Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Teze bakalářská práce

Modelovací software podporující model tříd v UML

Ladislav Böhm

Souhrn

Datové modelování je součástí softwarového inženýrství. Je to technika, kterou je možno analyzovat a popsat data a vazby mezi nimi. Umožňuje tak vytvořit model bez znalosti konkrétního programovacího jazyka či databázové technologie. Jeden z možných přístupů k datovému modelování pochází od Scotta Amblera, odborníka na objektově orientovaný přístup a agilní metody. Zmíněný přístup, který respektuje standard UML diagramu tříd, je v práci implementován do zjednodušené podoby CASE nástroje. Z vytvořeného modelu také umožňuje generovat základ kódu v několika objektově orientovaných jazycích a SQL.

Klíčová slova: UML, CASE, OOP, SQL, C#, SmallTalk, Java, MVVM, diagram tříd, návrhové vzory, agilní metody, unit test

Vlastní text

Cíl práce a metodika

Cílem práce je popsat vývoj objektového přístupu ke tvorbě softwaru, zejména techniky datového modelování. Poskytnout přehled o aktuálně využívaných metodikách řízení vývoje se zaměření na agilní metodiky a ukázat současné možnosti modelování pomocí diagramu tříd ve standardu UML. Praktická část práce má za cíl naprogramovat CASE nástroj podporující zmíněný standard UML a generovat kód do objektových jazyků a SQL.

Program byl vyvinut v prostředí Microsoft Visual Studio 2013, v jazyce C#, na platformě WPF. Použit byl framework .NET 4.5 a knihovna Json.NET. Vývoj byl průběžně konzultován s vedoucím práce a na základě jeho požadavků upravován, zejména v oblasti přístupu k datovému modelování a doporučenému UML profilu, který byl z velké části implementován.

Teoretická část

V úvodní části práce jsou vymezeny a popsány hlavní vlastnosti agilního přístupu k řízení softwarových projektů a jejich porovnání s rigorózními metodami. Hlavním rozdílem v přístupu obou metod je v jejich pohledu na možnost rozdělení vývoje softwaru na menší celky. [Fowler, 2009]

V rigorózních metodách se uplatňuje zejména vodopádový model, jehož fáze jsou vykonávány sekvenčně v obou směrech. Snahou v rigorózních metodách je především anticipovat všechny požadavky již ve fázi příprav a analyzování, aby se předešlo neočekávaným změnám. V agilních metodikách se naopak uplatňuje iterativní přístup a častá komunikace se zákazníkem, který se tím aktivně podílí na vývoji. Upřednostňován je fungující software, komunikace mezi vývojáři a reakce na změny před plněním plánu a tvorbou rozsáhlé dokumentace. [Buchalcevová, 2005]

Významnou částí teoretické části je popis a přednosti užití standardu UML, který je široce rozšířen a přijat v komunitě vývojářů, což výrazně usnadňuje komunikaci a přenositelnost modelů. Užití UML se liší v závislosti na osobě, která model vytváří, smyslu modelu i užité metodice. Neformální rozdělení užití UML, které uvádí Martin Fowler v [Fowler, 2009] je následující:

1. **Náčrtek** je nejpoužívanějším stylem užití UML. Obsahuje pouze nejdůležitější část řešeného problému a pouze v minimálně dostačujícím detailu

2. **Detailní plán** naopak obsahuje rozsáhlou a velmi detailní reprezentaci problému. Jeho tvorba již vyžaduje použití CASE nástroje.

3. **Programovací jazyk** je užití UML, které nastává v případě, že z modelů lze generovat plnohodnotný kód. Zdrojový kód a model se tak stávají tímtéž.

Zbytek teoretické části práce zpracovává problematiku objektově orientovaného vývoje, popisuje implementované návrhové a architektonické vzory, včetně jejich srovnání. V práci je díky využití platformy WPF implementován architektonický vzor MVVM, který umožňuje aplikaci dekomponovat na izolované vrstvy, které lze snadno testovat a spravovat.

Praktická část

Praktická část je uspořádána od obecné struktury a popisu projektu, náhled UI a popis jeho funkcí, až po vybrané detaily z implementací. Program umožňuje vytvářet diagram ze tří typů objektů:

1. **Třída (Class)** obsahuje název, kolekci atributů a triggerů. Je hlavní komponentou modelu.

2. **Pohled (View)** reprezentuje pohled z relačně databázových technologií. Vztahem závislost je možné dosáhnout projekce atributů množiny propojených tříd.

3. **Komentář (Comment)** obsahuje pouze text a datum jeho vytvoření. Slouží pro popis modelu či jeho části.

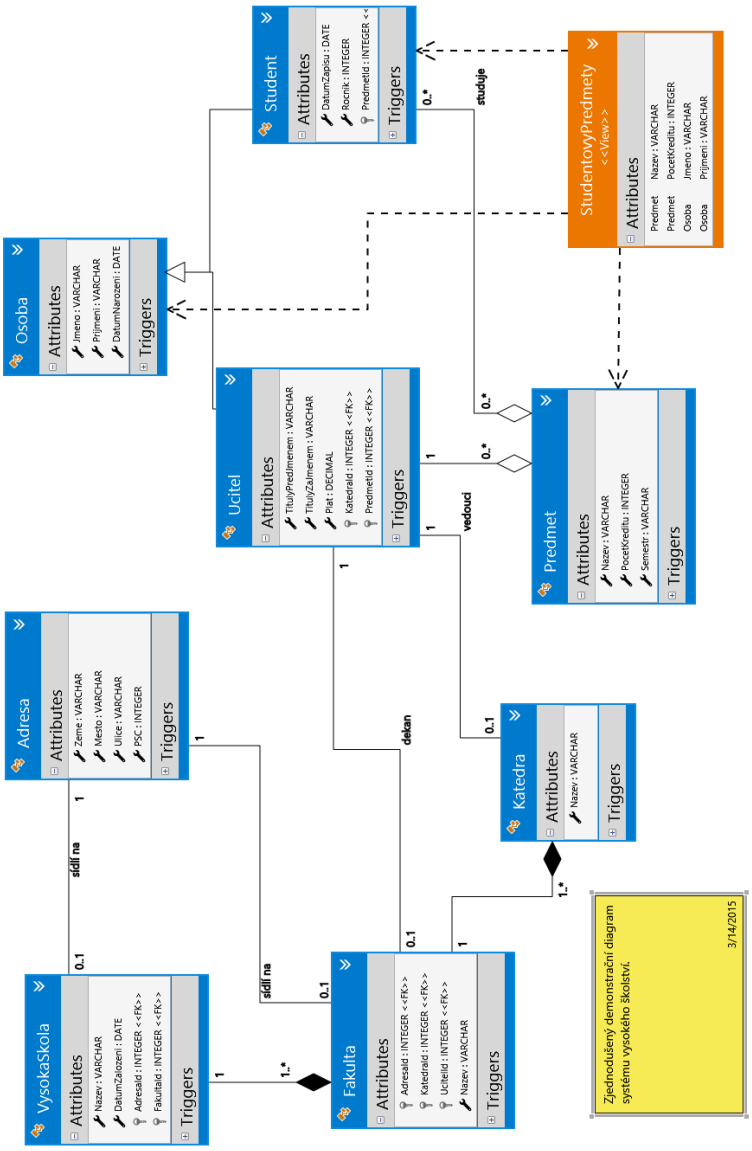
Výše zmíněné objekty je možné spojovat (s výjimkou komentáře) asociacemi a vytvářet tak komplexní systém, který odpovídá diagramu tříd UML. Z vytvořeného diagramu poté program umožní vygenerovat SQL skript, který vytvoří příslušné tabulky a pohledy v databázi. Ze stejného diagramu je možné vygenerovat kód do vybraného objektově orientovaného jazyka, na který je možné data z databáze mapovat.

Mezi další funkce programu patří možnost exportu a importu do formátu JSON, exportu diagramu do libovolného formátu obrázku a nastavení několika vizuálních parametrů diagramu.

Zhodnocení a závěr

Výstup práce v podobě programu autor hodnotí jako zdařilý, nikoli však úplný a konečný. V programu chybí podpora kandidátních klíčů, složených klíčů a rozmisťování vztahů neodpovídá kvalitě jiných komerčních řešení. Aplikace je však do velké míry rozšiřitelná a oba tyto nedostatky lze opravit.

Uplatnění může program nalézt jako podpora při výuce předmětů souvisejících s modelováním či pro tvorbu standardních modelů, zejména ve formě náčrtků. Přínosem je také již zmíněná dekomponovanost programu, která dovoluje využít pouze některé z jeho částí, například generování kódu a distribuovat ho v rámci jiné aplikace.



Obrázek 1: Ukázkový diagram vytvořený ve vyvinutém programu

Literatura

BUCHALCEVOVÁ, A. *Metodiky vývoje a údržby informačních systémů*. Grada, 1. vyd. edition, 2005.

FOWLER, M. *Destilované UML*. Grada, 1. vyd. edition, 2009.