, 2021, S. 242–251

[4] International, E. : *JavaScript Object Notation (JSON)*. 2013. – http://www.JSON.org/

[5] Jim, A. ; Ila, N. : *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Brno, CZE : Computer Press, 2007. – ISBN 978–80–251–1503–9

[6] Koupil, P. : *Towards Unified Management of Multi-Model Data*. https://www.ksi.mff.cuni.cz/~koupil/misc/Towards%20Unified%20Management%20of%20Multi-Model%20Data.pdf, 2023

[7] Koupil, P. ; Holubová, I. : A Unified Representation and Transformation of Multi-Model Data using Category Theory. In: *J. Big Data* 9 (2022), Nr. 1, S. 61

[8] Koupil, P. ; Hricko, S. ; Holubová, I. : A universal approach for multi-model schema inference. In: *J. Big Data* 9 (2022), Nr. 1, 97. http://dx.doi.org/10.1186/s40537-022-00645-9. – DOI 10.1186/s40537–022–00645–9

[9] Sadalage, P. J. ; Fowler, M. : *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2013

[10] Svoboda, M. ; Čontoš, P. ; Holubová, I. : Categorical Modeling of Multi-Model Data: One Model to Rule Them All. In: *Model and Data Engineering*. Cham : Springer International Publishing, 2021 (Lecture Notes in Computer Science). – ISBN 978–3–030–78428–7, S. 190–198

[11] W3C: *RDF/XML Syntax Specification (Revised)*. W3C, 2004. – http://www.w3.org/TR/rdf-syntax-grammar/

[12] W3C: *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých skratiek

BFS Breadth-First Search. 38

CAT Category modeling. 5, 6, 12, 14, 18, 33, 36, 38, 45–47, 49, 50, 56, 57, 65, 68

CSV Comma-Separated Values. 30

DFS Depth-First Search. 38

DOC Document. 28

EDDX súbor vytvorený Edraw Max and Network Diagram Maker. 30

ER Entity-Relationship. 5, 6, 9, 17

HTML Hypertext Markup Language. 30

JPEG Joint Photographic Experts Group. 24–26, 30

JSON JavaScript Object Notation. 5, 40, 44, 48, 50, 76

PDF Property Domain Footprint. 21, 24–26, 28, 30

PNG Portable Network Graphics. 21, 24–26, 30

PPTX PowerPoint. 28

RDF Resource Description Framework. 5

SVG Scalable Vector Graphics. 30

UML Unified Modeling Language. 5–7, 9, 12, 14, 15, 18, 20, 21, 24–28, 30, 31, 33, 35, 36, 38, 41, 45–47, 49, 50, 56, 57, 65, 68

VDX Virtual Document eXchange. 30

VSD Microsoft Visio Drawing File. 30

VSDX Visio drawing. 30

XML eXtensible Markup Language. 5, 22, 30

Dodatok A

Prílohy

A.1 Testovanie editoru

Editor a jeho funkčnosť som testoval pomocou dvoch druhov testov, ktorými sú: jednotkové testy a integračné testy.

A.1.1 Jednotkové testy

Na testovanie jednotkovými testami som vybral metódy, ktoré sú verejné a má zmysel ich testovať. To znamená, že metóda je verejná a na základe vstupných atribútov overí správnosť alebo napríklad vykoná výpočet. Na samotné testovanie som použil framework MSTests[1].

Testovanými metódami sú:

- `SetPossibleIdentifiers` triedy `CATBoxObject`,
- `ClickedOnConnection` triedy `ConnectionsOnCanvas`

A.1.2 Integračné testy

Integračnými testami som overoval správnosť fungovania editoru a korektnosť prekladu modelu z UML do CAT. Boli vykonávané v editore kreslením jednotlivých testov z pozície používateľa. Jednotivé testy a ich výsledky sú zobrazené v prílohe A.2

A.1.3 Zhodnotenie výsledkov testov

Na základe vykonaných testov môžeme zhodnotiť, že editor pracuje správne a preklad je korektný. Výsledky testov sú v prílohe A.2.

A.2 Integračné testy a ich výsledky

Nižšie sú zobrazené jednotlivé testy s predlohou v UML, ktorá sa prekreslila do editoru a následne jej preklad do CAT vygenerovaný editorom, ktorý sa porovnal s CAT predlohou.

[1] https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest

A.2.1 Test č. 1 – UML trieda

Test prekladu UML triedy a správneho prepojenia jej vlastností (pôvodných atribútov UML triedy) v CAT modeli.

Test je zobrazený na obrázku A.1.

A.2.2 Test č. 2 – UML dedičnosť

Test prekladu UML dedičnosti, správneho prepojenia objektov a označenie hrán tagom #ISA.

Test je zobrazený na obrázku A.2.

A.2.3 Test č. 3 – UML asociácia

Test prekladu pomenovanej UML asociácie. V CAT modeli musí byť pôvodné pomenovanie asociácie v samostatnom objekte, hrany vedú z tohto objektu smerom k prepojeným v pôvodnom UML modeli s tagom #role.

Test je zobrazený na obrázku A.3.

A.2.4 Test č. 4 – UML asociačná trieda

Test prekladu UML asociačnej triedy, ktorý má podobný výstup ako test A.2.3. Rozdiel je v tom, že namiesto pomenovania asociácie sa do objektu vkladá názov asociačnej triedy a z tohto objektu vedú hrany smerom k vlastnostiam (pôvodným atribútom UML asociačnej triedy) v CAT modeli.

Test je zobrazený na obrázku A.4.

A.2.5 Test č. 5 – UML agregácia

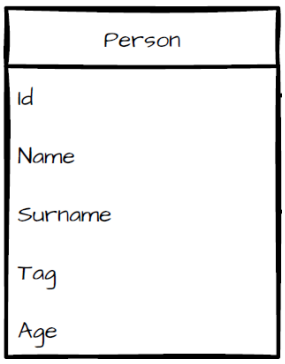
Test prekladu UML agregácie, ktorá sa preloží na CAT štruktúrovaný atribút. Test zahŕňa kontrolu smeru hrán, ktoré majú správne smerovať od objektu reprezentujúceho pôvodný UML celok do objektov reprezentujúcich jeho pôvodné UML súčasti.

Test je zobrazený na obrázku A.5.

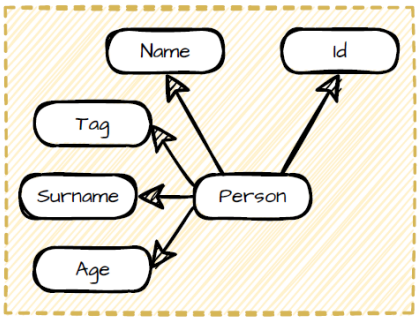
A.2.6 Test č. 6 – UML komplexný príklad

Tento test zahŕňa všetky predošlé testy, t. j. overuje celkovú správnosť fungovania prekladu.

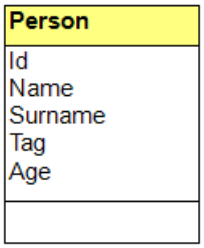
Test je zobrazený na obrázku A.6.



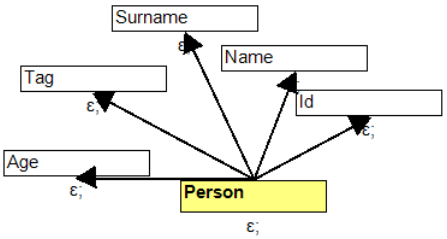
(a) UML predloha



(b) CAT predloha, prevzaté z [6]

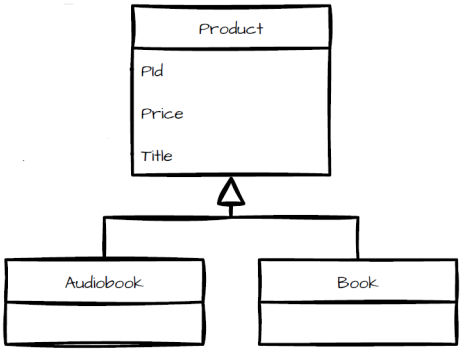


(c) UML model v editore

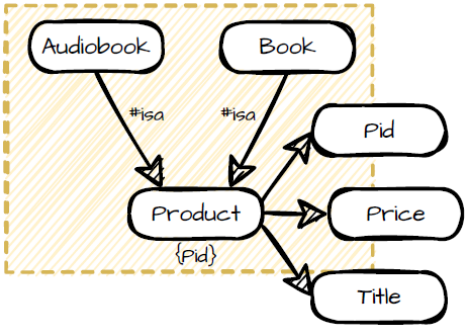


(d) CAT model preložený editorom

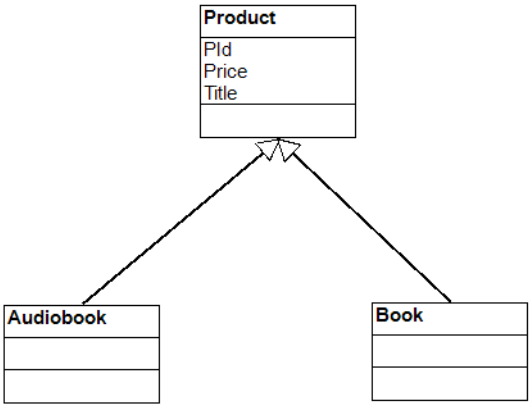
Obr. A.1: Test číslo 1, preklad UML triedy



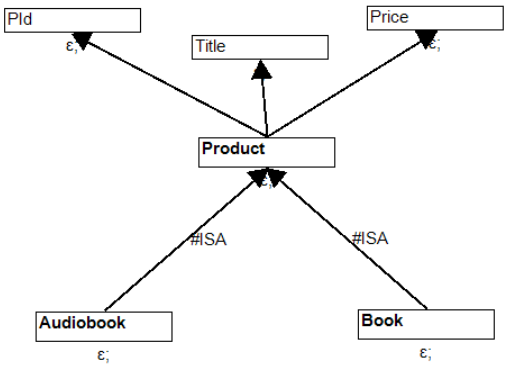
(a) UML predloha



(b) CAT predloha, prevzaté z [6]

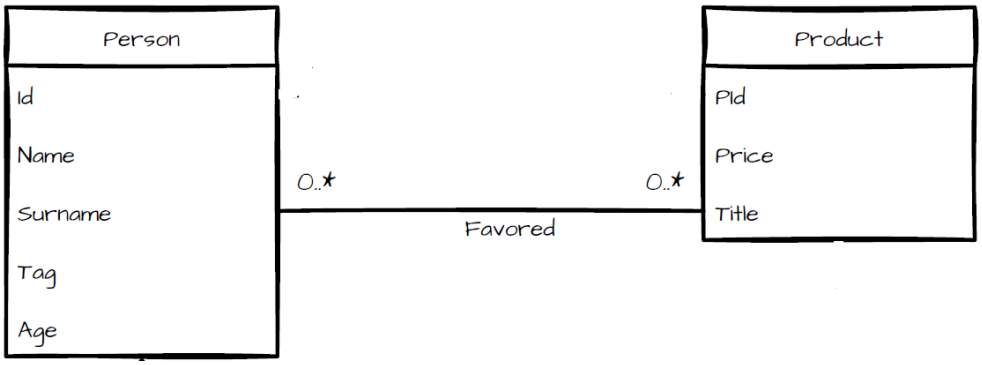


(c) UML model v editore

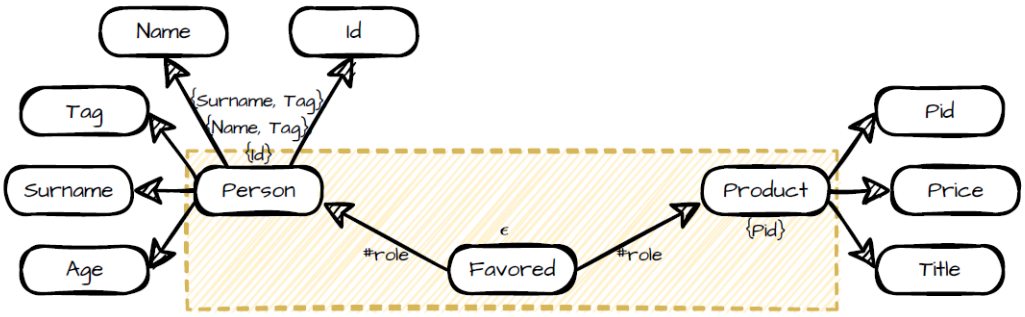


(d) CAT model preložený editorom

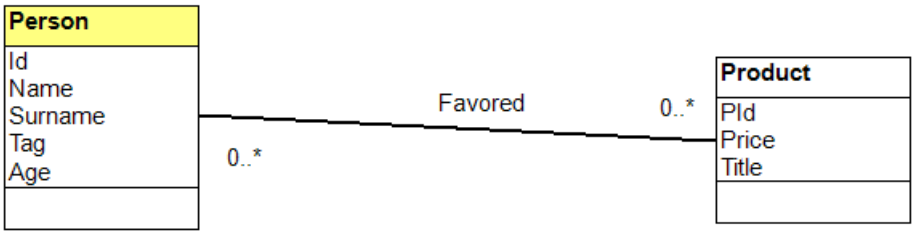
Obr. A.2: Test číslo 2, preklad UML dedičnosti



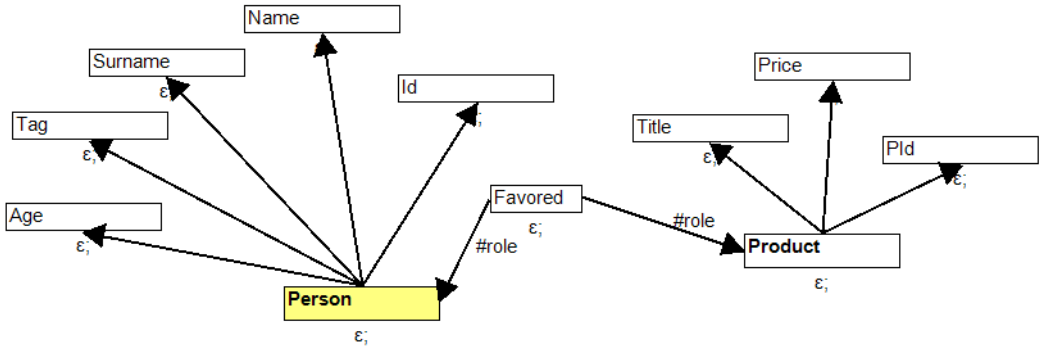
(a) UML predloha



(b) CAT predloha, prevzaté z [6]

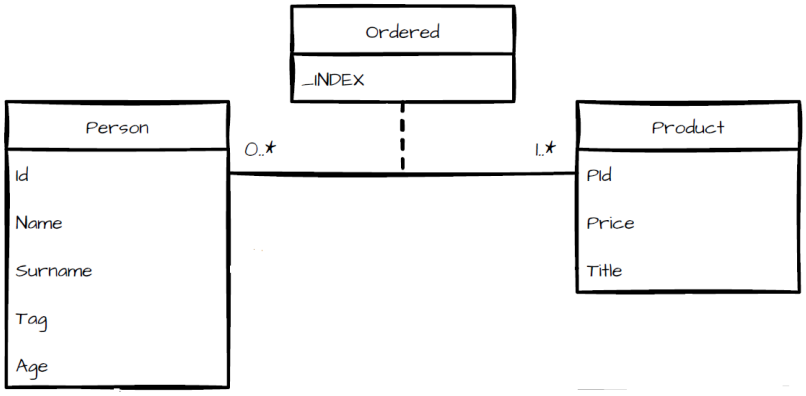


(c) UML model v editore

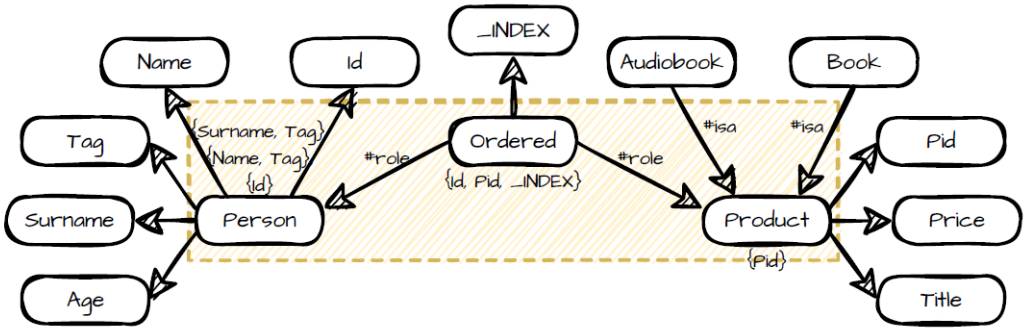


(d) CAT model preložený editorom

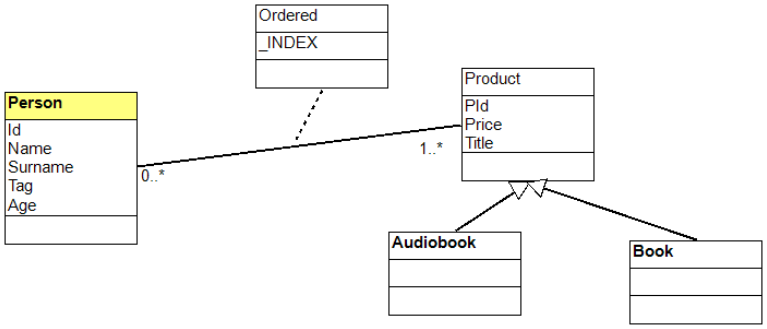
Obr. A.3: Test číslo 3, preklad UML asociácie



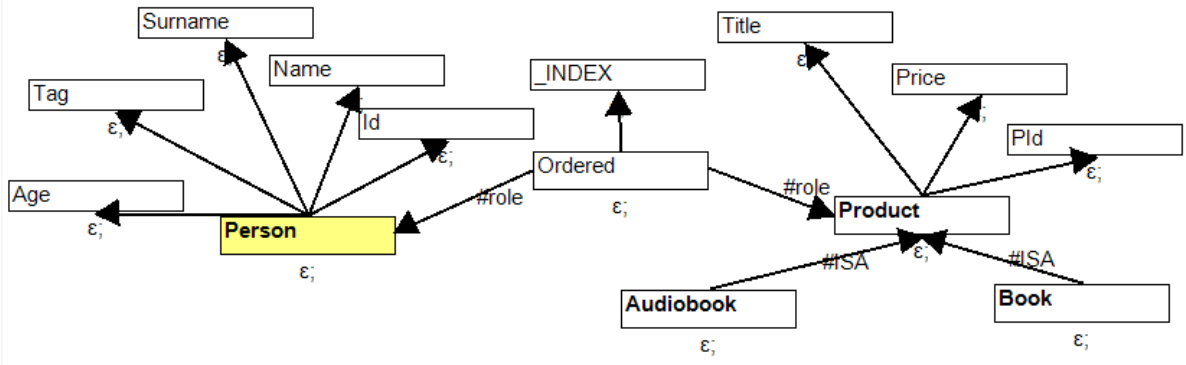
(a) UML predloha



(b) CAT predloha, prevzaté z [6]

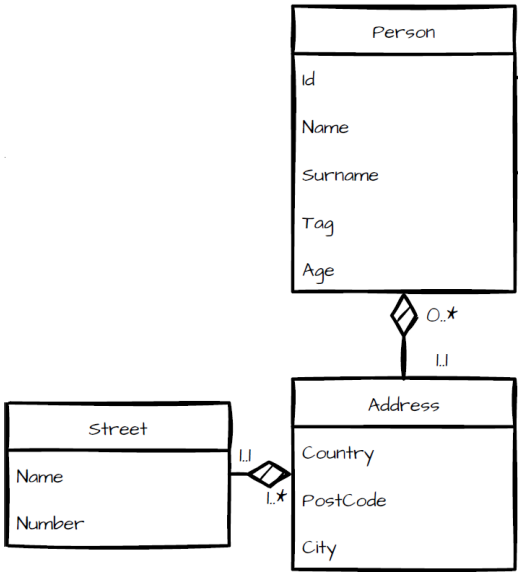


(c) UML model v editore

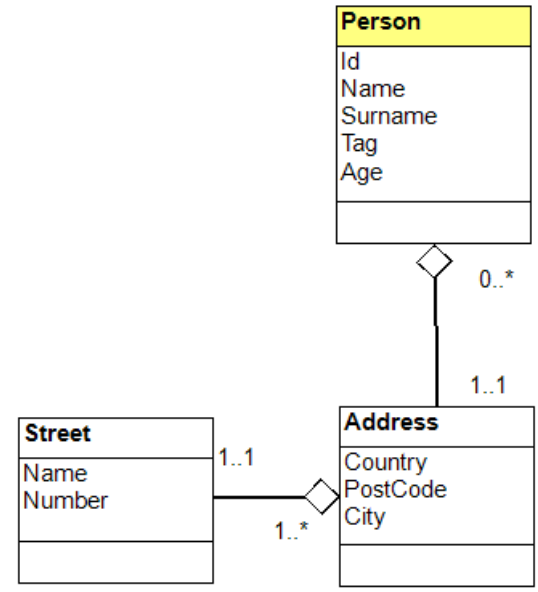


(d) CAT model preložený editorom

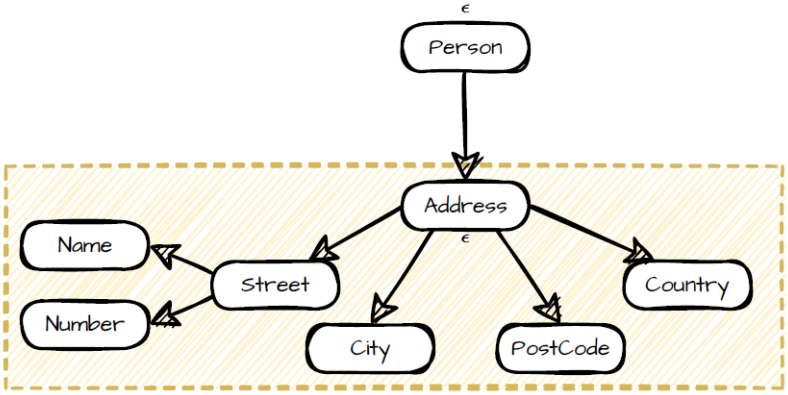
Obr. A.4: Test číslo 4, preklad UML asociačnej triedy



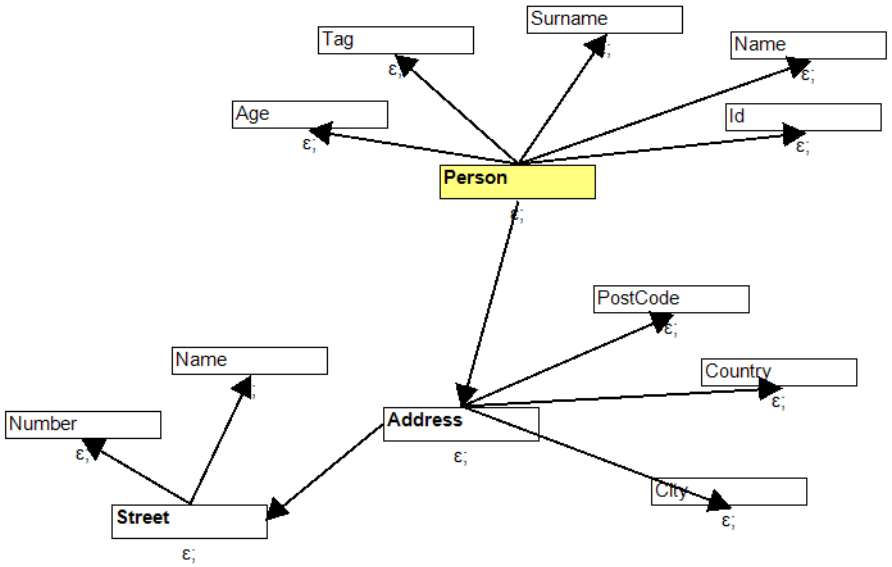
(a) UML predloha



(b) UML model v editore

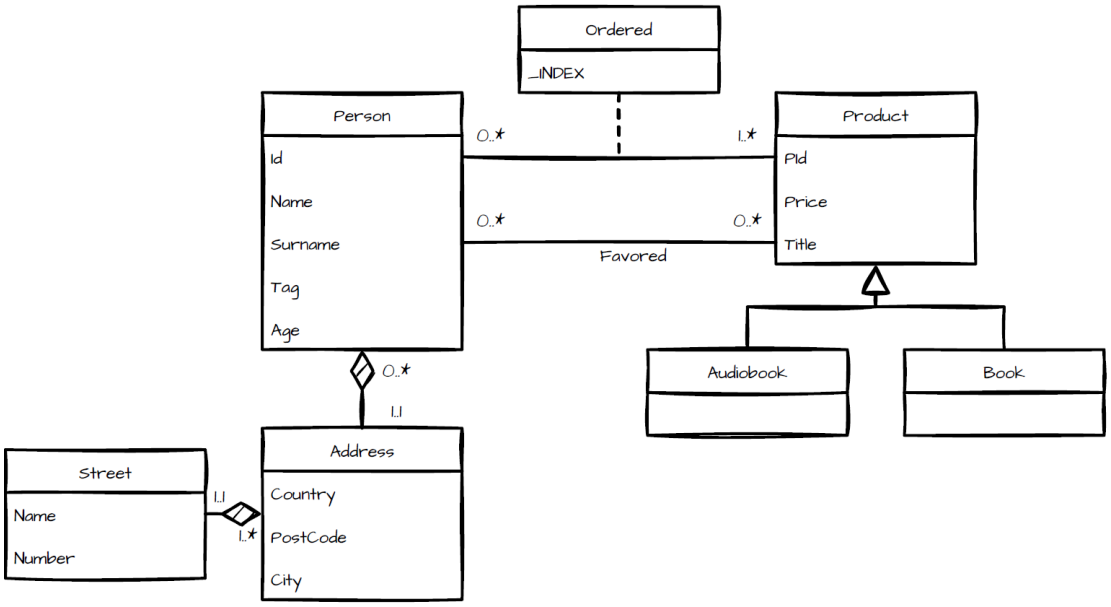


(c) CAT predloha, prevzaté z [6]

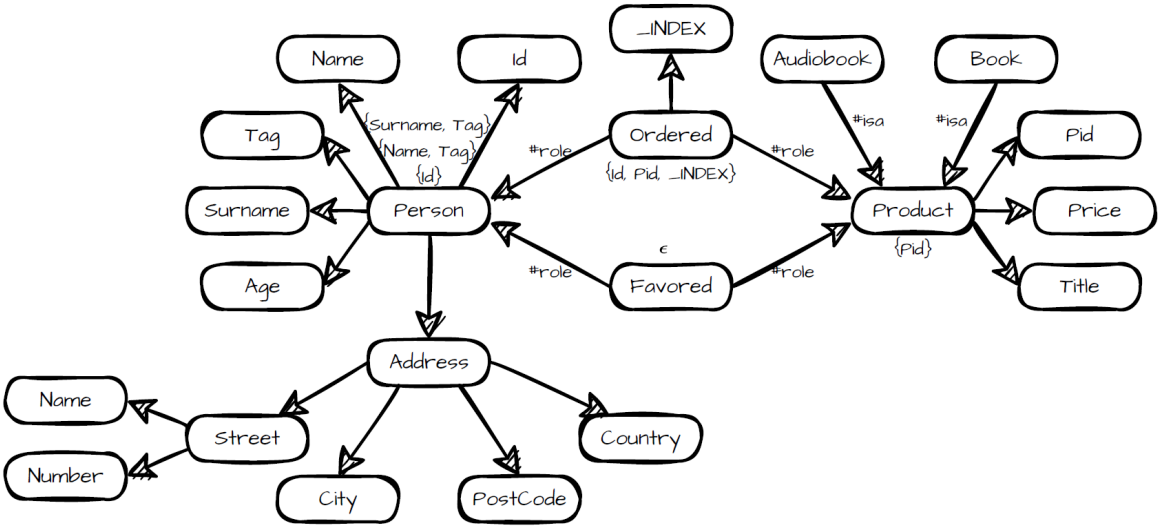


(d) CAT model preložený editorom

Obr. A.5: Test číslo 5, preklad UML agregácie

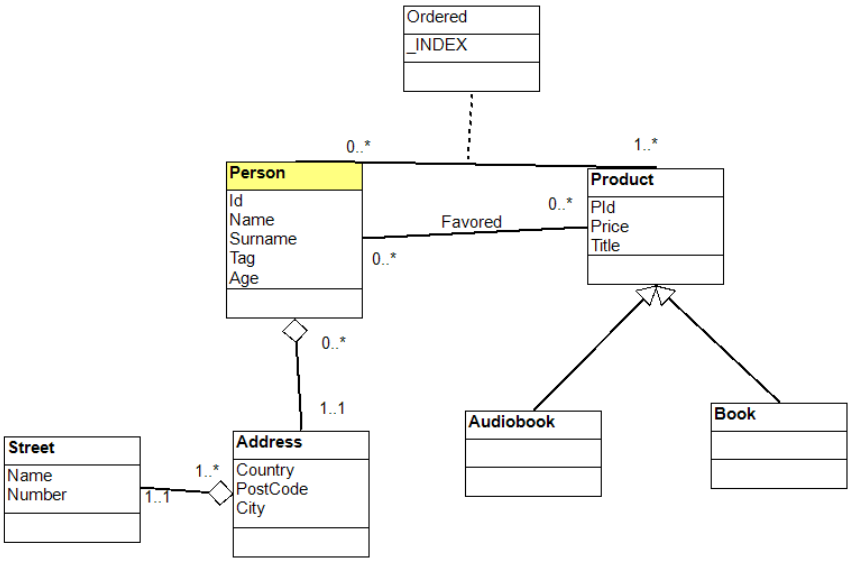


(a) UML predloha

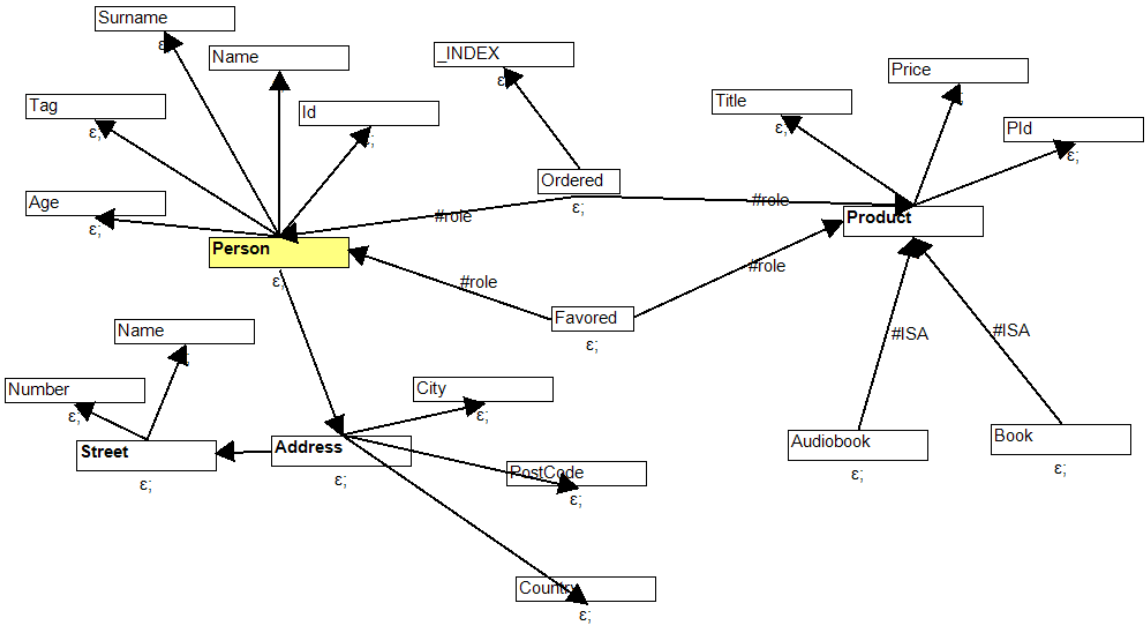


(b) CAT predloha, prevzaté z [6]

Obr. A.6: Test číslo 6, preklad UML komplexného príkladu



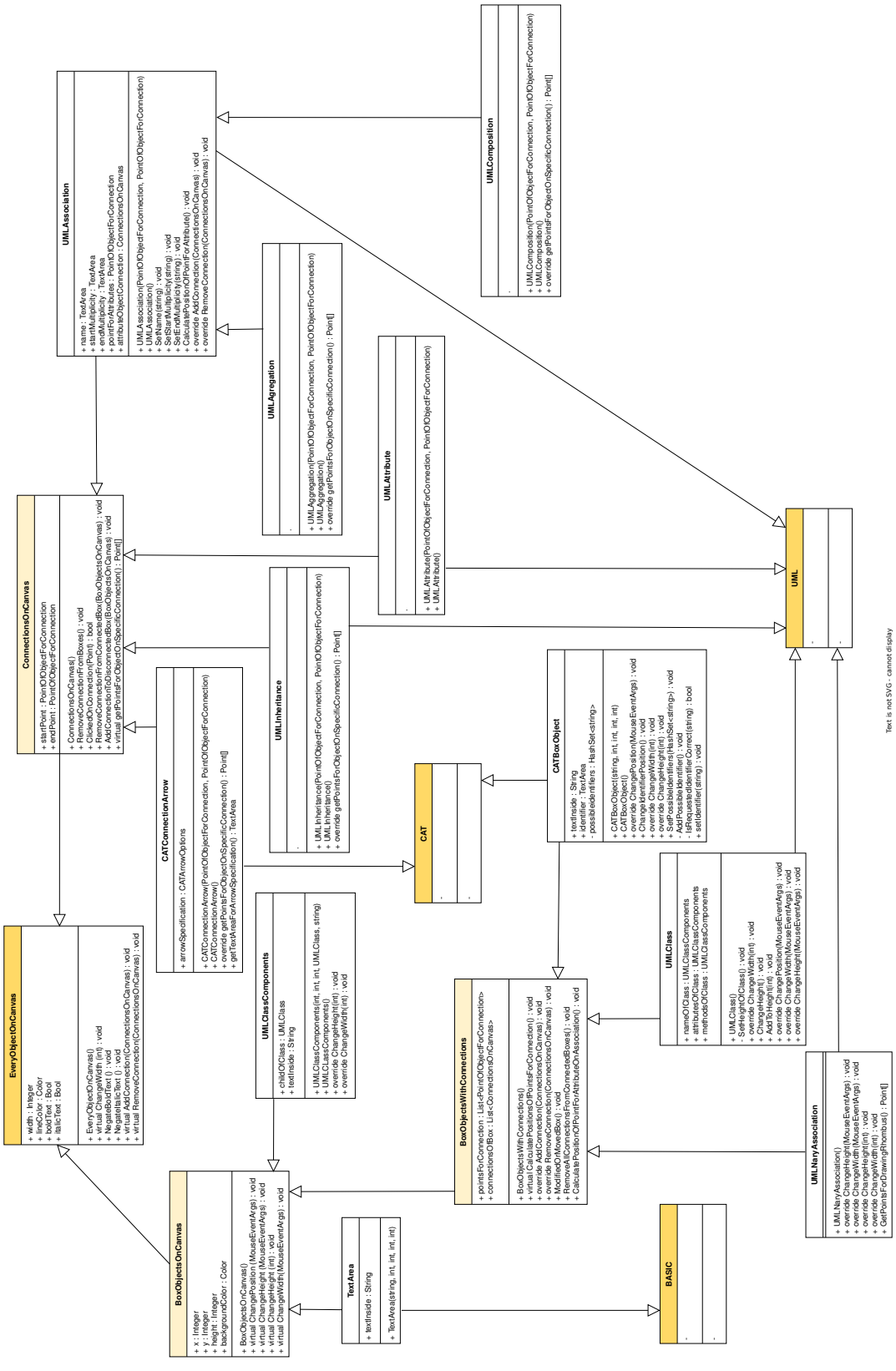
(c) UML model v editore



(d) CAT model preložený editorom

Obr. A.6: Test číslo 6, preklad UML komplexného príkladu

A.3 Návrhový vzor objektov vykresľovaných na canvas



Obr. A.7: Návrhový vzor objektov vykresľovaných na canvas

A.4 Pseudokód algoritmu prekladu UML modelu na CAT model

V tejto prílohe sa nachádza pseudokód algoritmu prekladajúceho UML model na CAT model. Niektoré metódy triedy `UMLandCATtranslator` nie sú v pseudokóde zobrazené z dôvodu lepšej prehľadnosti. Nezobrazené metódy sú iba pomocné, preto ich skrytím algoritmus nestratí na zmysle ani na možnosti jeho pochopenia.

```
1  Trieda UMLandCATtranslator
2
3     Metóda TrasnlateFromUMLToCAT
4     vstup: list listCanvas objektov vykresľovaných na canvase
5        Ak listCanvas obsahuje objekty iné ako UML alebo BASIC:
6           vráť NULL
7
8        zásobník canvasObjects <- objekty BoxObjectsWithConnections z listu
             listCanvas
9        visitedObjects <- nový list
10       visitedConnections <- nový list
11       translated <- nový list
12       translatedUMLtoCAT <- nový slovník
13
14       Pokým existuje objekt v canvasObjects:
15          vyber posledný objekt v z canvasObjects
16          Ak v nie je v liste visitedObjects:
17             zavolaj UMLtoCATrecursionDFS s parametrami v, visitedObjects,
                   visitedConnections, translated, translatedUMLtoCAT
18    výstup: list preložených objektov translated
19
20    Metóda UMLtoCATrecursionDFS
21    vstup: objekt box, list už navštívených objektov visitedObjects, list už navš
          tívených prepojení visitedConnections, list už preložených objektov
          translated, slovník preložených objektov translatedUMLtoCAT
22       pridaj box do visitedObjects
23       Ak box je UMLclass:
24          cbox <- preklad boxu na CAT objekt
25          pridaj cbox do translated
26          pridaj kľúč box a hodnotu cbox do translatedUMLtoCAT
27          zavolaj translateUMLattributesToCat s parametrami box, cbox, translated
28
29          Pre každé prepojenie connection objektu box:
30             Ak list visitedConnections obsahuje connection:
31                pokračuj na ďalšie prepojenie
32
33             pridaj connection do visitedConnections
34
35             Ak začiatok prepojenia nie je na objekte box a objekt na začiatku
                   prepojenia je BoxObjectsWithConnections b:
36                Ak objekt b už bol preložený:
37                   zavolaj createCATconnectionBetweenObjects s parametrami
                         connection, prekladom b, cbox, translated, visitedObjects,
                          translatedUMLtoCAT
38                Inak:
39                   prelož b zavolaním UMLtoCATrecursionDFS s parametrami b,
                         visitedObjects, visitedConnections, translated,
                         translatedUMLtoCAT
40                   zavolaj createCATconnectionBetweenObjects s parametrami
                         connection, prekladom b, cbox, translated, visitedObjects,
                          translatedUMLtoCAT
41             Inak ak na konci prepojenia je BoxObjectsWithConnections b:
42                Ak objekt b už bol preložený:
43                   zavolaj createCATconnectionBetweenObjects s parametrami
                         connection, cbox, prekladom b, translated, visitedObjects,
                          translatedUMLtoCAT
44                Inak:
45                   prelož b zavolaním UMLtoCATrecursionDFS s parametrami b,
                         visitedObjects, visitedConnections, translated,
                         translatedUMLtoCAT
```

```
46                  zavolaj createCATconnectionBetweenObjects s parametrami
                        connection, cbox, prekladom b, translated, visitedObjects,
                         translatedUMLtoCAT
47              Inak:
48                  pokračuj na ďalšie prepojenie
49          vráť cbox
50      Inak ak box je UMLNaryAssociation:
51          cbox <- preklad boxu na CAT objekt
52          pridaj cbox do translated
53          pridaj kľúč box a hodnotu cbox do translatedUMLtoCAT
54
55          Pre každé prepojenie connection objektu box:
56              Ak list visitedConnections obsahuje connection:
57                  pokračuj na ďalšie prepojenie
58
59              pridaj connection do visitedConnections
60
61              Ak connection je UMLAssociation:
62                  Ak začiatok prepojenia nie je na objekte box a objekt na začiatku
                        prepojenia je BoxObjectsWithConnections b:
63                      Ak objekt b už bol preložený:
64                          zavolaj createCATconnectionBetweenObjects s parametrami
                                connection, prekladom b, cbox, translated,
                                visitedObjects, translatedUMLtoCAT
65                      Inak:
66                          prelož b zavolaním UMLtoCATrecursionDFS s parametrami b,
                                visitedObjects, visitedConnections, translated,
                                translatedUMLtoCAT
67                          zavolaj createCATconnectionBetweenObjects s parametrami
                                connection, prekladom b, cbox, translated,
                                visitedObjects, translatedUMLtoCAT
68                  Inak ak na konci prepojenia je BoxObjectsWithConnections b:
69                      Ak objekt b už bol preložený:
70                          zavolaj createCATconnectionBetweenObjects s parametrami
                                connection, cbox, prekladom b, translated,
                                visitedObjects, translatedUMLtoCAT
71                      Inak:
72                          prelož b zavolaním UMLtoCATrecursionDFS s parametrami b,
                                visitedObjects, visitedConnections, translated,
                                translatedUMLtoCAT
73                          zavolaj createCATconnectionBetweenObjects s parametrami
                                connection, cbox, prekladom b, translated,
                                visitedObjects, translatedUMLtoCAT
74                  Inak:
75                      pokračuj na ďalšie prepojenie
76
77          vráť cbox
78      výstup: preklad objektu box
79
80      Metóda createCATconnectionBetweenObjects
81      vstup: prepojenie connection, preložený objekt cStart na začiatku prepojenia,
             preložený objekt cEnd na konci prepojenia, list translated, list visited
            , slovník translatedUMLtoCAT
82          Ak connection je UMLAssociation a nie (UMLAggregation alebo UMLComposition
                ):
83              Ak je ku connection pripojená asociačná trieda:
84                  Ak je asociačná trieda nesprávne prepojená:  // korektná asociačná
                        trieda je prepojenie medzi UMLClass a UMLAssociation
85                      vyhoď výnimku
86                  Ak UMLClass c nie je v liste visited:
87                      associationBox <- preklad c na CAT objekt
88                      zavolaj translateUMLattributesToCat s parametrami c,
                            associationBox, translated
89                      pridaj c do visited
90                      pridaj associationBox do translated
91                      pridaj kľúč c a hodnotu associationBox do translatedUMLtoCAT
92                  Inak:
93                      associationBox <- vyber hodnotu z translatedUMLtoCAT na základe k
                            ľúča c
94              Inak:
95                  associationBox <- preklad asociácie na CAT objekt
96                  pridaj associationBox do translated
97              arrowA <- vytvor CAT prepojenie medzi associationBox a cStart, tag #
```

```
                role    // smer spojenia je ->
98              arrowB <- vytvor CAT prepojenie medzi associationBox a cEnd,  tag #role
                        // smer spojenia je ->
99              pridaj arrowA medzi prepojenia objektov associationBox a cStart
100             pridaj arrowB medzi prepojenia objektov associationBox a cEnd
101
102         Inak ak connection je UMLAggregation alebo UMLComposition:
103             arrow <- vytvor CAT prepojenie medzi medzi cEnd a cStart, bez tagu   //
                     smer prepojenia je ->
104             pridaj arrow do translated
105             pridaj arrow medzi prepojenia objektov cStart a cEnd
106         Inak ak connection je UMLInheritance:
107             arrow <- vytvor CAT prepojenie medzi medzi cStart a cEnd, tag #ISA   //
                     smer prepojenia je ->
108             pridaj arrow do translated
109             pridaj arrow medzi prepojenia objektov cStart a cEnd
110     výstup: —
111
112     Metóda translateUMLattributesToCat:
113     vstup: UML trieda uc, preložená UML trieda createdCATbox, list preložených
            objektov translated
114         attributes <- rozdeľ atribúty uc na jednotlivé časti podľa oddeľovačov
                riadkov
115         Pre každý atribút z attributes:
116             associationBox <- vytvor CAT objekt podľa atribútu
117             pridaj associationBox do translated
118             arrow <- vytvor CAT prepojenie medzi createdCATbox a associationBox  //
                     smer prepojenia je ->
119             pridaj arrow do translated
120             pridaj arrow medzi prepojenia objektov associationBox a createdCATbox
121     výstup: —
```

A.5 Sekvenčné diagramy

V tejto prílohe sa nachádzajú vybrané sekvenčné diagramy vytvorené podľa prípadov použitia v časti 4.2. To znamená, že priložené diagramy modelujú komplexné postupnosti akcií. Triviálne postupnosti obsahujúce napríklad jedno alebo dve kliknutia nie sú priložené. Sekvenčný diagram pre takúto postupnosť by nebol veľmi prínosný.

Základný úvod do sekvenčných diagramov Kľúčovými prvkami sekvenčných diagramov sú čiary života a správy.

Čiara života zastupuje jedného účastníka interakcie. Môže mať názov, ktorý sa používa ako identifikátor v rámci interakcie a typ určujúci názov klasifikátoru, ktorého inštanciu čiara života znázorňuje. [5] Od názvu sa oddeľuje dvojbodkou. Vykresľuje sa vo forme obdĺžniku, ktorý obsahuje text a uprostred spodnej hrany začína čiarkovaná polpriamka.

Správa vyjadruje špecifický typ komunikácie medzi čiarami života. Komunikácia môže zahŕňať: volanie operácie, tvorbu alebo uvoľnenie inštancie, či odoslanie signálu. Pre každú volajúcu správu prijatú čiarou života, musí v klasifikátore tejto čiary života existovať odpovedajúca operácia. V diagrame sú znázornené jednoduchou šípkou synchrónne operácie, kedy odosielateľ čaká na dokončenie úlohy príjemcom správy a čiarkovanou šípkou je znázornený návrat aktivity jej odosielateľovi od jej príjemcu. Šípka s nápisom `<<create>>` znázorňuje správu určenú k tvorbe objektu. [5]

Rámec s názvom `alt` znázorňuje vetvenie na základe podmienok uvedených v hranatých zátvorkách, alebo kľúčového slova `else`. Rámec s názvom `loop` zastupuje iterovanie objektu alebo na základe podmienky uvedenej v hranatej zátvorke a rámec s názvom `distruptable` zabezpečuje bezpečné vykonanie príkazov v časti `[try]`. V prípade ich neúspechu sa vykonávanie preskočí, pokračuje sa vo vetve podľa typu výnimky v hranatých zátvorkách a následne vykonávanie príkazov ďalšej časti kódu.

A.5.1 Stlačenie tlačidla na UML modelovanie

Sekvenčný diagram na obrázku A.8 modeluje postupnosť akcií vykonávaných po stlačení tlačidla na UML modelovanie.

A.5.2 Stlačenie tlačidla na CAT modelovanie

Sekvenčný diagram na obrázku A.9 modeluje postupnosť akcií vykonávaných po stlačení tlačidla na CAT modelovanie.

A.5.3 Stlačenie tlačidla na vyčistenie canvasu

Sekvenčný diagram na obrázku A.10 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Clear Canvas`.

A.5.4 Stlačenie tlačidla na zatvorenie canvasu

Sekvenčný diagram na obrázku A.11 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Close Canvas`.

A.5.5 Stlačenie tlačidla Undo

Sekvenčný diagram na obrázku A.12 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Undo` alebo kláves `ctrl+Z`.

A.5.6 Stlačenie tlačidla Redo

Sekvenčný diagram na obrázku A.13 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Redo` alebo kláves `ctrl+Y`.

A.5.7 Stlačenie tlačidla Erase

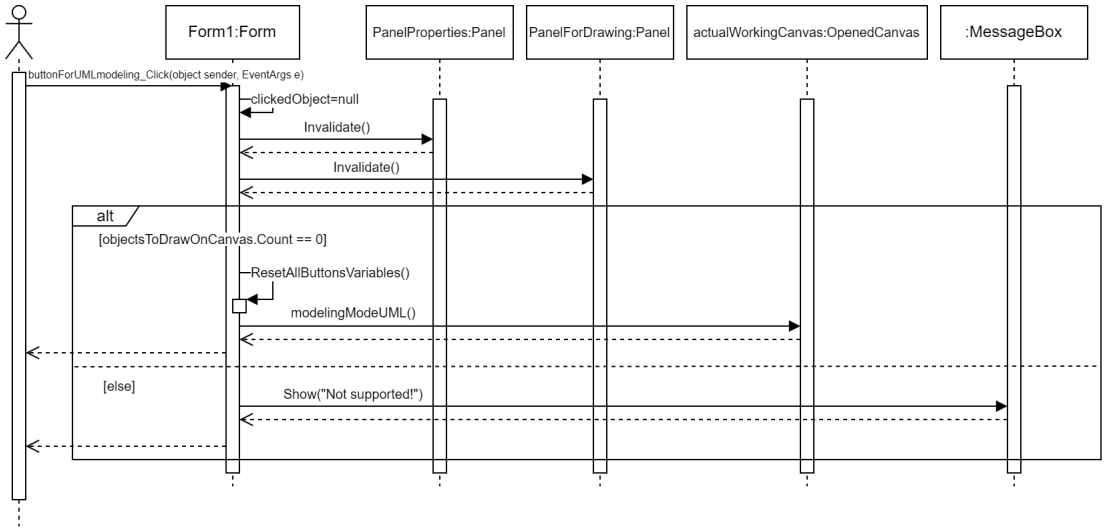
Sekvenčný diagram na obrázku A.14 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Erase` alebo stlačení kláves `ctrl+D`.

A.5.8 Stlačenie tlačidla Save as...

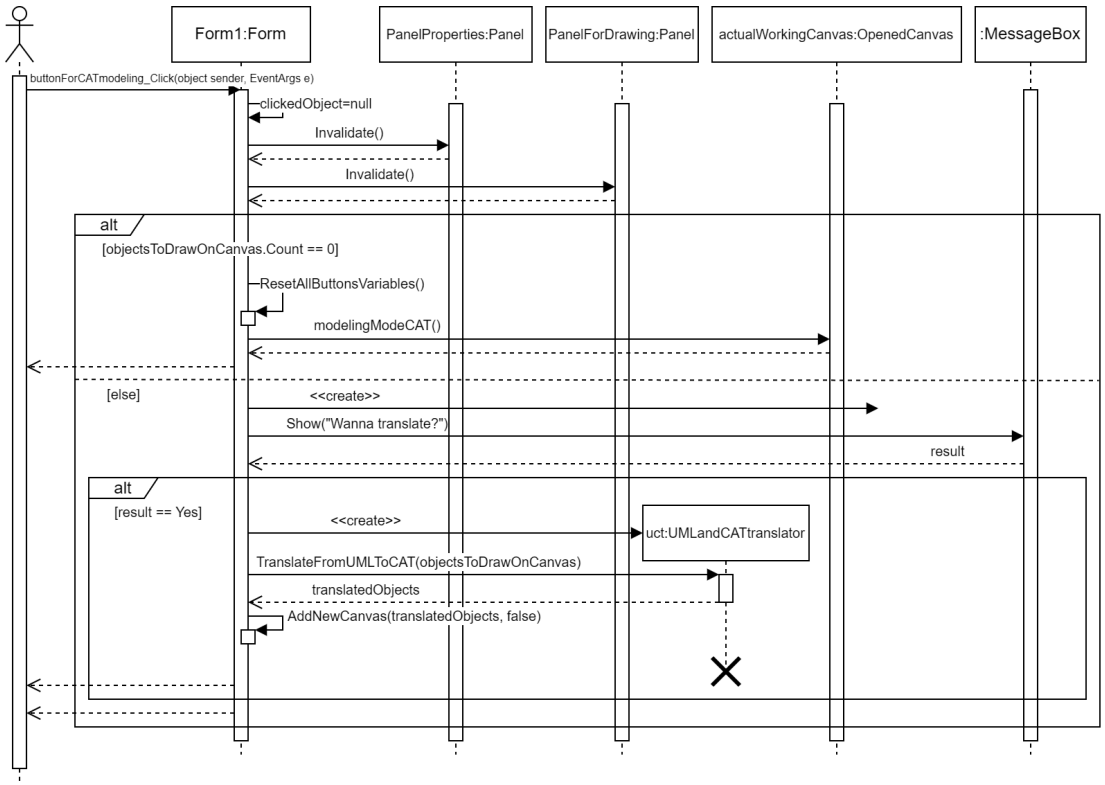
Sekvenčný diagram na obrázku A.15 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Save as...` alebo kláves `ctrl+S`.

A.5.9 Stlačenie tlačidla Open file

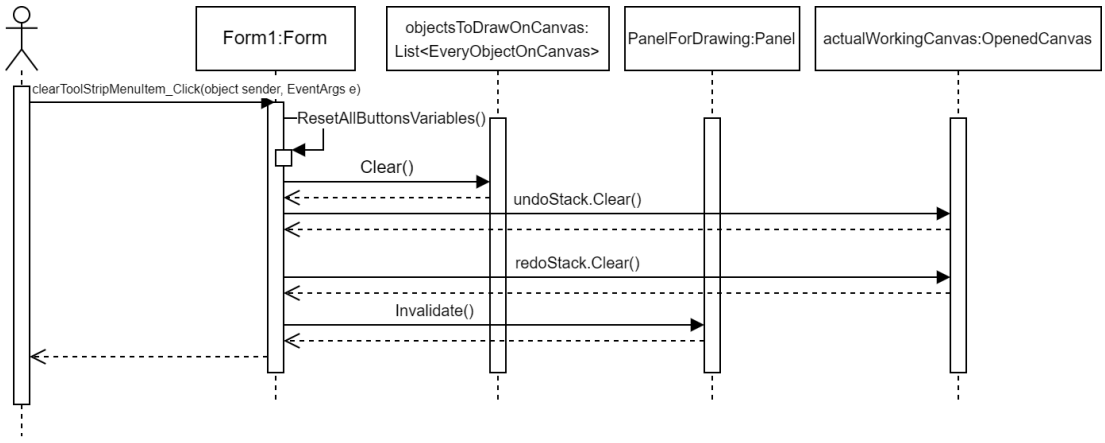
Sekvenčný diagram na obrázku A.16 modeluje postupnosť akcií vykonávaných po stlačení tlačidla `Open file` alebo kláves `ctrl+O`.



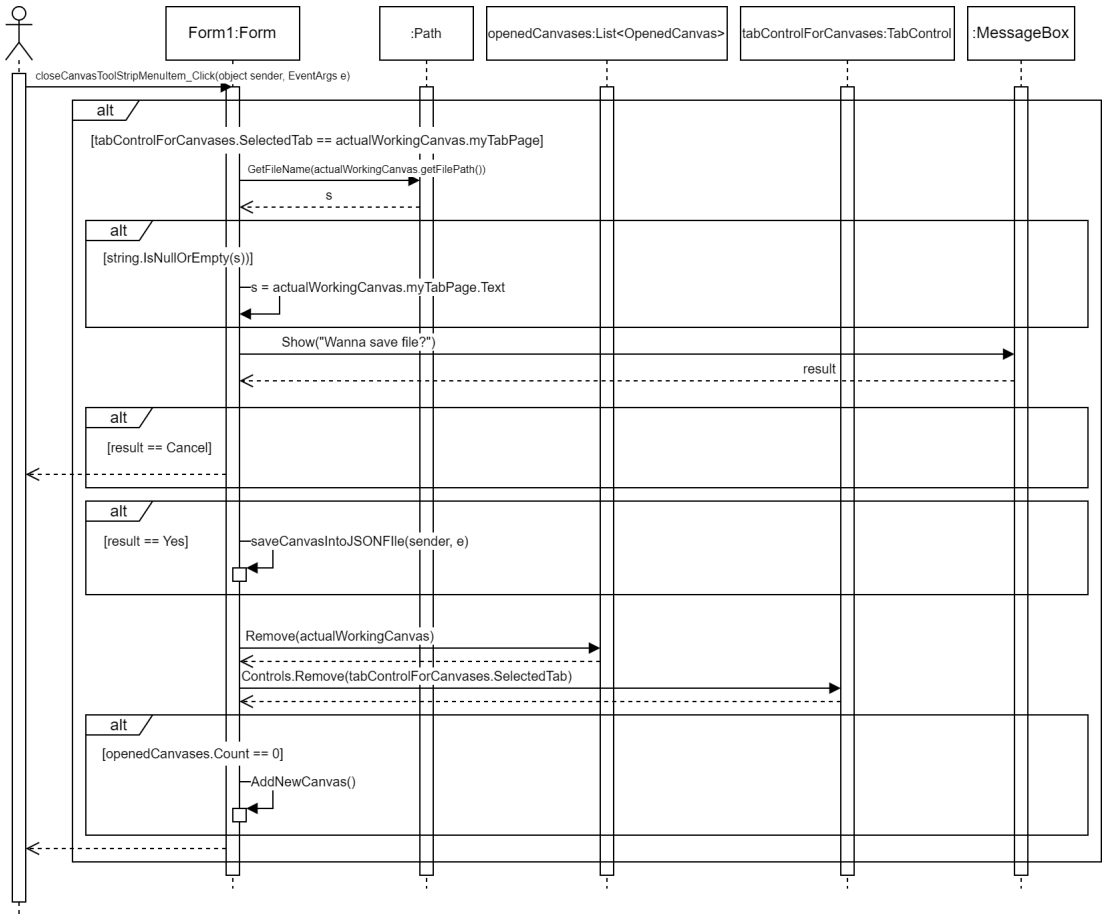
Obr. A.8: Sekvenčný diagram po stlačení tlačidla na UML modelovanie



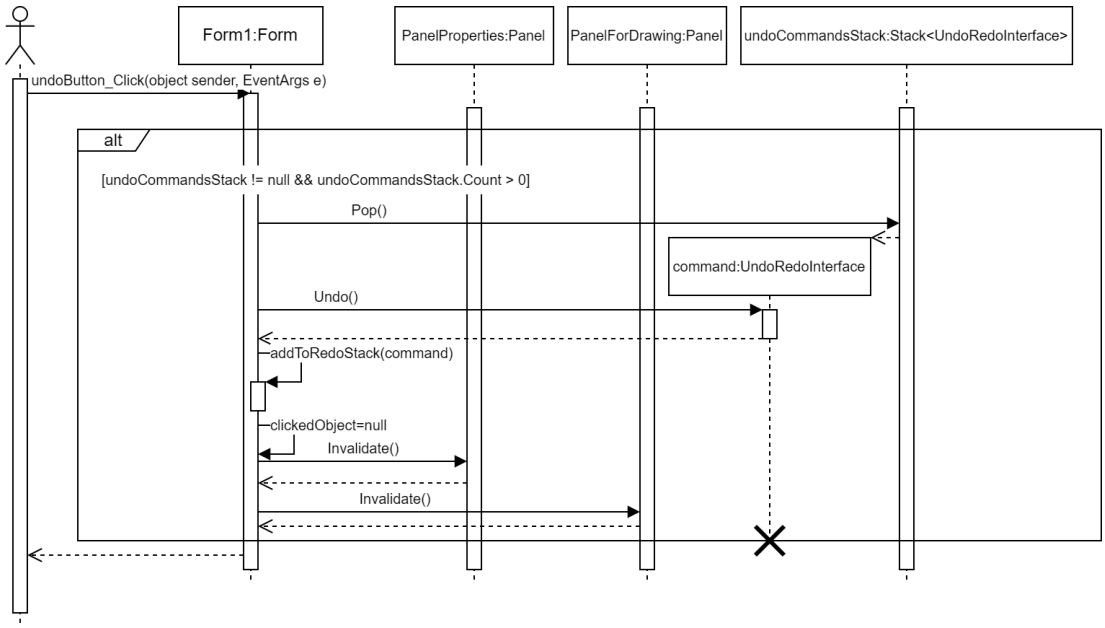
Obr. A.9: Sekvenčný diagram po stlačení tlačidla na CAT modelovanie



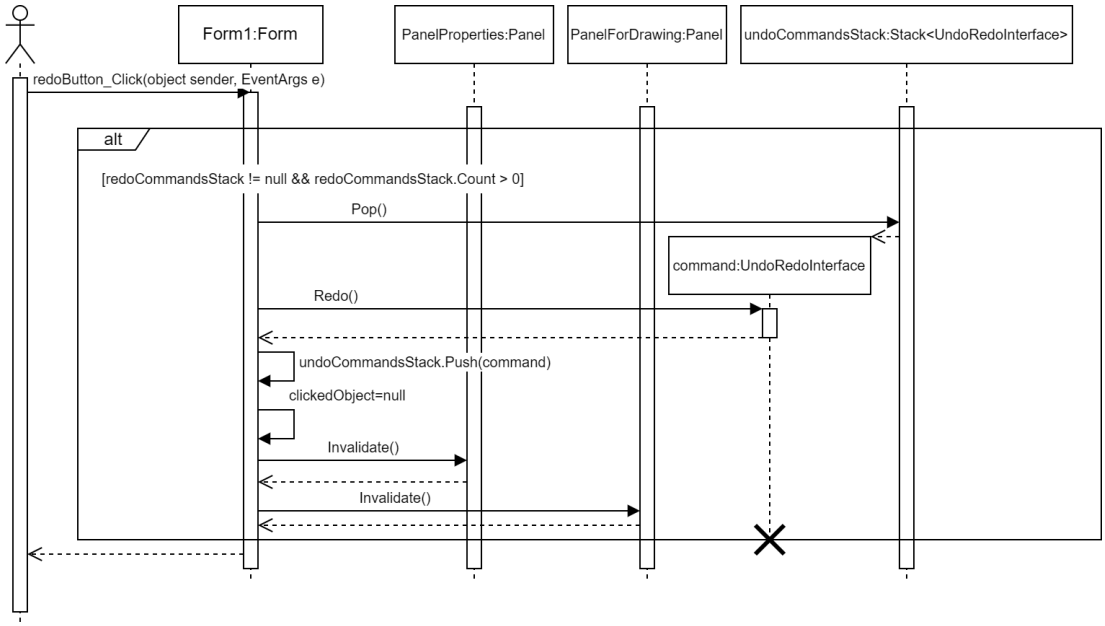
Obr. A.10: Sekvenčný diagram po stlačení tlačidla Clear Canvas



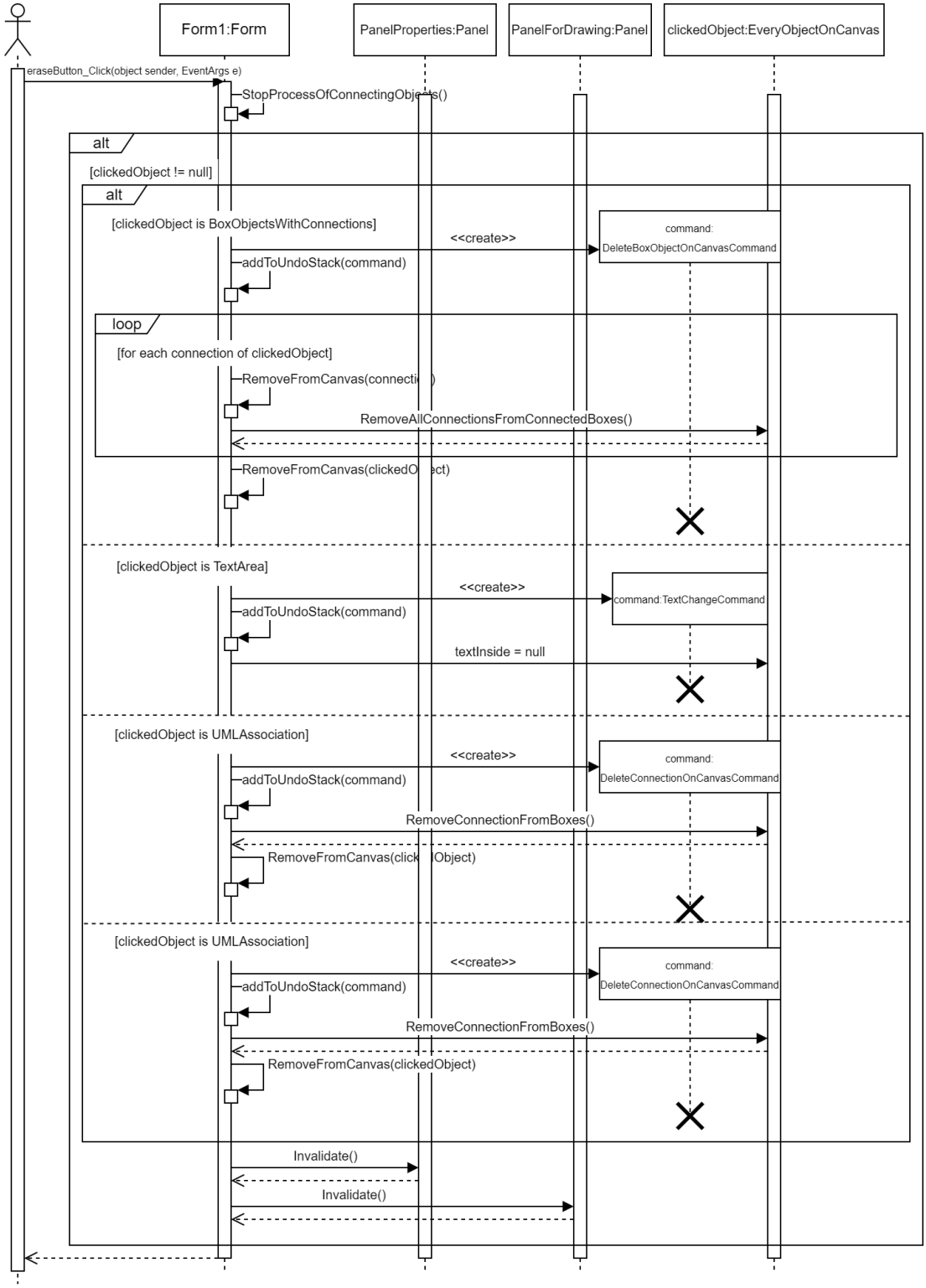
Obr. A.11: Sekvenčný diagram po stlačení tlačidla Close Canvas



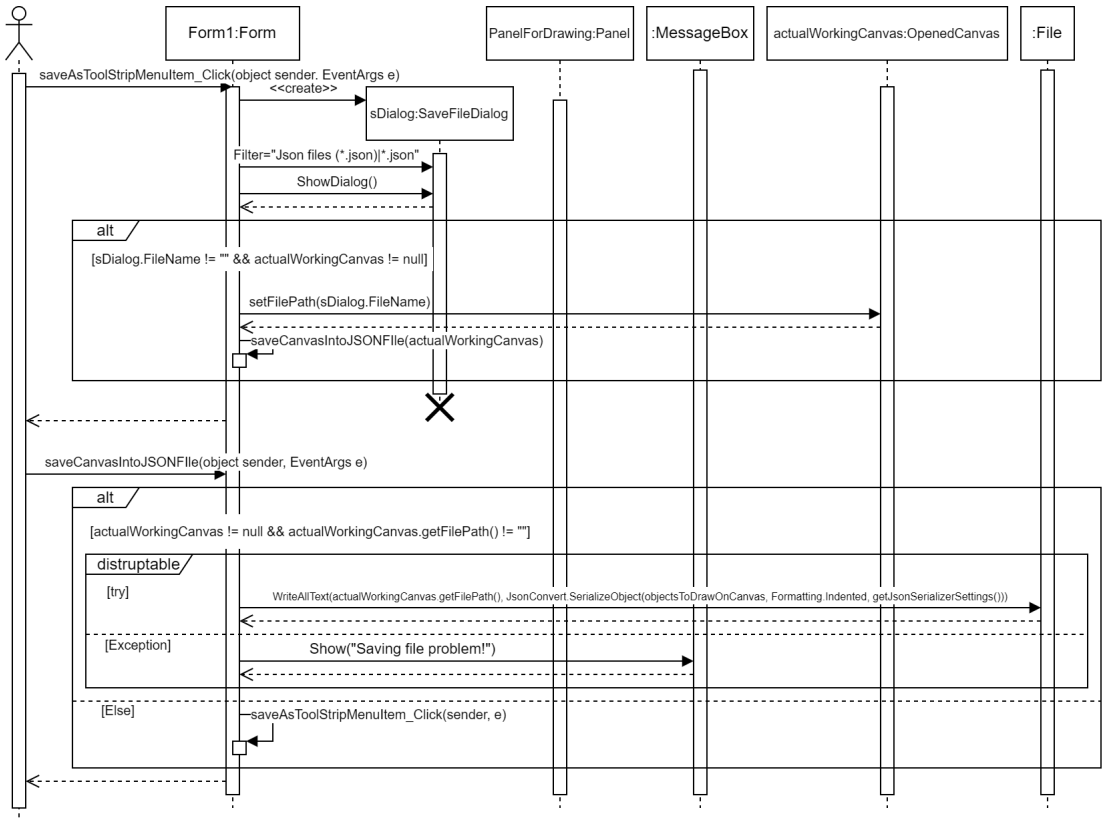
Obr. A.12: Sekvenčný diagram po stlačení tlačidla Undo



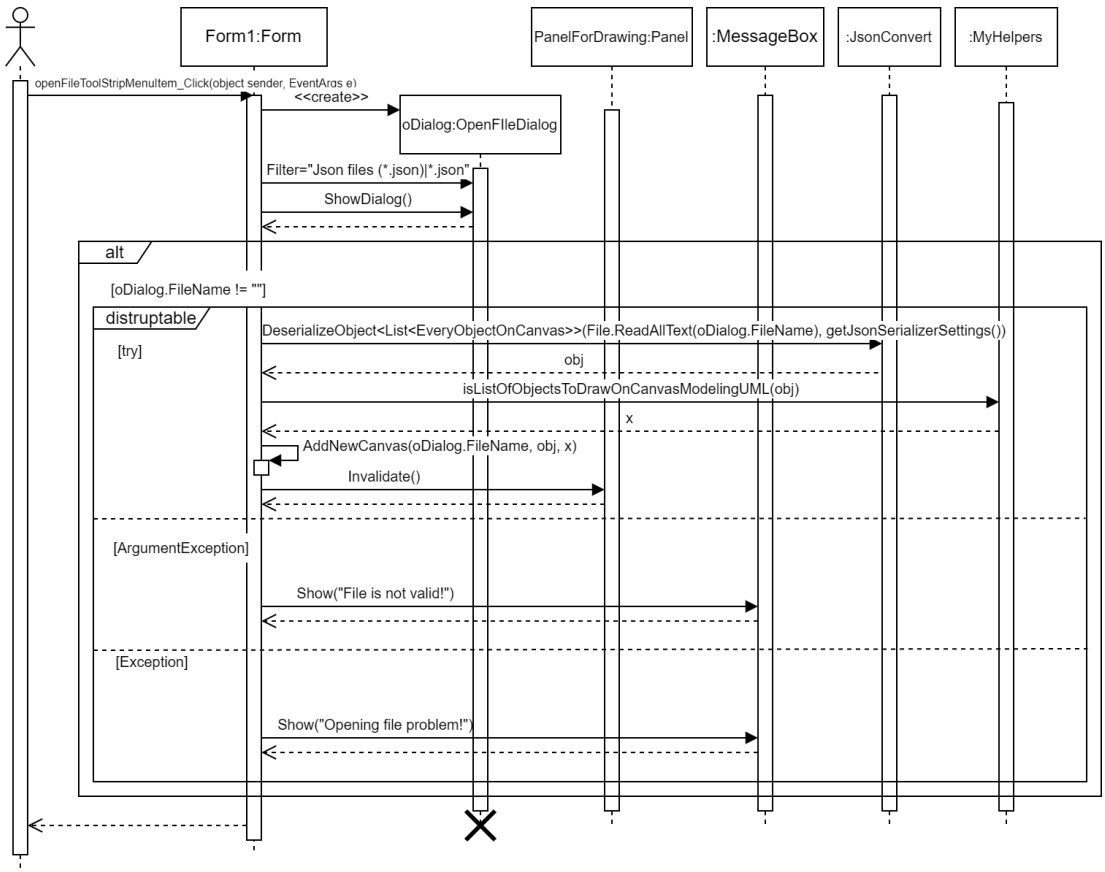
Obr. A.13: Sekvenčný diagram po stlačení tlačidla Redo



Obr. A.14: Sekvenčný diagram po stlačení tlačidla Erase



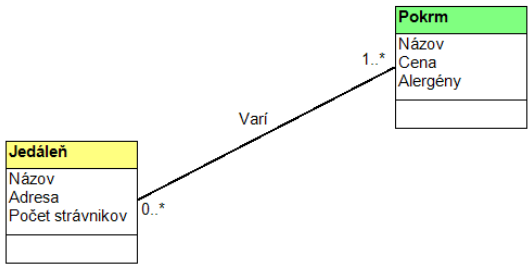
Obr. A.15: Sekvenčný diagram po stlačení tlačidla Save as... a stlačení kláves ctrl+S



Obr. A.16: Sekvenčný diagram po stlačení tlačidla Open file

A.6 Ukážka vygenerovaného JSON súboru

Nižšie zobrazený príklad ukazuje, ako môže vyzerať vygenerovaný súbor vo formáte JSON, na základe príkladu na obrázku A.17.



Obr. A.17: Príklad použitý na ukážku súboru vo formáte JSON


```
{
  "$id": "1",
  "$values": [
    {
      "TypeDiscriminator": "umlClass",
      "$id": "2",
      "nameOfClass": {
        "TypeDiscriminator": "umlClassComponent",
        "$id": "3",
        "childOfClass": {
          "$ref": "2"
        },
        "textInside": "Pokrm",
        "x": 0,
        "y": 0,
        "height": 25,
        "backgroundColor": "128, 255, 128",
        "width": 120,
        "lineColor": "Black",
        "boldText": true,
        "italicText": false
      },
      "attributesOfClass": {
        "TypeDiscriminator": "umlClassComponent",
        "$id": "4",
        "childOfClass": {
          "$ref": "2"
        },
        "textInside": "Názov\r\nCena\r\nAlergény",
        "x": 0,
        "y": 25,
        "height": 61,
        "backgroundColor": "White",
        "width": 120,
        "lineColor": "Black",
        "boldText": false,
        "italicText": false
      },
      "methodsOfClass": {
        "TypeDiscriminator": "umlClassComponent",
        "$id": "5",
        "childOfClass": {
          "$ref": "2"
        },
        "textInside": "",
        "x": 0,
        "y": 86,
        "height": 25,
```

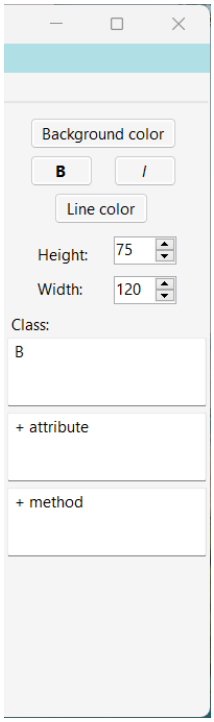
```
      "backgroundColor": "White",
      "width": 120,
      "lineColor": "Black",
      "boldText": false,
      "italicText": false
    },
    "pointsForConnection": {
      "$id": "6",
      "$values": [
        {
          "$id": "7",
          "x": 475,
          "y": 117,
          "width": 7,
          "onObject": {
            "$ref": "2"
          },
          "fillColor": "OrangeRed",
          "correctionX": 7,
          "correctionY": 5,
          "position": 0
        },
        {
          "$id": "8",
          "x": 542,
          "y": 172,
          "width": 7,
          "onObject": {
            "$ref": "2"
          },
          "fillColor": "OrangeRed",
          "correctionX": 0,
          "correctionY": 5,
          "position": 1
        },
        {
          "$id": "9",
          "x": 475,
          "y": 235,
          "width": 7,
          "onObject": {
            "$ref": "2"
          },
          "fillColor": "OrangeRed",
          "correctionX": 7,
          "correctionY": 2,
          "position": 2
        },
        {
          "$id": "10",
          "x": 415,
          "y": 172,
          "width": 7,
          "onObject": {
            "$ref": "2"
          },
          "fillColor": "OrangeRed",
          "correctionX": 5,
          "correctionY": 7,
          "position": 3
        }
      ]
    },
    "connectionsOfBox": {
      "$id": "11",
      "$values": [
        {
          "TypeDiscriminator": "umlAssociation",
          "$id": "12",
          "startMultiplicity": {
            "TypeDiscriminator": "textArea",
            "$id": "13",
            "textInside": "0..*",
```

```
      "x": 185,
      "y": 301,
      "height": 20,
      "backgroundColor": "Transparent",
      "width": 35,
      "lineColor": "Transparent",
      "boldText": false,
      "italicText": false
    },
    "endMultiplicity": {
      "TypeDiscriminator": "textArea",
      "$id": "14",
      "textInside": "1..*",
      "x": 388,
      "y": 163,
      "height": 20,
      "backgroundColor": "Transparent",
      "width": 35,
      "lineColor": "Transparent",
      "boldText": false,
      "italicText": false
    },
    "name": {
      "TypeDiscriminator": "textArea",
      "$id": "15",
      "textInside": "Varí",
      "x": 275,
      "y": 218,
      "height": 20,
      "backgroundColor": "Transparent",
      "width": 70,
      "lineColor": "Transparent",
      "boldText": false,
      "italicText": false
    },
    "pointForAttributes": {
      "$id": "16",
      "x": 299,
      "y": 234,
      "width": 7,
      "onObject": {
        "$ref": "12"
      },
      "fillColor": "OrangeRed",
      "correctionX": 0,
      "correctionY": 0,
      "position": 4
    },
    "attributeObjectConnection": null,
    "startPoint": {
      "$id": "17",
      "x": 183,
      "y": 296,
      "width": 7,
      "onObject": {
        "TypeDiscriminator": "umlClass",
        "$id": "18",
        "nameOfClass": {
          "TypeDiscriminator": "umlClassComponent",
          "$id": "19",
          "childOfClass": {
            "$ref": "18"
          },
          "textInside": "Jedáleň",
          "x": 0,
          "y": 0,
          "height": 25,
          "backgroundColor": "255, 255, 128",
          "width": 120,
          "lineColor": "Black",
          "boldText": true,
          "italicText": false
        },
```

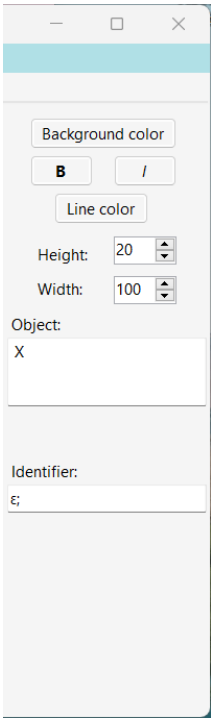
```
"attributesOfClass": {
  "TypeDiscriminator": "umlClassComponent",
  "$id": "20",
  "childOfClass": {
    "$ref": "18"
  },
  "textInside": "Názov\r\nAdresa\r\nPočet strávnikov",
  "x": 0,
  "y": 25,
  "height": 61,
  "backgroundColor": "White",
  "width": 120,
  "lineColor": "Black",
  "boldText": false,
  "italicText": false
},
"methodsOfClass": {
  "TypeDiscriminator": "umlClassComponent",
  "$id": "21",
  "childOfClass": {
    "$ref": "18"
  },
  "textInside": "",
  "x": 0,
  "y": 86,
  "height": 25,
  "backgroundColor": "White",
  "width": 120,
  "lineColor": "Black",
  "boldText": false,
  "italicText": false
},
"pointsForConnection": {
  "$id": "22",
  "$values": [
    {
      "$id": "23",
      "x": 116,
      "y": 241,
      "width": 7,
      "onObject": {
        "$ref": "18"
      },
      "fillColor": "OrangeRed",
      "correctionX": 7,
      "correctionY": 5,
      "position": 0
    },
    {
      "$ref": "17"
    },
    {
      "$id": "24",
      "x": 116,
      "y": 359,
      "width": 7,
      "onObject": {
        "$ref": "18"
      },
      "fillColor": "OrangeRed",
      "correctionX": 7,
      "correctionY": 2,
      "position": 2
    },
    {
      "$id": "25",
      "x": 56,
      "y": 296,
      "width": 7,
      "onObject": {
        "$ref": "18"
      },
      "fillColor": "OrangeRed",
```

```
                    "correctionX": 5,
                    "correctionY": 7,
                    "position": 3
                  }
                ]
              },
              "connectionsOfBox": {
                "$id": "26",
                "$values": [
                  {
                    "$ref": "12"
                  }
                ]
              },
              "x": 63,
              "y": 248,
              "height": 111,
              "backgroundColor": "White",
              "width": 120,
              "lineColor": "Black",
              "boldText": false,
              "italicText": false
            },
            "fillColor": "OrangeRed",
            "correctionX": 0,
            "correctionY": 5,
            "position": 1
          },
          "endPoint": {
            "$ref": "10"
          },
          "width": 2,
          "lineColor": "Black",
          "boldText": false,
          "italicText": false
        }
      ]
    },
    "x": 422,
    "y": 124,
    "height": 111,
    "backgroundColor": "White",
    "width": 120,
    "lineColor": "Black",
    "boldText": false,
    "italicText": false
  },
  {
    "$ref": "18"
  },
  {
    "$ref": "12"
  },
  {
    "$ref": "15"
  },
  {
    "$ref": "13"
  },
  {
    "$ref": "14"
  }
 ]
}
```

A.7 Ukážka panelu na zmenu vlastností pri výbere jednotlivých objektov

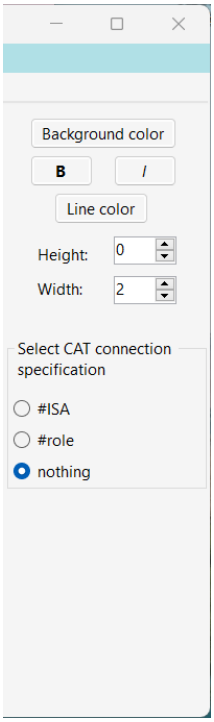


(a) Ukážka panelu, keď je vybraný objekt class



(b) Ukážka panelu, keď je vybraný CAT objekt

(c) Ukážka panelu, keď je vybraný objekt asociácia



(d) Ukážka panelu, keď je vybraný objekt CAT prepojenie

Obr. A.18: Ukážka panelu na zmenu vlastností